Inhaltsübersicht für heute:

Skalierung und Steilster Abstieg

Quasi-Newton

Trust-Region-Verfahren

Numerisches Differenzieren

Automatisches Differenzieren

Inhaltsübersicht für heute:

Skalierung

Bei Verfahren 1. Ordnung (nur Gradient) hat die Skalierung der Variablen (z.B. ob Daten in Metern oder Millimetern gegeben sind) großen Einfluss auf das Konvergenzverhalten und ist kaum vernünftig anpassbar.
Das Newtonverfahren (nutzt 2. Ordnung) ist jedoch skalierungsunabhängig!

Bsp: $f(x) = \frac{1}{2}x^T Qx + q^T x + c$, $\nabla f(x) = Qx + q$, $\nabla^2 f(x) = Q$, $Q \succ 0$
exakter Line-Search für Abstiegsrichtung $h = -B^{-1}\nabla f(\bar{x})$ in $\bar{x}$ ($B \succ 0$):
$\Phi(\alpha) = \frac{1}{2}\bar{x}^T Q\bar{x} + q^T\bar{x} + c + \alpha\nabla f(\bar{x})^T h + \frac{\alpha^2}{2}h^T Qh$
$\Phi'(\alpha) = \nabla f(\bar{x})^T h + \alpha h^T Qh = 0 \quad \Rightarrow \quad \bar{\alpha} = \frac{-\nabla f^T h}{h^T Qh}$

Skalierung

Bei Verfahren 1. Ordnung (nur Gradient) hat die Skalierung der Variablen (z.B. ob Daten in Metern oder Millimetern gegeben sind) großen Einfluss auf das Konvergenzverhalten und ist kaum vernünftig anpassbar.
Das Newtonverfahren (nutzt 2. Ordnung) ist jedoch skalierungsunabhängig!

Bsp: $f(x) = \frac{1}{2}x^T Qx + q^T x + c$, $\nabla f(x) = Qx + q$, $\nabla^2 f(x) = Q$, $Q \succ 0$
exakter Line-Search für Abstiegsrichtung $h = -B^{-1}\nabla f(\bar{x})$ in $\bar{x}$ ($B \succ 0$):

$$\Phi(\alpha) = \frac{1}{2}\bar{x}^T Q\bar{x} + q^T\bar{x} + c + \alpha\nabla f(\bar{x})^T h + \frac{\alpha^2}{2}h^T Qh$$

$$\Phi'(\alpha) = \nabla f(\bar{x})^T h + \alpha h^T Qh = 0 \quad \Rightarrow \quad \bar{\alpha} = \frac{-\nabla f^T h}{h^T Qh}$$

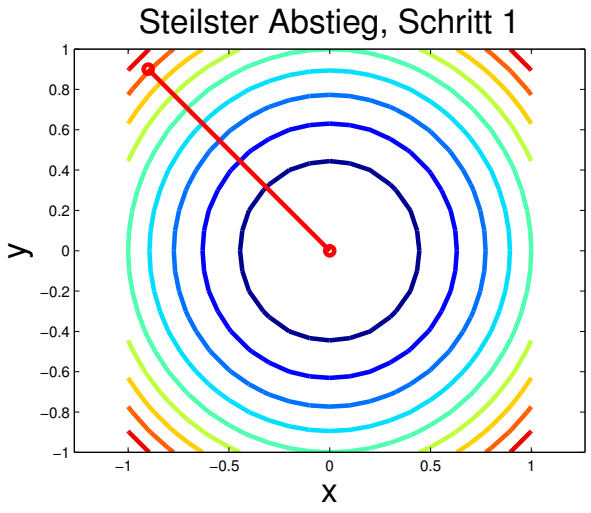
Ideal skaliert ($Q = I$):

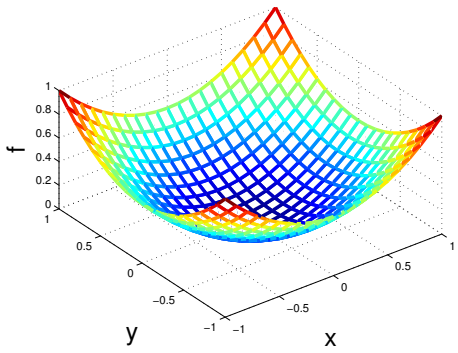
$$Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},\ q = 0, c = 0$$

Startpunkt $\begin{bmatrix} x_1^{(0)} \\ x_2^{(0)} \end{bmatrix} = \begin{bmatrix} -0.9 \\ 0.9 \end{bmatrix}$

Steilster Abstieg ($B = I$):

$$h = -\nabla f,\ \ \bar{\alpha} = \frac{\|h\|^2}{h^T Qh} = 1$$





Skalierung

Bei Verfahren 1. Ordnung (nur Gradient) hat die Skalierung der Variablen (z.B. ob Daten in Metern oder Millimetern gegeben sind) großen Einfluss auf das Konvergenzverhalten und ist kaum vernünftig anpassbar.
Das Newtonverfahren (nutzt 2. Ordnung) ist jedoch skalierungsunabhängig!

Bsp: $f(x) = \frac{1}{2}x^T Qx + q^T x + c$, $\nabla f(x) = Qx + q$, $\nabla^2 f(x) = Q$, $Q \succ 0$
exakter Line-Search für Abstiegsrichtung $h = -B^{-1}\nabla f(\bar{x})$ in $\bar{x}$ ($B \succ 0$):

$$\Phi(\alpha) = \frac{1}{2}\bar{x}^T Q\bar{x} + q^T\bar{x} + c + \alpha\nabla f(\bar{x})^T h + \frac{\alpha^2}{2}h^T Qh$$

$$\Phi'(\alpha) = \nabla f(\bar{x})^T h + \alpha h^T Qh = 0 \quad \Rightarrow \quad \bar{\alpha} = \frac{-\nabla f^T h}{h^T Qh}$$

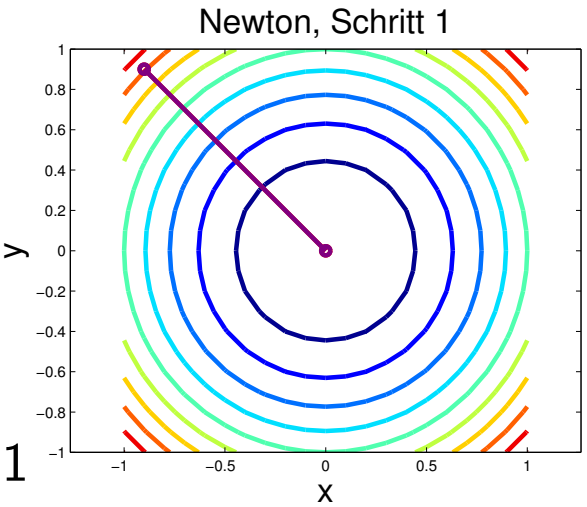
Ideal skaliert ($Q = I$):

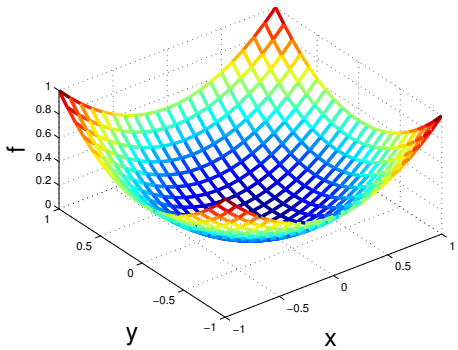
$$Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},\ q = 0, c = 0$$

Startpunkt $\begin{bmatrix} x_1^{(0)} \\ x_2^{(0)} \end{bmatrix} = \begin{bmatrix} -0.9 \\ 0.9 \end{bmatrix}$

Newton ($B = Q$):

$$h = -Q^{-1}\nabla f,\ \bar{\alpha} = \frac{\nabla f^T Q^{-1}\nabla f}{\nabla f^T Q^{-1}\nabla f} = 1$$





Skalierung

Bei Verfahren 1. Ordnung (nur Gradient) hat die Skalierung der Variablen (z.B. ob Daten in Metern oder Millimetern gegeben sind) großen Einfluss auf das Konvergenzverhalten und ist kaum vernünftig anpassbar.
Das Newtonverfahren (nutzt 2. Ordnung) ist jedoch skalierungsunabhängig!

Bsp: $f(x) = \frac{1}{2}x^T Qx + q^T x + c$, $\nabla f(x) = Qx + q$, $\nabla^2 f(x) = Q$, $Q \succ 0$
exakter Line-Search für Abstiegsrichtung $h = -B^{-1}\nabla f(\bar{x})$ in $\bar{x}$ ($B \succ 0$):

$$\Phi(\alpha) = \frac{1}{2}\bar{x}^T Q\bar{x} + q^T\bar{x} + c + \alpha\nabla f(\bar{x})^T h + \frac{\alpha^2}{2}h^T Qh$$

$$\Phi'(\alpha) = \nabla f(\bar{x})^T h + \alpha h^T Qh = 0 \quad \Rightarrow \quad \bar{\alpha} = \frac{-\nabla f^T h}{h^T Qh}$$

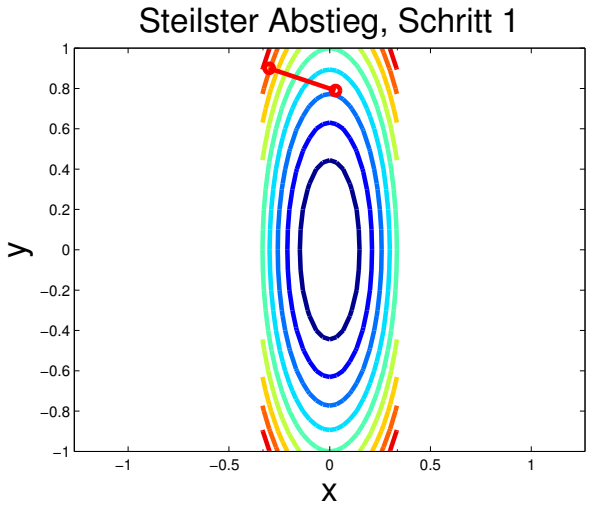
Noch gut skaliert! ($x_1 = 3\tilde{x}_1$):

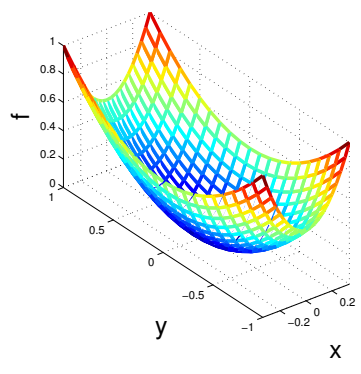
$$\tilde{Q} = \begin{bmatrix} 3^2 & 0 \\ 0 & 1 \end{bmatrix},\ q = 0,\ c = 0$$

Startpunkt $\begin{bmatrix} \tilde{x}_1^{(0)} \\ x_2^{(0)} \end{bmatrix} = \begin{bmatrix} -0.3 \\ 0.9 \end{bmatrix}$

Steilster Abstieg ($B = I$):

$$h = -\nabla f,\quad \bar{\alpha} = \frac{\|h\|^2}{h^T\tilde{Q}h}$$





Skalierung

Bei Verfahren 1. Ordnung (nur Gradient) hat die Skalierung der Variablen (z.B. ob Daten in Metern oder Millimetern gegeben sind) großen Einfluss auf das Konvergenzverhalten und ist kaum vernünftig anpassbar.
Das Newtonverfahren (nutzt 2. Ordnung) ist jedoch skalierungsunabhängig!

Bsp: $f(x) = \frac{1}{2}x^T Qx + q^T x + c$, $\nabla f(x) = Qx + q$, $\nabla^2 f(x) = Q$, $Q \succ 0$
exakter Line-Search für Abstiegsrichtung $h = -B^{-1}\nabla f(\bar{x})$ in $\bar{x}$ ($B \succ 0$):

$$\Phi(\alpha) = \frac{1}{2}\bar{x}^T Q\bar{x} + q^T\bar{x} + c + \alpha\nabla f(\bar{x})^T h + \frac{\alpha^2}{2}h^T Qh$$

$$\Phi'(\alpha) = \nabla f(\bar{x})^T h + \alpha h^T Qh = 0 \quad \Rightarrow \quad \bar{\alpha} = \frac{-\nabla f^T h}{h^T Qh}$$

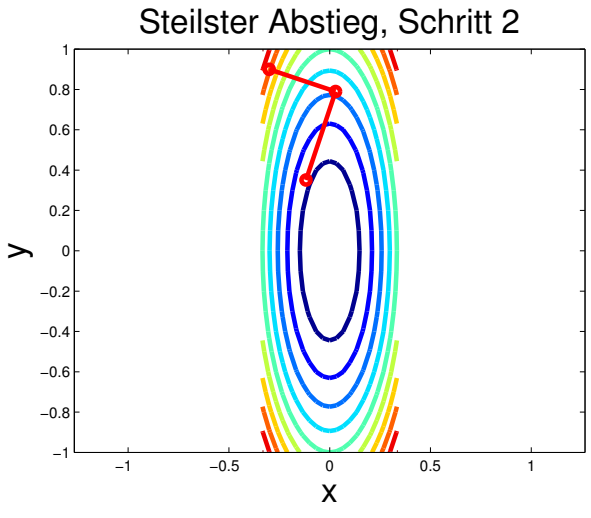
Noch gut skaliert! ($x_1 = 3\tilde{x}_1$):

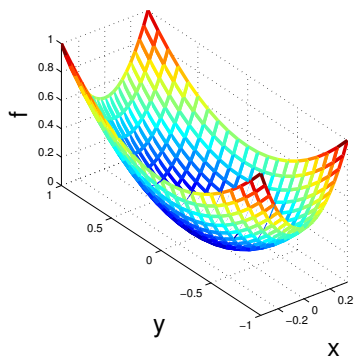
$$\tilde{Q} = \begin{bmatrix} 3^2 & 0 \\ 0 & 1 \end{bmatrix},\ q = 0,\ c = 0$$

Startpunkt $\begin{bmatrix} \tilde{x}_1^{(0)} \\ x_2^{(0)} \end{bmatrix} = \begin{bmatrix} -0.3 \\ 0.9 \end{bmatrix}$

Steilster Abstieg ($B = I$):

$$h = -\nabla f,\quad \bar{\alpha} = \frac{\|h\|^2}{h^T\tilde{Q}h}$$





Skalierung

Bei Verfahren 1. Ordnung (nur Gradient) hat die Skalierung der Variablen (z.B. ob Daten in Metern oder Millimetern gegeben sind) großen Einfluss auf das Konvergenzverhalten und ist kaum vernünftig anpassbar.
Das Newtonverfahren (nutzt 2. Ordnung) ist jedoch skalierungsunabhängig!

Bsp: $f(x) = \frac{1}{2}x^T Qx + q^T x + c$, $\nabla f(x) = Qx + q$, $\nabla^2 f(x) = Q$, $Q \succ 0$
exakter Line-Search für Abstiegsrichtung $h = -B^{-1}\nabla f(\bar{x})$ in $\bar{x}$ ($B \succ 0$):

$$\Phi(\alpha) = \frac{1}{2}\bar{x}^T Q\bar{x} + q^T\bar{x} + c + \alpha\nabla f(\bar{x})^T h + \frac{\alpha^2}{2}h^T Qh$$

$$\Phi'(\alpha) = \nabla f(\bar{x})^T h + \alpha h^T Qh = 0 \quad \Rightarrow \quad \bar{\alpha} = \frac{-\nabla f^T h}{h^T Qh}$$

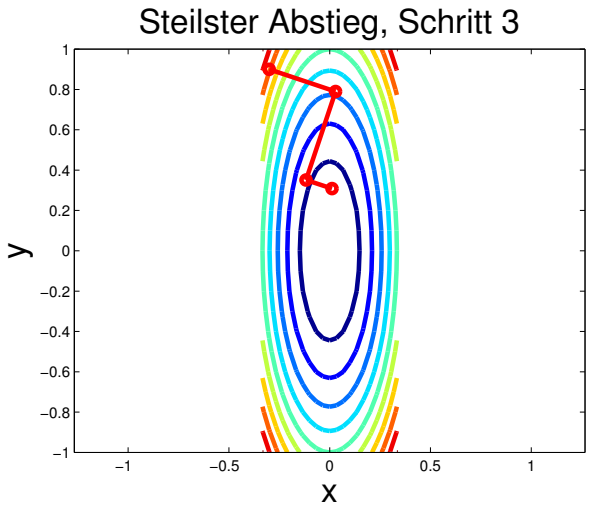
Noch gut skaliert! ($x_1 = 3\tilde{x}_1$):

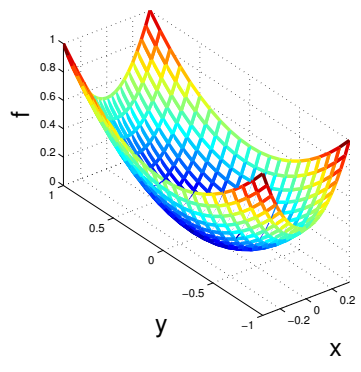
$$\tilde{Q} = \begin{bmatrix} 3^2 & 0 \\ 0 & 1 \end{bmatrix},\ q = 0,\ c = 0$$

Startpunkt $\begin{bmatrix} \tilde{x}_1^{(0)} \\ x_2^{(0)} \end{bmatrix} = \begin{bmatrix} -0.3 \\ 0.9 \end{bmatrix}$

Steilster Abstieg ($B = I$):

$$h = -\nabla f,\quad \bar{\alpha} = \frac{\|h\|^2}{h^T\tilde{Q}h}$$





Skalierung

Bei Verfahren 1. Ordnung (nur Gradient) hat die Skalierung der Variablen (z.B. ob Daten in Metern oder Millimetern gegeben sind) großen Einfluss auf das Konvergenzverhalten und ist kaum vernünftig anpassbar.
Das Newtonverfahren (nutzt 2. Ordnung) ist jedoch skalierungsunabhängig!

Bsp: $f(x) = \frac{1}{2}x^T Qx + q^T x + c$, $\nabla f(x) = Qx + q$, $\nabla^2 f(x) = Q$, $Q \succ 0$
exakter Line-Search für Abstiegsrichtung $h = -B^{-1}\nabla f(\bar{x})$ in $\bar{x}$ ($B \succ 0$):

$$\Phi(\alpha) = \frac{1}{2}\bar{x}^T Q\bar{x} + q^T\bar{x} + c + \alpha\nabla f(\bar{x})^T h + \frac{\alpha^2}{2}h^T Qh$$

$$\Phi'(\alpha) = \nabla f(\bar{x})^T h + \alpha h^T Qh = 0 \quad \Rightarrow \quad \bar{\alpha} = \frac{-\nabla f^T h}{h^T Qh}$$

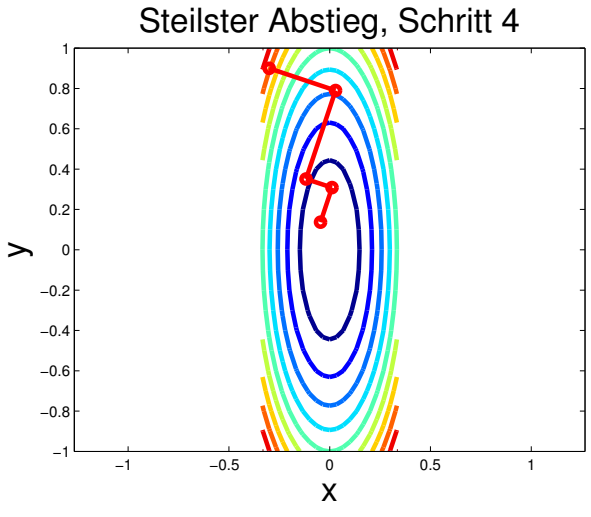
Noch gut skaliert! ($x_1 = 3\tilde{x}_1$):

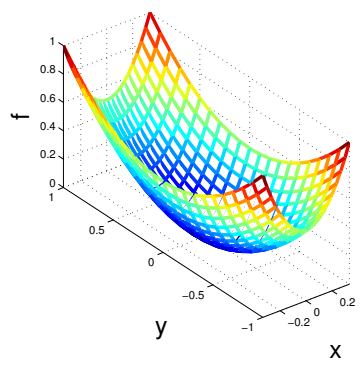
$$\tilde{Q} = \begin{bmatrix} 3^2 & 0 \\ 0 & 1 \end{bmatrix},\ q = 0,\ c = 0$$

Startpunkt $\begin{bmatrix} \tilde{x}_1^{(0)} \\ x_2^{(0)} \end{bmatrix} = \begin{bmatrix} -0.3 \\ 0.9 \end{bmatrix}$

Steilster Abstieg ($B = I$):

$$h = -\nabla f,\ \ \bar{\alpha} = \frac{\|h\|^2}{h^T\tilde{Q}h}$$





Skalierung

Bei Verfahren 1. Ordnung (nur Gradient) hat die Skalierung der Variablen (z.B. ob Daten in Metern oder Millimetern gegeben sind) großen Einfluss auf das Konvergenzverhalten und ist kaum vernünftig anpassbar.
Das Newtonverfahren (nutzt 2. Ordnung) ist jedoch skalierungsunabhängig!

Bsp: $f(x) = \frac{1}{2}x^T Qx + q^T x + c$, $\nabla f(x) = Qx + q$, $\nabla^2 f(x) = Q$, $Q \succ 0$
exakter Line-Search für Abstiegsrichtung $h = -B^{-1}\nabla f(\bar{x})$ in $\bar{x}$ ($B \succ 0$):

$$\Phi(\alpha) = \frac{1}{2}\bar{x}^T Q\bar{x} + q^T\bar{x} + c + \alpha\nabla f(\bar{x})^T h + \frac{\alpha^2}{2}h^T Qh$$

$$\Phi'(\alpha) = \nabla f(\bar{x})^T h + \alpha h^T Qh = 0 \quad \Rightarrow \quad \bar{\alpha} = \frac{-\nabla f^T h}{h^T Qh}$$

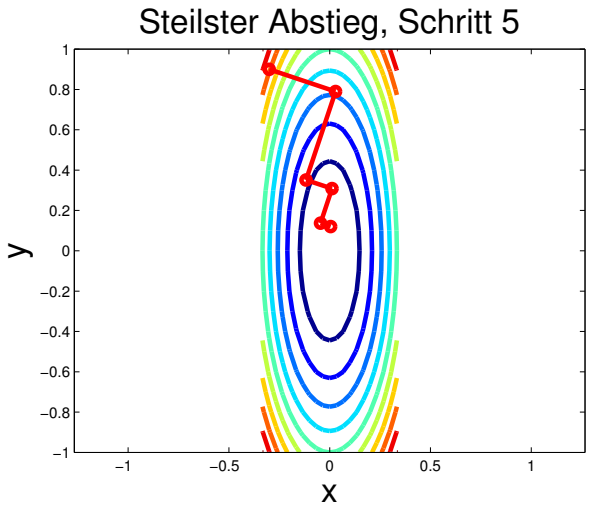
Noch gut skaliert! ($x_1 = 3\tilde{x}_1$):

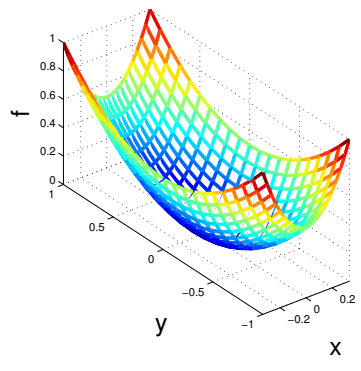
$$\tilde{Q} = \begin{bmatrix} 3^2 & 0 \\ 0 & 1 \end{bmatrix},\ q = 0,\ c = 0$$

Startpunkt $\begin{bmatrix} \tilde{x}_1^{(0)} \\ x_2^{(0)} \end{bmatrix} = \begin{bmatrix} -0.3 \\ 0.9 \end{bmatrix}$

Steilster Abstieg ($B = I$):

$$h = -\nabla f,\ \ \bar{\alpha} = \frac{\|h\|^2}{h^T\tilde{Q}h}$$





Skalierung

Bei Verfahren 1. Ordnung (nur Gradient) hat die Skalierung der Variablen (z.B. ob Daten in Metern oder Millimetern gegeben sind) großen Einfluss auf das Konvergenzverhalten und ist kaum vernünftig anpassbar.
Das Newtonverfahren (nutzt 2. Ordnung) ist jedoch skalierungsunabhängig!

Bsp: $f(x) = \frac{1}{2}x^T Qx + q^T x + c$, $\nabla f(x) = Qx + q$, $\nabla^2 f(x) = Q$, $Q \succ 0$
exakter Line-Search für Abstiegsrichtung $h = -B^{-1}\nabla f(\bar{x})$ in $\bar{x}$ ($B \succ 0$):

$$\Phi(\alpha) = \frac{1}{2}\bar{x}^T Q\bar{x} + q^T\bar{x} + c + \alpha\nabla f(\bar{x})^T h + \frac{\alpha^2}{2}h^T Qh$$

$$\Phi'(\alpha) = \nabla f(\bar{x})^T h + \alpha h^T Qh = 0 \quad \Rightarrow \quad \bar{\alpha} = \frac{-\nabla f^T h}{h^T Qh}$$

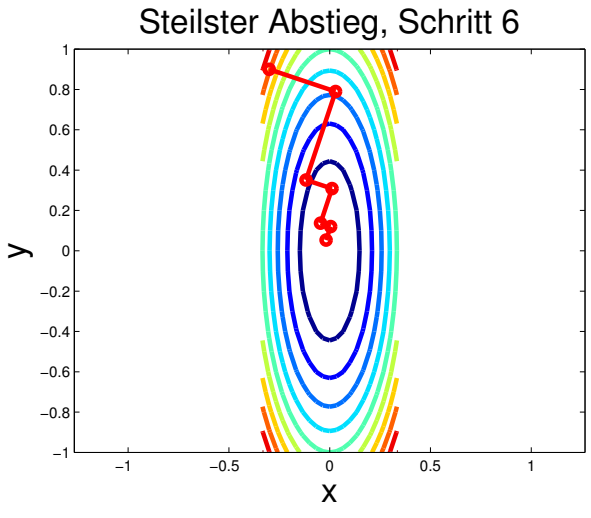
Noch gut skaliert! ($x_1 = 3\tilde{x}_1$):

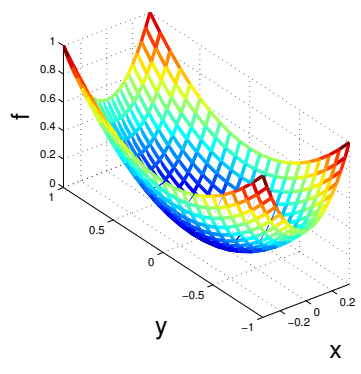
$$\tilde{Q} = \begin{bmatrix} 3^2 & 0 \\ 0 & 1 \end{bmatrix},\ q = 0,\ c = 0$$

Startpunkt $\begin{bmatrix} \tilde{x}_1^{(0)} \\ x_2^{(0)} \end{bmatrix} = \begin{bmatrix} -0.3 \\ 0.9 \end{bmatrix}$

Steilster Abstieg ($B = I$):

$$h = -\nabla f,\quad \bar{\alpha} = \frac{\|h\|^2}{h^T\tilde{Q}h}$$





Skalierung

Bei Verfahren 1. Ordnung (nur Gradient) hat die Skalierung der Variablen (z.B. ob Daten in Metern oder Millimetern gegeben sind) großen Einfluss auf das Konvergenzverhalten und ist kaum vernünftig anpassbar.
Das Newtonverfahren (nutzt 2. Ordnung) ist jedoch skalierungsunabhängig!

Bsp: $f(x) = \frac{1}{2}x^T Qx + q^T x + c$, $\nabla f(x) = Qx + q$, $\nabla^2 f(x) = Q$, $Q \succ 0$
exakter Line-Search für Abstiegsrichtung $h = -B^{-1}\nabla f(\bar{x})$ in $\bar{x}$ ($B \succ 0$):

$$\Phi(\alpha) = \frac{1}{2}\bar{x}^T Q\bar{x} + q^T\bar{x} + c + \alpha\nabla f(\bar{x})^T h + \frac{\alpha^2}{2}h^T Qh$$

$$\Phi'(\alpha) = \nabla f(\bar{x})^T h + \alpha h^T Qh = 0 \quad \Rightarrow \quad \bar{\alpha} = \frac{-\nabla f^T h}{h^T Qh}$$

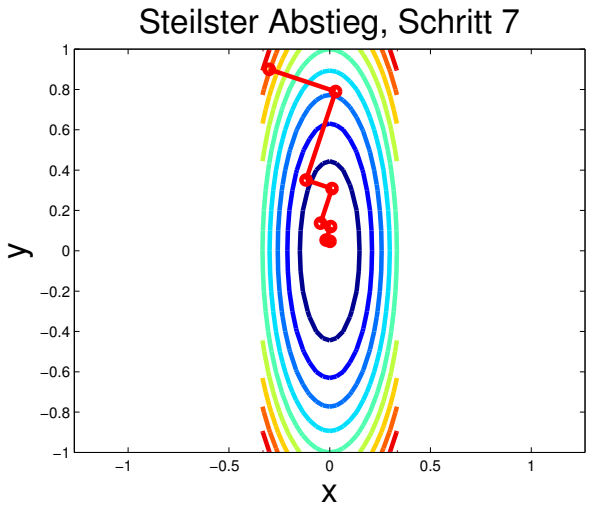
Noch gut skaliert! ($x_1 = 3\tilde{x}_1$):

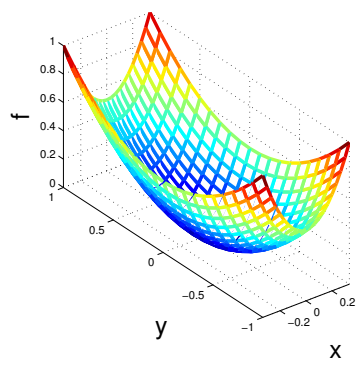
$$\tilde{Q} = \begin{bmatrix} 3^2 & 0 \\ 0 & 1 \end{bmatrix},\ q = 0,\ c = 0$$

Startpunkt $\begin{bmatrix} \tilde{x}_1^{(0)} \\ x_2^{(0)} \end{bmatrix} = \begin{bmatrix} -0.3 \\ 0.9 \end{bmatrix}$

Steilster Abstieg ($B = I$):

$$h = -\nabla f,\quad \bar{\alpha} = \frac{\|h\|^2}{h^T\tilde{Q}h}$$





Skalierung

Bei Verfahren 1. Ordnung (nur Gradient) hat die Skalierung der Variablen (z.B. ob Daten in Metern oder Millimetern gegeben sind) großen Einfluss auf das Konvergenzverhalten und ist kaum vernünftig anpassbar.
Das Newtonverfahren (nutzt 2. Ordnung) ist jedoch skalierungsunabhängig!

Bsp: $f(x) = \frac{1}{2}x^T Qx + q^T x + c$, $\nabla f(x) = Qx + q$, $\nabla^2 f(x) = Q$, $Q \succ 0$
exakter Line-Search für Abstiegsrichtung $h = -B^{-1}\nabla f(\bar{x})$ in $\bar{x}$ ($B \succ 0$):

$$\Phi(\alpha) = \frac{1}{2}\bar{x}^T Q\bar{x} + q^T\bar{x} + c + \alpha\nabla f(\bar{x})^T h + \frac{\alpha^2}{2}h^T Qh$$

$$\Phi'(\alpha) = \nabla f(\bar{x})^T h + \alpha h^T Qh = 0 \quad \Rightarrow \quad \bar{\alpha} = \frac{-\nabla f^T h}{h^T Qh}$$

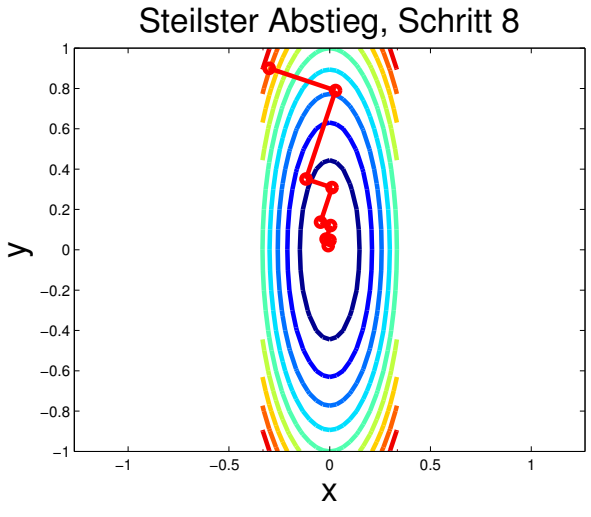
Noch gut skaliert! ($x_1 = 3\tilde{x}_1$):

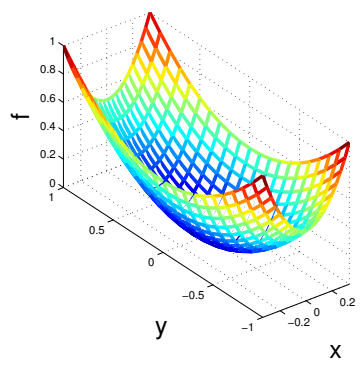
$$\tilde{Q} = \begin{bmatrix} 3^2 & 0 \\ 0 & 1 \end{bmatrix},\ q = 0,\ c = 0$$

Startpunkt $\begin{bmatrix} \tilde{x}_1^{(0)} \\ x_2^{(0)} \end{bmatrix} = \begin{bmatrix} -0.3 \\ 0.9 \end{bmatrix}$

Steilster Abstieg ($B = I$):

$$h = -\nabla f,\quad \bar{\alpha} = \frac{\|h\|^2}{h^T\tilde{Q}h}$$





Skalierung

Bei Verfahren 1. Ordnung (nur Gradient) hat die Skalierung der Variablen (z.B. ob Daten in Metern oder Millimetern gegeben sind) großen Einfluss auf das Konvergenzverhalten und ist kaum vernünftig anpassbar.
Das Newtonverfahren (nutzt 2. Ordnung) ist jedoch skalierungsunabhängig!

Bsp: $f(x) = \frac{1}{2}x^T Qx + q^T x + c$, $\nabla f(x) = Qx + q$, $\nabla^2 f(x) = Q$, $Q \succ 0$
exakter Line-Search für Abstiegsrichtung $h = -B^{-1}\nabla f(\bar{x})$ in $\bar{x}$ ($B \succ 0$):

$$\Phi(\alpha) = \frac{1}{2}\bar{x}^T Q\bar{x} + q^T\bar{x} + c + \alpha\nabla f(\bar{x})^T h + \frac{\alpha^2}{2}h^T Qh$$

$$\Phi'(\alpha) = \nabla f(\bar{x})^T h + \alpha h^T Qh = 0 \quad \Rightarrow \quad \bar{\alpha} = \frac{-\nabla f^T h}{h^T Qh}$$

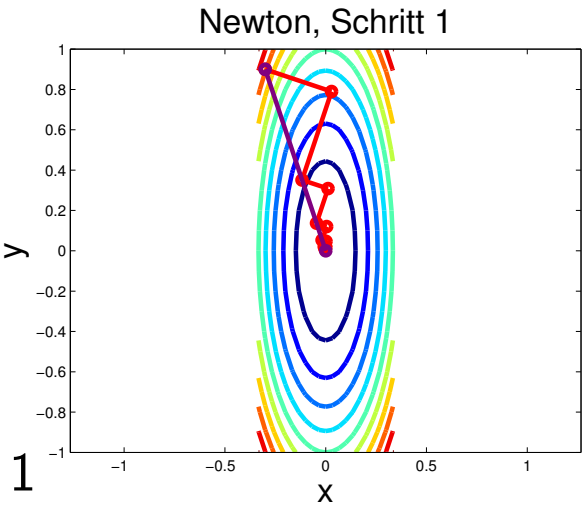
Noch gut skaliert! ($x_1 = 3\tilde{x}_1$):

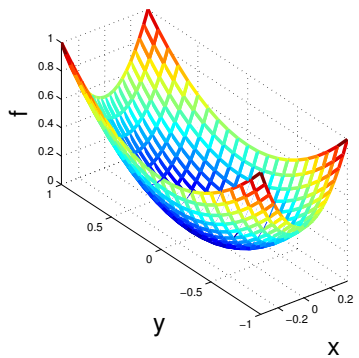
$$\tilde{Q} = \begin{bmatrix} 3^2 & 0 \\ 0 & 1 \end{bmatrix},\ q = 0,\ c = 0$$

Startpunkt $\begin{bmatrix} \tilde{x}_1^{(0)} \\ x_2^{(0)} \end{bmatrix} = \begin{bmatrix} -0.3 \\ 0.9 \end{bmatrix}$

Newton ($B = \tilde{Q}$):

$$h = -\tilde{Q}^{-1}\nabla f,\ \bar{\alpha} = \frac{\nabla f^T \tilde{Q}^{-1}\nabla f}{\nabla f^T \tilde{Q}^{-1}\nabla f} = 1$$





Steilster Abstieg konvergiert SEHR langsam

Ein Verfahren konvergiert linear gegen x^*, wenn es eine Konstante $0 < \gamma < 1$ gibt mit $\|x_{k+1} - x^*\| \leq \gamma \|x_k - x^*\|$.

Satz

Für $f(x) = \frac{1}{2} x^T Q x + q^T x + c$ mit $Q \succ 0$ und exaktem Line-Search konvergiert das steilste Abstiegsverfahren im Allgemeinen linear mit Konstante $\gamma = \frac{\lambda_{\max}(Q) - \lambda_{\min}(Q)}{\lambda_{\max}(Q) + \lambda_{\min}(Q)}$.

Für $\lambda_{\max} \gg \lambda_{\min}$ (die Niveaumengen sind ganz schmale Ellipsen) ist $\gamma \approx 1$ und man sieht kaum Verbesserung.

Steilster Abstieg konvergiert SEHR langsam

Ein Verfahren konvergiert linear gegen x^*, wenn es eine Konstante $0 < \gamma < 1$ gibt mit $\|x_{k+1} - x^*\| \leq \gamma \|x_k - x^*\|$.

Satz

Für $f(x) = \frac{1}{2} x^T Q x + q^T x + c$ mit $Q \succ 0$ und exaktem Line-Search konvergiert das steilste Abstiegsverfahren im Allgemeinen linear mit Konstante $\gamma = \frac{\lambda_{\max}(Q) - \lambda_{\min}(Q)}{\lambda_{\max}(Q) + \lambda_{\min}(Q)}$.

Für $\lambda_{\max} \gg \lambda_{\min}$ (die Niveaumengen sind ganz schmale Ellipsen) ist $\gamma \approx 1$ und man sieht kaum Verbesserung.

In der Nähe eines Optimums mit hinreichenden Optimalitätsbedingungen ist die Funktion annähernd quadratisch streng konvex

$\rightarrow$ steilster Abstieg konvergiert am Ende fast immer schlecht

$\rightarrow$ Newton konvergiert am Ende fast immer hervorragend

Steilster Abstieg konvergiert SEHR langsam

Ein Verfahren konvergiert linear gegen x^*, wenn es eine Konstante $0 < \gamma < 1$ gibt mit $\|x_{k+1} - x^*\| \leq \gamma \|x_k - x^*\|$.

Satz

Für $f(x) = \frac{1}{2} x^T Q x + q^T x + c$ mit $Q \succ 0$ und exaktem Line-Search konvergiert das steilste Abstiegsverfahren im Allgemeinen linear mit Konstante $\gamma = \frac{\lambda_{\max}(Q) - \lambda_{\min}(Q)}{\lambda_{\max}(Q) + \lambda_{\min}(Q)}$.

Für $\lambda_{\max} \gg \lambda_{\min}$ (die Niveaumengen sind ganz schmale Ellipsen) ist $\gamma \approx 1$ und man sieht kaum Verbesserung.

In der Nähe eines Optimums mit hinreichenden Optimalitätsbedingungen ist die Funktion annähernd quadratisch streng konvex

$\rightarrow$ steilster Abstieg konvergiert am Ende fast immer schlecht

$\rightarrow$ Newton konvergiert am Ende fast immer hervorragend

---

Da für große n jede Iteration des Newton-Verfahrens wegen der Berechnung von $\nabla^2 f$ und $-(\nabla^2 f)^{-1} \nabla f$ sehr aufwendig ist, versucht man $(\nabla^2 f)^{-1}$ sukzessive aus den Werten von ∇f zu approximieren.

Inhaltsübersicht für heute:

Quasi-Newton-Verfahren

Ein Verfahren konvergiert superlinear gegen x^*, falls $\lim_{k\to\infty} \frac{\|x^{(k+1)}-x^*\|}{\|x^{(k)}-x^*\|} = 0$ (wird kleiner als jede Konstante der linearen Konvergenz).

Satz

Konvergiert $x^{(k+1)} = x^{(k)} + h^{(k)}$ gegen ein x^*, das die hinreichenden Opt.-Bed. erfüllt, so ist die Konvergenz superlinear genau dann, wenn die Schrittrichtung $h^{(k)}$ sich schneller der Newton-Richtung $h_N^{(k)}$ annähert als sie klein wird, $\|h^{(k)} - h_N^{(k)}\| = \mathbf{o}(\|h^{(k)}\|)$.

Quasi-Newton-Verfahren

Ein Verfahren konvergiert superlinear gegen x^*, falls $\lim_{k\to\infty} \frac{\|x^{(k+1)}-x^*\|}{\|x^{(k)}-x^*\|} = 0$ (wird kleiner als jede Konstante der linearen Konvergenz).

Satz

Konvergiert $x^{(k+1)} = x^{(k)} + h^{(k)}$ gegen ein x^*, das die hinreichenden Opt.-Bed. erfüllt, so ist die Konvergenz superlinear genau dann, wenn die Schrittrichtung $h^{(k)}$ sich schneller der Newton-Richtung $h_N^{(k)}$ annähert als sie klein wird, $\|h^{(k)} - h_N^{(k)}\| = \mathbf{o}(\|h^{(k)}\|)$.

$\Rightarrow$ Superlineare Konvergenz erfordert Approximation der Newton-Richtung.
Für $h = -B^{-1}\nabla f$ sollte B also die Hessematrix nachbauen. Diese erfüllt

$$\nabla f(x+h) = \nabla f(x) + \nabla^2 f(x)h + \mathbf{o}(h). \quad \text{[Taylor]}$$

Quasi-Newton-Verfahren

Ein Verfahren konvergiert superlinear gegen x^*, falls $\lim_{k\to\infty} \frac{\|x^{(k+1)}-x^*\|}{\|x^{(k)}-x^*\|} = 0$ (wird kleiner als jede Konstante der linearen Konvergenz).

Satz

Konvergiert $x^{(k+1)} = x^{(k)} + h^{(k)}$ gegen ein x^*, das die hinreichenden Opt.-Bed. erfüllt, so ist die Konvergenz superlinear genau dann, wenn die Schrittrichtung $h^{(k)}$ sich schneller der Newton-Richtung $h_N^{(k)}$ annähert als sie klein wird, $\|h^{(k)} - h_N^{(k)}\| = \mathbf{o}(\|h^{(k)}\|)$.

$\Rightarrow$ Superlineare Konvergenz erfordert Approximation der Newton-Richtung.
Für $h = -B^{-1}\nabla f$ sollte B also die Hessematrix nachbauen. Diese erfüllt

$$\nabla f(x+h) = \nabla f(x) + \nabla^2 f(x)h + \mathbf{o}(h). \quad \text{[Taylor]}$$

Forderungen an B_{k+1} für Schrittrichtung: $h^{(k+1)} = -B_{k+1}^{-1}\nabla f_{k+1}$
$\rightarrow B_{k+1} \succ 0$, damit $h^{(k+1)}$ Abstiegsrichtung ist

Quasi-Newton-Verfahren

Ein Verfahren konvergiert superlinear gegen x^*, falls $\lim_{k\to\infty} \frac{\|x^{(k+1)}-x^*\|}{\|x^{(k)}-x^*\|} = 0$ (wird kleiner als jede Konstante der linearen Konvergenz).

Satz

Konvergiert $x^{(k+1)} = x^{(k)} + h^{(k)}$ gegen ein x^*, das die hinreichenden Opt.-Bed. erfüllt, so ist die Konvergenz superlinear genau dann, wenn die Schrittrichtung $h^{(k)}$ sich schneller der Newton-Richtung $h_N^{(k)}$ annähert als sie klein wird, $\|h^{(k)} - h_N^{(k)}\| = \mathbf{o}(\|h^{(k)}\|)$.

$\Rightarrow$ Superlineare Konvergenz erfordert Approximation der Newton-Richtung.
Für $h = -B^{-1}\nabla f$ sollte B also die Hessematrix nachbauen. Diese erfüllt

$$\nabla f(x+h) = \nabla f(x) + \nabla^2 f(x)h + \mathbf{o}(h). \quad \text{[Taylor]}$$

Forderungen an B_{k+1} für Schrittrichtung: $h^{(k+1)} = -B_{k+1}^{-1}\nabla f_{k+1}$

$\rightarrow$ $B_{k+1} \succ 0$, damit $h^{(k+1)}$ Abstiegsrichtung ist

$\rightarrow$ Das quadratische Modell $m_{k+1}(h) := f_{k+1} + \nabla f_{k+1}^T h + \frac{1}{2} h^T B_{k+1} h$ sollte in x_k den Gradienten ∇f_k gut approximieren:

$$\nabla f_k = \nabla_h m_{k+1}(x^{(k)} - x^{(k+1)}) = \nabla f_{k+1} + B_{k+1}(x^{(k)} - x^{(k+1)})$$

Quasi-Newton-Verfahren

Ein Verfahren konvergiert superlinear gegen x^*, falls $\lim_{k\to\infty} \frac{\|x^{(k+1)}-x^*\|}{\|x^{(k)}-x^*\|} = 0$ (wird kleiner als jede Konstante der linearen Konvergenz).

Satz

Konvergiert $x^{(k+1)} = x^{(k)} + h^{(k)}$ gegen ein x^*, das die hinreichenden Opt.-Bed. erfüllt, so ist die Konvergenz superlinear genau dann, wenn die Schrittrichtung $h^{(k)}$ sich schneller der Newton-Richtung $h_N^{(k)}$ annähert als sie klein wird, $\|h^{(k)} - h_N^{(k)}\| = \mathbf{o}(\|h^{(k)}\|)$.

$\Rightarrow$ Superlineare Konvergenz erfordert Approximation der Newton-Richtung.
Für $h = -B^{-1}\nabla f$ sollte B also die Hessematrix nachbauen. Diese erfüllt

$$\nabla f(x+h) = \nabla f(x) + \nabla^2 f(x)h + \mathbf{o}(h). \quad \text{[Taylor]}$$

Forderungen an B_{k+1} für Schrittrichtung: $h^{(k+1)} = -B_{k+1}^{-1}\nabla f_{k+1}$

$\rightarrow$ $B_{k+1} \succ 0$, damit $h^{(k+1)}$ Abstiegsrichtung ist

$\rightarrow$ Das quadratische Modell $m_{k+1}(h) := f_{k+1} + \nabla f_{k+1}^T h + \frac{1}{2} h^T B_{k+1} h$ sollte in x_k den Gradienten ∇f_k gut approximieren:

$$\nabla f_k = \nabla_h m_{k+1}(x^{(k)} - x^{(k+1)}) = \nabla f_{k+1} + B_{k+1}(x^{(k)} - x^{(k+1)})$$

Sekanten-Gleichung: $B_{k+1}(x^{(k+1)} - x^{(k)}) = \nabla f_{k+1} - \nabla f_k$

Quasi-Newton mit BFGS-Update

Wegen Sekanten-Glg., $B_{k+1} \succ 0$ und $x^{(k+1)} = x^{(k)} + \alpha_k h^{(k)}$ muss

$$0 < \alpha_k^2 (h^{(k)})^T B_{k+1} h^{(k)} = \alpha (\nabla f_{k+1} - \nabla f_k)^T h^{(k)}$$

gelten. Das garantiert die Krümmung in den Wolfe-Bedingungen:

$$\alpha_k [\nabla f_{k+1}^T h^{(k)} - \nabla f_k^T h^{(k)}] > \alpha_k \underbrace{(c_2 - 1)}_{<0} \underbrace{\nabla f_k^T h^{(k)}}_{<0} > 0$$

Quasi-Newton mit BFGS-Update

Wegen Sekanten-Glg., $B_{k+1} \succ 0$ und $x^{(k+1)} = x^{(k)} + \alpha_k h^{(k)}$ muss

$$0 < \alpha_k^2 (h^{(k)})^T B_{k+1} h^{(k)} = \alpha (\nabla f_{k+1} - \nabla f_k)^T h^{(k)}$$

gelten. Das garantiert die Krümmung in den Wolfe-Bedingungen:

$$\alpha_k [\nabla f_{k+1}^T h^{(k)} - \nabla f_k^T h^{(k)}] > \alpha_k \underbrace{(c_2 - 1)}_{<0} \underbrace{\nabla f_k^T h^{(k)}}_{<0} > 0$$

Für $h^{(k+1)} = -B_{k+1}^{-1} \nabla f_{k+1}$ ist die Inverse $H_{k+1} := B_{k+1}^{-1}$ günstiger.
Die Matrix $H_{k+1} \succ 0$ mit $H_{k+1}(\nabla f_{k+1} - \nabla f_k) = x^{(k+1)} - x^{(k)}$, die sich gegenüber H_k in geeigneter Norm am wenigsten ändert, erhält man durch die Rang-2-Korrektur von Broyden, Fletcher, Goldfarb und Shanno:

$$H_{k+1} = (I - \frac{1}{s_k^T y_k} s_k y_k^T) H_k (I - \frac{1}{s_k^T y_k} y_k s_k^T) + \frac{1}{s_k^T y_k} s_k s_k^T$$

wobei $s_k := x^{(k+1)} - x^{(k)}$, $y_k := \nabla f_{k+1} - \nabla f_k$.

Quasi-Newton mit BFGS-Update

Wegen Sekanten-Glg., $B_{k+1} \succ 0$ und $x^{(k+1)} = x^{(k)} + \alpha_k h^{(k)}$ muss

$$0 < \alpha_k^2 (h^{(k)})^T B_{k+1} h^{(k)} = \alpha (\nabla f_{k+1} - \nabla f_k)^T h^{(k)} \quad [= y_k^T s_k]$$

gelten. Das garantiert die Krümmung in den Wolfe-Bedingungen:

$$\alpha_k [\nabla f_{k+1}^T h^{(k)} - \nabla f_k^T h^{(k)}] > \alpha_k \underbrace{(c_2 - 1)}_{<0} \underbrace{\nabla f_k^T h^{(k)}}_{<0} > 0$$

Für $h^{(k+1)} = -B_{k+1}^{-1} \nabla f_{k+1}$ ist die Inverse $H_{k+1} := B_{k+1}^{-1}$ günstiger.
Die Matrix $H_{k+1} \succ 0$ mit $H_{k+1}(\nabla f_{k+1} - \nabla f_k) = x^{(k+1)} - x^{(k)}$, die sich gegenüber H_k in geeigneter Norm am wenigsten ändert, erhält man durch die Rang-2-Korrektur von Broyden, Fletcher, Goldfarb und Shanno:

$$H_{k+1} = (I - \frac{1}{s_k^T y_k} s_k y_k^T) H_k (I - \frac{1}{s_k^T y_k} y_k s_k^T) + \frac{1}{s_k^T y_k} s_k s_k^T$$

wobei $s_k := x^{(k+1)} - x^{(k)}$, $y_k := \nabla f_{k+1} - \nabla f_k$. [$H_k \succ 0$ und Wolfe $\Rightarrow$ $H_{k+1} \succ 0$, denn $s_k^T H_{k+1} s_k \geq \frac{1}{s_k^T y_k} (s_k^T s_k)^2 > 0$, $y_k^T H_{k+1} y_k = s_k^T y_k > 0$.]

Quasi-Newton mit BFGS-Update

Wegen Sekanten-Glg., $B_{k+1} \succ 0$ und $x^{(k+1)} = x^{(k)} + \alpha_k h^{(k)}$ muss

$$0 < \alpha_k^2 (h^{(k)})^T B_{k+1} h^{(k)} = \alpha (\nabla f_{k+1} - \nabla f_k)^T h^{(k)} \quad [= y_k^T s_k]$$

gelten. Das garantiert die Krümmung in den Wolfe-Bedingungen:

$$\alpha_k [\nabla f_{k+1}^T h^{(k)} - \nabla f_k^T h^{(k)}] > \alpha_k \underbrace{(c_2 - 1)}_{<0} \underbrace{\nabla f_k^T h^{(k)}}_{<0} > 0$$

Für $h^{(k+1)} = -B_{k+1}^{-1} \nabla f_{k+1}$ ist die Inverse $H_{k+1} := B_{k+1}^{-1}$ günstiger.
Die Matrix $H_{k+1} \succ 0$ mit $H_{k+1}(\nabla f_{k+1} - \nabla f_k) = x^{(k+1)} - x^{(k)}$, die sich gegenüber H_k in geeigneter Norm am wenigsten ändert, erhält man durch die Rang-2-Korrektur von Broyden, Fletcher, Goldfarb und Shanno:

$$H_{k+1} = (I - \frac{1}{s_k^T y_k} s_k y_k^T) H_k (I - \frac{1}{s_k^T y_k} y_k s_k^T) + \frac{1}{s_k^T y_k} s_k s_k^T$$

wobei $s_k := x^{(k+1)} - x^{(k)}$, $y_k := \nabla f_{k+1} - \nabla f_k$. [$H_k \succ 0$ und Wolfe $\Rightarrow$ $H_{k+1} \succ 0$, denn $s_k^T H_{k+1} s_k \geq \frac{1}{s_k^T y_k} (s_k^T s_k)^2 > 0$, $y_k^T H_{k+1} y_k = s_k^T y_k > 0$.]

Man startet mit $H_0 = \nabla^2 f_0^{-1}$ (falls $\succ 0$) oder $H_0 = \frac{1}{\|\nabla f_0\|} I$.
Für große n bildet man H_k nicht explizit, sondern speichert nur die letzten $\bar{k}$ Paare (s_k, y_k) für ein festes $\bar{k} \in \mathbb{N}$ (limited memory BFGS).

Man kann zeigen: Für eine streng konvexe quadratische Funktion wird H_k eine immer bessere Approximation von $\nabla^2 f^{-1}$ und Line-Search Verfahren mit BFGS-Richtung und Wolfe-Bedingungen konvergieren superlinear.

Man kann zeigen: Für eine streng konvexe quadratische Funktion wird H_k eine immer bessere Approximation von $\nabla^2 f^{-1}$ und Line-Search Verfahren mit BFGS-Richtung und Wolfe-Bedingungen konvergieren superlinear.

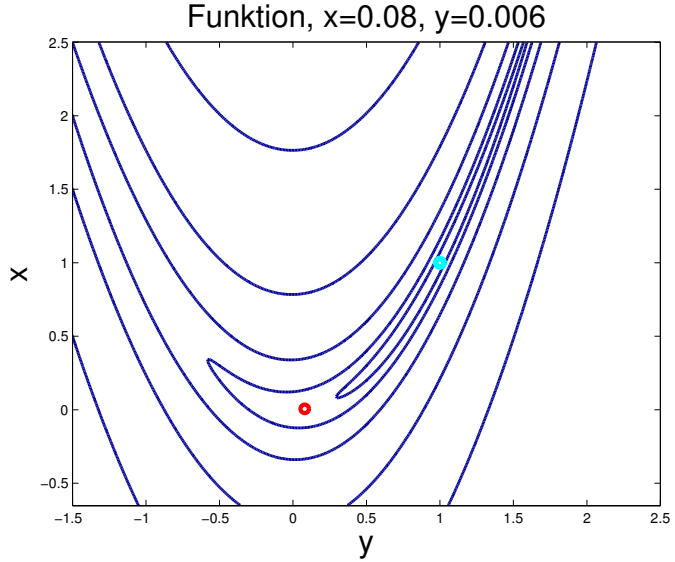
In der Nähe eines x^*, das die hinreichenden Optimalitätsbedingungen erfüllt, ist ein hinreichend glattes f annähernd streng konvex und quadratisch.

Konsequenz: Obwohl sowohl BFGS als auch Steilster Abstieg nur ein Orakel 1. Ordnung benötigen, konvergiert BFGS viel besser als Steilster Abstieg bei etwa vergleichbarem Aufwand pro Iteration.

Inhaltsübersicht für heute:

Trust-Region-Verfahren

In Line-Search-Verf. wird getrennt Schrittrichtung und -länge ermittelt, in Trust-Region-Verfahren beides zugleich nach folgendem Schema:

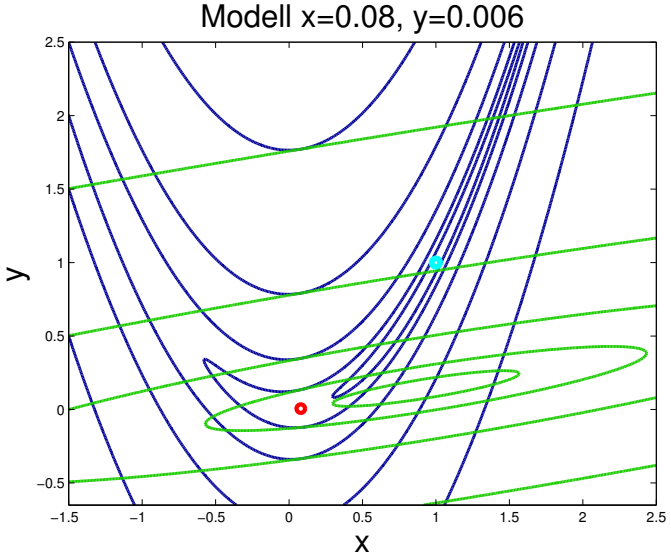


Trust-Region-Verfahren

In Line-Search-Verf. wird getrennt Schrittrichtung und -länge ermittelt, in Trust-Region-Verfahren beides zugleich nach folgendem Schema:

---

0. Wähle ein Model m_0 von f um $x^{(0)}$,
$$m_0(h) = f_0 + \nabla f_0^T h + \frac{1}{2} h^T B_0 h$$
[B_0 z.B. aus Newton oder Quasi-Newton]

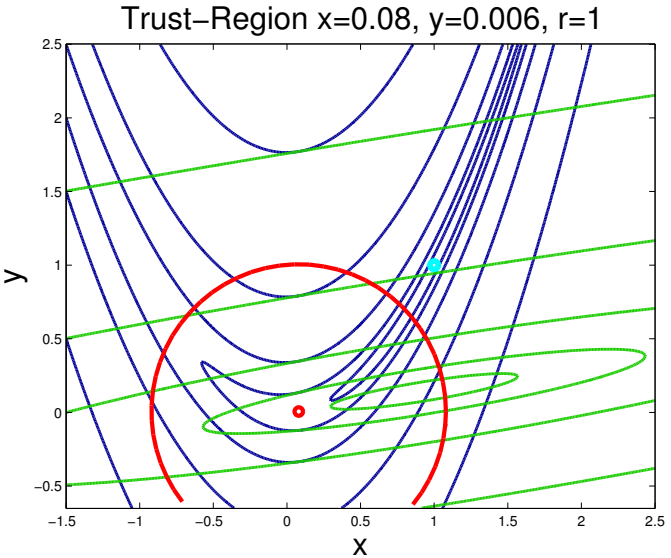


Trust-Region-Verfahren

In Line-Search-Verf. wird getrennt Schrittrichtung und -länge ermittelt, in Trust-Region-Verfahren beides zugleich nach folgendem Schema:

---

0. Wähle ein Model m_0 von f um $x^{(0)}$,
$$m_0(h) = f_0 + \nabla f_0^T h + \frac{1}{2} h^T B_0 h$$
[B_0 z.B. aus Newton oder Quasi-Newton]
und einen Trust-Region-Radius Δ_0 mit
$$m_0(h) \approx f(x^{(0)} + h) \; \forall \|h\| \leq \Delta_0$$

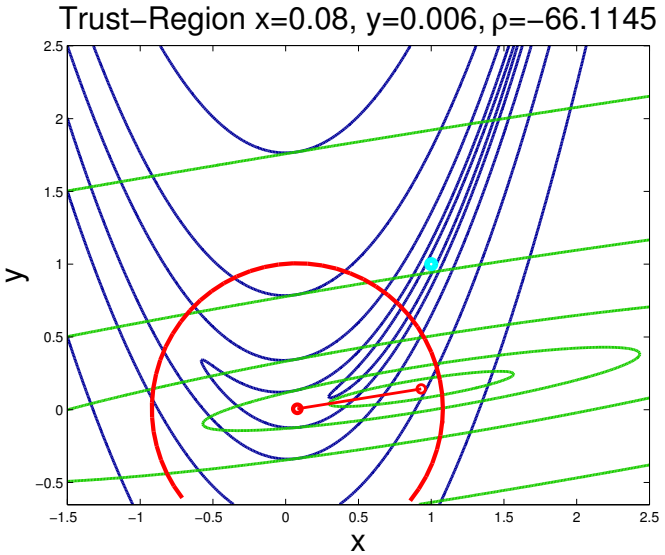


Trust-Region-Verfahren

In Line-Search-Verf. wird getrennt Schrittrichtung und -länge ermittelt, in Trust-Region-Verfahren beides zugleich nach folgendem Schema:

---

0. Wähle ein Model m_0 von f um $x^{(0)}$,
$$m_0(h) = f_0 + \nabla f_0^T h + \frac{1}{2} h^T B_0 h$$
[B_0 z.B. aus Newton oder Quasi-Newton]
und einen Trust-Region-Radius Δ_0 mit
$$m_0(h) \approx f(x^{(0)} + h) \; \forall \|h\| \leq \Delta_0$$
1. Löse das Trust-Region-Unterproblem:
$$h^{(k)} \approx \operatorname*{Argmin}_{\|h\| \leq \Delta_k} m_k(h)$$ [nur annähern]

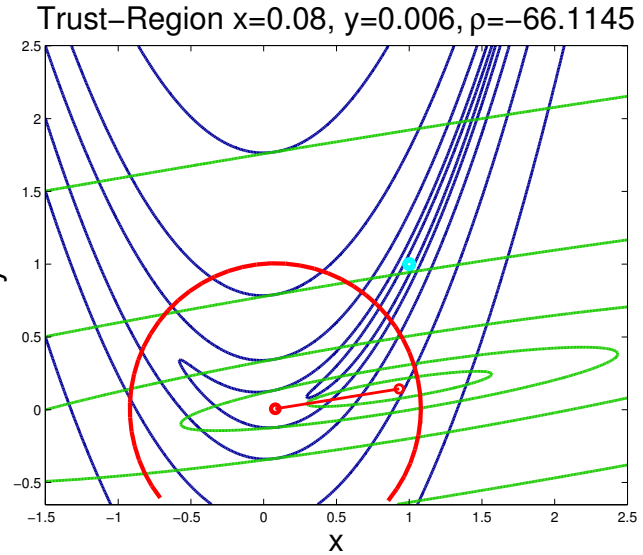


Trust-Region-Verfahren

In Line-Search-Verf. wird getrennt Schrittrichtung und -länge ermittelt, in Trust-Region-Verfahren beides zugleich nach folgendem Schema:

---

0. Wähle ein Model m_0 von f um $x^{(0)}$,
$$m_0(h) = f_0 + \nabla f_0^T h + \tfrac{1}{2} h^T B_0 h$$
[B_0 z.B. aus Newton oder Quasi-Newton]
und einen Trust-Region-Radius Δ_0 mit
$$m_0(h) \approx f(x^{(0)} + h) \; \forall \|h\| \leq \Delta_0$$
1. Löse das Trust-Region-Unterproblem:
$h^{(k)} \approx \underset{\|h\| \leq \Delta_k}{\operatorname{Argmin}} \, m_k(h)$ [nur annähern]



2. Berechne $f(x^{(k)} + h^{(k)})$ und den Fortschrittsfaktor $\rho_k := \frac{f(x^{(k)}) - f(x^{(k)} + h^{(k)})}{m_k(0) - m_k(h^{(k)})}$,

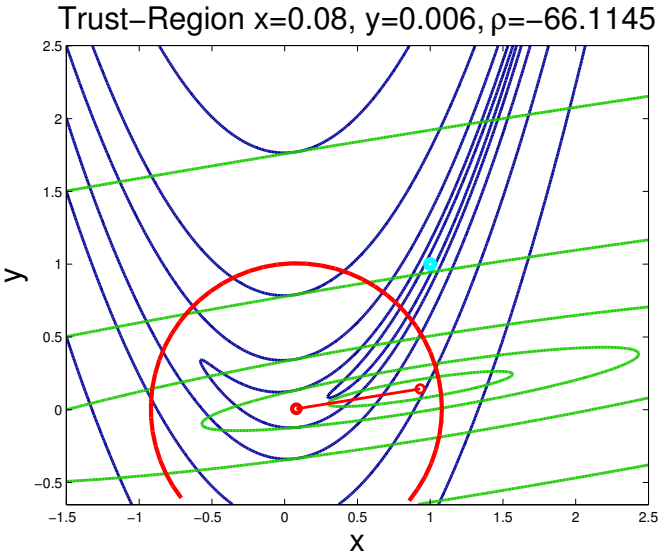
setze $\Delta_{k+1} := \begin{cases} \frac{1}{4}\|h^{(k)}\| & \text{für } \rho_k < \frac{1}{4} \quad [m_k(h^{(k)}) \text{ zu schlecht}] \\ \min\{2\Delta_k, \bar{\Delta}\} & \rho_k > \frac{3}{4} \text{ und } \|h^{(k)}\| = \Delta_k \quad [\Delta \text{ zu klein}] \\ \Delta_k & \text{sonst.} \quad [\text{Wert ok}] \end{cases}$

Trust-Region-Verfahren

In Line-Search-Verf. wird getrennt Schrittrichtung und -länge ermittelt, in Trust-Region-Verfahren beides zugleich nach folgendem Schema:

---

0. Wähle ein Model m_0 von f um $x^{(0)}$,
$$m_0(h) = f_0 + \nabla f_0^T h + \tfrac{1}{2} h^T B_0 h$$
[B_0 z.B. aus Newton oder Quasi-Newton]
und einen Trust-Region-Radius Δ_0 mit
$$m_0(h) \approx f(x^{(0)} + h) \; \forall \|h\| \leq \Delta_0$$
1. Löse das Trust-Region-Unterproblem:
$h^{(k)} \approx \underset{\|h\| \leq \Delta_k}{\operatorname{Argmin}} \, m_k(h)$ [nur annähern]



2. Berechne $f(x^{(k)} + h^{(k)})$ und den Fortschrittsfaktor $\rho_k := \frac{f(x^{(k)}) - f(x^{(k)} + h^{(k)})}{m_k(0) - m_k(h^{(k)})}$,

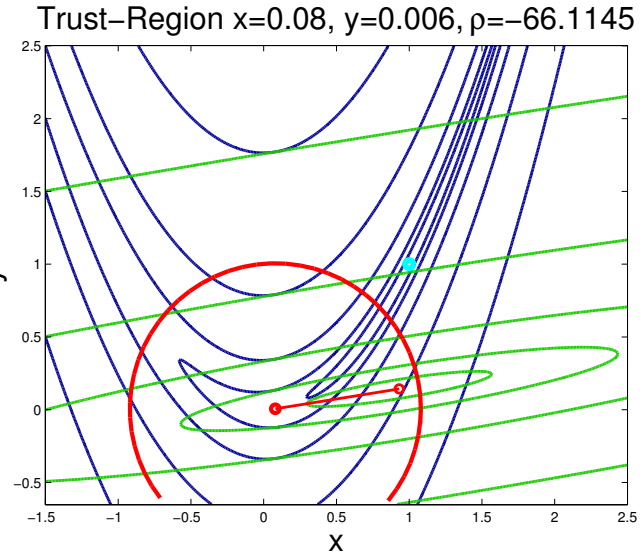
setze $\Delta_{k+1} := \begin{cases} \frac{1}{4}\|h^{(k)}\| & \text{für } \rho_k < \frac{1}{4} \quad [m_k(h^{(k)}) \text{ zu schlecht}] \\ \min\{2\Delta_k, \bar{\Delta}\} & \rho_k > \frac{3}{4} \text{ und } \|h^{(k)}\| = \Delta_k \quad [\Delta \text{ zu klein}] \\ \Delta_k & \text{sonst.} \quad [\text{Wert ok}] \end{cases}$

Trust-Region-Verfahren

In Line-Search-Verf. wird getrennt Schrittrichtung und -länge ermittelt, in Trust-Region-Verfahren beides zugleich nach folgendem Schema:

---

0. Wähle ein Model m_0 von f um $x^{(0)}$,
$$m_0(h) = f_0 + \nabla f_0^T h + \tfrac{1}{2} h^T B_0 h$$
[B_0 z.B. aus Newton oder Quasi-Newton]
und einen Trust-Region-Radius Δ_0 mit
$$m_0(h) \approx f(x^{(0)} + h) \; \forall \|h\| \leq \Delta_0$$



1. Löse das Trust-Region-Unterproblem:
$h^{(k)} \approx \underset{\|h\| \leq \Delta_k}{\operatorname{Argmin}} m_k(h)$ [nur annähern]

2. Berechne $f(x^{(k)} + h^{(k)})$ und den Fortschrittsfaktor $\rho_k := \frac{f(x^{(k)}) - f(x^{(k)} + h^{(k)})}{m_k(0) - m_k(h^{(k)})}$,
setze $\Delta_{k+1} := \begin{cases} \frac{1}{4}\|h^{(k)}\| & \text{für } \rho_k < \frac{1}{4} \quad [m_k(h^{(k)}) \text{ zu schlecht}] \\ \min\{2\Delta_k, \bar{\Delta}\} & \rho_k > \frac{3}{4} \text{ und } \|h^{(k)}\| = \Delta_k \quad [\Delta \text{ zu klein}] \\ \Delta_k & \text{sonst.} \quad [\text{Wert ok}] \end{cases}$

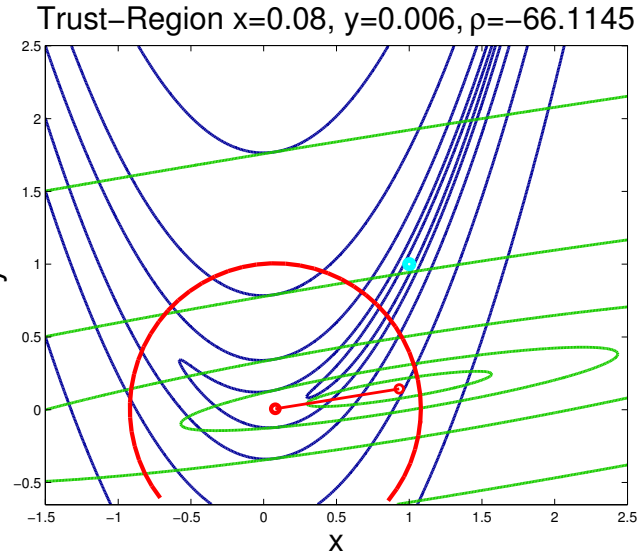
3. Ist $\rho_k > \eta$ für ein festes Akzeptanzniveau $\eta \in [0, 1)$,
setze $x^{(k+1)} := x^{(k)} + h^{(k)}$, $m_{k+1}(h) = f_{k+1} + \nabla f_{k+1}^T h + \frac{1}{2} h^T B_{k+1} h$,
sonst ändere nichs, $x^{(k+1)} = x^{(k)}$, $m_{k+1} := m_k$.

Trust-Region-Verfahren

In Line-Search-Verf. wird getrennt Schrittrichtung und -länge ermittelt, in Trust-Region-Verfahren beides zugleich nach folgendem Schema:

---

0. Wähle ein Model m_0 von f um $x^{(0)}$,
$$m_0(h) = f_0 + \nabla f_0^T h + \tfrac{1}{2} h^T B_0 h$$
[B_0 z.B. aus Newton oder Quasi-Newton]
und einen Trust-Region-Radius Δ_0 mit
$$m_0(h) \approx f(x^{(0)} + h) \; \forall \|h\| \leq \Delta_0$$



1. Löse das Trust-Region-Unterproblem:
$h^{(k)} \approx \underset{\|h\| \leq \Delta_k}{\operatorname{Argmin}} m_k(h)$ [nur annähern]

2. Berechne $f(x^{(k)} + h^{(k)})$ und den Fortschrittsfaktor $\rho_k := \frac{f(x^{(k)}) - f(x^{(k)} + h^{(k)})}{m_k(0) - m_k(h^{(k)})}$,
setze $\Delta_{k+1} := \begin{cases} \frac{1}{4}\|h^{(k)}\| & \text{für } \rho_k < \frac{1}{4} \quad [m_k(h^{(k)}) \text{ zu schlecht}] \\ \min\{2\Delta_k, \bar{\Delta}\} & \rho_k > \frac{3}{4} \text{ und } \|h^{(k)}\| = \Delta_k \quad [\Delta \text{ zu klein}] \\ \Delta_k & \text{sonst.} \quad [\text{Wert ok}] \end{cases}$

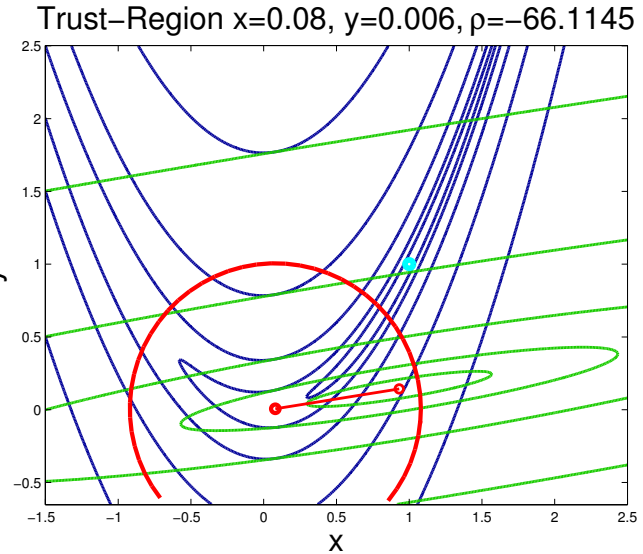
3. Ist $\rho_k > \eta$ für ein festes Akzeptanzniveau $\eta \in [0, 1)$,
setze $x^{(k+1)} := x^{(k)} + h^{(k)}$, $m_{k+1}(h) = f_{k+1} + \nabla f_{k+1}^T h + \frac{1}{2} h^T B_{k+1} h$,
sonst ändere nichs, $x^{(k+1)} = x^{(k)}$, $m_{k+1} := m_k$.

Trust-Region-Verfahren

In Line-Search-Verf. wird getrennt Schrittrichtung und -länge ermittelt, in Trust-Region-Verfahren beides zugleich nach folgendem Schema:

---

0. Wähle ein Model m_0 von f um $x^{(0)}$,
$$m_0(h) = f_0 + \nabla f_0^T h + \tfrac{1}{2} h^T B_0 h$$
[B_0 z.B. aus Newton oder Quasi-Newton]
und einen Trust-Region-Radius Δ_0 mit
$$m_0(h) \approx f(x^{(0)} + h) \;\forall \|h\| \le \Delta_0$$



1. Löse das Trust-Region-Unterproblem:
$h^{(k)} \approx \underset{\|h\| \le \Delta_k}{\operatorname{Argmin}} m_k(h)$ [nur annähern]

2. Berechne $f(x^{(k)} + h^{(k)})$ und den Fortschrittsfaktor $\rho_k := \frac{f(x^{(k)}) - f(x^{(k)} + h^{(k)})}{m_k(0) - m_k(h^{(k)})}$,
setze $\Delta_{k+1} := \begin{cases} \frac{1}{4}\|h^{(k)}\| & \text{für } \rho_k < \frac{1}{4} \quad [m_k(h^{(k)}) \text{ zu schlecht}] \\ \min\{2\Delta_k, \bar{\Delta}\} & \rho_k > \frac{3}{4} \text{ und } \|h^{(k)}\| = \Delta_k \quad [\Delta \text{ zu klein}] \\ \Delta_k & \text{sonst.} \quad [\text{Wert ok}] \end{cases}$

3. Ist $\rho_k > \eta$ für ein festes Akzeptanzniveau $\eta \in [0, 1)$,
setze $x^{(k+1)} := x^{(k)} + h^{(k)}$, $m_{k+1}(h) = f_{k+1} + \nabla f_{k+1}^T h + \frac{1}{2} h^T B_{k+1} h$,
sonst ändere nichs, $x^{(k+1)} = x^{(k)}$, $m_{k+1} := m_k$.

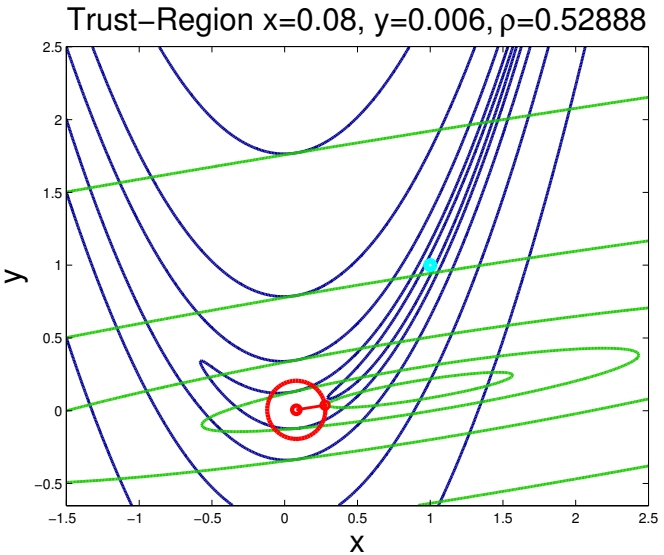
4. $k \leftarrow k + 1$. Falls $\|\nabla f_k\| > \varepsilon$, GOTO 1, sonst STOP.

Trust-Region-Verfahren

In Line-Search-Verf. wird getrennt Schrittrichtung und -länge ermittelt, in Trust-Region-Verfahren beides zugleich nach folgendem Schema:

---

0. Wähle ein Model m_0 von f um $x^{(0)}$,
$$m_0(h) = f_0 + \nabla f_0^T h + \tfrac{1}{2} h^T B_0 h$$
[B_0 z.B. aus Newton oder Quasi-Newton]
und einen Trust-Region-Radius Δ_0 mit
$$m_0(h) \approx f(x^{(0)} + h) \;\forall \|h\| \le \Delta_0$$



1. Löse das Trust-Region-Unterproblem:
$h^{(k)} \approx \underset{\|h\| \le \Delta_k}{\operatorname{Argmin}} m_k(h)$ [nur annähern]

2. Berechne $f(x^{(k)} + h^{(k)})$ und den Fortschrittsfaktor $\rho_k := \frac{f(x^{(k)}) - f(x^{(k)} + h^{(k)})}{m_k(0) - m_k(h^{(k)})}$,
setze $\Delta_{k+1} := \begin{cases} \frac{1}{4}\|h^{(k)}\| & \text{für } \rho_k < \frac{1}{4} \quad [m_k(h^{(k)}) \text{ zu schlecht}] \\ \min\{2\Delta_k, \bar{\Delta}\} & \rho_k > \frac{3}{4} \text{ und } \|h^{(k)}\| = \Delta_k \quad [\Delta \text{ zu klein}] \\ \Delta_k & \text{sonst.} \quad [\text{Wert ok}] \end{cases}$

3. Ist $\rho_k > \eta$ für ein festes Akzeptanzniveau $\eta \in [0, 1)$,
setze $x^{(k+1)} := x^{(k)} + h^{(k)}$, $m_{k+1}(h) = f_{k+1} + \nabla f_{k+1}^T h + \frac{1}{2} h^T B_{k+1} h$,
sonst ändere nichs, $x^{(k+1)} = x^{(k)}$, $m_{k+1} := m_k$.

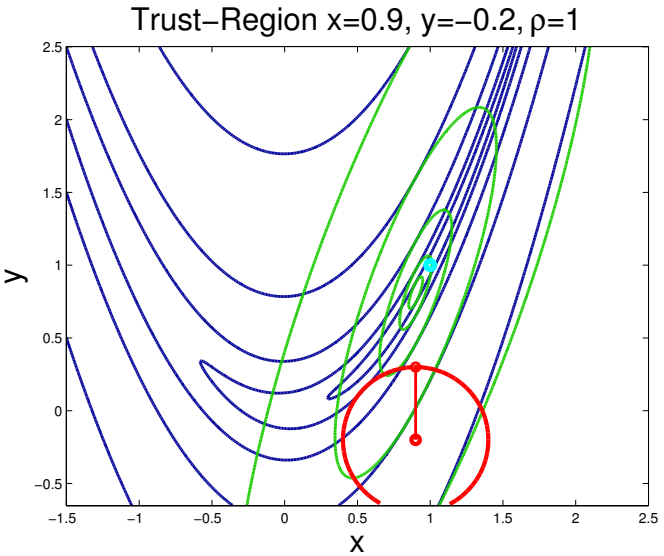
4. $k \leftarrow k + 1$. Falls $\|\nabla f_k\| > \varepsilon$, GOTO 1, sonst STOP.

Trust-Region-Verfahren

In Line-Search-Verf. wird getrennt Schrittrichtung und -länge ermittelt, in Trust-Region-Verfahren beides zugleich nach folgendem Schema:

---

0. Wähle ein Model m_0 von f um $x^{(0)}$,
$$m_0(h) = f_0 + \nabla f_0^T h + \tfrac{1}{2} h^T B_0 h$$
[B_0 z.B. aus Newton oder Quasi-Newton]
und einen Trust-Region-Radius Δ_0 mit
$$m_0(h) \approx f(x^{(0)} + h) \;\forall \|h\| \le \Delta_0$$



1. Löse das Trust-Region-Unterproblem:
$h^{(k)} \approx \underset{\|h\| \le \Delta_k}{\operatorname{Argmin}} m_k(h)$ [nur annähern]

2. Berechne $f(x^{(k)} + h^{(k)})$ und den Fortschrittsfaktor $\rho_k := \frac{f(x^{(k)}) - f(x^{(k)} + h^{(k)})}{m_k(0) - m_k(h^{(k)})}$,
setze $\Delta_{k+1} := \begin{cases} \frac{1}{4}\|h^{(k)}\| & \text{für } \rho_k < \frac{1}{4} \quad [m_k(h^{(k)}) \text{ zu schlecht}] \\ \min\{2\Delta_k, \bar{\Delta}\} & \rho_k > \frac{3}{4} \text{ und } \|h^{(k)}\| = \Delta_k \quad [\Delta \text{ zu klein}] \\ \Delta_k & \text{sonst.} \quad [\text{Wert ok}] \end{cases}$

3. Ist $\rho_k > \eta$ für ein festes Akzeptanzniveau $\eta \in [0, 1)$,
setze $x^{(k+1)} := x^{(k)} + h^{(k)}$, $m_{k+1}(h) = f_{k+1} + \nabla f_{k+1}^T h + \frac{1}{2} h^T B_{k+1} h$,
sonst ändere nichs, $x^{(k+1)} = x^{(k)}$, $m_{k+1} := m_k$.

4. $k \leftarrow k + 1$. Falls $\|\nabla f_k\| > \varepsilon$, GOTO 1, sonst STOP.

Konvergenz von Trust-Region-Verfahren

Weil m_k in 0 auch den Gradienten ∇f_k hat, gilt $\rho_k > \eta$ für Δ klein genug.

Konvergenz von Trust-Region-Verfahren

Weil m_k in 0 auch den Gradienten ∇f_k hat, gilt $\rho_k > \eta$ für Δ klein genug.
Globale Konvergenz ist (unter milden technischen Voraussetzungen) garantiert, wenn für eine Konstante $c_1 \in (0, 1)$ und für jedes k die Näherungslösung $h^{(k)}$ die Trust-Region-Bedingung

$$\text{(TRC)} \qquad m_k(0) - m_k(h^{(k)}) \geq c_1 \|\nabla f_k\| \min\left\{\Delta_k, \frac{\|\nabla f_k\|}{\|B_k\|}\right\}$$

erfüllt. [$\hat{=}$ red. Abstieg mal Schrittlänge, stellt *sufficient decrease* sicher]

Konvergenz von Trust-Region-Verfahren

Weil m_k in 0 auch den Gradienten ∇f_k hat, gilt $\rho_k > \eta$ für Δ klein genug.
Globale Konvergenz ist (unter milden technischen Voraussetzungen) garantiert, wenn für eine Konstante $c_1 \in (0,1)$ und für jedes k die Näherungslösung $h^{(k)}$ die Trust-Region-Bedingung

$$\text{(TRC)} \qquad m_k(0) - m_k(h^{(k)}) \geq c_1 \|\nabla f_k\| \min\left\{\Delta_k, \frac{\|\nabla f_k\|}{\|B_k\|}\right\}$$

erfüllt. [$\hat{=}$ red. Abstieg mal Schrittlänge, stellt *sufficient decrease* sicher]

Diese Bedingung erfüllt der **Cauchy-Punkt** $h_C^{(k)}$ für $c_1 = \frac{1}{2}$. Er minimiert das Modell in Richtung des steilsten Abstiegs: Löse für $h = -\frac{\nabla f_k}{\|\nabla f_k\|}$

$$\min_{\alpha \in [0,\Delta_k]} m_k(\alpha h) = f_k + \alpha \nabla f_k^T h + \frac{\alpha^2}{2} h^T B_k h = f_k - \alpha \|\nabla f_k\| + \frac{\alpha^2}{2\|\nabla f_k\|^2} \nabla f_k^T B_k \nabla f_k$$

Konvergenz von Trust-Region-Verfahren

Weil m_k in 0 auch den Gradienten ∇f_k hat, gilt $\rho_k > \eta$ für Δ klein genug.
Globale Konvergenz ist (unter milden technischen Voraussetzungen) garantiert, wenn für eine Konstante $c_1 \in (0,1)$ und für jedes k die Näherungslösung $h^{(k)}$ die Trust-Region-Bedingung

$$\text{(TRC)} \qquad m_k(0) - m_k(h^{(k)}) \geq c_1 \|\nabla f_k\| \min\left\{\Delta_k, \frac{\|\nabla f_k\|}{\|B_k\|}\right\}$$

erfüllt. [$\hat{=}$ red. Abstieg mal Schrittlänge, stellt *sufficient decrease* sicher]

Diese Bedingung erfüllt der **Cauchy-Punkt** $h_C^{(k)}$ für $c_1 = \frac{1}{2}$. Er minimiert das Modell in Richtung des steilsten Abstiegs: Löse für $h = -\frac{\nabla f_k}{\|\nabla f_k\|}$

$$\min_{\alpha \in [0,\Delta_k]} m_k(\alpha h) = f_k + \alpha \nabla f_k^T h + \frac{\alpha^2}{2} h^T B_k h = f_k - \alpha \|\nabla f_k\| + \frac{\alpha^2}{2\|\nabla f_k\|^2} \nabla f_k^T B_k \nabla f_k$$

$$\Rightarrow \alpha_k^C = \begin{cases} \Delta_k & \text{falls } \nabla f_k^T B_k \nabla f_k \leq 0, \\ \min\{\Delta_k, \frac{\|\nabla f_k\|^3}{\nabla f_k^T B_k \nabla f_k}\} & \text{sonst.} \end{cases} \qquad \left[-\|\nabla f_k\| + \alpha \frac{\nabla f_k^T B_k \nabla f_k}{\|\nabla f_k\|^2} = 0\right]$$

$$\boxed{h_C^{(k)} = -\alpha_k^C \frac{\nabla f_k}{\|\nabla f_k\|}}$$

Konvergenz von Trust-Region-Verfahren

Weil m_k in 0 auch den Gradienten ∇f_k hat, gilt $\rho_k > \eta$ für Δ klein genug. Globale Konvergenz ist (unter milden technischen Voraussetzungen) garantiert, wenn für eine Konstante $c_1 \in (0,1)$ und für jedes k die Näherungslösung $h^{(k)}$ die Trust-Region-Bedingung

$$\text{(TRC)} \qquad m_k(0) - m_k(h^{(k)}) \geq c_1 \|\nabla f_k\| \min\left\{\Delta_k, \frac{\|\nabla f_k\|}{\|B_k\|}\right\}$$

erfüllt. [$\hat{=}$ red. Abstieg mal Schrittlänge, stellt *sufficient decrease* sicher]

Diese Bedingung erfüllt der **Cauchy-Punkt** $h_C^{(k)}$ für $c_1 = \frac{1}{2}$. Er minimiert das Modell in Richtung des steilsten Abstiegs: Löse für $h = -\frac{\nabla f_k}{\|\nabla f_k\|}$

$$\min_{\alpha \in [0,\Delta_k]} m_k(\alpha h) = f_k + \alpha \nabla f_k^T h + \frac{\alpha^2}{2} h^T B_k h = f_k - \alpha \|\nabla f_k\| + \frac{\alpha^2}{2\|\nabla f_k\|^2} \nabla f_k^T B_k \nabla f_k$$

$$\Rightarrow \alpha_k^C = \begin{cases} \Delta_k & \text{falls } \nabla f_k^T B_k \nabla f_k \leq 0, \\ \min\{\Delta_k, \frac{\|\nabla f_k\|^3}{\nabla f_k^T B_k \nabla f_k}\} & \text{sonst.} \end{cases} \qquad \left[-\|\nabla f_k\| + \alpha \frac{\nabla f_k^T B_k \nabla f_k}{\|\nabla f_k\|^2} = 0\right]$$

$$\boxed{h_C^{(k)} = -\alpha_k^C \frac{\nabla f_k}{\|\nabla f_k\|}}$$

Jede Verbesserung gegenüber $h_C^{(k)}$ erfüllt ebenso (TRC) für $c_1 = \frac{1}{2}$ und ist global konvergent.

Wie berechnet man ∇f ($\nabla^2 f$) für das Orakel 1. (2.) Ordnung?

Wie berechnet man ∇f ($\nabla^2 f$) für das Orakel 1. (2.) Ordnung?
Falls f analytisch vorliegt:
Selber differenzieren oder Matlab, Maple, ... nutzen.

Wie berechnet man ∇f ($\nabla^2 f$) für das Orakel 1. (2.) Ordnung?
Falls f analytisch vorliegt:
Selber differenzieren oder Matlab, Maple, ... nutzen.

Falls f nur als Unterprogramm gegeben ist oder symbolisches Differenzieren fehlschlägt:

- Numerisches Differenzieren
- Automatisches Differenzieren (falls Source-Code verfügbar)

(Für beides gibt es kommerzielle und frei verfügbare Software)

Inhaltsübersicht für heute:

Numerisches Differenzieren

Für die Berechnung von $\nabla f(\bar{x})$ werden die partiellen Ableitungen durch endliche Differenzen angenähert [Vorsicht: Stellenauslöschung!]:

$$\frac{\partial f}{\partial x_i}(\bar{x}) \approx \frac{f(\bar{x} + \varepsilon e_i) - f(\bar{x} - \varepsilon e_i)}{2\varepsilon}$$

Numerisches Differenzieren

Für die Berechnung von $\nabla f(\bar{x})$ werden die partiellen Ableitungen durch endliche Differenzen angenähert [Vorsicht: Stellenauslöschung!]:

$$\frac{\partial f}{\partial x_i}(\bar{x}) \approx \frac{f(\bar{x} + \varepsilon e_i) - f(\bar{x} - \varepsilon e_i)}{2\varepsilon}$$

Die Numerik lehrt: Ist u das Maschinenepsilon (größte Fließkomma-Zahl mit float$(1 + u)$=float(1)), liefert $\varepsilon = u^{\frac{2}{3}}$ das beste Ergebnis mit einem Fehler von Konstante mal ε^2.

Numerisches Differenzieren

Für die Berechnung von $\nabla f(\bar{x})$ werden die partiellen Ableitungen durch endliche Differenzen angenähert [Vorsicht: Stellenauslöschung!]:

$$\frac{\partial f}{\partial x_i}(\bar{x}) \approx \frac{f(\bar{x} + \varepsilon e_i) - f(\bar{x} - \varepsilon e_i)}{2\varepsilon}$$

Die Numerik lehrt: Ist u das Maschinenepsilon (größte Fließkomma-Zahl mit float$(1 + u)$=float(1)), liefert $\varepsilon = u^{\frac{2}{3}}$ das beste Ergebnis mit einem Fehler von Konstante mal ε^2.

Pro Koordinate sind so zwei Funktionsberechnungen erforderlich, aber der Fehler der einseitigen Formel $\frac{f(\bar{x}+\varepsilon e_i)-f(\bar{x})}{\varepsilon}$ wird für $\varepsilon = \sqrt{u}$ minimiert und wäre nur durch Konstante mal ε beschränkt, ist also meist zu groß.

Numerisches Differenzieren

Für die Berechnung von $\nabla f(\bar{x})$ werden die partiellen Ableitungen durch endliche Differenzen angenähert [Vorsicht: Stellenauslöschung!]:

$$\frac{\partial f}{\partial x_i}(\bar{x}) \approx \frac{f(\bar{x} + \varepsilon e_i) - f(\bar{x} - \varepsilon e_i)}{2\varepsilon}$$

Die Numerik lehrt: Ist u das Maschinenepsilon (größte Fließkomma-Zahl mit float$(1 + u)$=float(1)), liefert $\varepsilon = u^{\frac{2}{3}}$ das beste Ergebnis mit einem Fehler von Konstante mal ε^2.

Pro Koordinate sind so zwei Funktionsberechnungen erforderlich, aber der Fehler der einseitigen Formel $\frac{f(\bar{x}+\varepsilon e_i)-f(\bar{x})}{\varepsilon}$ wird für $\varepsilon = \sqrt{u}$ minimiert und wäre nur durch Konstante mal ε beschränkt, ist also meist zu groß.

Bei der Berechung von $\nabla^2 f$ oder der Jacobimatrix J_g einer Funktion $g : \mathbb{R}^n \to \mathbb{R}^m$ rentiert es sich zu untersuchen, ob eine eventuelle Dünnbesetztheit der Matrizen genutzt werden kann, um die Anzahl der Funktionsauswertungen zu reduzieren. Dies wird hier nicht weiter untersucht.

Inhaltsübersicht für heute:

Automatisches Differenzieren

Die Kettenregel für $g(y(x))$ mit $g : \mathbb{R}^m \to \mathbb{R}^k$, $y : \mathbb{R}^n \to \mathbb{R}^m$,

$$\nabla_x g(y(x)) = \sum_{i=1}^{m} \frac{\partial g}{\partial y_i}(y) \nabla_x y_i(x)$$

wird konsequent rekursiv auf alle Teilausdrücke angewandt.

Automatisches Differenzieren

Die Kettenregel für $g(y(x))$ mit $g : \mathbb{R}^m \to \mathbb{R}^k$, $y : \mathbb{R}^n \to \mathbb{R}^m$,

$$\nabla_x g(y(x)) = \sum_{i=1}^{m} \frac{\partial g}{\partial y_i}(y) \nabla_x y_i(x)$$

wird konsequent rekursiv auf alle Teilausdrücke angewandt.

Bsp: $f(x) = (x_1 x_2 + e^{x_1 x_2})/x_3$
Beginne mit $\nabla_x x_1 = e_1$, $\nabla_x x_2 = e_2$, $\nabla_x x_3 = e_3$,

$x_4 := x_1 x_2$

$x_5 := e^{x_4}$

$x_6 := x_4 + x_5$

$x_7 := x_6 / x_3$

Automatisches Differenzieren

Die Kettenregel für $g(y(x))$ mit $g : \mathbb{R}^m \to \mathbb{R}^k$, $y : \mathbb{R}^n \to \mathbb{R}^m$,

$$\nabla_x g(y(x)) = \sum_{i=1}^{m} \frac{\partial g}{\partial y_i}(y) \nabla_x y_i(x)$$

wird konsequent rekursiv auf alle Teilausdrücke angewandt.

Bsp: $f(x) = (x_1 x_2 + e^{x_1 x_2})/x_3$

Beginne mit $\nabla_x x_1 = e_1$, $\nabla_x x_2 = e_2$, $\nabla_x x_3 = e_3$,

$x_4 := x_1 x_2 \quad \to \nabla_x x_4 = \frac{\partial x_1 x_2}{\partial x_1} \nabla_x x_1 + \frac{\partial x_1 x_2}{\partial x_2} \nabla_x x_2 = x_2 \nabla_x x_1 + x_1 \nabla_x x_2 = \begin{bmatrix} x_2 \\ x_1 \\ 0 \end{bmatrix}$

$x_5 := e^{x_4}$

$x_6 := x_4 + x_5$

$x_7 := x_6/x_3$

Automatisches Differenzieren

Die Kettenregel für $g(y(x))$ mit $g : \mathbb{R}^m \to \mathbb{R}^k$, $y : \mathbb{R}^n \to \mathbb{R}^m$,

$$\nabla_x g(y(x)) = \sum_{i=1}^{m} \frac{\partial g}{\partial y_i}(y) \nabla_x y_i(x)$$

wird konsequent rekursiv auf alle Teilausdrücke angewandt.

Bsp: $f(x) = (x_1 x_2 + e^{x_1 x_2})/x_3$

Beginne mit $\nabla_x x_1 = e_1$, $\nabla_x x_2 = e_2$, $\nabla_x x_3 = e_3$,

$x_4 := x_1 x_2 \quad \to \nabla_x x_4 = \frac{\partial x_1 x_2}{\partial x_1} \nabla_x x_1 + \frac{\partial x_1 x_2}{\partial x_2} \nabla_x x_2 = x_2 \nabla_x x_1 + x_1 \nabla_x x_2 = \begin{bmatrix} x_2 \\ x_1 \\ 0 \end{bmatrix}$

$x_5 := e^{x_4} \quad \to \nabla_x x_5 = \frac{\partial e^{x_4}}{\partial x_4} \nabla_x x_4 = e^{x_4} \nabla_x x_4$

$x_6 := x_4 + x_5$

$x_7 := x_6/x_3$

Automatisches Differenzieren

Die Kettenregel für $g(y(x))$ mit $g : \mathbb{R}^m \to \mathbb{R}^k$, $y : \mathbb{R}^n \to \mathbb{R}^m$,

$$\nabla_x g(y(x)) = \sum_{i=1}^{m} \frac{\partial g}{\partial y_i}(y) \nabla_x y_i(x)$$

wird konsequent rekursiv auf alle Teilausdrücke angewandt.

Bsp: $f(x) = (x_1 x_2 + e^{x_1 x_2})/x_3$

Beginne mit $\nabla_x x_1 = e_1$, $\nabla_x x_2 = e_2$, $\nabla_x x_3 = e_3$,

$$x_4 := x_1 x_2 \quad \to \nabla_x x_4 = \frac{\partial x_1 x_2}{\partial x_1} \nabla_x x_1 + \frac{\partial x_1 x_2}{\partial x_2} \nabla_x x_2 = x_2 \nabla_x x_1 + x_1 \nabla_x x_2 = \begin{bmatrix} x_2 \\ x_1 \\ 0 \end{bmatrix}$$

$$x_5 := e^{x_4} \quad \to \nabla_x x_5 = \frac{\partial e^{x_4}}{\partial x_4} \nabla_x x_4 = e^{x_4} \nabla_x x_4$$

$$x_6 := x_4 + x_5 \to \nabla_x x_6 = \frac{\partial x_4 + x_5}{\partial x_4} \nabla_x x_4 + \frac{\partial x_4 + x_5}{\partial x_5} \nabla_x x_5 = \nabla_x x_4 + \nabla_x x_5$$

$$x_7 := x_6/x_3$$

Automatisches Differenzieren

Die Kettenregel für $g(y(x))$ mit $g : \mathbb{R}^m \to \mathbb{R}^k$, $y : \mathbb{R}^n \to \mathbb{R}^m$,

$$\nabla_x g(y(x)) = \sum_{i=1}^{m} \frac{\partial g}{\partial y_i}(y) \nabla_x y_i(x)$$

wird konsequent rekursiv auf alle Teilausdrücke angewandt.

Bsp: $f(x) = (x_1 x_2 + e^{x_1 x_2})/x_3$

Beginne mit $\nabla_x x_1 = e_1$, $\nabla_x x_2 = e_2$, $\nabla_x x_3 = e_3$,

$$x_4 := x_1 x_2 \quad \to \nabla_x x_4 = \frac{\partial x_1 x_2}{\partial x_1} \nabla_x x_1 + \frac{\partial x_1 x_2}{\partial x_2} \nabla_x x_2 = x_2 \nabla_x x_1 + x_1 \nabla_x x_2 = \begin{bmatrix} x_2 \\ x_1 \\ 0 \end{bmatrix}$$

$$x_5 := e^{x_4} \quad \to \nabla_x x_5 = \frac{\partial e^{x_4}}{\partial x_4} \nabla_x x_4 = e^{x_4} \nabla_x x_4$$

$$x_6 := x_4 + x_5 \to \nabla_x x_6 = \frac{\partial x_4 + x_5}{\partial x_4} \nabla_x x_4 + \frac{\partial x_4 + x_5}{\partial x_5} \nabla_x x_5 = \nabla_x x_4 + \nabla_x x_5$$

$$x_7 := x_6/x_3 \to \nabla_x x_7 = \frac{\partial x_6/x_3}{\partial x_6} \nabla_x x_6 + \frac{\partial x_6/x_3}{\partial x_3} \nabla_x x_3 = x_3^{-1} \nabla_x x_6 - x_6 x_3^{-2} \nabla_x x_3$$

Automatisches Differenzieren

Die Kettenregel für $g(y(x))$ mit $g : \mathbb{R}^m \to \mathbb{R}^k$, $y : \mathbb{R}^n \to \mathbb{R}^m$,

$$\nabla_x g(y(x)) = \sum_{i=1}^{m} \frac{\partial g}{\partial y_i}(y) \nabla_x y_i(x)$$

wird konsequent rekursiv auf alle Teilausdrücke angewandt.

Bsp: $f(x) = (x_1 x_2 + e^{x_1 x_2})/x_3$

Beginne mit $\nabla_x x_1 = e_1$, $\nabla_x x_2 = e_2$, $\nabla_x x_3 = e_3$,

$$x_4 := x_1 x_2 \quad \to \nabla_x x_4 = \frac{\partial x_1 x_2}{\partial x_1} \nabla_x x_1 + \frac{\partial x_1 x_2}{\partial x_2} \nabla_x x_2 = x_2 \nabla_x x_1 + x_1 \nabla_x x_2 = \begin{bmatrix} x_2 \\ x_1 \\ 0 \end{bmatrix}$$

$$x_5 := e^{x_4} \quad \to \nabla_x x_5 = \frac{\partial e^{x_4}}{\partial x_4} \nabla_x x_4 = e^{x_4} \nabla_x x_4$$

$$x_6 := x_4 + x_5 \to \nabla_x x_6 = \frac{\partial x_4 + x_5}{\partial x_4} \nabla_x x_4 + \frac{\partial x_4 + x_5}{\partial x_5} \nabla_x x_5 = \nabla_x x_4 + \nabla_x x_5$$

$$x_7 := x_6/x_3 \to \nabla_x x_7 = \frac{\partial x_6/x_3}{\partial x_6} \nabla_x x_6 + \frac{\partial x_6/x_3}{\partial x_3} \nabla_x x_3 = x_3^{-1} \nabla_x x_6 - x_6 x_3^{-2} \nabla_x x_3$$

Beachte:
Ein Compiler zerlegt die Berechnung von f ebenso in elementare Schritte mit Zwischenvariablen x_i, am Ende ist $f = x_7$ und $\nabla_x f = \nabla_x x_7$.

Automatisches Differenzieren

Die Kettenregel für $g(y(x))$ mit $g : \mathbb{R}^m \to \mathbb{R}^k$, $y : \mathbb{R}^n \to \mathbb{R}^m$,

$$\nabla_x g(y(x)) = \sum_{i=1}^{m} \frac{\partial g}{\partial y_i}(y) \nabla_x y_i(x)$$

wird konsequent rekursiv auf alle Teilausdrücke angewandt.

Bsp: $f(x) = (x_1 x_2 + e^{x_1 x_2})/x_3$

Beginne mit $\nabla_x x_1 = e_1$, $\nabla_x x_2 = e_2$, $\nabla_x x_3 = e_3$,

$$x_4 := x_1 x_2 \quad \to \nabla_x x_4 = \frac{\partial x_1 x_2}{\partial x_1} \nabla_x x_1 + \frac{\partial x_1 x_2}{\partial x_2} \nabla_x x_2 = x_2 \nabla_x x_1 + x_1 \nabla_x x_2 = \begin{bmatrix} x_2 \\ x_1 \\ 0 \end{bmatrix}$$

$$x_5 := e^{x_4} \quad \to \nabla_x x_5 = \frac{\partial e^{x_4}}{\partial x_4} \nabla_x x_4 = e^{x_4} \nabla_x x_4$$

$$x_6 := x_4 + x_5 \to \nabla_x x_6 = \frac{\partial x_4 + x_5}{\partial x_4} \nabla_x x_4 + \frac{\partial x_4 + x_5}{\partial x_5} \nabla_x x_5 = \nabla_x x_4 + \nabla_x x_5$$

$$x_7 := x_6/x_3 \to \nabla_x x_7 = \frac{\partial x_6/x_3}{\partial x_6} \nabla_x x_6 + \frac{\partial x_6/x_3}{\partial x_3} \nabla_x x_3 = x_3^{-1} \nabla_x x_6 - x_6 x_3^{-2} \nabla_x x_3$$

Beachte:
Ein Compiler zerlegt die Berechnung von f ebenso in elementare Schritte mit Zwischenvariablen x_i, am Ende ist $f = x_7$ und $\nabla_x f = \nabla_x x_7$.
Für jeden elementaren Schritt eines Programms können die partiellen Ableitungen explizit angegeben werden! Die Gradienten aufzusummieren ist aber ineffizient und besser organisierbar.

Automatische Vorwärts-Berechnung von $\nabla f(x)^T h$

Gleichzeitig mit f kann für ein gegebenes $h \in \mathbb{R}^n$ die Richtungsableitung $D_h f(x) := \nabla f(x)^T h$ berechnet werden. Für D_h lautet die Kettenregel

$$D_h g(y(x)) = \nabla_x g(y(x))^T h = \sum_{i=1}^{m} \frac{\partial g}{\partial y_i}(y) \nabla_x y_i(x)^T h = \sum_{i=1}^{m} \frac{\partial g}{\partial y_i}(y) D_h y_i(x)$$

Automatische Vorwärts-Berechnung von $\nabla f(x)^T h$

Gleichzeitig mit f kann für ein gegebenes $h \in \mathbb{R}^n$ die Richtungsableitung $D_h f(x) := \nabla f(x)^T h$ berechnet werden. Für D_h lautet die Kettenregel

$$D_h g(y(x)) = \nabla_x g(y(x))^T h = \sum_{i=1}^{m} \frac{\partial g}{\partial y_i}(y) \nabla_x y_i(x)^T h = \sum_{i=1}^{m} \frac{\partial g}{\partial y_i}(y) D_h y_i(x)$$

Am Bsp: $f(x) = (x_1 x_2 + e^{x_1 x_2})/x_3$, D_h ... Richtungsableitung für h

Automatische Vorwärts-Berechnung von $\nabla f(x)^T h$

Gleichzeitig mit f kann für ein gegebenes $h \in \mathbb{R}^n$ die Richtungsableitung $D_h f(x) := \nabla f(x)^T h$ berechnet werden. Für D_h lautet die Kettenregel

$$D_h g(y(x)) = \nabla_x g(y(x))^T h = \sum_{i=1}^{m} \frac{\partial g}{\partial y_i}(y) \nabla_x y_i(x)^T h = \sum_{i=1}^{m} \frac{\partial g}{\partial y_i}(y) D_h y_i(x)$$

Am Bsp: $f(x) = (x_1 x_2 + e^{x_1 x_2})/x_3$, D_h ... Richtungsableitung für h

Beginne mit $D_h x_1 := \nabla_x x_1^T h = e_1^T h = h_1$, $D_h x_2 = \nabla_x x_2^T h = h_2$, $D_h x_3 = h_3$,

$x_4 := x_1 x_2$

$x_5 := e^{x_4}$

$x_6 := x_4 + x_5$

$x_7 := x_6 / x_3$

Automatische Vorwärts-Berechnung von $\nabla f(x)^T h$

Gleichzeitig mit f kann für ein gegebenes $h \in \mathbb{R}^n$ die Richtungsableitung $D_h f(x) := \nabla f(x)^T h$ berechnet werden. Für D_h lautet die Kettenregel

$$D_h g(y(x)) = \nabla_x g(y(x))^T h = \sum_{i=1}^{m} \frac{\partial g}{\partial y_i}(y) \nabla_x y_i(x)^T h = \sum_{i=1}^{m} \frac{\partial g}{\partial y_i}(y) D_h y_i(x)$$

Am Bsp: $f(x) = (x_1 x_2 + e^{x_1 x_2})/x_3$, D_h ... Richtungsableitung für h

Beginne mit $D_h x_1 := \nabla_x x_1^T h = e_1^T h = h_1$, $D_h x_2 = \nabla_x x_2^T h = h_2$, $D_h x_3 = h_3$,

$x_4 := x_1 x_2 \quad \rightarrow D_h x_4 = x_2 \nabla_x x_1^T h + x_1 \nabla_x x_2^T h = x_2 D_h x_1 + x_1 D_h x_2 = x_2 h_1 + x_1 h_2$

$x_5 := e^{x_4}$

$x_6 := x_4 + x_5$

$x_7 := x_6 / x_3$

Automatische Vorwärts-Berechnung von $\nabla f(x)^T h$

Gleichzeitig mit f kann für ein gegebenes $h \in \mathbb{R}^n$ die Richtungsableitung $D_h f(x) := \nabla f(x)^T h$ berechnet werden. Für D_h lautet die Kettenregel

$$D_h g(y(x)) = \nabla_x g(y(x))^T h = \sum_{i=1}^{m} \frac{\partial g}{\partial y_i}(y) \nabla_x y_i(x)^T h = \sum_{i=1}^{m} \frac{\partial g}{\partial y_i}(y) D_h y_i(x)$$

Am Bsp: $f(x) = (x_1 x_2 + e^{x_1 x_2})/x_3$, $\quad D_h$ ... Richtungsableitung für h

Beginne mit $D_h x_1 := \nabla_x x_1^T h = e_1^T h = h_1$, $D_h x_2 = \nabla_x x_2^T h = h_2$, $D_h x_3 = h_3$,

$x_4 := x_1 x_2 \quad \rightarrow D_h x_4 = x_2 \nabla_x x_1^T h + x_1 \nabla_x x_2^T h = x_2 D_h x_1 + x_1 D_h x_2 = x_2 h_1 + x_1 h_2$

$x_5 := e^{x_4} \quad \rightarrow D_h x_5 = e^{x_4} \nabla_x x_4^T h = e^{x_4} D_h x_4$

$x_6 := x_4 + x_5$

$x_7 := x_6 / x_3$

Automatische Vorwärts-Berechnung von $\nabla f(x)^T h$

Gleichzeitig mit f kann für ein gegebenes $h \in \mathbb{R}^n$ die Richtungsableitung $D_h f(x) := \nabla f(x)^T h$ berechnet werden. Für D_h lautet die Kettenregel

$$D_h g(y(x)) = \nabla_x g(y(x))^T h = \sum_{i=1}^{m} \frac{\partial g}{\partial y_i}(y) \nabla_x y_i(x)^T h = \sum_{i=1}^{m} \frac{\partial g}{\partial y_i}(y) D_h y_i(x)$$

Am Bsp: $f(x) = (x_1 x_2 + e^{x_1 x_2})/x_3$, $\quad D_h$ ... Richtungsableitung für h

Beginne mit $D_h x_1 := \nabla_x x_1^T h = e_1^T h = h_1$, $D_h x_2 = \nabla_x x_2^T h = h_2$, $D_h x_3 = h_3$,

$x_4 := x_1 x_2 \quad \to D_h x_4 = x_2 \nabla_x x_1^T h + x_1 \nabla_x x_2^T h = x_2 D_h x_1 + x_1 D_h x_2 = x_2 h_1 + x_1 h_2$

$x_5 := e^{x_4} \quad \to D_h x_5 = e^{x_4} \nabla_x x_4^T h = e^{x_4} D_h x_4$

$x_6 := x_4 + x_5 \to D_h x_6 = \ldots = D_h x_4 + D_h x_5$

$x_7 := x_6 / x_3$

Automatische Vorwärts-Berechnung von $\nabla f(x)^T h$

Gleichzeitig mit f kann für ein gegebenes $h \in \mathbb{R}^n$ die Richtungsableitung $D_h f(x) := \nabla f(x)^T h$ berechnet werden. Für D_h lautet die Kettenregel

$$D_h g(y(x)) = \nabla_x g(y(x))^T h = \sum_{i=1}^{m} \frac{\partial g}{\partial y_i}(y) \nabla_x y_i(x)^T h = \sum_{i=1}^{m} \frac{\partial g}{\partial y_i}(y) D_h y_i(x)$$

Am Bsp: $f(x) = (x_1 x_2 + e^{x_1 x_2})/x_3$, $\quad D_h$ ... Richtungsableitung für h

Beginne mit $D_h x_1 := \nabla_x x_1^T h = e_1^T h = h_1$, $D_h x_2 = \nabla_x x_2^T h = h_2$, $D_h x_3 = h_3$,

$x_4 := x_1 x_2 \quad \to D_h x_4 = x_2 \nabla_x x_1^T h + x_1 \nabla_x x_2^T h = x_2 D_h x_1 + x_1 D_h x_2 = x_2 h_1 + x_1 h_2$

$x_5 := e^{x_4} \quad \to D_h x_5 = e^{x_4} \nabla_x x_4^T h = e^{x_4} D_h x_4$

$x_6 := x_4 + x_5 \to D_h x_6 = \ldots = D_h x_4 + D_h x_5$

$x_7 := x_6 / x_3 \to D_h x_7 = \ldots = x_3^{-1} D_h x_6 - x_6 x_3^{-2} D_h x_3$

Automatische Vorwärts-Berechnung von $\nabla f(x)^T h$

Gleichzeitig mit f kann für ein gegebenes $h \in \mathbb{R}^n$ die Richtungsableitung $D_h f(x) := \nabla f(x)^T h$ berechnet werden. Für D_h lautet die Kettenregel

$$D_h g(y(x)) = \nabla_x g(y(x))^T h = \sum_{i=1}^{m} \frac{\partial g}{\partial y_i}(y) \nabla_x y_i(x)^T h = \sum_{i=1}^{m} \frac{\partial g}{\partial y_i}(y) D_h y_i(x)$$

Am Bsp: $f(x) = (x_1 x_2 + e^{x_1 x_2})/x_3$, $\quad D_h$ ... Richtungsableitung für h

Beginne mit $D_h x_1 := \nabla_x x_1^T h = e_1^T h = h_1$, $D_h x_2 = \nabla_x x_2^T h = h_2$, $D_h x_3 = h_3$,

$x_4 := x_1 x_2 \quad \to D_h x_4 = x_2 \nabla_x x_1^T h + x_1 \nabla_x x_2^T h = x_2 D_h x_1 + x_1 D_h x_2 = x_2 h_1 + x_1 h_2$

$x_5 := e^{x_4} \quad \to D_h x_5 = e^{x_4} \nabla_x x_4^T h = e^{x_4} D_h x_4$

$x_6 := x_4 + x_5 \to D_h x_6 = \ldots = D_h x_4 + D_h x_5$

$x_7 := x_6 / x_3 \to D_h x_7 = \ldots = x_3^{-1} D_h x_6 - x_6 x_3^{-2} D_h x_3$

$D_h f(x) = \nabla f(x)^T h$ kann also zugleich mit f und mit relativ geringem Zusatzaufwand und Speicher berechnet werden!

Automatische Rückwärts-Berechnung von $\nabla f(x)$

Taucht x_i nur in den Variablen $j \in N_i$ explizit auf, sagt die Kettenregel

$$\frac{\partial f}{\partial x_i} = \sum_{j \in N_i} \frac{\partial f}{\partial x_j} \frac{\partial x_j}{\partial x_i}$$

Die Werte $\frac{\partial x_j}{\partial x_i}$ werden in der Vorwärts-Berechnung ermittelt. Sind einmal alle Werte $\frac{\partial f}{\partial x_j}$ für $j \in N_i$ bekannt, ergibt sich $\frac{\partial f}{\partial x_i}$ aus obiger Formel.

Automatische Rückwärts-Berechnung von $\nabla f(x)$

Taucht x_i nur in den Variablen $j \in N_i$ explizit auf, sagt die Kettenregel

$$\frac{\partial f}{\partial x_i} = \sum_{j \in N_i} \frac{\partial f}{\partial x_j} \frac{\partial x_j}{\partial x_i}$$

Die Werte $\frac{\partial x_j}{\partial x_i}$ werden in der Vorwärts-Berechnung ermittelt. Sind einmal alle Werte $\frac{\partial f}{\partial x_j}$ für $j \in N_i$ bekannt, ergibt sich $\frac{\partial f}{\partial x_i}$ aus obiger Formel.

Bsp: $f(x) = (x_1 x_2 + e^{x_1 x_2})/x_3$ für $(x_1, x_2, x_3) = (2, 0, 3)$

x_i	N_i	$\frac{\partial x_j}{\partial x_i}$	$\frac{\partial f}{\partial x_i}$
$x_1 =$	$\{4\}$	$\frac{\partial x_4}{\partial x_1} =$	
$x_2 =$	$\{4\}$	$\frac{\partial x_4}{\partial x_2} =$	
$x_3 =$	$\{7\}$	$\frac{\partial x_7}{\partial x_3} =$	
$x_4 := x_1 x_2 =$	$\{5, 6\}$	$\frac{\partial x_5}{\partial x_4} =$, $\frac{\partial x_6}{\partial x_4} =$	
$x_5 := e^{x_4} =$	$\{6\}$	$\frac{\partial x_6}{\partial x_5} =$	
$x_6 := x_4 + x_5 =$	$\{7\}$	$\frac{\partial x_7}{\partial x_6} =$	
$x_7 := x_6/x_3 =$	$\emptyset$		

Vorwärts-Berechnung:

Automatische Rückwärts-Berechnung von $\nabla f(x)$

Taucht x_i nur in den Variablen $j \in N_i$ explizit auf, sagt die Kettenregel

$$\frac{\partial f}{\partial x_i} = \sum_{j \in N_i} \frac{\partial f}{\partial x_j} \frac{\partial x_j}{\partial x_i}$$

Die Werte $\frac{\partial x_j}{\partial x_i}$ werden in der Vorwärts-Berechnung ermittelt. Sind einmal alle Werte $\frac{\partial f}{\partial x_j}$ für $j \in N_i$ bekannt, ergibt sich $\frac{\partial f}{\partial x_i}$ aus obiger Formel.

Bsp: $f(x) = (x_1 x_2 + e^{x_1 x_2})/x_3$ für $(x_1, x_2, x_3) = (2, 0, 3)$

x_i		N_i	$\frac{\partial x_j}{\partial x_i}$		$\frac{\partial f}{\partial x_i}$
x_1	$=2$	$\{4\}$	$\frac{\partial x_4}{\partial x_1} =$		
x_2	$=$	$\{4\}$	$\frac{\partial x_4}{\partial x_2} =$		
x_3	$=$	$\{7\}$	$\frac{\partial x_7}{\partial x_3} =$		
$x_4 := x_1 x_2$	$=$	$\{5, 6\}$	$\frac{\partial x_5}{\partial x_4} =$	, $\frac{\partial x_6}{\partial x_4} =$	
$x_5 := e^{x_4}$	$=$	$\{6\}$	$\frac{\partial x_6}{\partial x_5} =$		
$x_6 := x_4 + x_5$	$=$	$\{7\}$	$\frac{\partial x_7}{\partial x_6} =$		
$x_7 := x_6/x_3$	$=$	$\emptyset$			

Vorwärts-Berechnung: x_1

Automatische Rückwärts-Berechnung von $\nabla f(x)$

Taucht x_i nur in den Variablen $j \in N_i$ explizit auf, sagt die Kettenregel

$$\frac{\partial f}{\partial x_i} = \sum_{j \in N_i} \frac{\partial f}{\partial x_j} \frac{\partial x_j}{\partial x_i}$$

Die Werte $\frac{\partial x_j}{\partial x_i}$ werden in der Vorwärts-Berechnung ermittelt. Sind einmal alle Werte $\frac{\partial f}{\partial x_j}$ für $j \in N_i$ bekannt, ergibt sich $\frac{\partial f}{\partial x_i}$ aus obiger Formel.

Bsp: $f(x) = (x_1 x_2 + e^{x_1 x_2})/x_3$ für $(x_1, x_2, x_3) = (2, 0, 3)$

x_i		N_i	$\frac{\partial x_j}{\partial x_i}$	$\frac{\partial f}{\partial x_i}$
x_1	$=2$	$\{4\}$	$\frac{\partial x_4}{\partial x_1} =$	
x_2	$=0$	$\{4\}$	$\frac{\partial x_4}{\partial x_2} =$	
x_3	$=$	$\{7\}$	$\frac{\partial x_7}{\partial x_3} =$	
$x_4 := x_1 x_2$	$=$	$\{5, 6\}$	$\frac{\partial x_5}{\partial x_4} =$, $\frac{\partial x_6}{\partial x_4} =$	
$x_5 := e^{x_4}$	$=$	$\{6\}$	$\frac{\partial x_6}{\partial x_5} =$	
$x_6 := x_4 + x_5$	$=$	$\{7\}$	$\frac{\partial x_7}{\partial x_6} =$	
$x_7 := x_6/x_3$	$=$	$\emptyset$		

Vorwärts-Berechnung: x_2

Automatische Rückwärts-Berechnung von $\nabla f(x)$

Taucht x_i nur in den Variablen $j \in N_i$ explizit auf, sagt die Kettenregel

$$\frac{\partial f}{\partial x_i} = \sum_{j \in N_i} \frac{\partial f}{\partial x_j} \frac{\partial x_j}{\partial x_i}$$

Die Werte $\frac{\partial x_j}{\partial x_i}$ werden in der Vorwärts-Berechnung ermittelt. Sind einmal alle Werte $\frac{\partial f}{\partial x_j}$ für $j \in N_i$ bekannt, ergibt sich $\frac{\partial f}{\partial x_i}$ aus obiger Formel.

Bsp: $f(x) = (x_1 x_2 + e^{x_1 x_2})/x_3$ für $(x_1, x_2, x_3) = (2, 0, 3)$

x_i		N_i	$\frac{\partial x_j}{\partial x_i}$		$\frac{\partial f}{\partial x_i}$
x_1	$=2$	$\{4\}$	$\frac{\partial x_4}{\partial x_1} =$		
x_2	$=0$	$\{4\}$	$\frac{\partial x_4}{\partial x_2} =$		
x_3	$=3$	$\{7\}$	$\frac{\partial x_7}{\partial x_3} =$		
$x_4 := x_1 x_2$	$=$	$\{5, 6\}$	$\frac{\partial x_5}{\partial x_4} =$	, $\frac{\partial x_6}{\partial x_4} =$	
$x_5 := e^{x_4}$	$=$	$\{6\}$	$\frac{\partial x_6}{\partial x_5} =$		
$x_6 := x_4 + x_5$	$=$	$\{7\}$	$\frac{\partial x_7}{\partial x_6} =$		
$x_7 := x_6/x_3$	$=$	$\emptyset$			

Vorwärts-Berechnung: x_3

Automatische Rückwärts-Berechnung von $\nabla f(x)$

Taucht x_i nur in den Variablen $j \in N_i$ explizit auf, sagt die Kettenregel

$$\frac{\partial f}{\partial x_i} = \sum_{j \in N_i} \frac{\partial f}{\partial x_j} \frac{\partial x_j}{\partial x_i}$$

Die Werte $\frac{\partial x_j}{\partial x_i}$ werden in der Vorwärts-Berechnung ermittelt. Sind einmal alle Werte $\frac{\partial f}{\partial x_j}$ für $j \in N_i$ bekannt, ergibt sich $\frac{\partial f}{\partial x_i}$ aus obiger Formel.

Bsp: $f(x) = (x_1 x_2 + e^{x_1 x_2})/x_3$ für $(x_1, x_2, x_3) = (2, 0, 3)$

x_i		N_i	$\frac{\partial x_j}{\partial x_i}$	$\frac{\partial f}{\partial x_i}$
x_1	$=2$	$\{4\}$	$\frac{\partial x_4}{\partial x_1} = x_2 = 0$	
x_2	$=0$	$\{4\}$	$\frac{\partial x_4}{\partial x_2} = x_1 = 2$	
x_3	$=3$	$\{7\}$	$\frac{\partial x_7}{\partial x_3} =$	
$x_4 := x_1 x_2$	$=0$	$\{5, 6\}$	$\frac{\partial x_5}{\partial x_4} =$, $\frac{\partial x_6}{\partial x_4} =$	
$x_5 := e^{x_4}$	$=$	$\{6\}$	$\frac{\partial x_6}{\partial x_5} =$	
$x_6 := x_4 + x_5$	$=$	$\{7\}$	$\frac{\partial x_7}{\partial x_6} =$	
$x_7 := x_6/x_3$	$=$	$\emptyset$		

Vorwärts-Berechnung: x_4

Automatische Rückwärts-Berechnung von $\nabla f(x)$

Taucht x_i nur in den Variablen $j \in N_i$ explizit auf, sagt die Kettenregel

$$\frac{\partial f}{\partial x_i} = \sum_{j \in N_i} \frac{\partial f}{\partial x_j} \frac{\partial x_j}{\partial x_i}$$

Die Werte $\frac{\partial x_j}{\partial x_i}$ werden in der Vorwärts-Berechnung ermittelt. Sind einmal alle Werte $\frac{\partial f}{\partial x_j}$ für $j \in N_i$ bekannt, ergibt sich $\frac{\partial f}{\partial x_i}$ aus obiger Formel.

Bsp: $f(x) = (x_1 x_2 + e^{x_1 x_2})/x_3$ für $(x_1, x_2, x_3) = (2, 0, 3)$

x_i		N_i	$\frac{\partial x_j}{\partial x_i}$	$\frac{\partial f}{\partial x_i}$
x_1	$=2$	$\{4\}$	$\frac{\partial x_4}{\partial x_1} = x_2 = 0$	
x_2	$=0$	$\{4\}$	$\frac{\partial x_4}{\partial x_2} = x_1 = 2$	
x_3	$=3$	$\{7\}$	$\frac{\partial x_7}{\partial x_3} =$	
$x_4 := x_1 x_2$	$=0$	$\{5, 6\}$	$\frac{\partial x_5}{\partial x_4} = e^{x_4} = 1$, $\frac{\partial x_6}{\partial x_4} =$	
$x_5 := e^{x_4}$	$=1$	$\{6\}$	$\frac{\partial x_6}{\partial x_5} =$	
$x_6 := x_4 + x_5$	$=$	$\{7\}$	$\frac{\partial x_7}{\partial x_6} =$	
$x_7 := x_6/x_3$	$=$	$\emptyset$		

Vorwärts-Berechnung: x_5

Automatische Rückwärts-Berechnung von $\nabla f(x)$

Taucht x_i nur in den Variablen $j \in N_i$ explizit auf, sagt die Kettenregel

$$\frac{\partial f}{\partial x_i} = \sum_{j \in N_i} \frac{\partial f}{\partial x_j} \frac{\partial x_j}{\partial x_i}$$

Die Werte $\frac{\partial x_j}{\partial x_i}$ werden in der Vorwärts-Berechnung ermittelt. Sind einmal alle Werte $\frac{\partial f}{\partial x_j}$ für $j \in N_i$ bekannt, ergibt sich $\frac{\partial f}{\partial x_i}$ aus obiger Formel.

Bsp: $f(x) = (x_1 x_2 + e^{x_1 x_2})/x_3$ für $(x_1, x_2, x_3) = (2, 0, 3)$

x_i		N_i	$\frac{\partial x_j}{\partial x_i}$	$\frac{\partial f}{\partial x_i}$
x_1	$=2$	$\{4\}$	$\frac{\partial x_4}{\partial x_1} = x_2 = 0$	
x_2	$=0$	$\{4\}$	$\frac{\partial x_4}{\partial x_2} = x_1 = 2$	
x_3	$=3$	$\{7\}$	$\frac{\partial x_7}{\partial x_3} =$	
$x_4 := x_1 x_2$	$=0$	$\{5, 6\}$	$\frac{\partial x_5}{\partial x_4} = e^{x_4} = 1$, $\frac{\partial x_6}{\partial x_4} = 1$	
$x_5 := e^{x_4}$	$=1$	$\{6\}$	$\frac{\partial x_6}{\partial x_5} = 1$	
$x_6 := x_4 + x_5$	$=1$	$\{7\}$	$\frac{\partial x_7}{\partial x_6} =$	
$x_7 := x_6/x_3$	$=$	$\emptyset$		

Vorwärts-Berechnung: x_6

Automatische Rückwärts-Berechnung von $\nabla f(x)$

Taucht x_i nur in den Variablen $j \in N_i$ explizit auf, sagt die Kettenregel

$$\frac{\partial f}{\partial x_i} = \sum_{j \in N_i} \frac{\partial f}{\partial x_j} \frac{\partial x_j}{\partial x_i}$$

Die Werte $\frac{\partial x_j}{\partial x_i}$ werden in der Vorwärts-Berechnung ermittelt. Sind einmal alle Werte $\frac{\partial f}{\partial x_j}$ für $j \in N_i$ bekannt, ergibt sich $\frac{\partial f}{\partial x_i}$ aus obiger Formel.

Bsp: $f(x) = (x_1 x_2 + e^{x_1 x_2})/x_3$ für $(x_1, x_2, x_3) = (2, 0, 3)$

x_i		N_i	$\frac{\partial x_j}{\partial x_i}$	$\frac{\partial f}{\partial x_i}$
x_1	$=2$	$\{4\}$	$\frac{\partial x_4}{\partial x_1} = x_2 = 0$	
x_2	$=0$	$\{4\}$	$\frac{\partial x_4}{\partial x_2} = x_1 = 2$	
x_3	$=3$	$\{7\}$	$\frac{\partial x_7}{\partial x_3} = \frac{x_6}{x_3^2} = \frac{1}{9}$	
$x_4 := x_1 x_2$	$=0$	$\{5, 6\}$	$\frac{\partial x_5}{\partial x_4} = e^{x_4} = 1$, $\frac{\partial x_6}{\partial x_4} = 1$	
$x_5 := e^{x_4}$	$=1$	$\{6\}$	$\frac{\partial x_6}{\partial x_5} = 1$	
$x_6 := x_4 + x_5$	$=1$	$\{7\}$	$\frac{\partial x_7}{\partial x_6} = \frac{1}{x_3} = \frac{1}{3}$	
$x_7 := x_6/x_3$	$=\frac{1}{3}$	$\emptyset$	— ($f = x_7$!!!)	

Vorwärts-Berechnung: x_7

Automatische Rückwärts-Berechnung von $\nabla f(x)$

Taucht x_i nur in den Variablen $j \in N_i$ explizit auf, sagt die Kettenregel

$$\frac{\partial f}{\partial x_i} = \sum_{j \in N_i} \frac{\partial f}{\partial x_j} \frac{\partial x_j}{\partial x_i}$$

Die Werte $\frac{\partial x_j}{\partial x_i}$ werden in der Vorwärts-Berechnung ermittelt. Sind einmal alle Werte $\frac{\partial f}{\partial x_j}$ für $j \in N_i$ bekannt, ergibt sich $\frac{\partial f}{\partial x_i}$ aus obiger Formel.

Bsp: $f(x) = (x_1 x_2 + e^{x_1 x_2})/x_3$ für $(x_1, x_2, x_3) = (2, 0, 3)$

x_i		N_i	$\frac{\partial x_j}{\partial x_i}$	$\frac{\partial f}{\partial x_i}$
x_1	$=2$	$\{4\}$	$\frac{\partial x_4}{\partial x_1} = x_2 = 0$	
x_2	$=0$	$\{4\}$	$\frac{\partial x_4}{\partial x_2} = x_1 = 2$	
x_3	$=3$	$\{7\}$	$\frac{\partial x_7}{\partial x_3} = \frac{x_6}{x_3^2} = \frac{1}{9}$	
$x_4 := x_1 x_2$	$=0$	$\{5, 6\}$	$\frac{\partial x_5}{\partial x_4} = e^{x_4} = 1$, $\frac{\partial x_6}{\partial x_4} = 1$	
$x_5 := e^{x_4}$	$=1$	$\{6\}$	$\frac{\partial x_6}{\partial x_5} = 1$	
$x_6 := x_4 + x_5$	$=1$	$\{7\}$	$\frac{\partial x_7}{\partial x_6} = \frac{1}{x_3} = \frac{1}{3}$	
$x_7 := x_6/x_3$	$=\frac{1}{3}$	$\emptyset$	— $(f = x_7$!!!)	$\frac{\partial f}{\partial x_7} = \frac{\partial x_7}{\partial x_7} = 1$

Rückwärts-Berechnung: $\frac{\partial f}{\partial x_7}$

Automatische Rückwärts-Berechnung von $\nabla f(x)$

Taucht x_i nur in den Variablen $j \in N_i$ explizit auf, sagt die Kettenregel

$$\frac{\partial f}{\partial x_i} = \sum_{j \in N_i} \frac{\partial f}{\partial x_j} \frac{\partial x_j}{\partial x_i}$$

Die Werte $\frac{\partial x_j}{\partial x_i}$ werden in der Vorwärts-Berechnung ermittelt. Sind einmal alle Werte $\frac{\partial f}{\partial x_j}$ für $j \in N_i$ bekannt, ergibt sich $\frac{\partial f}{\partial x_i}$ aus obiger Formel.

Bsp: $f(x) = (x_1 x_2 + e^{x_1 x_2})/x_3$ für $(x_1, x_2, x_3) = (2, 0, 3)$

x_i		N_i	$\frac{\partial x_j}{\partial x_i}$	$\frac{\partial f}{\partial x_i}$
x_1	$=2$	$\{4\}$	$\frac{\partial x_4}{\partial x_1} = x_2 = 0$	
x_2	$=0$	$\{4\}$	$\frac{\partial x_4}{\partial x_2} = x_1 = 2$	
x_3	$=3$	$\{7\}$	$\frac{\partial x_7}{\partial x_3} = \frac{x_6}{x_3^2} = \frac{1}{9}$	
$x_4 := x_1 x_2$	$=0$	$\{5, 6\}$	$\frac{\partial x_5}{\partial x_4} = e^{x_4} = 1$, $\frac{\partial x_6}{\partial x_4} = 1$	
$x_5 := e^{x_4}$	$=1$	$\{6\}$	$\frac{\partial x_6}{\partial x_5} = 1$	
$x_6 := x_4 + x_5$	$=1$	$\{7\}$	$\frac{\partial x_7}{\partial x_6} = \frac{1}{x_3} = \frac{1}{3}$	$\frac{\partial f}{\partial x_6} = 1 \cdot \frac{1}{3} = \frac{1}{3}$
$x_7 := x_6/x_3$	$=\frac{1}{3}$	$\emptyset$	— $(f = x_7$!!!)	$\frac{\partial f}{\partial x_7} = \frac{\partial x_7}{\partial x_7} = 1$

Rückwärts-Berechnung: $\frac{\partial f}{\partial x_6}$

Automatische Rückwärts-Berechnung von $\nabla f(x)$

Taucht x_i nur in den Variablen $j \in N_i$ explizit auf, sagt die Kettenregel

$$\frac{\partial f}{\partial x_i} = \sum_{j \in N_i} \frac{\partial f}{\partial x_j} \frac{\partial x_j}{\partial x_i}$$

Die Werte $\frac{\partial x_j}{\partial x_i}$ werden in der Vorwärts-Berechnung ermittelt. Sind einmal alle Werte $\frac{\partial f}{\partial x_j}$ für $j \in N_i$ bekannt, ergibt sich $\frac{\partial f}{\partial x_i}$ aus obiger Formel.

Bsp: $f(x) = (x_1 x_2 + e^{x_1 x_2})/x_3$ für $(x_1, x_2, x_3) = (2, 0, 3)$

x_i		N_i	$\frac{\partial x_j}{\partial x_i}$	$\frac{\partial f}{\partial x_i}$
x_1	$=2$	$\{4\}$	$\frac{\partial x_4}{\partial x_1} = x_2 = 0$	
x_2	$=0$	$\{4\}$	$\frac{\partial x_4}{\partial x_2} = x_1 = 2$	
x_3	$=3$	$\{7\}$	$\frac{\partial x_7}{\partial x_3} = \frac{x_6}{x_3^2} = \frac{1}{9}$	
$x_4 := x_1 x_2$	$=0$	$\{5, 6\}$	$\frac{\partial x_5}{\partial x_4} = e^{x_4} = 1,\ \frac{\partial x_6}{\partial x_4} = 1$	
$x_5 := e^{x_4}$	$=1$	$\{6\}$	$\frac{\partial x_6}{\partial x_5} = 1$	$\frac{\partial f}{\partial x_5} = \frac{1}{3} \cdot 1 = \frac{1}{3}$
$x_6 := x_4 + x_5$	$=1$	$\{7\}$	$\frac{\partial x_7}{\partial x_6} = \frac{1}{x_3} = \frac{1}{3}$	$\frac{\partial f}{\partial x_6} = 1 \cdot \frac{1}{3} = \frac{1}{3}$
$x_7 := x_6/x_3$	$=\frac{1}{3}$	$\emptyset$	— $(f = x_7$!!!)	$\frac{\partial f}{\partial x_7} = \frac{\partial x_7}{\partial x_7} = 1$

Rückwärts-Berechnung: $\frac{\partial f}{\partial x_5}$

Automatische Rückwärts-Berechnung von $\nabla f(x)$

Taucht x_i nur in den Variablen $j \in N_i$ explizit auf, sagt die Kettenregel

$$\frac{\partial f}{\partial x_i} = \sum_{j \in N_i} \frac{\partial f}{\partial x_j} \frac{\partial x_j}{\partial x_i}$$

Die Werte $\frac{\partial x_j}{\partial x_i}$ werden in der Vorwärts-Berechnung ermittelt. Sind einmal alle Werte $\frac{\partial f}{\partial x_j}$ für $j \in N_i$ bekannt, ergibt sich $\frac{\partial f}{\partial x_i}$ aus obiger Formel.

Bsp: $f(x) = (x_1 x_2 + e^{x_1 x_2})/x_3$ für $(x_1, x_2, x_3) = (2, 0, 3)$

x_i		N_i	$\frac{\partial x_j}{\partial x_i}$	$\frac{\partial f}{\partial x_i}$
x_1	$=2$	$\{4\}$	$\frac{\partial x_4}{\partial x_1} = x_2 = 0$	
x_2	$=0$	$\{4\}$	$\frac{\partial x_4}{\partial x_2} = x_1 = 2$	
x_3	$=3$	$\{7\}$	$\frac{\partial x_7}{\partial x_3} = \frac{x_6}{x_3^2} = \frac{1}{9}$	
$x_4 := x_1 x_2$	$=0$	$\{5, 6\}$	$\frac{\partial x_5}{\partial x_4} = e^{x_4} = 1,\ \frac{\partial x_6}{\partial x_4} = 1$	$\frac{\partial f}{\partial x_4} = \frac{1}{3} \cdot 1 + \frac{1}{3} \cdot 1 = \frac{2}{3}$
$x_5 := e^{x_4}$	$=1$	$\{6\}$	$\frac{\partial x_6}{\partial x_5} = 1$	$\frac{\partial f}{\partial x_5} = \frac{1}{3} \cdot 1 = \frac{1}{3}$
$x_6 := x_4 + x_5$	$=1$	$\{7\}$	$\frac{\partial x_7}{\partial x_6} = \frac{1}{x_3} = \frac{1}{3}$	$\frac{\partial f}{\partial x_6} = 1 \cdot \frac{1}{3} = \frac{1}{3}$
$x_7 := x_6/x_3$	$=\frac{1}{3}$	$\emptyset$	— $(f = x_7$!!!)	$\frac{\partial f}{\partial x_7} = \frac{\partial x_7}{\partial x_7} = 1$

Rückwärts-Berechnung: $\frac{\partial f}{\partial x_4}$

Automatische Rückwärts-Berechnung von $\nabla f(x)$

Taucht x_i nur in den Variablen $j \in N_i$ explizit auf, sagt die Kettenregel

$$\frac{\partial f}{\partial x_i} = \sum_{j \in N_i} \frac{\partial f}{\partial x_j} \frac{\partial x_j}{\partial x_i}$$

Die Werte $\frac{\partial x_j}{\partial x_i}$ werden in der Vorwärts-Berechnung ermittelt. Sind einmal alle Werte $\frac{\partial f}{\partial x_j}$ für $j \in N_i$ bekannt, ergibt sich $\frac{\partial f}{\partial x_i}$ aus obiger Formel.

Bsp: $f(x) = (x_1 x_2 + e^{x_1 x_2})/x_3$ für $(x_1, x_2, x_3) = (2, 0, 3)$

x_i		N_i	$\frac{\partial x_j}{\partial x_i}$	$\frac{\partial f}{\partial x_i}$
x_1	$=2$	$\{4\}$	$\frac{\partial x_4}{\partial x_1} = x_2 = 0$	
x_2	$=0$	$\{4\}$	$\frac{\partial x_4}{\partial x_2} = x_1 = 2$	
x_3	$=3$	$\{7\}$	$\frac{\partial x_7}{\partial x_3} = \frac{x_6}{x_3^2} = \frac{1}{9}$	$\frac{\partial f}{\partial x_3} = 1 \cdot \frac{1}{9} = \frac{1}{9}$
$x_4 := x_1 x_2$	$=0$	$\{5, 6\}$	$\frac{\partial x_5}{\partial x_4} = e^{x_4} = 1,\ \frac{\partial x_6}{\partial x_4} = 1$	$\frac{\partial f}{\partial x_4} = \frac{1}{3} \cdot 1 + \frac{1}{3} \cdot 1 = \frac{2}{3}$
$x_5 := e^{x_4}$	$=1$	$\{6\}$	$\frac{\partial x_6}{\partial x_5} = 1$	$\frac{\partial f}{\partial x_5} = \frac{1}{3} \cdot 1 = \frac{1}{3}$
$x_6 := x_4 + x_5$	$=1$	$\{7\}$	$\frac{\partial x_7}{\partial x_6} = \frac{1}{x_3} = \frac{1}{3}$	$\frac{\partial f}{\partial x_6} = 1 \cdot \frac{1}{3} = \frac{1}{3}$
$x_7 := x_6/x_3$	$=\frac{1}{3}$	$\emptyset$	— $(f = x_7$!!!)	$\frac{\partial f}{\partial x_7} = \frac{\partial x_7}{\partial x_7} = 1$

Rückwärts-Berechnung: $\frac{\partial f}{\partial x_3}$

Automatische Rückwärts-Berechnung von $\nabla f(x)$

Taucht x_i nur in den Variablen $j \in N_i$ explizit auf, sagt die Kettenregel

$$\frac{\partial f}{\partial x_i} = \sum_{j \in N_i} \frac{\partial f}{\partial x_j} \frac{\partial x_j}{\partial x_i}$$

Die Werte $\frac{\partial x_j}{\partial x_i}$ werden in der Vorwärts-Berechnung ermittelt. Sind einmal alle Werte $\frac{\partial f}{\partial x_j}$ für $j \in N_i$ bekannt, ergibt sich $\frac{\partial f}{\partial x_i}$ aus obiger Formel.

Bsp: $f(x) = (x_1 x_2 + e^{x_1 x_2})/x_3$ für $(x_1, x_2, x_3) = (2, 0, 3)$

x_i		N_i	$\frac{\partial x_j}{\partial x_i}$	$\frac{\partial f}{\partial x_i}$
x_1	$=2$	$\{4\}$	$\frac{\partial x_4}{\partial x_1} = x_2 = 0$	
x_2	$=0$	$\{4\}$	$\frac{\partial x_4}{\partial x_2} = x_1 = 2$	$\frac{\partial f}{\partial x_2} = \frac{2}{3} \cdot 2 = \frac{4}{3}$
x_3	$=3$	$\{7\}$	$\frac{\partial x_7}{\partial x_3} = \frac{x_6}{x_3^2} = \frac{1}{9}$	$\frac{\partial f}{\partial x_3} = 1 \cdot \frac{1}{9} = \frac{1}{9}$
$x_4 := x_1 x_2$	$=0$	$\{5, 6\}$	$\frac{\partial x_5}{\partial x_4} = e^{x_4} = 1$, $\frac{\partial x_6}{\partial x_4} = 1$	$\frac{\partial f}{\partial x_4} = \frac{1}{3} \cdot 1 + \frac{1}{3} \cdot 1 = \frac{2}{3}$
$x_5 := e^{x_4}$	$=1$	$\{6\}$	$\frac{\partial x_6}{\partial x_5} = 1$	$\frac{\partial f}{\partial x_5} = \frac{1}{3} \cdot 1 = \frac{1}{3}$
$x_6 := x_4 + x_5$	$=1$	$\{7\}$	$\frac{\partial x_7}{\partial x_6} = \frac{1}{x_3} = \frac{1}{3}$	$\frac{\partial f}{\partial x_6} = 1 \cdot \frac{1}{3} = \frac{1}{3}$
$x_7 := x_6/x_3$	$=\frac{1}{3}$	$\emptyset$	— $(f = x_7$!!!)	$\frac{\partial f}{\partial x_7} = \frac{\partial x_7}{\partial x_7} = 1$

Rückwärts-Berechnung: $\frac{\partial f}{\partial x_2}$

Automatische Rückwärts-Berechnung von $\nabla f(x)$

Taucht x_i nur in den Variablen $j \in N_i$ explizit auf, sagt die Kettenregel

$$\frac{\partial f}{\partial x_i} = \sum_{j \in N_i} \frac{\partial f}{\partial x_j} \frac{\partial x_j}{\partial x_i}$$

Die Werte $\frac{\partial x_j}{\partial x_i}$ werden in der Vorwärts-Berechnung ermittelt. Sind einmal alle Werte $\frac{\partial f}{\partial x_j}$ für $j \in N_i$ bekannt, ergibt sich $\frac{\partial f}{\partial x_i}$ aus obiger Formel.

Bsp: $f(x) = (x_1 x_2 + e^{x_1 x_2})/x_3$ für $(x_1, x_2, x_3) = (2, 0, 3)$

x_i		N_i	$\frac{\partial x_j}{\partial x_i}$	$\frac{\partial f}{\partial x_i}$
x_1	$=2$	$\{4\}$	$\frac{\partial x_4}{\partial x_1} = x_2 = 0$	$\frac{\partial f}{\partial x_1} = \frac{2}{3} \cdot 0 = 0$
x_2	$=0$	$\{4\}$	$\frac{\partial x_4}{\partial x_2} = x_1 = 2$	$\frac{\partial f}{\partial x_2} = \frac{2}{3} \cdot 2 = \frac{4}{3}$
x_3	$=3$	$\{7\}$	$\frac{\partial x_7}{\partial x_3} = \frac{x_6}{x_3^2} = \frac{1}{9}$	$\frac{\partial f}{\partial x_3} = 1 \cdot \frac{1}{9} = \frac{1}{9}$
$x_4 := x_1 x_2$	$=0$	$\{5, 6\}$	$\frac{\partial x_5}{\partial x_4} = e^{x_4} = 1,\ \frac{\partial x_6}{\partial x_4} = 1$	$\frac{\partial f}{\partial x_4} = \frac{1}{3} \cdot 1 + \frac{1}{3} \cdot 1 = \frac{2}{3}$
$x_5 := e^{x_4}$	$=1$	$\{6\}$	$\frac{\partial x_6}{\partial x_5} = 1$	$\frac{\partial f}{\partial x_5} = \frac{1}{3} \cdot 1 = \frac{1}{3}$
$x_6 := x_4 + x_5$	$=1$	$\{7\}$	$\frac{\partial x_7}{\partial x_6} = \frac{1}{x_3} = \frac{1}{3}$	$\frac{\partial f}{\partial x_6} = 1 \cdot \frac{1}{3} = \frac{1}{3}$
$x_7 := x_6/x_3$	$=\frac{1}{3}$	$\emptyset$	— $(f = x_7$!!!)	$\frac{\partial f}{\partial x_7} = \frac{\partial x_7}{\partial x_7} = 1$

Rückwärts-Berechnung: $\frac{\partial f}{\partial x_1}$

Automatische Rückwärts-Berechnung von $\nabla f(x)$

Taucht x_i nur in den Variablen $j \in N_i$ explizit auf, sagt die Kettenregel

$$\frac{\partial f}{\partial x_i} = \sum_{j \in N_i} \frac{\partial f}{\partial x_j} \frac{\partial x_j}{\partial x_i}$$

Die Werte $\frac{\partial x_j}{\partial x_i}$ werden in der Vorwärts-Berechnung ermittelt. Sind einmal alle Werte $\frac{\partial f}{\partial x_j}$ für $j \in N_i$ bekannt, ergibt sich $\frac{\partial f}{\partial x_i}$ aus obiger Formel.

Bsp: $f(x) = (x_1 x_2 + e^{x_1 x_2})/x_3$ für $(x_1, x_2, x_3) = (2, 0, 3)$

x_i		N_i	$\frac{\partial x_j}{\partial x_i}$	$\frac{\partial f}{\partial x_i}$
x_1	$=2$	$\{4\}$	$\frac{\partial x_4}{\partial x_1} = x_2 = 0$	$\frac{\partial f}{\partial x_1} = \frac{2}{3} \cdot 0 = 0$
x_2	$=0$	$\{4\}$	$\frac{\partial x_4}{\partial x_2} = x_1 = 2$	$\frac{\partial f}{\partial x_2} = \frac{2}{3} \cdot 2 = \frac{4}{3}$
x_3	$=3$	$\{7\}$	$\frac{\partial x_7}{\partial x_3} = \frac{x_6}{x_3^2} = \frac{1}{9}$	$\frac{\partial f}{\partial x_3} = 1 \cdot \frac{1}{9} = \frac{1}{9}$
$x_4 := x_1 x_2$	$=0$	$\{5, 6\}$	$\frac{\partial x_5}{\partial x_4} = e^{x_4} = 1,\ \frac{\partial x_6}{\partial x_4} = 1$	$\frac{\partial f}{\partial x_4} = \frac{1}{3} \cdot 1 + \frac{1}{3} \cdot 1 = \frac{2}{3}$
$x_5 := e^{x_4}$	$=1$	$\{6\}$	$\frac{\partial x_6}{\partial x_5} = 1$	$\frac{\partial f}{\partial x_5} = \frac{1}{3} \cdot 1 = \frac{1}{3}$
$x_6 := x_4 + x_5$	$=1$	$\{7\}$	$\frac{\partial x_7}{\partial x_6} = \frac{1}{x_3} = \frac{1}{3}$	$\frac{\partial f}{\partial x_6} = 1 \cdot \frac{1}{3} = \frac{1}{3}$
$x_7 := x_6/x_3$	$=\frac{1}{3}$	$\emptyset$	— $(f = x_7$!!!)	$\frac{\partial f}{\partial x_7} = \frac{\partial x_7}{\partial x_7} = 1$

Rückwärts-Berechnung: $\nabla f = (\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \frac{\partial f}{\partial x_3})^T = (0, \frac{4}{3}, \frac{1}{9})^T$

Bemerkungen zum Automatischen Differenzieren

- Die Rückwärts-Ber. benötigt den gesamten Berechnungsablauf im Speicher, dafür ist die Laufzeit viel kürzer, als wenn $[\nabla f]_i$ über $\nabla f^T e_i$ für jedes i vorwärts berechnet wird.

Bemerkungen zum Automatischen Differenzieren

- Die Rückwärts-Ber. benötigt den gesamten Berechnungsablauf im Speicher, dafür ist die Laufzeit viel kürzer, als wenn $[\nabla f]_i$ über $\nabla f^T e_i$ für jedes i vorwärts berechnet wird.
- Der Gradient der Funktion $\nabla f(x)^T h$ von x ist $\nabla^2 f(x)h$ und benötigt eine Vorwärts- und Rückwärts-Berechnung. Ohne $\nabla^2 f$ zu bilden, ist so Hessematrix mal Vektor in etwa 12 mal #Operationen zur Berechnung von f berechenbar, aber der Speicherbedarf ist groß.

Bemerkungen zum Automatischen Differenzieren

- Die Rückwärts-Ber. benötigt den gesamten Berechnungsablauf im Speicher, dafür ist die Laufzeit viel kürzer, als wenn $[\nabla f]_i$ über $\nabla f^T e_i$ für jedes i vorwärts berechnet wird.
- Der Gradient der Funktion $\nabla f(x)^T h$ von x ist $\nabla^2 f(x)h$ und benötigt eine Vorwärts- und Rückwärts-Berechnung. Ohne $\nabla^2 f$ zu bilden, ist so Hessematrix mal Vektor in etwa 12 mal #Operationen zur Berechnung von f berechenbar, aber der Speicherbedarf ist groß.
- Ist f als Source-code (z.B. in C) gegeben, gibt es Software, die daraus automatisch alle oberen Varianten in Orakelform erzeugt (wurde z.B. schon eingesetzt, wenn f über einen Löser für partielle Differentialgleichungen berechnet wurde).

Bemerkungen zum Automatischen Differenzieren

- Die Rückwärts-Ber. benötigt den gesamten Berechnungsablauf im Speicher, dafür ist die Laufzeit viel kürzer, als wenn $[\nabla f]_i$ über $\nabla f^T e_i$ für jedes i vorwärts berechnet wird.
- Der Gradient der Funktion $\nabla f(x)^T h$ von x ist $\nabla^2 f(x)h$ und benötigt eine Vorwärts- und Rückwärts-Berechnung. Ohne $\nabla^2 f$ zu bilden, ist so Hessematrix mal Vektor in etwa 12 mal #Operationen zur Berechnung von f berechenbar, aber der Speicherbedarf ist groß.
- Ist f als Source-code (z.B. in C) gegeben, gibt es Software, die daraus automatisch alle oberen Varianten in Orakelform erzeugt (wurde z.B. schon eingesetzt, wenn f über einen Löser für partielle Differentialgleichungen berechnet wurde).
- $\nabla f^T h$ ($\nabla^2 fh$) in Rechengenauigkeit! (im Gegensatz zu num. Diff.)

Bemerkungen zum Automatischen Differenzieren

- Die Rückwärts-Ber. benötigt den gesamten Berechnungsablauf im Speicher, dafür ist die Laufzeit viel kürzer, als wenn $[\nabla f]_i$ über $\nabla f^T e_i$ für jedes i vorwärts berechnet wird.
- Der Gradient der Funktion $\nabla f(x)^T h$ von x ist $\nabla^2 f(x)h$ und benötigt eine Vorwärts- und Rückwärts-Berechnung. Ohne $\nabla^2 f$ zu bilden, ist so Hessematrix mal Vektor in etwa 12 mal #Operationen zur Berechnung von f berechenbar, aber der Speicherbedarf ist groß.
- Ist f als Source-code (z.B. in C) gegeben, gibt es Software, die daraus automatisch alle oberen Varianten in Orakelform erzeugt (wurde z.B. schon eingesetzt, wenn f über einen Löser für partielle Differentialgleichungen berechnet wurde).
- $\nabla f^T h$ ($\nabla^2 f h$) in Rechengenauigkeit! (im Gegensatz zu num. Diff.)
- Grenzen: AD-Software kommt z.B. nicht immer mit Verzweigungen/bedingten Berechnungen im Source-code zurecht.

Bemerkungen zum Automatischen Differenzieren

- Die Rückwärts-Ber. benötigt den gesamten Berechnungsablauf im Speicher, dafür ist die Laufzeit viel kürzer, als wenn $[\nabla f]_i$ über $\nabla f^T e_i$ für jedes i vorwärts berechnet wird.
- Der Gradient der Funktion $\nabla f(x)^T h$ von x ist $\nabla^2 f(x)h$ und benötigt eine Vorwärts- und Rückwärts-Berechnung. Ohne $\nabla^2 f$ zu bilden, ist so Hessematrix mal Vektor in etwa 12 mal #Operationen zur Berechnung von f berechenbar, aber der Speicherbedarf ist groß.
- Ist f als Source-code (z.B. in C) gegeben, gibt es Software, die daraus automatisch alle oberen Varianten in Orakelform erzeugt (wurde z.B. schon eingesetzt, wenn f über einen Löser für partielle Differentialgleichungen berechnet wurde).
- $\nabla f^T h$ ($\nabla^2 f h$) in Rechengenauigkeit! (im Gegensatz zu num. Diff.)
- Grenzen: AD-Software kommt z.B. nicht immer mit Verzweigungen/bedingten Berechnungen im Source-code zurecht.

Verfahren, die $\nabla^2 f(x)$ nicht explizit benötigen, sondern nur eine Routine, die $\nabla^2 f(x)$ mal Vektor berechnet, heißen „Hessian-free methods“.