

Algorithmik kontinuierlicher Systeme

Nichtlineare Optimierung



- Optimierung ist ein Grundproblem (nicht nur) der Informatik
- Diskrete Optimierung
 - ▶ Diskrete Variablen (z.B. binär, ganzzahlig)
 - ▶ Endlich viele „Zustände“ zu durchsuchen
 - ▶ Enge Beziehungen zu diskreten Strukturen, wie Bäumen, Graphen,
 - ▶ z.B. *Dynamisches Programmieren* (bekannt aus Vorlesung AuD)
- Kontinuierliche Optimierung
 - ▶ Kontinuierliche Variablen
 - ▶ Unbeschränkte vs. Optimierung mit Nebenbedingungen
 - ▶ Enge Verwandtschaft zur Nullstellensuche
 - ▶ Bei dynamischen Systemen: Optimalsteuerung - Bezüge zur Regelungstechnik

- Gegeben eine **skalare Kostenfunktion** oder **Zielfunktion**

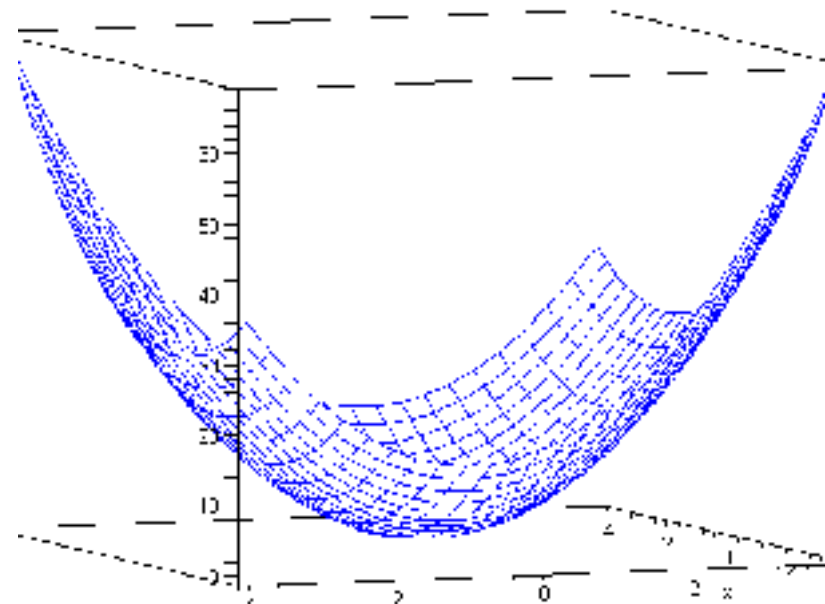
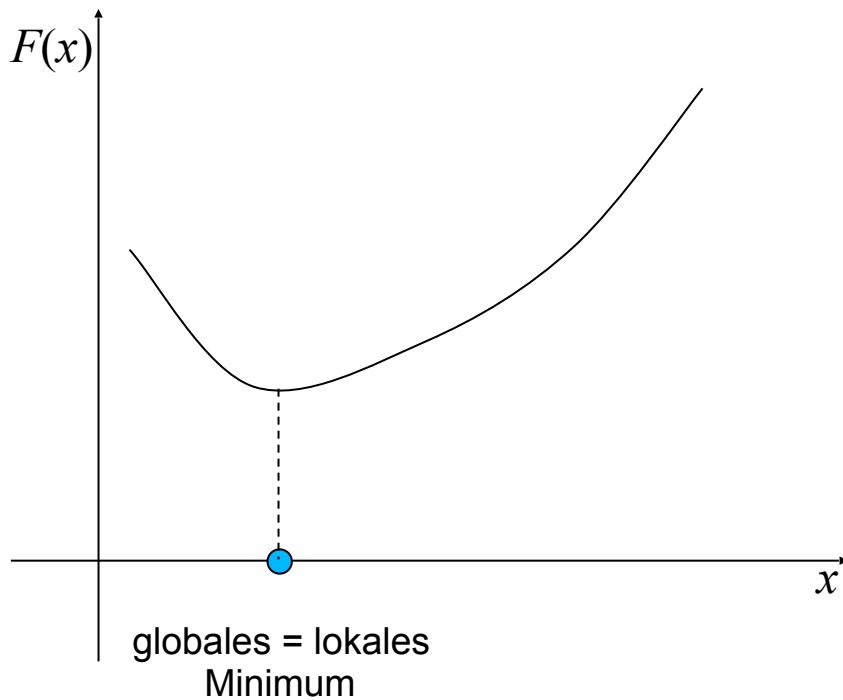
$$F : \mathbb{R}^n \rightarrow \mathbb{R}$$
 von mehreren Veränderlichen (x_1, x_2, \dots, x_n)
- und ein zulässiger Bereich $X \subseteq \mathbb{R}^n$
- Gesucht eine/die Stelle $x^* \in X$ so dass

$$F(x^*) \leq F(x) \quad \text{für alle } x \in X$$
- Bezeichnung $x^* = \mathbf{argmin}_{x \in X} \{ F(x) \}$
 (sofern eindeutig lösbar!)
- Varianten:
 - ▶ Maximum statt Minimum : $-F$ statt F
 - ▶ vektorwertige Kostenfunktion \rightarrow multi-kriterielle Optimierung

- Geometrische Interpretation der Kostenfunktion F

- 1D : Graph = $\{(x, F(x))\}$

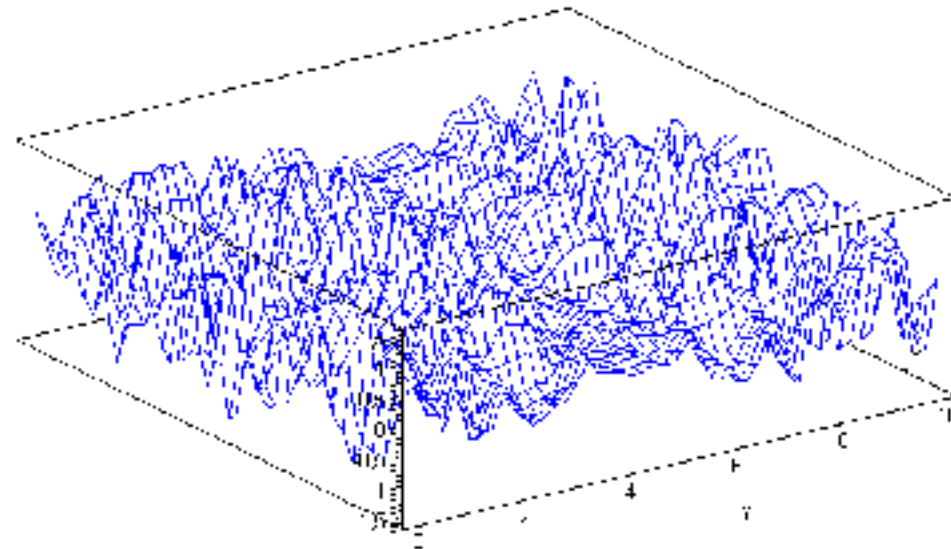
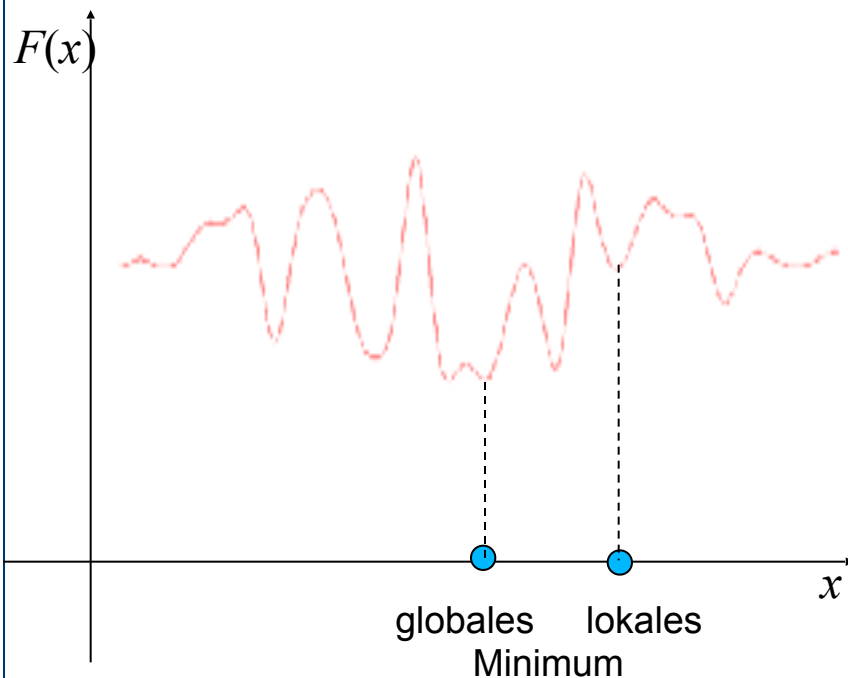
- 2D : $\{(x, y, F(x,y))\}$
(Höhenfeld „Gebirge“)



- Geometrische Interpretation der Kostenfunktion F

• 1D : Graph = $\{(x, F(x))\}$

2D : $\{(x, y, F(x,y))\}$
(Höhenfeld „Gebirge“)



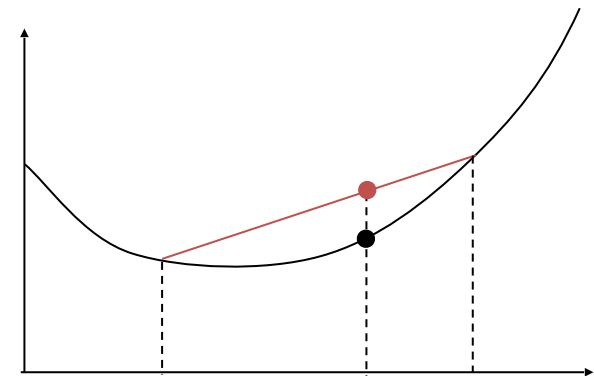
- Der zulässige Bereich X ist meist von folgender Form

$$X = \{ x \in \mathbb{R}^n : g_1(x) \leq 0, \dots, g_m(x) \leq 0 \}$$

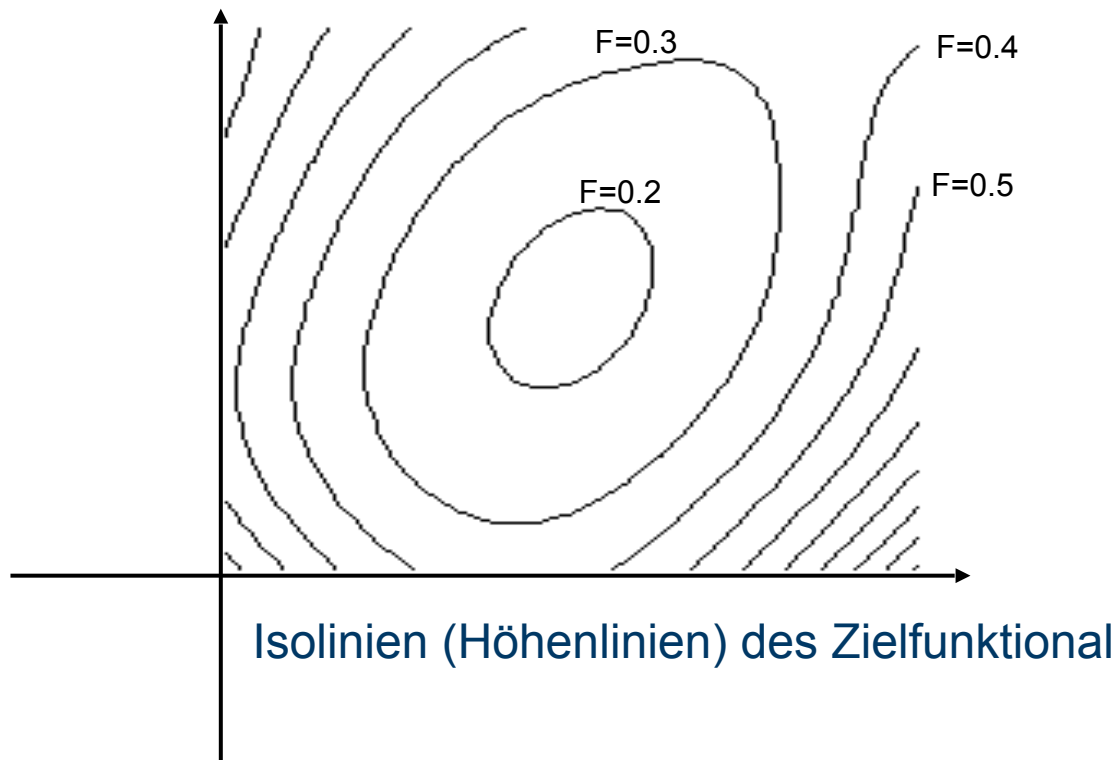
- Die Forderungen $g_i(x) \leq 0$ heißen **Nebenbedingungen**
- Ungleichungs-Nebenbedingungen
- Beachte:
Eine Gleichungs-Nebenbedingung „ $g(x) = 0$ “ kann durch die beiden Bedingungen „ $g(x) \leq 0$ “ und „ $-g(x) \leq 0$ “ realisiert werden

Beispiele

- $x_j \geq a_j$ bzw. $x_j \leq b_j$ bzw. $a_j \leq x_j \leq b_j$
- $\sum_i x_i^2 \leq R^2$ bzw. $\sum_i x_i^2 = R^2$
- lineare Nebenbedingung: $\sum_i a_i x_i \leq c$
- $X = \{x : g(x) \leq c\}$ ist *konvex* sofern g **konvexe Funktion** d.h.
 $g((1-t)x_1 + tx_2) \leq (1-t)g(x_1) + tg(x_2)$
 (falls $0 < t < 1$)

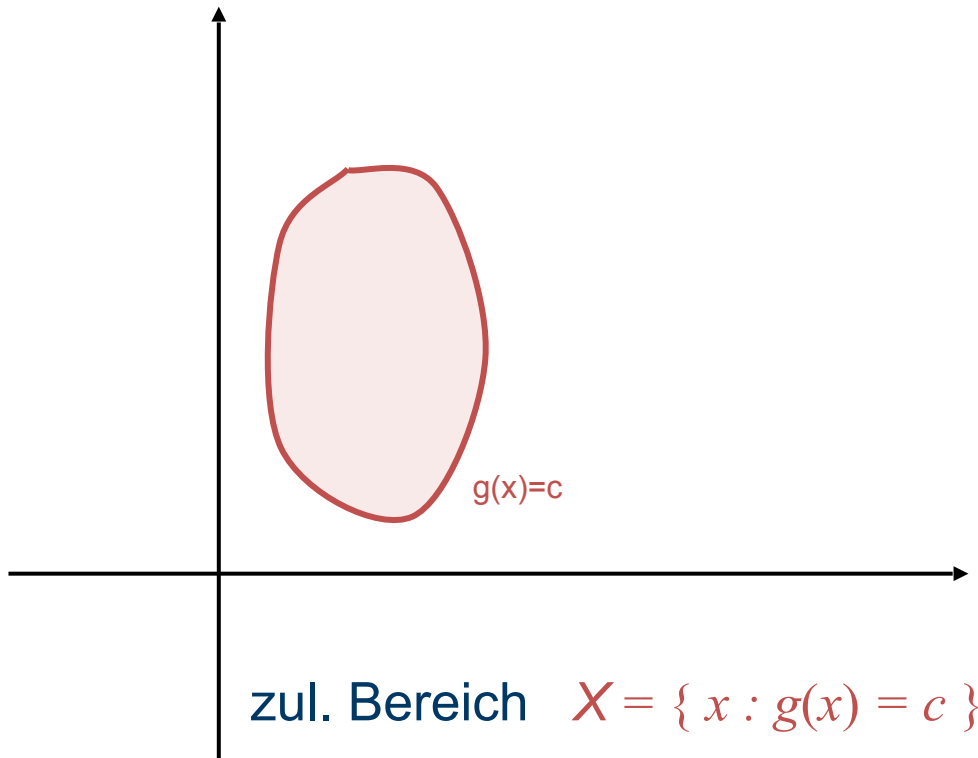


- $\min \{ F(x,y) \}$ unter der Nebenbedingung $g(x,y) = c$

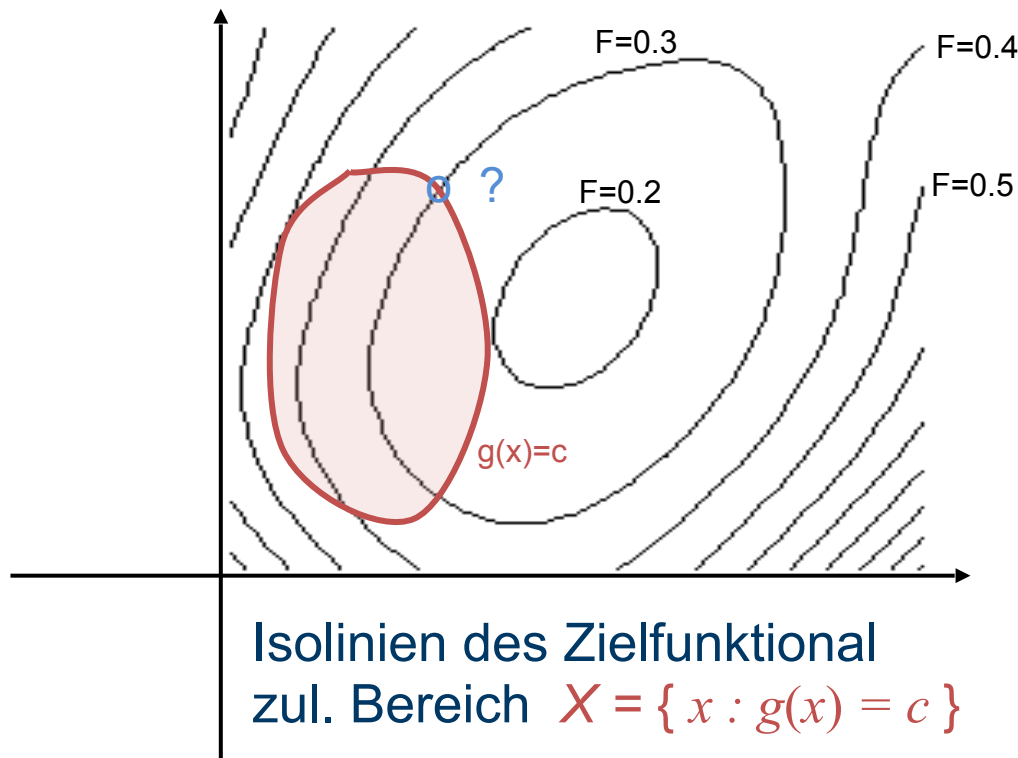


Isolinien (Höhenlinien) des Zielfunktional

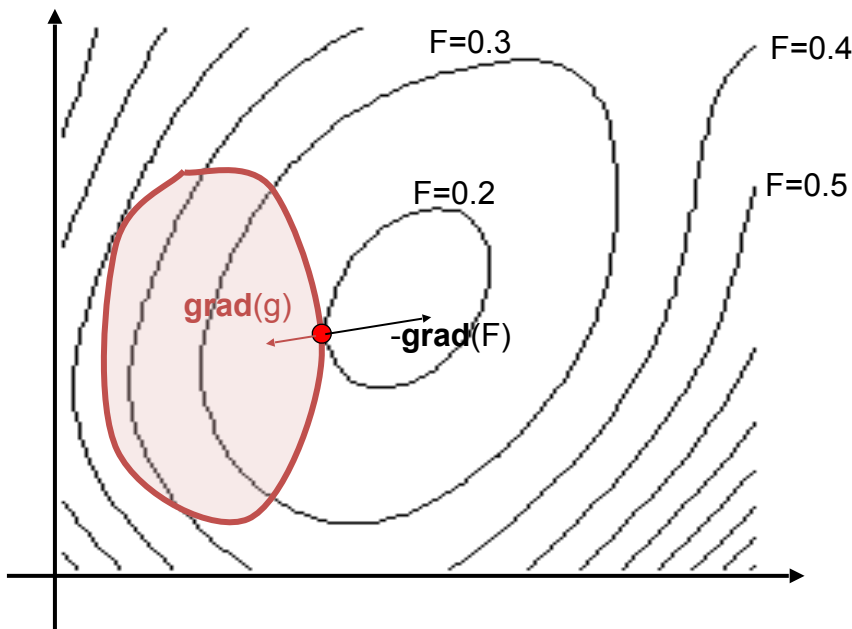
- $\min \{ F(x,y) \}$ unter der Nebenbedingung $g(x,y) = c$



- $\min \{ F(x,y) \}$ unter der Nebenbedingung $g(x,y) = c$



- $\min \{ F(x,y) \}$ unter der Nebenbedingung $g(x,y) = c$
- die Isolinien berühren sich tangentiell
- gleichwertig dazu: **die Gradienten sind parallel**



Lösung (x^*, y^*)

$\text{grad}(F)(x^*, y^*) \parallel \text{grad}(g)(x^*, y^*)$

Lösung:

$$\text{grad}(F)(x,y) = \lambda \text{grad}(g)(x,y)$$

$$g(x,y) = c$$

3 Gleichungen für (x^*, y^*, λ)

λ Lagrange Multiplikator

- Das Minimum/Maximum x^* einer diff.-baren Funktion

$$F : \mathbb{R}^n \rightarrow \mathbb{R}$$

unter den Nebenbedingungen

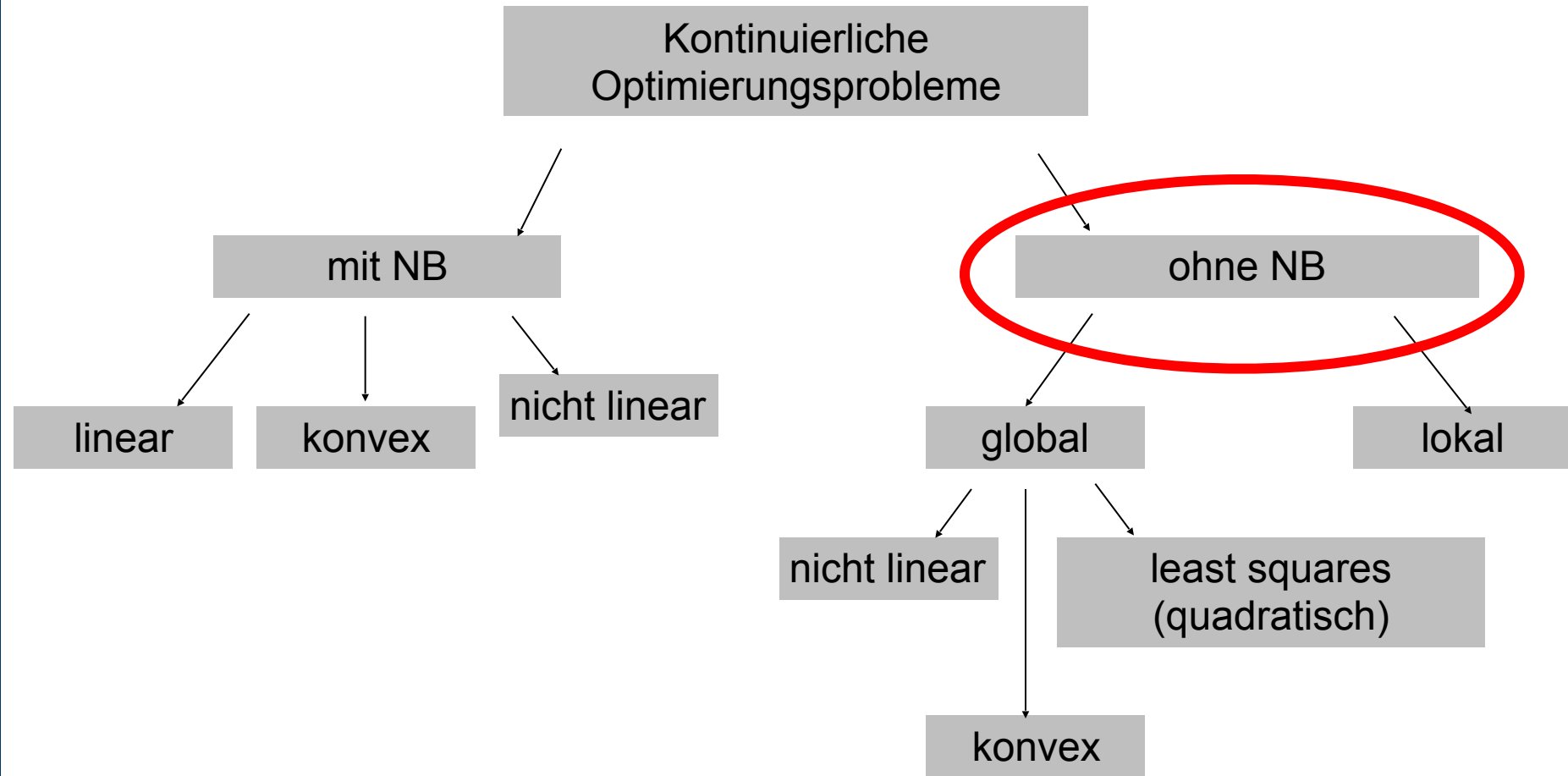
$$g_1(x) = c_1, g_2(x) = c_2, \dots, g_m(x) = c_m$$

erfüllt folgende Bedingung:

- Es gibt Skalare $(\lambda_1, \lambda_2, \dots, \lambda_m)$ so dass

$$\mathbf{grad}(F)(x^*) = \lambda_1 \mathbf{grad}(g_1)(x^*) + \dots + \lambda_m \mathbf{grad}(g_m)(x^*)$$

- Dies sind n Gleichungen. Zusammen mit den m Nebenbedingungen hat man also $n+m$ Gleichungen für die $n+m$ Unbekannten $x_1^*, \dots, x_n^*, \lambda_1, \dots, \lambda_m$



Bekanntlich gilt für differenzierbare Kostenfunktion F

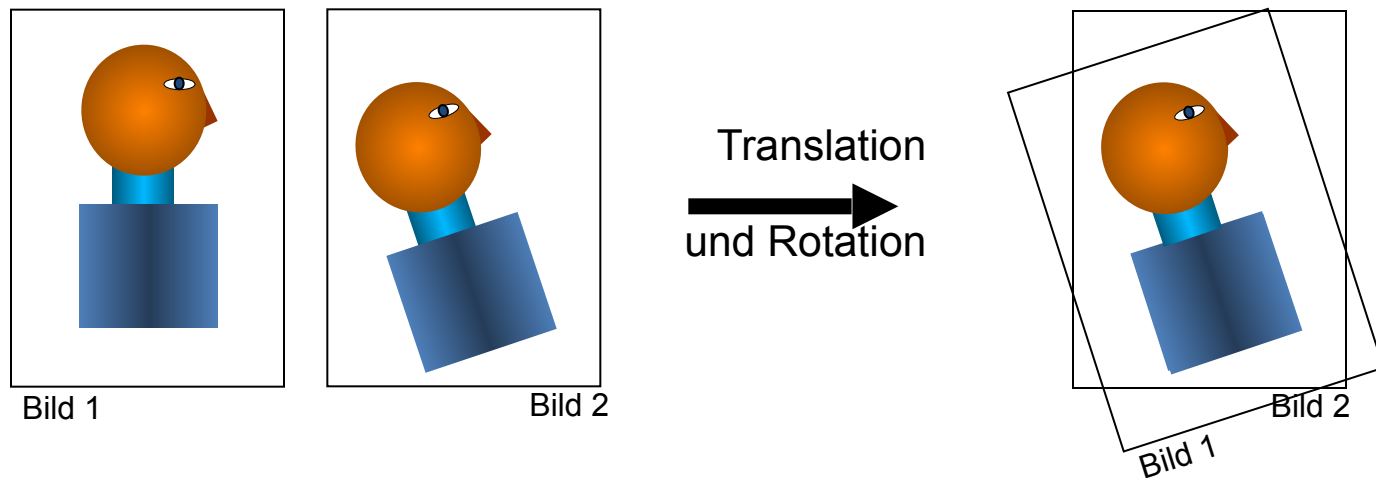
- x^* lokales Minimum $\Rightarrow \mathbf{grad}(F)(x^*) = 0$

- $\mathbf{grad}(F)(x^*) = 0$ & $D^2F(x^*)$ positiv definit
 $\Rightarrow x^*$ lokales Minimum von F

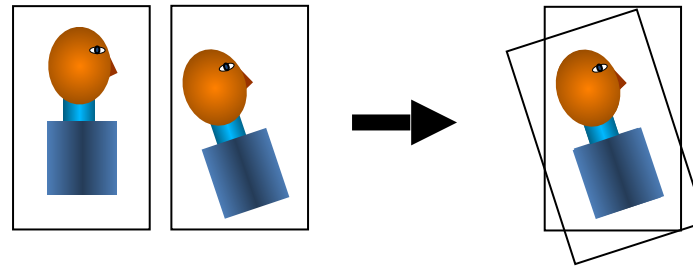
dabei ist $D^2F(x^*) = \left[\frac{\partial^2 F}{\partial x_i \partial x_j} \right] = \mathbf{H}_F(x^*) = \mathbf{Hesse-Matrix}$

- Oft begnügt man sich damit, die notwendige Bedingung
 $\mathbf{grad}(F)(x^*) = 0$
zu lösen. Z.B. aus spezieller Problemstellung
(Energieminimierung) folgt Optimalität.

- Zwei verschiedene Aufnahmen zu des gleichen Objekts
 - zeitlich versetzt
 - unterschiedliche Modalität (CT - MR)
- Bringe ähnliche Bilder geometrisch zur Deckung

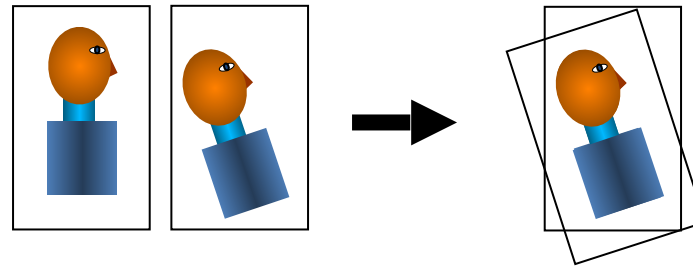


- Bringe die Bilder geometrisch zur Deckung



- Die Veränderlichen sind die Parameter, die die relative Lage beschreiben: starre Bewegung
Translation um t und Rotation um ϕ
→ 3-Parameter-Problem
- Besonders interessant und wichtig in Medizin für Tomographiedatensätze (3D-Daten)
→ 6-Parameter-Problem
(3D Verschiebung + 3 Eulerwinkel)

- Bringe die Bilder geometrisch zur Deckung



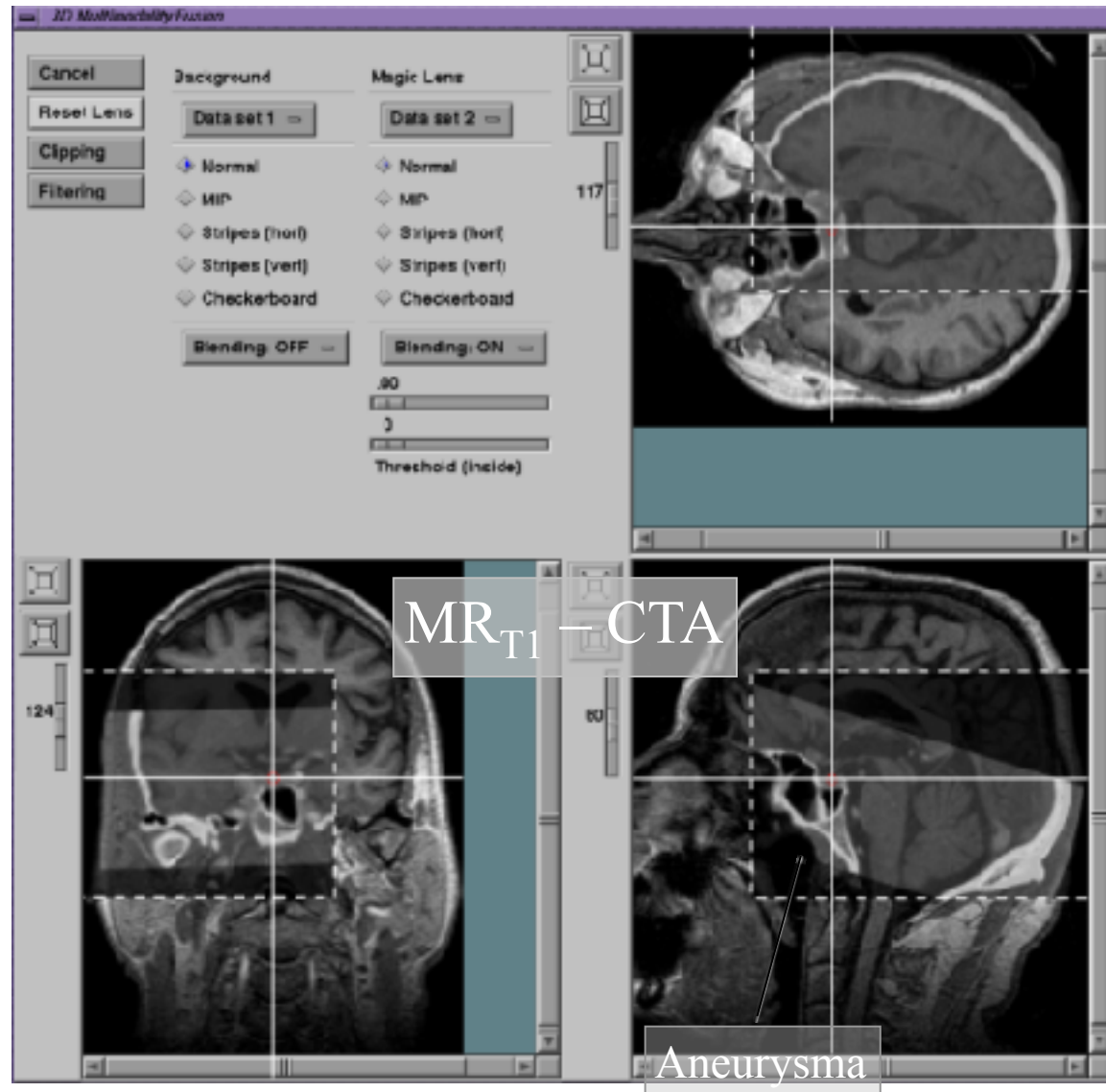
- Die Kostenfunktion $F(t, \phi)$ „misst“ den Abstand zwischen dem transformierten Bild 1 und Bild 2

z.B. Differenz der Grau-/Farbwerte

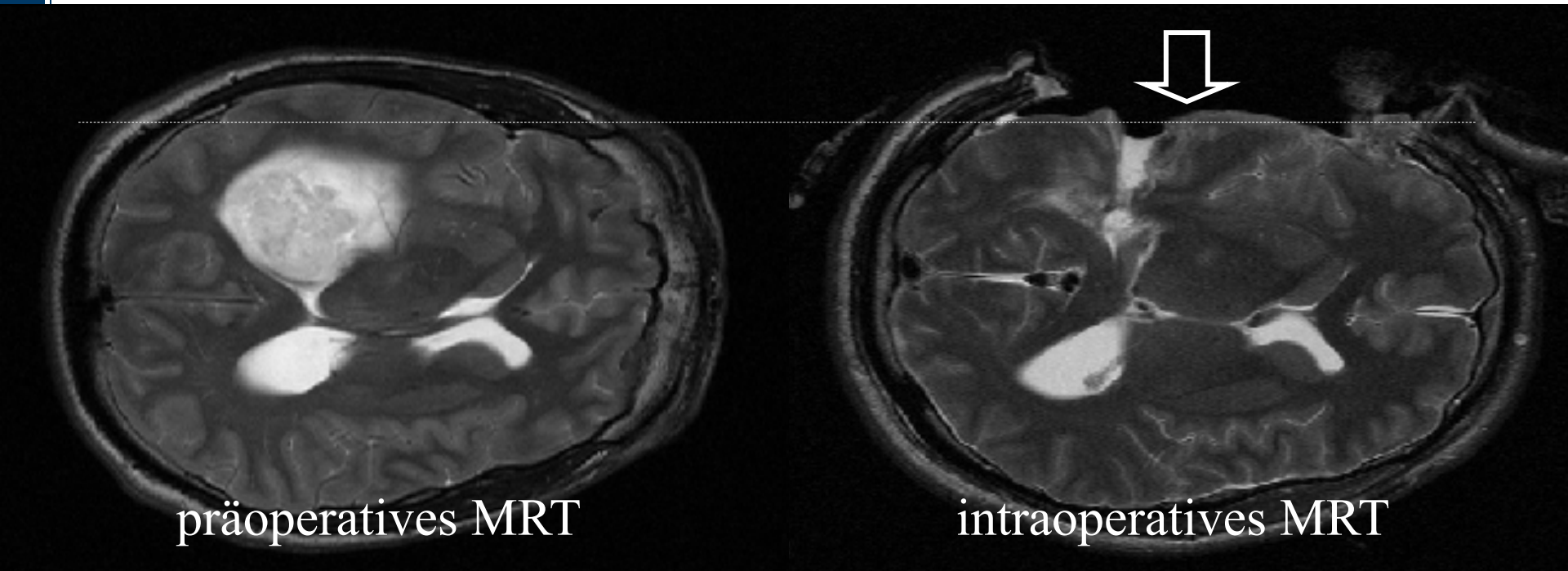
$$\sum_{i,j} |\text{Bild2}_{i,j} - \text{Transform}(\text{Bild1}, \vec{t}, \phi)_{i,j}|^2$$

■ 2D-Fusion

- Ansatz
 - Datensatz 1 im Hintergrund
 - Datensatz 2 in magischer Linse
 - kommunizierend: Fenster + Cursor
- Eingeschränkte 3D-Navigation und räumliche Orientierung
- Übergänge von Grauwerten gut erkennbar



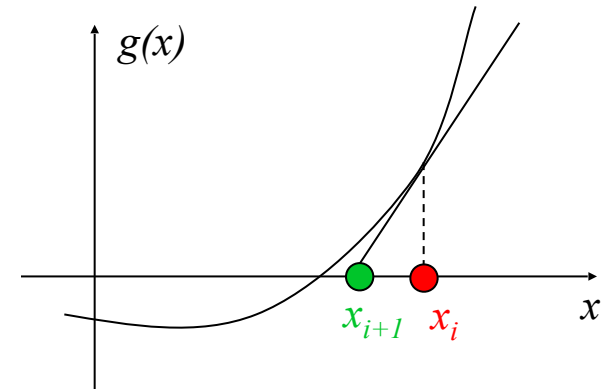
- Intraoperative Resektionskontrolle
 - ▶ Intraoperative Deformation des Gehirns („Brain Shift“)
 - ▶ Update der Navigation mit intraoperativen Bilddaten



Erinnerung: Newton-Verfahren zur Nullstellenbestimmung

- Newtonverfahren für 1D :

$$x_{i+1} = x_i - \frac{g(x_i)}{g'(x_i)}$$



- Newtonverfahren für nD

$$x_{i+1} = x_i - J_G(x_i)^{-1} G(x_i)$$

dabei ist J_G die Jacobimatrix der Funktion $G : \mathbb{R}^n \rightarrow \mathbb{R}^n$

- Zur Optimierung von $F : \mathbb{R}^n \rightarrow \mathbb{R}$ muss man die Nullstellen von

$$G(x) = \mathbf{grad}(F)(x) \quad (G : \mathbb{R}^n \rightarrow \mathbb{R}^n !!!)$$

bestimmen.

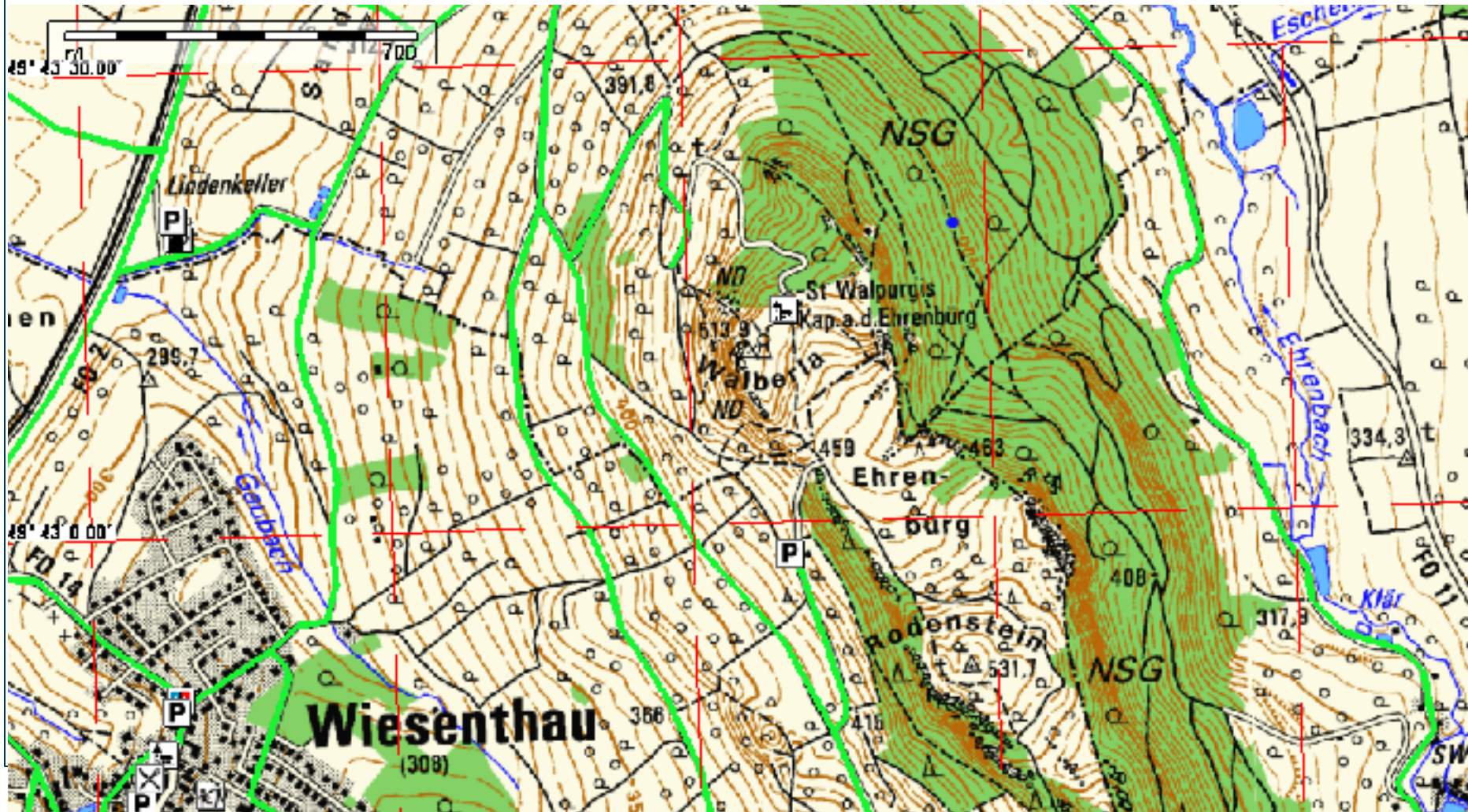
- zB mit dem Newton-Verfahren:

$$x_{i+1} = x_i - [\mathbf{H}_F(x_i)]^{-1} \mathbf{grad}(F)(x_i)$$

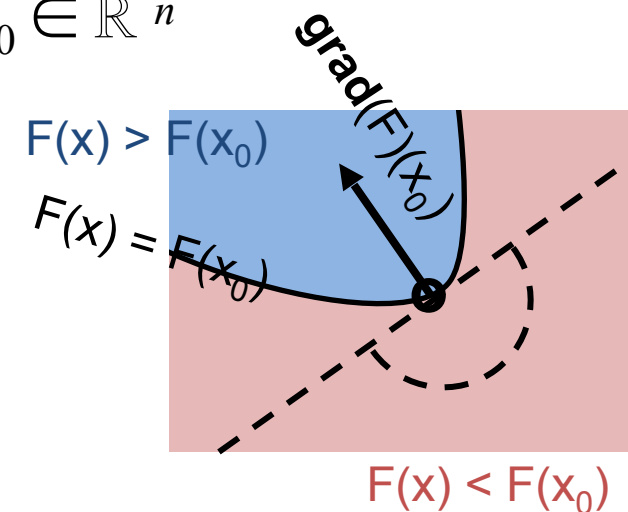
dabei ist $\mathbf{H}_F(x_i) = J_{\mathbf{grad}(F)} = \text{Hesse-Matrix}(F) =$

$$\begin{bmatrix} \frac{\partial^2 F}{\partial x_1 \partial x_1} & \dots & \frac{\partial^2 F}{\partial x_n \partial x_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 F}{\partial x_1 \partial x_n} & \dots & \frac{\partial^2 F}{\partial x_n \partial x_n} \end{bmatrix}$$

- Probleme
 - ▶ aufwändig für große n
 - ▶ oft ist es schwierig bzw. „teuer“ die $n^2/2$ zweiten Ableitungen zu bestimmen



- Graphische Interpretation: (Gebirge)
 - Gradienten \perp Höhenlinie,
 - weist in Richtung des größten Anstiegs.
- Startwert: $\mathbf{x}_0 \in \mathbb{R}^n$ und Suchrichtung $\mathbf{s}_0 \in \mathbb{R}^n$
- Iteration: $\mathbf{x}_{i+1} = \mathbf{x}_i + t_i \mathbf{s}_i$
 - $t_i > 0$ **Schrittweite**,
 - \mathbf{s}_i **Suchrichtung** (eventuell $\|\mathbf{s}_i\| = 1$).
- Falls $\mathbf{s}_i \circ \mathbf{grad}(F)(\mathbf{x}_i) < 0$ und $t_i > 0$ spricht man von einem **Abstiegsverfahren**



- Startwert: $\mathbf{x}_0 \in \mathbb{R}^n$ und Suchrichtung $\mathbf{s}_0 \in \mathbb{R}^n$
- Iteration: $\mathbf{x}_{i+1} = \mathbf{x}_i + t_i \mathbf{s}_i$
 - ▶ $t_i > 0$ **Schrittweite**,
 - ▶ \mathbf{s}_i **Suchrichtung** (eventuell $\|\mathbf{s}_i\| = 1$).
- **Abstiegsverfahren** falls $\mathbf{s}_i \circ \mathbf{grad}(F)(\mathbf{x}_i) < 0$ und $t_i > 0$
- **Fragen**
 - ▶ In welche Richtung? (Wahl der Suchrichtung \mathbf{s}_i)
 - ▶ Wie weit? (Wahl der Schrittweite t_i)

- Die optimale Schrittweite kann man durch (näherungsweise) Lösen des folgenden eindimensionalen Minimierungsproblems bestimmen:

$$t_i = \mathbf{argmin} \{ F(x_i + ts_i) : t > 0 \}$$

d.h. löse 1D-Minimierung für $f_i(t) = F(x_i + ts_i)$

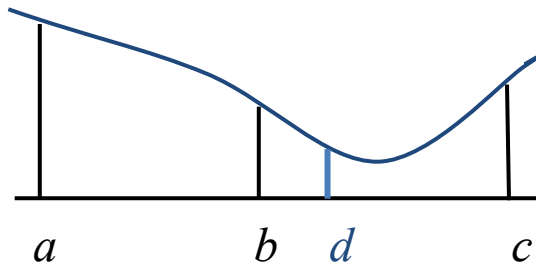
- z.B. mit 1D-Newton (sh. oben), benötigt **grad**(F) und \mathbf{H}_F

oder

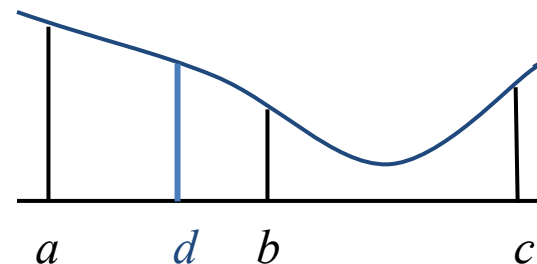
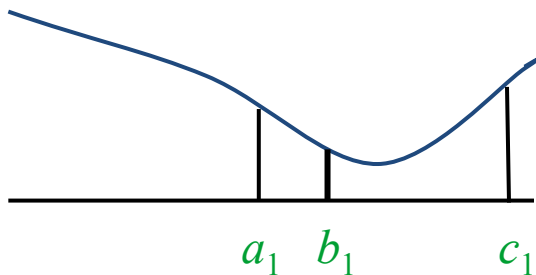
- Approximiere durch ein „Bisektions“-ähnliches Verfahren: sog. **Goldenes-Schnitt-Verfahren**: für 1D-Funktion

$$f(t) = F(x_i + ts_i)$$

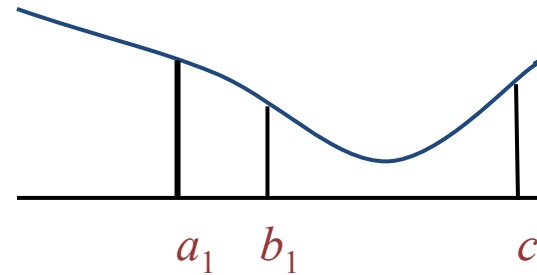
- ★ Starte mit 3 Werten $a < b < c$ für die $f(a) > f(b) < f(c)$ gilt;
- ★ Wähle neuen Punkt $d : a < d < c$;
- ★ Ist $f(d) < f(b)$ bildet man das neue Tripel mit d als mittleren Punkt **andernfalls** mit b als mittleren Punkt;



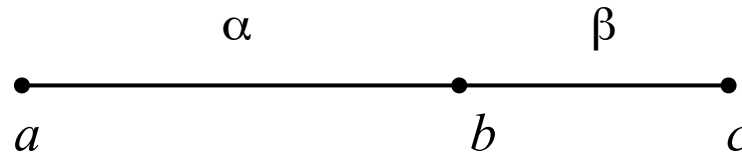
$$f(d) < f(b)$$



$$f(d) > f(b)$$



- Strategie zur Wahl von d :
Teile das größere Intervall gem. **Goldenem Schnitt**
- Definition: *Drei Zahlen $a < b < c$ sind gemäß Goldenem Schnitt positioniert, falls die Teilstrecken $\alpha = b-a$ und $\beta = c-b$ das Teilungsverhältnis $\beta : \alpha = \alpha : (\alpha + \beta)$ haben*



- $$\alpha = \frac{\sqrt{5}+1}{2} \beta \approx 1.618 \cdot \beta \quad \text{oder} \quad \alpha = \frac{\sqrt{5}-1}{2} (\alpha + \beta) \approx 0.618 \cdot (\alpha + \beta)$$



- Strategie zur Wahl von d :

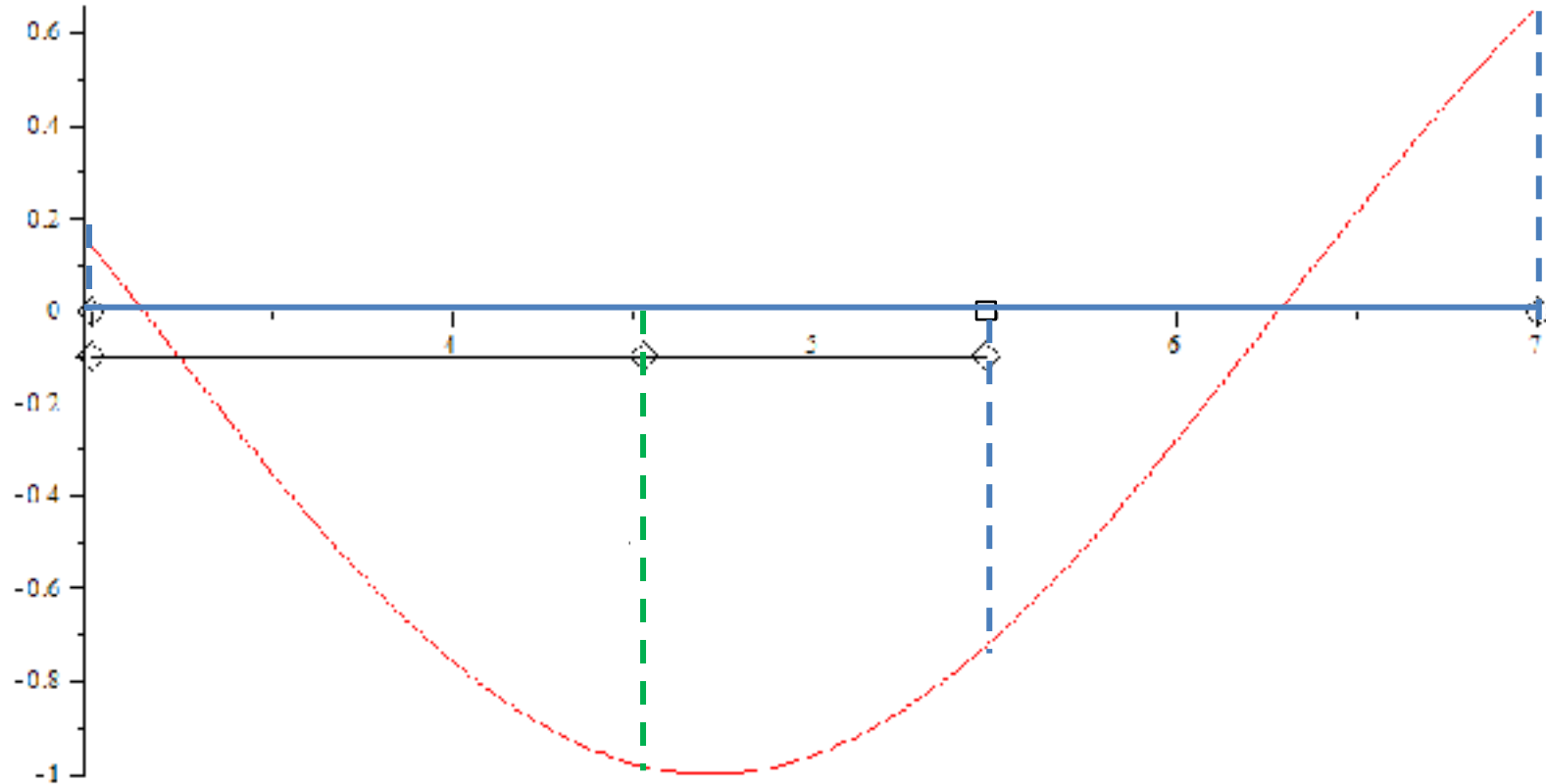
Teile das größere Intervall gem. Goldenen Schnitt so dass das größere Teil außen liegt



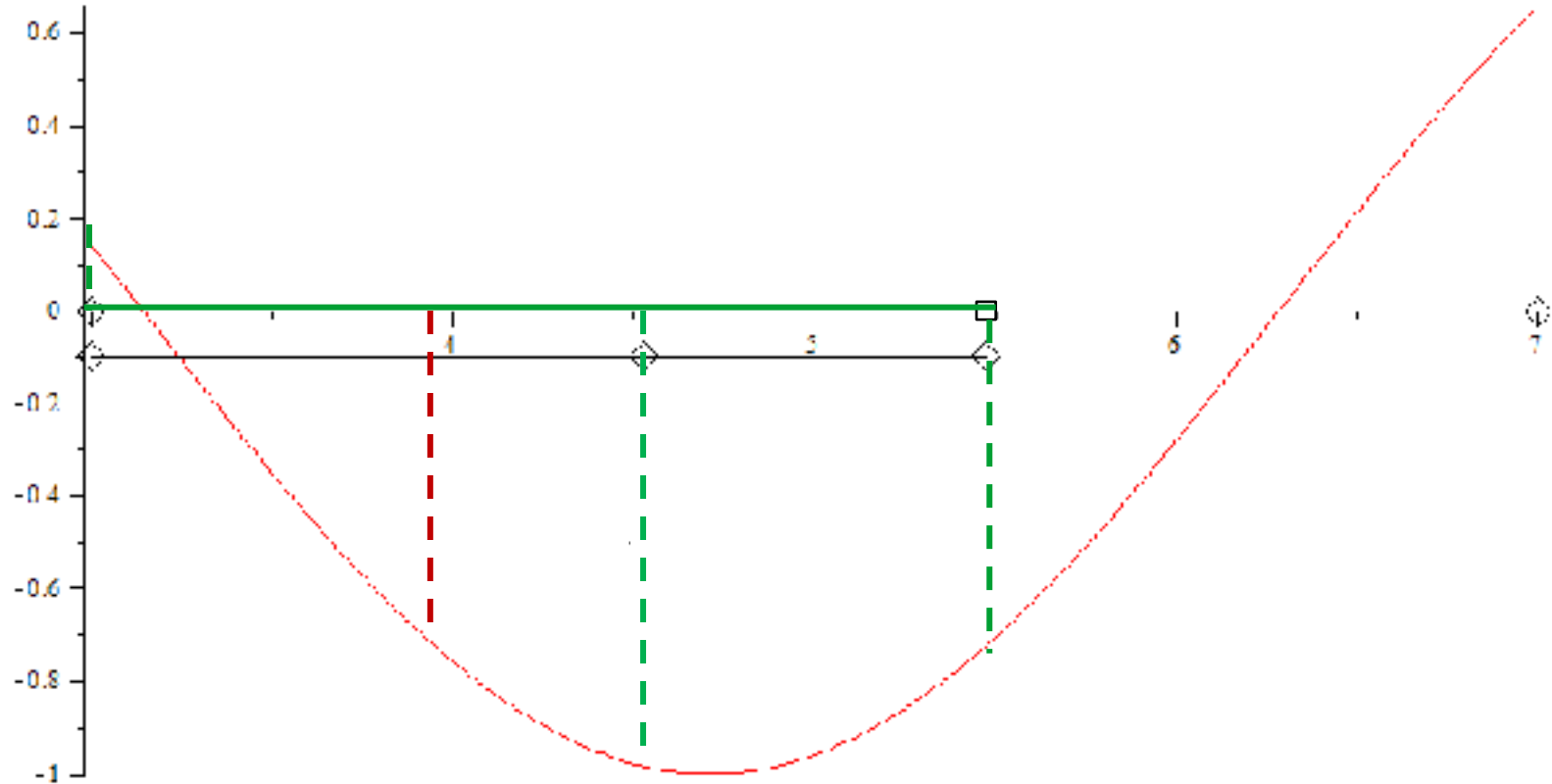
- Falls a, b, c gemäß Goldenem Schnitt positioniert sind und $d = c - (b - a)$, dann erfüllen sowohl a, d, b als auch d, b, c den Goldenen Schnitt.



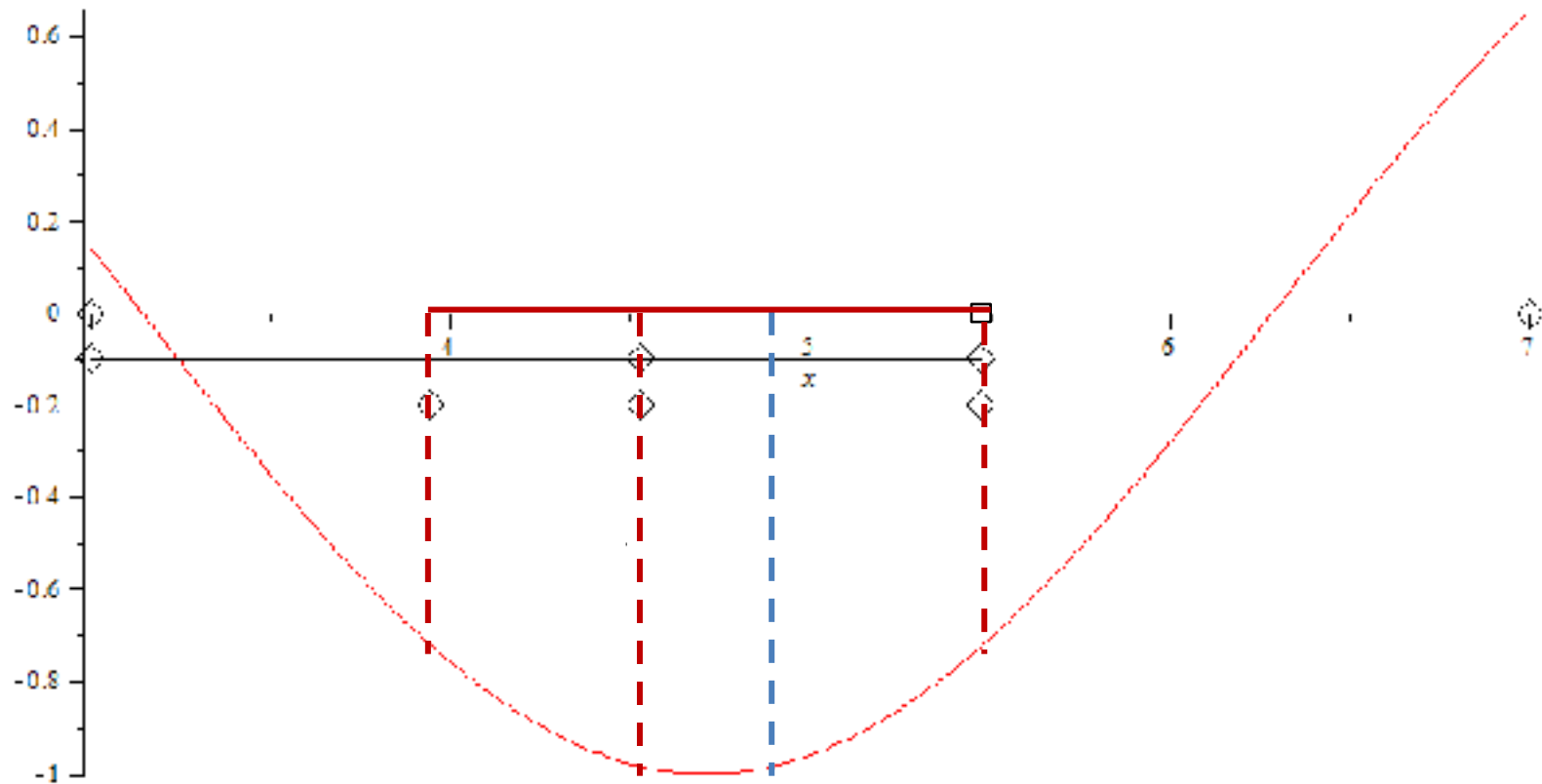
- Beispiel



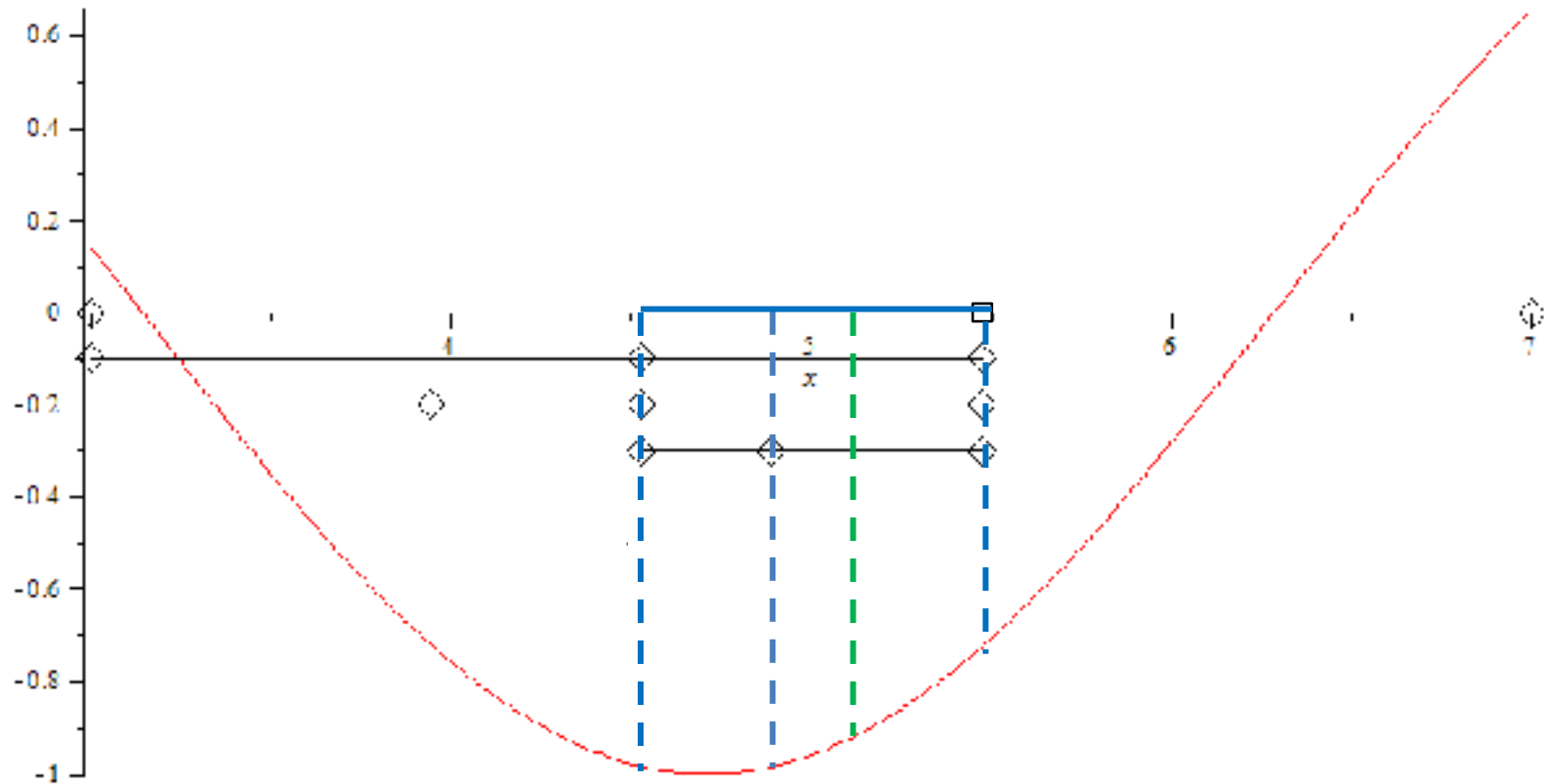
- Beispiel



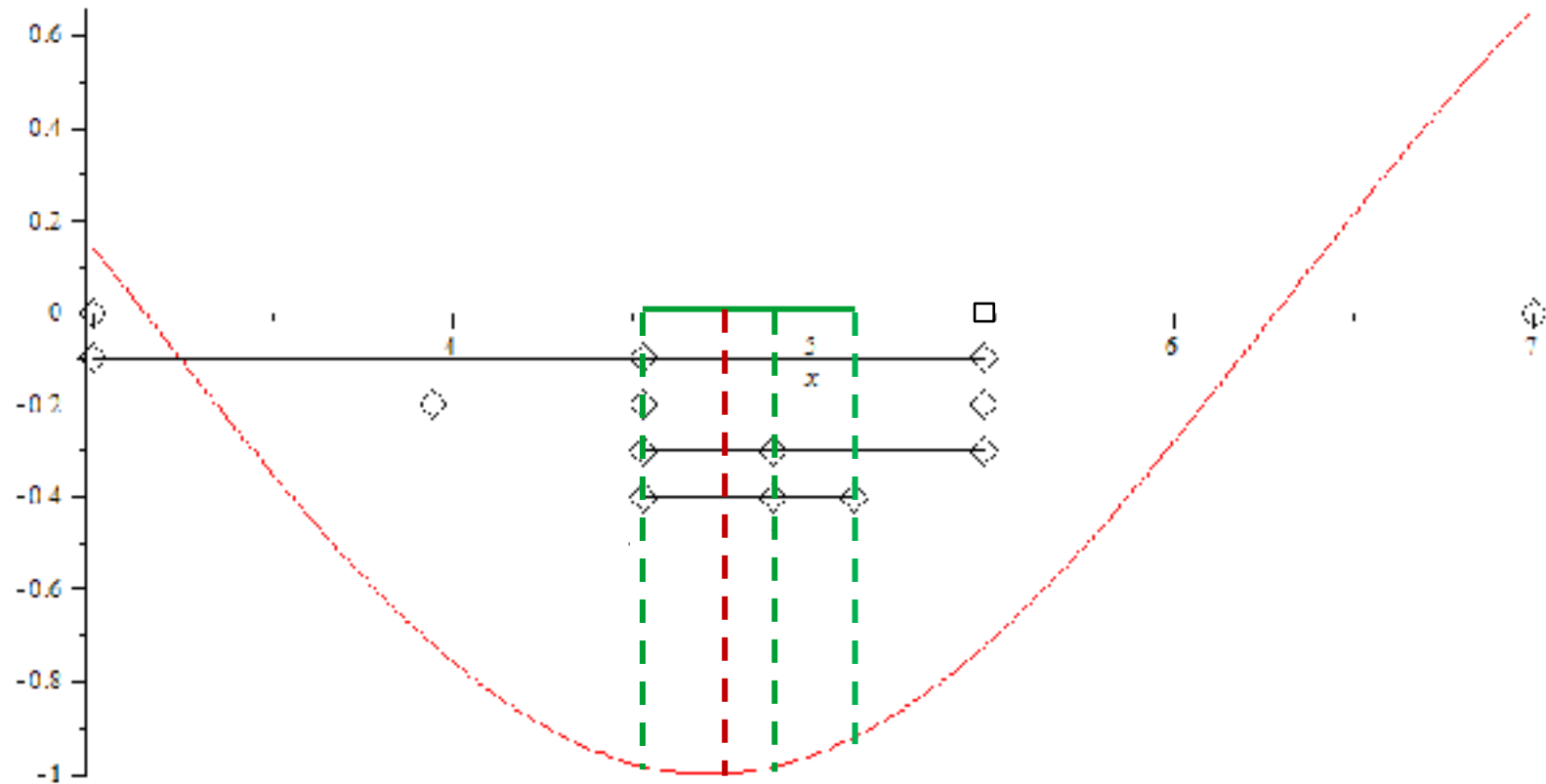
- Beispiel



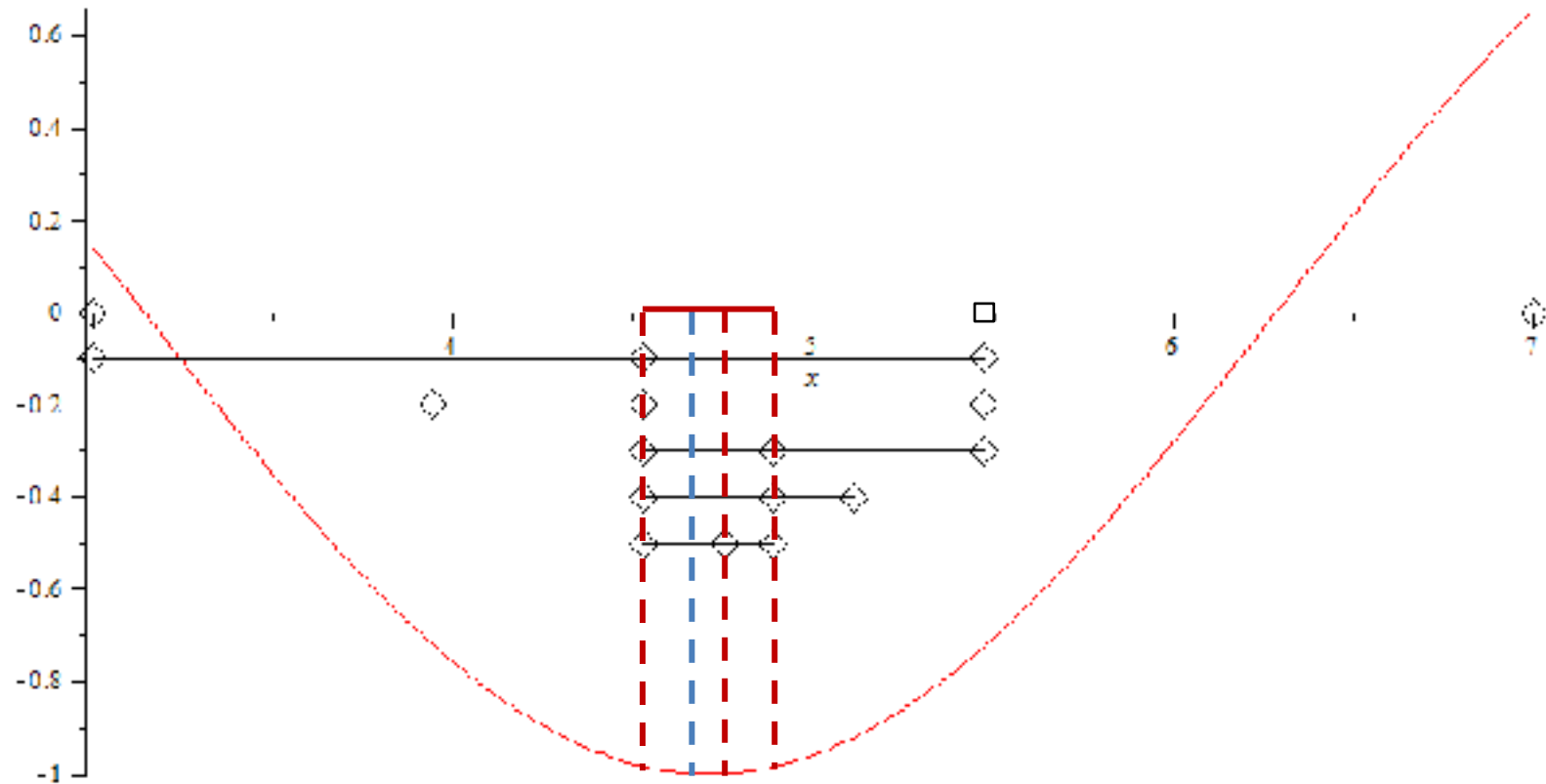
- Beispiel



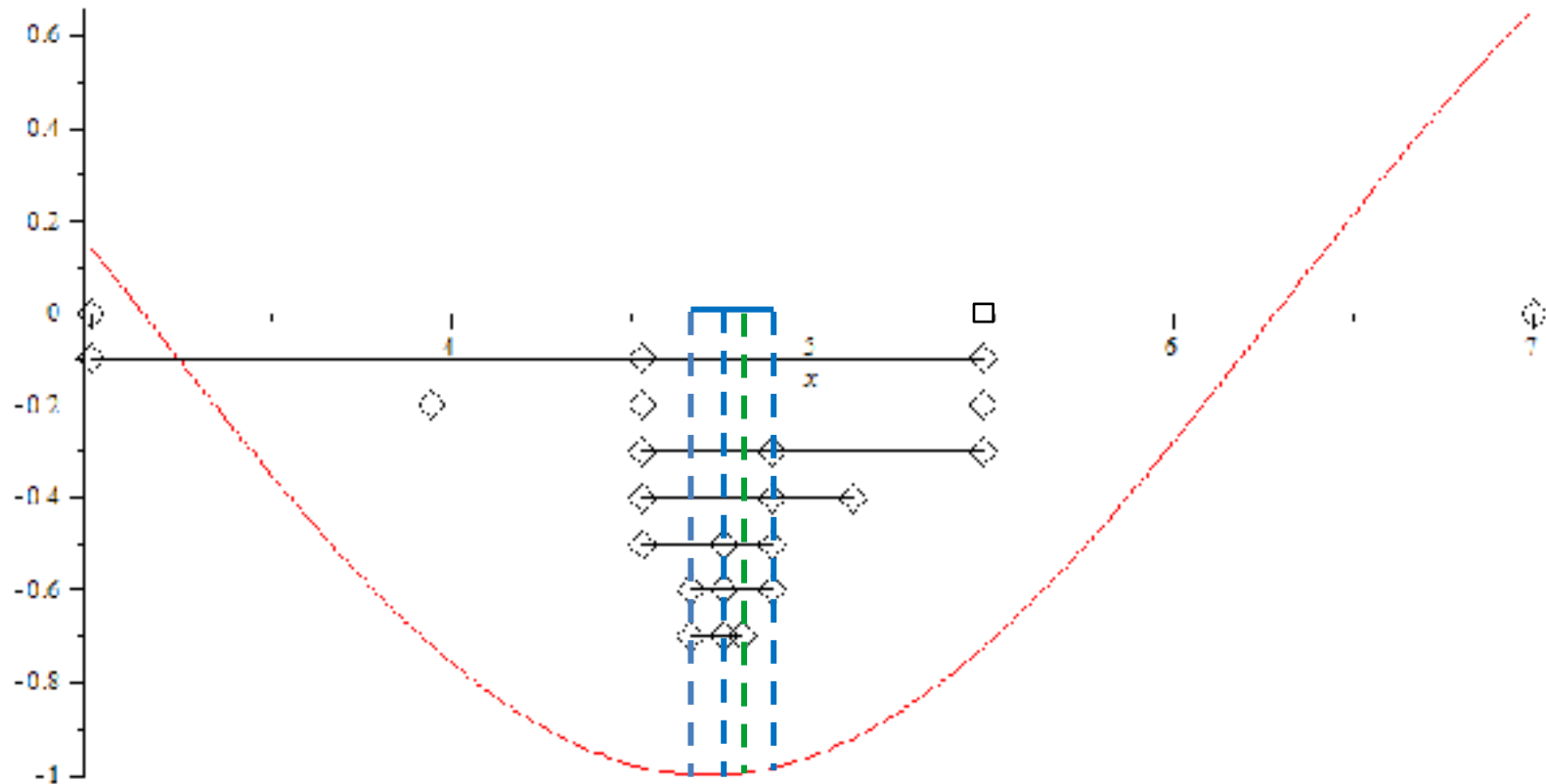
- Beispiel



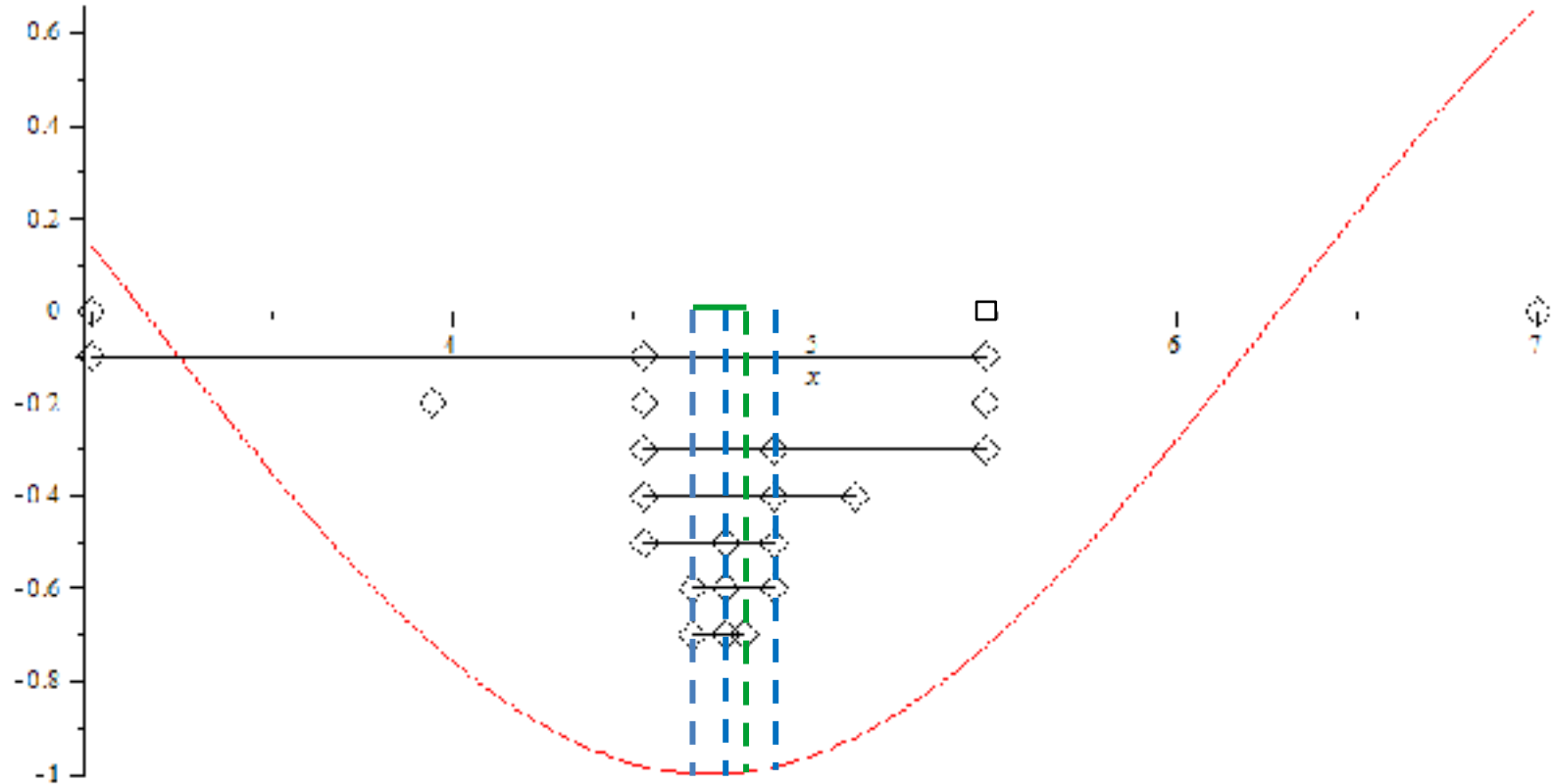
- Beispiel



- Beispiel



- Beispiel



- Strategien zum Auffinden der Suchrichtung
 - ▶ **Newton-Verfahren:** $s_i = - [\mathbf{H}_F(x_i)]^{-1} \mathbf{grad}(F)(x_i)$, $t_i = 1$
 - ▶ Gradientenverfahren (**steepest descent**)
 $s_i = - \mathbf{grad}(F)(x_i)$, $t_i = ?$
 - ▶ Konjugierte Gradienten → **cg-Verfahren**
 - ▶ **Quasi-Newton-Verfahren (s.u.)**

- Strategie zur Schrittweitenwahl
 - ▶ Bei Newton $t_i = 1$, besser gedämpft: $t_i < 1$
 - ▶ sonst 1D- Minimierer „*LineSearch*“
 (z.B. Goldener Schnitt)

- Ist das Newton-Verfahren ein Abstiegsverfahren?

bzw.

- Wann ist das Newton-Verfahren ein Abstiegsverfahren

$$g \circ s = -g^T H^{-1} g = -g^T H^{-1} H H^{-1} g = -(H^{-1} g)^T H (H^{-1} g) < 0$$

sofern $H = \mathbf{H}_F(x_i)$ positiv definit ist
(eine sinnvolle Bedingung)

- Beispiel: Finde Minimum von

$$F(x, y) = x^2 y^2 + x^2 + 2xy + 4y^2 - 3x - 4y + 6$$

- Startwert $(x_0, y_0) = (2, 0)$

- Gradient und Hesseform:

$$\text{grad}(F)(x, y) = \begin{bmatrix} 2xy^2 + 2x + 2y - 3 \\ 2x^2 y + 2x + 8y - 4 \end{bmatrix}$$

$$\mathbf{H}_F(x, y) = \begin{bmatrix} 2y^2 + 2 & 4xy + 2 \\ 4xy + 2 & 2x^2 + 8 \end{bmatrix}$$

- Beispiel: $F(x, y) = x^2 y^2 + x^2 + 2xy + 4y^2 - 3x - 4y + 6$
 - ▶ Startwert: $x_0 = 2.0, y_0 = 0.0$
 - ▶ Exakte Lösung: $x^* = 1.379694469, y^* = 0.1050731871$

i	Newton	Gradient (t=0.25)	Gradient(t=0.125)	Gradient mit LineSearch
0	2.0, 0.0	2.0, 0.0	2.0, 0.0	2.0, 0.0
1	1.42857, 0.07142	1.75000, 0.0	1.87500, 0.0	1.50000, 0.0
2	1.38258, 0.10376	1.62500 0.12500	1.78125, 0.03125	1.50007, 0.07993
3	1.37969, 0.10507	1.48730 -0.10254	1.70269, 0.02990	1.41025, 0.08026
4	1.37969, 0.10507	1.53710 0.47230	1.64416, 0.05266	1.41032, 0.09842
9		-260.04, -276.19	1.47657, 0.08143	1.38021, 0.10465
10		9.92 10 ⁶ , 9.34 10 ⁶	1.45962, 0.08647	1.38021, 0.10496

- cg steht für *conjugate gradients*
- Ein **quadratisches Funktional** hat folgende Form

$$F(\mathbf{x}) = \mathbf{x}^T \mathbf{A} \mathbf{x} + 2\mathbf{b}^T \mathbf{x} + \gamma$$

dabei ist \mathbf{A} eine **positiv definite** $n \times n$ - Matrix,
 \mathbf{b}, \mathbf{x} sind n -Vektoren, γ eine Konstante

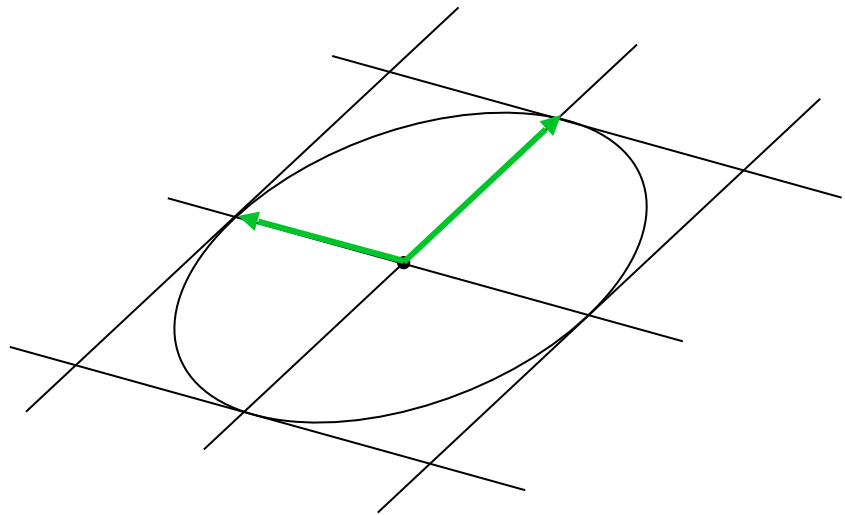
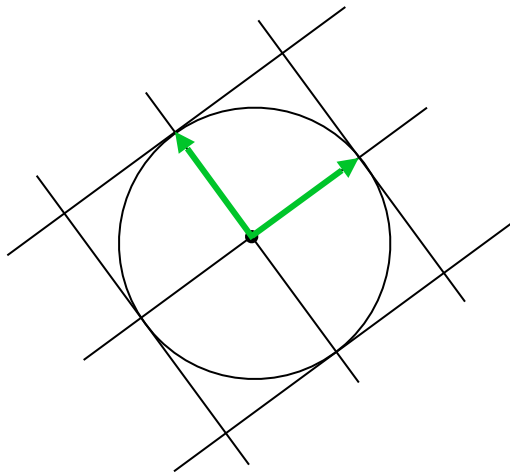
- Was sind konjugierte Richtungen?

\mathbf{u} und \mathbf{v} sind **\mathbf{A} -konjugiert** (oder **\mathbf{A} -orthogonal**) falls

$$\mathbf{u}^T \mathbf{A} \mathbf{v} = 0$$

- \mathbf{u} und \mathbf{v} sind \mathbf{A} -konjugiert (oder \mathbf{A} -orthogonal) falls
 $\mathbf{u}^T \mathbf{A} \mathbf{v} = 0$ (auf die Länge kommt es nicht an!)
oder $\mathbf{u} \perp \mathbf{A} \mathbf{v}$ (\mathbf{u} steht senkrecht auf $\mathbf{A} \mathbf{v}$)
- Geometrische Veranschaulichung (für $n=2$) am Kegelschnitt

$$\{ \mathbf{x} : \mathbf{x}^T \mathbf{A} \mathbf{x} = 1 \}$$



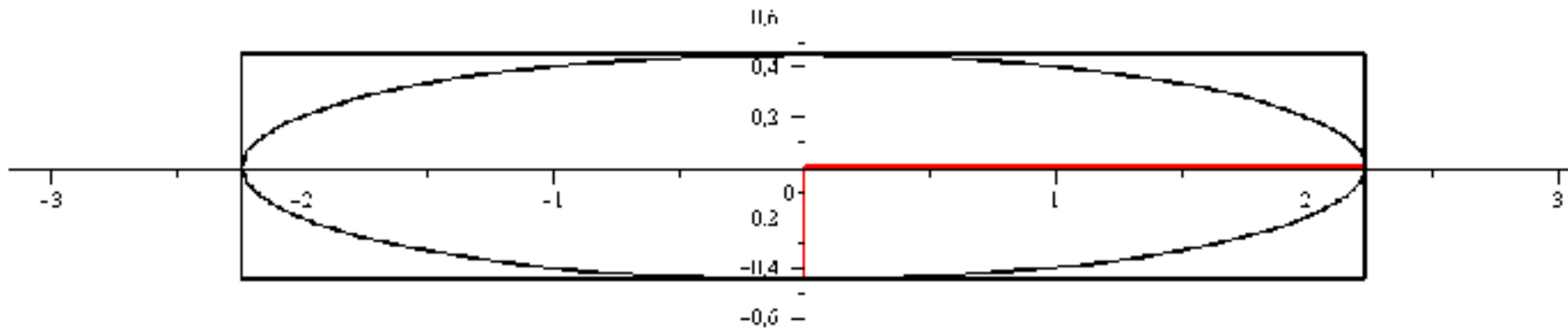
- 2D : $A = \begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix}$, $u_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$, $v_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$, $w_1 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

- zu u_1, v_1 , bzw. w_1 , A -konjugierte Vektoren:

$$u_2 = \begin{bmatrix} -1 \\ 2 \end{bmatrix}, v_2 = \begin{bmatrix} 2 \\ -3 \end{bmatrix}, w_2 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

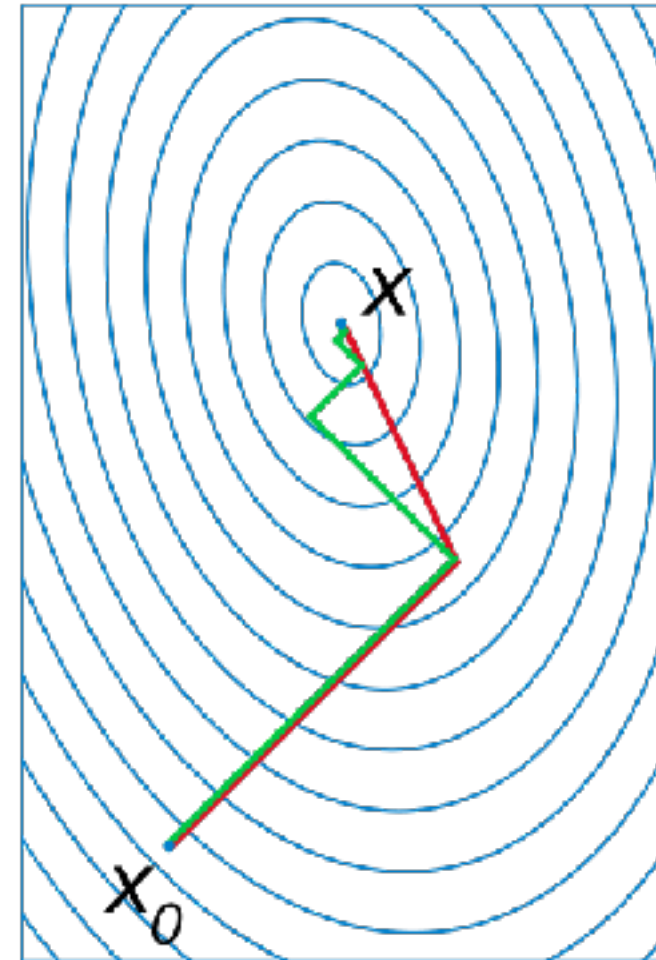
- 3D : $A = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 2 & 0 \\ 1 & 0 & 3 \end{bmatrix}$, $u_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$, $u_2 = \begin{bmatrix} -1 \\ 1 \\ 0 \end{bmatrix}$, $u_3 = \begin{bmatrix} -5 \\ -1 \\ 3 \end{bmatrix}$

- paarweise A -konjugiert u_1, u_2, u_3 :



Anmerkung: Konjugierte Richtungen sind affin invariant, Orthogonalität gilt nicht!

- **Theorem:**
Wählt man als Suchrichtungen (s_i) eine Folge paarweise konjugierter Richtungen und dazu optimale Schrittweiten t_i , dann ist man nach (spätestens) n Schritten im Minimum.
- **Anmerkung:**
Es gibt nur n paarweise konjugierte Richtungen



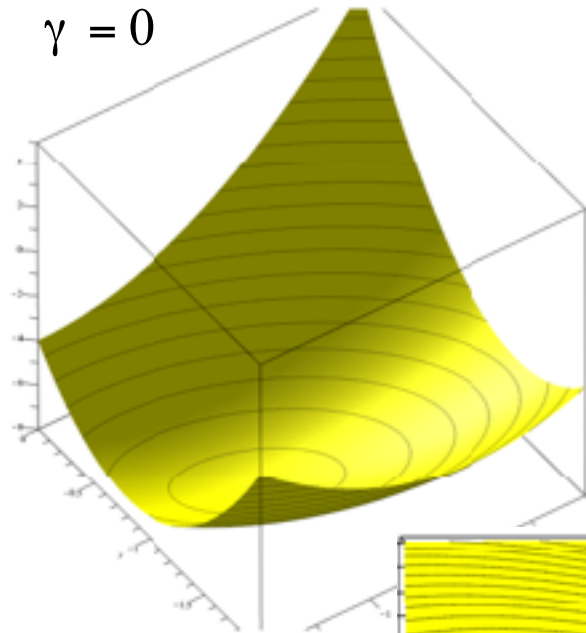
grün: Gradienten rot: konj. Grad.

- Konkretes Beispiel: $Q(x,y) = x^2 + 2xy + 5y^2 + 4x + 12y$

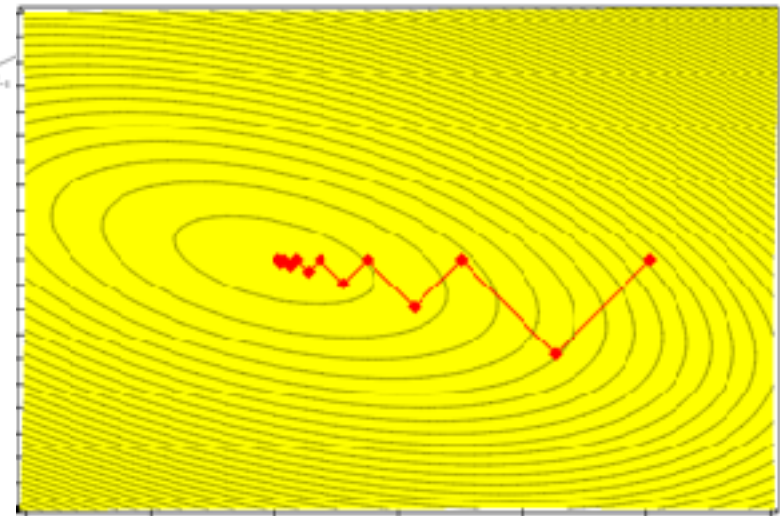
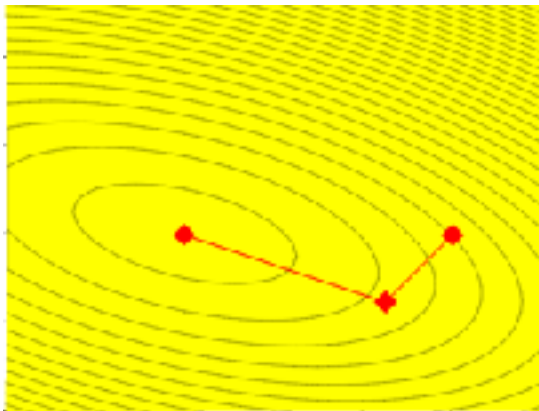
$$A = \begin{bmatrix} 1 & 1 \\ 1 & 5 \end{bmatrix}, \quad b = \begin{bmatrix} 2 \\ 6 \end{bmatrix}, \quad \gamma = 0$$

$$[x_0, y_0] = [0.5, 1.0]$$

i	x_i	y_i
0	0.5	-1.0
1	0.125	-1.375
2	-1.0	-1.0



i	x_i	y_i
0	0.5	-1.0
1	0.125	-1.375
2	-0.25	-1.0
3	-0.4375	-1.1875
4	-0.625	-1.0



- Initialisierung: Wähle $\mathbf{x}^{(0)}$ beliebig und berechne
 - $\mathbf{g}^{(0)} = \mathbf{b} + \mathbf{A}\mathbf{x}^{(0)}$ [$\mathbf{g} = 1/2 \mathbf{grad}(F)(\mathbf{x})$]
 - $\mathbf{s}^{(0)} = -\mathbf{g}^{(0)}$ [\mathbf{s} Suchrichtung]
 - $\alpha_0 = \frac{\mathbf{g}^{(0)} \circ \mathbf{g}^{(0)}}{\mathbf{s}^{(0)} \circ \mathbf{A}\mathbf{s}^{(0)}}$ [α optimale Schrittweite]
- for $k = 0, 1, 2, \dots, n-1$:
 - bestimme von $\mathbf{x}^{(k)}$ in Richtung $\mathbf{s}^{(k)}$ das Minimum
 $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_0 \cdot \mathbf{s}^{(k)}$
 - aktualisiere den Gradienten (=Residuum)
 - aktualisiere die Suchrichtung, so dass die neue Suchrichtung \mathbf{A} -konjugiert zu allen bisherigen ist
- Stop falls $\|\mathbf{g}^{(k+1)}\| \approx 0$ (theoretisch nach n Schritten)

- Initialisierung: Wähle $\mathbf{x}^{(0)}$ beliebig und berechne

$$\triangleright \mathbf{g}^{(0)} = \mathbf{b} + \mathbf{A}\mathbf{x}^{(0)} \quad [\mathbf{g} = 1/2 \mathbf{grad}(F)(\mathbf{x})]$$

$$\triangleright \mathbf{s}^{(0)} = -\mathbf{g}^{(0)} \quad [\mathbf{s} \text{ Suchrichtung}]$$

$$\triangleright \alpha_0 = \frac{\mathbf{g}^{(0)} \circ \mathbf{g}^{(0)}}{\mathbf{s}^{(0)} \circ \mathbf{A}\mathbf{s}^{(0)}} \quad [\alpha \text{ optimale Schrittweite}]$$

- for $k = 0, 1, 2, \dots, n-1$:

$$\bullet \mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{s}^{(k)}$$

$$\bullet \mathbf{g}^{(k+1)} = \mathbf{g}^{(k)} + \alpha_k \mathbf{A}\mathbf{s}^{(k)}$$

$$\bullet \beta_k = \frac{\mathbf{g}^{(k+1)} \circ \mathbf{g}^{(k+1)}}{\mathbf{g}^{(k)} \circ \mathbf{g}^{(k)}}$$

$$\bullet \mathbf{s}^{(k+1)} = -\mathbf{g}^{(k+1)} + \beta_k \mathbf{s}^{(k)}$$

$$\bullet \alpha_{k+1} = \frac{\mathbf{g}^{(k+1)} \circ \mathbf{g}^{(k+1)}}{\mathbf{s}^{(k+1)} \circ \mathbf{A}\mathbf{s}^{(k+1)}}$$

- Stop falls $\|\mathbf{g}^{(k+1)}\| < \varepsilon$ (theoretisch nach $\leq n$ Schritten)

Beachte:

Die so generierten
Suchrichtungen

$\mathbf{s}^{(0)}, \mathbf{s}^{(1)}, \mathbf{s}^{(2)}, \dots$

sind paarweise konjugiert!

- cg-Verfahren ist – theoretisch - ein exaktes Verfahren:
nach n Schritten :

$$\mathbf{x}^{(n)} = \mathbf{argmin} (\mathbf{x}^T \mathbf{A} \mathbf{x} + 2\mathbf{b}^T \mathbf{x} + \gamma)$$

- wird oft verwendet wie ein iteratives Verfahren
 - ▶ wg Rundungsfehler evt. mehr als n Iterationen
 - ▶ bei guter Konvergenz oder wenn grobe Näherung genügt oft weniger als n Iterationen!
- Aufwand: Im wesentlichen die Berechnung von $\mathbf{A} \mathbf{s}^{(k)}$

- Fehlerabschätzung: Es gilt: $\|\mathbf{x}_{k+1} - \mathbf{x}^*\|_A \leq C \cdot \left(\frac{\sqrt{\kappa(\mathbf{A})} - 1}{\sqrt{\kappa(\mathbf{A})} + 1} \right)^k$
wobei $\|\mathbf{x}\|_A^2 = \mathbf{x}^T \mathbf{A} \mathbf{x}$

- cg-Verfahren ist – theoretisch - ein exaktes Verfahren:
nach n Schritten :

$$\mathbf{x}^{(n)} = \mathbf{argmin} (\mathbf{x}^T \mathbf{A} \mathbf{x} + 2\mathbf{b}^T \mathbf{x} + \gamma)$$

- Beschleunigung durch **Vorkonditionierer**
Substituiere \mathbf{A} durch $\mathbf{A}_C = \mathbf{C}^T \mathbf{A} \mathbf{C}$ wobei
 $\mathbf{C}^T \mathbf{C} \approx \mathbf{A}^{-1}$

- cg-Verfahren liefert auch ein weiteres Verfahren zu Lösung linearer Gleichungssysteme mit **positiv definiter Koeffizientenmatrix**
- Denn:
Die Lösung von $\operatorname{argmin} \{ \mathbf{x}^T \mathbf{A} \mathbf{x} + 2 \mathbf{b}^T \mathbf{x} + \gamma \}$ ist die eindeutige Lösung des lin. Gleichungssystems

$$\mathbf{A} \mathbf{x} + \mathbf{b} = 0 \quad (\text{falls } \mathbf{A} \text{ positiv definit, } \mathbf{b} \text{ beliebig})$$

- 1. Beweis: $\operatorname{grad}(Q) = 2(\mathbf{A} \mathbf{x} + \mathbf{b})$, $\mathbf{H}_Q = 2\mathbf{A}$.
- 2. Beweis: $0 \leq (\mathbf{A} \mathbf{x} + \mathbf{b})^T \mathbf{A}^{-1} (\mathbf{A} \mathbf{x} + \mathbf{b}) = Q(\mathbf{x}) + \text{const}$

- Initialisierung: Wähle $x^{(0)}$ beliebig und berechne
 - $g^{(0)} = b + Ax^{(0)}$
 - $s^{(0)} = -g^{(0)}$
 - $\alpha_0 = \frac{g^{(0)} \circ g^{(0)}}{s^{(0)} \circ As^{(0)}}$
- for $k = 0, 1, 2, \dots, n-1$:
 - $x^{(k+1)} = x^{(k)} + \alpha_k s^{(k)}$
 - $g^{(k+1)} = g^{(k)} + \alpha_k As^{(k)}$
 - $\beta_k = \frac{g^{(k+1)} \circ g^{(k+1)}}{g^{(k)} \circ g^{(k)}}$
 - $s^{(k+1)} = -g^{(k+1)} + \beta_k s^{(k)}$
 - $\alpha_{k+1} = \frac{g^{(k+1)} \circ g^{(k+1)}}{s^{(k+1)} \circ As^{(k+1)}}$
- Stop falls $\|g^{(k+1)}\| < \varepsilon$ (theoretisch nach $\leq n$ Schritten)

- Das cg-Verfahren als Gleichungslöser steht zwischen den direkten Verfahren (z.B. Cholesky) und den iterativen Verfahren (zB Gauss-Seidel, SOR)
- Es vereint Vorteile / Nachteile direkter und iterativer Verfahren
 - ▶ theoretisch hat man nach n Schritten die exakte Lösung (aber Rundungsfehler!)
 - ▶ Rundungsfehler verstärken sich nicht
 - ▶ man nähert sich mit jedem Schritt der Lösung an (erlaubt vorzeitigen Abbruch!)
- Komplexität: für voll besetzte Matrizen: $O(n^3)$
 dünnbesetzte Matrizen: $O(n^2)$

- Initialisierung: Wähle $\mathbf{x}^{(0)}$ beliebig und berechne
 - $\mathbf{g}^{(0)} = \mathbf{b} + \mathbf{A}\mathbf{x}^{(0)}$ [$\mathbf{g} = 1/2 \text{ grad}(F)(\mathbf{x})$]
 - $\mathbf{s}^{(0)} = -\mathbf{g}^{(0)}$ [\mathbf{s} Suchrichtung]
 - $\alpha_0 = \frac{\mathbf{g}^{(0)} \circ \mathbf{g}^{(0)}}{\mathbf{s}^{(0)} \circ \mathbf{A}\mathbf{s}^{(0)}}$ [α optimale Schrittweite]
- for $k = 0, 1, 2, \dots, n-1$:
 - $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{s}^{(k)}$
 - $\mathbf{g}^{(k+1)} = \mathbf{g}^{(k)} + \alpha_k \mathbf{A}\mathbf{s}^{(k)}$
 - $\beta_k = \frac{\mathbf{g}^{(k+1)} \circ \mathbf{g}^{(k+1)}}{\mathbf{g}^{(k)} \circ \mathbf{g}^{(k)}}$
 - $\mathbf{s}^{(k+1)} = -\mathbf{g}^{(k+1)} + \beta_k \mathbf{s}^{(k)}$
 - $\alpha_{k+1} = \frac{\mathbf{g}^{(k+1)} \circ \mathbf{g}^{(k+1)}}{\mathbf{s}^{(k+1)} \circ \mathbf{A}\mathbf{s}^{(k+1)}}$
- Stop falls $\|\mathbf{g}^{(k+1)}\| < \varepsilon$ (theoretisch nach $\leq n$ Schritten)

- Initialisierung: Wähle $\mathbf{x}^{(0)}$ beliebig und berechne

- $\mathbf{g}^{(0)} = \mathbf{grad}(F)(\mathbf{x}^{(0)})$

[$\mathbf{g} = 1/2 \mathbf{grad}(F)(\mathbf{x})$]

- $\mathbf{s}^{(0)} = -\mathbf{g}^{(0)}$

[\mathbf{s} Suchrichtung]

- $\alpha_0 = \operatorname{argmin}_t \{ F(\mathbf{x}^{(0)} + t\mathbf{s}^{(0)}) \}$ [α optimale Schrittweite]

- for $k = 0, 1, 2, \dots, n-1$:

- $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{s}^{(k)}$

- $\mathbf{g}^{(k+1)} = \mathbf{grad}(F)(\mathbf{x}^{(k+1)})$

- $\beta_k = \frac{\mathbf{g}^{(k+1)} \circ \mathbf{g}^{(k+1)}}{\mathbf{g}^{(k)} \circ \mathbf{g}^{(k)}}$

- $\mathbf{s}^{(k+1)} = -\mathbf{g}^{(k+1)} + \beta_k \mathbf{s}^{(k)}$

- $\alpha_{k+1} = \operatorname{argmin}_t \{ F(\mathbf{x}^{(k+1)} + t\mathbf{s}^{(k+1)}) \}$

- Stop falls $\|\mathbf{g}^{(k+1)}\| < \varepsilon$ ~~(theoretisch nach $\leq n$ Schritten)~~

Verfahren von Fletcher/Reeves

- Initialisierung: Wähle $\mathbf{x}^{(0)}$ beliebig und berechne
 - $\mathbf{g}^{(0)} = \mathbf{grad}(F)(\mathbf{x}^{(0)})$ [$\mathbf{g} = 1/2 \mathbf{grad}(F)(\mathbf{x})$]
 - $\mathbf{s}^{(0)} = -\mathbf{g}^{(0)}$ [\mathbf{s} Suchrichtung]
 - $\alpha_0 = \underset{t}{\operatorname{argmin}} \{ F(\mathbf{x}^{(0)} + t\mathbf{s}^{(0)}) \}$ [α optimale Schrittweite]
- for $k = 0, 1, 2, \dots, n-1$:
 - $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{s}^{(k)}$
 - $\mathbf{g}^{(k+1)} = \mathbf{grad}(F)(\mathbf{x}^{(k+1)})$
 - $\beta_k = \frac{\mathbf{g}^{(k+1)} \circ (\mathbf{g}^{(k+1)} - \mathbf{g}^{(k)})}{\mathbf{g}^{(k)} \circ \mathbf{g}^{(k)}}$ Verfahren von Pollack/Ribière
 - $\mathbf{s}^{(k+1)} = -\mathbf{g}^{(k+1)} + \beta_k \mathbf{s}^{(k)}$
 - $\alpha_{k+1} = \underset{t}{\operatorname{argmin}} \{ F(\mathbf{x}^{(k+1)} + t\mathbf{s}^{(k+1)}) \}$
- Stop falls $\|\mathbf{g}^{(k+1)}\| < \varepsilon$ ~~(theoretisch nach $\leq n$ Schritten)~~

- Beispiel: $F(x, y) = x^2 y^2 + x^2 + 2xy + 4y^2 - 3x - 4y + 6$
 - ▶ Startwert: $x_0 = 2.0, y_0 = 0.0$
 - ▶ Exakte Lösung: $x^* = 1.379694469, y^* = 0.1050731871$

i	Newton	Gradient mit LineSearch	cg-Verfahren Fletcher-Reeves	cg-Verfahren Pollack-Ribière
0	2.0, 0.0	2.0, 0.0	2.0, 0.0	2.0, 0.0
1	1.42857, 0.07142	1.50009, 0.0	1.50009, 0.0	1.50009, 0.0
2	1.38258, 0.10376	1.50007, 0.07993	1.39605, 0.10404	1.39606, 0.10404
3	1.37969, 0.10507	1.41025, 0.08026	1.39242, 0.10087	1.38036, 0.10450
4	1.37969, 0.10507	1.41032, 0.09842	1.38039, 0.10545	1.37969, 0.10507
				1.37969, 0.10507
9		1.38021, 0.10465	1.37969, 0.10507	
10		1.38021, 0.10496	1.37969, 0.10507	

Newton-Verfahren revisited

- Zur Optimierung von F muss man die Nullstellen von $G(x) = \mathbf{grad}(F)(x)$ (mit dem Newton-Verfahren) bestimmen

$$x_{i+1} = x_i - [\mathbf{H}_F(x_i)]^{-1} \mathbf{grad}(F)(x_i)$$

- Vorteil: Wenn konvergent, dann quadratisch (gegen lokales Minimum)
- Nachteil
 - aufwändig für große n
 - oft ist es schwierig bzw. teuer die $n^2/2$ zweiten Ableitungen zu bestimmen
 - Lösung des Systems teuer

$$x_{i+1} = x_i - [\mathbf{H}_F(x_i)]^{-1} \mathbf{grad}(F)(x_i)$$

- Idee: Ersetze die Matrix $[\mathbf{H}_F(x_i)]^{-1}$ durch eine (Folge) positiv definiter Matrizen, \mathbf{H}_k für die

$$\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)} = \mathbf{H}_k [\mathbf{grad}(F)(\mathbf{x}^{(k)}) - \mathbf{grad}(F)(\mathbf{x}^{(k-1)})]$$

- Man nennt diese Bedingung **Quasi-Newton-Bedingung**
- \mathbf{H}_k wird inkrementell und ohne 2. Ableitungen berechnet
- Vorteil: Konvergenz bleibt superlinear

- **DFP-Verfahren** (Davidon/Fletcher/Powell) [$\nabla = \mathbf{grad}$]

$$H_{k+1} = H_k + \frac{(x_{k+1} - x_k)(x_{k+1} - x_k)^t}{(x_{k+1} - x_k)^t (\nabla F_{k+1} - \nabla F_k)} - \frac{[H_k(\nabla F_{k+1} - \nabla F_k)][H_k(\nabla F_{k+1} - \nabla F_k)]^t}{(\nabla F_{k+1} - \nabla F_k)^t H_k(\nabla F_{k+1} - \nabla F_k)}$$

- **BFGS-Verfahren** (Broyden/Fletcher/Goldfarb/Shanno)

$$H_{k+1} = \dots \text{ (as above) } + [(\nabla F_{k+1} - \nabla F_k)^t H_k(\nabla F_{k+1} - \nabla F_k)] u u^t$$

wobei

$$u = \frac{x_{k+1} - x_k}{(x_{k+1} - x_k)^t (\nabla F_{k+1} - \nabla F_k)} - \frac{H_k(\nabla F_{k+1} - \nabla F_k)}{(\nabla F_{k+1} - \nabla F_k)^t H_k(\nabla F_{k+1} - \nabla F_k)}$$

Der Algorithmus (PseudoCode)

- Initialisierung:

- \mathbf{x}_0 beliebig
- $\mathbf{g}_0 = \nabla F(\mathbf{x}_0)$
- $\mathbf{H}_0 = Id$

- Iteration

for $k = 0, 1, 2, \dots$

- $\mathbf{s}_k = -\mathbf{H}_k \mathbf{g}_k$
- $\alpha_k = \mathbf{argmin}_{t>0} \{ F(\mathbf{x}_k + t\mathbf{s}_k) \}$
- $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{s}_k$
- $\mathbf{g}_{k+1} = \nabla F(\mathbf{x}_{k+1})$
- $\mathbf{H}_{k+1} = \dots$ (z.B. **DFP** oder **BFGS**)

- Anmerkungen zur Konvergenz
- Falls F strikt konvex ist und $\lim_{||x|| \rightarrow \infty} F(x) = \infty$
dann gibt es ein eindeutiges Minimum
 - cg-Verfahren und die Quasi-Newton-Verfahren
konvergieren gegen dieses Minimum

Aussagen zur Konvergenzgeschwindigkeit sind schwierig.

- Für beliebiges F :
Aussagen zur Konvergenz sind schwierig.
- Falls das Verfahren konvergiert dann ist der Grenzwert
(hoffentlich ein) ein **lokales** Minimum.

- Im beliebigen Fall ist das Auffinden des globalen Minimums (sofern es existiert) sehr schwierig!
- ein nach wie vor ungelöstes Problem

- Teillösungen (ohne Garantie das globale Min zu finden)
 - ▶ Metropolis-Algorithmus
 - ▶ Simulated Annealing
 - ▶ Evolutionäre Algorithmen (genetische Algorithmen)
 - ▶

- Die besprochenen Verfahren und einiges mehr findet man in

Numerical Recipes in C/C++/ ???

- wie vieles andere, was im Rahmen dieser Vorlesung besprochen wurde (und noch wird) und noch viiiieel mehr!

- Klassifizierung der Optimierungsprobleme
 - mit/ohne NB
 - quadratisch, konvex, linear, beliebig
 - lokal/global
- mit NB: Lagrangesche Multiplikatoren
- ohne NB: Abstiegsverfahren
 - Gradienten
 - Newton
 - konjugierte Gradienten (cg)
 - Quasi-Newton
- 1D-Minimierung mittels Goldenem Schnitt

- cg-Verfahren
 - Für quadratische Zielfunktion
 - als linearer Gleichungslöser
 - Allgemeiner Fall (Fletcher/Reeves & Pollack/Ribière)
- Quasi-Newton Verfahren