Prädiktive Regelung und Optimierung

von

Prof. Dr.-Ing. Oliver Nelles

Inhaltsverzeichnis

Inhaltsverzeichnis

3. Optimierung: Linear in den Parametern

Prof. Dr.-Ing.
Oliver Nelles

Inhalt Kapitel 3

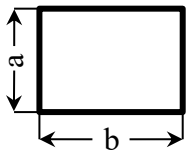
3.1 Einführung

Optimierungsproblem Einführungsbeispiel

Es soll eine Produktionshalle gebaut werden. Es wird eine Grundfläche von 250 m^2 benötigt, um alle Fertigungsanlagen unterzubringen. Die Halle soll dabei eine rechteckige Grundfläche besitzen. Um die Kosten zu senken wird eine Hallenform gesucht, die möglichst kurze Außenwänden besitzt.

Welche Hallen sind möglich ?

- Bedingung: Fläche muss 250m^2 betragen



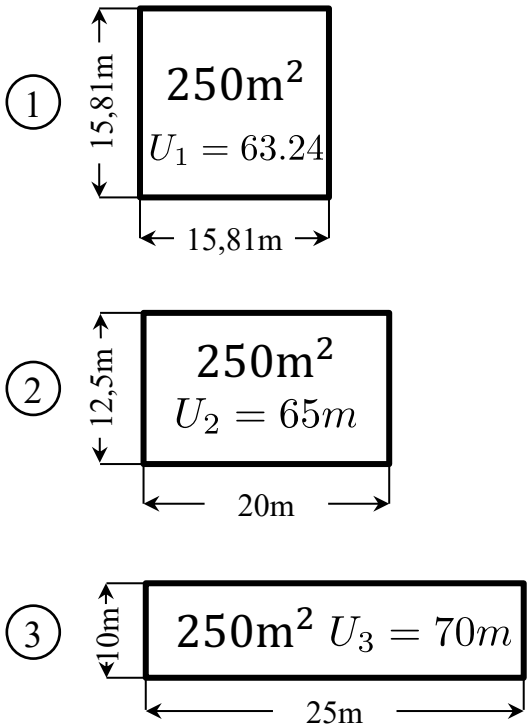
$$\rightarrow a \cdot b = 250m^2$$

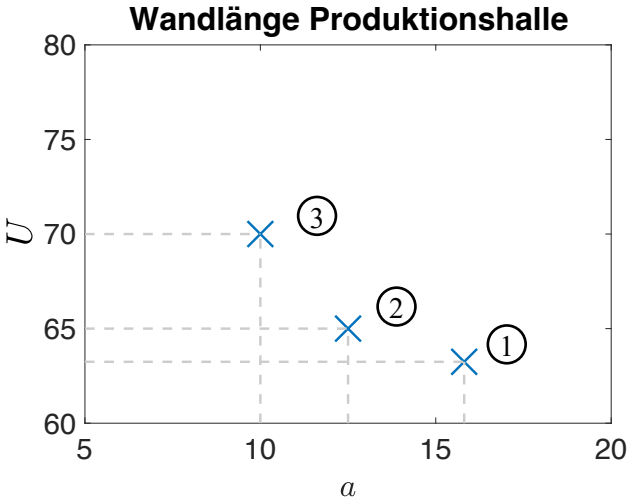
- Durch Kopplung sind Parameter nicht unabhängig voneinander wählbar

$$a = \frac{250m^2}{b} \quad \text{oder} \quad b = \frac{250m^2}{a}$$

→ Nur a <u>oder</u> b muss bestimmt werden

Beispiel Hallen



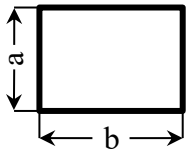


3.1 Einführung

Optimierungsproblem Einführungsbeispiel

Beschreibung als math. Funktion?

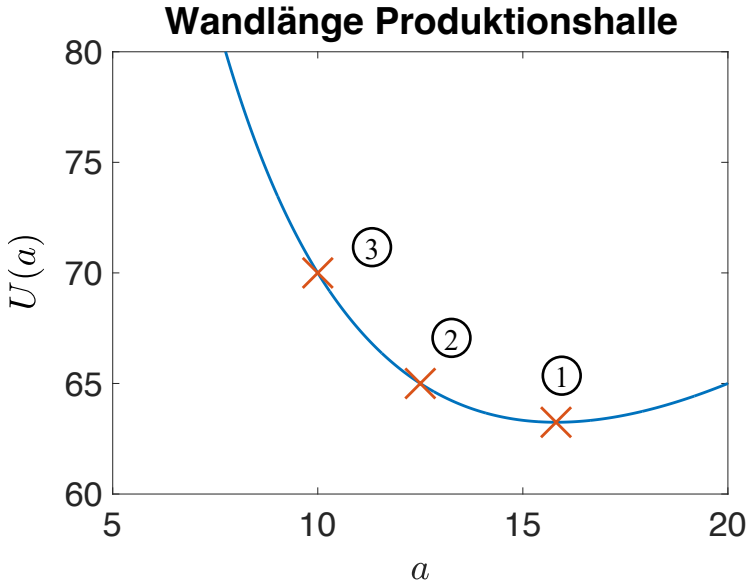
- Bedingung: Fläche muss 250m^2 betragen:



$$\rightarrow a \cdot b = 250m^2$$
$$b = 250m^2/a$$

- Berechnung der Wandlänge:

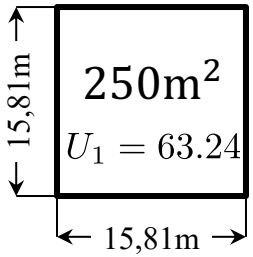
$$U(a) = 2 \cdot a + 2 \cdot b$$
$$= 2 \cdot a + 2 \cdot 250m^2/a$$



Wie lässt sich das Optimum finden?

Für das Minimum einer Funktion gilt, dass die Ableitung der Funktion zu Null wird.

$$\frac{\partial U(a)}{\partial a} = 2 - \frac{2 \cdot 250m^2}{a^2} \stackrel{!}{=} 0$$
$$\rightarrow a = \sqrt{\frac{2 \cdot 250m^2}{2}} = 15.81m$$



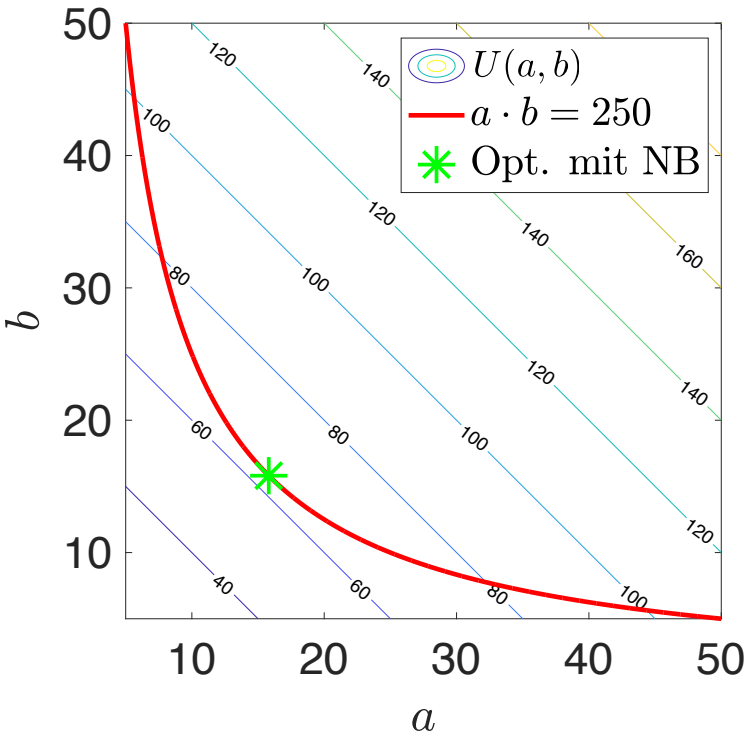
3.1 Einführung

Optimierungsproblem Einführungsbeispiel

Neben dem Ersetzen des Parameters b durch $b = 250m^2/a$ lässt sich das Problem auch als 2-dimensionales Optimierungsproblem mit einer Nebenbedingung formulieren.

$$U(a,b) = 2 \cdot a + 2 \cdot b$$
$$\text{s.t.} \quad a \cdot b = 250m^2$$

Bei der Nebenbedingung handelt es sich um eine Gleichungsnebenbedingung. Das bedeutet, dass nur Punkte auf der roten Linie für das Optimierungsproblem in Frage kommen.



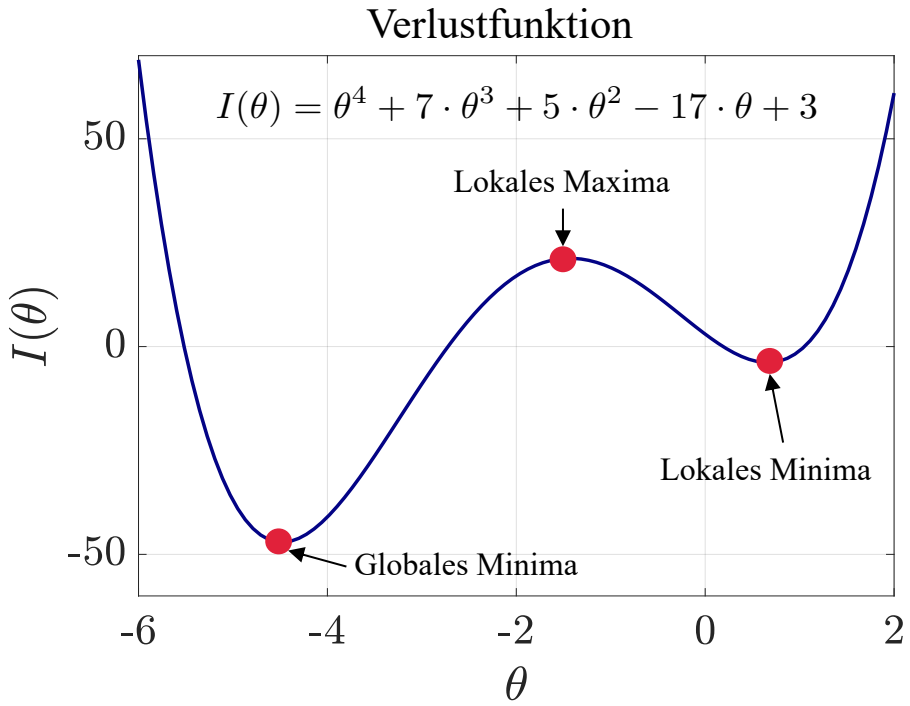
3.1 Einführung

Optimierungsproblem

Ein Optimierungsproblem beschreibt die Aufgabenstellung die beste Lösung für eine definierte Problemstellung zu finden. Typischerweise wird die Problemstellung dabei in Form einer mathematischen Funktion ausgedrückt. Das Ziel ist es ein Minimum oder Maximum dieser Funktion zu bestimmen. Dabei wird zwischen lokalen, globalen und Randoptima unterschieden.

Standartform Minimierung

$$\min_{\underline{\theta}} \boxed{I(\underline{\theta})}$$ Verlustfunktion



3.1 Einführung

Optimalitätsbedingungen ohne Nebenbedingungen

Problem

$$\min_{\underline{\theta} \in \mathbb{R}^n} I(\underline{\theta})$$

- Annahme: $I(\underline{\theta})$ lässt sich zwei mal nach $\underline{\theta}$ differenzieren

  → $\nabla_{\underline{\theta}} I(\underline{\theta})$ wird der Gradient von $I(\underline{\theta})$ genannt

  → $\nabla^2_{\underline{\theta}} I(\underline{\theta})$ wird die Hessematrix von $I(\underline{\theta})$ genannt

Notwendige Bedingung :

- Gradient der Funktion wird zu Null

  1st order: $\nabla_{\underline{\theta}} I(\underline{\theta}^*) = 0$

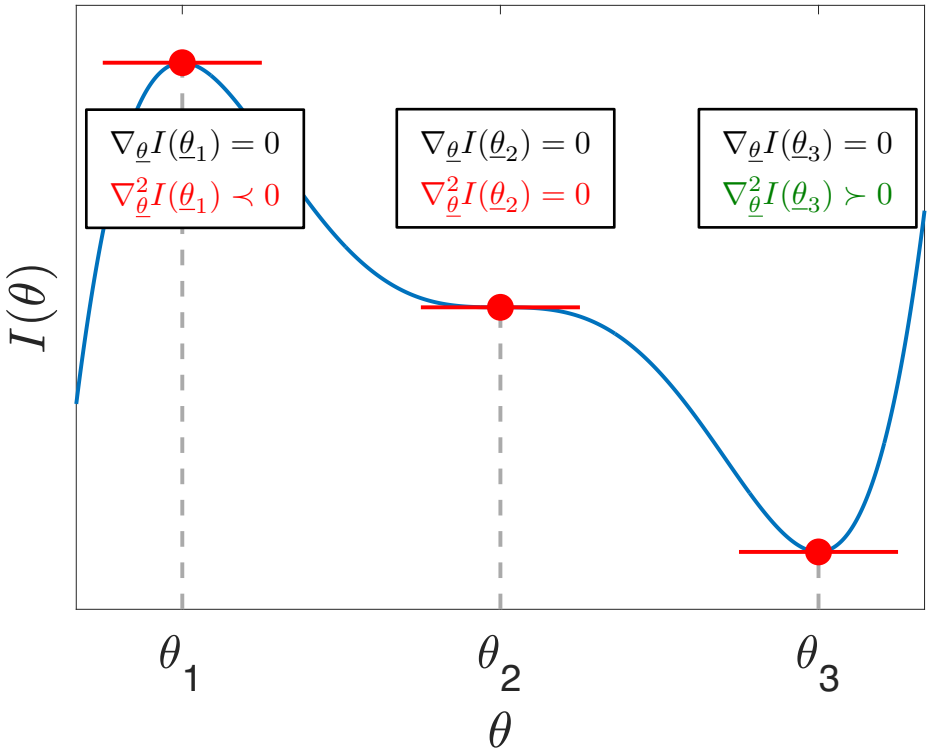
- Bei konvexen Problem auch hinreichende Bedingung

Hinreichende Bedingung:

- Hessematrix ist positiv definit

  2nd order: $\nabla^2_{\underline{\theta}} I(\underline{\theta}^*) \succ 0$

- D. h. die Funktion ist lokal um das Optimum $\underline{\theta}^*$ konvex!



3.1 Einführung

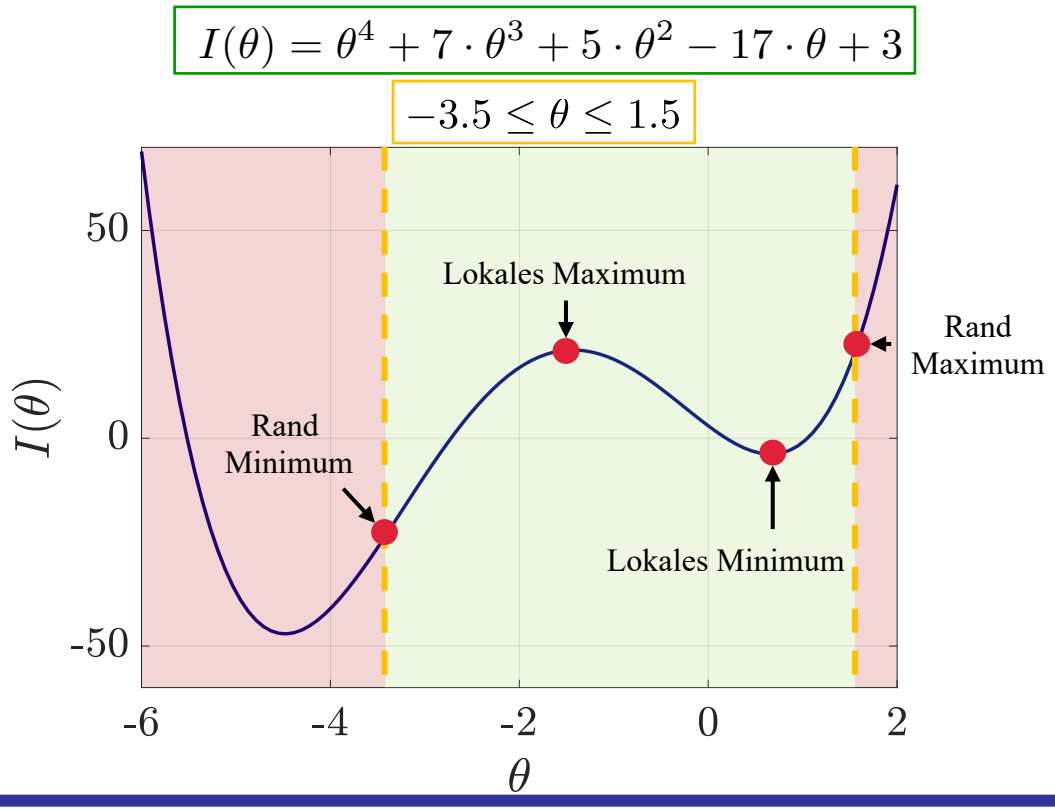
Optimierungsproblem mit Nebenbedingungen (NB)

Gibt es einen eingeschränkten Bereich in dem das Optimum gesucht werden soll, können zusätzlich Nebenbedingungen eingeführt werden. Die Optimalitätskriterien sind nicht mehr ausschließlich für die Bestimmung des Optimums verantwortlich. Die Nebenbedingungen können den gültigen Bereich so einschränken, dass das globale Optimum nicht mehr im gültigen Bereich liegt. Dadurch können zusätzliche Randoptima entstehen. Die mathematische Beschreibung wird um die NB ergänzt.

Standardform Minimierung

$$\min_{\underline{\theta}} \boxed{I(\underline{\theta})} \quad \text{Verlustfunktion}$$

$$\text{s.t.} \quad \boxed{\text{NB}} \quad \text{Nebenbedingungen}$$



3.1 Einführung

Terminologie (uneinheitlich)
Gleichungsnebenbedingungen oder Gleichheitsnebenbedingungen
Ungleichungsnebenbedingungen oder Ungleichheitsnebenbedingungen

Nebenbedingungen

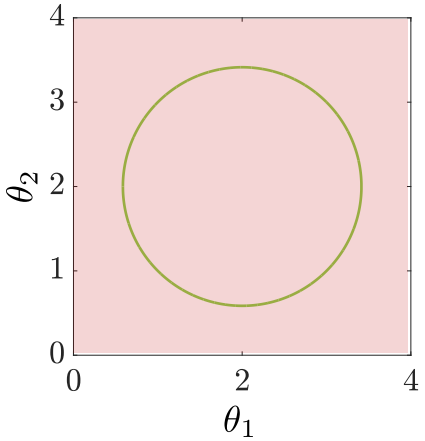
Häufig unterliegt ein Optimierungsproblem Nebenbedingungen. Zum Beispiel darf eine Stellgröße einen maximalen Werte nicht überschreiten oder die gesamt Anlagesumme ist begrenzt. Nebenbedingungen lassen sich in zwei Klassen unterteilen.

Gleichungsnebenbedingungen (=)

- Die Bedingung muss immer erfüllt sein
- Alle gültigen Punkte liegen auf dieser NB

Praktisches Beispiel: Verpackungsvolumen

$$(\theta_1 - 2)^2 + (\theta_2 - 2)^2 = 2$$

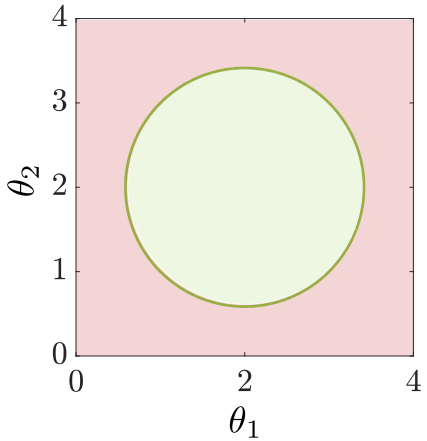


Ungleichungsnebenbedingungen ($\geq$)

- Die Bedingung darf nicht überschritten werden
- Gültige Punkte liegen auch auf dieser NB

Praktisches Beispiel: Verfügbares Tankvolumen

$$(\theta_1 - 2)^2 + (\theta_2 - 2)^2 \leq 2$$



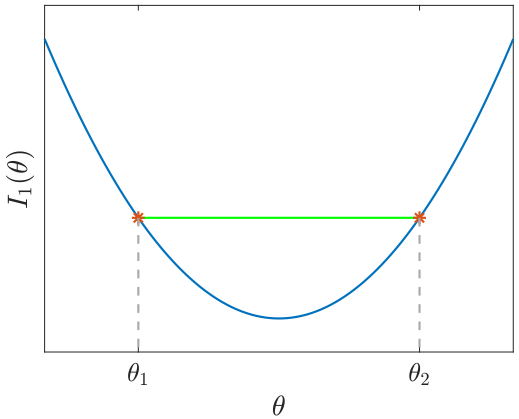
3.1 Einführung

Konvexität

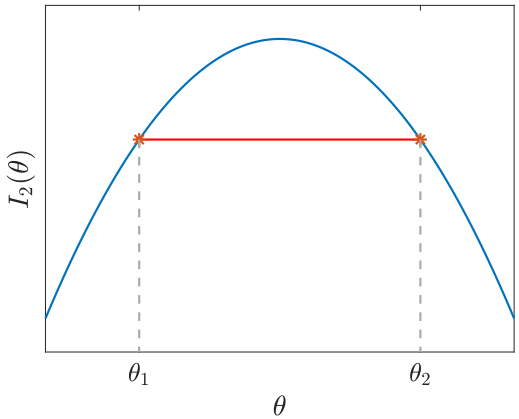
Konvexität ist eine zentrale Eigenschaft von Optimierungsproblemen. Wenn ein Problem konvex ist besitzt es nur ein Minimum. Demnach ist das lokale Minimum gleichzeitig das globale Minimum. Konvexe Probleme lassen sich gut lösen und es existieren verschiedene Lösungsmethoden. Allgemein ist eine Funktion $I(\theta)$ konvex wenn für alle $\lambda \in [0, 1]$ folgendes gilt:

$$I(\lambda\theta_1 + (1-\lambda)\theta_2) \leq \lambda I(\theta_1) + (1-\lambda) I(\theta_2)$$

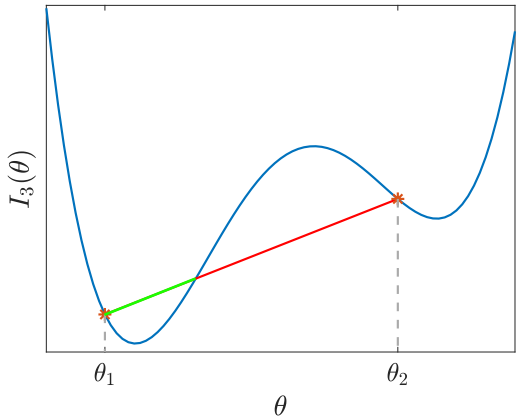
Ein konvexes Problem ist dadurch charakterisiert, dass die Verbindung zwischen zwei beliebig wählbaren Punkten dieser Funktion ausschließlich oberhalb der Funktion selbst verläuft.



konvex



nicht konvex (konkave)



nicht konvex

3.1 Einführung

Optimierungsproblem

Optimierungsprobleme können je nach ihren Eigenschaften in unterschiedliche Klassen eingeordnet werden. Je nach Klasse eignen sich unterschiedliche Lösungsverfahren.

	Problemklasse	Verlustfunktion	Gl. NB	Ungl. NB	Mgl. Solver
Lin	Linear Programm (LP)	Lin	Lin	Lin	Simplex
Quad	Quadratic Programm (QP)	Quad	Lin	Lin	Active Set / Interior Point → KKT
	Least Squares (LS)	Quad	-	-	LS / RLS / LMS
Nonlin	Nonlinear Programm	Nonlinear	?	?	Siehe Kap. 5
		Lin / quad	Nonlinear		

Kapitel 3

Kapitel 5

3.2 Lineare Probleme

Problembeschreibung

Ein lineares Problem (LP) besteht aus einer linearen Verlustfunktion und linearen Nebenbedingungen. Das Ziel ist es die Verlustfunktion unter Einhaltung der Nebenbedingungen zu minimieren.

$$\min_{\underline{\theta}} \quad \boxed{\underline{g}^T \underline{\theta} + c} \qquad \text{Verlustfunktion}$$

$$\text{s.t.} \quad \boxed{\underline{A}\,\underline{\theta} - \underline{b} = 0} \qquad \text{Gleichungsnebenbedingungen}$$

$$\boxed{\underline{G}\,\underline{\theta} - \underline{h} \leq 0} \qquad \text{Ungleichungsnebenbedingungen}$$

$\underline{\theta}$ ist der Parametervektor der optimal gewählt werden soll und $\underline{g}$ die Linearkombination in der die Parameter zusammengerechnet werden.

$\underline{A}$ und $\underline{G}$ beschreiben die Linearkombinationen der Parameter in den Nebenbedingungen. $\underline{b}$ und $\underline{h}$ beschreiben die Grenzen die von den Linearkombinationen eingehalten werden müssen.

Für die Lösung eines solchen Problems existiert keine direktes analytisches Lösungsverfahren. Deswegen werden zur Lösung verschiedene Suchverfahren eingesetzt.

3.2 Lineare Probleme

Problem

$$\begin{aligned} \min_{\underline{\theta}} \quad & -2\theta_1 - 5\theta_2 \\ \text{s.t.} \quad & 3\theta_1 + 2\theta_2 \leq 12 \\ & \theta_2 \leq 3 \end{aligned}$$

Beispiel

Im Beispiel wird ein lineare Verlustfunktion mit Ungleichungsnebenbedingungen betrachtet. Das Minimum einer solchen Funktion liegt immer auf einer Ecke des zulässigen Raums, der durch die Nebenbedingungen begrenzt wird.

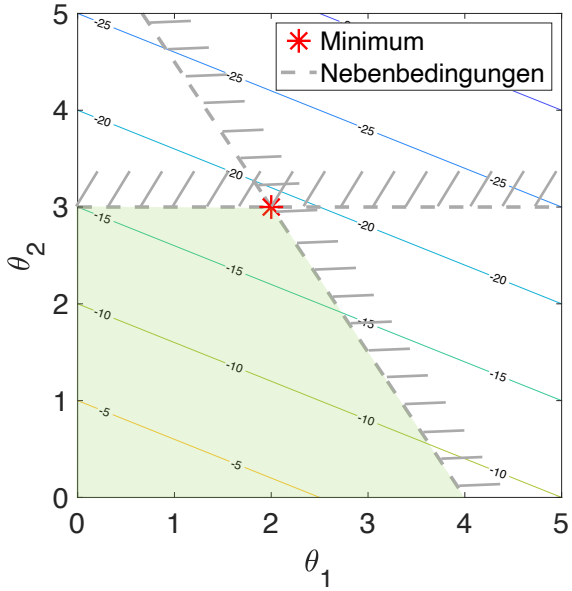
Das Problem lässt sich in die Standartform bringen:

Verlustfunktion

$$\underline{g}^T \underline{\theta} = \begin{bmatrix} g_1 & g_2 \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} = \begin{bmatrix} -2 & -5 \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} = -2\theta_1 - 5\theta_2$$

Nebenbedingungen

$$\underline{G}\,\underline{\theta} - \underline{h} = \begin{bmatrix} 3 & 2 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} - \begin{bmatrix} 12 \\ 3 \end{bmatrix} = \begin{bmatrix} 3\theta_1 + 2\theta_2 - 12 \\ \theta_2 - 3 \end{bmatrix} \leq \underline{0}$$



Lösungsverfahren: z.B. Simplex Algorithmus (Operations Research)

3.2 Lineare Probleme

Wirtschaftliche Bedeutung von LPs

- LPs werden z.B. gebraucht zur **Optimierung** von **Fahrplänen** oder ähnlichen logischen Aufgaben. Sie werden auf dem Gebiet des **Operations Research** behandelt.
- Z.B. bei **Airlines** oder dem Fahrplan der **Bahn** ergeben sich sich LPs mit
    1. Milliarden von Variablen
    2. Milliarden von Nebenbedingungen
    3. enormen wirtschaftlichen Implikationen bzgl. Zeit und Treibstoffkosten u.ä.
- 1983 schaffte ***Narendra K. Karmarker*** einen Durchbruch mit **Interior-Point-Methoden**, die den bis dahin üblichen Simplex-Algorithmus um Größenordnungen outperformen.
- Damit war es erstmal möglich, riesige LPs überhaupt (in der zur Verfügung stehenden Zeit) lösen zu können.
- Wegen seiner wirtschaftlichen Bedeutung schaffte es dieser Durchbruch ins Wall Street Journal, New York Times, Time Magazine.
- Quelle: https://vanderbei.princeton.edu/307/lectures/lec17_show.pdf

Breakthrough in Problem Solving

By JAMES GLEICK

A 28-year-old mathematician at A.T.&T. Bell Laboratories has made a startling theoretical breakthrough in the solving of systems of equations that often grow too vast and complex for the most powerful computers.

The discovery, which is to be formally published next month, is already circulating rapidly through the mathematical world. It has also set off a deluge of inquiries from brokerage houses, oil companies and airlines, industries with millions of dollars at stake in problems known as linear programming.

Faster Solutions Seen

These problems are fiendishly complicated systems, often with thousands of variables. They arise in a variety of commercial and government applications, ranging from allocating time on a communications satellite to routing millions of telephone calls over long distances, or whenever a limited, expensive resource must be spread most efficiently among competing users. And investment companies use them in creating portfolios with the best mix of stocks and bonds.

The Bell Labs mathematician, Dr. Narendra Karmarkar, has devised a radically new procedure that may speed the routine handling of such problems by businesses and Government agencies and also make it possible to tackle problems that are now far out of reach.

"This is a path-breaking result," said Dr, Ronald L. Graham, director of mathematical sciences for Bell Labs in Murray Hill, N.J.

"Science has its moments of great progress, and this may well be one of them."

Because problems in linear programming can have billions or more possible answers, even high-speed computers cannot check every one. So computers must use a special procedure, an algorithm, to examine as few answers as possible before finding the best one — typically the one that minimizes cost or maximizes efficiency.

A procedure devised in 1947, the simplex method, is now used for such problems,

Continued on Page A19, Column 1

THE NEW YORK TIMES, November 19, 1984

Karmarkar at Bell Labs: an equation to find a new way through the maze

Folding the Perfect Corner

A young Bell scientist makes a major math breakthrough

Every day 1,200 American Airlines jets crisscross the U.S., Mexico, Canada and the Caribbean, stopping in 110 cities and bearing over 80,000 passengers. More than 4,000 pilots, copilots, flight personnel, maintenance workers and baggage carriers are shuffled among the flights; a total of 3.6 million gal. of high-octane fuel is burned. Nuts, bolts, altimeters, landing gears and the like must be checked at each destination. And while performing these scheduling gymnastics, the company must keep a close eye on costs, projected revenue and profits.

Like American Airlines, thousands of companies must routinely untangle the myriad variables that complicate the efficient distribution of their resources. Solving such monstrous problems requires the use of an abstruse branch of mathematics known as linear programming. It is the kind of math that has frustrated theoreticians for years, and even the fastest and most powerful computers have had great difficulty juggling the bits and pieces of data. Now Narendra Karmarkar, a 28-year-old Indian-born mathematician at Bell Laboratories in Murray Hill, N.J., after only a years' work has cracked the puzzle of linear programming by devising a new algorithm, a step-by-step mathematical formula. He has translated the procedure into a program that should allow computers to track a greater combination of tasks than ever before and in a fraction of the time.

Unlike most advances in theoretical mathematics, Karmarkar's work will have an immediate and major impact on the real world. "Breakthrough is one of the most abused words in science," says Ronald Graham, director of mathematical sciences at Bell Labs. "But this is one situation where it is truly appropriate."

Before the Karmarkar method, linear equations could be solved only in a cumbersome fashion, ironically known as the simplex method, devised by Mathematician George Dantzig in 1947. Problems are conceived of as giant geodesic domes with thousands of sides. Each corner of a facet on the dome

TIME MAGAZINE, December 3, 1984

3.2 Lineare Probleme

Karmarkar Algorithm Proves Its Worth

Less than two years after discovery of a mathematical procedure that Bell Labs said could solve a broad range of complex business problems 50 to 100 times faster than current methods, AT&T is filing for patents covering its use. The Karmarkar algorithm, which drew headlines when discovered by researcher Narendra Karmarkar, will be applied first to AT&T's long-distance network.

Thus far, Bell Labs has verified the procedure's capabilities in developing plans for new fiber-optic transmission and satellite capacity linking 20 countries bordering the Pacific Ocean. That jointly owned network will be built during the next 10 years. Planning requires a tremendous number of "what if" scenarios involving 43,000 variables describing transmission capacity, location and construction schedules, all juggled amid political considerations of each connected country.

The Karmarkar algorithm was able to solve the Pacific Basin problem in four minutes, against 80 minutes by the method previously used, says Neil Dinn, head of Bell Labs' international transmission planning department. The speedier solutions will enable international committees to agree on network designs at one meeting instead of many meetings stretched out over months.

AT&T now is using the Karmarkar procedure to plan construction for its domestic network, a problem involving 800,000 variables. In addition, the procedure may be written into software controlling routing of domestic phone calls, boosting the capacity of AT&T's current network.

THE WALL STREET JOURNAL, July 18, 1986

THE STARTLING DISCOVERY BELL LABS KEPT IN THE SHADOWS

Now its breakthrough mathematical formula could save business millions

It happens all too often in science. An obscure researcher announces a stunning breakthrough and achieves instant fame. But when other scientists try to repeat his results, they fail. Fame quickly turns to notoriety, and eventually the episode is all but forgotten.

That seemed to be the case with Narendra K. Karmarkar, a young scientist at AT&T Bell Laboratories. In late 1984 the 28-year-old researcher astounded not only the scientific community but also the business world. He claimed he had cracked one of the thorniest aspects of computer-aided problem-solving. If so, his feat would have meant an instant windfall for many big companies. It could also have pointed to better software for small companies that use computers to help manage their business.

Karmarkar said he had discovered a quick way to solve problems so hideously complicated that they often defy even the most powerful supercomputers. Such problems bedevil a broad range of business activities, from assessing risk factors in stock portfolios to drawing up production schedules in factories. Just about any company that distributes products through more than a handful of warehouses bumps into such problems when calculating the cheapest routes for getting goods to customers. Even when the problems aren't terribly complex, solving them can chew up so much computer time that the answer is useless before it's found.

HEAD START. To most mathematicians, Karmarkar's precocious feat was hard to swallow. Because such questions are so common, a special branch of mathematics called linear programming (LP) has evolved, and most scientists thought that was as far as they could go. Sure enough, when other researchers independently tried to test Karmarkar's process, their results were disappointing. At scientific conferences skeptics attacked the algorithm's validity as well as Karmarkar's veracity.

KARMARKAR: SKEPTICS ATTACKED HIS PRECOCIOUS FEAT

But this story may end with a different twist. Other scientists weren't able to duplicate Karmarkar's work, it turns out, because his employer wanted it that way. Vital details about how best to translate the lgorithm, whose mathematical notations run on for about 20 printed pages, into digital computer code were withheld to give Bell Labs a head start at developing commercial products. Following the breakup of American Telephone & Telegraph Co. in January, 1984, Bell Labs was no longer prevented from exploiting its research for profit. While the underlying concept could not be patented or copyrighted because it is pure knowledge, any computer programs that AT&T developed to implement the procedure can be protected.

Now, AT&T may soon be selling the first product based on Karmarkar's work—to the U.S. Air Force. It includes a multiprocessor computer from Alliant Computer Systems Corp. and a software version of Karmarkar's algorithm that has been optimized for high-speed parallel processing. The system would be installed at St. Louis' Scott Air Force Base, headquarters of the Military Airlift Command (MAC). Neither party will comment on the deal's cost or where the negotiations stand, but the Air Force's interest is easy to fathom.

JUGGLING ACT. On a typical day thousands of planes ferry cargo and passengers among air fields scattered around the world. To keep those jets flying, MAC

BUSINESS WEEK, September 21, 1987

3.2 Lineare Probleme

Patents

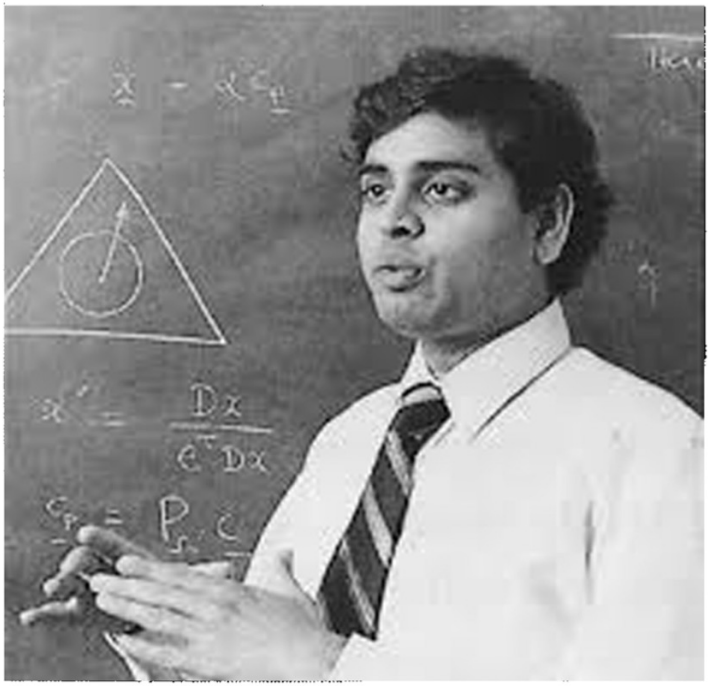
by Stacy V. Jones

A Method to Improve Resource Allocation

Scientists at Bell Laboratories in Murray Hill, N.J., were granted three patents this week for methods of improving the efficiency of allocation of industrial and commercial resources.

The American Telephone and Telegraph Company, the laboratory's sponsor, is using the methods internally to regulate such operations as long-distance services.

Narendra K. Karmarkar of the laboratory staff was granted patent 4,744,028 for methods of allocating telecommunication and other resources. With David A. Bayer and Jeffrey C. Lagarian as co-inventors, he was granted patent 4,744,027 on improvements of the basic method. Patent 4,744,026 went to Robert J. Vanderbei for enhanced procedures.



Narendra K. Karmarkar of the Bell Laboratories staff.

THE NEW YORK TIMES, May 14, 1988

AT&T Markets Problem Solver, Based On Math Whiz's Find, for $8.9 Million

By ROGER LOWENSTEIN
Staff Reporter of THE WALL STREET JOURNAL

NEW YORK—American Telephone & Telegraph Co. has called its math whiz, Narendra Karmarkar, a latter-day Isaac Newton. Now, it will see if he can make the firm some money.

Four years after AT&T announced an "astonishing" discovery by the Indian-born Mr. Karmarkar, it is marketing an $8.9 million problem solver based on his invention.

Dubbed Korbx, the computer-based system is designed to solve major operational problems of both business and government. AT&T predicts "substantial" sales for the product, but outsiders say the price is high and point out that its commercial viability is unproven.

"At $9 million a system, you're going to have a small number of users," says Thomas Magnanti, an operations-research specialist at Massachusetts Institute of Technology. "But for very large-scale problems, it might make the difference."

Korbx uses a unique algorithm, or step-by-step procedure, invented by Mr. Karmarkar, a 32-year old, an AT&T Bell Laboratories mathematician.

"It's designed to solve extremely difficult or previously unsolvable resource-allocation problems—which can involve hundreds of thousands of variables—such as personnel planning, vendor selection, and equipment scheduling," says Aristides Fronistas, president of an AT&T division created to market Korbx.

Potential customers might include an airline trying to determine how to route many planes between numerous cities and an oil company figuring how to feed different grades of crude oil into various refineries and have the best blend of refined products emerge.

AT&T says that fewer than 10 companies, which it won't name, are already using Korbx. It adds that, because of the price, it is targeting only very large companies—mostly in the Fortune 100.

Korbx "won't have a significant bottom-line impact initially" for AT&T, though it might in the long term, says Charles Nichols, an analyst with Bear, Stearns & Co. "They will have to expose it to users and demonstrate" it uses.

AMR Corp.'s American Airlines says it's considering buying AT&T's system. Like other airlines, the Fort Worth, Texas, carrier has the complex task of scheduling pilots, crews and flight attendants on thousands of flights every month.

Thomas M.Cook, head of operations research at American, says, "Every airline has programs that do this. The question is: Can AT&T do it better and faster? The jury is still out."

The U.S. Air Force says it is considering using the system at the Scott Air Force Base in Illinois.

One reason for the uncertainty is that AT&T has, for reasons of commercial secrecy, deliberately kept the specifics of Mr. Karmarkar's algorithm under wraps.

"I don't know the details of their system," says Eugene Bryan, president of Decision Dynamics Inc., a Portland, Ore., consulting firm that specializes in linear programming, a mathematical technique that employs a series of equations using many variables to find the most efficient way of allocating resources.

Mr. Bryan says, though, that if the Karmarkar system works, it would be extremely useful. "For every dollar you spend on optimization," he says, "you usually get them back many-fold."

AT&T has used the system in-house to help design equipment and routes on its Pacific Basin system, which involves 22 countries. It's also being used to plan AT&T's evolving domestic network, a problem involving some 800,000 variables.

THE WALL STREET JOURNAL, August 15, 1988

3.3 Quadratische Probleme

Problembeschreibung

Ein Quadratisches Problem ist die Erweiterung eines linearen Problems um einen quadratischen Anteil in der Verlustfunktion. Nebenbedingung sind weiterhin nur linear möglich.

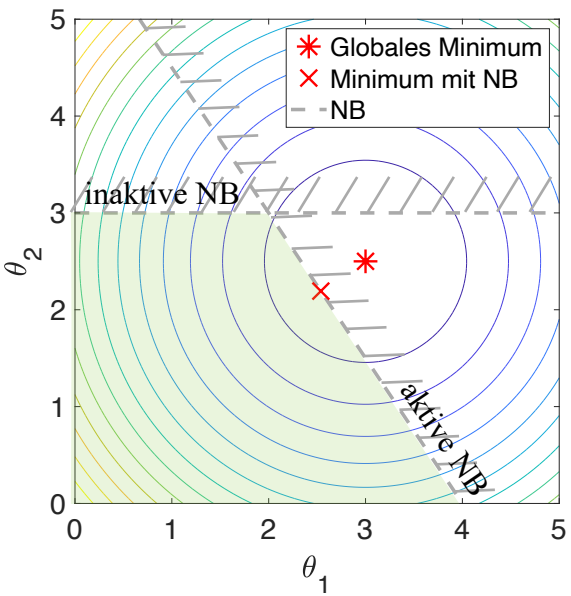
$$\min_{\underline{\theta}} \quad \boxed{\tfrac{1}{2}\underline{\theta}^T \underline{H}\,\underline{\theta} + \underline{g}^T\underline{\theta} + c} \qquad \text{Verlustfunktion}$$

$$\text{s.t.} \quad \boxed{\underline{A}\,\underline{\theta} - \underline{b} = 0} \qquad \text{Gleichungsnebenbedingungen}$$

$$\boxed{\underline{G}\,\underline{\theta} - \underline{h} \leq 0} \qquad \text{Ungleichungsnebenbedingungen}$$

Dieses Problem ist konvex wenn die Matrix $\underline{H}$ positiv definit ist. Es wird als *Quadratic Programm* QP bezeichnet. Der Sonderfall ohne Nebenbedingungen wird als *Least Squares* LS Problem bezeichnet.
Probleme dieser Klasse sind gut verstanden und es existieren verschiedene Lösungsmethoden:

- Gradientenverfahren
- Interior-Point Methoden
- Active-Set Methoden



3.3 Quadratische Probleme

Least Squares

Least Squares (LS, engl. für Methode der kleinsten Quadrate) ist ein Sonderfall eines Quadratischen Problems ohne Nebenbedingungen.

$$\text{konvexes Problem:} \quad \min_{\underline{\theta}} \quad \tfrac{1}{2}\underline{\theta}^T \underline{H}\,\underline{\theta} + \underline{g}^T\underline{\theta} + c$$

Erinnerung: Die Matrix $\underline{H}$ ist positiv definit.

Für dieses Problem gibt es eine geschlossene Lösung:

1st order Optimalitätsbedingung
(notwendig + hinreichend da konvex):

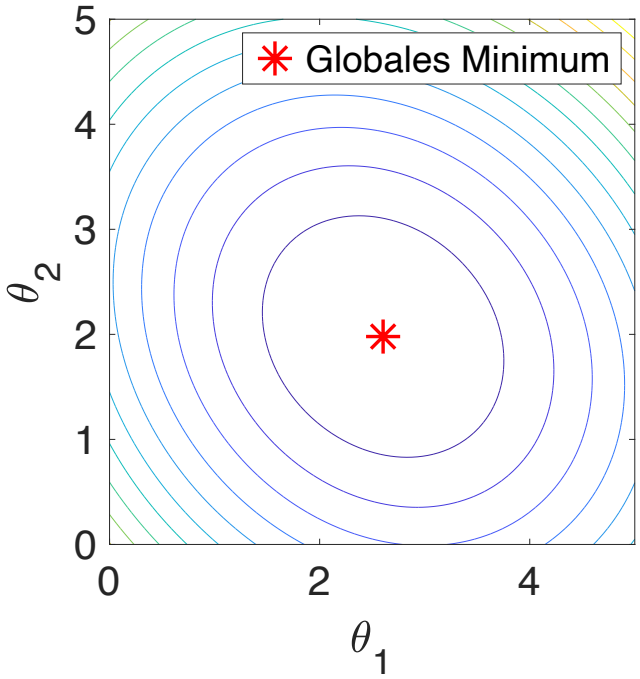
$$\nabla_{\underline{\theta}} \tfrac{1}{2}\underline{\theta}^T \underline{H}\,\underline{\theta} + \underline{g}^T\underline{\theta} + c \overset{!}{=} 0$$

$$\tfrac{1}{2}(\underline{H} + \underline{H}^T)\underline{\theta} + \underline{g} \overset{!}{=} 0$$

da $\underline{H}$ symmetrisch:

$$\underline{H}\,\underline{\theta} + \underline{g} \overset{!}{=} 0$$

$$\rightarrow \underline{\theta}^* = -\underline{H}^{-1}\underline{g}$$



3.3 Quadratische Probleme

Least Squares

Einschub: Symmetrie in quadratischen Matrix-Formen

- Die Matrix $\underline{H}$ einer quadratischen Funktion $\underline{\theta}^T \underline{H}\, \underline{\theta} + \underline{g}^T \underline{\theta} + c$ kann immer in eine symmetrische Matrix $\underline{\tilde{H}}$ umgeformt werden
- Die resultierende Funktion ist identisch zu der ursprünglichen Funktion
- Aus diesem Grund sollte jede quadratische Matrix-Form mittels einer symmetrischen Matrix $\underline{\tilde{H}}$ formuliert werden

$$\begin{bmatrix}\theta_1 & \theta_2\end{bmatrix} \underbrace{\begin{bmatrix} a & b \\ c & d \end{bmatrix}}_{\underline{H}} \begin{bmatrix}\theta_1 \\ \theta_2\end{bmatrix} = a\theta_1^2 + (b+c)\theta_1\theta_2 + d\theta_2^2 = \begin{bmatrix}\theta_1 & \theta_2\end{bmatrix} \underbrace{\begin{bmatrix} a & \frac{b+c}{2} \\ \frac{b+c}{2} & d \end{bmatrix}}_{\underline{\tilde{H}}} \begin{bmatrix}\theta_1 \\ \theta_2\end{bmatrix}$$

Einschub: Quadratisch symmetrische Matrizen

- Alle Matrizen der Form $\underline{H} = \underline{X}^T \underline{X}$ mit vollem Rang sind symmetrisch und positiv definit

3.3 Quadratische Probleme

Least Squares

Eine typische Anwendung des LS-Verfahrens ist die lineare Regression. Dabei wird ein Model gesucht, dass das Verhalten von gemessenen Punkten beschreibt. Beim LS-Verfahren wird dafür die quadratische Abweichung des Modells von den Messpunkten als Verlustfunktion gewählt. Ist das Modell $f(\underline{x}, \underline{\theta})$ linear in den Parametern $\underline{\theta}$ dann lässt es sich in Matrix Schreibweise wie folgt formulieren:

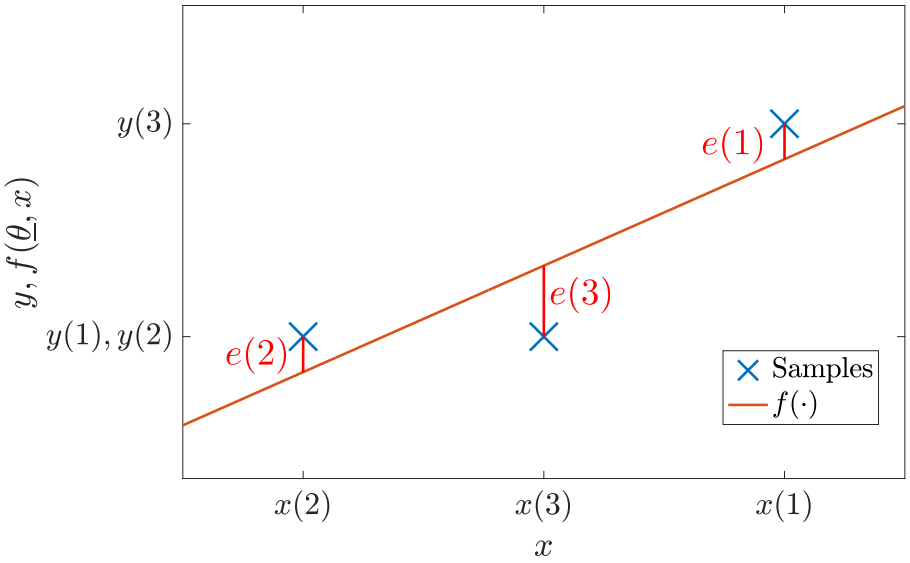
$$\min_{\underline{\theta}} \frac{1}{2} \sum_{i=1}^{N} \underbrace{(y(i) - f(x(i), \underline{\theta}))}_{e(i)}^2 = \min_{\underline{\theta}} \frac{1}{2} ||\underline{y} - f(\underline{x}, \underline{\theta})||^2 = \min_{\underline{\theta}} \frac{1}{2} (\underline{y} - \underline{X}\,\underline{\theta})^T (\underline{y} - \underline{X}\,\underline{\theta})$$

Durch Umformen der Gleichung:

$$\min_{\underline{\theta}} \frac{1}{2} \underline{\theta}^T \underbrace{\underline{X}^T \underline{X}}_{\underline{H}} \underline{\theta} \underbrace{- \underline{y}^T \underline{X}}_{\underline{g}^T} \underline{\theta} + \underbrace{\frac{1}{2} \underline{y}^T \underline{y}}_{c}$$

Dass führt zur bekannten Lösung:

$$\underline{\theta}^* = \left(\underline{X}^T \underline{X}\right)^{-1} \underline{X}^T \underline{y} = -\underline{H}^{-1} \underline{g}$$

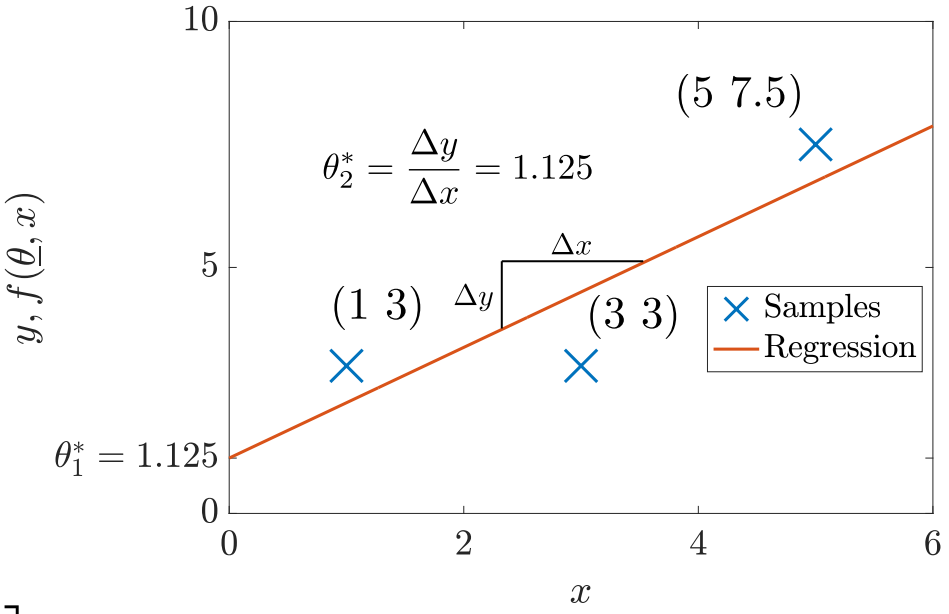


3.3 Quadratische Probleme

Least Squares

Ansatz: Lineare Funktion

$$\hat{y} = f(\underline{\theta}, x) = \theta_1 + \theta_2 \cdot x$$
$$= [1 \; x] \cdot \underbrace{\begin{bmatrix}\theta_1 \\ \theta_2\end{bmatrix}}_{\underline{\theta}}$$



Aufbau der Regressionsmatrix und des Ausgangsvektors:

$$\underline{X} = \begin{bmatrix}1 & x(1) \\ 1 & x(2) \\ 1 & x(3)\end{bmatrix} = \begin{bmatrix}1 & 1 \\ 1 & 3 \\ 1 & 5\end{bmatrix} \qquad \underline{y} = \begin{bmatrix}3 \\ 3 \\ 7.5\end{bmatrix}$$

Berechnung der optimalen Parameter:

$$\underline{\theta}^* = \left(\underline{X}^T \underline{X}\right)^{-1} \underline{X}^T \underline{y}$$
$$= \left(\begin{bmatrix}1 & 1 & 1 \\ 1 & 3 & 5\end{bmatrix} \begin{bmatrix}1 & 1 \\ 1 & 3 \\ 1 & 5\end{bmatrix}\right)^{-1} \begin{bmatrix}1 & 1 & 1 \\ 1 & 3 & 5\end{bmatrix} \begin{bmatrix}3 \\ 3 \\ 7.5\end{bmatrix} = \begin{bmatrix}3 & 9 \\ 9 & 35\end{bmatrix}^{-1} \begin{bmatrix}13.5 \\ 49.5\end{bmatrix} = \begin{bmatrix}1.125 \\ 1.125\end{bmatrix}$$

3.3 Quadratische Probleme

Least Squares – Regularisierung

Bei bestimmten Problemen kann es vorkommen, dass die Hessematrix $\underline{H}$ der quadratischen Verlustfunktion schlecht konditioniert ist. Ein möglicher Grund ist z.B. eine schlechte Anregung des Prozesses. Um in diesem Fall trotzdem eine geringe Parametervarianz der Schätzung zu erzeugen, kann *Regularisierung* genutzt werden. Regularisierung ist ein statistisches Verfahren, das die Varianz reduziert und als Preis dafür einen Bias in Kauf nimmt. Realisiert wird es über eine Ergänzung der normalen Kostenfunktion um eine Bestrafung der Parameter. Typischerweise wird die Summe der quadrierten Parameterwerte verwendet. Durch diese Wahl ist es weiterhin möglich das Problem analytisch zu lösen. Das Verfahren wird als *Ridge-Regression* bezeichnet. Für eine lineare Regression ergibt sich:

$$\min_{\underline{\theta}} I(\underline{\theta}, \lambda) = \min_{\underline{\theta}} \frac{1}{2} \left(\underline{e}^T \underline{e} + \lambda \underline{\theta}^T \underline{\theta} \right) = \min_{\underline{\theta}} \frac{1}{2} \sum_{i=1}^{N} e(i)^2 + \lambda \sum_{j=1}^{n} \theta_j^2$$

Dieses Optimierungsproblem **mit Strafterm ohne Nebenbedingungen** ist äquivalent zu dem folgenden Optimierungsproblem **ohne Strafterm aber mit Nebenbedingung**:

$$\min_{\underline{\theta}} I(\underline{\theta}, \lambda) = \min_{\underline{\theta}} \frac{1}{2} \underline{e}^T \underline{e}$$

$$\text{s.t.} \quad \sum_{j=1}^{n} \theta_j^2 \leq t^2$$

Mit dem Hyperparameter λ wird die Regularisierungsstärke bestimmt. Es gibt einen direkten 1:1 Zusammenhang zwischen λ und dem Grenzwert t.

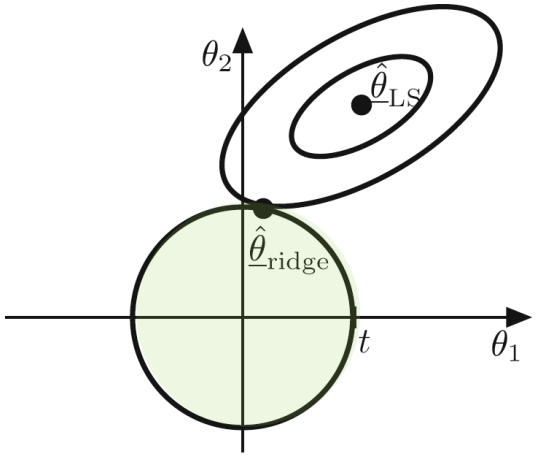
3.3 Quadratische Probleme

Least Squares – Regularisierung

Die Regularisierung sorgt bei der Schätzung der Parameter dafür, dass vor allem die Parameter genutzt werden, die einen signifikanten Einfluss haben. Bei Parametern ohne signifikanten Einfluss überwiegt der Bestrafungsterm und sie werden nicht bzw. wenig für die Optimierung genutzt.

Dies wird auch bei der Darstellung des äquivalenten Problems (Optimierungsproblems mit Nebenbedingung) deutlich

- Parameter θ_1 hat einen deutlich geringeren Einfluss auf das Optimierungsproblem als Parameter θ_2
- Durch die Regularisierung wird hauptsächlich θ_2 für die Lösung genutzt, θ_1 wird sehr klein



Für die Lösung des regularisierten Optimierungsproblems muss die Hessematrix des unregularisierten Problems modifiziert werden:

$$\underline{H}_{ridge} = \underline{H}_{LS} + \lambda \underline{I} \qquad \text{mit} \qquad \underline{I} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}_{n \times n}$$

3.3 Quadratische Probleme

Problem

$$\min_{\underline{\theta}} \quad \frac{1}{2}\underline{\theta}^T \underline{H}\,\underline{\theta} + \underline{g}^T\underline{\theta} + c$$

$$\text{s.t.} \quad \underline{A}\,\underline{\theta} - \underline{b} = 0 \quad \text{GNB}$$

$$\underline{G}\,\underline{\theta} - \underline{h} \leq 0 \quad \text{UNB}$$

Quadratic Programm QP

Mit Nebenbedingungen ist eine geschlossene Lösung des Optimierungsproblems nur dann möglich, wenn ausschließlich Lineare Gleichungsnebenbedingungen (GNB) existieren. Entweder können lineare Gleichungsnebenbedingungen direkt in die Kostenfunktion eingesetzt werden oder das Problem lässt sich umformulieren und mit den KKT–Bedingungen (siehe Kapitel 5) lösen.

Existieren neben Gleichungsnebenbedingungen auch Ungleichungsnebenbedingungen (UNB) existiert keine geschlossene Lösung des Optimierungsproblems. Allerdings lässt sich auch dieses Problem gut lösen. Zur Lösung existieren verschiedene Strategien.

- *Active Set Methoden* (genauere Erklärung nächste Folie)
- *Interior Point Methoden* (Innere-Punkte-Verfahren)

Die *Interior Point Methoden* sorgen durch Penalty-Funktionen dafür, dass bei der iterativen Minimumsuche nur Punkte innerhalb des gültigen Bereichs geprüft werden.

Für beide Strategien gibt es verschiedene Solver die standardmäßig in vielen Software-Optimierungspaketen enthalten sind. In Matlab lässt sich so ein Problem zum Beispiel mit dem **`quadprog`** Befehl lösen.

3.3 Quadratische Probleme

Problem

$$\min_{\underline{\theta}} \quad \frac{1}{2}\underline{\theta}^T \underline{H}\,\underline{\theta} + \underline{g}^T\underline{\theta} + c$$

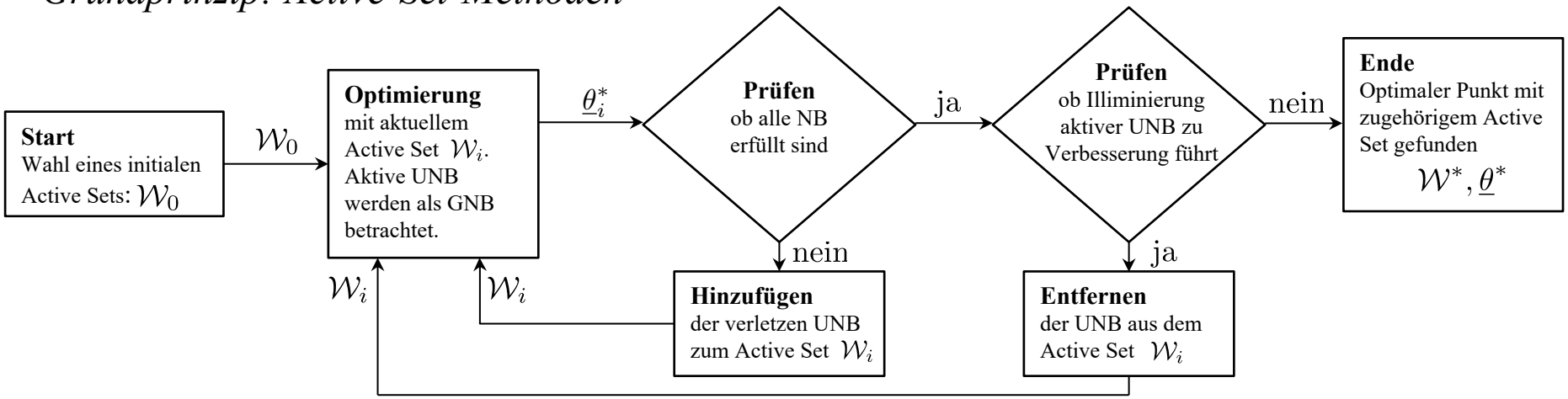
$$\text{s.t.} \quad \underline{A}\,\underline{\theta} - \underline{b} = 0 \quad \text{GNB}$$

$$\underline{G}\,\underline{\theta} - \underline{h} \leq 0 \quad \text{UNB}$$

Quadratic Programm QP

Die *Active-Set-Methoden* funktionieren alle nach dem selben Grundprinzip. Das Ziel dieser Verfahren ist es, zwischen aktiven und inaktiven UNBs zu unterscheiden. Durch diese Unterscheidung ist es möglich, die aktiven UNBs als GNBs zu bearbeiten und die inaktiven UNBs bei der Berechnung des Optimierungsproblems nicht berücksichtigen zu müssen. Es ergibt sich ein Optimierungsproblem, dass aus-schließlich durch GNBs beschränkt wird und damit analytisch lösbar ist.
Zu Beginn der Optimierung ist allerdings unklar welche UNBs aktiv und welche inaktiv sind. Durch geschicktes Ausprobieren versuchen die *Active-Set-Methoden* das benötigte Set an aktiven UNBs $\mathcal{W}^*$ herauszufinden. Dabei wird meist iterativ vorgegangen, bis das gesuchte Set gefunden ist.

Grundprinzip: Active-Set-Methoden



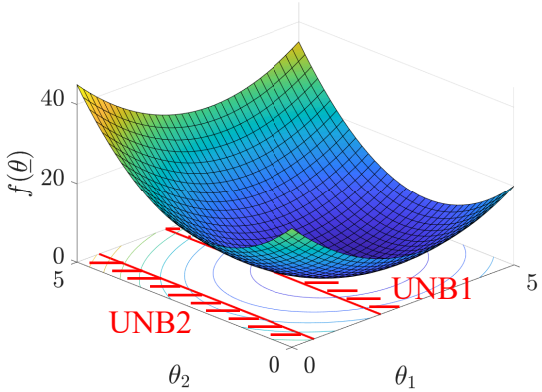
3.3 Quadratische Probleme

Quadratic Programm QP – Anwendungsbeispiel

Ein einfaches Beispiel für ein QP mit Nebenbedingungen ist das Einschränken des möglichen Parameterraumes für einen Parameter.

Optimierungsproblem:

$$\begin{aligned} \min_{\underline{\theta}} \quad & 2(\theta_1 - 3)^2 + 3(\theta_2 - 2)^2 \\ \text{s.t.} \quad & 0.5 \leq \theta_1 \leq 2 \end{aligned}$$



Vorbereitung: Umformen in Standard-Form

Verlustfunktion:

$$\begin{aligned} 2(\theta_1 - 3)^2 + 3(\theta_2 - 2)^2 &= 2\theta_1^2 + 3\theta_2^2 - 12\theta_1 - 12\theta_2 + 30 \\ &= \frac{1}{2} \underbrace{\begin{bmatrix} \theta_1 & \theta_2 \end{bmatrix}}_{\underline{\theta}^T} \underbrace{\begin{bmatrix} 4 & 0 \\ 0 & 6 \end{bmatrix}}_{\underline{H}} \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} + \underbrace{\begin{bmatrix} -12 & -12 \end{bmatrix}}_{\underline{g}^T} \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} + \underbrace{30}_{c} \end{aligned}$$

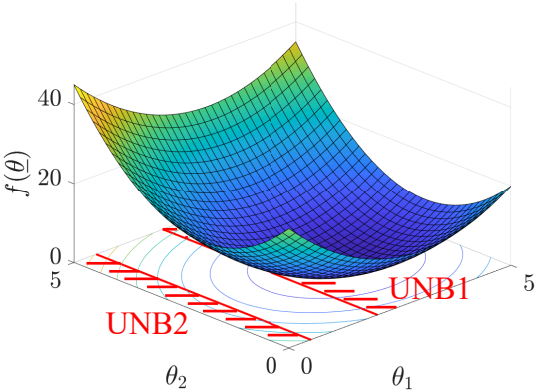
Nebenbedingungen:

UNB1 $\quad \theta_1 \leq 2$

UNB2 $\quad \theta_1 \geq 0.5$

$$\rightarrow \quad \begin{aligned} \theta_1 - 2 &\leq 0 \\ -\theta_1 + 0.5 &\leq 0 \end{aligned} \quad \rightarrow \quad \underbrace{\begin{bmatrix} 1 & 0 \\ -1 & 0 \end{bmatrix}}_{\underline{G}} \underline{\theta} - \underbrace{\begin{bmatrix} 2 \\ -0.5 \end{bmatrix}}_{\underline{h}} \leq \underline{0}$$

3.3 Quadratische Probleme



Quadratic Programm QP – Anwendungsbeispiel

Die UNBs beziehen sich in diesem Beispiel ausschließlich auf θ_1. Daher kann der Parameter θ_1 in der Verlustfunktion eliminiert werden, wenn ein fester Wert vorgegeben wird.

Für die allgemeine GNB $\theta_1 = a$ gilt:

$$\begin{aligned} 2(\theta_1 - 3)^2 + 3(\theta_2 - 2)^2 &= 2(a-3)^2 + 3(\theta_2 - 2)^2 \\ &= 3\theta_2^2 - 12\theta_2 + 2a^2 - 12a + 30 \\ &= \frac{1}{2}\theta_2 \underbrace{6}_{\tilde{h}} \theta_2 + \underbrace{(-12)}_{\tilde{g}} \theta_2 + \underbrace{2a^2 - 12a + 30}_{\tilde{c}} \end{aligned}$$

Damit ergibt sich in Abhängigkeit der Konstante a der optimale Parameter $\theta_2^*(a)$:

$$\theta_2^*(a) = -\tilde{h}^{-1}\tilde{g} = -\frac{1}{6} \cdot (-12) = 2$$

In diesem Sonderfall ist der optimale Parameter θ_2^* unabhängig von a.

Wieso ist das so in diesem Beispiel? Ursache?

3.3 Quadratische Probleme

Problem

$$\min_{\underline{\theta}} \quad 2(\theta_1 - 3)^2 + 3(\theta_2 - 2)^2$$

$$\text{s.t.} \quad 0.5 \leq \theta_1 \leq 2$$

Quadratic Programm QP – Anwendungsbeispiel

Möglichkeit 1: keine UNB ist aktiv

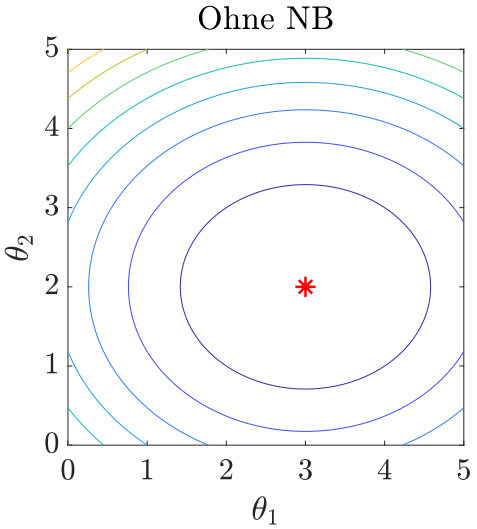
$$\mathcal{W}_1 = \{\}$$

$$\underline{\theta}_1^* = -\underline{\underline{H}}^{-1}\underline{g} = -\begin{bmatrix} \frac{1}{4} & 0 \\ 0 & \frac{1}{6} \end{bmatrix}\begin{bmatrix} -12 \\ -12 \end{bmatrix} = \begin{bmatrix} 3 \\ 2 \end{bmatrix}$$

Prüfen der NB

$$\begin{bmatrix} 1 & 0 \\ -1 & 0 \end{bmatrix}\begin{bmatrix} 3 \\ 2 \end{bmatrix} - \begin{bmatrix} 2 \\ -0.5 \end{bmatrix} = \begin{bmatrix} 1 \\ -2.5 \end{bmatrix} \leq \underline{0}$$

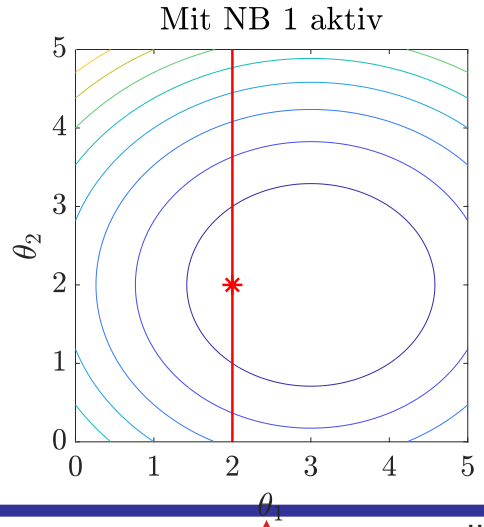
Nicht erfüllt



Möglichkeit 2: UNB1 ist aktiv

$$\mathcal{W}_2 = \{\text{UNB}_1\}$$
$$\Rightarrow \theta_1^* = 2$$

$$\theta_2^*(a) = -\tilde{h}^{-1}\tilde{g} = -\frac{1}{6}\cdot(-12) = 2$$



Prüfen der NB

$$\begin{bmatrix} 1 & 0 \\ -1 & 0 \end{bmatrix}\begin{bmatrix} 2 \\ 2 \end{bmatrix} - \begin{bmatrix} 2 \\ -0.5 \end{bmatrix} = \begin{bmatrix} 0 \\ -1.5 \end{bmatrix} \leq \underline{0}$$

Erfüllt

3.3 Quadratische Probleme

Problem

$$\min_{\underline{\theta}} \quad 2(\theta_1 - 3)^2 + 3(\theta_2 - 2)^2$$

$$\text{s.t.} \quad 0.5 \leq \theta_1 \leq 2$$

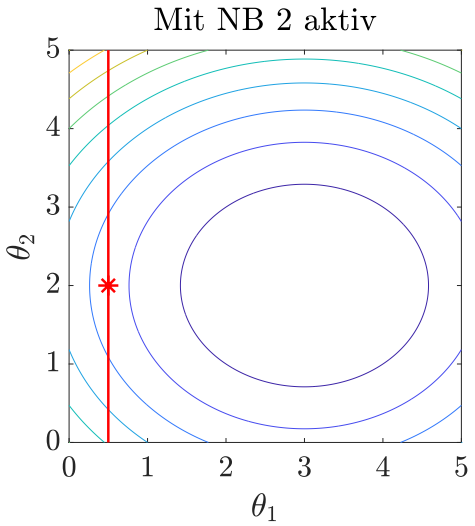
Quadratic Programm QP – Anwendungsbeispiel

Möglichkeit 3: NB2 ist aktiv $\Rightarrow \theta_1^* = 0.5$

$$\theta_2^*(a) = -\tilde{h}^{-1}\tilde{g} = -\frac{1}{6} \cdot (-12) = 2$$

$\mathcal{W}_3 = \{\text{UNB}_2\}$

$\Rightarrow \theta_1^* = 0.5$



Prüfen der NB

$$\begin{bmatrix} 1 & 0 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} 0.5 \\ 2 \end{bmatrix} - \begin{bmatrix} 2 \\ -0.5 \end{bmatrix} = \begin{bmatrix} -1.5 \\ 0 \end{bmatrix} \leq \underline{0}$$

Erfüllt

Möglichkeit 4: NB1 und NB2 sind aktiv $\mathcal{W}_4 = \{\text{UNB}_1, \text{UNB}_2\}$

Nicht möglich da sich NB1 und NB2, wenn sie als Gleichungsnebenbedingungen betrachtet werden, gegenseitig ausschließen.

→ Beste Lösung die alle NB erfüllt ist *Möglichkeit 2* (NB1 ist aktiv)

4. Prädiktive Regelung

Prof. Dr.-Ing.
Oliver Nelles

Inhalt Kapitel 4

4.1 Einführung

Reglersynthese

Vorwissen / Mächtigkeit

- PID-Regler mit Standard-Einstellungen — Kein Modell
- Fuzzy-Regler — Expertenwissen (Qualitativ)
- PID-Regler mit Einstellregeln (z.B. Ziegler-Nichols) — Primitiv Model (Approx.)
- Optimierung der Reglerparameter
  (Geg.: Reglerstruktur, z.B. PID)
- Polvorgabe $(N(s))$
  - Ein-/Ausgangsdarstellung
  - Zustandsraum
- Kompensationsregler $\left(\frac{Z(s)}{N(s)}\right)$
- Internal Model Control
- Prädiktive Regler

— Modell

4.1 Einführung

Geschichte der prädiktiven Regelung

1960er: Erste theoretische Grundlagen zur Prädiktiven Regelung: Gleitender Horizont

"... One technique for obtaining a feedback controller synthesis from knowledge of open-loop controllers is to measure the current control process state and then compute very rapidly for the open-loop control function. The first portion of this function is then used during a short time interval, after which a new measurement of the function is computed for this new measurement. The procedure is then repeated. ..."

Lee, E.B. und Markus, L. (1967)."Foundations of optimal control theory". New York: Wiley.

1970er: 1. Generation der Prädiktiven Regelung (Ohne Nebenbedingungen)

1973: DMC: entwickelt von Ingenieuren von Shell Oil

1976: IDCOM: von Richalet et al.

4.1 Einführung

Geschichte der prädiktiven Regelung

Anfang 1980er: 2. Generation der Prädiktiven Regelung mit Nebenbedingungen

1983: QDMC: Weiterentwicklung der DMC von Shell Oil

Ende 1980er: 3. Generation der Prädiktiven Regelung mit Nebenbedingungen

1988: IDCOM-M: SETPOINT Inc.:

Weiterentwicklung von IDCOM durch Priorisierung von Nebenbedingungen

SMOC: Shell Oil

Mitte 1990er: 4. Generation der Prädiktiven Regelung (Grafische Oberfläche)

1995: DMC-Plus: Honeywell Hi-Spec

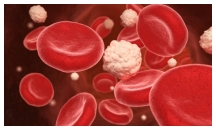
1996: RMPCT: Aspen Tech

- Erste Grafische Oberflächen und Einführung einer Regler-Hierarchie

4.1 Einführung

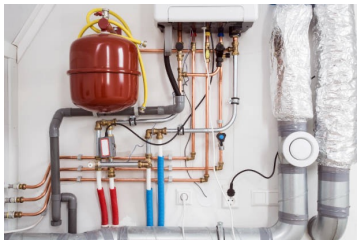
Zeitrahmen	Ebene	Aufgaben	Tools
Tage	Management	Strategie, Planung, Logistik	SAP
Stunden	Leitsystem	Optimierung der Arbeitspunkte und Profile, Data Mining	Prozessleitsystem
Minuten	langsame Regelkreise	überlagerte Regelung, Diagnose	Adaptive und prädiktive Regelung, Mehrgrößenregelung
Sekunden	schnelle Regelkreise	unterlagerte Regelung, Steuerungen	PID-Regelung

4.1 Einführung



Beispiele

- Petrochemischen und chemischen Anlagen,
- Lebensmittelverarbeitung,
- Autoindustrie,
- Steuerung von chemischen Rohrreaktoren,
- Normalisierung des Blutzuckerspiegels von kritisch kranken Patienten,
- Stromrichtern,
- Steuerung von stromerzeugenden Drachen bei wechselnden Windverhältnissen,
- mechatronischen Systemen (z. B. mobilen Robotern),
- Stromerzeugung,
- Luft- und Raumfahrt,
- HLK-Systemen (Gebäudesteuerung)

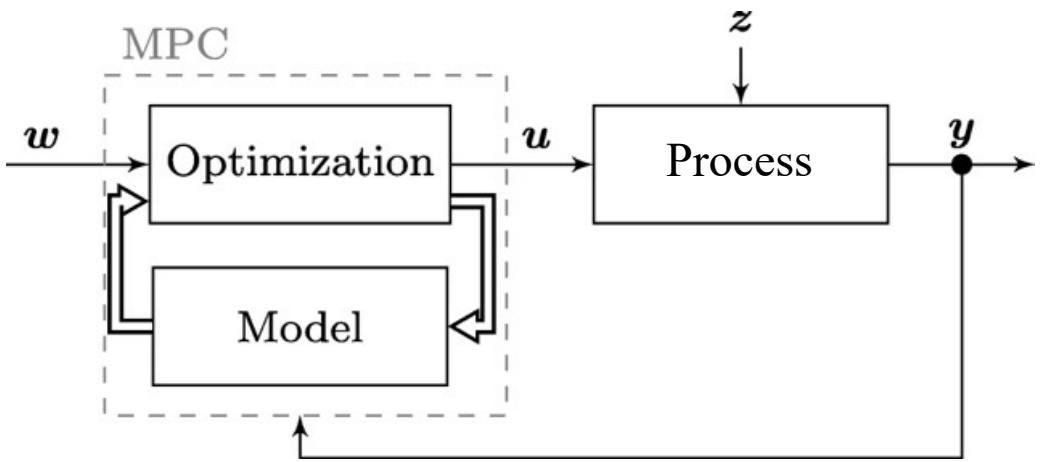


4.1 Einführung

Konzept prädiktive Regelung

Die modellprädiktive Regelung wird im Englischen oft mit MPC (*model predictive control*) abgekürzt. Der prädiktive Regler beobachtet und speichert alle Prozessvariablen (Stell- und Regelgrößen, messbare Störgrößen). Im Anschluss nutzt der Regler ein *Modell* der Prozessdynamik, um den zukünftigen Verlauf der der Regelgrößen vorherzusagen („*prädizieren*“). Dies unterscheidet die prädiktive Regelung von klassischen, da er durch das Modell ein Stückweit „in die Zukunft schauen“ kann.

Er berechnet, wohin sich die Regelgrößen bewegen werden, wenn der Regler nicht eingreift (*free response*). Darüber hinaus kann der Regler auch „ausprobieren" (simulieren), wie sich zukünftige Änderungen der Stellgrößen auf die Regelgrößen auswirken *(forced response)*. Mit Hilfe eines Optimierungsverfahrens wird die beste Stellstrategie ausgewählt.



w: Führungsgröße
u: Stellgröße
y: Regelgröße
z: Störgröße

4.1 Einführung

Notwendige Bestandteile einer prädiktive Regelung

- Modell (je genauer desto besser)
- Optimierer (ohne Nebenbedingungen existiert auch analytische Lösung)

Optionale Bestandteile einer prädiktive Regelung

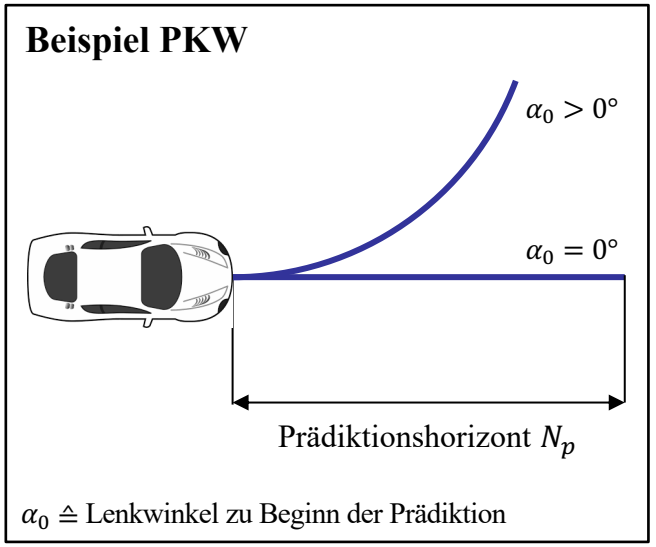
- Störübertragungsfunktion: Einbezug der Störgröße
- Zukünftiger Verlauf der Führungsgröße
- Nebenbedingungen: z.B. Limits in der Stellgröße

4.1 Einführung

Prädiktion

Die Prädiktion beschreibt die Vorhersage des zukünftigen Verlaufes der Regelgröße y. Man spricht dabei von der „freien Systemantwort“, da für die Prädiktion angenommen wird, dass die vorhandenen Stellgrößen u auf ihrem aktuellen Wert beruhen. Die Berechnung basiert auf folgender Grundlage:

- Auswertung eines dynamischen Prozessmodells bis zum **Prädiktionshorizont** N_p (*prediction horizon*) mit den aktuell eingestellten Stellgrößen $u(k)$
- Nutzung der für das Prozessmodell nötigen vergangenen, gemessenen Regel- und Stellgrößen (+ ggf. Störgrößen) $u(k-1) \dots u(k-n_1),\ y(k-1) \dots y(k-n_2),$ $\big(z(k-1) \dots z(k-n_3)\big)$



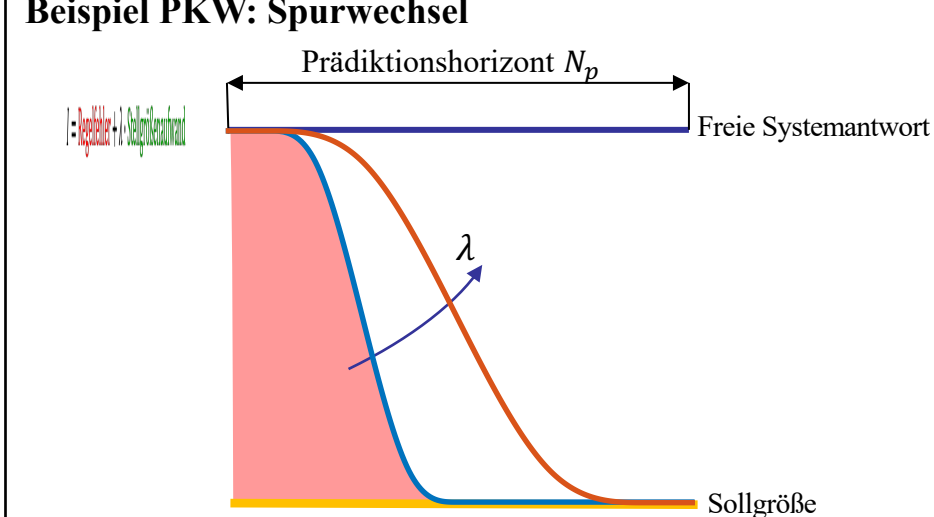
4.1 Einführung

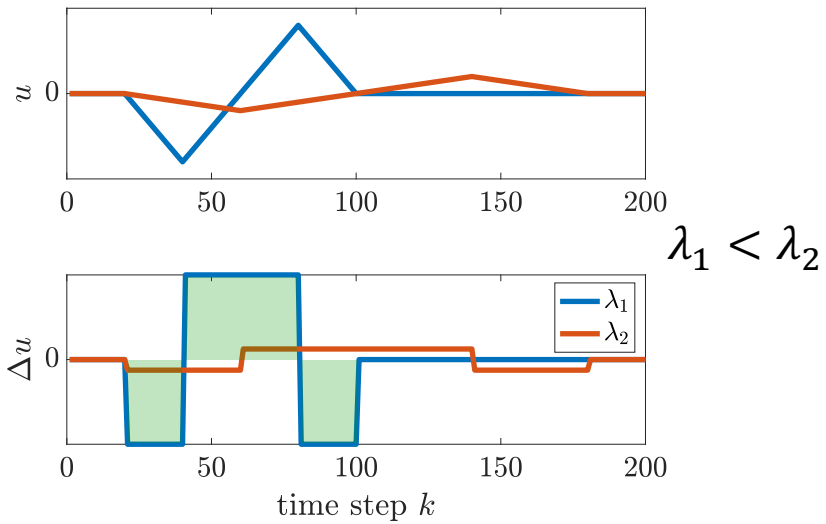
Dynamische Optimierung

Die dynamische Optimierung löst ein Optimierungsproblem in welchem der prädizierte Regelfehler e minimiert werden soll. Als zu optimierende Größen werden die zukünftigen Stellgrößenänderungen bis zum **Stellhorizont** (*control horizon*) N_u genutzt. Typischerweise wird die Stellgrößenänderung Δu selbst auch in der Kostenfunktion berücksichtigt, um zu großen Stelleingriffen entgegenzuwirken.

$$I = \text{Regelfehler} + \lambda \cdot \text{Stellgrößenaufwand}$$

Beispiel PKW: Spurwechsel





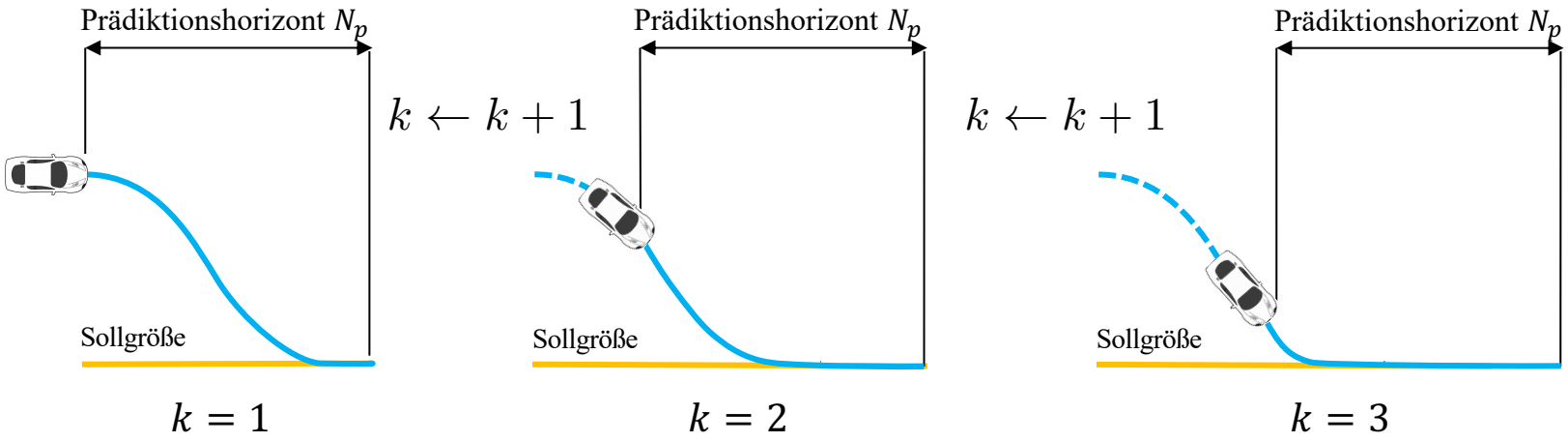
4.1 Einführung

Prinzip des gleitenden Horizontes (*Receding Horizon Strategy*)

Der gleitende Horizont beschreibt das iterative Verfahren des MPC-Reglers. Nachdem die berechnete Stellgröße für einen Zeitschritt gehalten wird, startet die dynamische Optimierung erneut. Dabei verschieben sich die Berechnungshorizonte um einen Zeitschritt in die Zukunft. D.h. die Optimierung minimiert folgende Verlustfunktion:

$$\min_{u(k+1)\ldots u(k+N_u-1+1)} I = \sum_{i=1}^{N_p} e(k+i+1|k+1)^2 + \lambda \sum_{i=0}^{N_u-1} \Delta u(k+i+1)^2$$

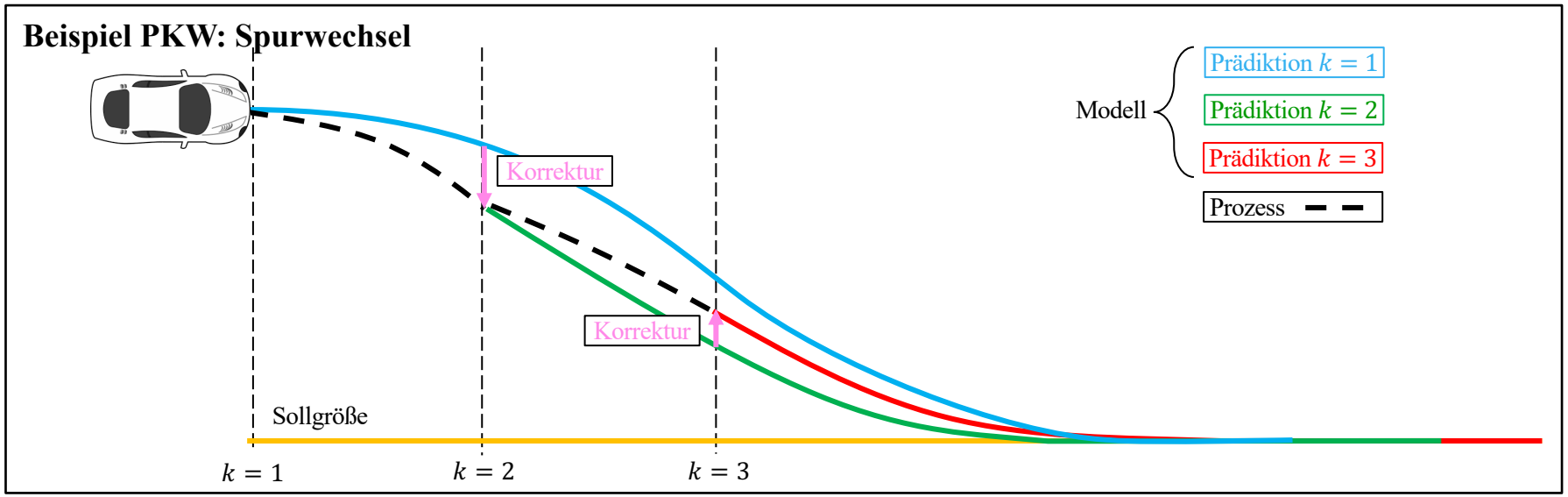
Beispiel PKW: Spurwechsel



4.1 Einführung

Korrektur der Vorhersage

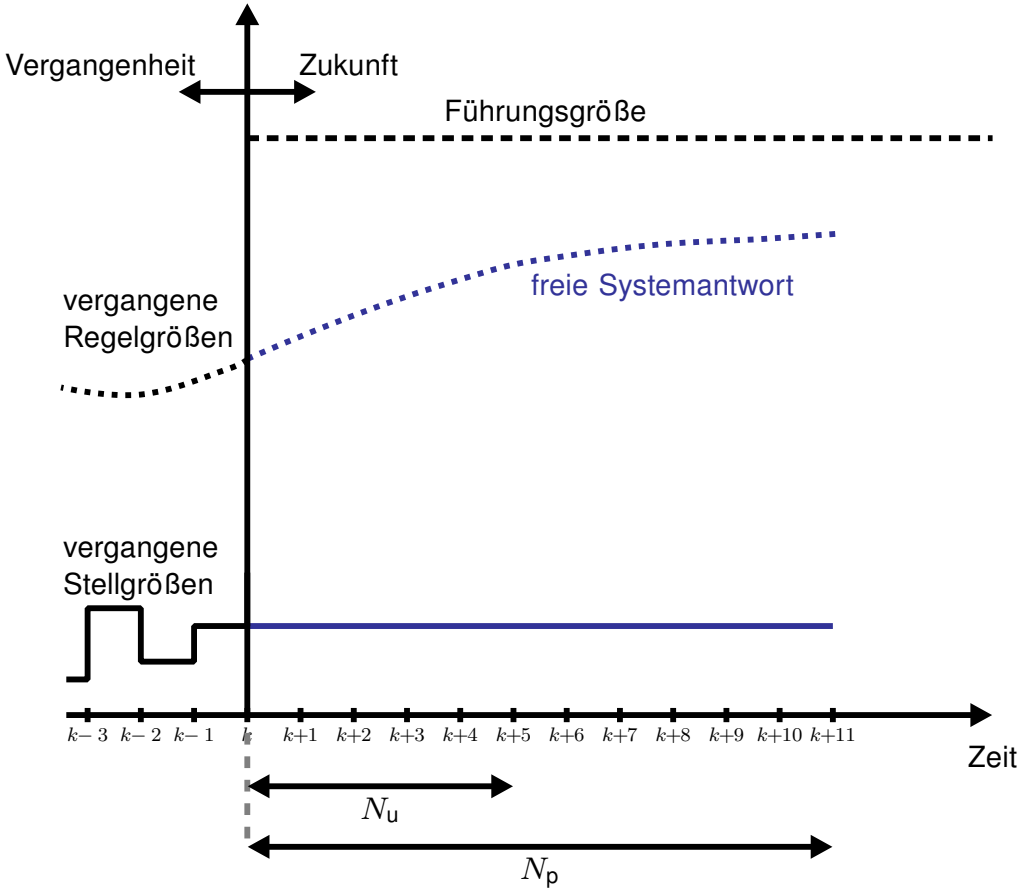
Das bisher erklärte Prinzip ist eine *Optimalsteuerung*. Charakteristisch für eine *Regelung* ist Rückkopplung einer oder mehrerer gemessener Größen. Dies wird im MPC-Regler durch die Korrektur der Vorhersage umgesetzt. In jeder Iteration wird die Optimierung aus einem Anfangszustand heraus gestartet. Dabei kann der prädizierte Anfangszustand aus dem vorherigen Zeitschritt Abweichungen zum tatsächlichen Zustand aufweisen (z.B. Modellfehler, Störung). Um dies bei der Regelung zu berücksichtigen, wird eine Korrektur der Vorhersage vorgenommen.



4.1 Einführung

Grundprinzip der prädiktiven Regelung

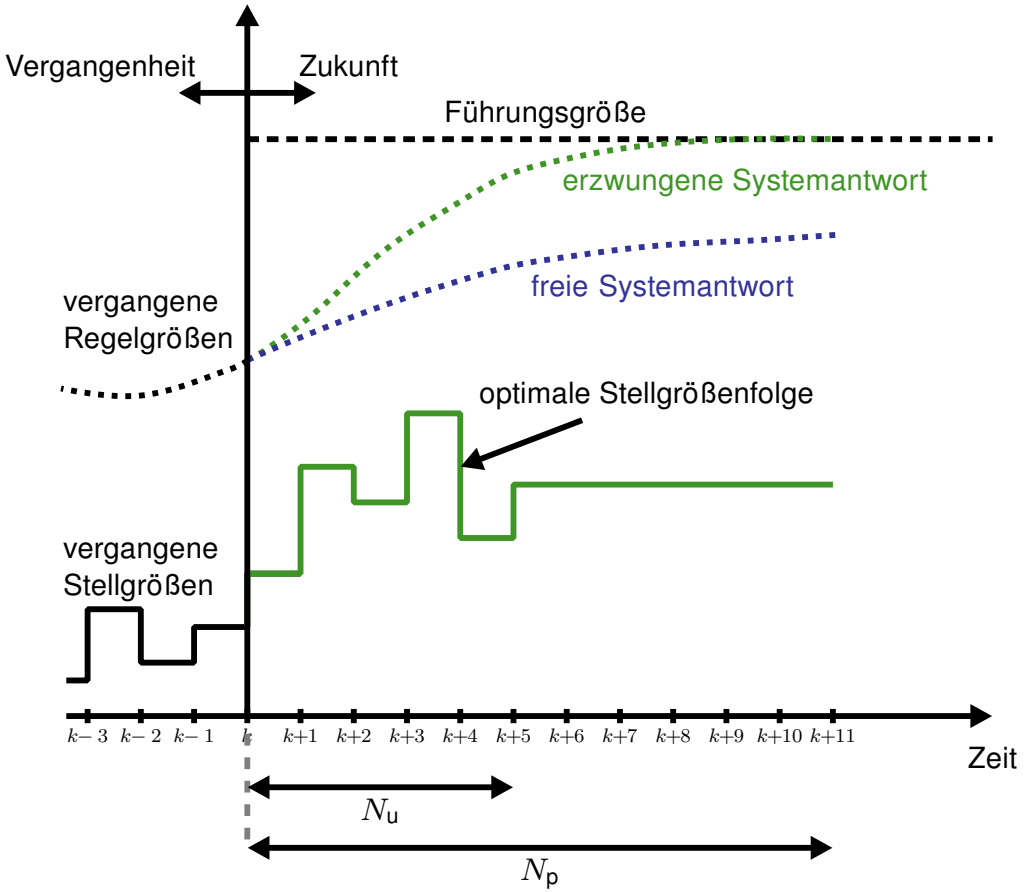
Zuerst wird prädiziert, wie der Prozess reagiert, falls kein Stelleingriff vorgenommen wird (Freie Systemantwort, *free response*).



4.1 Einführung

Grundprinzip der prädiktiven Regelung

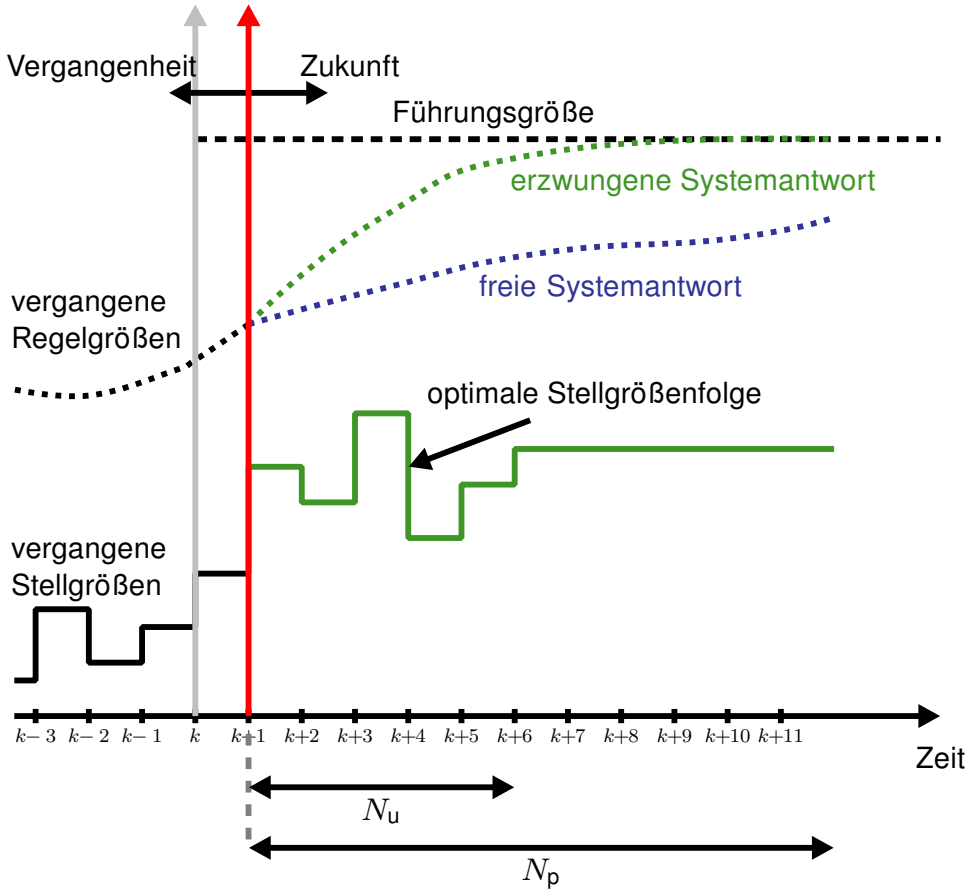
Nun wird eine optimale Stellgrößenfolge bestimmt. Diese bewirkt die erzwungene Systemantwort (*forced response*).



4.1 Einführung

Grundprinzip der prädiktiven Regelung

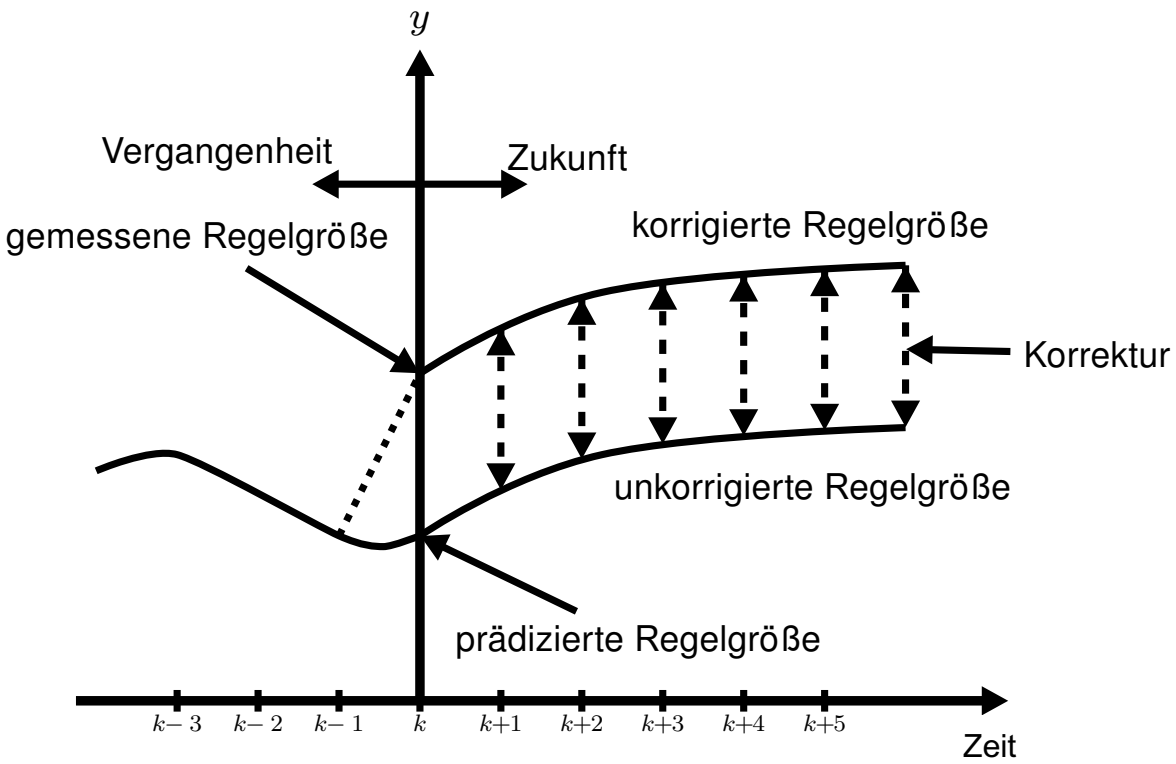
Im Anschluss gleitet der Horizont von Zeitschritt k auf $k + 1$.



4.1 Einführung

Korrektur der Vorhersage – Schließen des Regelkreises

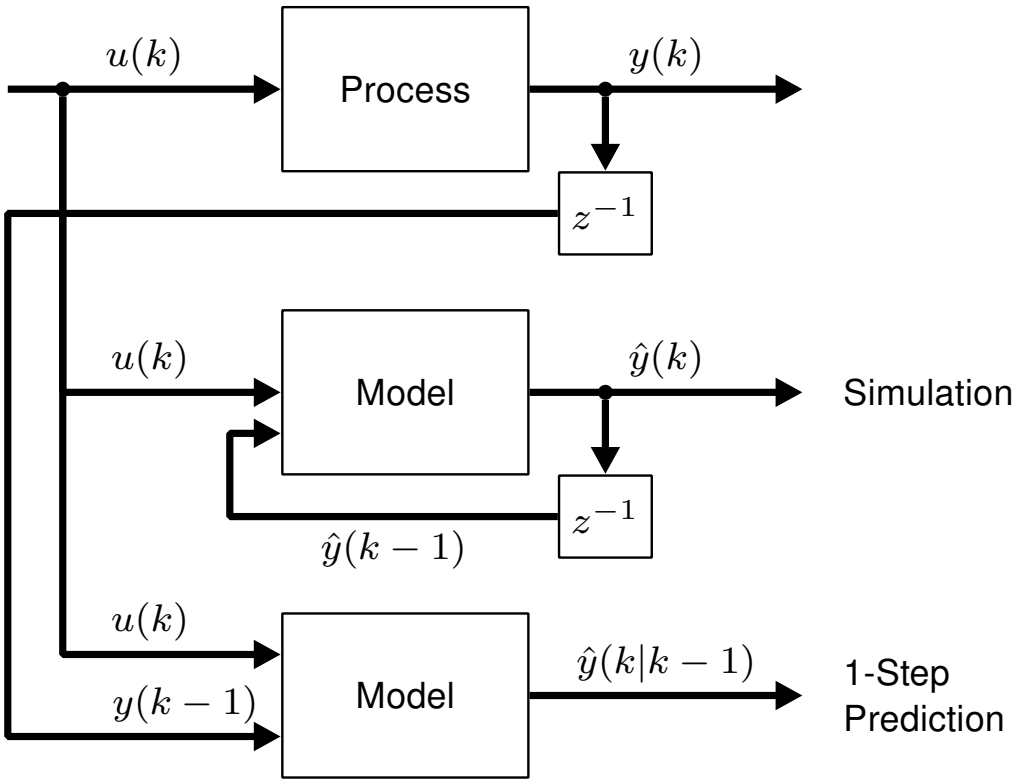
Die Regelgröße wird mit Hilfe des aktuellen Messwerts für die Regelgröße $y(k)$ korrigiert. Dabei wird der Prädiktionsfehler zum Zeitpunkt k durch $y(k) - \hat{y}(k|k-1)$ bestimmt. Dieser Fehler wird in jedem Prädiktionszeitpunkt auf die prädizierte Regelgröße addiert, um diese zu korrigieren.



4.1 Einführung

Prädiktion vs. Simulation

Bei der Simulation werden lediglich die Stellgrößen bzw. die prädizierten Regelgrößen genutzt. Im Gegensatz hierzu, werden bei der Ein-Schritt Prädiktion die *gemessenen* Regelgrößen genutzt. Im Fall der Prädiktion muss keine Korrektur der Vorhersage durchgeführt werden.

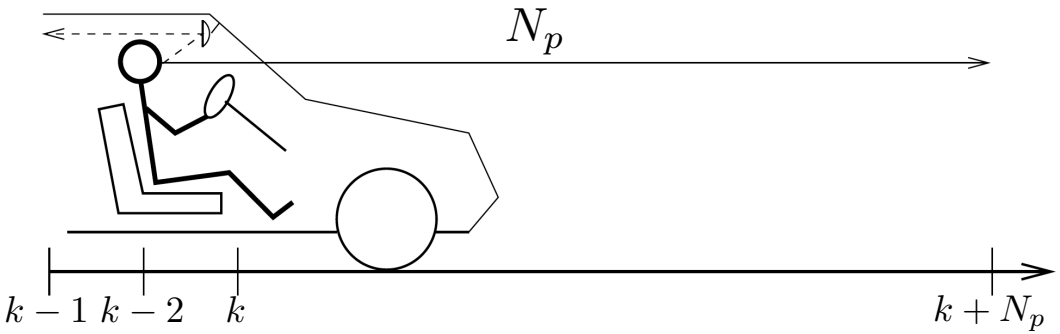


4.1 Einführung

Prädiktiver Regler: Analogie Auto

Die Strategie der prädiktiven Regelung kann mit Autofahren verglichen werden. Der Fahrer kennt die gewünschte Referenztrajektorie für einen endlichen Kontroll-/Prädiktionshorizont und entscheidet unter Berücksichtigung der Fahrzeugeigenschaften (*Modell des Fahrzeugs*), welche Steuerungsmaßnahmen (Beschleunigen, Bremsen, Lenken) zu ergreifen sind, um der gewünschten Trajektorie zu folgen. Dieses Vorgehen wird für die nächste Steuerungsentscheidung wiederholt, und zwar mit einem gleitenden Horizont.

Bei der Verwendung klassischer Steuerungs- oder Regelungsverfahren wie PID-Reglern werden die Regelungsmaßnahmen nur auf der Grundlage vergangener oder dem derzeitigen Regelfehler getroffen. Zukünftige Größen werden nicht betrachtet.



4.1 Einführung

Sollwertverlauf

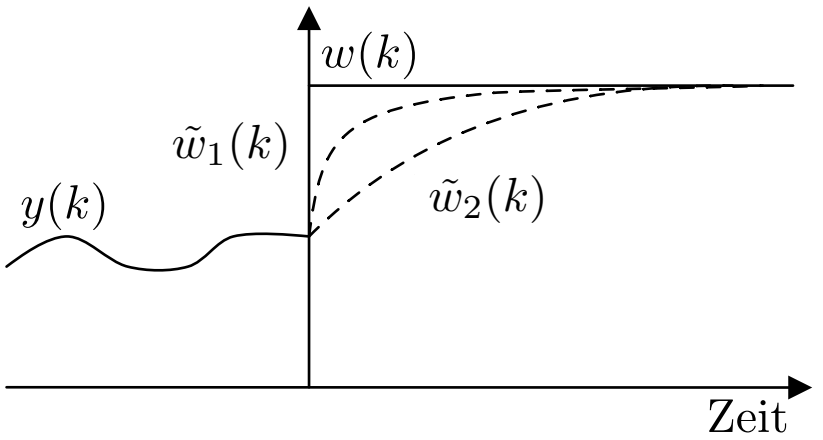
- Wenn die Entwicklung des Sollwerts a priori bekannt ist, kann das System reagieren, bevor die Änderung tatsächlich stattgefunden hat (z. B. in der Robotik, bei Servoantrieben oder Batch-Prozessen)
- Der Sollwert kann dabei auch gefiltert werden, damit glattere Regelgrößenverläufe erzielt werden können (z.B. Verzögerungsglied erster Ordnung):

$$\tilde{w}_j(k+i) = \alpha \tilde{w}_j(k+i-1) + (1-\alpha) w(k+i)$$

Der Sollwertverlauf muss aber *nicht zwingend bekannt* sein**:**

- Für den Sollwert kann ebenfalls angenommen werden, dass dieser konstant bleibt oder man extrapoliert diesen (z.B. linear)

→ Dadurch können prädiktive Regler auch bei nicht a priori bekannten Sollwertverläufen angewandt werden



4.1 Einführung

Ablauf der prädiktiven Regelung

Pseudo-Code:

```
for k=1:N_sim

    y(k) = Prozess(u(k-1)); % Prozess ohne Durchgriff

    du(k) = MPC(u(k-1), u(k-2),..., y(k), y(k-1)...);

    u(k) = u(k-1)+du(k);  % Bestimmung der aktuellen Stellgröße

end
```

Zum aktuellen Zeitschritt k wird der Prozessausgang gemessen. Anschließend wird eine Stellgröße optimiert, welche direkt auf den Prozess gegeben wird. *Ohne Durchgriff* wirkt sich diese dann in der nächsten Iteration $(k+1)$ auf den Prozess aus.

In den folgenden Algorithmen werden immer nur Systeme ohne Durchgriff betrachtet, da nur diese in der Realität auftreten.

4.1 Einführung

Wahl des Prädiktionshorizonts

$$\min_{u(k)\ldots u(k+N_u-1)} I = \sum_{i=N_1}^{N_p} e(k+i|k)^2 + \lambda \sum_{i=0}^{N_u-1} \Delta u(k+i)^2$$

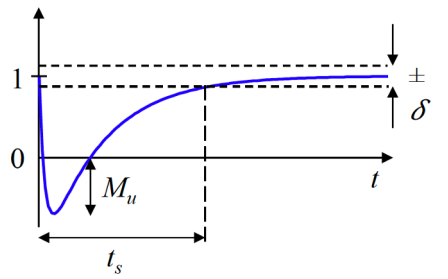
N_1:

Falls eine Totzeit bekannt ist, sollte diese als untere Grenze der Prädiktion gewählt werden: $N_1 = d$ $(T_d = d \cdot T_0)$. Die Regelgröße kann nämlich nicht durch die ersten Stellgrößen beeinflusst werden. Das Modell muss hierfür allerdings auch genügend viele vergangene Eingangsgrößen besitzen. Falls die Totzeit nicht bekannt oder variabel ist, sollte $N_1 = 1$ genutzt werden.

N_p:

- Falls das System nicht phasenminimal ist, sollte der der Prädiktionshorizont so gewählt werden, dass das Verhalten in die positive Richtung ebenfalls prädiziert werden kann.
- Bei langsameren Zeitkonstanten des Systems wird ein längerer Prädiktionshorizont benötigt.

Sprungantwort nicht phasenminimales System



4.1 Einführung

Wahl des Stellhorizonts

$$\min_{u(k)\ldots u(k+N_u-1)} I = \sum_{i=N_1}^{N_p} e(k+i|k)^2 + \lambda \sum_{i=0}^{N_u-1} \Delta u(k+i)^2$$

N_u:

Um eine optimale Regelung zu erreichen, muss $N_u = \infty$ gewählt werden. Dies ist in der Praxis zum einen nicht umsetzbar und zum anderen auch nicht notwendig. Durch die Wahl $N_u = 1$ können bereits akzeptable Regel-Ergebnisse erzielt werden. Je größer der Stellhorizont gewählt wird, desto flexibler kann die Regelgröße beeinflusst werden. Durch die höhere Anzahl an zu optimierenden Parametern wird aber auch das Optimierungsproblem immer komplexer. Für einfache und stabile Systeme genügt $N_u = 1$, da hier die Stellgröße kaum geändert werden muss.

Falls $N_u < N_p$ gilt, wird die Stellgröße für die nachfolgenden Prädiktionen konstant gehalten:

$\Delta u(i > N_u) = 0$

4.1 Einführung

$$\Delta u(k) = u(k) - u(k-1)$$

Berechnung der nächsten optimalen Stellgröße

Die übliche Verlustfunktion lautet:

$$\min_{u(k)\ldots u(k+N_u-1)} I = \sum_{i=1}^{N_p} e(k+i|k)^2 + \lambda \sum_{i=0}^{N_u-1} \Delta u(k+i)^2$$

Durch die Bestrafung der Stellgrößenänderung ist die optimale Lösung des zweiten Terms $\Delta u = 0$. In diesem Term wird also nur bestraft, falls sich eine Stellgröße ändert und nicht die absolute Höhe der Stellgröße.

Wenn allerdings u anstelle von Δu in die Verlustfunktion eingehen würde, wäre die optimale Lösung $u = 0$. Dadurch würde sich ein Kompromiss aus der Höhe der Stellgröße und der Regelabweichung einstellen.

Da aber immer eine gewisse Stellgröße benötigt wird, um eine Regelgröße zu erreichen, führt dies zu bleibenden Regelabweichungen und so zu nicht stationär genauen Reglern.

Die neue Stellgröße lautet dann:

$$u(k) = u(k-1) + \Delta u(k)$$

4.1 Einführung

Vorteile prädiktive Regelung (allgemein)

Vorrausschauende Regelung

- Durch Prädiktion kann der Regler auf zukünftige Führungsgrößenänderungen vorbereitet werden. Keine Regelabweichung für Reglereingriff nötig.
- Der Regler kann auch bei Regelabweichung $e = 0$ aktiv werden.
- Totzeitbehaftete Systeme können ohne Einschränkung der Aggressivität des Reglers geregelt werden. Verzögerung durch das System wird im Regler abgebildet.

(Nichtlineare) Mehrgrößensysteme

- Einfache Handhabung verschiedener Regelgrößen möglich. Durch Gewichtung in Gütefunktion kann eine Abwägung zwischen Zielen der Mehrgrößenregelung umgesetzt werden.
- Auch nichtlineare Mehrgrößensysteme gut regelbar. Formulierung der Reglerziele ändert sich im Vergleich zu linearen Systemen nicht.

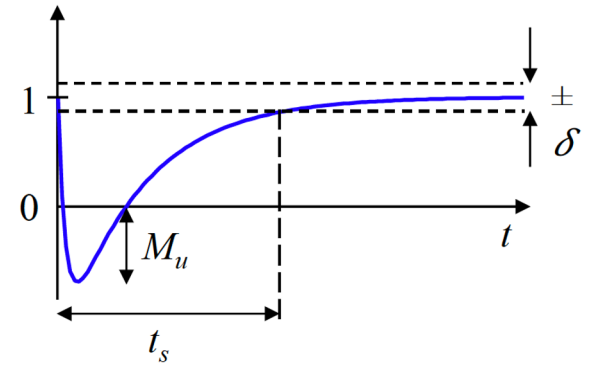
4.1 Einführung

Vorteile prädiktive Regelung (Details)

Prozesse, die ebenfalls gut geregelt werden können:

- *Nicht phasenminimale* Prozesse (instabile Nullstellen)
- *Instabile* Prozesse im offenen Regelkreis
- Prozesse mit *schwach gedämpften Polen* (schwingend)
- Prozesse mit *variabler/veränderlicher Totzeit*
- Prozesse *unbekannter Ordnung*

Sprungantwort nicht phasenminimales System



→ Polvorgabe- und LQ-Regler sind nicht geeignet, da hier Pol- und Nullstellen nicht kompensiert werden können

Begrenzungen/Limits

- Können durch die Modifikation der Gütefunktion direkt im Regler berücksichtigt werden. Ist sowohl für Stellgrößen als auch Regelgrößen möglich.

4.1 Einführung

Nachteile prädiktive Regelung

Rechenaufwand

- Optimierung ist Teil des Regelalgorithmus und muss in jedem Zeitschritt durchgeführt werden. Je nach System und Anforderungen (Nichtlinearität, Begrenzungen) kann das zu hohem Rechenaufwand führen.

Systemmodell

- Ein Modell des Systems wird für die Regelung benötigt. Die Modellgüte ist maßgeblich für die Regelgüte verantwortlich.
- Komplexe Modelle vergrößern den Rechenaufwand der Optimierung. Je komplexer der Prozess ist, desto komplexere Modelle werden benötigt. Ein Kompromiss muss gefunden werden.

4.2 Lineare Prädiktive Regelung

Algorithmen für die lineare prädiktive Regelung

- Dynamic Matrix Control (DMC)
- Model Algorithmic Control (MAC)
- Generalized Predicitve Control (GPC)
- State Space based MPC

4.2 Lineare Prädiktive Regelung

Lineare Prozessmodelle

Zur Prädiktion der zukünftigen Regelgrößen werden lineare Modelle verwendet. Unabhängig der Modelle ergibt sich daraus das Optimierungsproblem:

$$\min_{u(k),\cdots,u(k+N_u-1)} \sum_{i=1}^{N_\mathrm{p}} \left(w(k+i) - \hat{y}(k+i \mid k)\right)^2 + \lambda \sum_{i=0}^{N_\mathrm{u}-1} \Delta u(k+i)^2$$

Die Prädiktion der Regelgrößen $\hat{y}(k+i \mid k)$ wird je nach verwendetem Model unterschiedlich berechnet. Die Tabelle zeigt an, welche Modelltypen in den unterschiedlichen MPC Arten genutzt werden.

	DMC Dynamic Matrix Control	MAC Model Algorithmic Control	GPC Generalized Predictive Control	State Space based MPC
Prozess-modell	Sprungantwort FSR Modelle	Impulsantwort FIR Modelle	Übertragungsfunktionen CARIMA Modelle	Zustandsraum Zustandsraum Modelle
Modell Parameter	Sprungantworts-koeffizienten	Impulsantworts-koeffizienten	Übertragungsfunktion + Störübertragungsfunktion	Zustandsraummatrizen

4.2 Lineare Prädiktive Regelung

$$\frac{1}{1-z^{-1}} = \frac{1}{\Delta} : \text{Integrator}$$

Beziehungen zwischen Impuls- und Sprungantwort

Erinnerung: Im *Zeitkontinuierlichen* galten zwischen der Impulsantwort *g(t)* und der Sprungantwort *h(t)* die Beziehungen:

$$g(t) = \frac{d}{dt}h(t) \qquad h(t) = \int_0^t g(\tau)d\tau$$

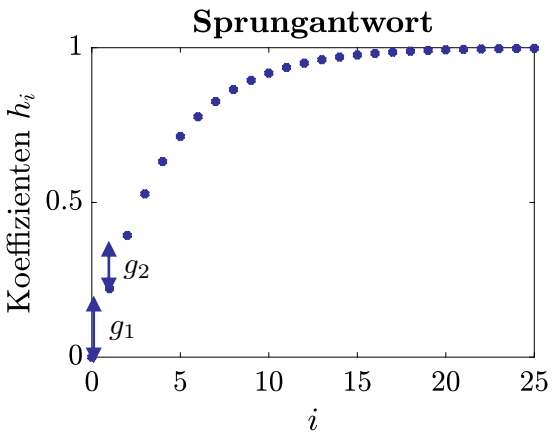
Aus den zuvor berechneten zeitdiskreten Impuls- und Sprungfolgen ergeben sich im *Zeitdiskreten* vergleichbare Zusammenhänge:

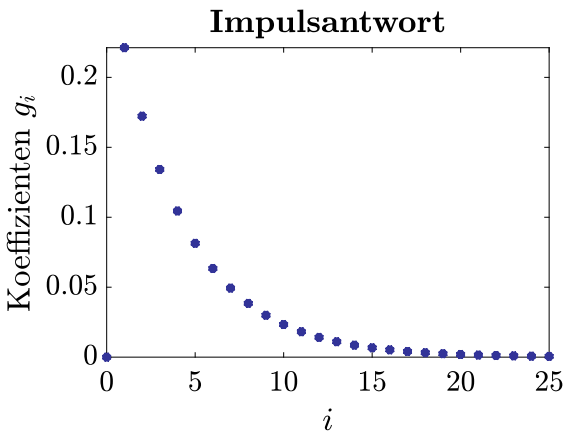
$$g_k = h_k - h_{k-1} \qquad h_k = \sum_{i=0}^{k} g_{k-i}$$

$$g(k) = h(k) - h(k-1) \qquad h(k) = \sum_{i=0}^{k} g(k-i)$$

Für den Modellausgang linearer Modelle gilt:

$$\begin{aligned}\hat{y}(k+1) &= \sum_{j=1}^{N} g_j u(k+1-j) \\ &= \sum_{j=1}^{N} \Delta h_j u(k+1-j) \\ &= \sum_{j=1}^{N} h_j \Delta u(k+1-j)\end{aligned}$$

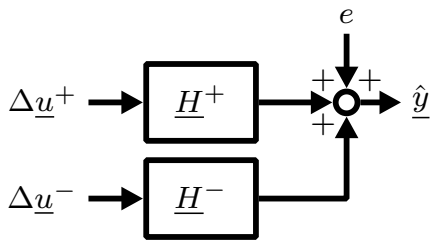




4.2 Lineare Prädiktive Regelung

N: Anzahl der Koeffizienten des Sprungantwort Modells

DMC – Dynamic Matrix Control



- Basiert auf Finite Step Response (FSR) Modellen:

$$\hat{y}(k+i \mid k) = \sum_{j=1}^{N-1} h_j \Delta u(k+i-j) + h_N u(k+i-N)$$

Matrix-Notation mit Korrektur der Vorhersage und für $N_p = N_u$:

$$\underline{\hat{y}} = \begin{bmatrix} \hat{y}(k+1 \mid k) \\ \vdots \\ \hat{y}\,(k+N_p \mid k) \end{bmatrix} = \underline{H}^- \underline{\Delta u}^- + \underline{H}^+ \underline{\Delta u}^+ + \underline{1}\,(y(k) - \hat{y}(k|k-1))$$

Vergangenheit — Zukunft Wird optimiert — Modell Fehler e

$$\underbrace{\begin{bmatrix} h_N & \cdots & \cdots & \cdots & \cdots & h_2 \\ h_N & h_N & & & & h_3 \\ \vdots & \ddots & \ddots & & & \vdots \\ h_N & \cdots & h_N & h_N & \cdots & h_{N_p+1} \end{bmatrix}}_{\underline{H}^-} \quad \underbrace{\begin{bmatrix} h_1 & 0 & \cdots & 0 \\ h_2 & h_1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ h_{N_p} & \cdots & h_2 & h_1 \end{bmatrix}}_{\underline{H}^+} \cdot \begin{bmatrix} u(k-N+1) \\ u(k-N+2) - u(k-N+1) \\ \vdots \\ u(k-1) - u(k-2) \\ \hline u(k) - u(k-1) \\ \vdots \\ u\,(k+N_p-1) - u\,(k+N_p-2) \end{bmatrix}$$

$\Delta \underline{u}^-$ Vergangenheit

$\Delta \underline{u}^+$ Zukunft

4.2 Lineare Prädiktive Regelung

DMC – Dynamic Matrix Control

Da in der Modellgleichung bereits Δu genutzt wird, wird auch hier Δu optimiert:

$$\min_{\Delta u(k),\cdots,\Delta u(k+N_u-1)} \sum_{i=1}^{N_p} \left(w(k+i) - \hat{y}(k+i \mid k)\right)^2 + \lambda \sum_{i=0}^{N_u-1} \Delta u(k+i)^2$$

Ausgeschrieben in Matrizen-Notation ergibt sich:

$$\min_{\Delta \underline{u}^+} \left(\underline{w} - \underline{H}^- \Delta \underline{u}^- - \underline{H}^+ \Delta \underline{u}^+ - \underline{1}(y(k) - \hat{y}(k|k-1))\right)^2 + \lambda \Delta \underline{u}^{+2}$$

Dies entspricht einer *Quadratischen Verlustfunktion*. Diese kann mit den Optimierungsalgorithmen aus Kapitel 3 gelöst werden.

Zur Vereinfachung der Formel wurde folgende Notation eingeführt:

$$\underline{x}^2 = \underline{x}^T \cdot \underline{x}$$

4.2 Lineare Prädiktive Regelung

DMC – Dynamic Matrix Control

Da in der Modellgleichung bereits Δu genutzt wird, wird auch hier Δu optimiert:

$$\min_{\Delta u(k),\cdots,\Delta u(k+N_u-1)} \sum_{i=1}^{N_p} \left(w(k+i) - \hat{y}(k+i \mid k)\right)^2 + \lambda \sum_{i=0}^{N_u-1} \Delta u(k+i)^2$$

Ohne Beschränkungen kann mithilfe des Least Squares (LS) Algorithmus inklusive Strafterm (Ridge Regression) die optimale Stellgrößenfolge analytisch bestimmt werden:

$$\Delta \underline{u}^+ = \underbrace{\left(\underbrace{\underline{H}^{+T}}_{N_p \times N_p} \underbrace{\underline{H}^{+}}_{N_p \times N_p} + \lambda \underbrace{\underline{I}}_{N_p \times N_p} \right)^{-1}}_{N_p \times N_p} \cdot \underbrace{\underline{H}^{+T}}_{N_p \times N_p} \underbrace{\left(\underbrace{\underline{w}}_{N_p \times 1} - \underbrace{\underline{H}^{-}}_{N_p \times (N-1)} \underbrace{\Delta \underline{u}^{-}}_{(N-1) \times 1} - \underbrace{\underline{1}(y(k) - \hat{y}(k|k-1))}_{N_p \times 1} \right)}_{N_p \times 1}}_{N_p \times 1}$$

Die nächste optimale Stellgröße berechnet sich dann aus der Summe der letzten Stellgröße und dem ersten Wert des Vektors $\Delta \underline{u}^+$

$$u(k) = u(k-1) + \Delta u(k)$$

4.2 Lineare Prädiktive Regelung

DMC – Dynamic Matrix Control

Der Fall $N_p > N_u$ tritt häufig auf, da hier ein langer Prädiktionshorizont gewählt werden kann. Um die Komplexität zum Lösen der Verlustfunktion gering zu halten wird $N_u < N_p$ gewählt.

$N_p = N_u$	$N_p > N_u$
$\underline{\hat{y}} = \underbrace{\underline{H}^+}_{N_p \times N_p} \cdot \underbrace{\Delta\underline{u}^+}_{N_p \times 1} + \underbrace{\underline{H}^-}_{N_p \times (N-1)} \cdot \underbrace{\Delta\underline{u}^-}_{(N-1)\times 1}$	$\underline{\hat{y}} = \underbrace{\underline{H}^+}_{N_p \times N_p} \cdot \underbrace{\underline{\tilde{H}}}_{N_p \times N_u} \cdot \underbrace{\Delta\underline{u}^+}_{N_u \times 1} + \underbrace{\underline{H}^-}_{N_p \times (N-1)} \cdot \underbrace{\Delta\underline{u}^-}_{(N-1)\times 1}$

Es wird angenommen, dass u für $N_u < k < N_p$ konstant bleibt, daher gilt: $\Delta u^+(i \geq N_u) = 0$

$$\underline{\tilde{H}} = \begin{bmatrix} \underline{I}_{N_u \times N_u} \\ \underline{0}_{(N_p - N_u)\times N_u} \end{bmatrix}$$

Beispiel: $N_p = 5, N_u = 3$

$$\underline{\tilde{H}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \underline{\tilde{H}}\,\Delta\underline{u}^+ = \begin{bmatrix} \Delta u(k) \\ \Delta u(k+1) \\ \Delta u(k+2) \\ 0 \\ 0 \end{bmatrix}$$

4.2 Lineare Prädiktive Regelung

DMC – Dynamic Matrix Control

Die Dimension der Inverse der vorher $N_p \times N_p$ Matrix

$$\Delta \underline{u}^{+} = \underbrace{\underbrace{\left(\underbrace{\underline{H}^{+T}}_{N_p \times N_p} \underbrace{\underline{H}^{+}}_{N_p \times N_p} + \lambda \underbrace{\underline{I}}_{N_p \times N_p} \right)^{-1}}_{N_p \times N_p} \cdot \underbrace{\underline{H}^{+T}}_{N_p \times N_p} \left(\underbrace{\underline{w}}_{N_p \times 1} - \underbrace{\underline{H}^{-}}_{N_p \times (N-1)} \underbrace{\Delta \underline{u}^{-}}_{(N-1) \times 1} - \underbrace{\underline{1}(y(k) - \hat{y}(k|k-1))}_{N_p \times 1} \right)}_{N_p \times 1}}_{N_p \times 1}$$

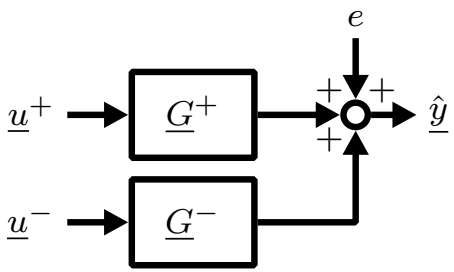
wird zu $N_u \times N_u$:

$$\Delta \underline{u}^{+} = \underbrace{\underbrace{\left(\underbrace{\left(\underline{H}^{+} \tilde{\underline{H}} \right)^{T}}_{N_u \times N_p} \underbrace{\left(\underline{H}^{+} \tilde{\underline{H}} \right)}_{N_p \times N_u} + \lambda \underbrace{\underline{I}}_{N_u \times N_u} \right)^{-1}}_{N_u \times N_u} \cdot \underbrace{\left(\underline{H}^{+} \tilde{\underline{H}} \right)^{T}}_{N_u \times N_p} \left(\underbrace{\underline{w}}_{N_p \times 1} - \underbrace{\underline{H}^{-}}_{N_p \times (N-1)} \underbrace{\Delta \underline{u}^{-}}_{(N-1) \times 1} - \underbrace{\underline{1}(y(k) - \hat{y}(k|k-1))}_{N_p \times 1} \right)}_{N_p \times 1}}_{N_u \times 1}$$

4.2 Lineare Prädiktive Regelung

N: Anzahl der Koeffizienten des Impulsantwort Modells

MAC – Model Algorithmic Control



- Basiert auf Finite Impulse Response (FIR) Modellen:

$$\hat{y}(k+i \mid k) = \sum_{j=1}^{N} g_j u(k+i-j \mid k)$$

Matrix-Notation mit Korrektur der Vorhersage für $N_p = N_u$:

$$\underline{\hat{y}} = \begin{bmatrix} \hat{y}(k+1 \mid k) \\ \vdots \\ \hat{y}\,(k+N_p \mid k) \end{bmatrix} = \underline{G}^{-}\,\underbrace{\underline{u}^{-}}_{\text{Vergangenheit}} + \underline{G}^{+}\,\underbrace{\underline{u}^{+}}_{\substack{\text{Zukunft} \\ \text{Wird optimiert}}} + \underline{1}\,\underbrace{(y(k) - \hat{y}(k|k-1))}_{\text{Modell Fehler } e}$$

mit

$$\underbrace{\begin{bmatrix} g_N & \cdots & \cdots & \cdots & \cdots & g_2 \\ 0 & g_N & & & & g_3 \\ \vdots & \ddots & \ddots & & & \vdots \\ 0 & \cdots & 0 & g_N & \cdots & g_{N_p+1} \end{bmatrix}}_{\underline{G}^{-}} \quad \underbrace{\begin{bmatrix} g_1 & 0 & \cdots & 0 \\ g_2 & g_1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ g_{N_p} & \cdots & g_2 & g_1 \end{bmatrix}}_{\underline{G}^{+}} \cdot \begin{bmatrix} u(k-N+1) \\ \vdots \\ u(k-1) \\ \hline u(k) \\ \vdots \\ u\,(k+N_p-1) \end{bmatrix} \begin{matrix} \} \underline{u}^{-} \text{ Vergangenheit} \\ \\ \} \underline{u}^{+} \text{ Zukunft} \end{matrix}$$

4.2 Lineare Prädiktive Regelung

MAC – Model Algorithmic Control

Im MAC Algorithmus werden die Stellgrößen $u(k)$ optimiert und nicht die Stellgrößenänderungen $\Delta u(k)$. Daher muss noch eine Umrechnung erfolgen, um die Stellgrößenänderungen zu bestrafen. Durch die Bestrafung der Stellgrößen würde sich ein Kompromiss aus der Höhe der Stellgröße und der Regelgröße einstellen. Dies führt zu einem nicht stationären Regler.

$$\Delta \underline{u}^{+} = \begin{bmatrix} u(k) & - & u(k-1) \\ u(k+1) & - & u(k) \\ & \vdots & \\ u(k+N_p-1) & - & u(k+N_p-2) \end{bmatrix} = \underline{T}\,\underline{u}^{+} - \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} u\,(k-1)$$

$$\underline{T} = \begin{bmatrix} 1 & 0 & \cdots & \cdots & 0 \\ -1 & 1 & \ddots & \ddots & \vdots \\ 0 & -1 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & -1 & 1 \end{bmatrix}, \quad \underline{u}^{+} = \begin{bmatrix} u(k+1) \\ u(k+2) \\ u(k+3) \\ \vdots \\ u\,(k+N_p) \end{bmatrix}$$

4.2 Lineare Prädiktive Regelung

MAC – Model Algorithmic Control

Da in der Modellgleichung u genutzt wird, werden mithilfe von $\underline{T}$ die Inkrementellen Eingänge Δu in der Verlustfunktion bestraft, jedoch nach u optimiert:

$$\min_{u(k),\cdots,u(k+N_u-1)} \sum_{i=1}^{N_p} \left(w(k+i) - \hat{y}(k+i \mid k)\right)^2 + \lambda \sum_{i=0}^{N_u-1} \Delta u(k+i)^2$$

Ausgeschrieben in Matrizen-Notation ergibt sich:

$$\min_{\underline{u}^+} \left(\underline{w} - \underline{H}^- \underline{u}^- - \underline{H}^+ \underline{u}^+ - \underline{1}(y(k) - \hat{y}(k|k-1))\right)^2 + \lambda \left(\underline{T}\,\underline{u}^+ - \begin{bmatrix} 1 \\ \underline{0} \end{bmatrix} u(k-1)\right)^2$$

Dies entspricht einer *Quadratischen Verlustfunktion*. Diese kann mit den Optimierungsalgorithmen aus Kapitel 3 gelöst werden.

Zur Vereinfachung der Formel wurde folgende Notation eingeführt:

$$\underline{x}^2 = \underline{x}^T \cdot \underline{x}$$

4.2 Lineare Prädiktive Regelung

MAC – Model Algorithmic Control

Der Fall $N_p > N_u$ tritt häufig auf, da hier ein langer Prädiktionshorizont gewählt werden kann. Um die Komplexität zum Lösen der Verlustfunktion gering zu halten wird $N_u < N_p$ gewählt.

$N_p = N_u$	$N_p > N_u$
$\underline{\hat{y}} = \underbrace{\underline{G}^+}_{N_p \times N_p} \cdot \underbrace{\underline{u}^+}_{N_p \times 1} + \underbrace{\underline{G}^-}_{N_p \times (N-1)} \cdot \underbrace{\underline{u}^-}_{(N-1) \times 1}$	$\underline{\hat{y}} = \underbrace{\underline{G}^+}_{N_p \times N_p} \cdot \underbrace{\tilde{\underline{G}}}_{N_p \times N_u} \cdot \underbrace{\underline{u}^+}_{N_u \times 1} + \underbrace{\underline{G}^-}_{N_p \times (N-1)} \cdot \underbrace{\underline{u}^-}_{(N-1) \times 1}$

Es wird angenommen, dass u nach N_u konstant bleibt, daher gilt: $u^+(i \geq N_u) = u^+(N_u)$

$$\tilde{\underline{G}} = \begin{bmatrix} \underline{I}_{N_u \times N_u} \\ \underline{0}_{(N_p - N_u) \times (N_u - 1)} \quad \underline{1}_{(N_p - N_u) \times 1} \end{bmatrix}$$

Beispiel: $N_p = 5, N_u = 3$

$$\tilde{\underline{G}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}, \tilde{\underline{G}}\,\underline{u}^+ = \begin{bmatrix} u(k+1) \\ u(k+2) \\ u(k+3) \\ u(k+3) \\ u(k+3) \end{bmatrix}$$

4.2 Lineare Prädiktive Regelung

MAC – Model Algorithmic Control

Da in der Modellgleichung u genutzt wird, werden mithilfe von $\underline{T}$ die Inkrementellen Eingänge Δu in der Verlustfunktion bestraft, jedoch nach u optimiert:

$$\min_{u(k),\cdots,u(k+N_u-1)} \sum_{i=1}^{N_p} \left(w(k+i) - \hat{y}(k+i \mid k)\right)^2 + \lambda \sum_{i=0}^{N_u-1} \Delta u(k+i)^2$$

Ohne Beschränkungen kann mithilfe des Least Squares (LS) Algorithmus die optimale Stellgrößenfolge analytisch bestimmt werden:

$$\underline{u}^+ = \underbrace{\left(\underbrace{\left(\underline{G}^+\tilde{\underline{G}}\right)^T}_{N_u \times N_p} \underbrace{\left(\underline{G}^+\tilde{\underline{G}}\right)}_{N_p \times N_u} + \lambda \underbrace{\underline{T}^T\underline{T}}_{N_u \times N_u} \right)^{-1}}_{N_u \times N_u} \underbrace{\left(\underbrace{\left(\underline{G}^+\tilde{\underline{G}}\right)^T}_{N_u \times N_p} \left(\underbrace{\underline{w}}_{N_p \times 1} - \underbrace{\underline{G}^-}_{N_p \times (N-1)} \underbrace{\underline{u}^-}_{(N-1)\times 1} - \underbrace{\underline{1}(y(k) - \hat{y}(k|k-1))}_{N_p \times 1} \right) + \lambda \underline{T}^T \begin{pmatrix} 1 \\ \underline{0} \end{pmatrix} u(k-1) \right)}_{N_u \times 1}$$

$$N_u \times 1$$

Die nächste optimale Stellgröße entspricht dem ersten Wert des Vektors $\underline{u}^+$.

4.2 Lineare Prädiktive Regelung

$$\frac{1}{1-z^{-1}} = \frac{1}{\Delta} : \text{Integrator}$$

CARIMA (Controlled Auto-Regressive Integrated Moving Average) Model

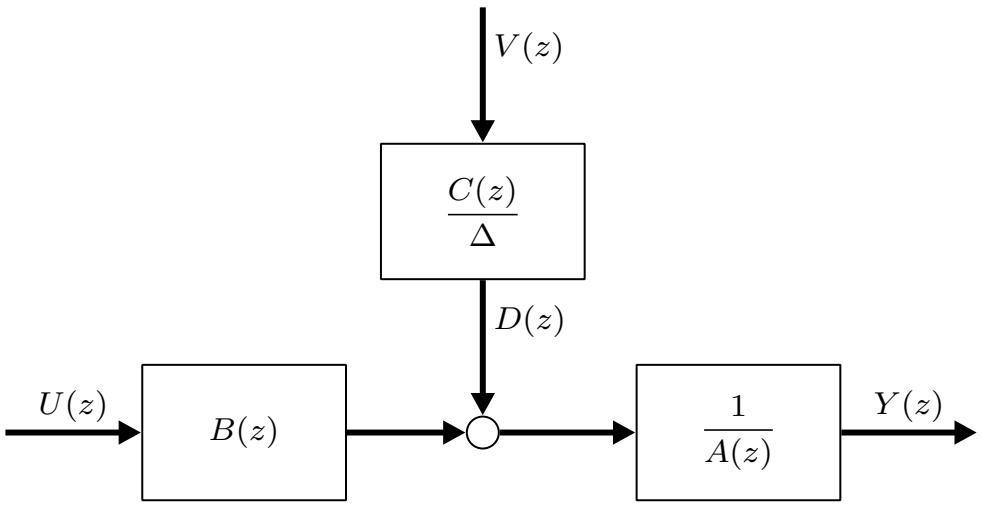
Modellstruktur besteht aus vergangen Eingangs- sowie Ausgangsgrößen. Die Nutzung des CARIMA Models ermöglicht eine Bias-freie Prädiktion im stationären Zustand. Dabei werden sowohl vergangene Eingangs- als auch Ausgangswerte genutzt.

Die allgemeine Übertragungsfunktion lautet:

$$A(z)Y(z) = B(z)U(z) + \underbrace{\frac{C(z)}{\Delta}V(z)}_{D(z)}$$

mit

$$A(z) = 1 + a_1 z^{-1} + \cdots + a_n z^{-n}$$
$$B(z) = b_1 z^{-1} + \cdots + b_m z^{-m}$$



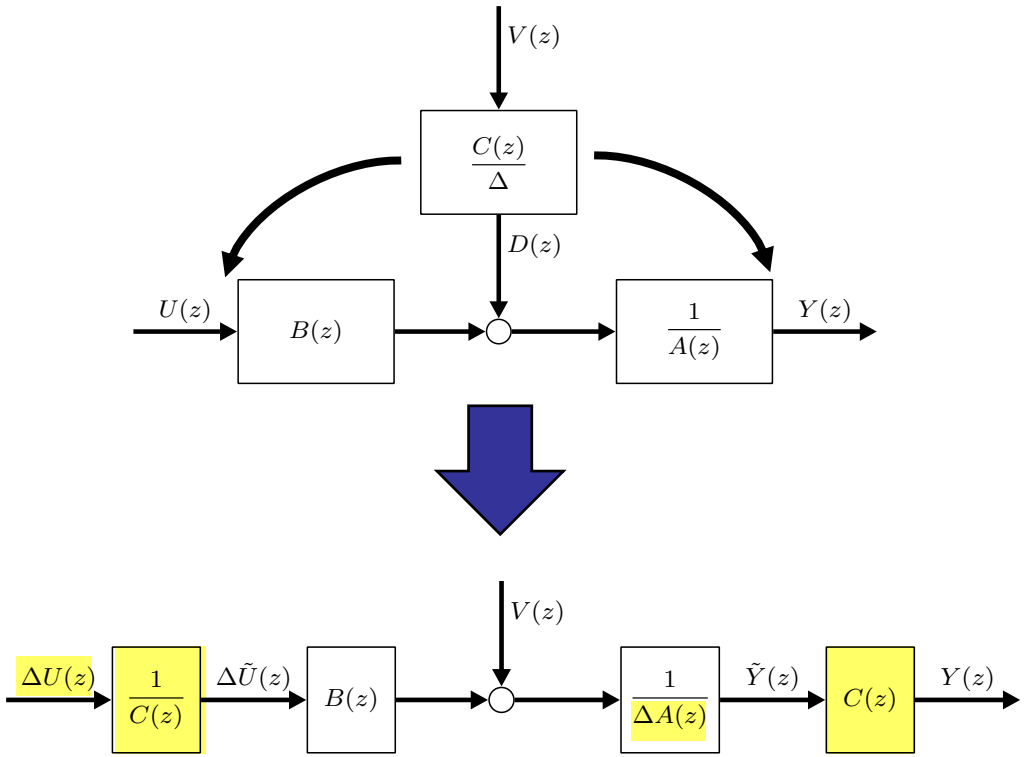
4.2 Lineare Prädiktive Regelung

$$\frac{1}{1-z^{-1}} = \frac{1}{\Delta} : \text{Integrator}$$

CARIMA (Controlled Auto-Regressive Integrated Moving Average) Model

Die Idee durch Verwendung dieser Modellstruktur besteht darin, dass durch geschicktes Umformen des Blockschaltbildes eine Abhängigkeit des Ausgangs $Y(z)$ mit dem inkrementellen Eingang $\Delta U(z)$ entsteht. Das Polynom $C(z)$ dient als Filter der Eingangsgröße. Um den ungefilterten Modellausgang zu berechnen, muss diese Filterung im Anschluss wieder rückgängig gemacht werden.

Die Rückkopplung der Ausgänge $\frac{1}{\Delta A(z)}$ wird nur mit gefilterten gemessenen Ausgangsgrößen durchgeführt. (Tiefpassfilterung der gemessenen Größen – Teilweise Eliminierung des Rauschens)



4.2 Lineare Prädiktive Regelung

$$\frac{1}{1-z^{-1}} = \frac{1}{\Delta} : \text{Integrator}$$

CARIMA (Controlled Auto-Regressive Integrated Moving Average) Model

Durch die Multiplikation mit Δ auf beiden Seiten, kann ein Zusammenhang der absoluten Ausgangsgröße mit inkrementellen Eingangsgröße realisiert werden. Zur Vereinfachung wird vorerst angenommen, dass $\boldsymbol{C(z) = 1}$ gilt.

$$A(z)Y(z) = B(z)U(z) + \frac{V(z)}{\Delta} \xrightarrow{\cdot\Delta} \underbrace{(A(z)\Delta)}_{\tilde{A}(z)} Y(z) = B(z)\,(\Delta U(z)) + V(z)$$

Da $V(z)$ eine mittelwertfreie Störung ist ergibt sich die Differenzgleichung durch Einsetzten der Polynome $\tilde{A}(z)$ und $B(z)$:

$$y(k+1) + \tilde{a}_1 y(k) + \ldots + \tilde{a}_n y(k-n+1) = b_1 \Delta u(k) + \ldots + b_m \Delta u(k-m+1)$$

Der Ein-Schritt Prädiktor hängt daher nur von vergangenen Ein- und Ausgangswerten ab. Durch Umstellen der Gleichungen nach $y(k+1)$ ergibt sich

$$y(k+1) = b_1 \Delta u(k) + \ldots + b_m \Delta u(k-m+1) - \tilde{a}_1 y(k) - \ldots - \tilde{a}_n y(k-n+1)$$

Achtung: Durch Multiplikation mit $\cdot\,\Delta$ werden zwar die inkrementellen Eingänge genutzt, aber $b_1, \ldots, b_m$ entsprechen den Impulsantwortskoeffizienten

4.2 Lineare Prädiktive Regelung

CARIMA (Controlled Auto-Regressive Integrated Moving Average) Model

Für Prädiktionen mehrerer Zeitschritte können die Gleichungen auch in Matrix/Vektor Notation gebracht werden:

$$
\begin{aligned}
y(k+1) =& b_1\Delta u(k) && + \ldots + b_m\Delta u(k-m+1) && - \tilde{a}_1 y(k) && - \ldots - \tilde{a}_n y(k-n+1)\\
y(k+2) =& b_1\Delta u(k+1) && + \ldots + b_m\Delta u(k-m+2) && - \tilde{a}_1 y(k+1) && - \ldots - \tilde{a}_n y(k-n+2)\\
y(k+3) =& b_1\Delta u(k+2) && + \ldots + b_m\Delta u(k-m+3) && - \tilde{a}_1 y(k+2) && - \ldots - \tilde{a}_n y(k-n+3)\\
y(k+4) =& b_1\Delta u(k+3) && + \ldots + b_m\Delta u(k-m+4) && - \tilde{a}_1 y(k+3) && - \ldots - \tilde{a}_n y(k-n+4)
\end{aligned}
$$

Dabei werden die Matrizen ähnlich zu der DMC und MAC aufgestellt, lediglich die vergangenen Ausgangswerte gehen zusätzlich mit ein:

$$\underline{J}^+\hat{\underline{y}}^+ = \underline{G}^-\underbrace{\Delta\underline{u}^-}_{\text{Vergangenheit}} + \underline{G}^+\underbrace{\Delta\underline{u}^+}_{\substack{\text{Zukunft}\\ \text{Wird optimiert}}} - \underline{J}^-\underbrace{\underline{y}^-}_{\text{Vergangenheit}}$$

Zukunft

$$\hat{\underline{y}}^+ = \underbrace{\left(\underline{J}^+\right)^{-1}\underline{G}^-}_{\tilde{\underline{G}}^-}\Delta\underline{u}^- + \underbrace{\left(\underline{J}^+\right)^{-1}\underline{G}^+}_{\tilde{\underline{G}}^+}\Delta\underline{u}^+ - \underbrace{\left(\underline{J}^+\right)^{-1}\underline{J}^-}_{\tilde{\underline{J}}^-}\underline{y}^-$$

Vergangenheit

Wie die Matrizen aufgestellt werden, siehe nächste Folie

4.2 Lineare Prädiktive Regelung

CARIMA (Controlled Auto-Regressive Integrated Moving Average) Model

Die Matrizen werden identisch wie bei DMC/MAC aufgebaut:

$$\underbrace{\begin{bmatrix} b_m & \cdots & \cdots & \cdots & \cdots & b_2 \\ 0 & b_m & & & & b_3 \\ \vdots & \ddots & \ddots & & & \vdots \\ 0 & \cdots & 0 & b_m & \cdots & b_{N_p+1} \end{bmatrix}}_{\underline{G}^-} \Bigg| \underbrace{\begin{bmatrix} b_1 & 0 & \cdots & 0 \\ b_2 & b_1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ b_{N_p} & \cdots & b_2 & b_1 \end{bmatrix}}_{\underline{G}^+} \cdot \left[\begin{array}{c} \Delta u(k-m+1) \\ \vdots \\ \Delta u(k-1) \\ \hline \Delta u(k) \\ \vdots \\ \Delta u\left(k+N_p-1\right) \end{array}\right] \begin{matrix} \} \Delta \underline{u}^- \\ \\ \} \Delta \underline{u}^+ \end{matrix}$$

$$\underbrace{\begin{bmatrix} \tilde{a}_n & \cdots & \cdots & \cdots & \cdots & \tilde{a}_1 \\ 0 & \tilde{a}_n & & & & \tilde{a}_2 \\ \vdots & \ddots & \ddots & & & \vdots \\ 0 & \cdots & 0 & \tilde{a}_n & \cdots & \tilde{a}_{N_p} \end{bmatrix}}_{\underline{J}^-} \Bigg| \underbrace{\begin{bmatrix} 1 & 0 & \cdots & 0 \\ \tilde{a}_1 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ \tilde{a}_{N_p-1} & \cdots & \tilde{a}_1 & 1 \end{bmatrix}}_{\underline{J}^+} \cdot \left[\begin{array}{c} y(k-n+1) \\ \vdots \\ y(k) \\ \hline \hat{y}(k+1) \\ \vdots \\ \hat{y}\left(k+N_p\right) \end{array}\right] \begin{matrix} \} \underline{y}^- \\ \\ \} \underline{\hat{y}}^+ \end{matrix}$$

4.2 Lineare Prädiktive Regelung

GPC – Generalized Predictive Control

Basiert auf Controlled Autoregressive Integrated Moving Average (CARIMA) Modellen:

$$\underline{\hat{y}}^{+} = \underline{\tilde{G}}^{+}\,\Delta\underline{u}^{+} + \underline{\tilde{G}}^{-}\,\Delta\underline{u}^{-} + \underline{\tilde{J}}^{-}\,\underline{y}^{-}$$

Durch Minimierung der Verlustfunktion

$$\min_{\Delta u(k+1),\cdots,\Delta u(k+N_u)} \sum_{i=1}^{N_p} \left(w(k+i) - \hat{y}(k+i \mid k)\right)^2 + \lambda \sum_{i=1}^{N_u} \Delta u(k+i)^2$$

ergibt sich folgende optimale Lösung, falls keine Nebenbedingungen betrachtet werden:

$$\Delta\underline{u}^{+} = \left(\underline{\tilde{G}}^{+T}\,\underline{\tilde{G}}^{+} + \lambda\underline{I}\right)^{-1} \cdot \underline{\tilde{G}}^{+T}\left(\underline{w} - \underline{\tilde{G}}^{-}\,\Delta\underline{u}^{-} - \underline{\tilde{J}}^{-}\,\underline{y}^{-}\right)$$

Enthält neben den vergangenen Eingängen nun auch vergangene Ausgänge

DMC/MAC hängen jeweils nur von den vergangenen Eingängen ab. Bei GPC werden die vergangenen Ausgänge rückgekoppelt.

4.2 Lineare Prädiktive Regelung

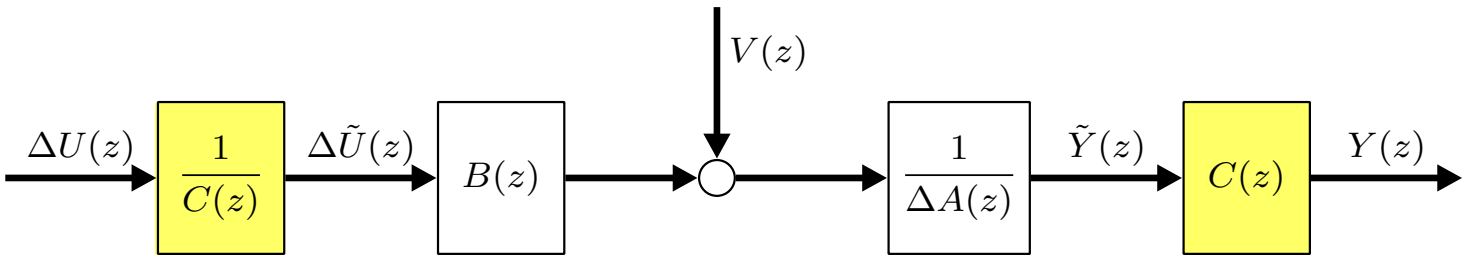
GPC – Generalized Predictive Control

Filter $C(z)$

Sowohl die Eingangs- als auch die Ausgangsgröße wird nun mit dem Polynom $\frac{1}{C(z)}$ gefiltert.

$$A(z)Y(z) = B(z)U(z) + \frac{C(z)}{\Delta}V(z) \overset{\cdot\frac{\Delta}{C(z)}}{\Longrightarrow} \underbrace{(A(z)\Delta)}_{\tilde{A}(z)}\underbrace{\frac{Y(z)}{C(z)}}_{\tilde{Y}(z)} = B(z)\left(\underbrace{\frac{\Delta U(z)}{C(z)}}_{\Delta\tilde{U}(z)}\right) + V(z)$$

Hierdurch werden nicht die gemessenen, **verrauschten** Ausgangsgrößen zur Prädiktion, sondern gefilterte Ausgangsgrößen genutzt. Um dennoch die ungefilterten Ausgangsgrößen zu Prädizieren, und als Eingang auch ungefilterte Stellgrößen zu nutzen, muss diese Umrechnung im Anschluss wieder rückgängig gemacht werden.



4.2 Lineare Prädiktive Regelung

GPC – Generalized Predictive Control

Wahl des Filters $C(z)$

Typischerweise wird durch das Polynom $C(z)$ ein Tiefpassfilter realisiert. Durch Wahl eines Polynoms 1. Ordnung ergibt sich so ein Verzögerungsglied erster Ordnung. Dies entspricht einem einfachen *Tiefpassfilter*. Die Wahl eines Tiefpasses ist sehr effektiv und genügt in den meisten Fällen. Eine Filterung mit einem Polynom lässt sich auch in Matrix/Vektor Notation schreiben. Hierfür werden allerdings auch vergangene Werte der jeweiligen Größe benötigt:

$$\Delta\underline{\tilde{u}}^{+} = \frac{1}{C(z)}\Delta\underline{u}^{+}$$

$$\rightarrow \quad \Delta\underline{u}^{+} = C(z)\Delta\underline{\tilde{u}}^{+} = \underline{T}^{+}\Delta\underline{\tilde{u}}^{+} + \underline{T}^{-}\Delta\underline{\tilde{u}}^{-}$$

$$\underline{y}^{+} = C(z)\underline{\tilde{y}}^{+} = \underline{T}^{+}\underline{\tilde{y}}^{+} + \underline{T}^{-}\underline{\tilde{y}}^{-}$$

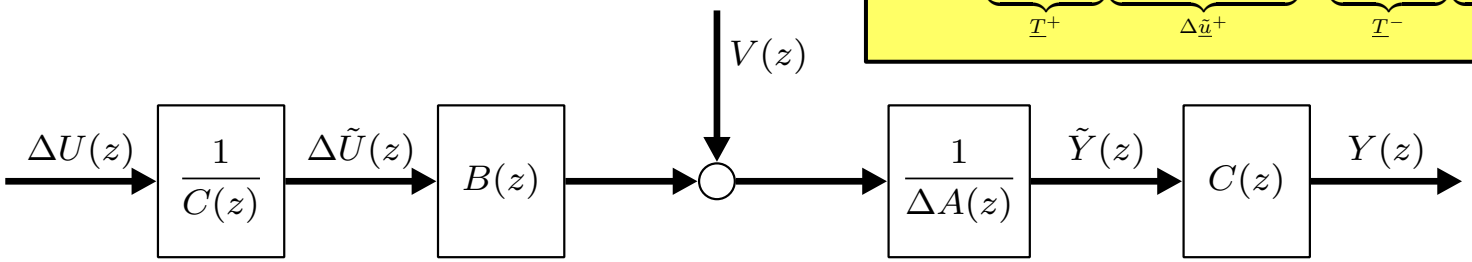
Beispiel:

$$C(z) = 1 + c_1 z^{-1}$$

$$\Delta\underline{u}^{+} = (1 + pz^{-1}) \begin{bmatrix} \Delta\tilde{u}(k+1) \\ \Delta\tilde{u}(k+2) \end{bmatrix}$$

$$= \begin{bmatrix} 1\cdot\Delta\tilde{u}(k+1) & + & p\cdot\Delta\tilde{u}(k) \\ 1\cdot\Delta\tilde{u}(k+2) & + & p\cdot\Delta\tilde{u}(k+1) \end{bmatrix}$$

$$= \underbrace{\begin{bmatrix} 1 & 0 \\ p & 1 \end{bmatrix}}_{\underline{T}^{+}} \underbrace{\begin{bmatrix} \Delta\tilde{u}(k+1) \\ \Delta\tilde{u}(k+2) \end{bmatrix}}_{\Delta\underline{\tilde{u}}^{+}} + \underbrace{\begin{bmatrix} 0 & p \\ 0 & 0 \end{bmatrix}}_{\underline{T}^{-}} \underbrace{\begin{bmatrix} \Delta\tilde{u}(k-1) \\ \Delta\tilde{u}(k) \end{bmatrix}}_{\Delta\underline{\tilde{u}}^{-}}$$



4.2 Lineare Prädiktive Regelung

GPC – Generalized Predictive Control

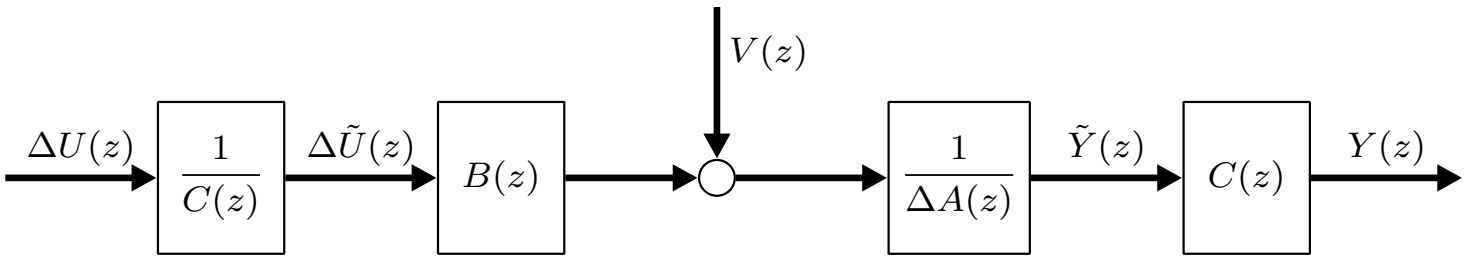
Um in der Verlustfunktion die ungefilterten Größen zu nutzen, wird die gefilterte Gleichung entsprechend umgeschrieben. (Einsetzen der Gleichung in vorheriger Folie)

$$\underline{\tilde{y}}^{+} = \underline{\tilde{G}}^{+}\,\Delta\underline{\tilde{u}}^{+} + \underline{\tilde{G}}^{-}\,\Delta\underline{\tilde{u}}^{-} + \underline{\tilde{J}}^{-}\,\underline{\tilde{y}}^{-}$$

$$\underline{y}^{+} = \underbrace{\underline{T}^{+}\underline{\tilde{G}}^{+}\left(\underline{T}^{+}\right)^{-1}}_{\underline{\tilde{G}}_2^{+}}\Delta\underline{u}^{+} + \underbrace{\left(\underline{T}^{+}\underline{\tilde{G}}^{-} - \underline{T}^{+}\underline{\tilde{G}}^{+}\left(\underline{T}^{+}\right)^{-1}\underline{T}^{-}\right)}_{\underline{\tilde{G}}_2^{-}}\Delta\underline{\tilde{u}}^{-} + \underbrace{\left(\underline{T}^{+}\underline{\tilde{J}}^{-} + \underline{T}^{-}\right)}_{\underline{\tilde{J}}_2^{-}}\underline{\tilde{y}}^{-}$$

Die Prädiktionsgleichungen mit einem C-Filter beruhen auf *gefilterten Werten von vergangenen Daten*. Die analytische Lösung ohne Nebenbedingungen ergibt sich dementsprechend aus:

$$\Delta\underline{u}^{+} = \left(\underline{\tilde{G}}_2^{+T}\,\underline{\tilde{G}}_2^{+} + \lambda\underline{I}\right)^{-1} \cdot \underline{\tilde{G}}_2^{+T}\left(\underline{w} - \underline{\tilde{G}}_2^{-}\,\underline{u}^{-} - \underline{\tilde{J}}_2^{-}\,\underline{y}^{-}\right)$$



4.2 Lineare Prädiktive Regelung

GPC – Generalized Predictive Control

Durch die Nutzung des CARIMA Modells können die Matrizen ebenfalls wieder in Übertragungsfunktionen umgewandelt werden. Dadurch kann die Rauschempfindlichkeit und weitere Regelkreismerkmale genauer untersucht werden.

Die Nutzung der Matrix/Vektor Notation ist vor allem bei der Implementierung sehr praktisch aber nicht notwendig.

Zur Vorhersage von i-Schritten ($i = 1,2, \dots, N_p$) in die Zukunft kann auch die Diophantische Gleichung mit den Polynomen $E_i(z)$ und $F_i(z)$ und unbekannten Koeffizienten rekursiv gelöst werden:

$$1 = E_i(z)A(z)(1 - z^{-1}) + z^{-i}F_i(z)$$

Die Koeffizienten von $E_i(z)B(z)$ und $F_i(z)$ entsprechen den i-ten Zeilen der Matrizen:

$$\underline{y}^+ = \underbrace{\begin{bmatrix} \underline{\tilde{G}}^- & \underline{\tilde{G}}^+ \end{bmatrix}}_{E_i B} \begin{bmatrix} \Delta\underline{u}^- \\ \Delta\underline{u}^+ \end{bmatrix} + \underbrace{\underline{\tilde{J}}^-}_{F_i} \underline{y}^-$$

Durch die Annahme $A(z) = 1$ und eine geeignete Wahl von $B(z)$ lassen sich auch der DMC und MAC Algorithmus in die GPC Form bringen.

4.2 Lineare Prädiktive Regelung

GPC vs. DMC

Der GPC ist eine Verallgemeinerung vieler Prädiktiven Regelungen. Mit verschiedenen Annahmen für den GPC ergibt sich so der MAC oder der DMC.

Um den DMC zu erhalten wird Folgendes für den GPC angenommen:

- $A(z) = 1$ (keine interne Rückkopplung der Ausgänge)
- $C(z) = 1$

In den folgenden zwei Folien werden die Matrizen Gleichungen umgeformt und gezeigt, dass das Ergebnis der Prädiktionsgleichungen beider Ansätze identisch ist.

4.2 Lineare Prädiktive Regelung

GPC vs. DMC

GPC entspricht DMC, wenn:
- GPC $A(z) = 1, C(z) = 1$
- DMC mit Korrektur der Vorhersage

Umformen des GPC:

Mit $A(z) = 1$: $\tilde{A}(z) = \Delta = \underbrace{1}_{\tilde{a}_0} + \underbrace{-1}_{\tilde{a}_1} \cdot z^{-1}$

$$
\begin{aligned}
y(k+1) &= b_1 \Delta u(k) + \ldots + b_m \Delta u(k-m+1) - \tilde{a}_1 y(k) \\
y(k+2) &= b_1 \Delta u(k+1) + \ldots + b_m \Delta u(k-m+2) - \tilde{a}_1 y(k+1) \\
y(k+3) &= b_1 \Delta u(k+2) + \ldots + b_m \Delta u(k-m+3) - \tilde{a}_1 y(k+2)
\end{aligned}
$$

$$
\underbrace{\begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix}}_{\underline{J}^-}
\begin{bmatrix} y(k+1) \\ y(k+2) \\ y(k+3) \end{bmatrix}
= \underline{G}^- \Delta \underline{u}^- + \underline{G}^+ \Delta \underline{u}^+ +
\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} y(k)
$$

$$
\begin{bmatrix} y(k+1) \\ y(k+2) \\ y(k+3) \end{bmatrix}
= \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}}_{(\underline{J}^-)^{-1}}
\underbrace{\begin{bmatrix} g_4 & g_3 & g_2 \\ g_5 & g_4 & g_3 \\ g_6 & g_5 & g_4 \end{bmatrix}}_{\underline{G}^-} \Delta \underline{u}^-
+ \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}}_{(\underline{J}^-)^{-1}}
\underbrace{\begin{bmatrix} g_1 & 0 & 0 \\ g_2 & g_1 & 0 \\ g_3 & g_2 & g_1 \end{bmatrix}}_{\underline{G}^+} \Delta \underline{u}^+ +
\underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}}_{(\underline{J}^-)^{-1}}
\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} y(k)
$$

$$
\begin{bmatrix} y(k+1) \\ y(k+2) \\ y(k+3) \end{bmatrix}
= \underbrace{\begin{bmatrix} g_4 & g_3 & g_2 \\ g_5+g_4 & g_4+g_3 & g_3+g_2 \\ g_6+g_5+g_4 & g_5+g_4+g_3 & g_4+g_3+g_2 \end{bmatrix}}_{\tilde{\underline{G}}^- = \underline{H}_c^-} \Delta \underline{u}^-
+ \underbrace{\begin{bmatrix} g_1 & 0 & 0 \\ g_2+g_1 & g_1 & 0 \\ g_3+g_2+g_1 & g_2+g_1 & g_1 \end{bmatrix}}_{\tilde{\underline{G}}^+ = \underline{H}^+} \Delta \underline{u}^+ +
\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} y(k)
$$

4.2 Lineare Prädiktive Regelung

GPC vs. DMC

$$h_k = \sum_{i=0}^{k} g_{k-1}$$

Umformen des DMC:

Letzte prädizierte Regelgröße, lässt sich durch vergangene Stellgrößen berechnen

$$\underline{\hat{y}} = \underline{H}^- \Delta \underline{u}^- + \underline{H}^+ \Delta \underline{u}^+ + \underline{1}(y(k) - \hat{y}(k|k-1)) = \underline{H}_c^- \Delta \underline{u}^- + \underline{H}^+ \Delta \underline{u}^+ + \underline{1} y(k)$$

$$\underbrace{\begin{bmatrix} \hat{y}(k+1) \\ \hat{y}(k+2) \\ \hat{y}(k+3) \end{bmatrix}}_{\underline{\hat{y}}} = \underbrace{\begin{bmatrix} h_4 & h_3 & h_2 \\ h_5 & h_4 & h_3 \\ h_6 & h_5 & h_4 \end{bmatrix}}_{\underline{H}^-} \underbrace{\begin{bmatrix} \Delta u(k-2) \\ \Delta u(k-1) \\ \Delta u\,(k) \end{bmatrix}}_{\Delta \underline{u}^-} + \underbrace{\begin{bmatrix} h_1 & 0 & 0 \\ h_2 & h_1 & 0 \\ h_3 & h_2 & h_1 \end{bmatrix}}_{\underline{H}^+} \underbrace{\begin{bmatrix} \Delta u(k+1) \\ \Delta u(k+2) \\ \Delta u(k+3)\,(k) \end{bmatrix}}_{\Delta \underline{u}^+}$$

$$+\underline{1} y(k) - \underbrace{\begin{bmatrix} h_3 & h_2 & h_1 \\ h_3 & h_2 & h_1 \\ h_3 & h_2 & h_1 \end{bmatrix} \begin{bmatrix} \Delta u(k-2) \\ \Delta u(k-1) \\ \Delta u\,(k) \end{bmatrix}}_{\underline{1}\hat{y}(k|k-1)}$$

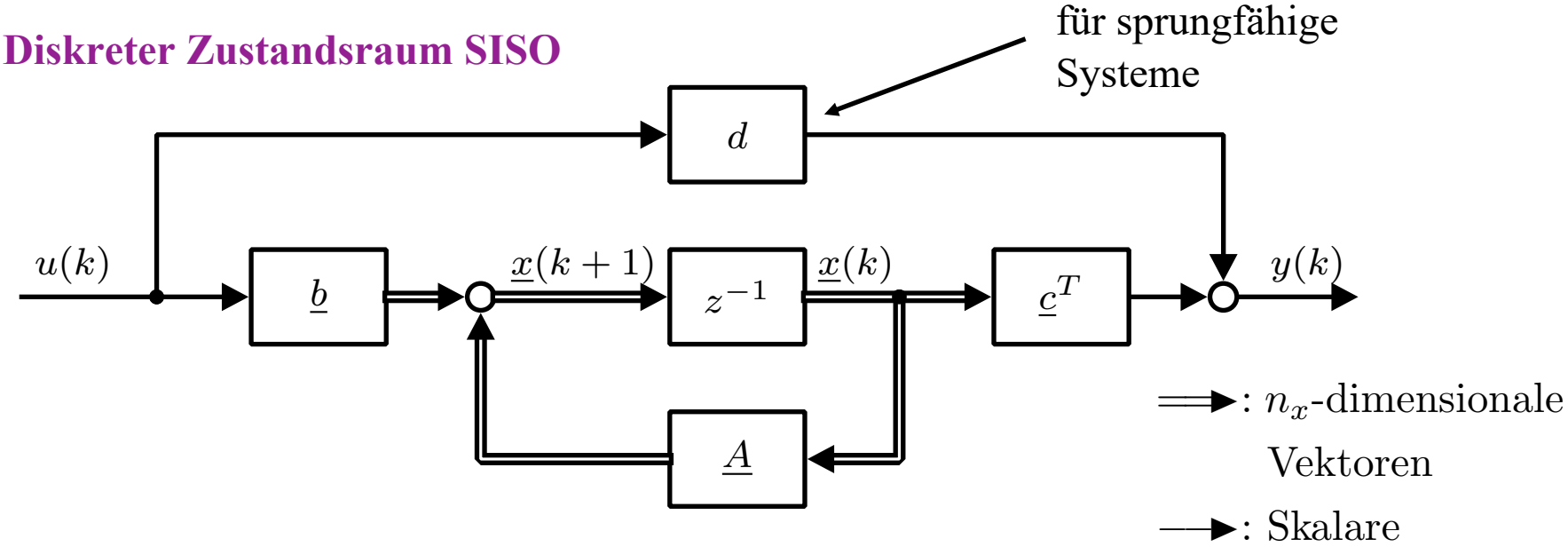
$$= \underbrace{\begin{bmatrix} h_4 - h_3 & h_3 - h_2 & h_2 - h_1 \\ h_5 - h_3 & h_4 - h_2 & h_3 - h_1 \\ h_6 - h_3 & h_5 - h_2 & h_4 - h_1 \end{bmatrix}}_{\underline{H}_c^-} \underbrace{\begin{bmatrix} \Delta u(k-2) \\ \Delta u(k-1) \\ \Delta u\,(k) \end{bmatrix}}_{\Delta \underline{u}^-} + \underbrace{\begin{bmatrix} h_1 & 0 & 0 \\ h_2 & h_1 & 0 \\ h_3 & h_2 & h_1 \end{bmatrix}}_{\underline{H}^+} \underbrace{\begin{bmatrix} \Delta u(k+1) \\ \Delta u(k+2) \\ \Delta u(k+3)\,(k) \end{bmatrix}}_{\Delta \underline{u}^+} + \underline{1} y(k)$$

$$= \underbrace{\begin{bmatrix} g_4 & g_3 & g_2 \\ g_5 + g_4 & g_4 + g_3 & g_3 + g_2 \\ g_6 + g_5 + g_4 & g_5 + g_4 + g_3 & g_4 + g_3 + g_2 \end{bmatrix}}_{\underline{H}_c^-} \Delta \underline{u}^- + \underbrace{\begin{bmatrix} h_1 & 0 & 0 \\ h_2 & h_1 & 0 \\ h_3 & h_2 & h_1 \end{bmatrix}}_{\underline{H}^+} \Delta \underline{u}^+ + \underline{1} y(k)$$

Das Gleiche Ergebnis wie bei der Umformung des GPC

4.2 Lineare Prädiktive Regelung

Diskreter Zustandsraum SISO



$$\underline{x}(k+1) = \underline{A}\,\underline{x}(k) + \underline{b}\,u(k)$$

$$y(k) = \underline{c}^T\,\underline{x}(k) + d\,u(k)$$

$\underline{A}$: Matrix: $n_x \times n_x$

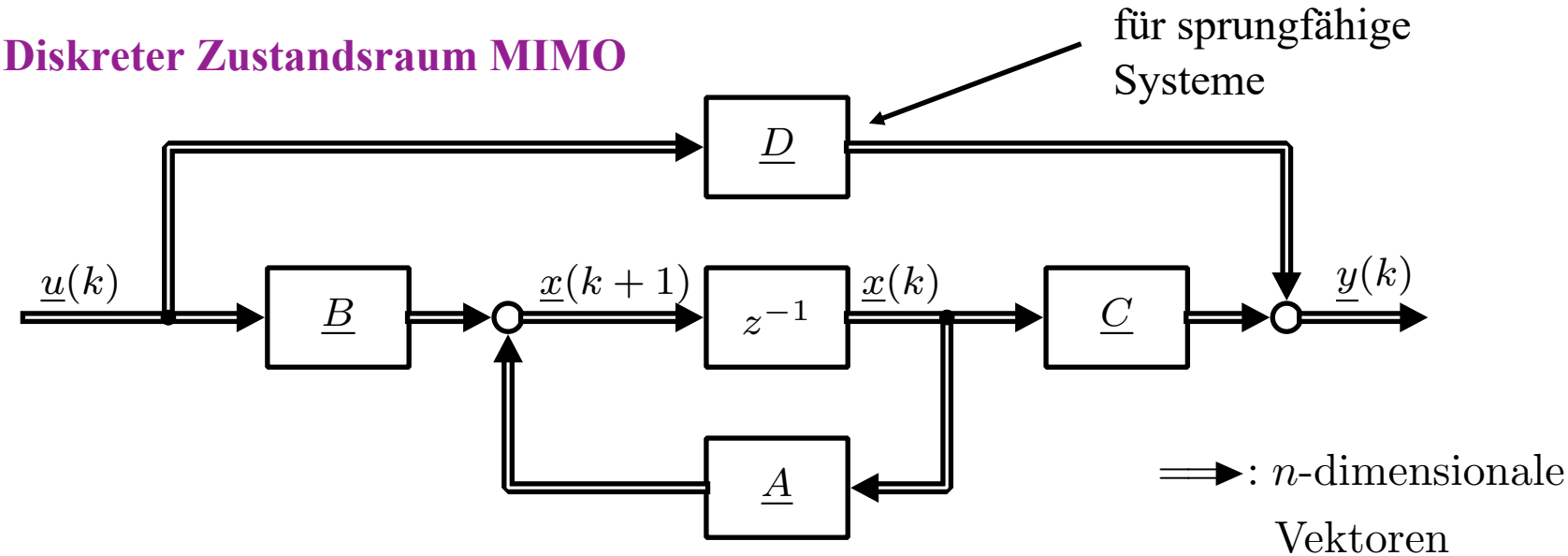
$\underline{b}$: Vektor: $n_x \times 1$

$\underline{c}^T$: Vektor: $1 \times n_x$

d : Skalar

4.2 Lineare Prädiktive Regelung

Diskreter Zustandsraum MIMO



$$\underline{x}(k+1) = \underline{A}\,\underline{x}(k) + \underline{B}\,\underline{u}(k)$$
$$\underline{y}(k) = \underline{C}\,\underline{x}(k) + \underline{D}\,\underline{u}(k)$$

$\underline{A}$: Matrix: $n_x \times n_x$

$\underline{B}$: Matrix: $n_x \times n_u$

$\underline{C}$: Matrix: $n_y \times n_x$

$\underline{D}$: Matrix: $n_y \times n_u$

4.2 Lineare Prädiktive Regelung

Diskreter Zustandsraum Markoff Entwicklung

$$\underline{x}(k+1) = \underline{A}\,\underline{x}(k) + \underline{b}\,u(k)$$

$$y(k) = \underline{c}^T\,\underline{x}(k) + d\,u(k)$$

+1

$$y(k+1) = \underline{c}^T\,\underline{x}(k+1) + d\,u(k+1)$$

$$= \underline{c}^T\,(\underline{A}\,\underline{x}(k) + \underline{b}\,u(k)) + d\,u(k+1)$$

+1

$$y(k+2) = \underline{c}^T\,\underline{x}(k+2) + d\,u(k+2)$$

$$= \underline{c}^T\,(\underline{A}\,\underline{x}(k+1) + \underline{b}\,u(k+1)) + d\,u(k+2)$$

$$= \underline{c}^T\,(\underline{A}\,(\underline{A}\,\underline{x}(k) + \underline{b}\,u(k)) + \underline{b}\,u(k+1)) + d\,u(k+2)$$

$$= \underline{c}^T\,\underline{A}^2\underline{x}(k) + \underline{c}^T\,\underline{A}\,\underline{b}\,u(k) + \underline{c}^T\,\underline{b}\,u(k+1) + d\,u(k+2)$$

$\vdots$

+i

$$y(k+i) = \underline{c}^T\,\underline{A}^i\underline{x}(k) + \sum_{j=1}^{i} \underline{c}^T\,\underline{A}^{j-1}\,\underline{b}\,u(k+i-j) \quad + d\,u(k+i)$$

4.2 Lineare Prädiktive Regelung

SS-based MPC – State Space based Model Predictive Control

- Basiert auf Zustandsraummodell ohne Durchgriff:

$$\hat{y}(k+i|k) = \underline{c}^T \, \underline{A}^i \underline{x}(k) + \sum_{j=1}^{i} \underline{c}^T \, \underline{A}^{j-1} \underline{b} u(k+i-j)$$

- Durch die Erweiterung des Zustandsraummodells, lässt sich die Stellgrößenänderung als Eingangsgröße des Systems nutzen:

$$\underbrace{\begin{bmatrix} \underline{x}(k+1) \\ u(k+1) \end{bmatrix}}_{\tilde{\underline{x}}(k+1)} = \underbrace{\begin{bmatrix} \underline{A} & \underline{b} \\ 0 & 1 \end{bmatrix}}_{\tilde{\underline{A}}} \underbrace{\begin{bmatrix} \underline{x}(k) \\ u(k) \end{bmatrix}}_{\tilde{\underline{x}}(k)} + \underbrace{\begin{bmatrix} \underline{b} \\ 1 \end{bmatrix}}_{\tilde{\underline{b}}} \Delta u(k)$$

$$y(k) = \underbrace{\begin{bmatrix} \underline{c}^T & 0 \end{bmatrix}}_{\tilde{\underline{c}}^T} \underbrace{\begin{bmatrix} \underline{x}(k) \\ u(k) \end{bmatrix}}_{\tilde{\underline{x}}(k)}$$

Einführung eines zusätzlichen Integrators

$$u(k+1) = u(k) + \Delta u(k)$$

- Die neue Systembeschreibung lautet:

$$\tilde{\underline{x}}(k+1) = \tilde{\underline{A}} \, \tilde{\underline{x}}(k) + \tilde{\underline{b}} \Delta u(k)$$

$$y(k) = \tilde{\underline{c}}^T \, \tilde{\underline{x}}(k)$$

4.2 Lineare Prädiktive Regelung

SS-based MPC – State Space based Model Predictive Control

- Die Prädiktion wird rekursiv durchgeführt. Dadurch ergeben sich für $N_p = N_u$ folgende prädizierte Regelgrößen

$$\underline{\hat{y}} = \begin{bmatrix} \hat{y}(k+1) \\ \hat{y}(k+2) \\ \vdots \\ \hat{y}\,(k+N_p) \end{bmatrix} = \begin{bmatrix} \underline{\tilde{c}}^T\,\underline{\tilde{A}}\,\underline{\tilde{x}}(k) + \underline{\tilde{c}}^T\,\underline{\tilde{b}}\Delta u(k) \\ \underline{\tilde{c}}^T\,\underline{\tilde{A}}^2\underline{\tilde{x}}(k) + \sum_{j=1}^{2}\underline{\tilde{c}}^T\,\underline{\tilde{A}}^{j-1}\underline{\tilde{b}}\Delta u(k+2-j) \\ \vdots \\ \underline{\tilde{c}}^T\,\underline{\tilde{A}}^{N_p}\underline{\tilde{x}}(k) + \sum_{j=1}^{N_p}\underline{\tilde{c}}^T\,\underline{\tilde{A}}^{j-1}\underline{\tilde{b}}\Delta u(k+N_p-j) \end{bmatrix}$$

- Mit der Matrix-Notation

$$\underline{\hat{y}} = \underline{F}\,\underline{\tilde{x}}(k) + \underline{H}\Delta\underline{u}^+$$

$$\underline{F} = \begin{bmatrix} \underline{\tilde{c}}^T\,\underline{\tilde{A}} \\ \underline{\tilde{c}}^T\,\underline{\tilde{A}}^2 \\ \vdots \\ \underline{\tilde{c}}^T\,\underline{\tilde{A}}^{N_p} \end{bmatrix} \quad \underline{H} = \begin{bmatrix} \underline{\tilde{c}}^T\,\underline{\tilde{b}} & 0 & 0 & \dots & 0 \\ \underline{\tilde{c}}^T\,\underline{\tilde{A}}\,\underline{\tilde{b}} & \underline{\tilde{c}}^T\,\underline{\tilde{b}} & 0 & \dots & 0 \\ \underline{\tilde{c}}^T\,\underline{\tilde{A}}^2\underline{\tilde{b}} & \underline{\tilde{c}}^T\,\underline{\tilde{A}}\,\underline{\tilde{b}} & \underline{\tilde{c}}^T\,\underline{\tilde{b}} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \underline{\tilde{c}}^T\,\underline{\tilde{A}}^{N_p-1}\underline{\tilde{b}} & \underline{\tilde{c}}^T\,\underline{\tilde{A}}^{N_p-2}\underline{\tilde{b}} & \underline{\tilde{c}}^T\,\underline{\tilde{A}}^{N_p-3}\underline{\tilde{b}} & \dots & \underline{\tilde{c}}^T\,\underline{\tilde{b}} \end{bmatrix}$$

4.2 Lineare Prädiktive Regelung

SS-based MPC – State Space based Model Predictive Control

- Durch Einsetzen der Modellgleichung in die Verlustfunktion

$$\underline{\hat{y}} = \underline{F}\,\underline{\tilde{x}}(k) + \underline{H}\Delta\underline{u}^{+}$$

$$\min_{\Delta u(k),\cdots,\Delta u(k+N_u-1)} \sum_{i=1}^{N_p} \left(w(k+i) - \hat{y}(k+i \mid k)\right)^2 + \lambda \sum_{i=0}^{N_\mathrm{u}-1} \Delta u(k+i)^2$$

ergibt sich ohne die Nebenbedingungen/Beschränkung folgende analytische Lösung:

$$\Delta\underline{u}^{+} = \left(\underline{H}^T\,\underline{H} + \lambda\underline{I}\right)^{-1} \cdot \underline{H}^T\left(\underline{w} - \underline{F}\,\underline{\tilde{x}}(k)\right)$$

4.2 Lineare Prädiktive Regelung

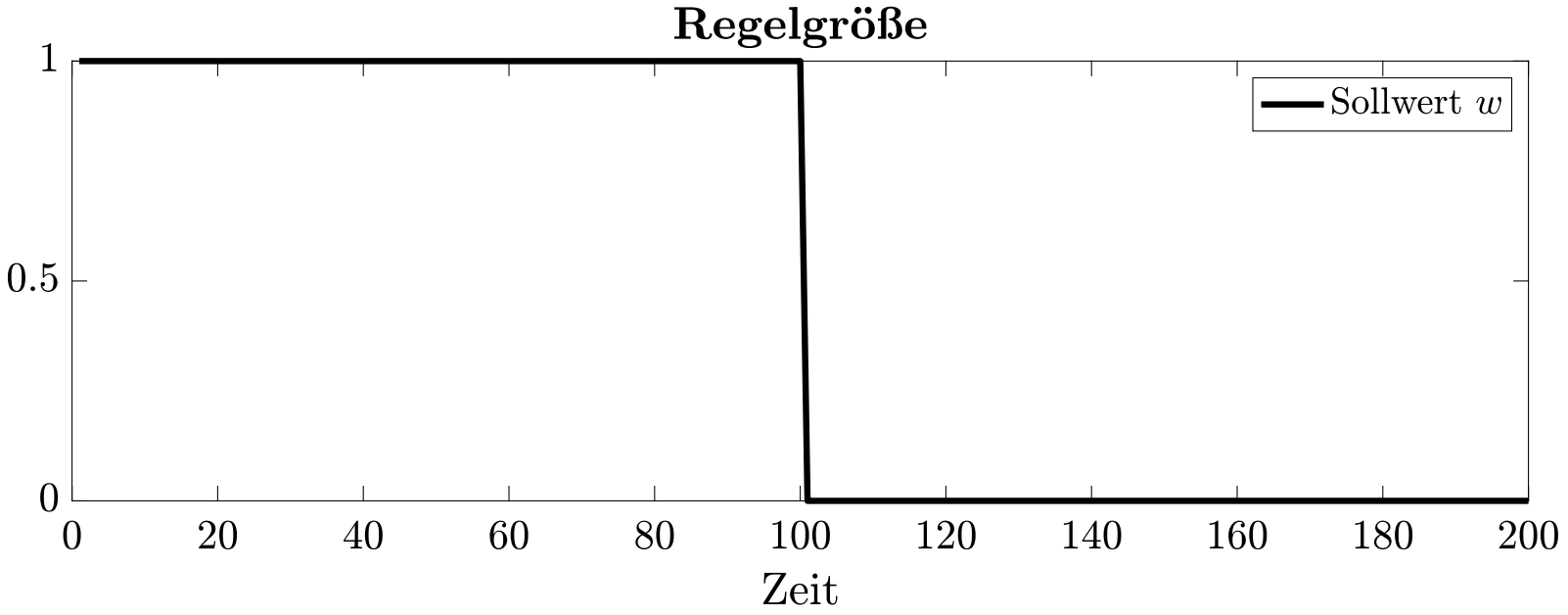
Beispiel: MPC mit System erster Ordnung mit Totzeit

$$G(s) = \frac{1}{10s+1} e^{-T_t s}, \; G(z) = \frac{0{,}095 z^{-1}}{1 - 0{,}905 z^{-1}} z^{-T_t}$$

Es wird angenommen, dass der Prozess dem Modell im MPC entspricht.

λ, N_p, N_u, T_t werden jeweils einzeln variiert und das Regelkreisverhalten untersucht.

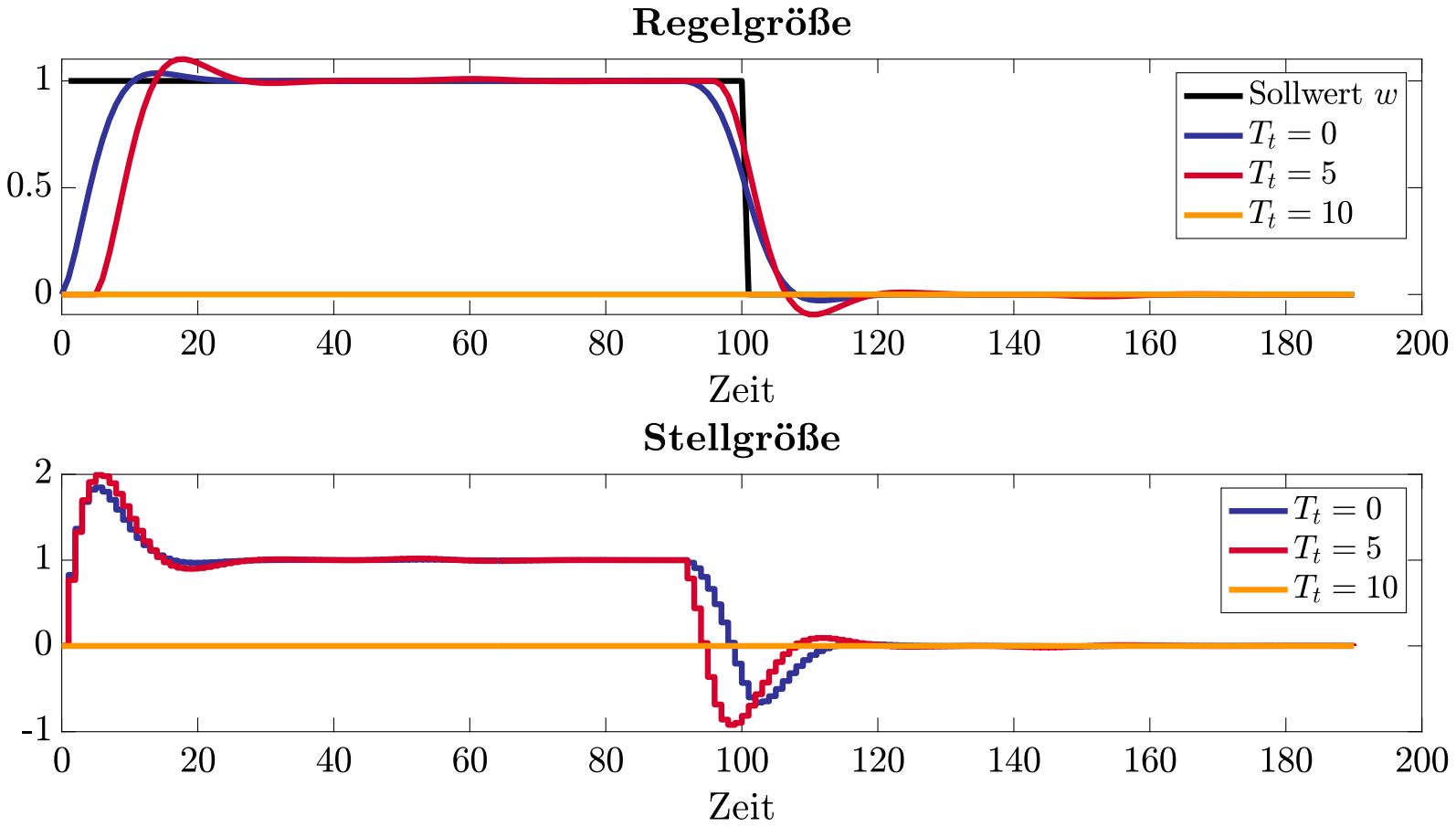
Im Nachfolgenden wird eine Solltrajektorie vorgegeben:



4.2 Lineare Prädiktive Regelung

Beispiel: Verschiedene Totzeiten im Prozess

$$G(s) = \frac{1}{10s+1} e^{-T_t s},\ G(z) = \frac{0{,}095 z^{-1}}{1 - 0{,}905 z^{-1}} z^{-T_t},\ N_p = 10,\ N_u = 10,\ \lambda = 1$$



4.2 Lineare Prädiktive Regelung

Beispiel: Verschiedene Totzeiten im Prozess

$$G(s) = \frac{1}{10s+1} e^{-T_t s},\ G(z) = \frac{0{,}095 z^{-1}}{1 - 0{,}905 z^{-1}} z^{-T_t},\ N_p = 10,\ N_u = 10,\ \lambda = 1$$

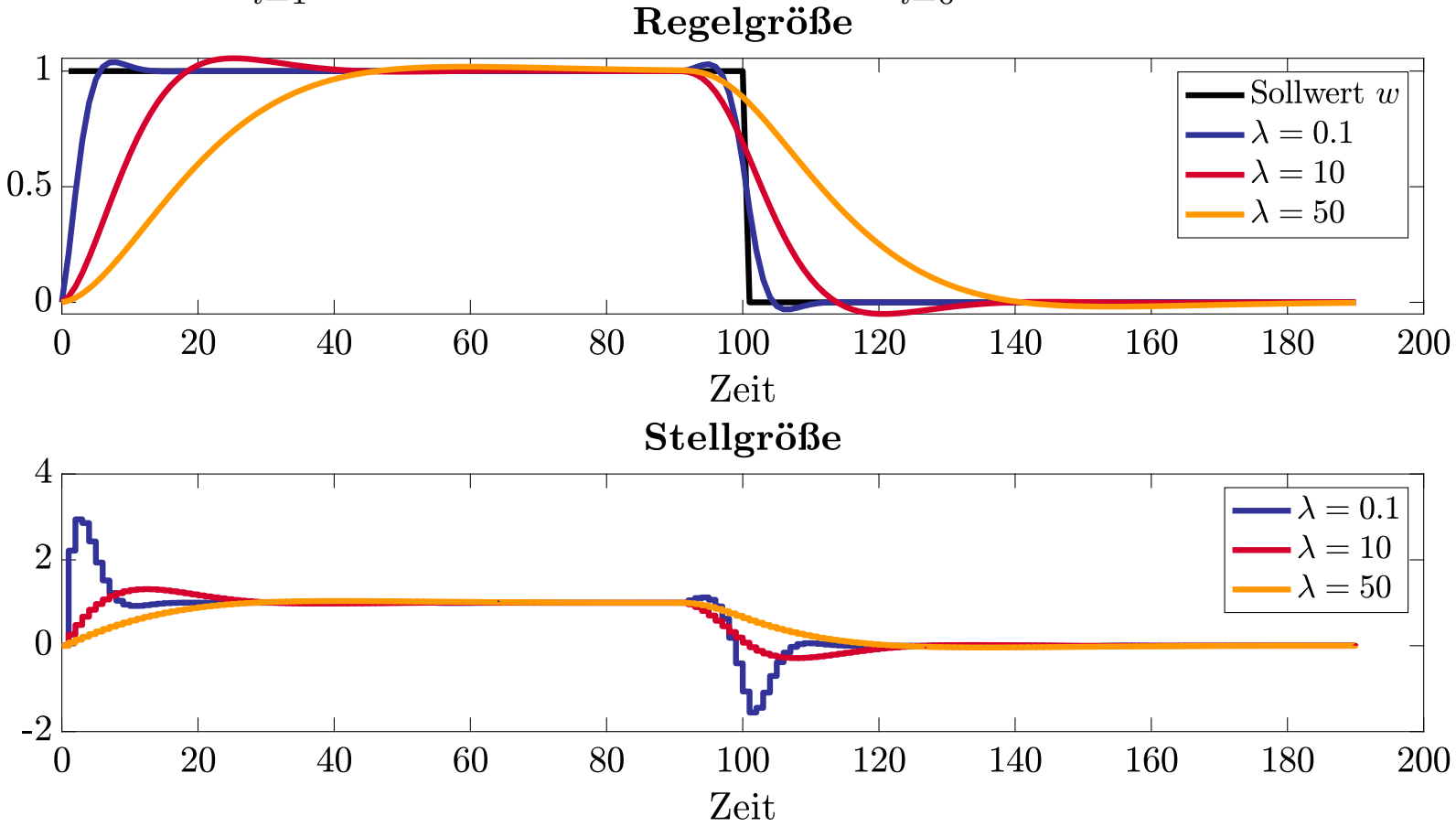
Bei einer Totzeit der Regelgröße, welche größer oder gleich dem Prädiktionshorizont ist, bewirken alle Stellgrößenfolgen keine Änderung des Prozesses. Um die Gütefunktion zu minimieren, wird daher die Stellgröße auf null optimiert. Es kommt zu keiner Regelung.

Bei Totzeiten im Prozess muss außerdem abgewartet werden bis der Prozess auf die Stellgrößen reagiert, daher ist der Regler bei Prozessen mit höherer Totzeit langsamer. Sobald eine Sollgrößenänderung innerhalb des Prädiktionshorizontes liegt, reagiert der Regler und passt die Stellgröße an, damit die zukünftigen Regelgrößen sich dem neuen Sollwert annähern.

4.2 Lineare Prädiktive Regelung

Beispiel: Variation der λ in der Verlustfunktion

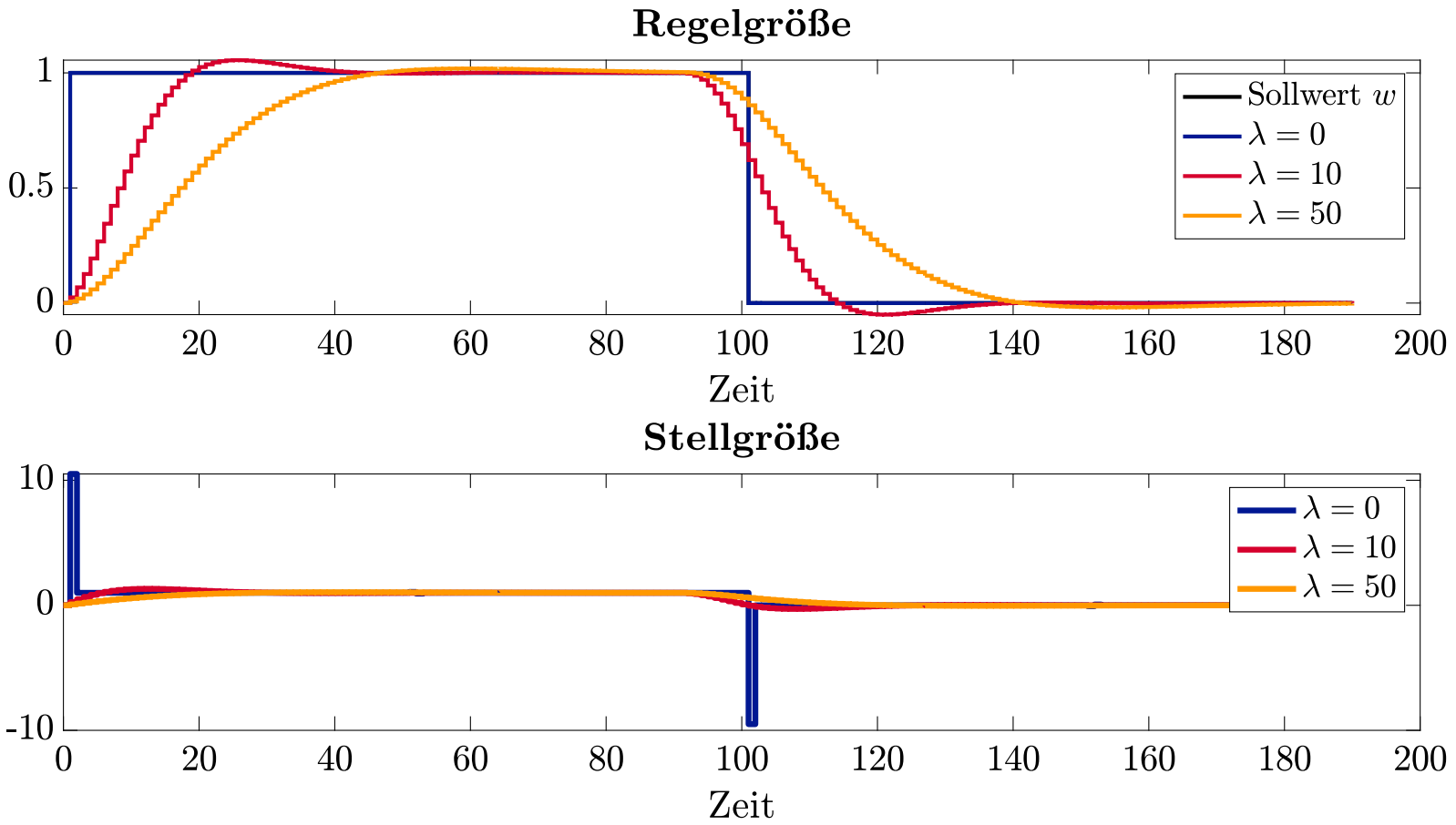
$$\min_{u(k),\cdots,u(k+N_u-1)} \sum_{i=1}^{N_p} \left(w(k+i) - \hat{y}(k+i \mid k)\right)^2 + \lambda \sum_{i=0}^{N_u-1} \Delta u(k+i)^2$$



4.2 Lineare Prädiktive Regelung

Beispiel: Spezialfall $\lambda = 0$

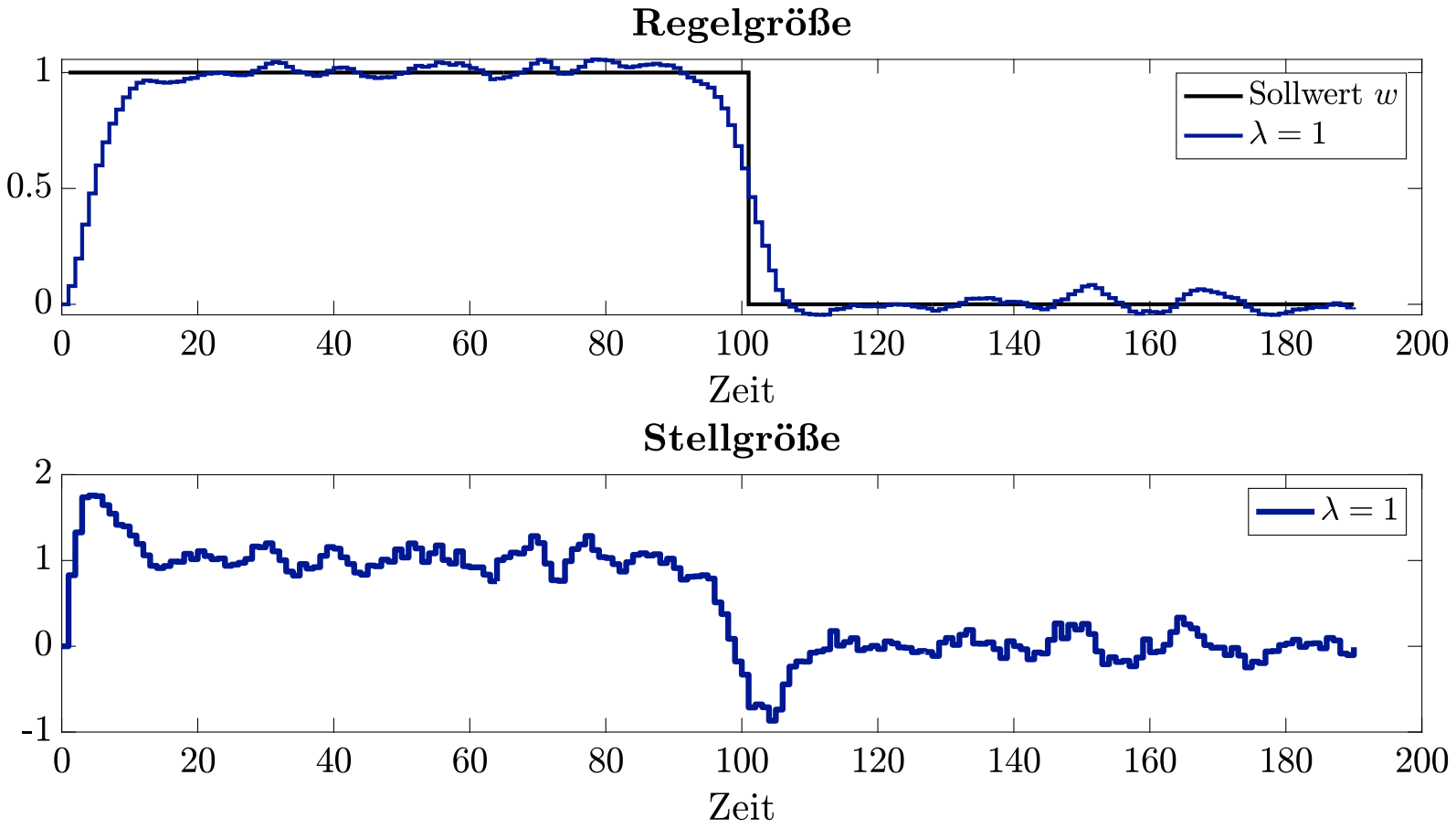
Entspricht dem Deadbeat-Regler:



4.2 Lineare Prädiktive Regelung

Beispiel: Verrauschter Prozess

Varianz: $\sigma^2 = 0.01$



4.2 Lineare Prädiktive Regelung

Beispiel: Variation der λ in der Verlustfunktion

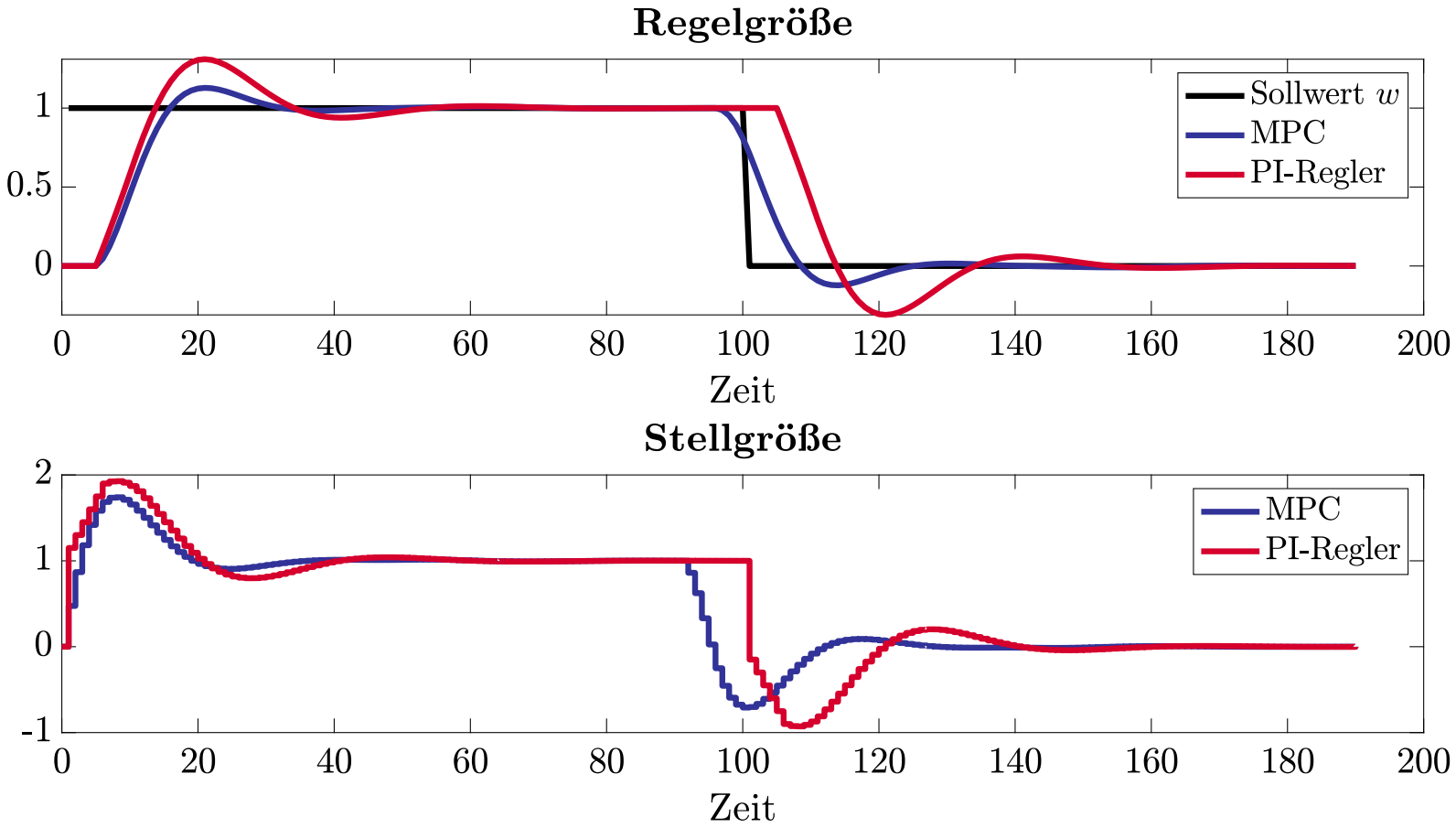
$$\min_{u(k),\cdots,u(k+N_u-1)} \sum_{i=1}^{N_p} \left(w(k+i) - \hat{y}(k+i \mid k)\right)^2 + \lambda \sum_{i=0}^{N_u-1} \Delta u(k+i)^2$$

Je höher λ ist, desto stärker wird die Stellgrößenänderung in Relation zur Regelabweichung bestraft. Für kleine Werte für λ ist es wichtiger, dass die Regelabweichung minimiert wird, da dieser Teil der Verlustfunktion höher gewichtet wird. Daraus resultiert ein aggressiver Regler. Für große λ soll die Änderung der Stellgröße möglichst gering sein. Daher wird die Regelabweichung nur langsam verringert. Somit ergibt sich eine langsame Regelung.

4.2 Lineare Prädiktive Regelung

Beispiel: Vergleich MPC mit PI-Regler

$$G(s) = \frac{1}{10s+1}e^{-5s},\ G(z) = \frac{0{,}095z^{-1}}{1-0{,}905z^{-1}}z^{-5},\ N_p = 10,\ N_u = 10,\ \lambda = 2$$



4.2 Lineare Prädiktive Regelung

Beispiel: Vergleich MPC mit PI-Regler

$$G(s) = \frac{1}{10s+1}e^{-5s},\ G(z) = \frac{0{,}095z^{-1}}{1-0{,}905z^{-1}}z^{-5},\ N_p = 10,\ N_u = 10,\ \lambda = 2$$

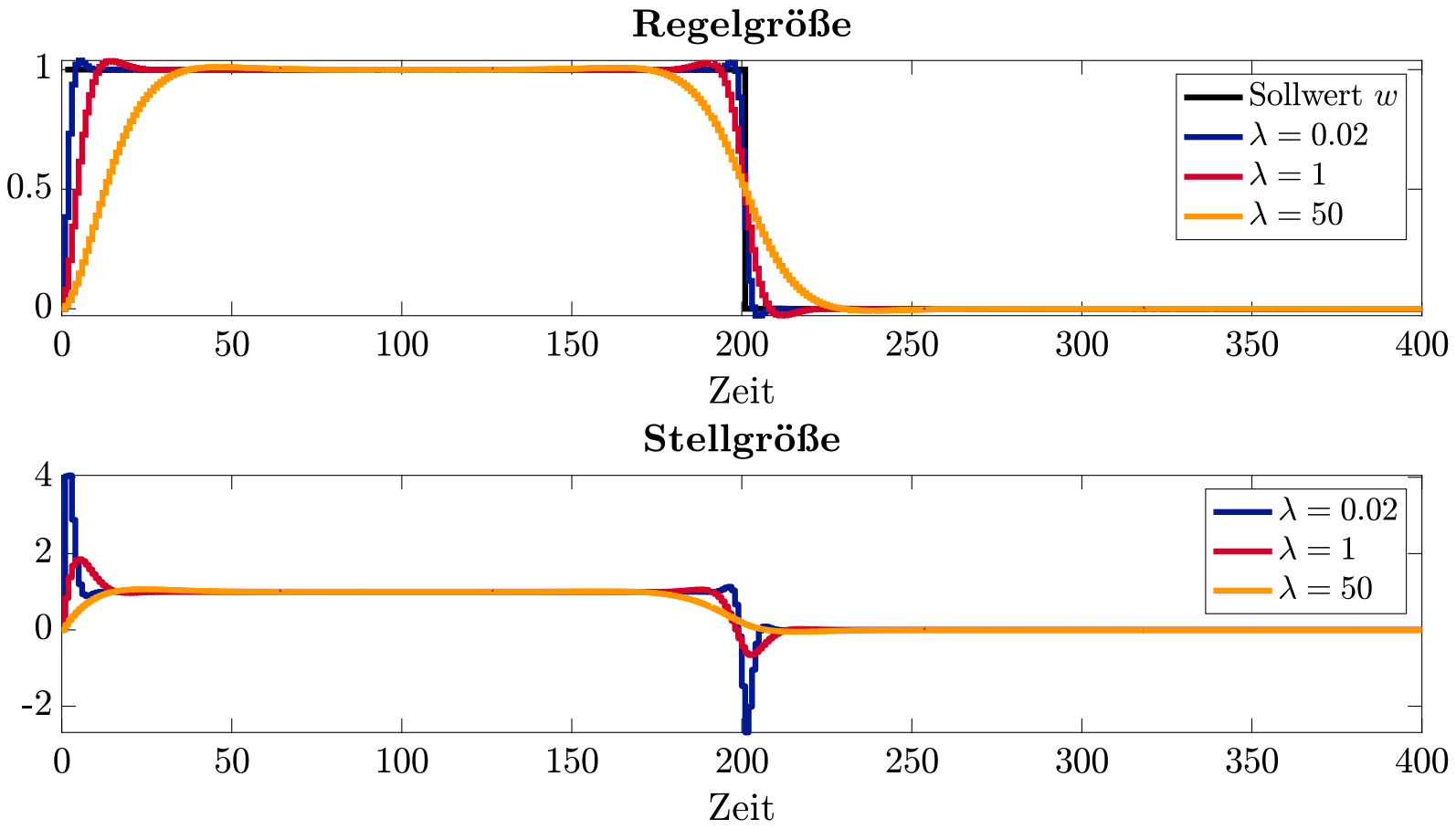
Der durch die Prädiktion im MPC kann dieser bis zum Prädiktionshorizont N_p in die Zukunft planen. Daher ist es möglich, dass bei dem Sollgrößensprung zum Zeitpunkt $k = 100$ der Regler bereits vorab (in diesem Fall ab $k = 90$) reagiert und die Stellgröße anpasst.

Der PI-Regler bestimmt die Regelabweichung erst zum Zeitpunkt $k = 100$ und durch die Totzeit $T_t = 5$ sec kann nur spät die Regelgröße beeinflusst werden.

4.2 Lineare Prädiktive Regelung

Beispiel: Variation des Prädiktionshorizontes und des Stellhorizontes

$$G(s) = \frac{1}{10s+1},\ G(z) = \frac{0{,}095z^{-1}}{1-0{,}905z^{-1}},\ N_p = N_u = 100$$



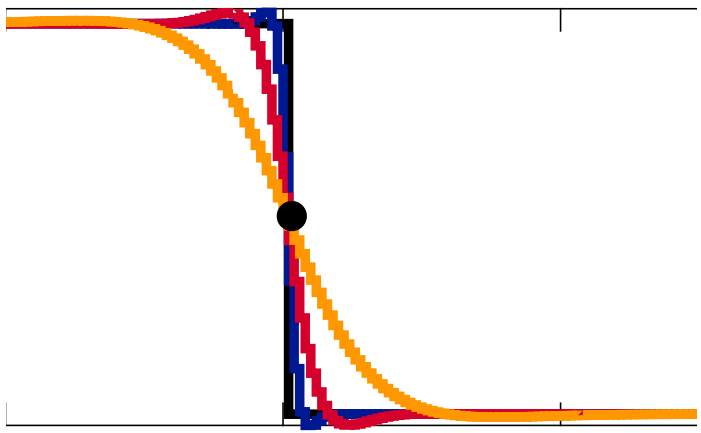
4.2 Lineare Prädiktive Regelung

Beispiel: Variation des Prädiktionshorizontes und des Stellhorizontes

$$G(s) = \frac{1}{10s+1},\ G(z) = \frac{0{,}095z^{-1}}{1-0{,}905z^{-1}},\ N_p = N_u = 100$$

Für $N_p = N_u$ konvergiert der optimale Stellgrößenverlauf.

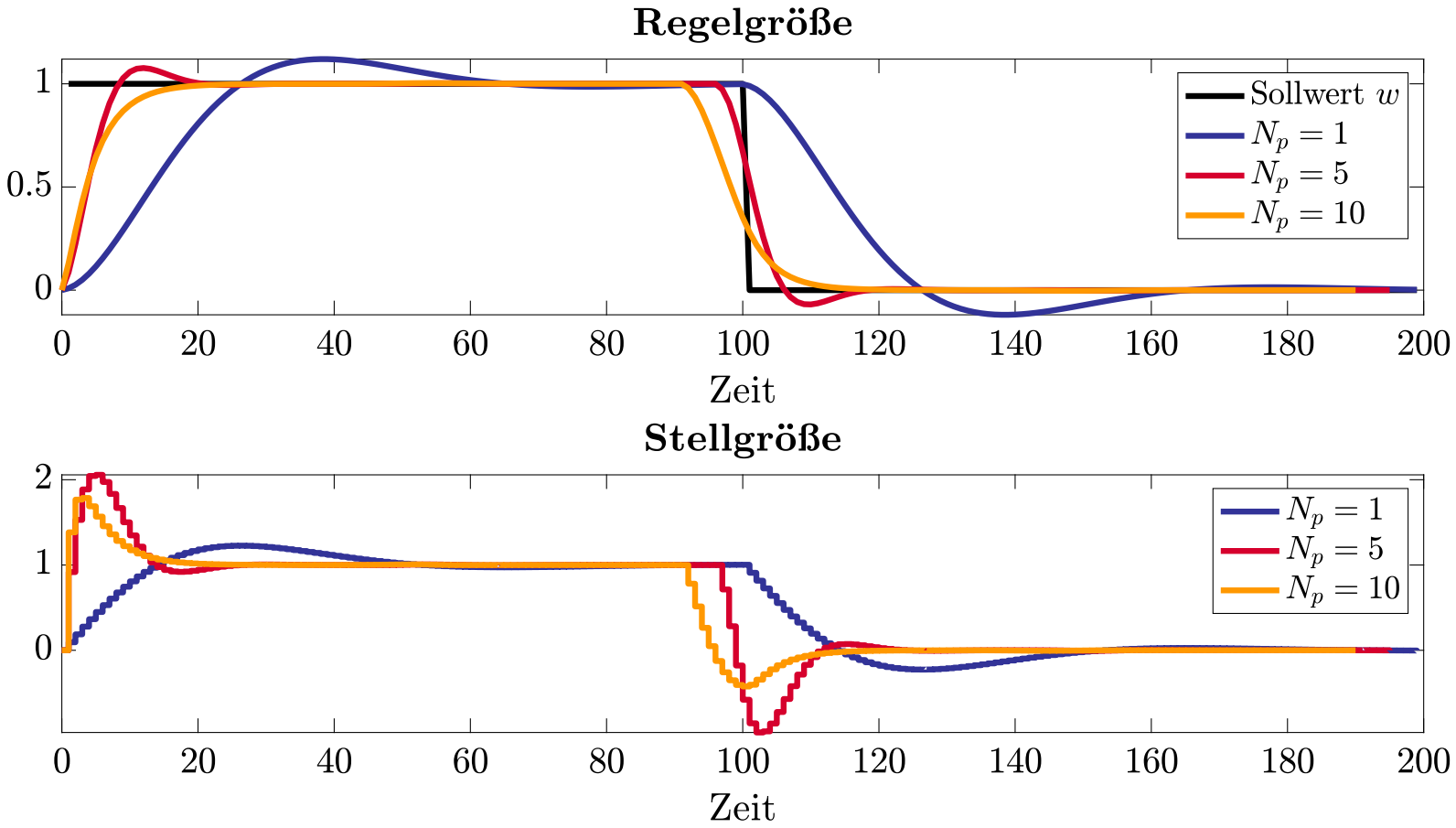
Dabei ist die Regelgröße punktsymmetrisch und das Verhalten beim Verlassen der alten Führungsgröße und beim Erreichen der neuen Führungsgröße ist identisch. Die Steigung beim Übergang wird durch λ festgelegt.



4.2 Lineare Prädiktive Regelung

Beispiel: Variation des Prädiktionshorizontes

$$G(s) = \frac{1}{10s+1},\ G(z) = \frac{0{,}095z^{-1}}{1-0{,}905z^{-1}},\ N_u = 1,\ \lambda = 1$$



4.2 Lineare Prädiktive Regelung

Beispiel: Variation des Prädiktionshorizontes

$$G(s) = \frac{1}{10s+1},\ G(z) = \frac{0{,}095z^{-1}}{1-0{,}905z^{-1}},\ N_u = 1,\ \lambda = 1$$

Je länger der Prädiktionshorizont, desto früher kann die Regelung reagieren. Durch den längeren Prädiktionshorizont kann der Regler auch besser abschätzen, wann die Regelgröße der Sollgröße entspricht und die Stellgröße darauf anpassen. Dadurch ergeben sich geringere Überschwinger. Dabei sollte der Prädiktionshorizont auch in Abhängigkeit der Zeitkonstante des Prozesses gewählt werden. Für langsamere Prozesse wird ein längerer Prädiktionshorizont benötigt als für schnelle Prozesse.

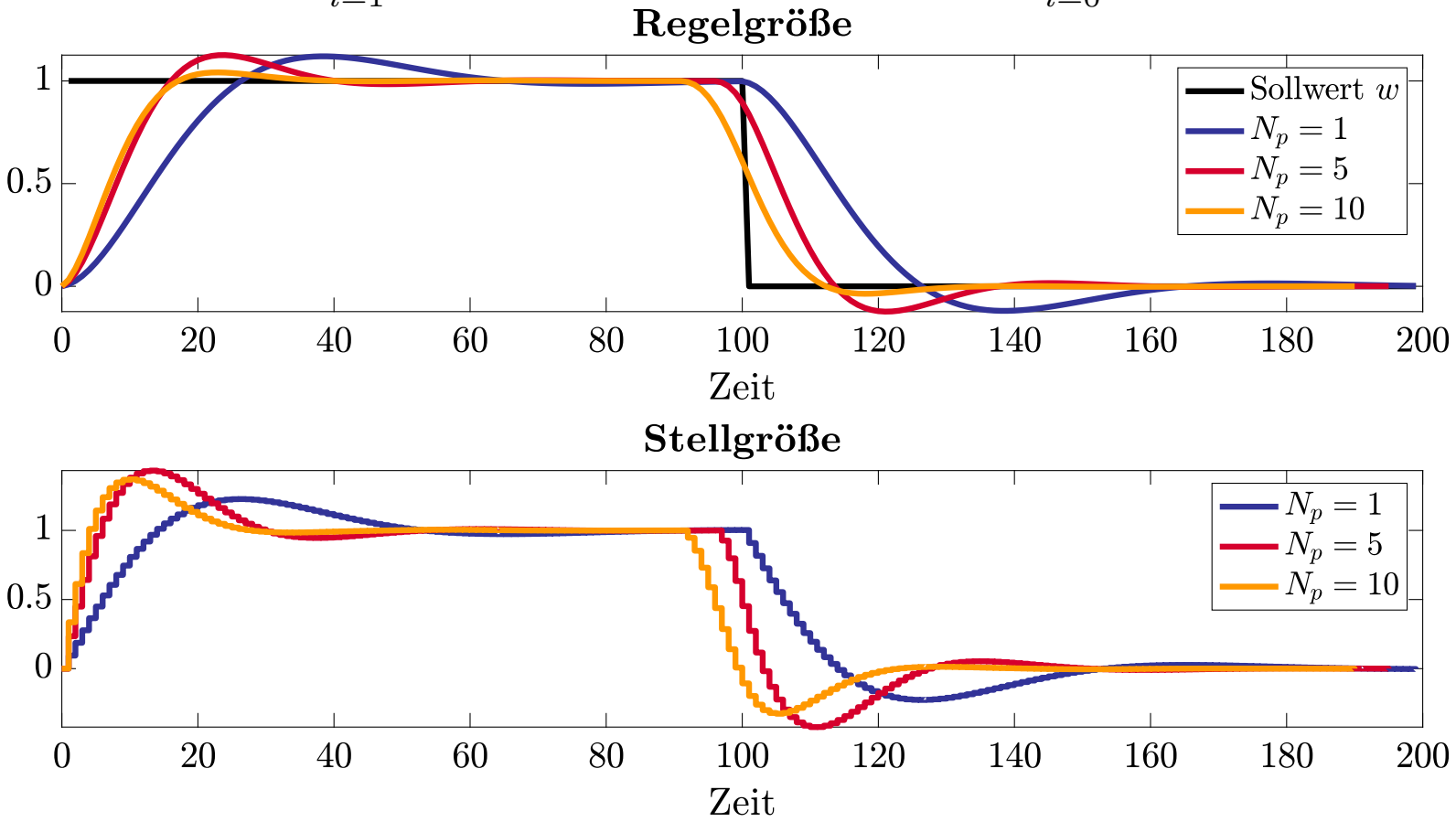
Durch Änderung des Prädiktionshorizonts oder des Stellhorizonts wird auch das Verhältnis zwischen den aufsummierten Regelfehlerquadrate und der summierten Stellgrößenänderungsquadrate verändert. Um das anzupassen wird die Verlustfunktion für die nachfolgenden Beispiele geändert:

$$\min_{u(k),\cdots,u(k+N_u-1)} \frac{1}{N_p}\sum_{i=1}^{N_p}\left(w(k+i)-\hat{y}(k+i\mid k)\right)^2 + \frac{1}{N_u}\lambda\sum_{i=0}^{N_u-1}\Delta u(k+i)^2$$

4.2 Lineare Prädiktive Regelung

Beispiel: Variation des Prädiktionshorizontes

$$\min_{u(k),\cdots,u(k+N_u-1)} \frac{1}{N_p}\sum_{i=1}^{N_p}\left(w(k+i)-\hat{y}(k+i\mid k)\right)^2+\frac{1}{N_u}\lambda\sum_{i=0}^{N_u-1}\Delta u(k+i)^2$$

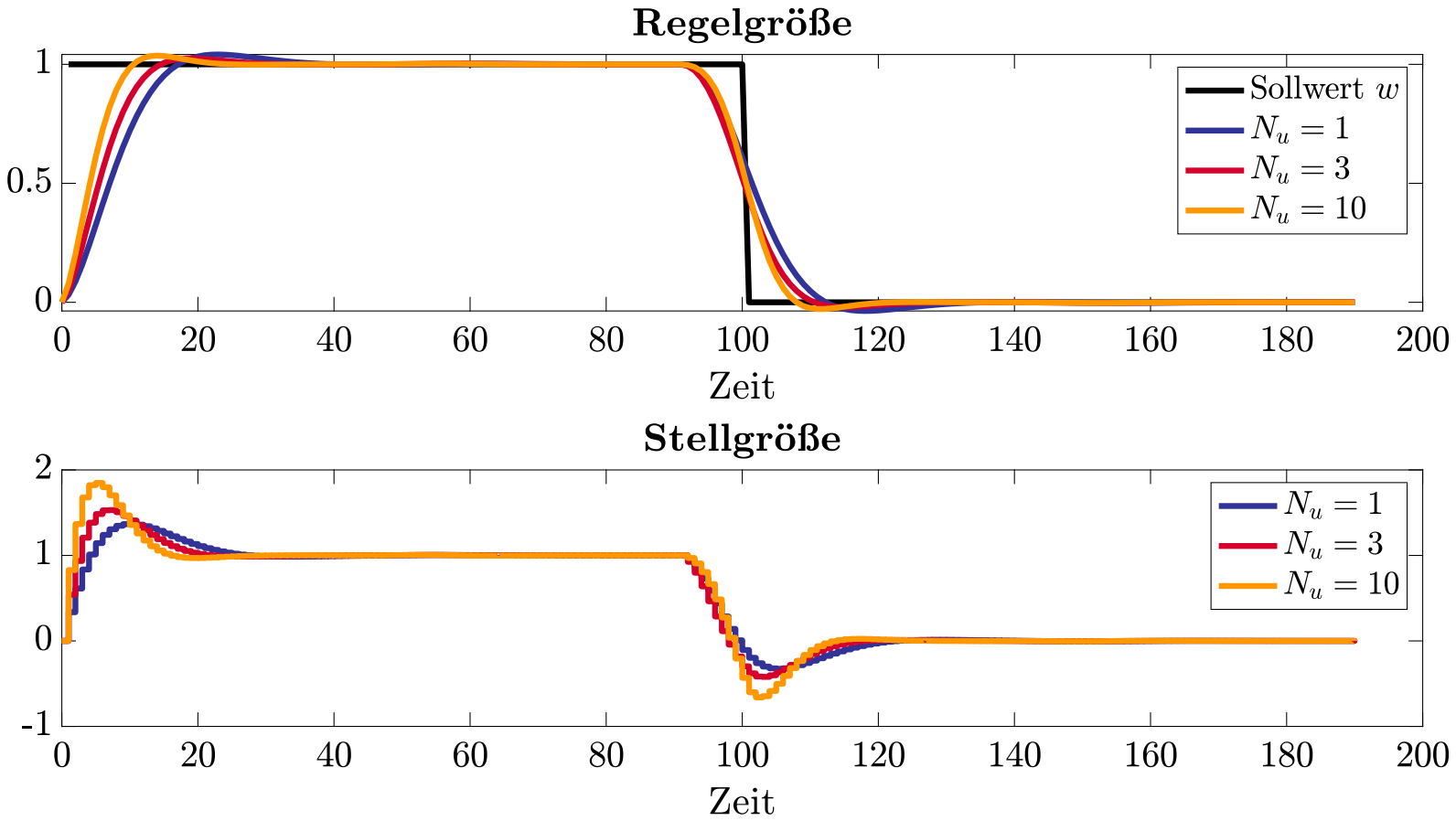


4.2 Lineare Prädiktive Regelung

$$I = \frac{1}{N_p}\sum_{i=1}^{N_p} e(k+i)^2 + \frac{1}{N_u}\lambda \sum_{i=0}^{N_u-1} \Delta u(k+i)^2$$

Beispiel: Variation des Stellhorizontes

$$G(s) = \frac{1}{10s+1},\ G(z) = \frac{0{,}095z^{-1}}{1-0{,}905z^{-1}},\ N_p = 10,\ \lambda = 1$$



4.2 Lineare Prädiktive Regelung

$$I = \frac{1}{N_p}\sum_{i=1}^{N_p} e(k+i)^2 + \frac{1}{N_u}\lambda \sum_{i=0}^{N_u-1} \Delta u(k+i)^2$$

Beispiel: Variation des Stellhorizontes

$$G(s) = \frac{1}{10s+1},\ G(z) = \frac{0{,}095z^{-1}}{1-0{,}905z^{-1}},\ N_p = 10,\ \lambda = 1$$

Durch den Stellhorizont wird festgelegt, wie viele verschiedene Stellgrößen unabhängig von einander optimiert werden. Alle Stellgrößen nach dem Stellhorizont entsprechen der letzten Stellgröße, welche somit konstant bis zum Prädiktionshorizont gehalten wird. Durch Erhöhung des Stellhorizonts, wird das Optimierungsproblem immer komplexer, da mehr Variablen optimiert werden müssen. Der Vorteil ist, dass durch einen längeren Stellhorizont eine geringere gesamt Regelabweichung erzielt werden kann, da mehr Freiheitsgrade gegeben sind.

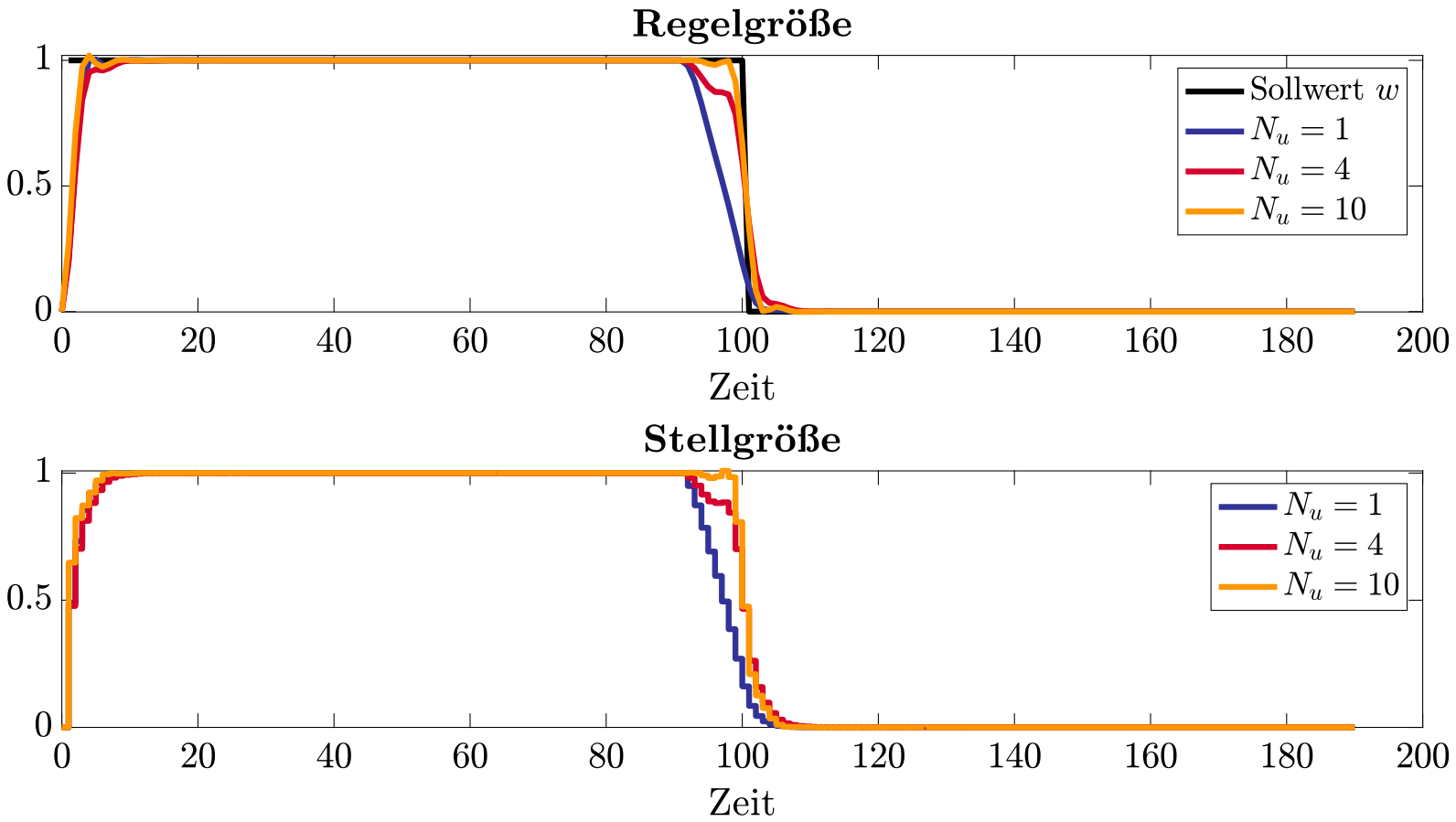
Die Regelungen der verschiedenen Stellhorizonte sind sehr ähnlich für einfache Prozesse und daher werden zur Einsparung von Rechenzeit oft ziemliche kurze Stellhorizonte genutzt.

4.2 Lineare Prädiktive Regelung

$$I = \frac{1}{N_p}\sum_{i=1}^{N_p} e(k+i)^2 + \frac{1}{N_u}\lambda \sum_{i=0}^{N_u-1} \Delta u(k+i)^2$$

Beispiel: Variation des Stellhorizontes für einen schwingungsfähigen Prozess

$$G(s) = \frac{1.25}{s^2 + s + 1.25},\ p_{1/2} = -0.5 \pm i,\ N_p = 10,\ \lambda = 1$$



4.2 Lineare Prädiktive Regelung

$$I = \frac{1}{N_p}\sum_{i=1}^{N_p} e(k+i)^2 + \frac{1}{N_u}\lambda \sum_{i=0}^{N_u-1} \Delta u(k+i)^2$$

Beispiel: Variation des Stellhorizontes für schwingungsfähigen Prozess

$$G(s) = \frac{1.25}{s^2 + s + 1.25},\ p_{1/2} = -0.5 \pm i,\ N_p = 10,\ \lambda = 1$$

Bei schwieriger zu regelnden Prozessen, wie einem schwingungsfähigen Prozess, zeigt sich der Vorteil eines längeren Stellhorizontes. Dabei kann die Regelung deutlich verbessert werden. Durch die hohe Eigenfrequenz dieses Prozesses werden mehrere Stellgrößen benötigt, um eine effektive Regelung zu realisieren.

4.2 Lineare Prädiktive Regelung

Vergleich: Prädiktive Regelung vs. LQ-Regelung

In beiden Reglertypen werden Optimierungsprobleme gelöst, um eine optimale Stellgrößenfolge zu bestimmen. Dabei kann man wie mit λ beim MPC mit der Wahl von Q und r die Stärke der Regelung beeinflusst werden.

Unterschiede

Prädiktive Regelung	LQ-Regelung
Einstellparameter: λ, N_p, N_u	Einstellparameter: Q, r
Optimale Stellgrößenfolge wird online immer wieder bestimmt	Optimale Rückführung wird offline berechnet
endlicher Zeithorizont	i.d.R. unendlicher Zeithorizont
Quadratische Verlustfunktion (konvex) mit einfachem Ergebnis	Kompliziertes mathematisches Problem für Mensch; für Computer recht einfach zu lösen
Einfach zu lösen mit Nebenbedingungen	Normalerweise ohne Nebenbedingungen
Schwere Stabilitätsanalyse	Vereinfachte Stabilitätsanalyse

Ohne Nebenbedingungen und für $N_p, N_u \to \infty$ liefern beide die gleiche Lösung!

4.3 Nebenbedingungen in der Prädiktiven Regelung

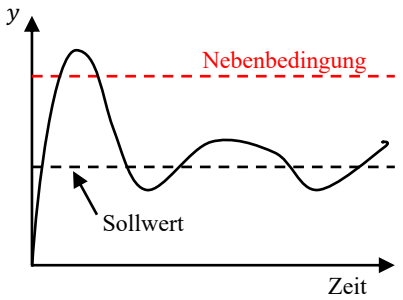
Alle physikalischen Systeme haben Nebenbedingungen:

- *Physikalische* Nebenbedingungen, z.B. Beschränkungen der Aktoren
- *Betriebs*-Nebenbedingungen, z.B. Überschwinger, Wirtschaftliche NB
- *Sicherheits*-Nebenbedingungen, z.B. Temperatur/Druck Beschränkungen

Optimale Sollwerte sind oft in der Nähe der Nebenbedingungen.

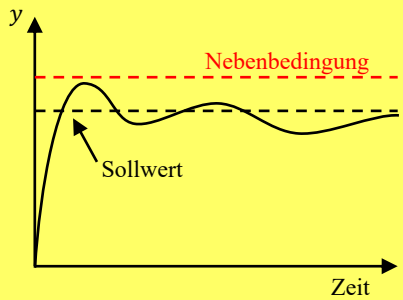
Klassische Regler

- Stellgröße: Abschneiden bei Verletzung der Beschränkung
- Regelgröße: Sollwert hinreichend weit von NB legen
- Suboptimaler Betrieb der Anlage



Prädiktive Regelung

- Nebenbedingungen werden im Optimierungsproblem berücksichtigt
- Optimaler Sollwert
- Optimaler Betrieb der Anlage



4.3 Nebenbedingungen in der Prädiktiven Regelung

Nebenbedingungen Stellgrößen

- Beschränkung der maximal/minimal Werte (z.B. Maximale Leistung Pumpe)
- Beschränkung der Änderung der Stellgrößen (z.B. möglichst wenig Abnutzung)

Nebenbedingungen Regelgrößen

- Beschränkung der maximal/minimal Werte (z.B. Tankhöhe)

Nebenbedingungen Zustände (Zustandsraum)

- Zustände beschreiben physikalische Größen

→ Können ebenfalls beschränkt werden

4.3 Nebenbedingungen in der Prädiktiven Regelung

Beispiel: Stellgrößen Nebenbedingungen

Die Verlustfunktion

$$\min_{u(k),\cdots,u(k+N_u-1)} \sum_{i=1}^{N_p} \left(w(k+i) - \hat{y}(k+i \mid k)\right)^2 + \lambda \sum_{i=0}^{N_\mathrm{u}-1} \Delta u(k+i)^2$$

wird mit Beschränkungen in den Stellgröße bzw. Stellgrößenänderungen gelöst. Hierfür wird eine Ober-/Untergrenze der Stellgrößen bzw. Stellgrößenänderungen eingeführt:

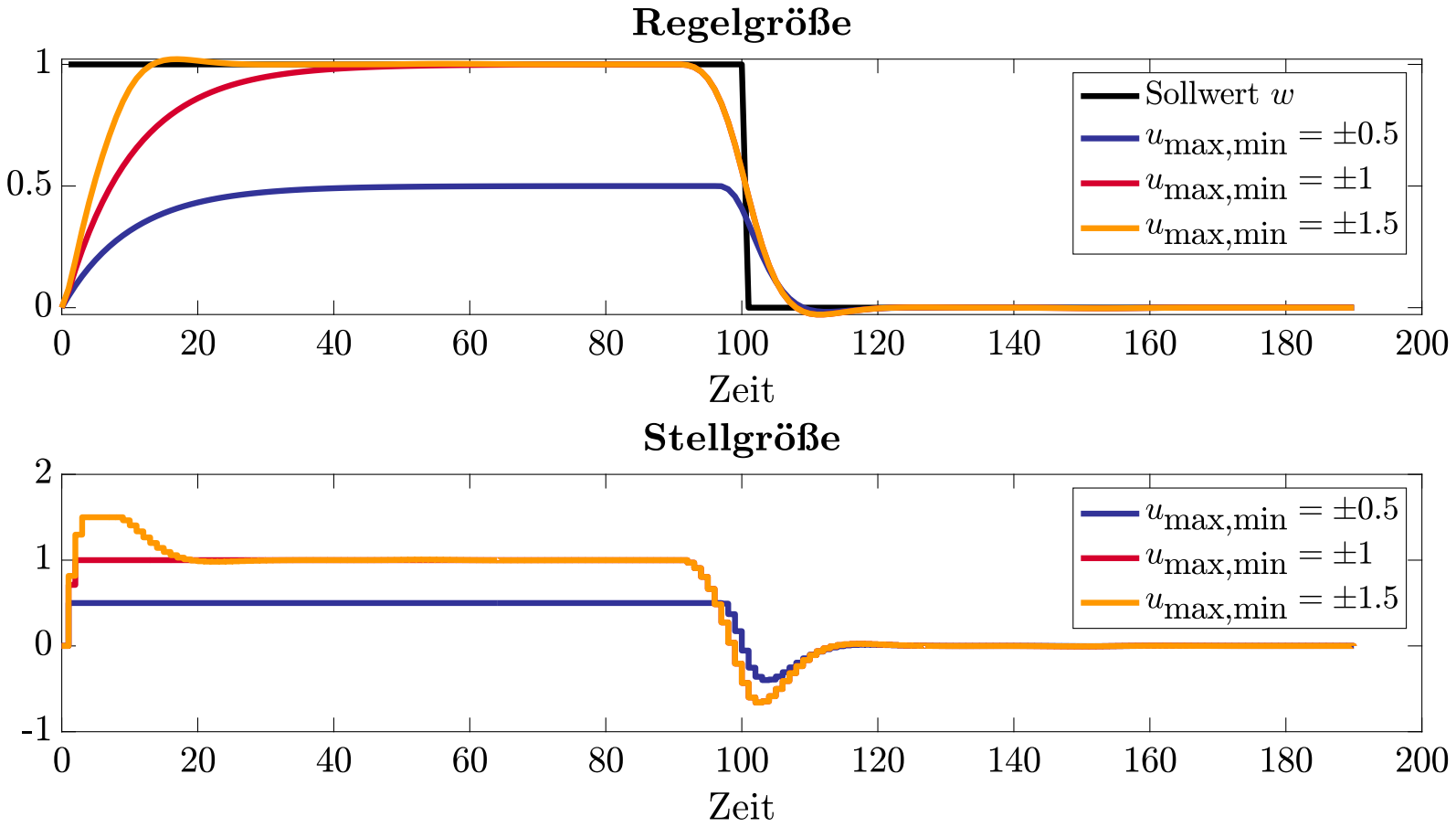
$$\begin{aligned} \forall i : \Delta u(k+i) &\leq \Delta u_{\max} \\ \Delta u(k+i) &\geq \Delta u_{\min} \\ u(k+i) &\leq u_{\max} \\ u(k+i) &\geq u_{\min} \end{aligned}$$

Dies führt zu einem QP, einer quadratischen Verlustfunktion mit linearen Nebenbedingungen und ist mit den Algorithmen aus Kapitel 3 lösbar.

4.3 Nebenbedingungen in der Prädiktiven Regelung

Beispiel: Stellgrößen Nebenbedingungen

$$G(s) = \frac{1}{10s+1},\ N_p = 10,\ N_u = 10,\ \lambda = 1$$



4.3 Nebenbedingungen in der Prädiktiven Regelung

Beispiel: Stellgrößen Nebenbedingungen

$$G(s) = \frac{1}{10s+1},\ N_p = 10,\ N_u = 10,\ \lambda = 1$$

1. Fall: Die Stellgrößenbeschränkung ist zu niedrig gewählt, sodass der Endwert von eins nicht erreicht werden kann.

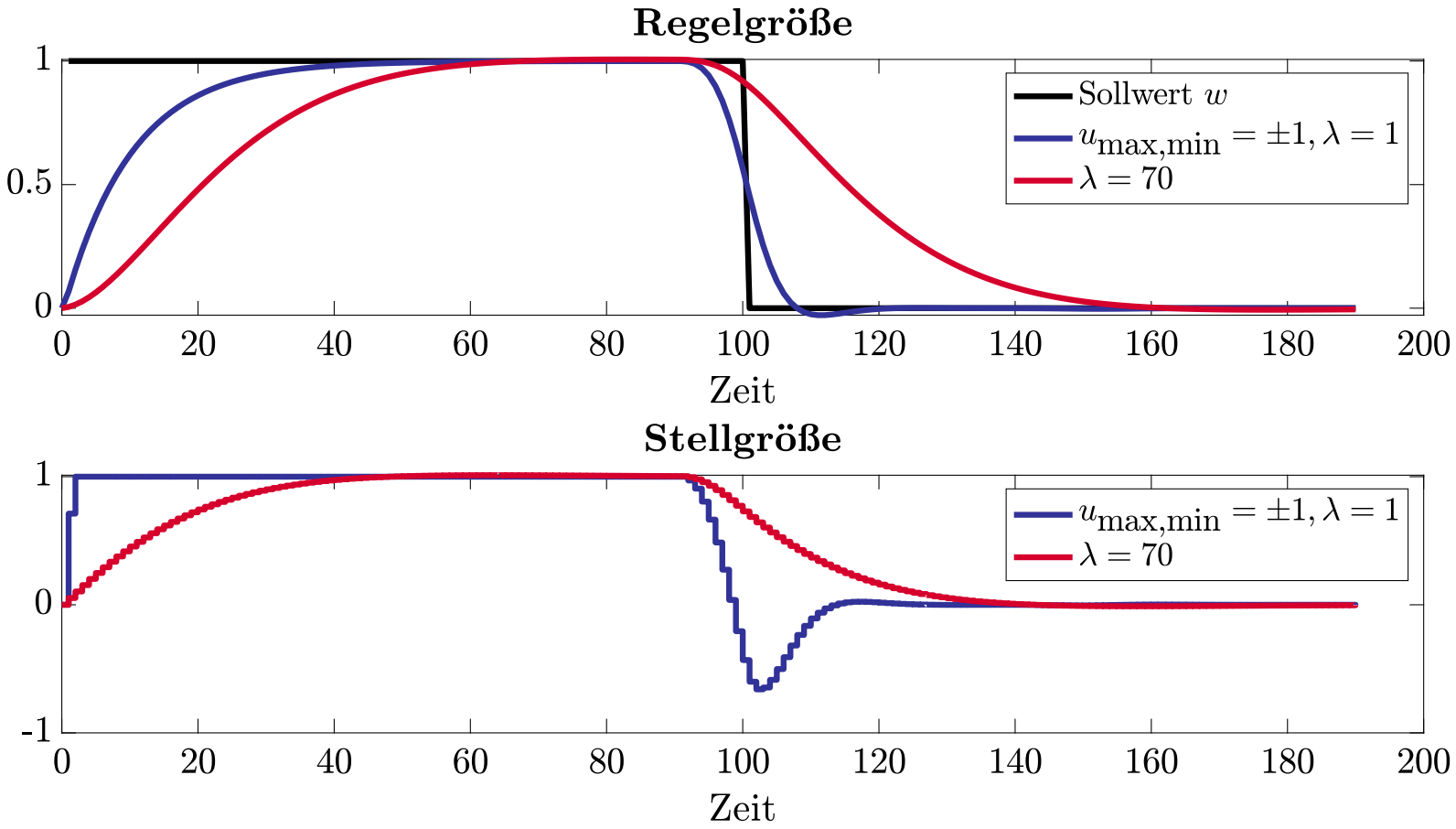
2. Fall: Die Stellgrößenbeschränkung liegt bei eins. So kann sichergestellt werden, dass der Aktor oder der Prozess nicht zerstört werden kann. Durch die Vermeidung von zu hohen Stellgrößen ist die Regelung aber langsam.

3. Fall: Die Stellgrößenbeschränkung ist höher, sodass die Regelung aggressiver ist und höhere Stellgrößen auf den Prozess gegeben werden können.

4.3 Nebenbedingungen in der Prädiktiven Regelung

Beispiel: Nebenbedingungen der Stellgrößenänderung vgl. mit hohem λ

$$G(s) = \frac{1}{10s+1},\ N_p = 10,\ N_u = 10,\ \lambda = 1,\ \text{vs. } \lambda = 70$$



4.3 Nebenbedingungen in der Prädiktiven Regelung

Beispiel: Nebenbedingungen der Stellgrößenänderung vgl. mit hohem λ

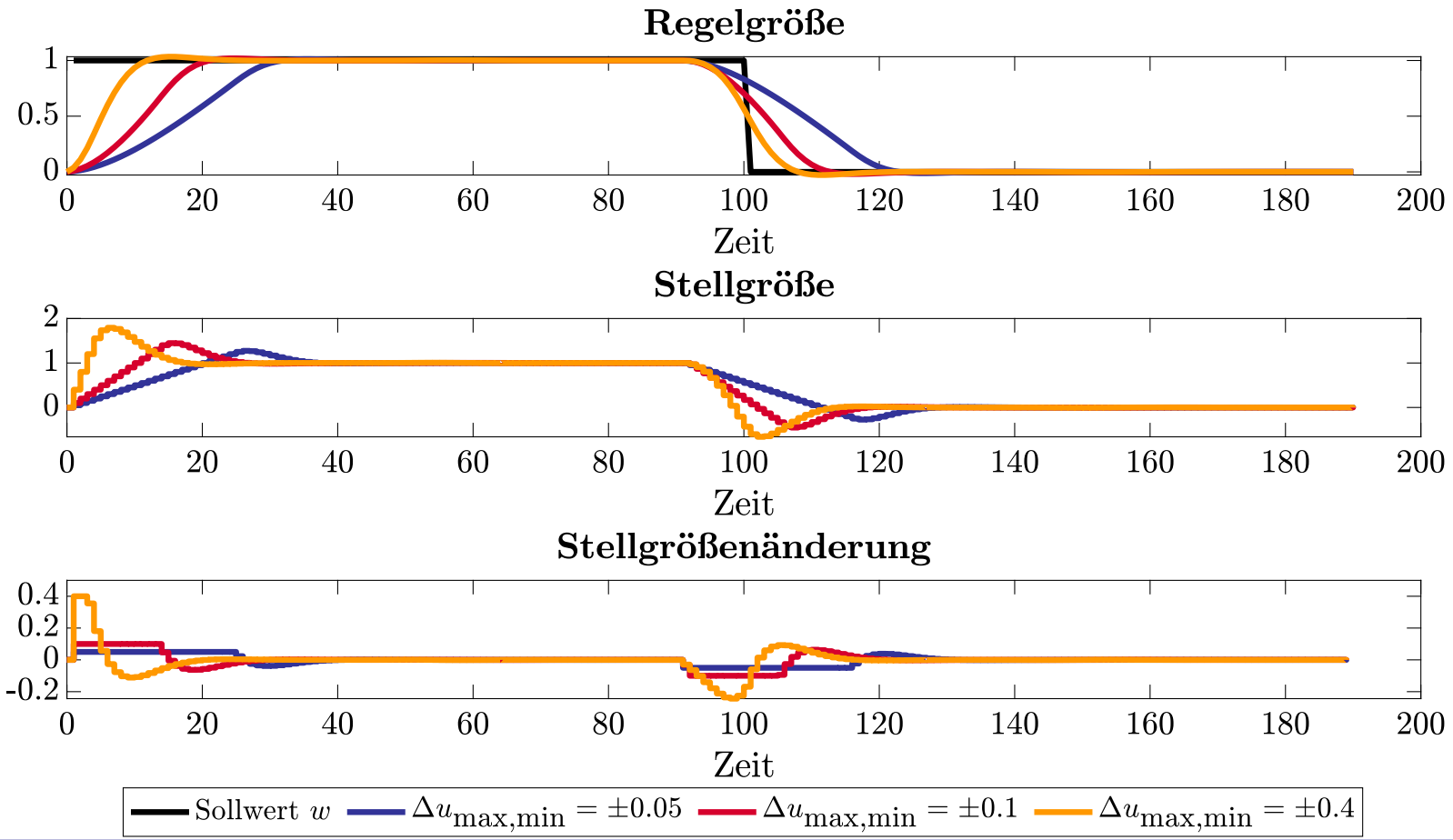
$$G(s) = \frac{1}{10s+1},\ N_p = 10,\ N_u = 10,\ \lambda = 1,\ \text{vs. } \lambda = 70$$

Die Beschränkung der Stellgröße kann ebenfalls durch ein sehr hohes λ umgesetzt werden. Dadurch wird allerdings der Regler langsamer und die Einhaltung ist nur ein *soft constraint*.

4.3 Nebenbedingungen in der Prädiktiven Regelung

Beispiel: Nebenbedingungen der Stellgrößenänderung

$$G(s) = \frac{1}{10s+1},\ N_p = 10,\ N_u = 10,\ \lambda = 1$$



4.3 Nebenbedingungen in der Prädiktiven Regelung

Beispiel: Nebenbedingungen der Stellgrößenänderung

$$G(s) = \frac{1}{10s+1},\ N_p = 10,\ N_u = 10,\ \lambda = 1$$

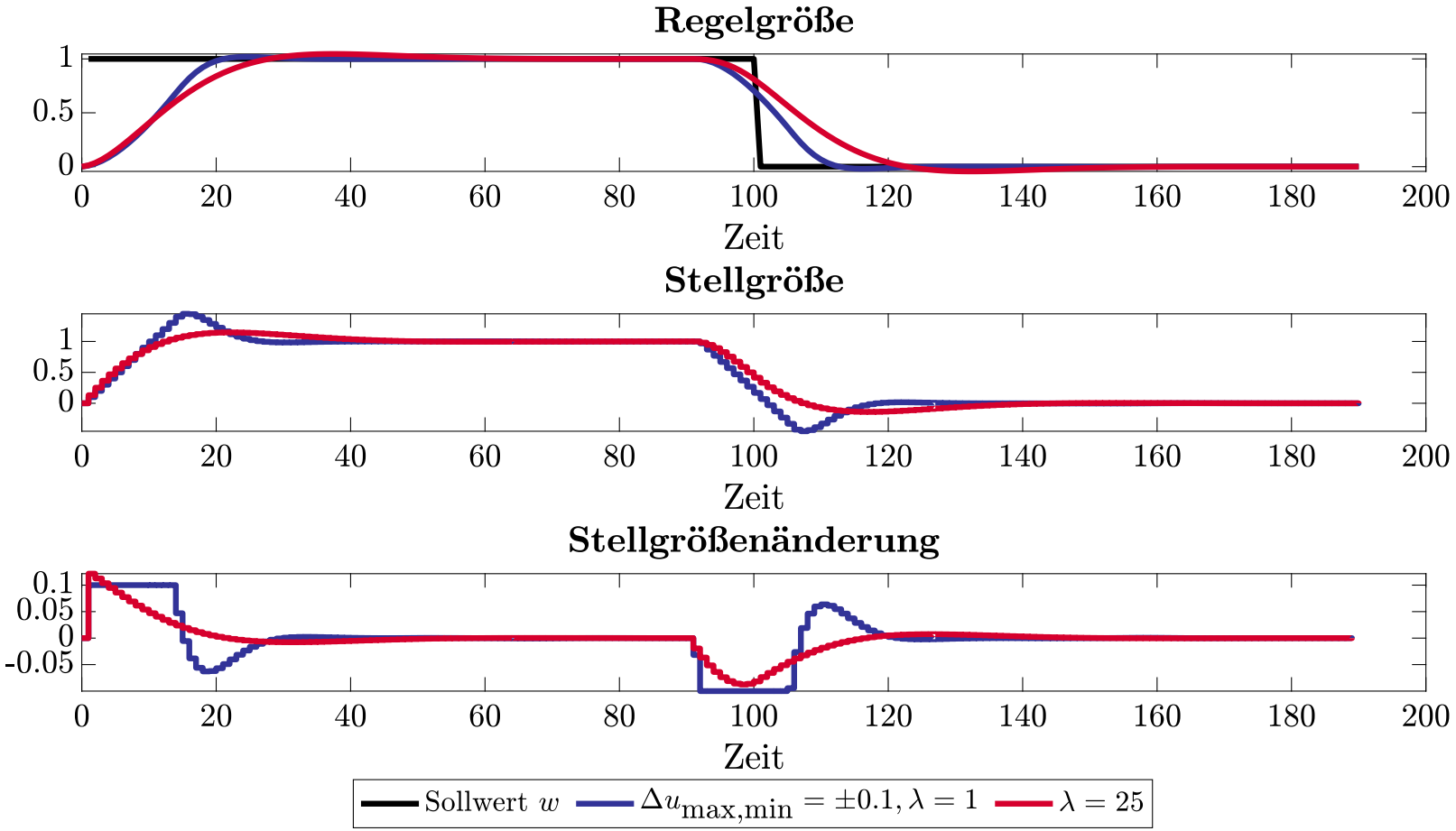
Durch die Nebenbedingungen der Stellgrößenänderung kann im MPC zwischen einer langsamen oder aggressiven Regelung gewählt werden. Dabei verhält sich der MPC ähnlich wie bei der Einstellung von λ, mit dem Unterschied, dass auch harte Grenzen vom Regler immer eingehalten werden. So kann die Änderungsrate der Stellgröße ebenfalls im Aktor limitiert sein, da dieser sehr verschleißanfällig ist.

Durch λ können ähnliche Regler Geschwindigkeiten eingestellt werden, wie bei der Beschränkung der Stellgrößenänderung. Ein Beispiel hierfür ist auf der nächsten Folie abgebildet. Mit Stellgrößenbeschränkungen kann der Regler aber komplett an die Grenzen gehen und ist daher etwas schneller.

4.3 Nebenbedingungen in der Prädiktiven Regelung

Beispiel: Nebenbedingungen der Stellgrößenänderung

$$G(s) = \frac{1}{10s+1},\ N_p = 10,\ N_u = 10,\ \lambda = 1,\ \text{vs. } \lambda = 25$$



4.4 Nichtlineare Prädiktive Regelung

Wenn ein nichtlineares Modell zur Prädiktion verwendet wird, oder nichtlineare Nebenbedingungen beachtet werden müssen handelt es sich um eine *nichtlineare prädiktive Regelung*.

Eigenschaften:

- Bessere Regler-Performance (bei nichtlinearen Prozessen)
- Weniger Robust
- Lokale Optima (Verlustfunktion nicht konvex)
- Hoher Rechenaufwand durch nichtlineare Optimierung
  - Wird vor allem für langsame Prozesse eingesetzt
  - Mit steigender Rechenleistung in immer mehr Prozessen umsetzbar

Kompromiss mit NEPSAC:

1. Berechnung einer Basis-Systemantwort mit dem nichtlinearen Modell (1. Iteration keine Änderung der Stellgröße)
2. Optimierung der Stellgrößen mit linearem Modell (QP)
3. Prädiktion der optimierten Stellgrößen mit nichtlinearem Modell und Addition auf Basis-Systemantwort
4. Wiederholung von Schritt 2-3 bis Algorithmus konvergiert

5. Optimierung: Nichtlinear in den Parametern

Prof. Dr.-Ing.
Oliver Nelles

Inhalt Kapitel 5

5 Nichtlineare Optimierung in der prädiktiven Regelung

Zielsetzung: Minimierung der Verlustfunktion

$$\min_{u(k),\cdots,u(k+N_u-1)} \sum_{i=1}^{N_p} \left(w(k+i) - \hat{y}(k+i \mid k)\right)^2 + \lambda \sum_{i=0}^{N_u-1} \Delta u(k+i)^2$$

Wenn das Modell und somit die Berechnung des Modellausgangs $\hat{y}$ *nichtlinear* ist oder *nichtlineare Nebenbedingungen* genutzt werden, handelt es sich um nicht mehr um ein QP. Somit muss eine nichtlineare Optimierung genutzt werden, um das Problem zu lösen. Dabei können beliebige nichtlineare Modelle ausgewählt werden.

Beispiele nichtlinearer Modelle:

- Physikalisch basierte Modelle
- Neuronale Netze
- …

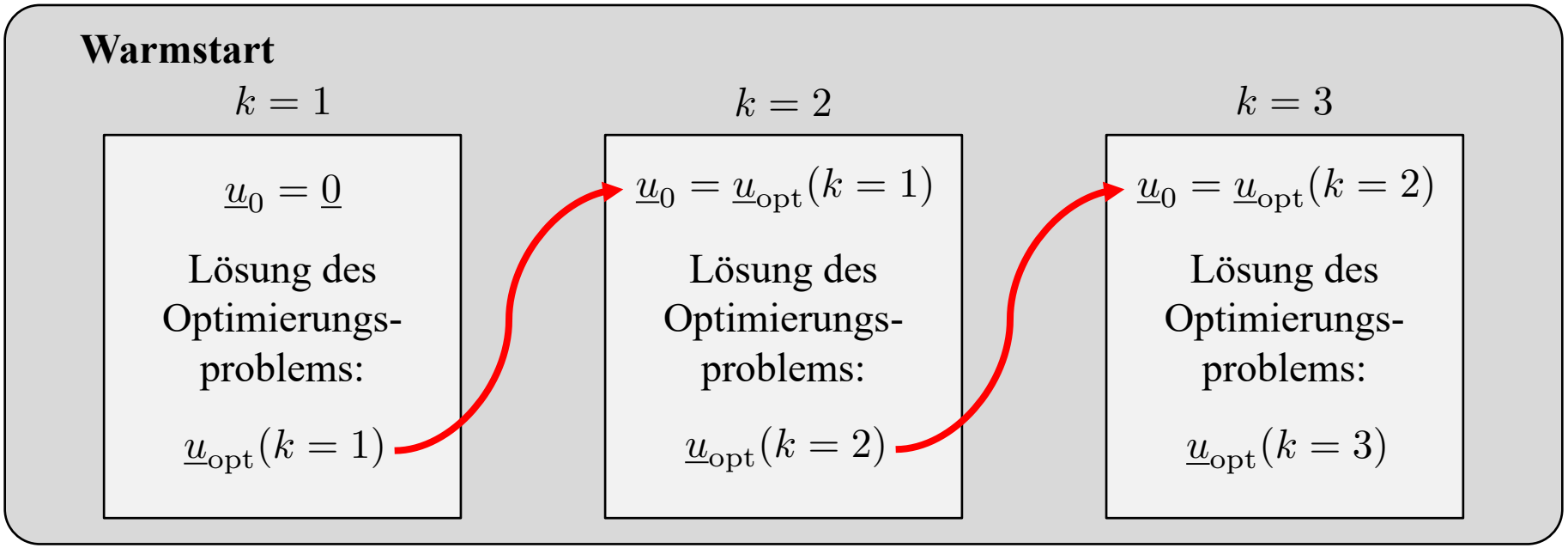
Je nach Modelltyp kann neben der Modellauswertung auch weitere Informationen wie z.B. der *Gradient* oder die *Hesse-Matrix* analytisch berechnet und in der Optimierung genutzt werden.

5 Nichtlineare Optimierung in der prädiktiven Regelung

Unterschied einer einmaligen Optimierung vs. Optimierung im MPC

Bei der wiederholten Optimierung in der prädiktiven Regelung können neu gewonnene Informationen genutzt werden z.B.:

- Warmstart: Nutzung des letzten Optimums als Initialisierung des Optimierungsproblems
- Informationen über die im letzten Zeitschritt aktiven Beschränkungen



5 Nichtlineare Optimierung in der prädiktiven Regelung

Abbruch der Optimierung

- Anzahl an maximaler Optimierungsiterationen erreicht
- Maximale Optimierungszeit erreicht
- Keine, bzw. lediglich geringe Verbesserungen der Gütefunktion
- Gradient sehr nahe bei 0
- Minimale Schrittweite erreicht

Abbruch der Optimierung in der Prädiktiven Regelung

Typischerweise wird bei der Anwendung von prädiktiven Reglern eine *Taktfrequenz* vorgegeben. Daher kann die Optimierung durch die *maximale Optimierungszeit* begrenzt werden und es kann nicht sichergestellt werden, dass zu einem Optimum konvergiert wurde.

5 Kangaroos for Optimization

Plate and Sarle give the following wonderful description of the most common nonlinear optimization algorithms with respect to neural networks (NN) in T. Plate 1993 (comments in brackets by the author):

Training a NN is a form of numerical optimization, which can be linked to a kangaroo searching the top of Mt. Everest. Everest is the *global optimum*, the highest mountain in the world, but the top of any other really tall mountain such as K2 (a good *local optimum*) would be satisfactory. On the other hand, the top of a small hill like Chapel Hill, NC, (a bad local optimum) would not be acceptable.

This analogy is framed in terms of maximization, while neural networks are usually discussed in terms of minimization of an error measure such as the least squares criterion, but if you multiply the error measure by *-1*, it works out the same. So in this analogy, the higher the altitude, the smaller the error.

The compass directions represent the values of synaptic weights *[parameters]* in the network. The north/south direction represents one weight, while the east/west direction represents another weight. Most networks have more than two weights, but representing additional weights would require a multi\-dimensional landscape, which is difficult to visualize. Keep in mind that when you are training a network with more than two weights, everything gets more complicated.

Initial weights are usually chosen randomly, which means that the kangaroo is dropped by parachute somewhere over Asia by a pilot who has lost the map. If you know something about the scales of input, you may be able to give the pilot adequate instructions to get the kangaroo to land near the Himalayas. However, if you make a really bad choice of distributions for the initial weights, the kangaroo may plummet into the Indian Ocean and drown.

With Newton-type (second order) algorithm *[with fixed step size $\eta = 1$]*, the Himalayas are covered with fog, and the kangaroo can only see a little way around her location *[first and second order derivative information]*. Judging from the local terrain, the kangaroo makes a

guess about where the top of the mountain is, assuming that the mountain has a nice, smooth, quadratic shape. The kangaroo then tries to leap all the way to the top in one jump.

Since most mountains do not have a perfect quadratic shape, the kangaroo will rarely reach the top in one jump. Hence, the kangaroo must *iterate*, i.e., jump repeatedly as previously described until she finds the top of the mountain. Unfortunately, there is no assurance that this mountain will be Everest.

In a stabilized Newton algorithm *[with variable step size η]*, the kangaroo has an altimeter, and if the jump takes her to a lower point, she backs up to where she was and takes a shorter jump. If ridge stabilization *[the Levenberg-Marquardt idea]* is used, the kangaroo also adjusts the direction of her jump to go up a steeper slope. If the algorithm isn't stabilized, the kangaroo may mistakenly jump to Shanghai and get served for dinner in a Chinese restaurant *[divergence]*.

In steepest ascent with line search, the fog is *very* dense, and the kangaroo can only tell, which direction leads up most steeply *[only first order derivative information]*. The kangaroo hops in this direction until the terrain starts going down. Then the kangaroo looks around again for the new steepest ascent direction and iterates.

Using an ODE (ordinary differential equation) solver is similar to steepest ascent, except that kangaroo crawls on all fives to the top of the nearest mountain, being sure to crawl in steepest direction at all times.

The following description of conjugate gradient methods was written by Tony Plate (1993):

"The environment for conjugate gradient search is just like that for steepest ascent with line search -- the fog is dense and the kangaroo can only tell, which direction leads up. The difference is that the kangaroo has some memory of the direction it has hopped in before, and the kangaroo assumes that the ridges are straight (i.e., the surface is quadratic). The kangaroo chooses a direction to hop that is upwards, but that does not result in it going downwards in the previous directions it has hopped in. That is, it chooses an upwards direction, moving along which will not undo the work of previous steps. It hops upwards until the terrain starts going down again, then chooses another direction."

In standard backprop, the most common NN training method, the kangaroo is blind and has to feel around on the grounds to make a guess about, which way is up. If the kangaroo ever gets near the peak, she may jump back and forth across the peak without ever landing on it. If you use a decaying step size, the kangaroo gets tired and makes smaller and smaller hops, so if she ever gets near the peak she has a better chance to actually landing on it before the Himalayas erode away. In backprop with momentum the kangaroo has poor traction and can't make sharp turns. With online training, there are frequent earthquakes, and mountains constantly appear and disappear. This makes it difficult for the blind kangaroo to tell whether she has ever reached the top of a mountain, and she has to take small hops to avoid falling into the gaping chasms that can open up at any moment.

Notice that in all the methods discussed so far, the kangaroo can hope at best to find the top of a mountain close to where she starts. In other words these are *local ascent* methods. There's no guarantee that this mountain will be Everest, or even a very high mountain. Many methods exist to try to find the global optimum.

In simulated annealing, the kangaroo is drunk and hops around randomly for a long time. However, she gradually sobers up and the more sober she is, the more likely she is to hop up hill *[temperature decreases according to the annealing schedule]*.

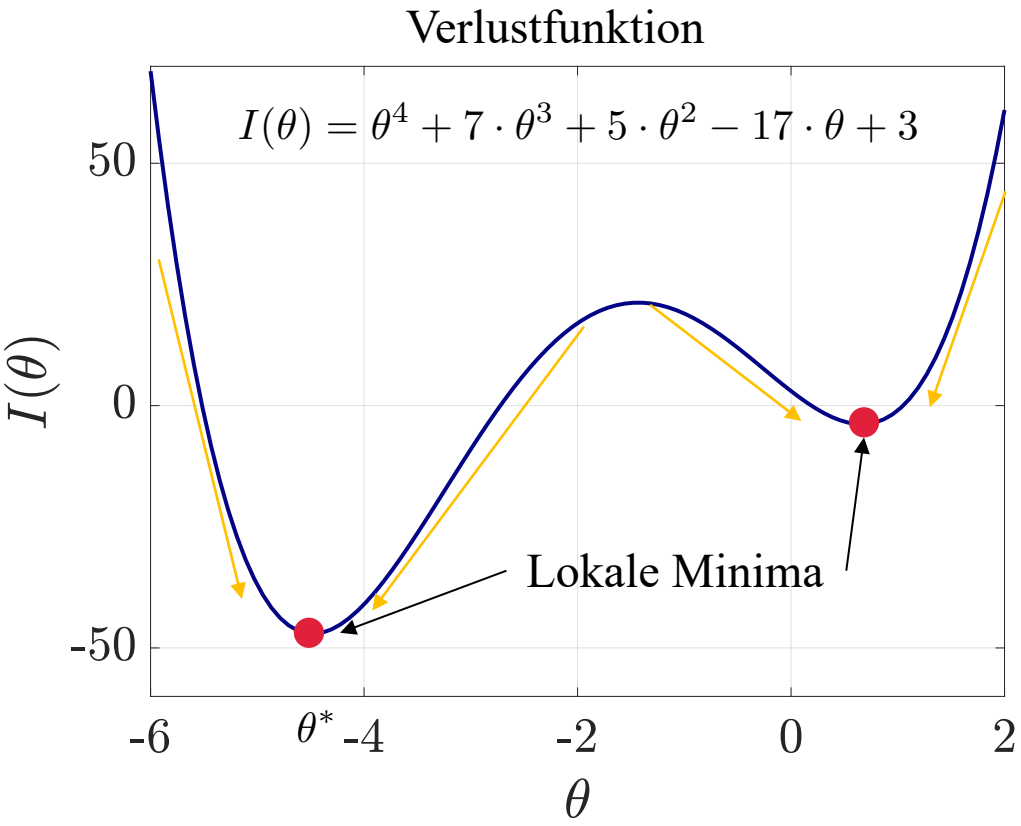
In a random multi-start method, lots of kangaroos are parachuted into the Himalayas at random places. You hope at least one of them will find Everest.

A genetic algorithm begins like random multi-start. However, these kangaroos do not know that they are supposed to be looking for the top of a mountain. Every few years, you shoot the kangaroos at low altitudes and hope that the ones that are left will be fruitful, multiply, and ascend. Current research suggests that fleas may be more effective than kangaroos in genetic algorithms, since their faster rate of reproduction more than compensates for their shorter hops *[crossover is more important than mutation]*.

A tunneling algorithm can be applied in combination with any local ascent method but requires divine intervention and a jet ski. The kangaroo first finds the top of any nearby mountain. Then the kangaroo calls upon her deity to flood the earth to the point that the waters just reach the top of the current mountain. She get on her ski, goes off in search of a higher mountain, and repeats the process until no higher mountains can be found.

T. Plate "Re: Kangaroos (Was Re: BackProp without Calculus?)", Usenet article <93Sep8.162519edt.997@neuron.ai.toronto.edu> in comp.ai.neural-nets, 1993.

5 Einführendes Beispiel



- Die gezeigte Verlustfunktion ist nichtlinear von dem Parameter θ abhängig.
- Gesucht wird ein Algorithmus / ein Verfahren, dass ein möglichst gutes Minimum findet, idealerweise in diesem Beispiel das globale Minimum $\theta^* \approx -4.5$.
- Im Normalfall ist eine solche Verlustfunktion nicht nur von einem Parameter θ sondern von n Parametern abhängig. Diese werden meistens in dem Parametervektor

$$\underline{\theta} = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix}$$

zusammengefasst.

5 Einführendes Beispiel

Nichtlinear

- In diesem Kapitel befassen wir uns ausschließlich mit nichtlinearen Optimierungsmethoden. Das bedeutet, dass die Verlustfunktion $I(\underline{\theta})$ nichtlinear von den Parametern $\underline{\theta}$ abhängt und eine nichtlineare Optimierungsmethode nötig ist, um die optimalen Parameter $\underline{\theta}_{\mathrm{opt}}$ zu finden.

Lokal vs. Global

- Ein Großteil der vorgestellten Methoden sind *lokale* Optimierungsmethoden. Dies bedeutet, dass der Optimierungsalgorithmus an einem initialen Punkt mit der Optimierung beginnt und in der Nachbarschaft des initialen Punktes ein lokales Optimum sucht.
- Globale Optimierungsmethoden machen einen Tradeoff zwischen Exploration (globale Suche) und Exploitation (lokale Suche).

Direkte vs. Gradientenbasiert

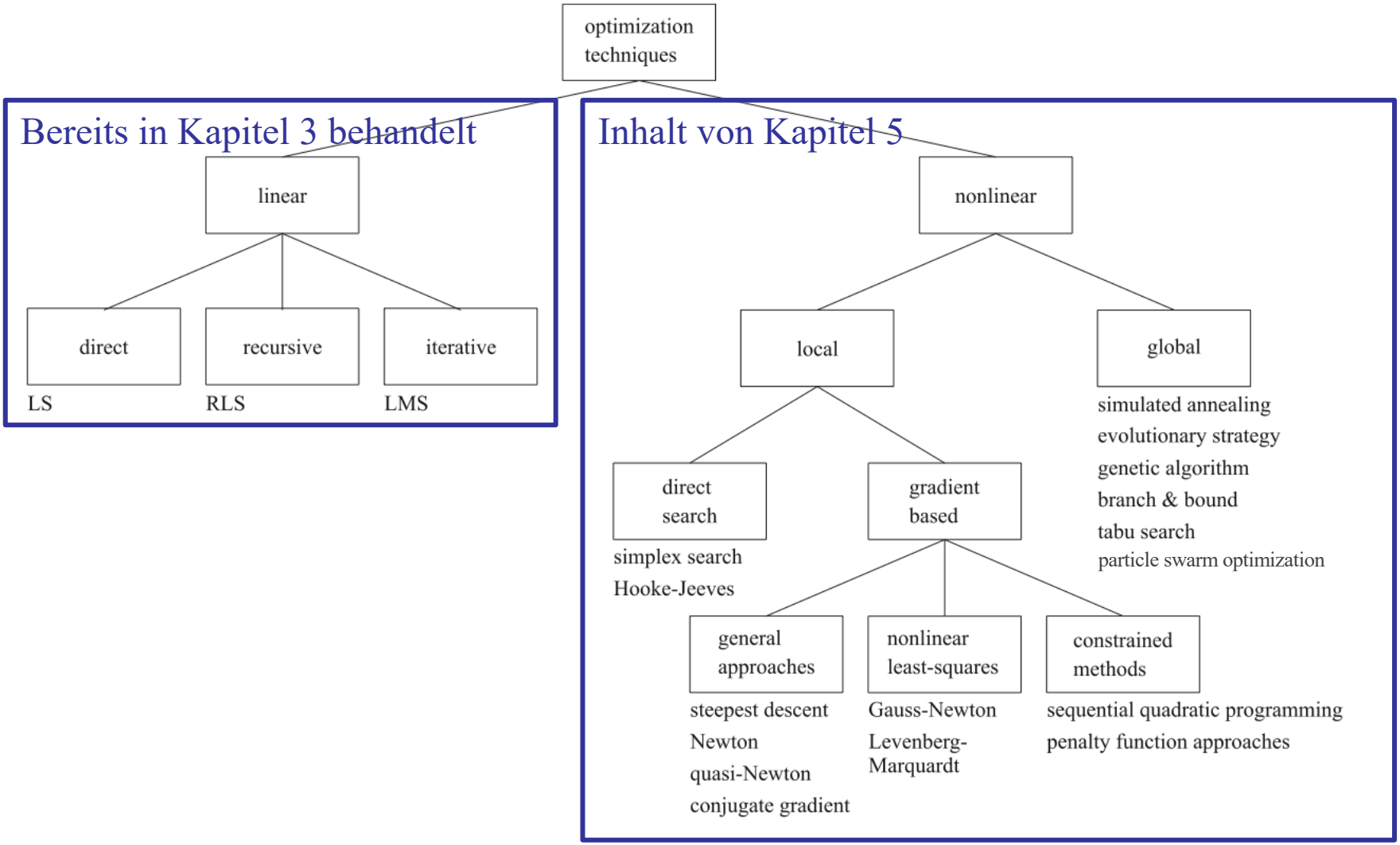
- Direkte Methoden suchen ein lokales Optimum ausschließlich mit der Hilfe von Verlustfunktionsauswertungen. Gradienten werden nicht benötigt.
- Gradientenbasierte Verfahren berechnen pro Iteration den Gradient und berechnen das Parameterupdate durch einen (möglicherweise) korrigierten Schritt in die Richtung des negativen Gradienten (steilster Abstieg).

5 Eigenschaften nichtlinearer Optimierungsprobleme

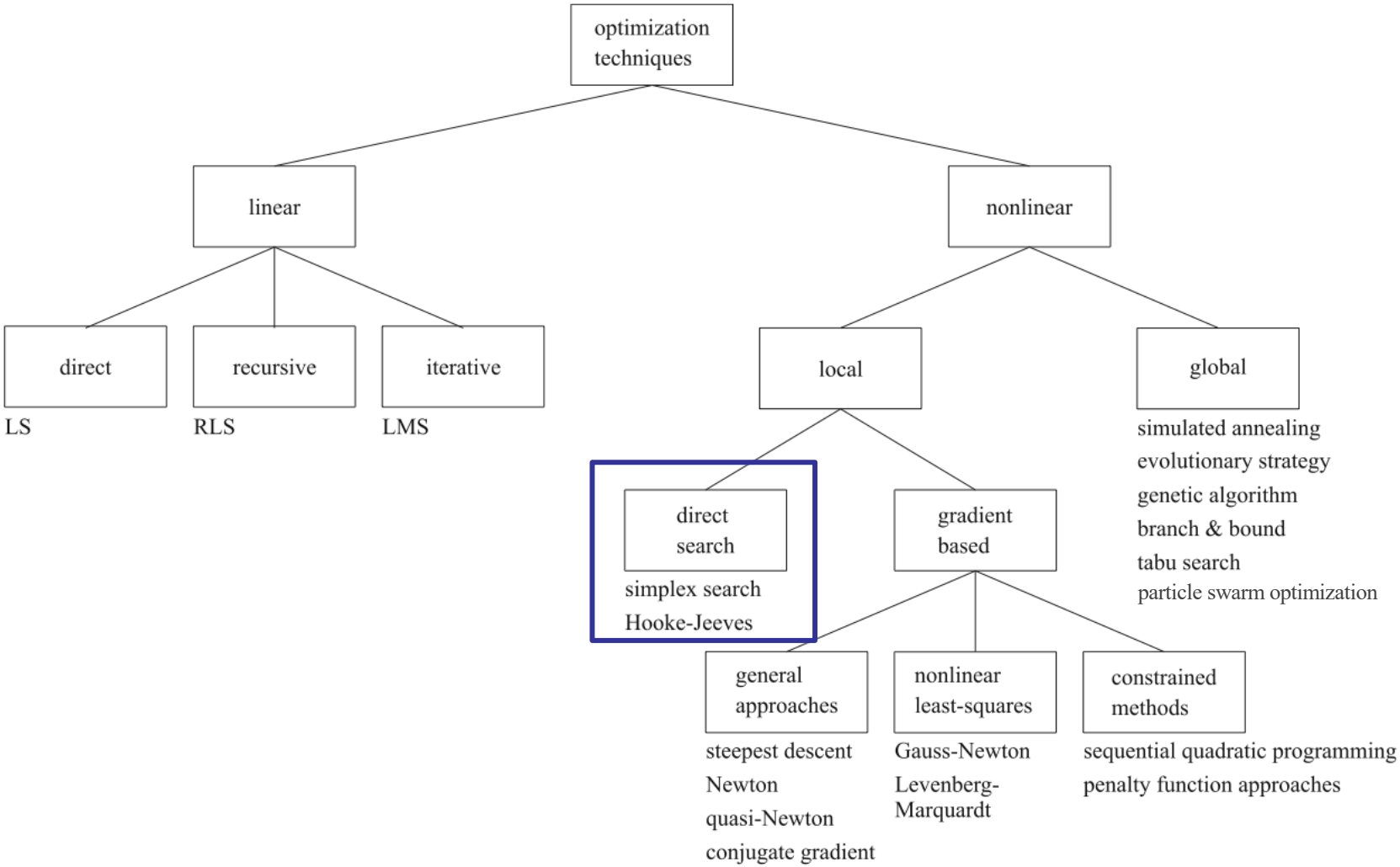
Oft anzutreffende Eigenschaften nichtlinearer Optimierungsprobleme

- Viele lokale Optima
- Die Region rund um ein lokales Optimum kann durch eine Parabel der Form $\frac{1}{2}\underline{\theta}^T \underline{\underline{H}}\, \underline{\theta} + \underline{g}^T \underline{\theta} + c$ angenähert werden (Taylor Reihenentwicklung zweiter Ordnung)
- Es existiert keine analytische Lösung
- Ein iterativer Algorithmus wird benötigt
- Online Verwendung sehr schwierig

5 Übersicht von Optimierungsmethoden



5.1 Suchverfahren



5.1 Suchverfahren

Suchverfahren verwenden zur Schätzung des Optimums ausschließlich Auswertungen der Verlustfunktion. Informationen über den Gradienten werden nicht benötigt und nicht berücksichtigt.

Anwendungsfall

- Probleme, bei denen der Gradient nicht zur Verfügung steht oder schwierig zu berechnen ist.

Vorteile

- Einfach zu verstehen und zu implementieren

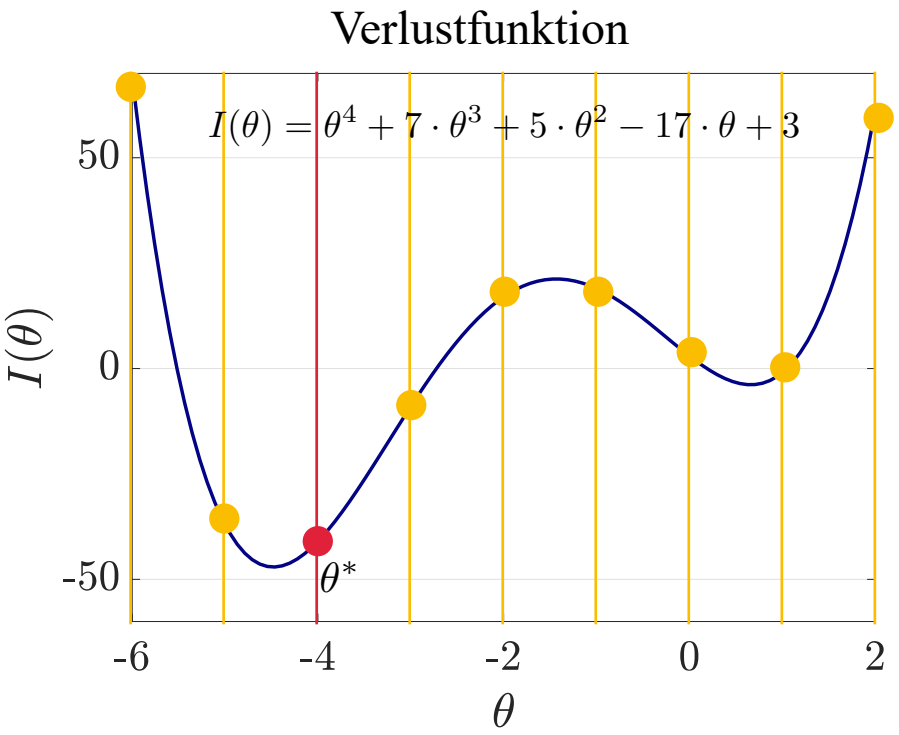
Nachteile

- Oft langsame Konvergenz

5.1 Suchverfahren

Grid-Search

- Der Eingangsraum (in dem Beispiel eindimensional) wird an einer definierten Anzahl Punkten N mit gleichem Abstand gerastert.
- Die Verlustfunktion wird an den Rasterpunkten ausgewertet und derjenige ausgesucht, der den kleinsten Verlustfunktionswert liefert



Vorteile

- Sehr einfach zu implementieren und zu verstehen

Nachteile

- Fluch der Dimensionalität. Die Anzahl an Rasterpunkten steigt exponentiell mit der Anzahl der Eingangsdimensionen

5.1 Suchverfahren

Random Search

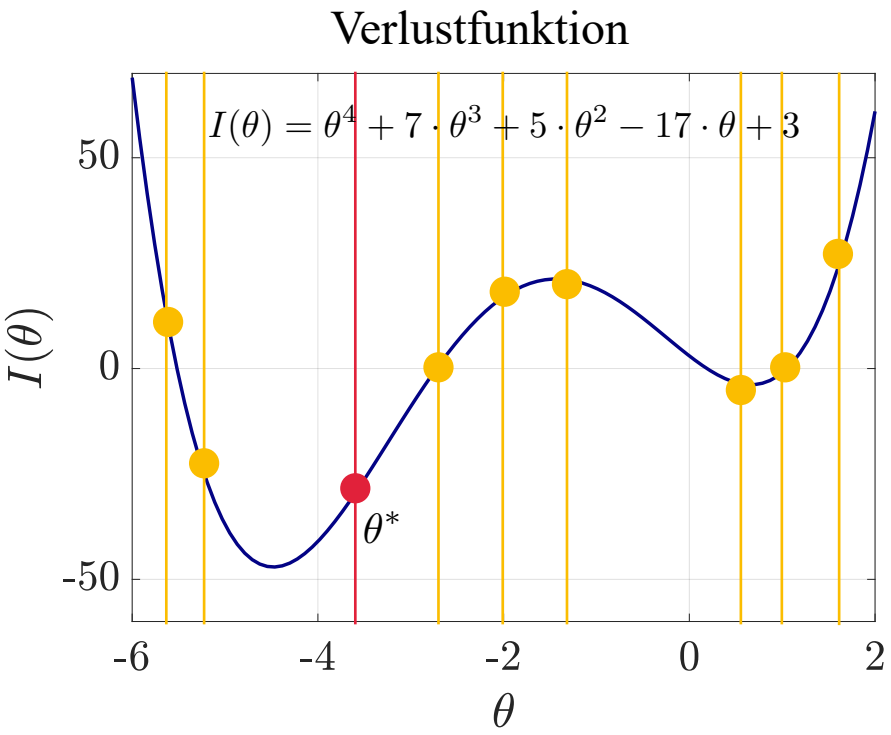
- Die Verlustfunktion wird an N zufällig ausgewählten Punkten ausgewertet.
- Der Punkt, der den geringsten Verlustfunktionswert liefert wird ausgewählt.

Vorteile

- Sehr einfach zu implementieren und zu verstehen

Nachteile

- Fluch der Dimensionalität. Um eine ähnlich gute Raumabdeckung zu erhalten muss die Anzahl Auswertungspunkte exponentiell mit der Anzahl Eingangsdimensionen ansteigen.

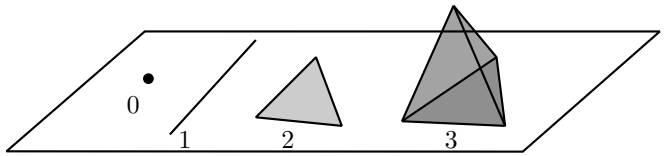


5.1 Suchverfahren

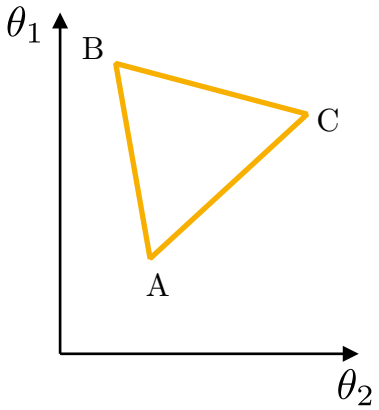
Downhill-Simplex-Verfahren (Nelder-Mead-Verfahren)

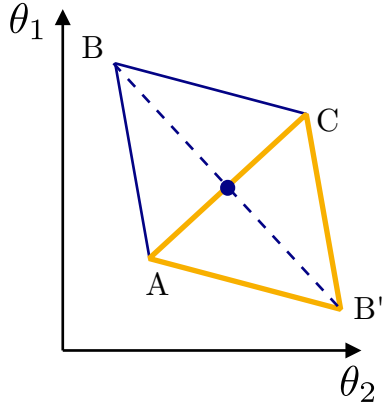
- Das Downhill-Simplex-Verfahren hat nichts mit dem Simplex-Verfahren der linearen Programmierung zu tun.
- Ein Simplex ist die Generalisierung eines Dreiecks (für zwei Dimensionen) zu einer beliebigen Anzahl Dimensionen.
- Die Verlustfunktion wird an den Ecken des Simplexes (A, B und C) ausgewertet. Die Ecke mit dem schlechtesten Verlustfunktionswert (in dem Beispiel Ecke B) wird am Schwerpunkt der anderen Eckpunkten gespiegelt, was zu dem neuen Punkt B` führt. Der neue Simplex besteht nun aus den Ecken A, C und B`.
- Das Verfahren wird so lange wiederholt, bis die Verlustfunktion nicht weiter minimiert werden kann.

Simplexe



2-Simplex im 2-dimensionalen Eingangsraum

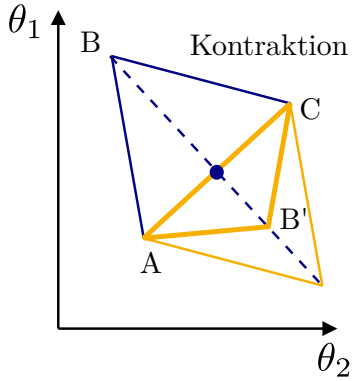


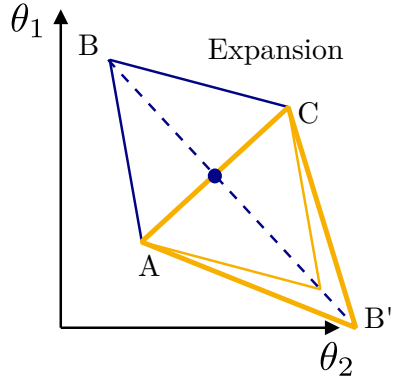


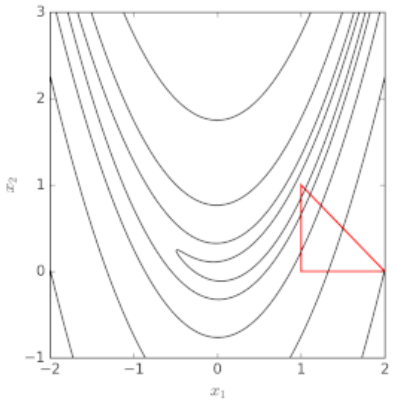
5.1 Suchverfahren

Downhill-Simplex-Verfahren oder Nelder-Mead-Verfahren

- Es wird nur eine Funktionsauswertung pro Iteration benötigt.
- Damit der Algorithmus aber tatsächlich konvergiert müssen noch ein paar besondere Fälle abgedeckt werden. Damit kein hin-und-her-springen zwischen zwei Simplexen entsteht, kann anstelle des schlechtesten der zweitschlechteste Eckpunkt gespiegelt werden oder die Größe des Simplexes reduziert werden.
- Nelder und Mead haben haben den Simplex Algorithmus so erweitert, damit folgende Probleme beseitigt werden
  1. Alle Richtungen (Parameter) werden mit einem Faktor skaliert, wenn die Simplex-Größe reduziert wird
  2. Die Größe des Simplexes kann nur reduziert werden, somit kann die Konvergenzgeschwindigkeit nicht erhöht werden
  3. Verkleinerung des Simplexes benötigt die Neuberechung aller Eckpunkte.

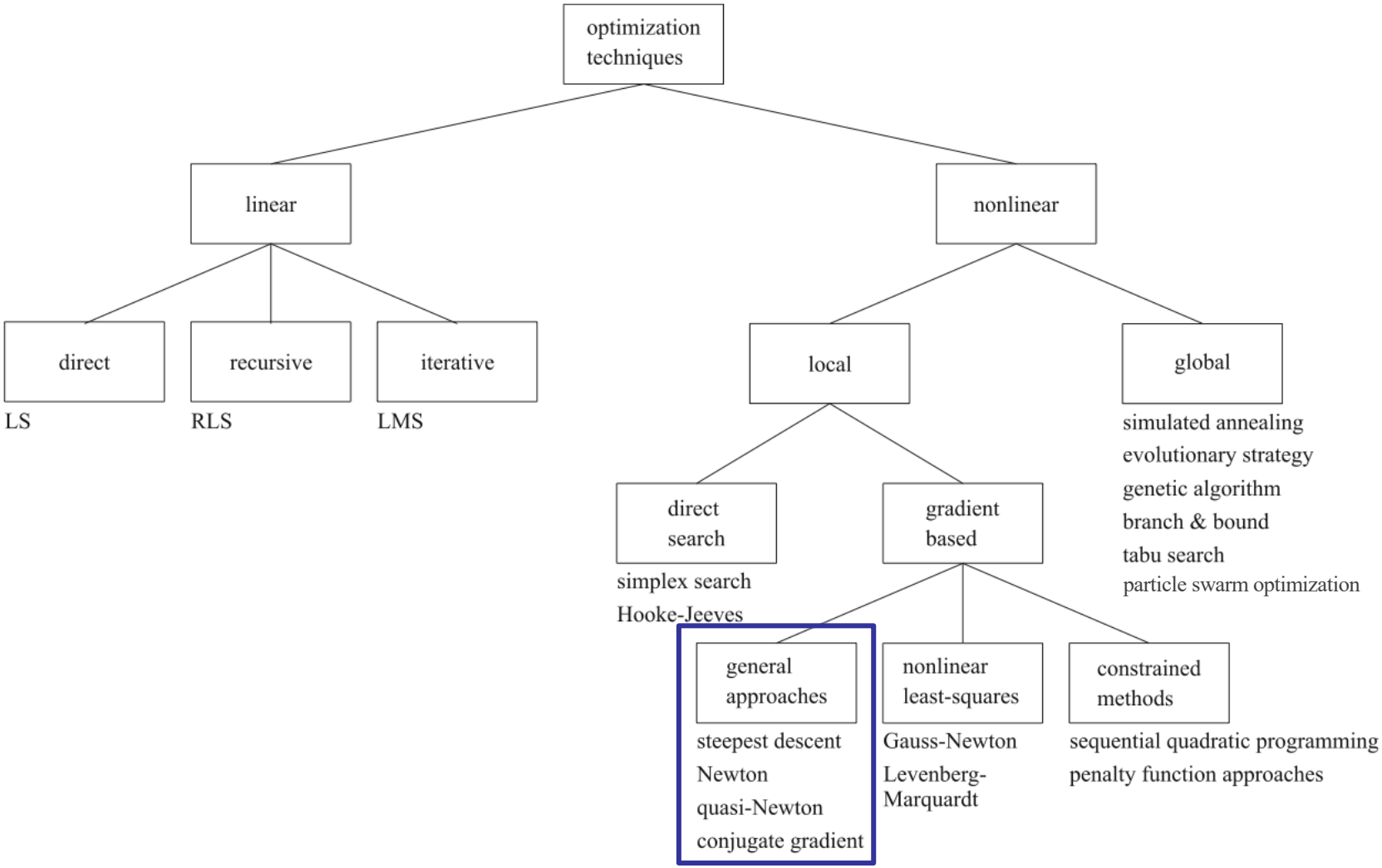






Quelle: https://en.wikipedia.org/wiki/Nelder–Mead_method

5.2 Gradientenverfahren



5.2 Gradientenverfahren

Gradientenbasierte Verfahren sind iterative Verfahren. Das heißt, dass der Parametervektor der nächsten Iteration $i+1$ eine korrigierte Version des aktuellen Parametervektors ist

$$\underline{\theta}_{i+1} = \underline{\theta}_i + \text{Korrektur}$$

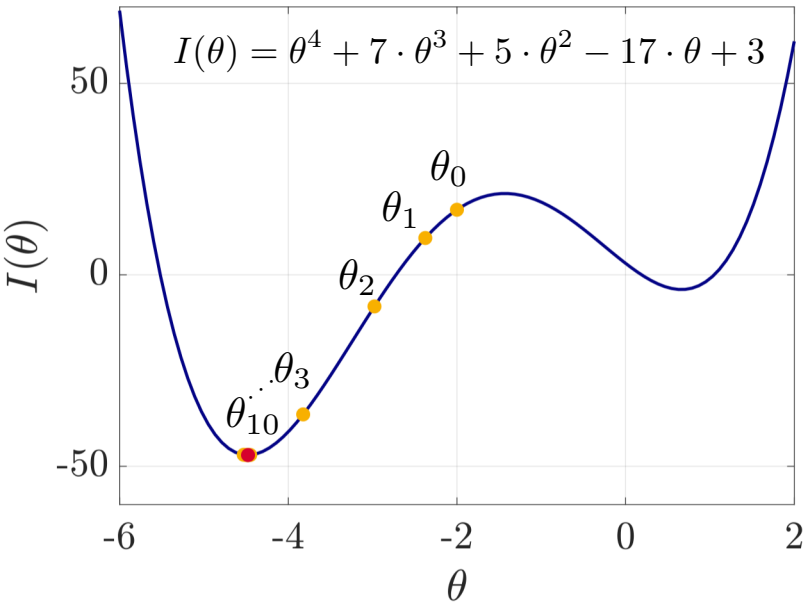
Der Gradient $\underline{g}_i = \frac{\partial I(\underline{\theta}_i)}{\partial \underline{\theta}_i}$ beinhaltet Informationen, wie die Verlustfunktion im direkten Umfeld von $\underline{\theta}_i$ „geformt“ ist. Da wir eine Verlustfunktion minimieren wollen, kann man den aktuellen Parametervektor um einen Schritt in die Richtung des negativen Gradienten korrigieren (der Gradient zeigt in die Richtung des steilsten Anstiegs!), was uns zur Update-gleichung

$$\underline{\theta}_{i+1} = \underline{\theta}_i - \eta \underline{g}_i$$

führt, der sogenannten „steepest-descent method“.

Beispiel

Verlustfunktion



$$\eta = 0.025$$

$$g_i = \frac{\partial I(\theta_i)}{\partial \theta_i} = 4 \cdot \theta_i^3 + 21 \cdot \theta_i^2 + 10 \cdot \theta_i - 17$$

5.2 Gradientenverfahren

Eigenschaften

- Einfache Berechnung
- Gradient wird benötigt
- Lineare Konvergenz

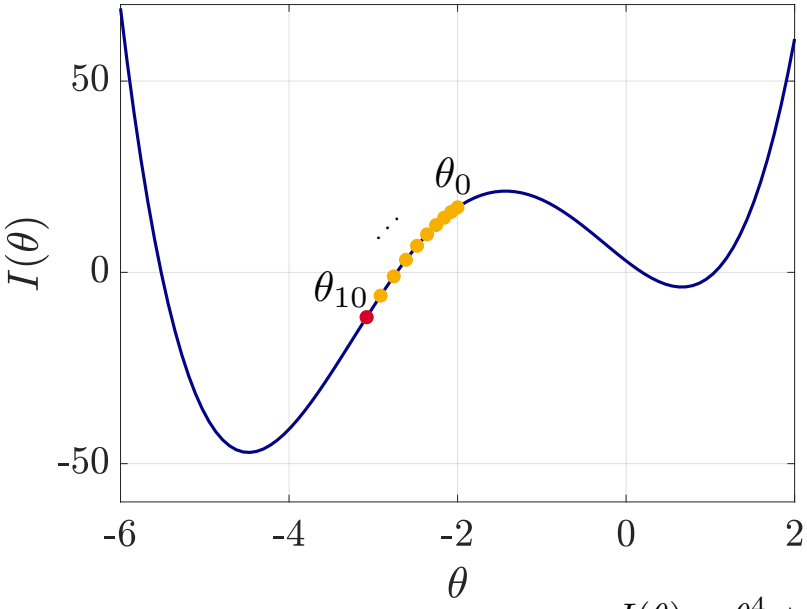
$$\frac{\|\theta_{i+1} - \theta_{\text{opt}}\|}{\|\theta_i - \theta_{\text{opt}}\|} < \beta_i, \beta_i \in (0,1)$$

5.2 Gradientenverfahren

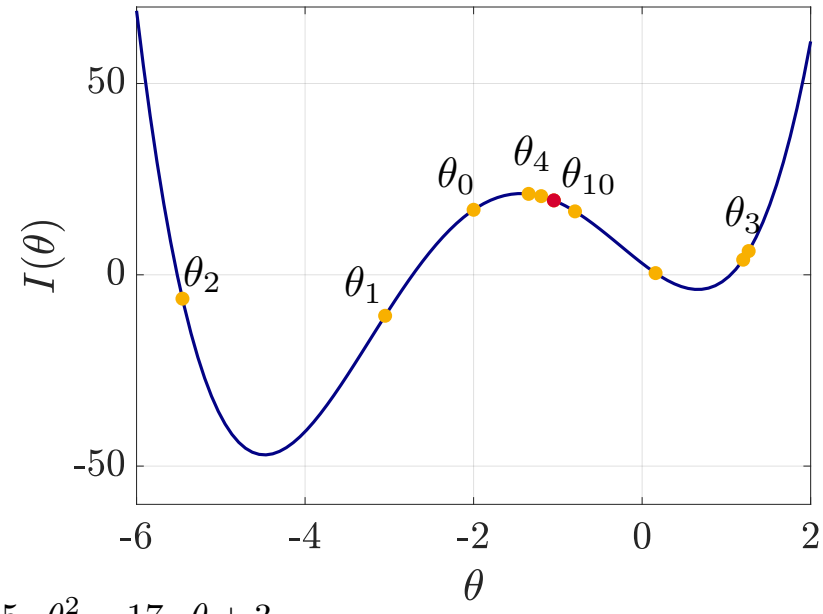
Steepest Descent - Wahl der Schrittweite

Ein kritischer Schritt ist die Wahl der Schrittweite η. Wird die Schrittweite zu klein gewählt konvergiert der Optimierer sehr langsam, wird die Schrittweite zu groß gewählt kann es passieren, dass der Optimierer in „schmale“ Täler nicht gut hineinlaufen kann und „oszillatorisches Verhalten“ zeigt oder sogar divergiert.

Zu kleine Schrittweite $\eta = 0.005$



Zu große Schrittweite $\eta = 0.07$



$$I(\theta) = \theta^4 + 7 \cdot \theta^3 + 5 \cdot \theta^2 - 17 \cdot \theta + 3$$

5.2 Gradientenverfahren

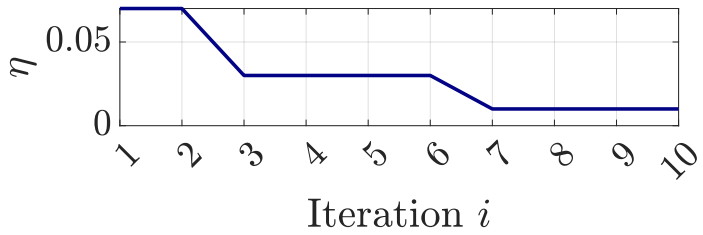
Lernrate (Machine Learning) = Schrittweite

Steepest Descent - Wahl der Schrittweite

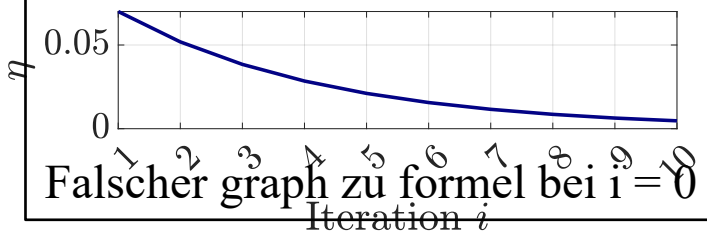
Ein beliebtes Mittel, um den Einfluss der Schrittweite besser zu kontrollieren ist das sogenannte „Lernraten Scheduling“. Die Schrittweite wird mit steigender Anzahl Iterationen verkleinert, damit die Schrittweite zunehmend kleiner wird.

$$\underline{\theta}_{i+1} = \underline{\theta}_i - \eta_i \underline{g}_i$$

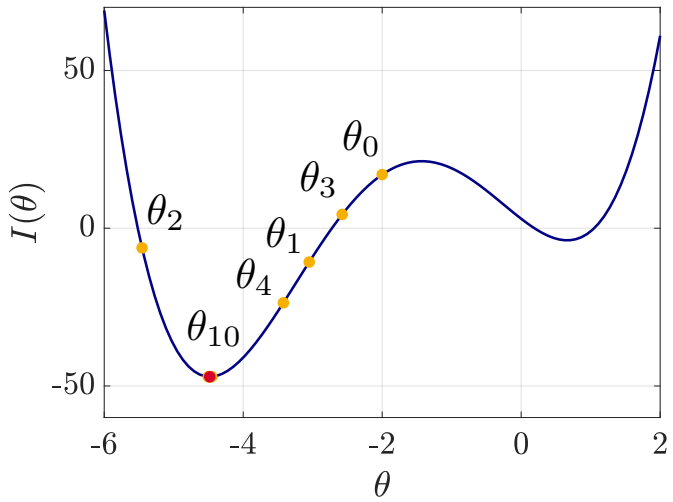
Step Decay

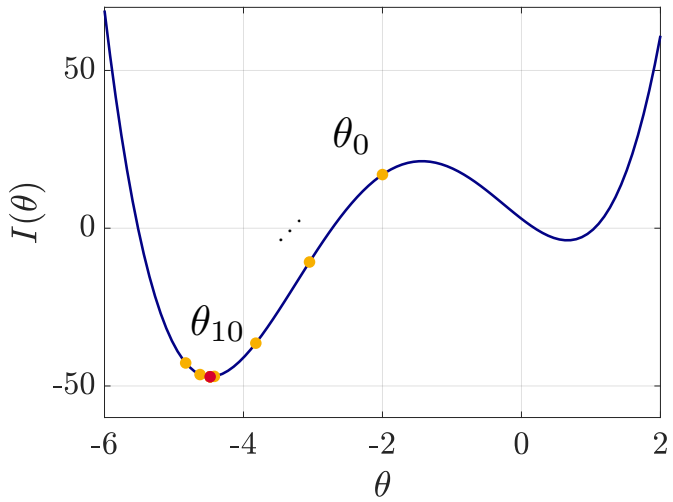


Exponential Decay $\eta_i = 0.07 \cdot e^{-0.3 \cdot i}$



Falscher graph zu formel bei i = 0

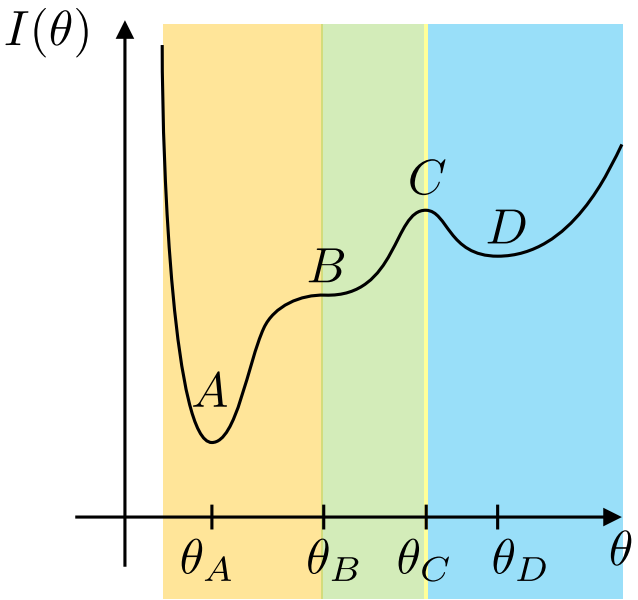




5.2 Gradientenverfahren

Initiale Parameter

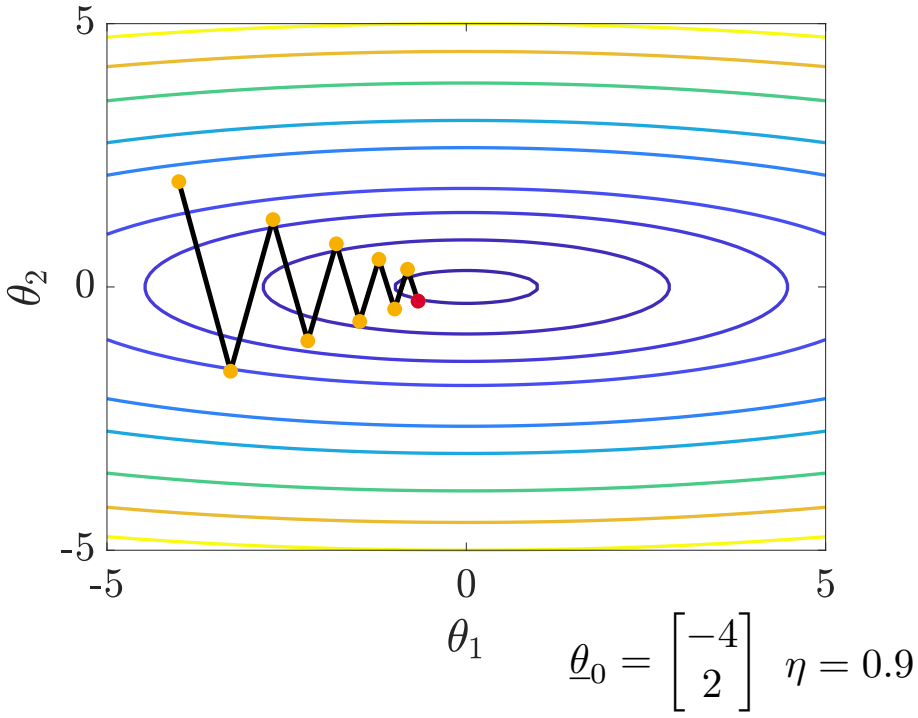
Nichtlineare Optimierungsmethoden sind iterative Verfahren. In der ersten Iteration $i = 0$ muss ein initialer Parametervektor $\underline{\theta}_0$ bestimmt werden. Wenn das Optimierungsproblem physikalischer Natur ist, kann oft durch Expertenwissen ein guter Startwert bestimmt werden. Der Startpunkt hat sehr großen Einfluss auf die Konvergenzgeschwindigkeit sowie die Qualität des lokalen Optimums, das gefunden wird.



5.2 Gradientenverfahren

Zig-Zagging

Bei schlecht konditionierten Optimierungsproblemen (Probleme mit großem Unterschied zwischen kleinstem und größtem Eigenwert der Hesse-Matrix) kommt es bei der steepest-descent Methode zum sogenannten „Zig-Zagging“. Dies liegt daran, dass die Richtung des Gradienten nur an wenigen Stellen in die Richtung des Minimums zeigt.



$$\underline{\theta}_0 = \begin{bmatrix} -4 \\ 2 \end{bmatrix} \quad \eta = 0.9$$

$$I(\underline{\theta}) = \begin{bmatrix} \theta_1 & \theta_2 \end{bmatrix} \underbrace{\begin{bmatrix} 0.1 & 0 \\ 0 & 1 \end{bmatrix}}_{\underline{H}} \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}$$

Einen tollen interaktiven Blog Eintrag zu diesem Thema und wie Momentum helfen kann gegen „Zig-Zagging“ vorzugehen gibt es hier: https://distill.pub/2017/momentum/

5.2 Gradientenverfahren

Allgemeine Formulierung

Gradientenbasierte Optimierungsverfahren lassen sich alle als

$$\underline{\theta}_{i+1} = \underline{\theta}_i - \eta_i \underbrace{\underline{R}_i \underline{g}_i}_{\underline{p}_i}$$

schreiben. Hier ist η_i die Schrittweite und $\underline{p}_i$ die Suchrichtung. Für alle positiv definiten Matrizen $\underline{R}_i$ sorgt diese Updategleichung für eine Minimierung der Verlustfunktion

$$I(\underline{\theta}_{i+1}) < I(\underline{\theta}_i)\,.$$

Die Matrix $\underline{R}_i$verändert die Richtung und Länge des Gradienten $\underline{g}_i$. Ersetzt man nun unterschiedliche Teile dieser Updategleichung gelangt man zu unterschiedlichen Algorithmen:

- Steepest Descent: $\underline{R}_i = \underline{I}$(Einheitsmatrix)
- Newton Methoden: $\eta_i \underline{R}_i = \underline{H}_i^{-1}$
- Quasi-Newton Methoden: $\eta_i \underline{R}_i = \hat{\underline{H}}_i^{-1}$
- Konjugiertes Gradientenverfahren: $\underline{p}_i = \underline{g}_i - \beta_i \underline{p}_{i-1}$

5.2 Gradientenverfahren

Nonlinear Least Squares

Wenn die Verlustfunktion aus der Summe der Fehlerquadrate besteht

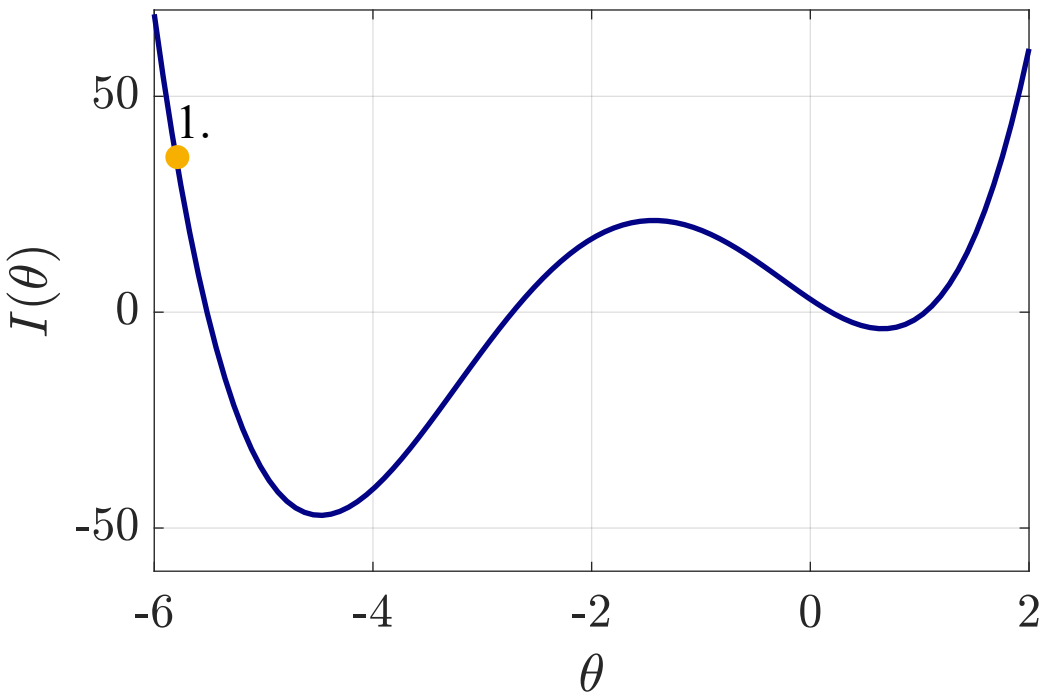
$$I(\underline{\theta}) = \sum_{i=1}^{N} e^2(i, \underline{\theta}) \,,$$

dann ist eine spezielle Klasse an Lösungsverfahren verwendbar. Ist der Fehler $e(i, \underline{\theta})$ linear von den Parametern $\underline{\theta}$ abhängig, kann *Least Squares* verwendet werden (siehe Kapitel 3). Ist der Fehler nichtlinear von $\underline{\theta}$ abhängig, dann handelt es sich um ein *Nonlinear Least Squares* Problem.

Zur Lösung dieser Problemklasse können der Gauss-Newton (Newton Verfahren auf *Nonlinear Least Squares* Problem angewandt) und Levenberg-Marquardt Algorithmus (Mischung aus steepest descent und Gauss-Newton) eingesetzt werden. Sie nutzen effiziente Matrix-Vektor-Schreibweise und Approximationen der Hesse-Matrix, die die spezielle Form der Verlustfunktion ausnutzt, zur Optimierung von $\underline{\theta}$.

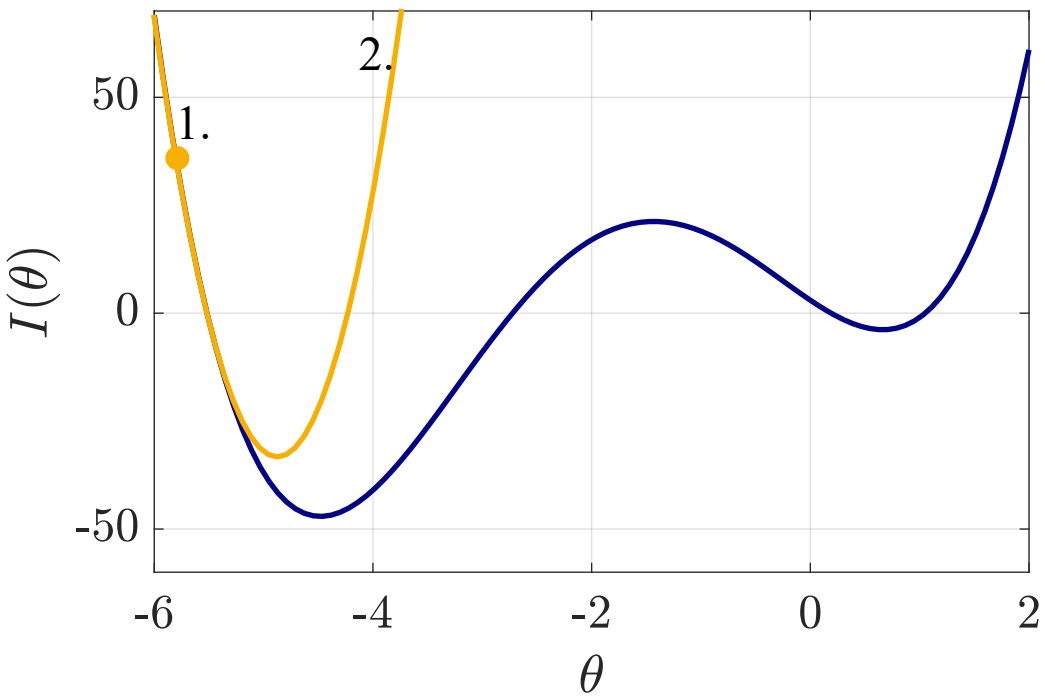
5.3 Newton-Verfahren

1. Wähle initialen Parameter θ_0



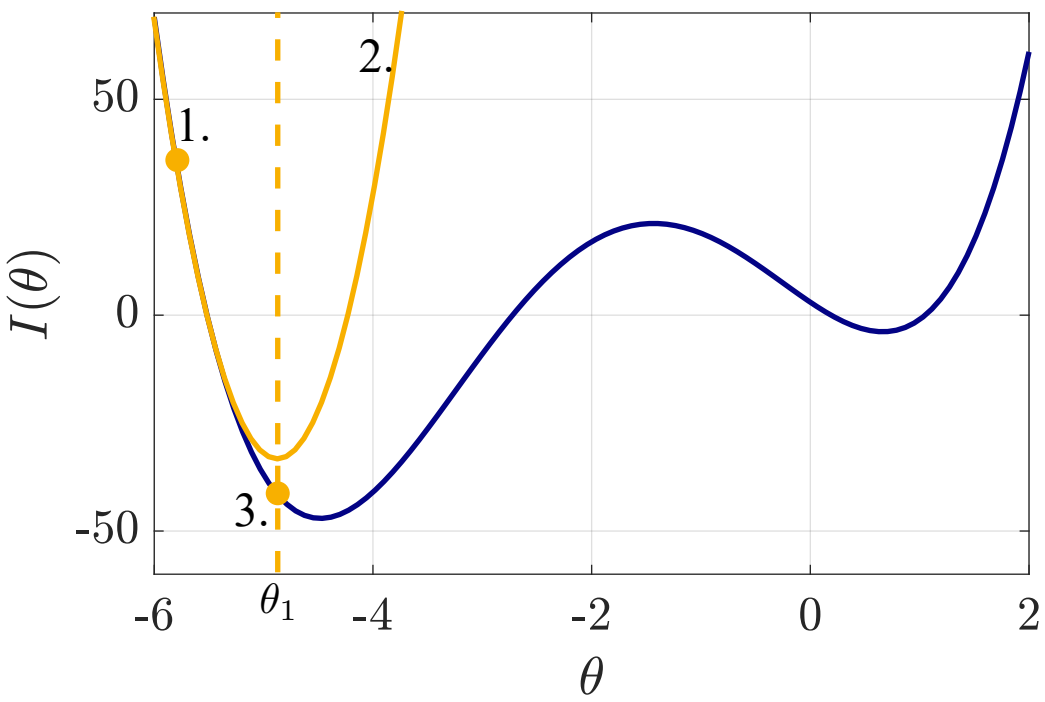
5.3 Newton-Verfahren

1. Wähle initialen Parameter θ_0
2. Bilde quadratische Approximation um $I(\theta_0)$



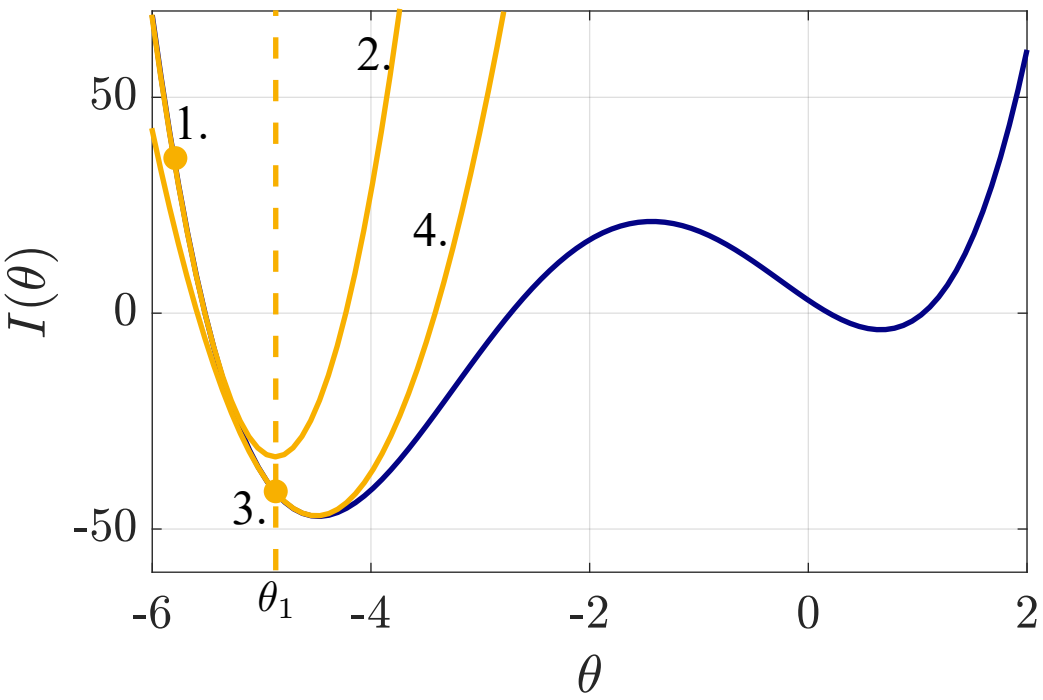
5.3 Newton-Verfahren

1. Wähle initialen Parameter θ_0
2. Bilde quadratische Approximation um $I(\theta_0)$
3. Finde das Minimum θ_1 der quadratischen Approximation.



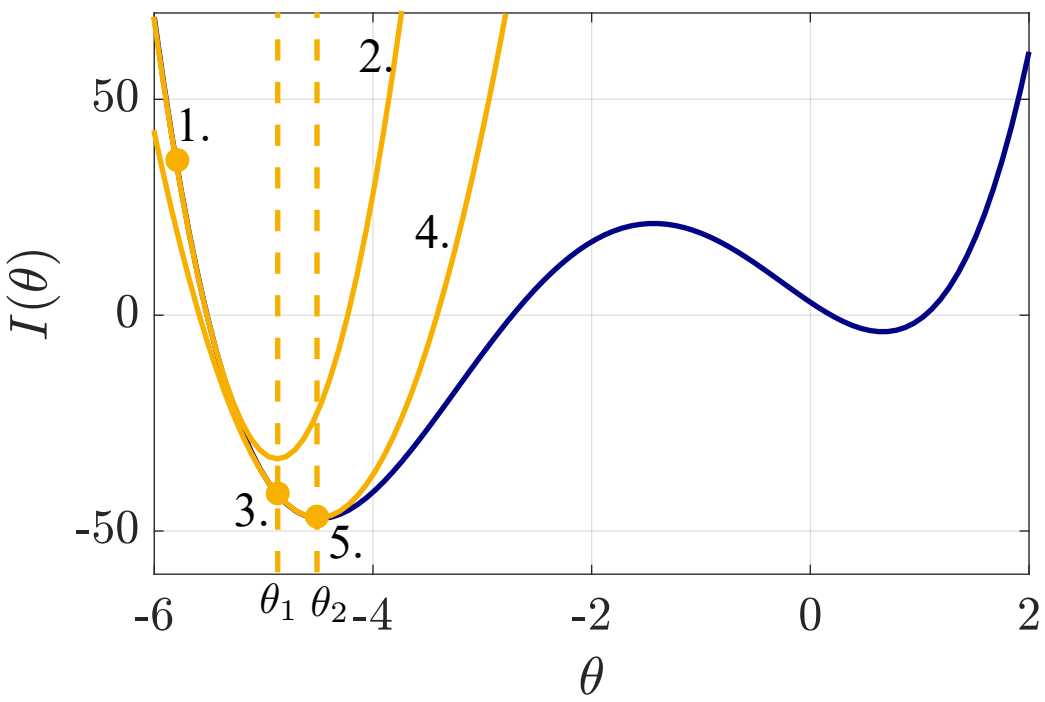
5.3 Newton-Verfahren

1. Wähle initialen Parameter θ_0
2. Bilde quadratische Approximation um $I(\theta_0)$
3. Finde das Minimum θ_1 der quadratischen Approximation.
4. Bilde quadratische Approximation um $I(\theta_1)$



5.3 Newton-Verfahren

1. Wähle initialen Parameter θ_0
2. Bilde quadratische Approximation um $I(\theta_0)$
3. Finde das Minimum θ_1 der quadratischen Approximation.
4. Bilde quadratische Approximation um $I(\theta_1)$
5. Finde das Minimum θ_2 der quadratischen Approximation.
6. ...
7. Konvergenz



5.3 Newton-Verfahren

Das Newton Verfahren basiert auf einer quadratischen Approximation der nichtlinearen Verlustfunktion um den alten Parametervektor $\underline{\theta}_i$. Das heißt im ersten Schritt um den initialen Parametervektor $\underline{\theta}_0$.

$$I(\underline{\theta}) \approx I\left(\underline{\theta}_0\right) + \left(\underline{\theta} - \underline{\theta}_0\right)^T \underline{g}_0 + \frac{1}{2}\left(\underline{\theta} - \underline{\theta}_0\right)^T \underline{H}_0 \left(\underline{\theta} - \underline{\theta}_0\right)$$

Der Gradient $\underline{g}_0$ und die Hesse-Matrix $\underline{H}_0$ werden beide am Punkt $\underline{\theta}_0$ berechnet. Nun kann die Ableitung der Approximation 0 gesetzt werden, um ein Optimalitätskriterium der Approximation zu bekommen

$$\underline{g}_0 + \underline{H}_0(\underline{\theta} - \underline{\theta}_0) \stackrel{!}{=} 0$$

Umformen liefert den optimalen Parametervektor der quadratischen Approximation

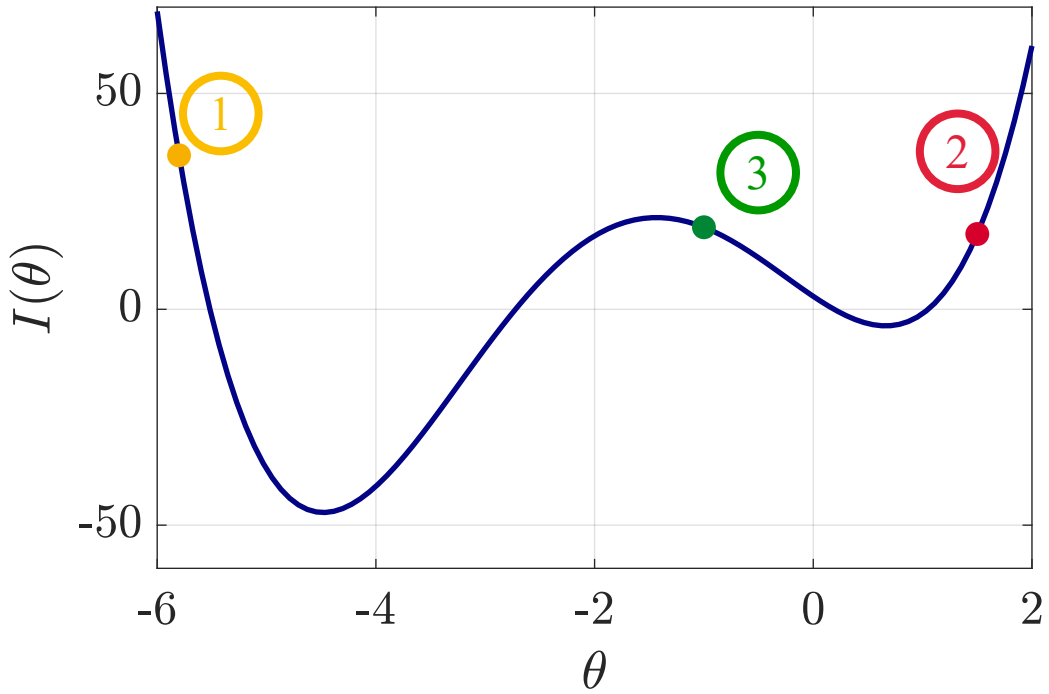
$$\underline{\theta}^* = \underline{\theta}_0 - \underline{H}_0^{-1}\underline{g}_0$$

Der optimale Parametervektor entspricht dem globalen Optimum der Verlustfunktion, wenn diese quadratisch ist (siehe Kapitel 3). Für allgemeine nichtlineare Verlustfunktionen ist das Newton Verfahren ein iteratives Verfahren mit der Updategleichung

$$\underline{\theta}_{i+1} = \underline{\theta}_i - \underline{H}_i^{-1}\underline{g}_i$$

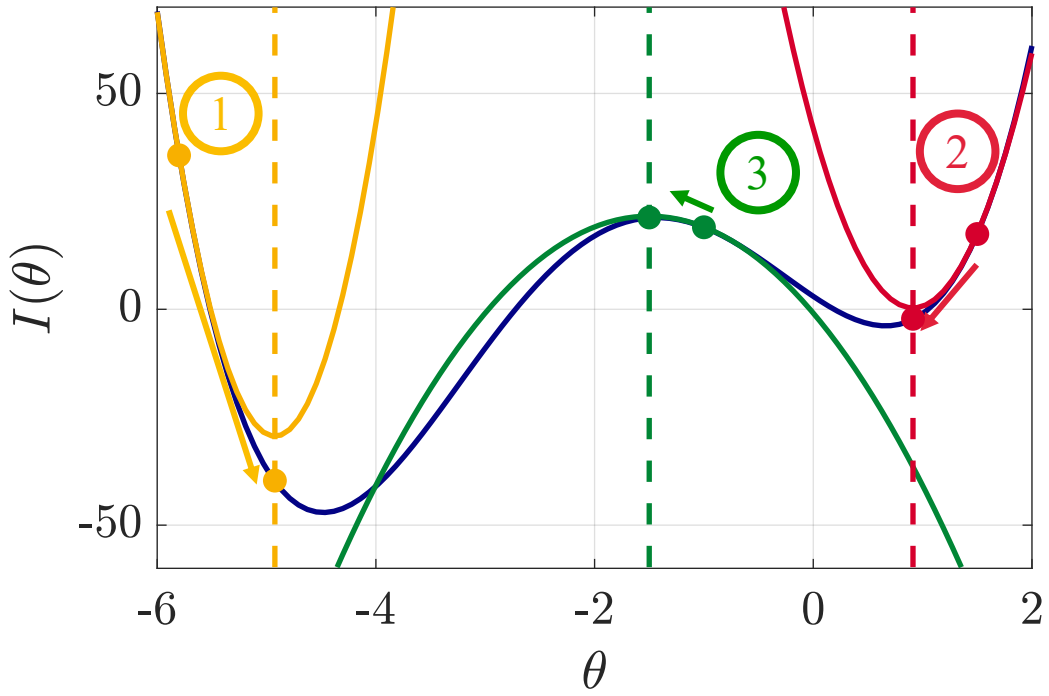
5.3 Newton-Verfahren

Was passiert, wenn man für die Suche des Optimums mit dem Newtonverfahren bei folgender Verlustfunktion von unterschiedlichen Startpunkten aus beginnt?



5.3 Newton-Verfahren

Was passiert, wenn man für die Suche des Optimums mit dem Newtonverfahren bei folgender Verlustfunktion von unterschiedlichen Startpunkten aus beginnt?



1) konvergiert zum globalen Optimum, 2) zum rechten lokalen Optimum

3) konvergiert zum globalem Maximum ($\theta^* \approx -1.43$)! Warum ist das so?

5.3 Newton-Verfahren

Updategleichung Newton Verfahren

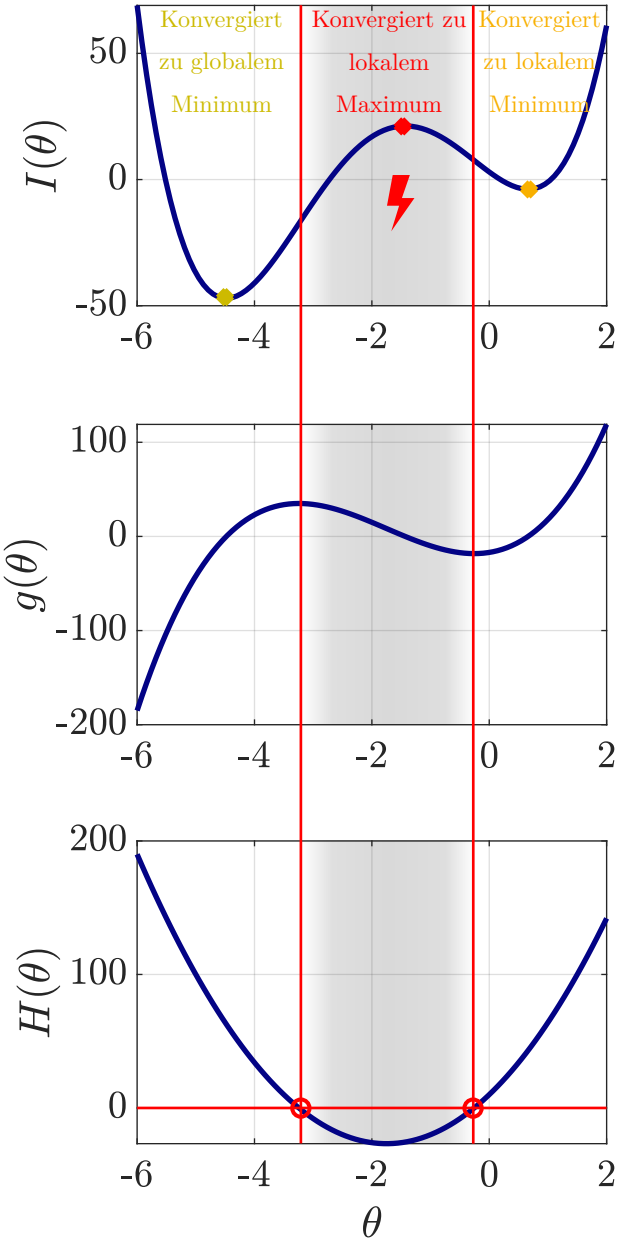
$$\underline{\theta}_{i+1} = \underline{\theta}_i - \underline{H}_i^{-1}\underline{g}_i$$

Die Hesse-Matrix (im eindimensionalen Beispiel ein Skalar), ist negativ im Bereich [-3,25 -0.25]. Somit läuft der Algorithmus nicht entgegengesetzt der Gradientenrichtung (zu kleineren Verlustfunktionswerten) sondern *in* Gradientenrichtung und somit zu höheren Verlustfunktionswerten.

Das Optimalitätkriterium $\underline{g}_0 + \underline{H}_0(\underline{\theta} - \underline{\theta}_0) \stackrel{!}{=} 0$ gibt keine Aussage über Minimum oder Maximum, sondern besagt lediglich, dass es sich um ein Optimum handelt.

Ein Optimum ist nur ein Minimum, wenn die Hesse-Matrix positiv-definit ist (im skalaren Fall, wenn die zweite Ableitung > 0 ist).

(Startet das Newton-Verfahren in der Nähe der Null-Durchgänge von der zweiten Ableitung, dann ist die Inverse der zweiten Ableitung sehr groß. Der neue Parameter liegt somit weit weg vom vorherigen und das Newton-Verfahren konvergiert möglicherweise zu einem völlig anderem Optimum. Die roten Linien sind somit weniger eine exakte Grenze ab der das Newton-Verfahren zum Maximum konvergiert als vielmehr ein Bereich, in dem das Verfahren zunehmend zum lokalen Maximum konvergiert.)



5.3 Newton-Verfahren

Die Updategleichung des Newton-Verfahrens

$$\underline{\theta}_{i+1} = \underline{\theta}_i - \underline{H}_i^{-1} \underline{g}_i$$

reduziert die Verlustfunktion nur, wenn $\underline{H}_i$ positiv definit ist. Dies ist nahe eines lokalen Minimums gegeben, aber nicht notwendigerweise zu Beginn der Optimierung. Zwei mögliche Modifikationen, falls $\underline{H}_i$ nicht positiv definit ist

1. Laufe in die steepest-descent Richtung $\underline{\theta}_{i+1} = \underline{\theta}_i - g_i$, falls $\underline{H}_i$ nicht positiv definit ist
2. Modifiziere die Hesse-Matrix, sodass sie positiv definit wird. Hierfür kann die Update-Richtung hin zur steepest-descent Richtung korrigiert werden

$$\underline{\theta}_{i+1} = \underline{\theta}_i - (\underline{H}_i + \nu \underline{I})^{-1} \underline{g}_i$$

Einheitsmatrix

5.3 Newton-Verfahren

Eigenschaften

- Benötigt zweite Ableitungen
- Kubischer Rechenaufwand wegen Matrix-Inversion
- Quadratischer Speicherbedarf, um Hesse-Matrix zu speichern
- Normalerweise schnellere Konvergenz als Verfahren erster Ordnung
- Benötigt nur einen Schritt zum Optimum, wenn es sich um eine quadratische Verlustfunktion ohne Nebenbedingungen handelt (Least Squares)
- Für kleine Anzahl Parameter gut geeignet ($n < 10$)
- Quadratische Konvergenz (Anzahl der signifikanten Stellen steigt quadratisch an)

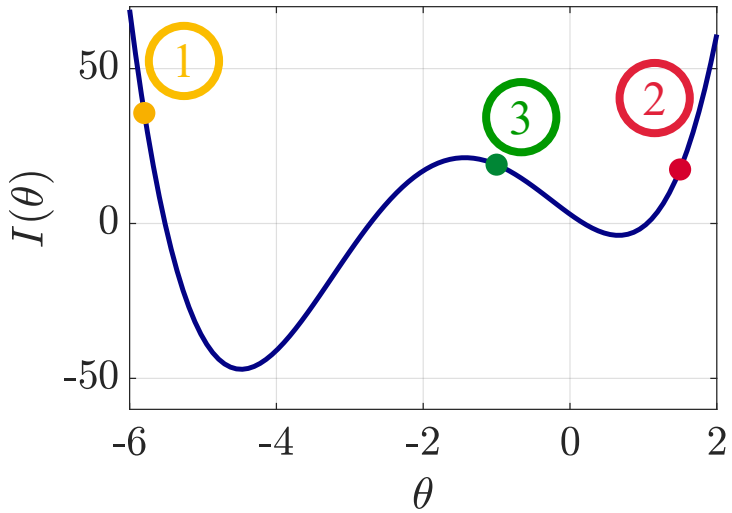
$$\frac{\|\theta_{i+1} - \theta_{\text{opt}}\|}{\|\theta_i - \theta_{\text{opt}}\|^2} < \beta_i, \beta_i \in (0,1)$$

5.3 Newton-Verfahren

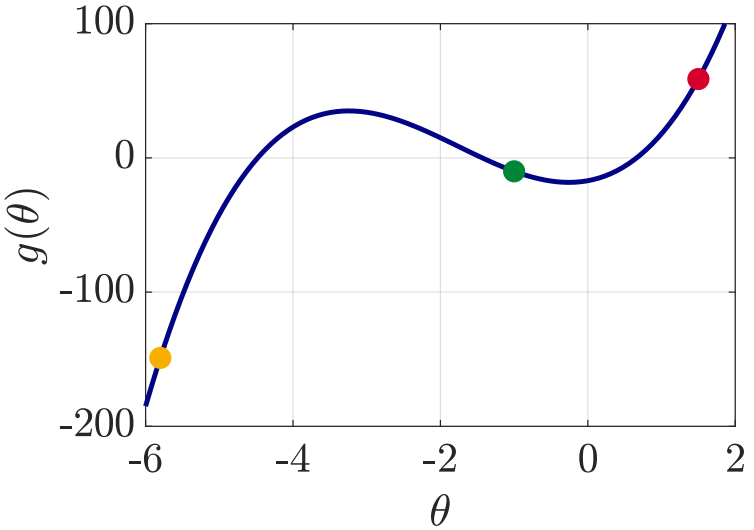
Parallele zu Nullstellensuche

$$I'(\cdot) = g(\cdot)$$
$$g'(\cdot) = H(\cdot)$$

Newton-Verfahren zur Optimierung



Newton-Verfahren zur Nullstellensuche



Ziel

Suche von Extrema

$$\underline{\theta}_{i+1} = \underline{\theta}_i - \underline{H}_i^{-1}\underline{g}_i$$

Suche von Nullstellen

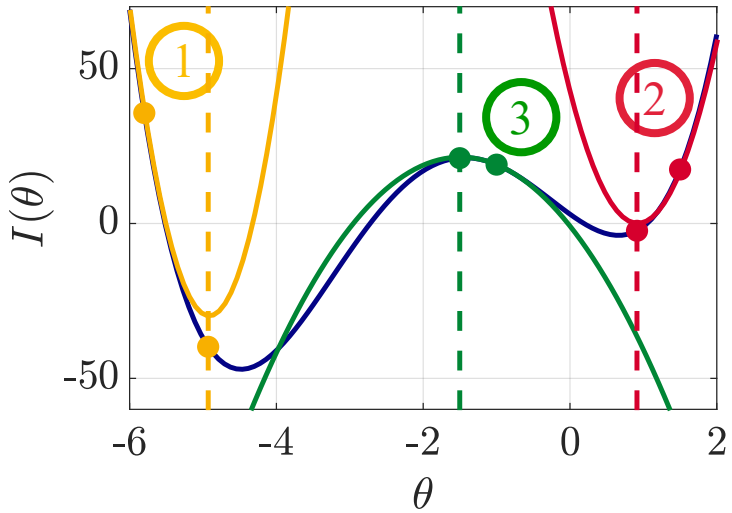
$$\theta_{i+1} = \theta_i - \frac{g(\theta_i)}{g'(\theta_i)}$$

5.3 Newton-Verfahren

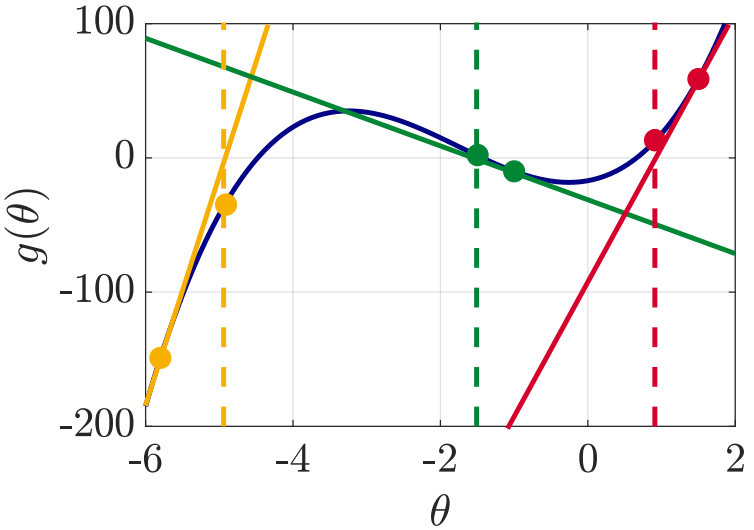
Parallele zu Nullstellensuche

$$I'(\cdot) = g(\cdot)$$
$$g'(\cdot) = H(\cdot)$$

Newton-Verfahren zur Optimierung



Newton-Verfahren zur Nullstellensuche



Ziel

Suche von Extrema

$$\underline{\theta}_{i+1} = \underline{\theta}_i - \underline{H}_i^{-1}\underline{g}_i$$

Suche von Nullstellen

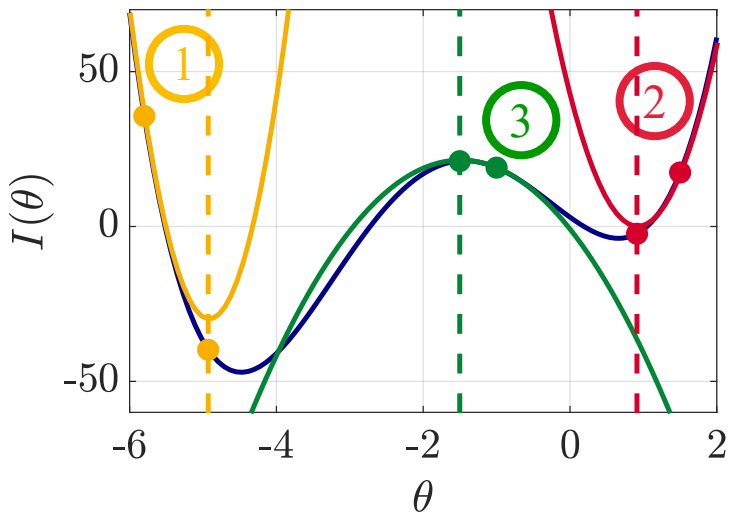
$$\theta_{i+1} = \theta_i - \frac{g(\theta_i)}{g'(\theta_i)}$$

5.3 Newton-Verfahren

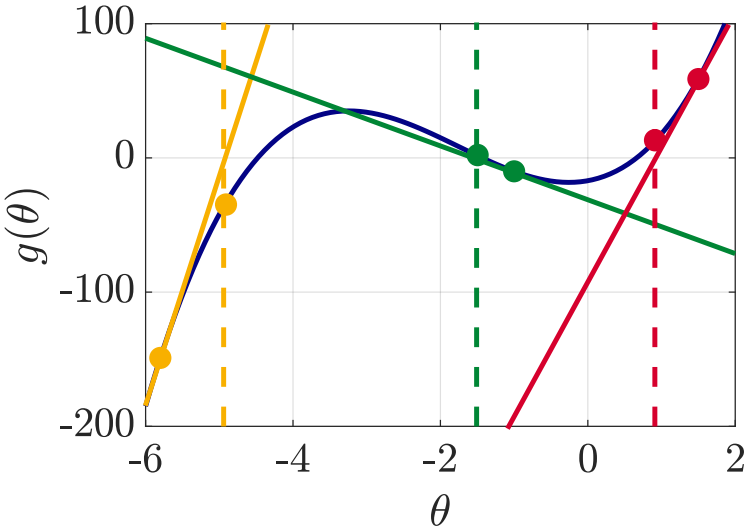
Parallele zu Nullstellensuche

$$I'(\cdot) = g(\cdot)$$
$$g'(\cdot) = H(\cdot)$$

Newton-Verfahren zur Optimierung



Newton-Verfahren zur Nullstellensuche



Ziel

Suche von Extrema

$$\underline{\theta}_{i+1} = \underline{\theta}_i - \underline{H}_i^{-1}\underline{g}_i$$

⇩ für den 1-D Fall

$$\theta_{i+1} = \theta_i - \frac{I'(\theta_i)}{I''(\theta_i)}$$

Suche von Nullstellen

$$\theta_{i+1} = \theta_i - \frac{g(\theta_i)}{g'(\theta_i)}$$

⇩ für den Fall $I'(\cdot) = g(\cdot)$

$$\theta_{i+1} = \theta_i - \frac{I'(\theta_i)}{I''(\theta_i)}$$

Die Anwendung des Newton-Verfahrens zur **Optimierung** und zur **Nullstellensuche** auf den Gradienten der Verlustfunktion sind identisch!

5.4 Quasi-Newton-Verfahren

Wenn die Hesse-Matrix analytisch nicht berechnet werden kann, muss sie numerisch berechnet werden. Dies benötigt $\mathcal{O}(n^2)$ Gradientenberechnungen. Außerdem muss eine $n \times n$ Hesse-Matrix invertiert werden. Das Newton-Verfahren kann deshalb schon für mittelgroße Probleme an seine Grenzen stoßen.

Die Idee von Quasi-Newton Methoden ist es, die Hesse-Matrix (oder ihre Inverse) zu approximieren.

$$\underline{\theta}_{i+1} = \underline{\theta}_i - \hat{\underline{H}}_i^{-1} \underline{g}_i \qquad \begin{array}{ll} \text{entweder mit} & \hat{\underline{H}}_{i+1} = \hat{\underline{H}}_i + \underline{Q}_i \\ \text{oder} & \hat{\underline{H}}_{i+1}^{-1} = \hat{\underline{H}}_i^{-1} + \tilde{\underline{Q}}_i \end{array}$$

Die Approximation der inversen Hesse-Matrix hat den Vorteil, dass keine Inversion mehr stattfinden muss. Diese Variante ist auch die am meisten verwendete. Gestartet wird der Algorithmus mit der initialen Hesse-Matrix $\underline{H}_0 = \underline{I}$ (steepest-descent Suchrichtung).

5.4 Quasi-Newton-Verfahren

Die bekanntesten Quasi-Newton Verfahren basieren auf der Broyden-Fletcher-Goldfrarb-Shanno (BFGS) Formel mit $\Delta\underline{\theta}_i = \underline{\theta}_{i+1} - \underline{\theta}_i$ und $\Delta\underline{g}_i = \underline{g}_{i+1} - \underline{g}_i$

$$\hat{\underline{H}}_{i+1}^{-1} = \left(\underline{I} - \frac{\Delta\underline{\theta}_i \Delta\underline{g}_i^T}{\Delta\underline{\theta}_i^T \Delta\underline{g}_i}\right) \hat{\underline{H}}_i^{-1} \left(\underline{I} - \frac{\Delta\underline{\theta}_i \Delta\underline{g}_i^T}{\Delta\underline{\theta}_i^T \Delta\underline{g}_i}\right)^T + \frac{\Delta\underline{\theta}_i \Delta\underline{\theta}_i^T}{\Delta\underline{\theta}_i^T \Delta\underline{g}_i} \quad .$$

Wichtigste Eigenschaften:

- Keine Ableitungen zweiter Ordnung benötigt
- Quadratische Rechenkomplexität wegen Matrix-Multiplikation
- Quadratischer Speicherbedarf
- Sehr schnelle Konvergenz
- Benötigt maximal n Iterationen für eine quadratische Verlustfunktion
- Gut geeignet für Probleme mit Anzahl Parametern in Größenordnung um 100
- Wenn $\Delta\underline{\theta}_i^T \Delta\underline{g}_i > 0$, dann ist die Hesse-Matrix nach dem Update weiterhin positiv definit

5.5 Konjugiertes Gradientenverfahren

Alle Quasi-Newton Methoden haben quadratisch ansteigenden Speicherbedarf mit der Anzahl Parameter n. Für große Probleme lohnt es sich oft nicht die Hesse-Matrix zu approximieren. Konjugierte Gradientenverfahren approximieren deshalb nicht die Hesse-Matrix (linearer skalierender Speicherbedarf und Rechenaufwand) sondern suchen pro Iteration ein Update der Such-Richtung $\underline{p}$ in

$$\underline{\theta}_{i+1} = \underline{\theta}_i - \eta_i \underline{p}_i \quad \text{mit} \quad \underline{p}_i = \underline{g}_i - \beta_i \underline{p}_{i-1} \,.$$

Der Parameter β_i unterscheidet unterschiedliche konjugiere Gradientenverfahren. Die bekannteste Wahl ist nach Fletcher und Reeves

$$\beta_i = \frac{\underline{g}_i^T \underline{g}_i}{\underline{g}_{i-1}^T \underline{g}_{i-1}} \,.$$

Der Faktor β_i beinhaltet das Wissen, das von vorherigen Iterationen mitgenommen wird. Er stellt den Kompromiss zwischen steepest descent (kein Wissen über vergangene Update-Richtungen) und von der Quasi-Newton Methode (Approximationen der Hesse-Matrix) ein.

5.5 Konjugiertes Gradientenverfahren

Eigenschaften

- Benötigt für schnelle Konvergenz eine genaue Liniensuche (wird in Kapitel 5.6 erklärt) von η
- Auf einer quadratischen Verlustfunktion konvergiert das Verfahren in maximal n Iterationen
- Konjugierte Gradientenverfahren werden auch genutzt um Least Squares Probleme zu lösen, wenn diese groß oder „sparse“ sind.
- Das Verfahren bekommt seinen Namen durch die Updategleichung

$$\underline{p}_i = \underline{g}_i - \beta_i \underline{p}_{i-1}$$

in der konjugierte Suchrichtungen berechnet werden. Zwei Suchrichtungen sind konjugiert, wenn gilt $\underline{p}_i^T \underline{H}\, \underline{p}_j = 0$. Dies impliziert, dass $\underline{p}_i$ und $\underline{p}_j$ unabhängige Suchrichtungen sind. „Konjugiert“ ($\underline{H}$ beliebig) ist eine Erweiterung von „orthogonal“ ($\underline{H} = \underline{I}$ Einheitsmatrix).

- Es werden keine zweiten Ableitungen benötigt
- Linearer Rechenaufwand
- Schnelle Konvergenz
- Am besten für große Probleme geeignet (Ordnung 1000 oder mehr Parameter)

5.6 Liniensuche

Die Liniensuche ist ein wichtiges Verfahren überall dort, wo Schrittweiten η nicht als fester Wert gewählt sondern eingestellt werden müssen. Gradientenbasierte Optimierungsverfahren lassen sich alle als

$$\underline{\theta}_{i+1} = \underline{\theta}_i - \eta_i \underbrace{\underline{R}_i \underline{g}_i}_{\underline{p}_i}$$

schreiben. Hier ist $\underline{p}_i$ die Suchrichtung. Für alle positiv definiten Matrizen $\underline{R}_i$ sorgt diese Updategleichung für eine Minimierung der Verlustfunktion $I(\underline{\theta}_{i+1}) < I(\underline{\theta}_i)$.

Anstelle einfach eine Schrittweite (Lernrate) η_i „heuristisch“ zu bestimmen, welche die Verlustfunktion beliebig viel oder wenig minimiert, probiert die Liniensuche, die optimale Schrittweite mit maximaler Reduktion der Verlustfunktion in die gegebene Richtung $\underline{p}_i$ zu finden.

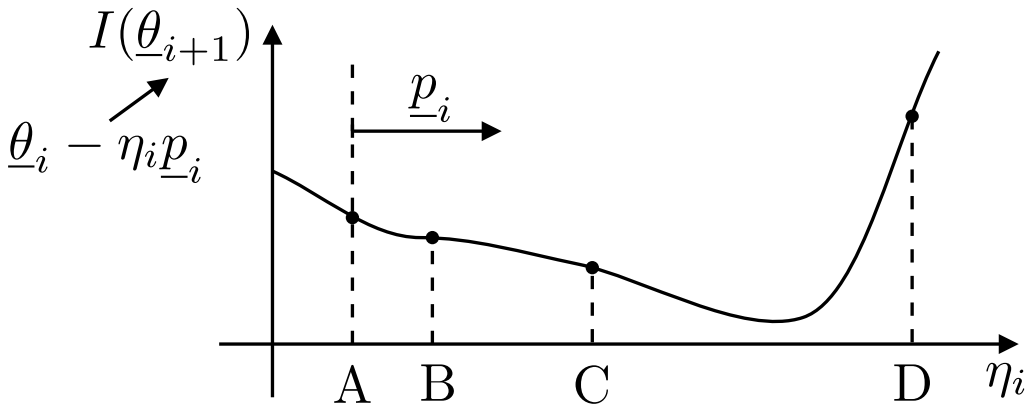
Das Verfahren besteht aus zwei Schritten:

1. Das Intervall, das entlang $\underline{p}_i$ das Minimum enthält, muss gefunden werden.
2. Das Minimum muss in diesem Intervall gefunden werden.

5.6 Liniensuche

Intervallsuche

Um das Intervall zu finden, in dem das lokales Minimum bezüglich der Schrittweite η_i liegt, wird die Verlustfunktion an unterschiedlichen Stellen ausgewertet (im Beispiel A (Startpunkt), B, C und D) bis die Verlustfunktion wieder anfängt zu steigen. Um schnelle Konvergenz zu garantieren, wird die Schrittweite für jede Stelle verdoppelt.

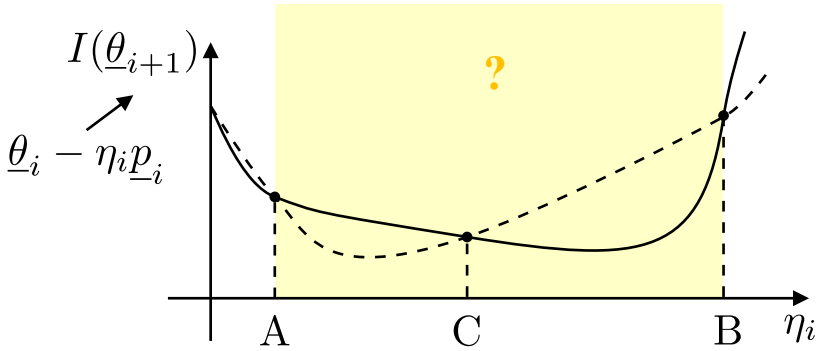


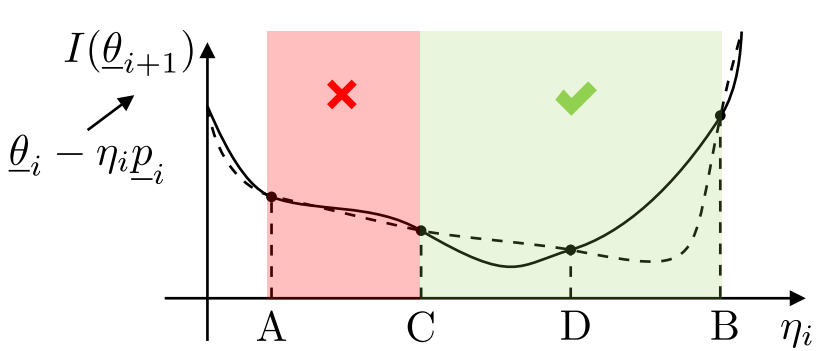
5.6 Liniensuche

Intervallreduktion

Ist ein Intervall [A, B] (B = D aus vorheriger Folie!) gefunden, in dem ein Minimum liegt, muss die Intervallgröße reduziert werden, um näher ans Optimum zu gelangen. Teilt man das Intervall nur durch **einen** Punkt C (linkes Bild), kann das Intervall nicht reduziert werden. Das lokale Minimum kann links oder rechts von C liegen.

Unterteilt man [A, B] hingegen durch **zwei** Punkte C und D also in in drei Teile (rechtes Bild) kann das Intervall auf [C, B] reduziert werden**. Dies ist möglich, weil davon ausgegangen wird, dass nur ein Minimum in [A, B] existiert.** Intelligente Platzierung von den Punkten C und D (bspw. Fibonacci oder Golden Section Search) machen es möglich, in der nächsten Iteration die Auswertung am Punkt D weiter zu verwenden und in jeder Iteration nur eine weitere, neue Auswertung der Verlustfunktion zu machen.



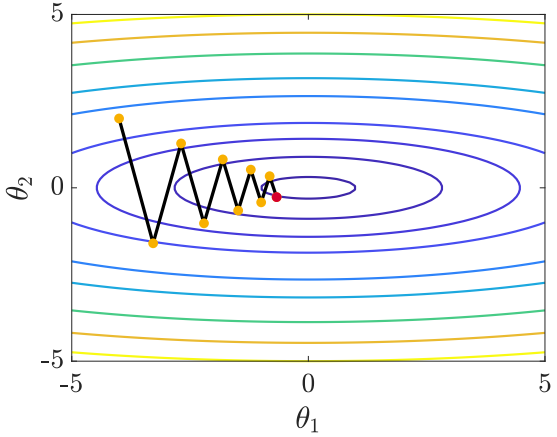


5.6 Liniensuche im Gradientenverfahren

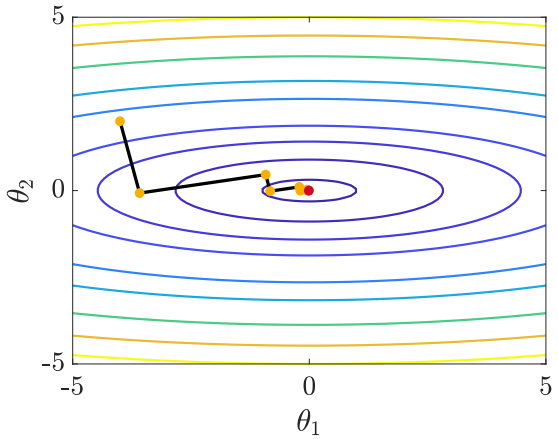
Das Problem beim Gradientenverfahren „Zig-Zagging“ wurde bereits beschrieben. Anstelle einer fixen Schrittweite soll nun in jedem Schritt die ideale Schrittweite berechnet werden.

$$I(\theta) = \begin{bmatrix}\theta_1 & \theta_2\end{bmatrix} \underbrace{\begin{bmatrix}0.1 & 0\\ 0 & 1\end{bmatrix}}_{\underline{H}} \begin{bmatrix}\theta_1\\ \theta_2\end{bmatrix} \quad \underline{\theta}_0 = \begin{bmatrix}-4\\ 2\end{bmatrix}$$

Ohne Liniensuche $\eta = 0.9$

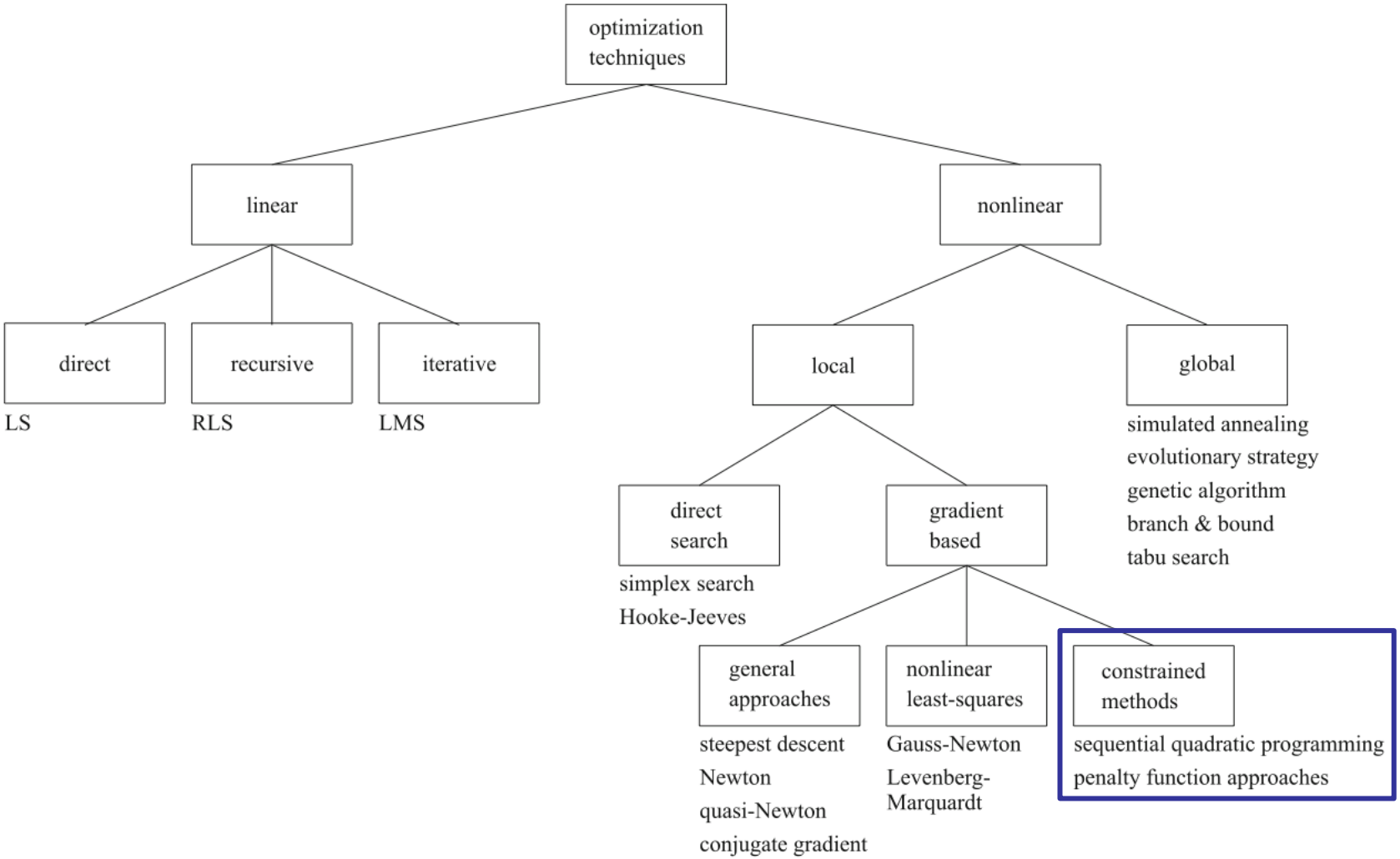


Mit Liniensuche



Mit Liniensuche benötigt das Verfahren weniger Iterationen. Jedoch muss abgewogen werden, ob der zusätzliche Rechenaufwand der Liniensuche lohnt. Ist die Berechnung der Suchrichtung aufwendig (bspw. bei Newton-Verfahren), ist die Liniensuche im Verhältnis weniger aufwendig und somit sinnvoll. Ist hingegen die Berechnung der Suchrichtung wenig rechenaufwändig (bspw. Steepest Descent) ist eine Liniensuche im Verhältnis deutlich rechenaufwendiger und somit nicht sinnvoll.

5.7 Nichtlineare Probleme mit Nebenbedingungen



5.7 Nichtlineare Probleme mit Nebenbedingungen

Bis jetzt können die Parameter im Parameterraum beliebige Größen annehmen. Häufig liegen allerdings **Nebenbedingungen (NB)** vor, die während der Optimierung eingehalten werden müssen. Die zu optimierenden Parameter sind in solchen Fällen meist physikalisch interpretierbar, da sonst keine Nebenbedingungen für sie sinnvoll erstellt werden könnten.

Im allgemeinen kann ein nichtlineares Optimierungsproblem mit Ungleichungs- und Gleichungsnebenbedingungen wie folgt formuliert werden:

$$\min_{\underline{\theta}} I(\underline{\theta}) \quad \text{subject to} \begin{cases} g_i(\underline{\theta}) \leq 0, & i = 1, \ldots, m \\ h_j(\underline{\theta}) = 0, & j = 1, \ldots, l \end{cases}$$

Wir führen nun einen Strafterm ein, der bei Verletzung der Nebenbedingungen die Verlustfunktion vergrößert, um den Optimierer davon wegzutreiben:

$$\min_{\underline{\theta}} I(\underline{\theta}) + P(g_i(\underline{\theta}), h_j(\underline{\theta})) \text{ für alle } i, j$$

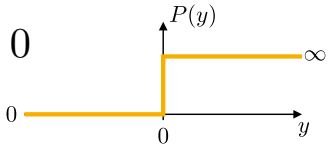
5.7 Nichtlineare Probleme mit Nebenbedingungen

$$\min_{\underline{\theta}} I_p(\underline{\theta}, p) = \min_{\underline{\theta}} I(\underline{\theta}) + P(g_i(\underline{\theta}), h_j(\underline{\theta}))$$

Was ist der beste Strafterm $P(y(\underline{\theta})) = P(g_i(\underline{\theta}), h_j(\underline{\theta}))$ für alle i, j?

Am einfachsten wäre ein Strafterm, der im zulässigen Bereich 0 hinzuaddiert und ∞ überall sonst. Die Verlustfunktion würde nur im unzulässigen Bereich „verzerrt“ werden. Dieser ideale Strafterm sähe dann so aus:

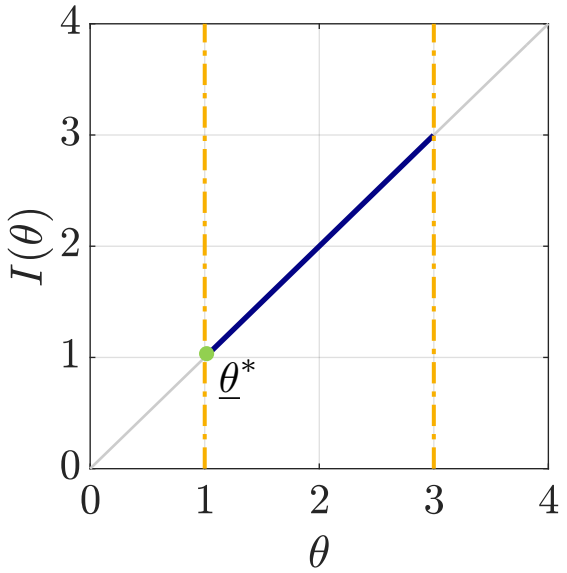
$$P(y(\underline{\theta})) = \begin{cases} 0 \text{ wenn } y \leq 0 \\ \infty \text{ sonst} \end{cases}$$



Beispiel

$$I(\theta) = \theta$$
$$\text{s.t. } g(\theta) = (\theta - 2)^2 - 1 \leq 0$$

$$\min_{\theta} I(\theta) + P(\underbrace{(\theta - 2)^2 - 1}_{y(\theta)})$$



Problem: $P(y)$ ist nicht differenzierbar und die Optimierung unmöglich, wenn der Initialwert außerhalb des gültigen Bereiches liegt. Wie kann man $P(y)$ sinnvoll annähern?

5.7 Nichtlineare Probleme mit Nebenbedingungen

Äußere Barrierefunktionen

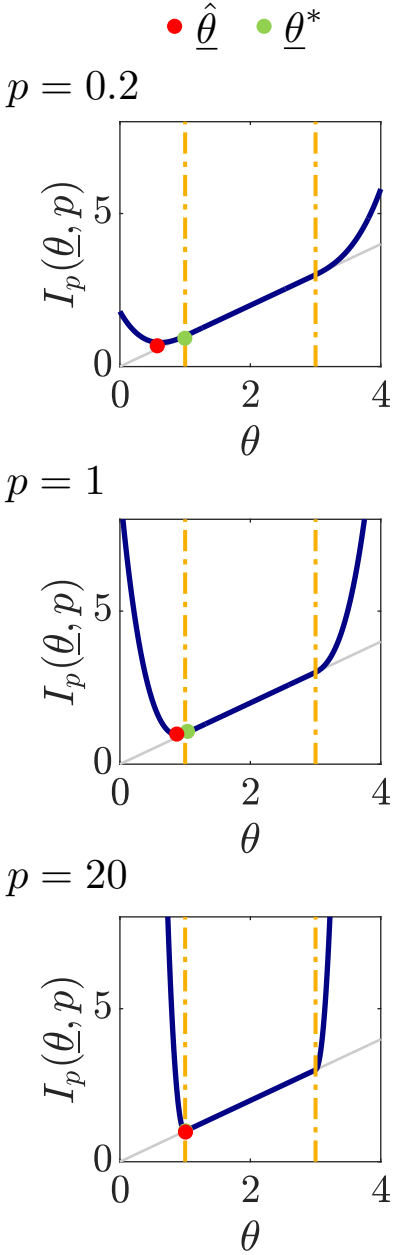
$$\min_{\underline{\theta}} I_p(\underline{\theta}, p) = \min_{\underline{\theta}} I(\underline{\theta}) + P(g_i(\underline{\theta}), h_j(\underline{\theta}))$$

beispielsweise

$$P_{\text{exterior}}\left(y(\underline{\theta})\right) = p \cdot \left(\sum_{i=1}^{m} \left(\max\left(0, g_i(\underline{\theta})\right)\right)^2 + \sum_{j=1}^{l} \left(h_j(\underline{\theta})\right)^2 \right)$$

Beispielsweise durch Barrierefunktionen. Wenn alle NB erfüllt sind ($g_i(\underline{\theta}) \leq 0$ und $h_j(\underline{\theta}) = 0$, für alle i, j), dann soll auch $P(y(\underline{\theta})) = 0$ sein. Durch die quadratischen Terme ist ein glatter Übergang von zulässigem und unzulässigem Bereich sichergestellt. Da nun alle Nebenbedingungen in die neue Verlustfunktion $I_p(\underline{\theta}, p)$ eingebaut wurden, kann ein beliebiger nichtlinearer Optimierer ohne Nebenbedingungen verwendet werden.

Für kleine p kann das Optimum außerhalb des zulässigen Bereiches liegen, dafür ist die eigentliche Verlustfunktion $I(\underline{\theta})$ wenig verändert. Für große p kann die Initialisierung schwierig sein, dafür muss das Optimum näher am zulässigen Bereich liegen. Strategie: Mit kleinem p die Optimierung beginnen und dann immer weiter erhöhen.



5.7 Nichtlineare Probleme mit Nebenbedingungen

Innere Barrierefunktionen

$$\min_{\underline{\theta}} I_p(\underline{\theta}, p) = \min_{\underline{\theta}} I(\underline{\theta}) + P(g_i(\underline{\theta}), h_j(\underline{\theta}))$$

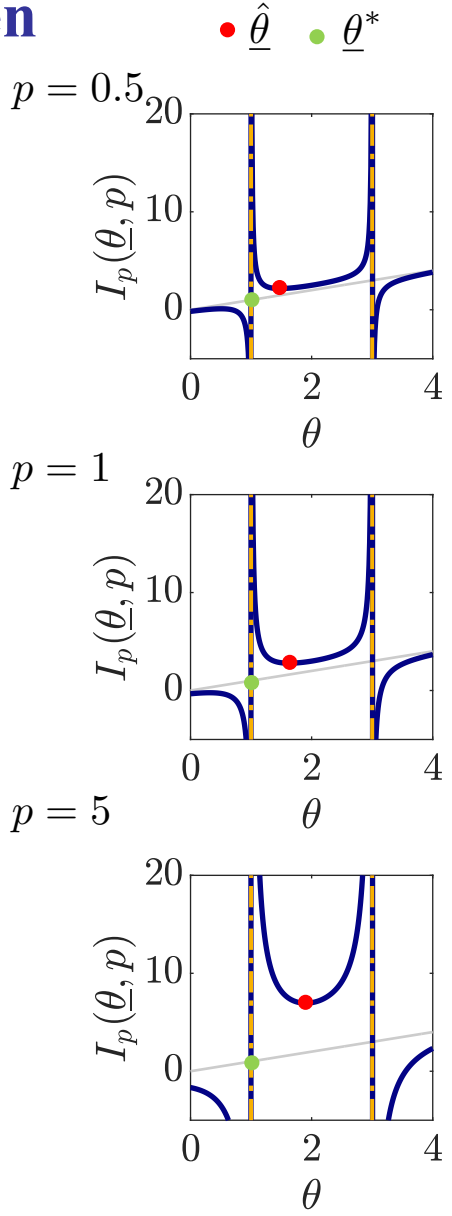
beispielsweise

$$P_{\text{interior}}\left(y(\underline{\theta})\right) = p \cdot \left(\sum_{i=1}^{m} \frac{-1}{g_i(\underline{\theta})} + \sum_{j=1}^{l} \left(h_j(\underline{\theta})\right)^2 \right)$$

Die inneren Barrierefunktionen verfolgen die gegensätzliche Philosophie zu den äußeren Barrierefunktionen. Die Barrierefunktion strebt nach unendlich, wenn man sich den Grenzen der NB vom zulässigen Bereich aus nähert.

Für kleine p liegt das geschätzte Optimum nahe dem echten Optimum. Für große p wird die Verlustfunktion immer weiter verzerrt.

Für Probleme mit einfachen Nebenbedingungen können Barrierefunktionen effektiv und einfach eingesetzt werden. Zum Beispiel sind Stellgrößenbeschränkungen oft durch solche Barrierefunktionen gut umsetzbar.

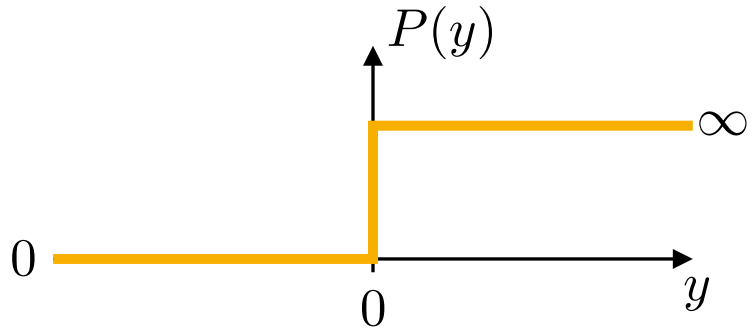


5.7 Nichtlineare Probleme mit Nebenbedingungen

Lagrange-Funktion

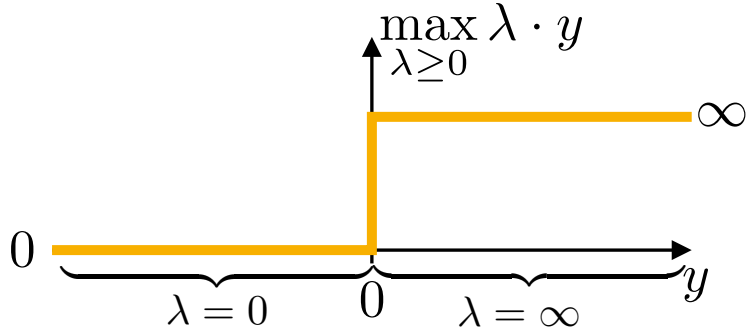
Äußere und innere Barrierefunktionen verzerren die ursprüngliche Verlustfunktion.
Die 0-Unendlich-Barrierefunktion führt zu einer nicht differenzierbaren Verlustfunktion.
Wir versuchen nun die ideale 0-Unendlich Verlustfunktion anders aufzufassen.

$$P(y(\underline{\theta})) = \begin{cases} 0 \text{ wenn } y \leq 0 \\ \infty \text{ sonst} \end{cases}$$



Diese Funktion kann man auch als Maximierungs-Problem schreiben:

$$P(y(\underline{\theta})) = \max_{\lambda \geq 0} \lambda \cdot y(\underline{\theta})$$



5.7 Nichtlineare Probleme mit Nebenbedingungen

Lagrange-Funktion

Nun setzten wir in unser Beispielproblem die umgeschriebene Barrierefunktion ein.

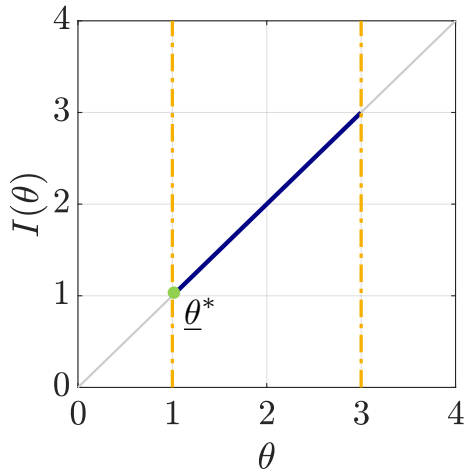
Problem:

$$I(\theta) = \theta$$

$$\text{s.t. } g(\theta) = (\theta - 2)^2 - 1 \leq 0$$

$$\min_{\theta} I(\theta) + P(\underbrace{(\theta - 2)^2 - 1}_{y(\theta)})$$

$$\min_{\theta} \left(\theta + \max_{\lambda \geq 0} \lambda \left((\theta - 2)^2 - 1 \right) \right)$$



Der max-Operator kann nun nach vorne gezogen werden (kein λ in $I(\underline{\theta})$)

$$\min_{\theta} \left(\max_{\lambda \geq 0} \underbrace{(\theta + \lambda \left((\theta - 2)^2 - 1 \right))}_{\mathcal{L}(\underline{\theta}, \lambda)} \right)$$

Dieses Problem wird primales Problem genannt. Dies ist oft nicht einfach zu lösen. Das Funktional $\mathcal{L}(\underline{\theta}, \lambda)$ heißt Lagrange-Funktion.

5.7 Nichtlineare Probleme mit Nebenbedingungen

Lagrange-Funktion

Oft ist es viel einfacher das duale Problem (vertauschen von min- und max-Operator) zu lösen:

$$\max_{\lambda \geq 0} \left(\min_{\theta} \underbrace{(\theta + \lambda \left((\theta - 2)^2 - 1 \right))}_{\mathcal{L}(\underline{\theta}, \lambda)} \right)$$

Das Lösen dieses Problems ist oft deutlich einfacher und führt (wenn gewisse Annahmen erfüllt sind, was bei diesem Beispiel der Fall ist) zum gleichen Ergebnis wie das Lösen des primalen Problems.

Im Allgemeinen kann jedes nichtlineare Problem mit Nebenbedingungen als Lagrange-Funktion geschrieben werden:

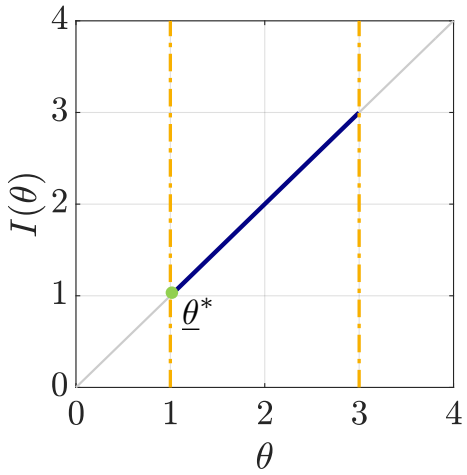
$$\mathcal{L}(\underline{\theta}, \underline{\lambda}, \underline{\mu}) = I(\underline{\theta}) + \sum_{\substack{i=1 \\ \lambda_i \geq 0}}^{m} \lambda_i g_i(\underline{\theta}) + \sum_{j=1}^{l} \mu_j \, h_j(\underline{\theta})$$

Für Gleichungsbedingungen kann μ_j beliebige positive wie negative Werte annehmen, für Ungleichungsbedingungen ist nur $\lambda_i \geq 0$ zulässig.

5.7 Nichtlineare Probleme mit Nebenbedingungen

Lösen des Beispielproblems via des dualen Problems (im allgemeinen Fall nicht immer so einfach möglich)

$$\max_{\lambda \geq 0} \left(\min_{\theta} \underbrace{(\theta + \lambda\,((\theta-2)^2 - 1))}_{\mathcal{L}(\underline{\theta},\lambda)} \right)$$



Lösen des Minimierungsproblems:

$$\frac{\partial \mathcal{L}(\underline{\theta},\lambda)}{\partial \underline{\theta}} = 1 + 2\lambda(\theta - 2) \stackrel{!}{=} 0 \quad \rightarrow \quad \theta = -\frac{1}{2\lambda} + 2$$

Einsetzen in Lagrange-Funktion liefert Maximierungsproblem

$$\max_{\lambda \geq 0} -\frac{1}{2\lambda} + 2 + \lambda(-\frac{1}{2\lambda})^2 - \lambda = \max_{\lambda \geq 0} -\frac{1}{4\lambda} + 2 - \lambda$$

Lösen des Maximierungsproblems liefert

$$\frac{\mathrm{d}}{\mathrm{d}\lambda}\left(-\frac{1}{4\lambda} + 2 - \lambda\right) \stackrel{!}{=} 0 \rightarrow \lambda = \frac{1}{2}$$

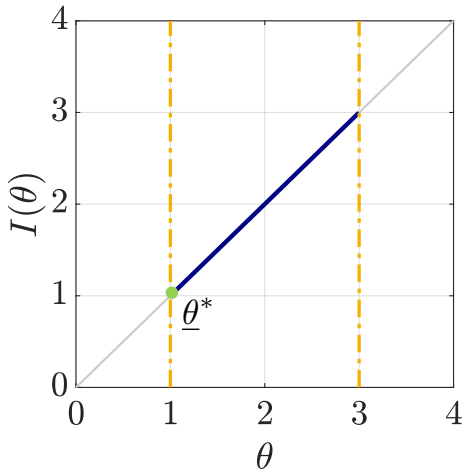
Einsetzen in obige Beziehung zwischen θ und λ liefert das Ergebnis

$$\theta^* = 1$$

5.7 Nichtlineare Probleme mit Nebenbedingungen

Lösen des Beispielproblems via des primalen Problems

$$\min_{\theta} \left(\max_{\lambda \geq 0} \underbrace{\left(\theta + \lambda \left((\theta-2)^2 - 1\right)\right)}_{\mathcal{L}(\underline{\theta},\lambda)} \right)$$



Lösen des Maximierungsproblems:

$$\frac{\mathrm{d}\mathcal{L}(\underline{\theta},\lambda)}{\mathrm{d}\lambda} = \left((\theta-2)^2 - 1\right) \stackrel{!}{=} 0 \;\rightarrow\; \theta \in [1,3]$$

Einsetzen von θ in Verlustfunktion liefert Ergebnis

$$\theta^* = 1$$

5.7 Nichtlineare Probleme mit Nebenbedingungen

Primales Problem

$$\min_{\theta} I(\theta) = \min_{\theta} \theta$$

$$\text{s.t. } g(\theta) = (\theta - 2)^2 - 1 \leq 0$$

Duales Problem

$$\max_{\lambda} d(\lambda)$$

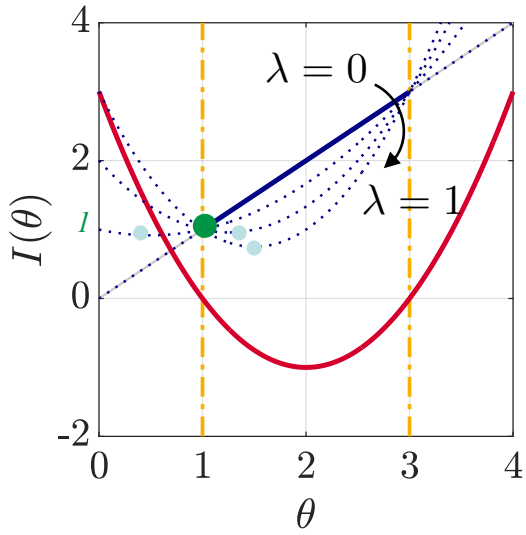
mit dualer Funktion

$$d(\lambda) = \min_{\theta} \mathcal{L}(\theta, \lambda)$$

und Lagrange-Funktion

$$\mathcal{L}(\theta, \lambda) = \theta + \lambda((\theta - 2)^2 - 1)$$

Lagrange-Multiplikator λ stellt ein, in wie weit die Lagrange-Funktion $\mathcal{L}(\theta, \lambda)$ der Verlustfunktion $I(\theta)$ oder der NB $g(\theta)$ entspricht.



Die duale Funktion ist immer konkav und ist kleiner oder gleich dem Optimum des primalen Problems (siehe nächste Folie)

$$d(\lambda) = \min_{\theta} \mathcal{L}(\theta, \lambda) \leq I(\theta^*) = 1$$

Hier starke Dualität (*strong duality*), da

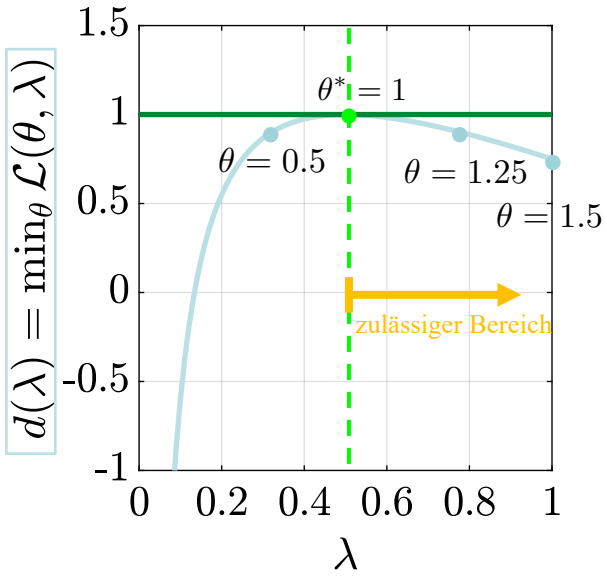
$$d(\lambda^*) = I(\theta^*)$$

Duality gap

$$I(\theta^*) - d(\lambda^*)$$

Schwache Dualität (*weak duality*) wäre

$$d(\lambda^*) \leq I(\theta^*)$$



5.7 Nichtlineare Probleme mit Nebenbedingungen

Dualität und Duality Gap

Lagrange-Funktion

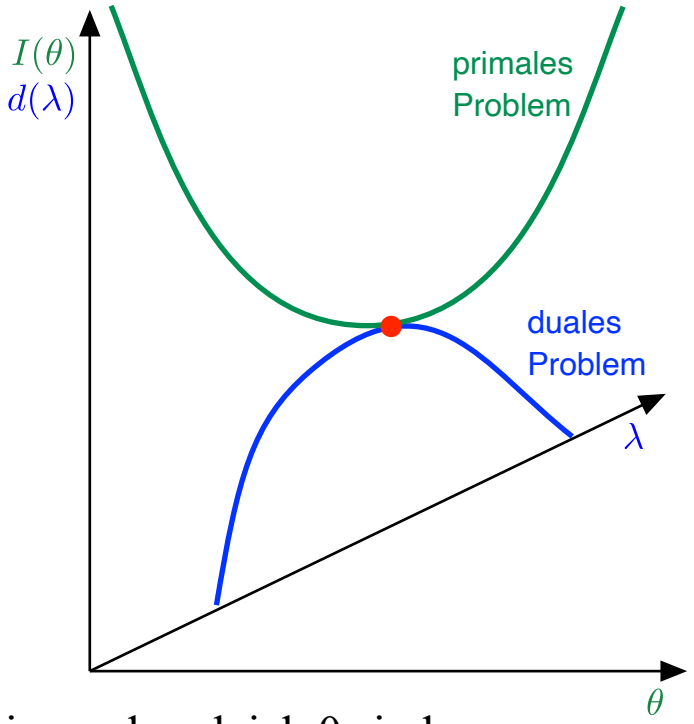
$$\mathcal{L}(\underline{\theta}, \underline{\lambda}, \underline{\mu}) = I(\underline{\theta}) + \sum_{\substack{i=1 \\ \lambda_i \geq 0}}^{m} \lambda_i g_i(\underline{\theta}) + \sum_{j=1}^{l} \mu_j \, h_j(\underline{\theta})$$

Duale Lagrange-Funktion oder nur duale Funktion

$$d(\underline{\lambda}, \underline{\mu}) = \min_{\underline{\theta}} \mathcal{L}(\underline{\theta}, \underline{\lambda}, \underline{\mu})$$

Die duale Funktion ist im zulässigen Bereich (NB erfüllt) eine untere Schwelle für das Optimierungsproblem

$$d(\underline{\lambda}, \underline{\mu}) \leq I(\underline{\theta}^*)$$



Dies gilt im zulässigen Bereich (NB erfüllt), da alle NB kleiner oder gleich 0 sind

$$\sum_{\substack{i=1 \\ \lambda_i \geq 0}}^{m} \lambda_i g_i(\underline{\theta}) + \sum_{j=1}^{l} \mu_j \, h_j(\underline{\theta}) \leq 0$$

Die Duality-Gap ist 0 ($d(\underline{\lambda}^*, \underline{\mu}^*) = I(\underline{\theta}^*)$) z.B. für LPs und QPs und die meisten anderen konvexen Optimierungsprobleme*, siehe Bild.

* Wenn Slater‘s Bedingung erfüllt ist

5.7 Nichtlineare Probleme mit Nebenbedingungen

Warum ist die duale Funktion immer konkav, unabhängig davon wie $I(\theta)$ aussieht?

Lagrange-Funktion (ein Parameter, eine NB)

$$\mathcal{L}(\theta, \lambda) = I(\theta) + \lambda g(\theta) = \theta + \lambda\left((\theta - 2)^2 - 1\right)$$

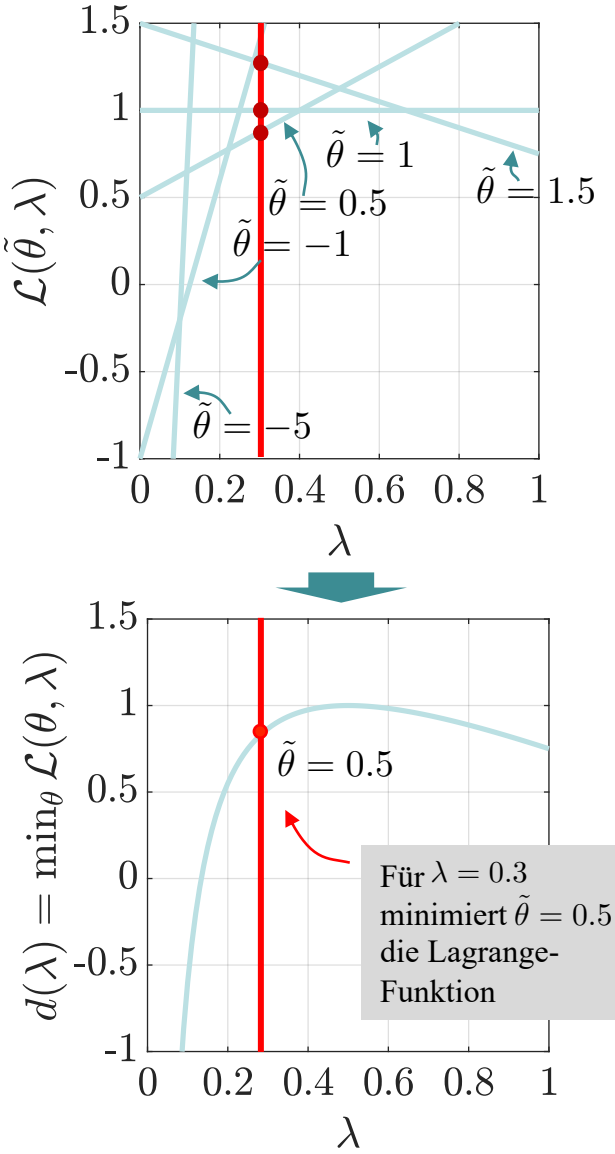
Es sei $\tilde{\theta}$ ein beliebiger fester Wert für θ, dann ist die Lagrange-Funktion ausgewertet an $\tilde{\theta}$ immer eine affine Funktion in λ (siehe Bild oben rechts für unterschiedliche $\tilde{\theta}$)

$$\mathcal{L}(\tilde{\theta}, \lambda) = I(\tilde{\theta}) + \lambda g(\tilde{\theta}) = m\lambda + b$$

Sucht man nun das Minimum der Lagrange-Funktion bezüglich „aller möglichen $\tilde{\theta}$ für ein bestimmtes λ“, erhält man die duale Funktion

$$d(\lambda) = \min_{\theta} \mathcal{L}(\theta, \lambda).$$

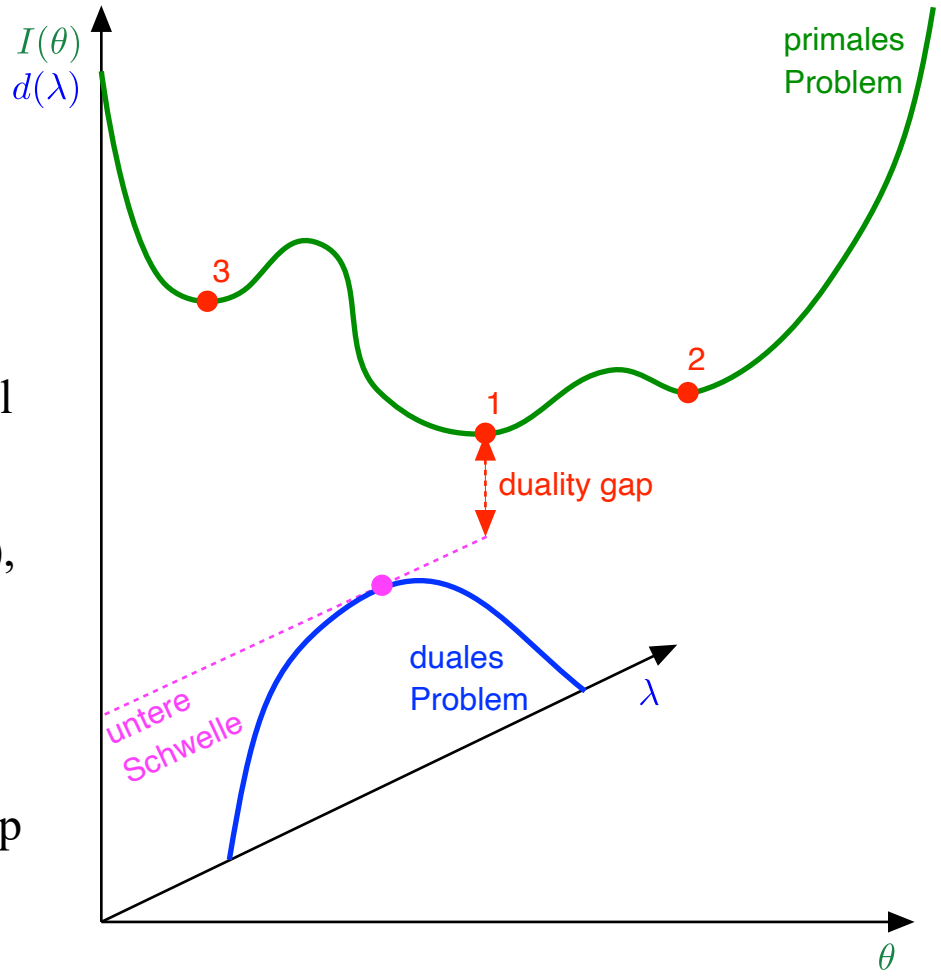
Da $d(\lambda)$ unterhalb aller affinen Funktionen $\mathcal{L}(\tilde{\theta}, \lambda)$ liegt, ist $d(\lambda)$ konkav.



5.7 Nichtlineare Probleme mit Nebenbedingungen

Nützlichkeit einer unteren Schwelle (*lower bound*), selbst wenn Duality Gap > 0

- Die untere Schwelle gibt Orientierung, wie gut ein gefundenes Optimum sein könnte.
- Findet man Optimum 1 (globales) oder 2 (gutes lokales) und der Abstand zur unteren Schwelle in I ist klein, weiß man, dass nicht mehr viel Verbesserungspotential existiert.
- Findet man Optimum 3 (schlechtes lokales), dann ist der Abstand zur unteren Schwelle groß. Dann könnte sich eine weitere Suche nach einem besseren Optimum lohnen.
- Da man nicht weiß, wie groß der duality gap ist, sind diese Überlegungen oft nur grobe Anhaltspunkte – aber besser als nichts!



5.7 Nichtlineare Probleme mit Nebenbedingungen

Die berühmten Karush-Kuhn-Tucker-Gleichungen geben die folgende notwendigen (und oft auch hinreichenden) Bedingungen für Optimalität für nichtlineare Probleme mit Nebenbedingungen

$\underline{\theta}^*$ is feasible if

$$\text{Stationarity:} \quad \frac{\partial I\left(\underline{\theta}^*\right)}{\partial \underline{\theta}} + \sum_{i=1}^{m} \lambda_i^* \frac{\partial g_i\left(\underline{\theta}^*\right)}{\partial \underline{\theta}} + \sum_{j=1}^{l} \mu_j^* \frac{\partial h_j\left(\underline{\theta}^*\right)}{\partial \underline{\theta}} = \underline{0}$$

$$\text{Primal feasibility:} \quad g_i(\underline{\theta}^*) \leq 0, \quad i = 1, \ldots, m$$

$$h_j(\underline{\theta}^*) = 0, \quad j = 1, \ldots, l$$

$$\text{Dual feasibility:} \quad \lambda_i^* \geq 0, \quad i = 1, \ldots, m$$

$$\text{Complementary slackness:} \quad \lambda_i^* g_i(\underline{\theta}^*) = 0, \quad i = 1, \ldots, m$$

An einem Optimum sollen die Nebenbedingungen keinen Einfluss auf die Verlustfunktion haben.

$g_i(\underline{\theta}^*) < 0 \rightarrow \lambda_i^* = 0$

$g_i(\underline{\theta}^*) = 0 \rightarrow \lambda_i^* \geq 0$

Die meisten nichtlinearen Optimierer, die Nebenbedingungen berücksichtigen, tun dies mittels der Lagrange-Multiplikatoren. Das wahrscheinlich mächtigste Verfahren ist hier das Sequential Quadratic Programming (SQP), welches die Karush-Kuhn-Tucker-Bedingungen quadratisch approximiert.

Interior Point: Möglichkeit die KKT-Bedingungen zu erfüllen, um das Problem mit Nebenbedingungen zu einem unconstrained nichtlinearen Problem zu machen.

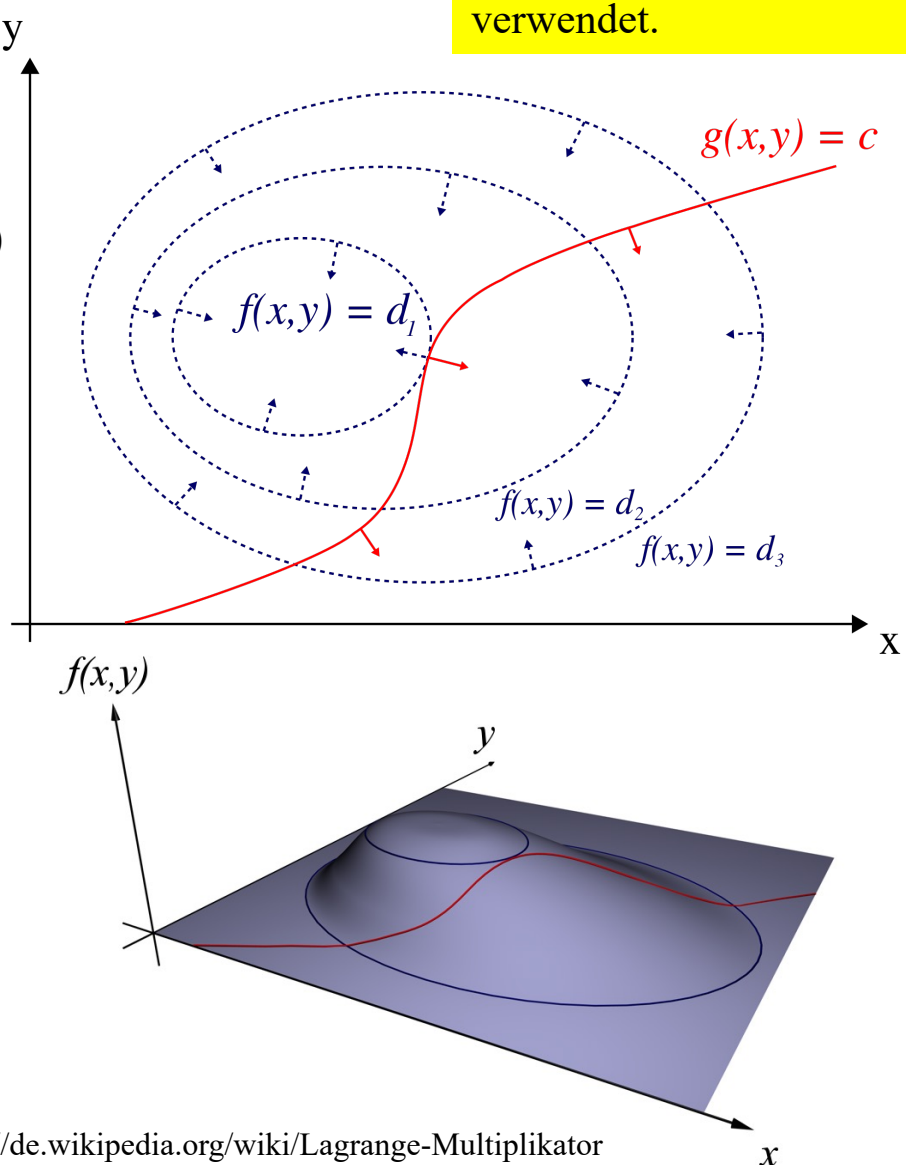
5.7 Nichtlineare Probleme mit Nebenbedingungen

Hier wird $f(\cdot)$, x und y anstelle von $I(\cdot)$, θ_1 und θ_2 verwendet.

Wichtige Eigenschaften der Lagrange-Funktion

- Es wird ein **Sattelpunkt (stationärer Punkt)** der Lagrange-Funktion gesucht:
  ein Minimum bzgl. $\underline{x}$ bzw. $\underline{\theta}$
  ein Maximum bzgl. $\underline{\lambda}$
- Das Bild zeigt in blau die Höhenlinien der Verlustfunktion in $x = x_1$, $y = x_2$ und in rot die Nebenbedingung
- Am Optimalpunkt bildet die Nebendingung eine Tangente zur niedrigsten Höhenline d_1
- An dieser Stelle sind die Gradienten entgegen gerichtet:

$$\underline{\nabla} f(x,y) = -\lambda \underline{\nabla} g(x,y)$$



Quelle: https://de.wikipedia.org/wiki/Lagrange-Multiplikator

5.7 Nichtlineare Probleme mit Nebenbedingungen

Beide Beispiele hier sind konvex und haben „nur“ lineare Nebenbedingungen. Sie ermöglichen einen einfachen, ersten Start.

Wichtige Eigenschaften der Dualität

- Da das **duale** Problem immer konkav ist, kann es relativ leicht über **konvexe** Optimierung (statt max $I \rightarrow$ min $-I$) gelöst werden und das **globale** Optimum wird gefunden.
- Die **Dimension** des dualen Problems (d.h. Anzahl an dualen Variablen) ist gleich der Anzahl an Nebenbedingungen des primalen Problems. Oft ist es einfacher mit hochdimensionalen Problemen umzugehen als mit sehr viele Nebenbedingungen.
- Die Anzahl an **Nebenbedingungen** des dualen Problems ist je nach Problemtyp anders; oft einfacher. Immer gelten aber die Nebenbedingungen, dass alle Lagrange-Multiplikatoren der Ungleichheits-nebenbedingungen $\lambda_i \geq 0$.

LP: Z.B. Erntemenge ist proportional zu vielen Produktionsfaktoren, wie Fläche, Arbeitszeit, Dünger, Wasser, ...

- **Beispiel 1:** Lineares Programm (LP) (Standardproblem aus dem Operations Research)

Primales Problem

$$\max_{\underline{\theta}} \ \underline{c}^T \underline{\theta}$$

s.t. $\underline{A}\,\underline{\theta} = \underline{b}\,, \quad \underline{\theta} \geq \underline{0}$

n Variablen, m Nebenbedingungen

Duales Problem

$$\max_{\underline{\lambda}} \ \underline{b}^T \underline{\lambda}$$

s.t. $\underline{A}^T \underline{\lambda} \leq \underline{c}$

m duale Variablen, n Nebenbedingungen

$n = \dim\{\underline{\theta}\}$
$m = \dim\{\underline{b}\}$

5.7 Nichtlineare Probleme mit Nebenbedingungen

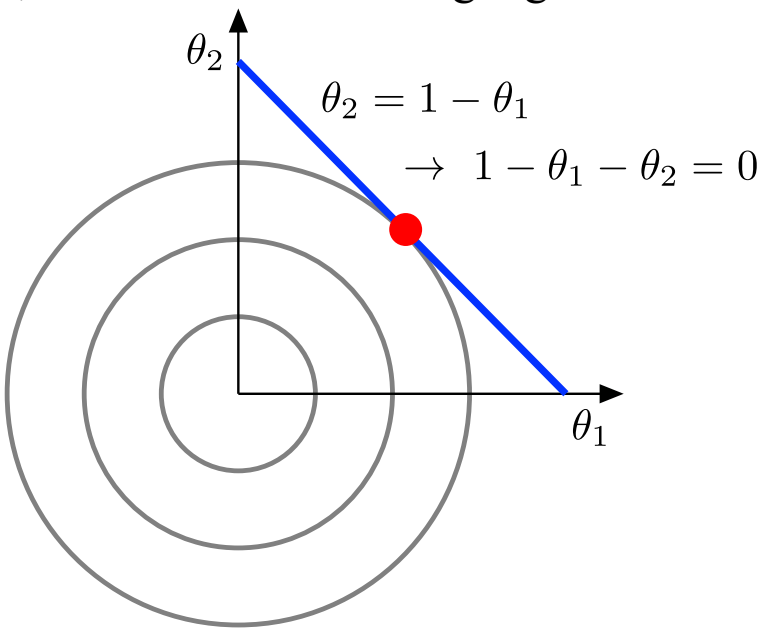
Optimum:
$\lambda^* = 1$
$\theta_1^* = \theta_2^* = 1/2$

Beispiel 2: 2-dimensionale quadratische Verlustfunktion, 1 lineare Nebenbedingung

$$I(\underline{\theta}) = \theta_1^2 + \theta_2^2 \rightarrow \min_{\underline{\theta}}$$

s.t. $1 - \theta_1 - \theta_2 = 0$

Primales Problem (2-D)



Lagrange-Funktion:

$$\mathcal{L}(\underline{\theta}, \lambda) = \theta_1^2 + \theta_2^2 + \lambda(1 - \theta_1 - \theta_2)$$

$$\frac{\partial \mathcal{L}}{\partial \theta_1} = 2\theta_1 - \lambda \;\rightarrow\; 2\theta_1 = \lambda$$

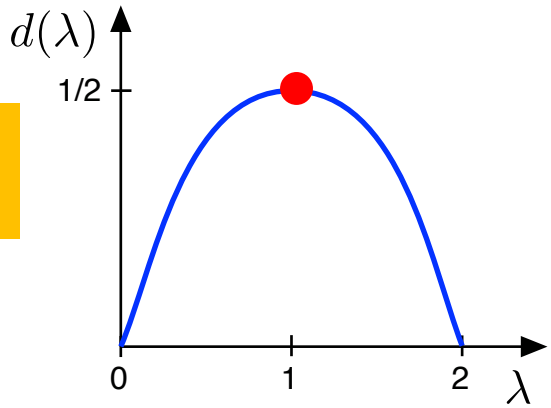
$$\frac{\partial \mathcal{L}}{\partial \theta_2} = 2\theta_2 - \lambda \;\rightarrow\; 2\theta_2 = \lambda$$

$$\frac{\partial \mathcal{L}}{\partial \lambda} = 1 - \theta_1 - \theta_2 = 0$$

θ_1 und θ_2 aus der Lagrange-Funktion eliminieren:

Duales Problem (1-D)

$$d(\lambda) = \left(\frac{\lambda}{2}\right)^2 + \left(\frac{\lambda}{2}\right)^2 + \lambda\left(1 - \frac{\lambda}{2} - \frac{\lambda}{2}\right) = -\frac{\lambda^2}{2} + \lambda \rightarrow \max_{\lambda}$$



5.7 Nichtlineare Probleme mit Nebenbedingungen

Interpretation der Lagrange-Multiplikatoren

- Besonders intuitiv im ökonomischen Kontext
- Zielfunktion: Gewinn eines Unternehmens oder Wohlstand einer Volkswirtschaft
- Nebenbedingungen: Limitationen auf Abgase, z.B. CO_2: $g(\underline{\theta}) \leq c$ ← Emissionslimit

$$\mathcal{L} = I(\underline{\theta}) + \lambda(c - g(\underline{\theta}))$$

$$\rightarrow \quad \frac{\partial \mathcal{L}}{\partial c} = \lambda \quad \text{über den Umhüllungssatz gilt auch:} \quad \frac{\partial I(\underline{\theta}^*)}{\partial c} = \lambda^*$$

am Optimum

- Was bedeutet das? Die Größe von λ ist der Preis, den man für eine Reduktion von c zahlen muss (Gewinnrückgang bzw. Wohlstandsverlust). Je größer λ^*, desto empfindlicher reagiert die Lagrange-Funktion und die Verlustfunktion darauf. In der Ökonomie verwendet man typischerweise *Nutzen*funktionen, die maximiert werden statt *Verlust*funktionen, die minimiert werden – das Prinzip bleibt aber das Gleiche, nur die Vorzeichen drehen sich.
- λ nennt man in der Mikroökonomie und Wohlfahrtsökonomik **Schattenpreis**.

5.7 Nichtlineare Probleme mit Nebenbedingungen

Null-Summen-Spiel

- Bei einem Spiel (z.B. Schach oder Mühle) versucht der eine Spieler einen möglichst großen Wert für I zu erreichen, maximiert also. Der Gegenspieler versucht einen möglichst kleinen Wert für I zu erreichen, minimiert also.
- Jeder Spieler muss ich für den besten Zug des Gegenübers wappnen. Daher handelt es sich um Maximin- bzw. Minimax-Optimierungsprobleme.
- Der Duality Gap kann als der Vorteil interpretiert werden, den Zug des Gegners zu kennen.

Optimale Regelung

- Zielfunktion: Regelperformance, z.B. als Integral über den quadratischen Regelfehler
- Nebenbedingung: Systemdynamik in Zustandsraumdarstellung, die der Zustand erfüllen muss.
- Primale Variablen Zustand $\underline{x}(t)$ und **duale** Variablen **Kozustand** (*costate*) $\underline{\lambda}(t)$ sind *zeitabhängig*. Der Kozustand folgt der sog. **adjungierten** Differentialgleichung.

5.7 Nichtlineare Probleme mit Nebenbedingungen

2D-Beispiel mit Lagrange-Funktionen

Problem:

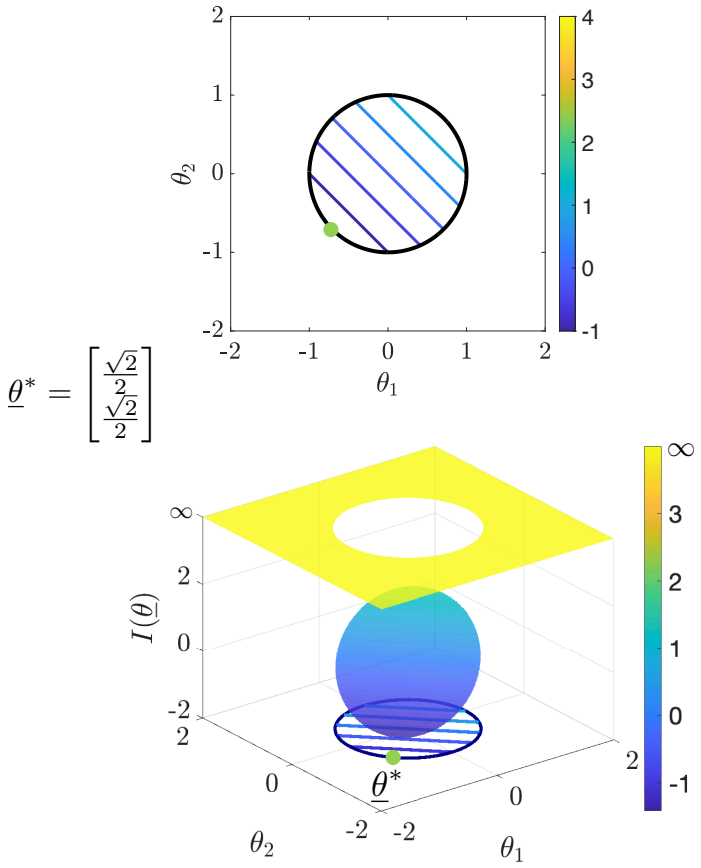
$$I(\underline{\theta}) = \theta_1 + \theta_2$$
$$\text{s.t. } g(\underline{\theta}) = \theta_1^2 + \theta_2^2 - 1 \leq 0$$

$$\min_{\underline{\theta}} I(\underline{\theta}) + P(\underbrace{\theta_1^2 + \theta_2^2 - 1}_{y(\underline{\theta})})$$

$$\min_{\underline{\theta}} \left((\theta_1 + \theta_2) + \max_{\lambda \geq 0} \lambda \, (\theta_1^2 + \theta_2^2 - 1) \right)$$

Der max-Operator kann nun nach vorne gezogen werden (kein λ in $I(\underline{\theta})$)

$$\min_{\underline{\theta}} \left(\max_{\lambda \geq 0} \underbrace{(\theta_1 + \theta_2 + \lambda \, (\theta_1^2 + \theta_2^2 - 1))}_{\mathcal{L}(\underline{\theta}, \lambda)} \right)$$



5.7 Nichtlineare Probleme mit Nebenbedingungen

Lösen des Beispielproblems via dualem Problem (im allgemeinen auch nicht so einfach möglich)

$$\max_{\lambda \geq 0} \left(\min_{\underline{\theta}} \underbrace{(\theta_1 + \theta_2 + \lambda\,(\theta_1^2 + \theta_2^2 - 1))}_{\mathcal{L}(\underline{\theta},\lambda)} \right)$$

Lösen des Minimierungsproblems:

$$\frac{\partial \mathcal{L}(\underline{\theta},\lambda)}{\partial \underline{\theta}} = \begin{bmatrix} \frac{\partial \mathcal{L}(\underline{\theta},\lambda)}{\partial \theta_1} \\ \frac{\partial \mathcal{L}(\underline{\theta},\lambda)}{\partial \theta_2} \end{bmatrix} = \begin{bmatrix} 1 + 2\lambda\theta_1 \\ 1 + 2\lambda\theta_2 \end{bmatrix} \overset{!}{=} 0 \;\rightarrow\; \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} = \begin{bmatrix} -\frac{1}{2\lambda} \\ -\frac{1}{2\lambda} \end{bmatrix}$$

Einsetzen in Maximierungsproblem

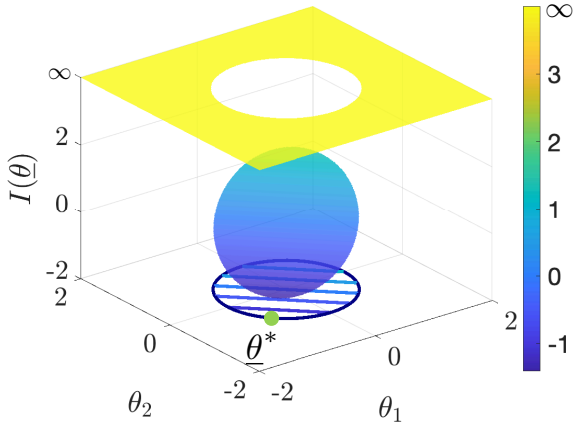
$$\max_{\lambda \geq 0} \left(-\frac{2}{2\lambda} + \lambda \left(\frac{2}{4\lambda^2} - 1 \right) \right) = \max_{\lambda \geq 0} -\frac{1}{2\lambda} - \lambda$$

Lösen des Maximierungsproblems liefert

$$\frac{\mathrm{d}}{\mathrm{d}\lambda} \left(-\frac{1}{2\lambda} - \lambda \right) \overset{!}{=} 0 \quad \rightarrow \quad \lambda = \frac{\sqrt{2}}{2}$$

Einsetzen in obige Beziehung zwischen θ und λ liefert

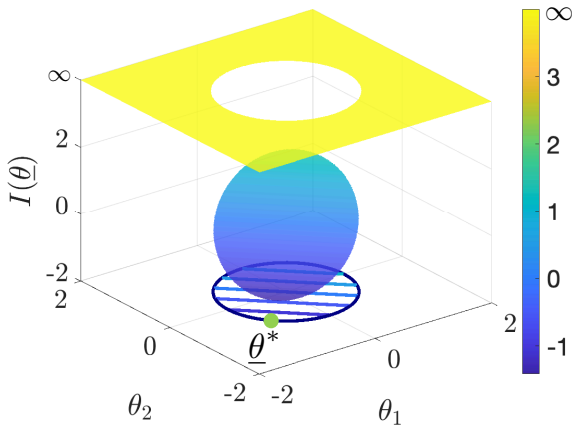
$$\underline{\theta}^* = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} = \begin{bmatrix} -\frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} \end{bmatrix}$$



5.7 Nichtlineare Probleme mit Nebenbedingungen

Lösen des Beispielproblems via des primalen Problems

$$\min_{\underline{\theta}} \left(\max_{\lambda \geq 0} \underbrace{(\theta_1 + \theta_2 + \lambda\,(\theta_1^2 + \theta_2^2 - 1))}_{\mathcal{L}(\underline{\theta},\lambda)} \right)$$



Lösen des Maximierungsproblems:

$$\frac{\mathrm{d}\mathcal{L}(\underline{\theta},\lambda)}{\mathrm{d}\lambda} = (\theta_1^2 + \theta_2^2 - 1) \stackrel{!}{=} 0 \quad \rightarrow \quad \theta_1 = \pm\sqrt{1-\theta_2^2}$$

Einsetzen in Minimierungsproblem

$$\min_{\theta_2} \left(\pm\sqrt{1-\theta_2^2} + \theta_2 + \cancel{\lambda(1-\theta_2^2+\theta_2^2-1)} \right)$$

Lösen des Minimierungsproblems liefert

$$\frac{\mathrm{d}}{\mathrm{d}\theta_2}\left(\pm(1-\theta_2^2)^{\frac{1}{2}} + \theta_2\right) = \pm\frac{\cancel{2}\theta_2}{\cancel{2}\sqrt{1-\theta_2^2}} + 1 \stackrel{!}{=} 0 \quad \rightarrow \quad \theta_2 = \pm\frac{\sqrt{2}}{2}$$

Einsetzen in θ_1 liefert die beiden Kandidaten

$$\underline{\theta}^{[-]} = \begin{bmatrix} -\frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} \end{bmatrix} \qquad \underline{\theta}^{[+]} = \begin{bmatrix} \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} \end{bmatrix}$$

einsetzen in Verlustfunktion führt zur Lösung:

$$\rightarrow \underline{\theta}^* = \underline{\theta}^{[-]} = \begin{bmatrix} -\frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} \end{bmatrix}$$

5.7 Nichtlineare Probleme mit Nebenbedingungen

- **Lagrange Multipliers | Geometric Meaning & Full Example von Dr. Trefor Bazett:**
  https://www.youtube.com/watch?v=8mjcnxGMwFo
- **Mini-Serie zu konvexer Optimierung (inkl. KKT Bedingungen) von Visually Explained:**
  1) What Is Mathematical Optimization?
  https://www.youtube.com/watch?v=AM6BY4btj-M

  2) Convexity and The Principle of Duality
  https://www.youtube.com/watch?v=d0CF3d5aEGc

  3) The Karush–Kuhn–Tucker (KKT) Conditions and the Interior Point Method for Convex Optimization
  https://www.youtube.com/watch?v=uh1Dk68cfWs

5.8 Globale Suchverfahren

Die bisherigen nichtlinearen Optimierungsverfahren sind alle **lokale** Verfahren. Dies bedeutet, dass Informationen der direkten Umgebung (Gradient, Hesse-Matrix) zur Optimierung verwendet werden. Alle diese Verfahren konvergieren zu einem lokalen Minimum. Es ist aber völlig unklar, ob andere initiale Parameter nicht zu einem viel besseren lokalen Optimum hätten führen können.

Globale Suchverfahren werden eingesetzt, um nicht nur ein lokales sondern das **globale** Minimum (oder zumindest ein **gutes lokales** Minimum) zu finden.

Multistart

Der Multistart-Ansatz ist dabei der triviale. Man startet unabhängige lokale Optimierungen von unterschiedlichen initialen Parametern aus und wählt aus allen erreichten Minima, dasjenige mit dem kleinsten Verlustfunktionswert.

Parameter verrauschen

In jeder Iteration werden die Parameter (leicht) verrauscht. Dies verlangsamt die Konvergenz, kann aber dazu führen, dass der Optimierer aus flachen lokalen Optima entkommt.

5.8 Globale Suchverfahren

Wann sollten globale Suchverfahren eingesetzt werden?

- Suche nach einem sehr guten lokalen oder Suche nach dem yglobalen Optimum wird benötigt
- Oberfläche der Verlustfunktion oder der Ableitungen ist sehr unglatt
- Einige Parameter sind nicht reell sondern ganze Zahlen oder binäre Größen etc.
- Es handelt sich um kombinatorische Optimierungsprobleme

Nachteile

- Sehr großer Rechenaufwand

Globale Suchverfahren sind gut, um Regionen zu finden,
lokale Verfahren sind gut, um Punkte zu finden

→ Globale Suchverfahren konvergieren langsam zu einem exakten Parametervektor (Punkt).

5.8 Globale Suchverfahren

Mögliche Kombinationen

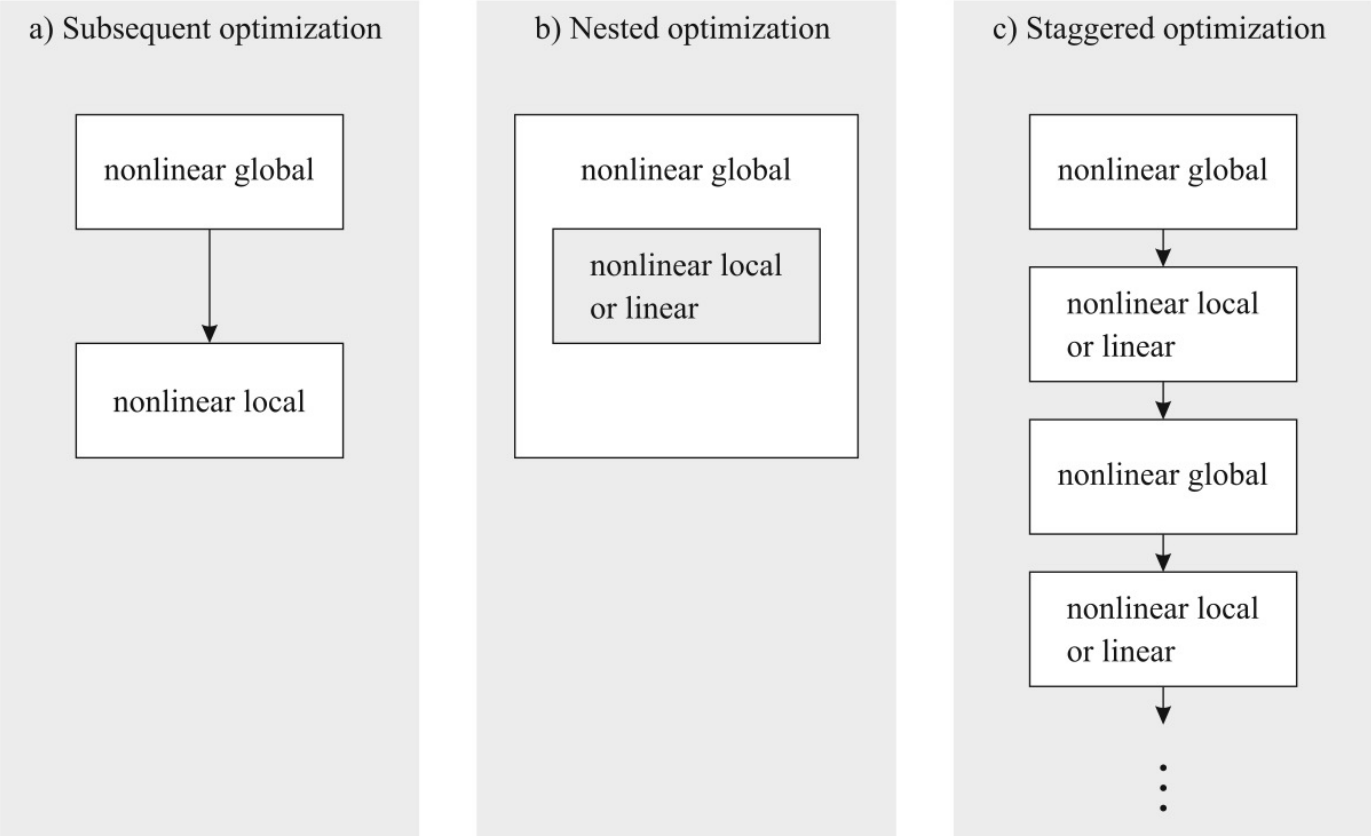


Fig. 5.1 Combinations of different optimization techniques: (**a**) global optimization of all parameters and a subsequent local optimization of all parameters, (**b**) global optimization of some parameters or structures and within (in each iteration) a nonlinear local or linear optimization of the other parameters, (**c**) repeated global optimization of some parameters and subsequently nonlinear local or linear optimization of the other parameters

5.8 Globale Suchverfahren

Simulated Annealing (SA)

- Idee: Ein warmes Partikel wird in einem Potentialfeld simuliert. Grundsätzlich bewegt sich das Partikel zu geringeren potentiellen Energien. Da es aber eine gewisse Temperatur hat (es besitzt kinetische Energie) bewegt es sich zum gewissen Teil zufällig und springt ab und zu zu höheren Energielevels. Der Grad der zufälligen Bewegung hängt von der Temperatur T ab:
  T hoch → große zufällige Bewegung, **T niedrig → kleine zufällige Bewegung**
  Somit kann dieses Verfahren lokalen Optima entfliehen, je höher T, desto besser.
- Das Partikel wird über die Iterationen hinweg nach einem **Annealing Schedule** „abgekühlt“, d.h. die stochastische Bewegung wird weniger im Laufe der Zeit und die Wahrscheinlichkeit zu höheren Energielevels zu steigen sinkt.
- Im Falle der Optimierung entspricht das Partikel dem Punkt im Parameterraum und die potentielle Energie entspricht der Verlustfunktion.
- Spezielle Form ist Boltzmann Annealing (BA), für das garantiert werden kann, dass in unendlicher Zeit das globale Optimum gefunden werden kann. Wichtig zu wissen: Man bleibt in keinem lokalen Optimum “stecken“.

5.8 Globale Suchverfahren

Simulated Annealing (SA)

1. Choose an initial "large enough" temperature T_0.
2. Choose an initial parameter vector $\underline{\theta}_0$, that is, a point in search space.
3. Evaluate the loss function value for the initial parameters $I(\underline{\theta}_0)$.
4. Iterate for $k = 1, 2, \ldots$
5. Generate a new point in search space $\underline{\theta}_{\text{new}}$, which has the deviation $\Delta\underline{\theta}_k = \underline{\theta}_{\text{new}} - \underline{\theta}_{k-1}$ from the old point with the generation probability density function $g(\Delta\underline{\theta}_k, T_k)$.
6. Evaluate the loss function value for the new parameters $I(\underline{\theta}_{\text{new}})$.
7. Accept this new point with an acceptance probability $h(\Delta I_k, T_k)$ where $\Delta I_k = I(\underline{\theta}_{\text{new}}) - I(\underline{\theta}_{k-1})$, that is, set $\underline{\theta}_k = \underline{\theta}_{\text{new}}$, or otherwise keep the old point, that is, set $\underline{\theta}_k = \underline{\theta}_{k-1}$.
8. Decrease the temperature according to the annealing schedule T_k.
9. Test for the termination criterion and either go to Step 4 or stop.

5.8 Globale Suchverfahren

Evolutionäre Algorithmen (EA)

- Idee basiert auf Evolution.
- Population von Individuen (unterschiedliche Punkte im Parameterraum)
- SA: Ein Partikel entspricht einem Punkt im Parameterraum
- EA: Viele Individuen werden gleichzeitig (parallel) verwendet
- Neue Individuen entstehen aus Mutation und Rekombination

Evolutionary algorithm	Classical optimization
Individual	Parameter vector
Population	Set of parameter vectors
Fitness	Inverse of loss function value
Fitness function	Inverse of loss function
Generation	Iteration
Application of genetic operators	Parameter vector update

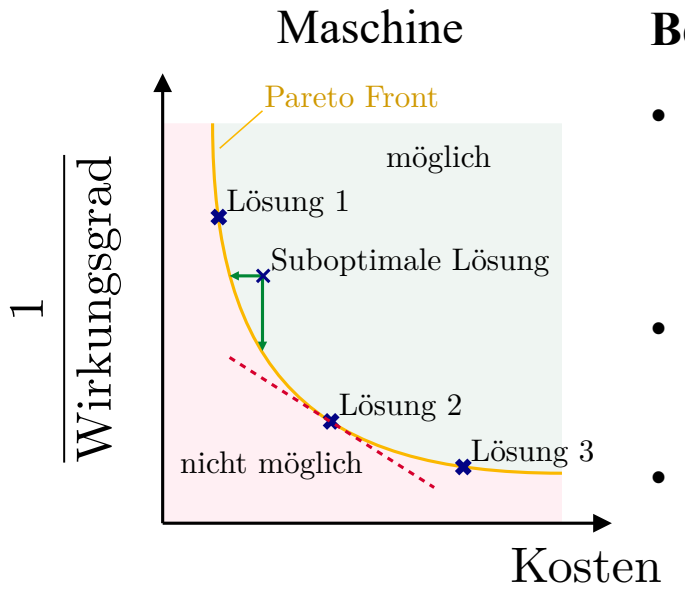
1. Choose an initial population of S individuals (parameter points) $\underline{\Theta}_0 = [\underline{\theta}_{1,0}\ \underline{\theta}_{2,0}\ \cdots\ \underline{\theta}_{S,0}]$.
2. Evaluate the fitness (some inverse of the loss function) of all individuals of the initial population $\mathrm{Fit}(\underline{\Theta}_0)$.
3. Iterate for $k = 1, 2, \ldots$
4. Perform selection $\underline{\Theta}_{\text{new}} = \mathrm{Select}(\underline{\Theta}_{k-1})$.
5. Apply genetic operators $\underline{\Theta}_k = \mathrm{GenOps}(\underline{\Theta}_{\text{new}})$ (mutation, recombination, etc.).
6. Test for the termination criterion and either go to Step 3 or stop.

5.9 Multikriterielle Optimierung

Oft kommt es vor, dass nicht nur ein Kriterium sondern mehrere Kriterien gleichzeitig optimiert werden sollen. Beispiele:

- Reglerperformance vs. geringer Stellaufwand in der Regelungstechnik
- Effizienz eines Verbrennungsmotors vs. NO_X-Emissionen
- Produktqualität vs. Produktkosten
- Erwarteter Gewinn vs. Risiko eines Investments
- Modellperformance vs. Modellkomplexität

Statt dem inversen Wirkungsgrad hätte man auch 1-Wirkungsgrad o.ä. an der senkrechten Achse abtragen können.



Beispiel

- Lösungen 1, 2 und 3 liegen auf der **Pareto-Front**. Für die jeweiligen Kosten gibt es keine Maschine mit besserem Wirkungsgrad und anders herum.
- Eine Lösung ist suboptimal, wenn zu gleichen Kosten ein höherer Wirkungsgrad möglich wäre (und anders herum).
- Die **Steigung** der Kurve (gestrichelt rote Linie bei Lösung 2) in einem Punkt gibt an, wieviel Mehrkosten zu wieviel Wirkungsgradsteigerung führen (und anders herum).

5.9 Multikriterielle Optimierung

Berücksichtigung von mehreren Zielen

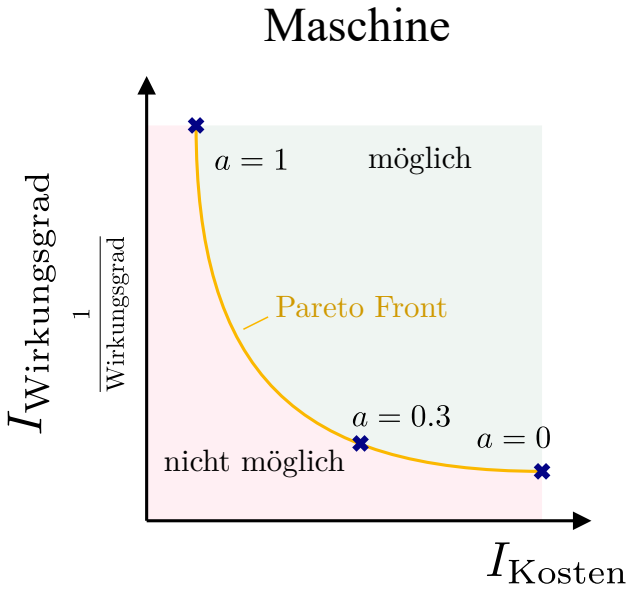
Neben echten multikriteriellen (*multi-objective*) Optimierungsverfahren, welche die **gesamte Pareto-Front** erkunden, kann man das Problem aus auf gewöhnliche Optimierungen zurückführen:

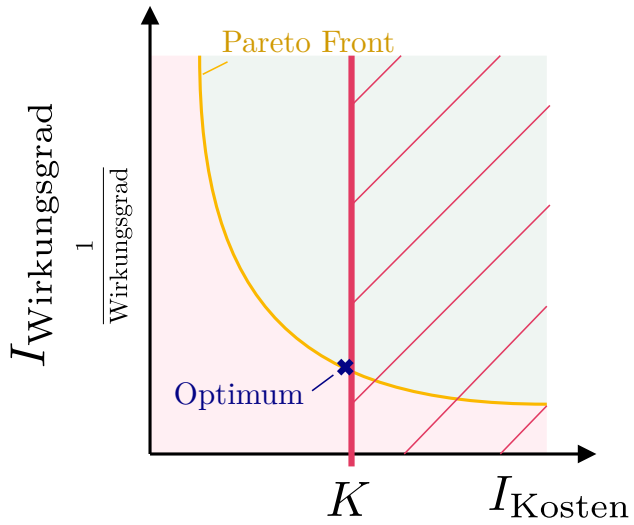
- Verlustfunktion als **gewichtete Summe**: Eine einfache Möglichkeit mehrere Ziele zu berücksichtigen ist durch die Gewichtung der unterschiedlichen Ziele in einer Verlustfunktion, bspw.:

$$I_{\text{Gesamt}} = a\, I_{\text{Kosten}} + (1-a) I_{\text{Wirkungsgrad}} \quad .$$

- Durch **Nebenbedingungen**: Mehrere Ziele können durch Nebenbedingungen berücksichtigt werden. Es könnte bspw. nur bezüglich des Wirkungsgrades optimiert werden unter der Nebenbedingung, dass die Kosten unter einem Maximalwert $I_{\text{Kosten}} < K$ liegen:

$$\min I_{\text{Wirkungsgrad}} \quad \text{s.t. } I_{\text{Kosten}} < K$$





5 Literaturhinweise

- Kapitel 7, M. P. Deisenroth, A. A. Faisal, and C. S. Ong, Mathematics for Machine Learning. Cambridge: Cambridge University Press, 2020. https://mml-book.com
- Kapitel 2, 4, 5, O. Nelles, Nonlinear System Identification, 2 ed. Springer International Publishing, 2020. doi: https://doi.org/10.1007/978-3-030-47439-3.
- R. Fletcher, Practical Methods of Optimization. John Wiley & Sons, Ltd, 2000. doi: 10.1002/9781118723203.
- L. E. Scales, *Introduction to Non-Linear Optimization*. Macmillan Education UK, 1985. doi: 10.1007/978-1-349-17741-7.