technische universität
dortmund

Experimentelle Physik Vb
Astroteilchenphysik

Vorlesung
Statistische Methoden der Datenanalyse
Prof. Dr. Dr. Wolfgang Rhode

Exkurs: Numerische Minimierung

Inhalt

- Warum numerische Minimierung?
- Eindimensionale Minimierung
  - Bisektionsmethode
  - Newtonverfahren
- Mehrdimensionale Minimierung
  - Random Descent
  - Gradient Descent
  - Gradient Descent mit Momentum (Adam)
  - Newtonverfahren
- Minimierung in Python

Warum numerische Minimierung?

- Bisher: Sehr einfache Funktionen mit einfachen, analytischen Lösungen

  → Ausnahmefall!

- Regel: Hochdimensionale Likelihood-Funktionen oder Kostenfunktionen von komplexen Modellen

  → Viele Parameter

  → Mehrere lokale Minima

Einstieg: 1-dimensionale Minimierung – Bisektionsmethode

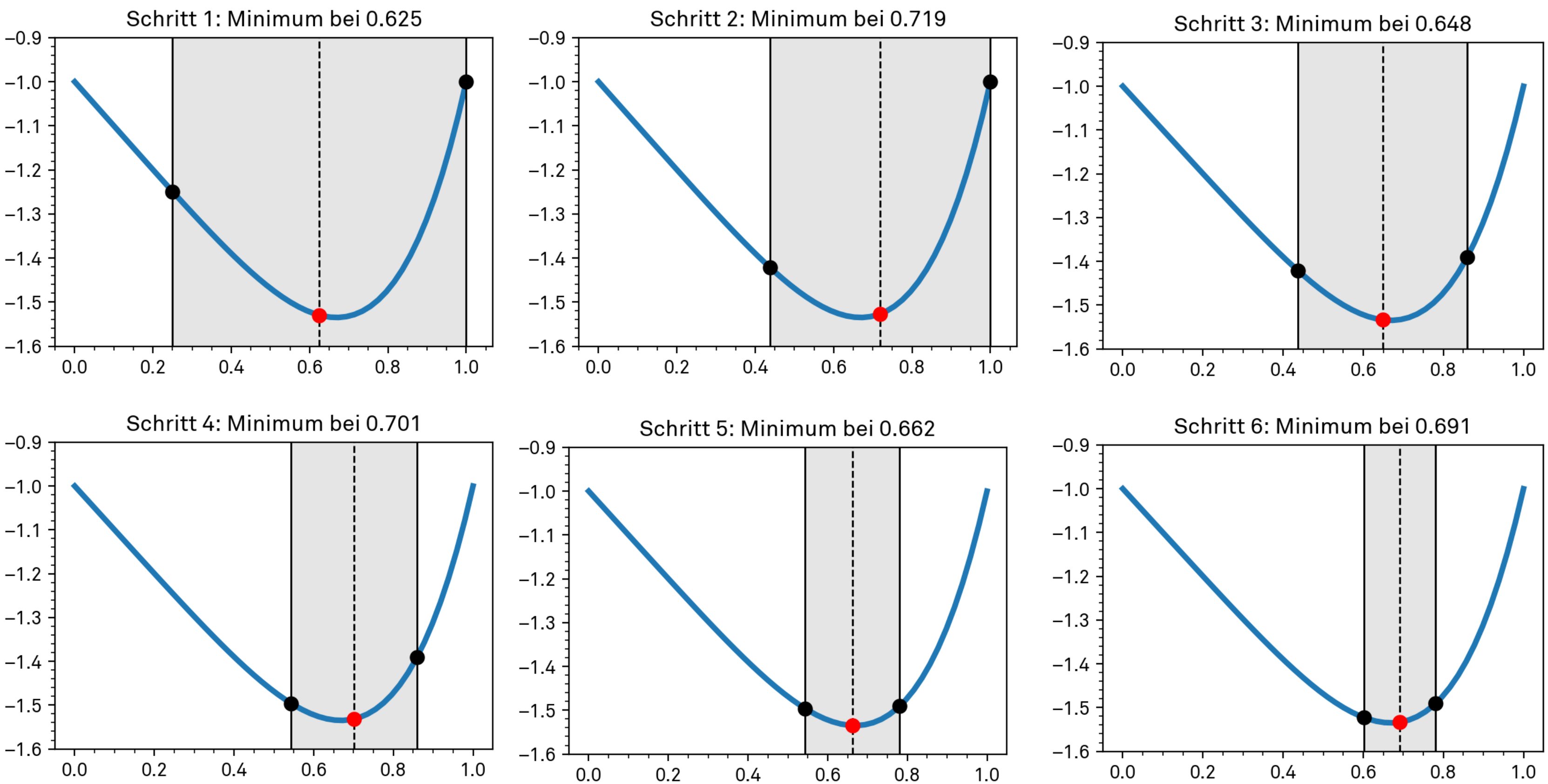
- Einfach und robust — keine Kenntnis über die Ableitung der zu minimierenden Funktion notwendig.
- Initialisierung: Man wähle *a, b* so, dass das Minimum zwischen diesen Werten liegt, oder:

$$f(a) > f\left(\frac{a+b}{2}\right) \quad \text{und} \quad f(b) > f\left(\frac{a+b}{2}\right)$$

- Im nächsten Schritt werden die aktualisiert, wobei die Option gewählt wird, die die obige Bedingung nicht verletzt:

$$a^{(k+1)} = \frac{3a^{(k)} + b^{(k)}}{4} \text{ und } b^{(k+1)} = b^{(k)} \text{ oder}$$
$$b^{(k+1)} = \frac{a^{(k)} + 3b^{(k)}}{4} \text{ und } a^{(k+1)} = a^{(k)}$$

Einstieg: 1-dimensionale Minimierung – Bisektionsmethode



Einstieg: 1-dimensionale Minimierung – Newtonverfahren

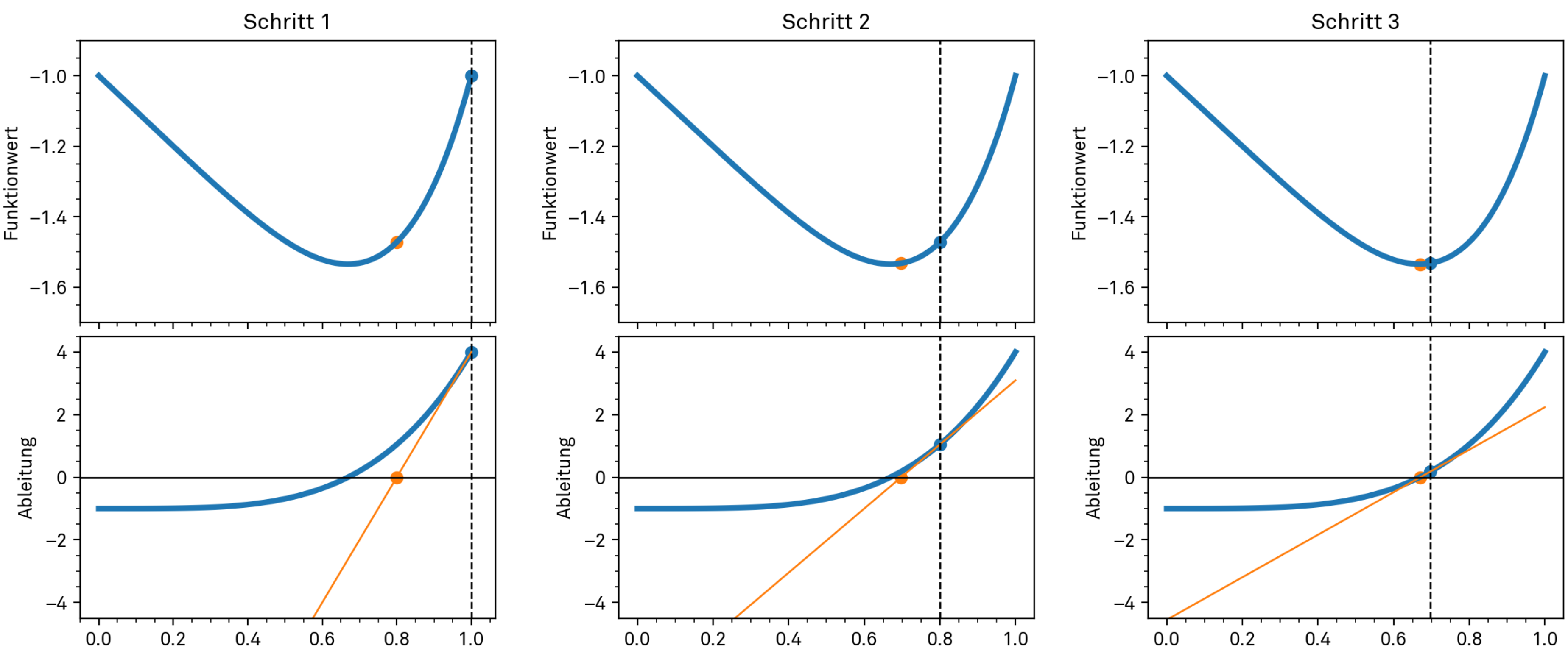
- Diesmal: Kenntnis über erste *und* zweite Ableitung der zu minimierenden Funktion.
- Minimierung → Nullstellensuche der ersten Ableitung
- Bekanntes Newtonverfahren: Berechne Tangente durch letzten Iterationswert → Aktualisiere Iterationswert durch Nullstelle der Tangente

$$x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})}{f'(x^{(k)})}$$

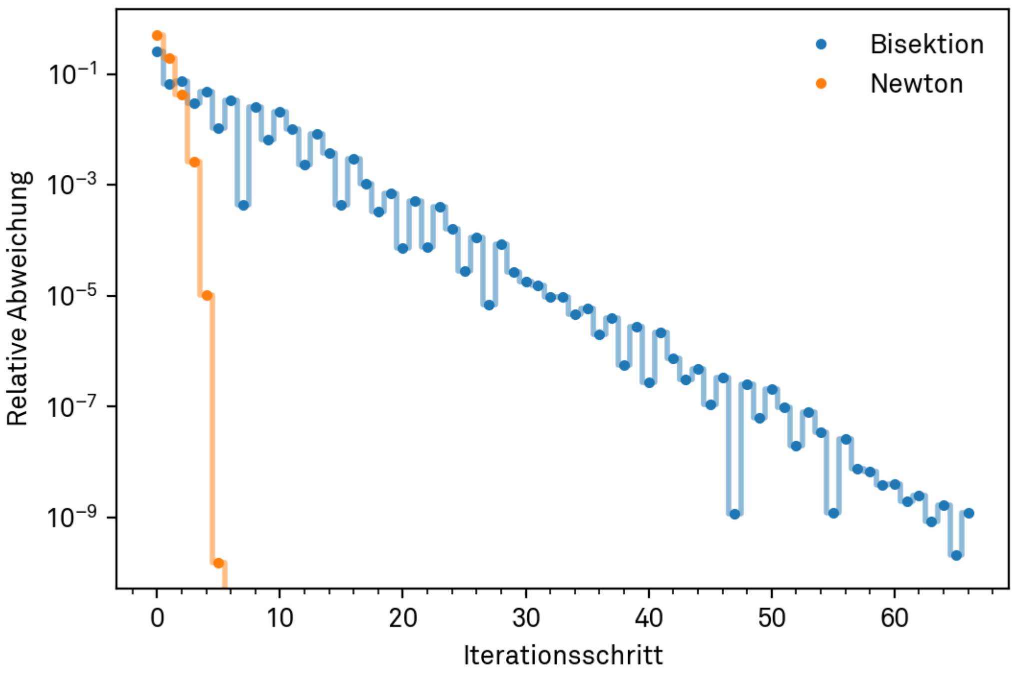
- Ersetze Funktion durch erste Ableitung:

$$x^{(k+1)} = x^{(k)} - \frac{f'(x^{(k)})}{f''(x^{(k)})}$$

Einstieg: 1-dimensionale Minimierung – Newtonverfahren



1-dimensionale Minimierung: Bisektion vs. Newton



Mehrdimensionale Minimierung – Random Descent

- Ohne Kenntnis über den Gradienten der zu minimierenden Funktion kann keine optimale Richtung ausgehend vom letzten Iterationspunkt bestimmt werden

  → Zufällige Richtung

- Einfachster Fall: Keine Berücksichtigung der letzten Iterationen und keine Schrittweitenoptimierung

  → Random Descent

Mehrdimensionale Minimierung – Random Descent

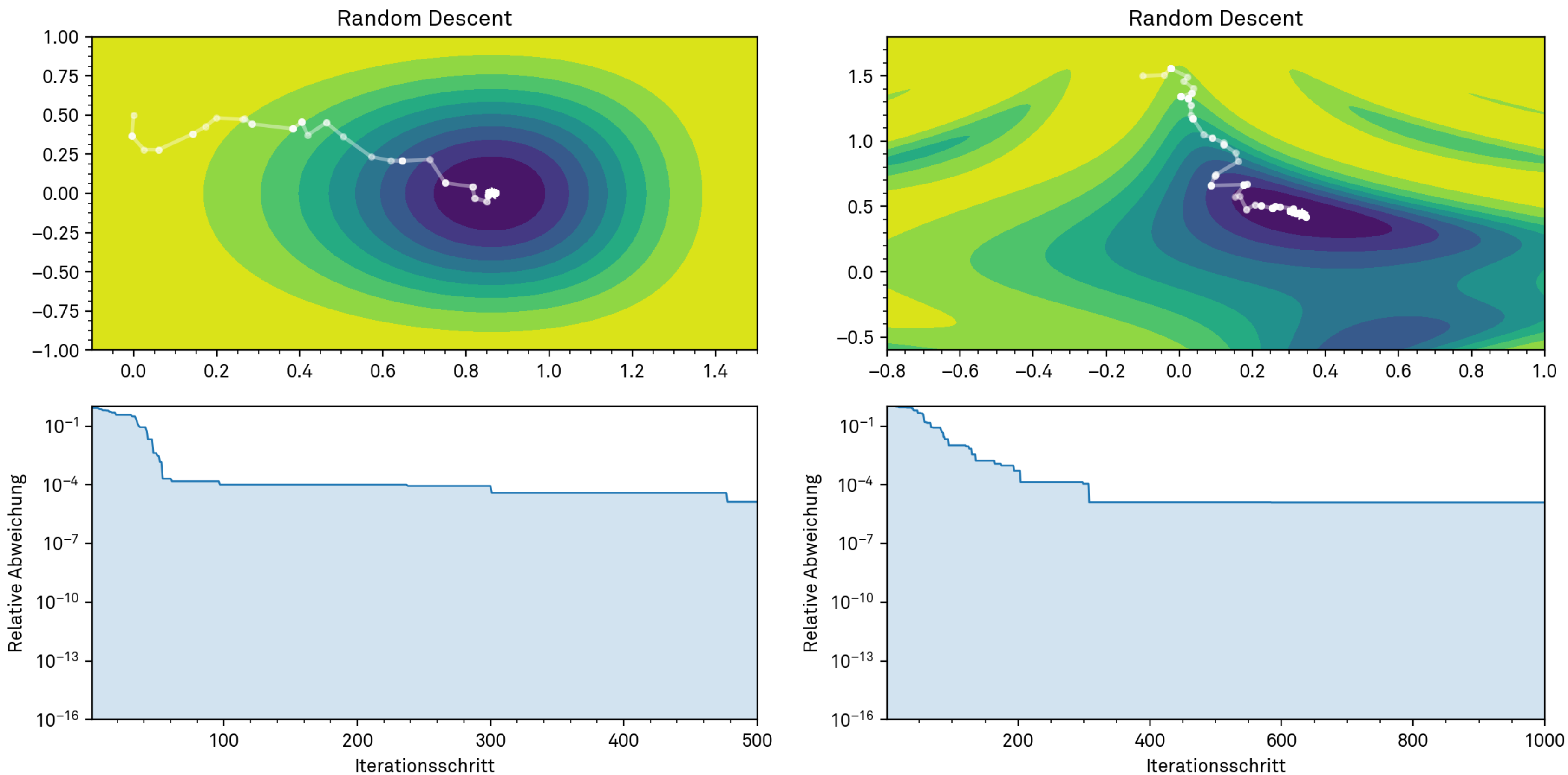
- Setze Initialwert $x^{(0)}$
- Variiere Initialwert durch gemäß einer Vorschlags-PDF (z.B. Normalverteilung)

$$x^{(k+1)} = x^{(k)} + X \quad \text{mit } X \sim g_{\text{vorschlag}}(x)$$

- Akzeptiere diesen Schritt nur wenn gilt

$$f(x^{(k+1)}) < f(x^{(k)})$$

Mehrdimensionale Minimierung – Random Descent



Mehrdimensionale Minimierung – Gradient Descent

- Nun: Kenntnis über den Gradienten vorhanden → Nutze Information um den Iterationswert entlang der Steigung zu aktualisieren

- Ist das Minimum gefunden, so verschwindet der Gradient → Keine weitere Änderung in der nächsten Iteration

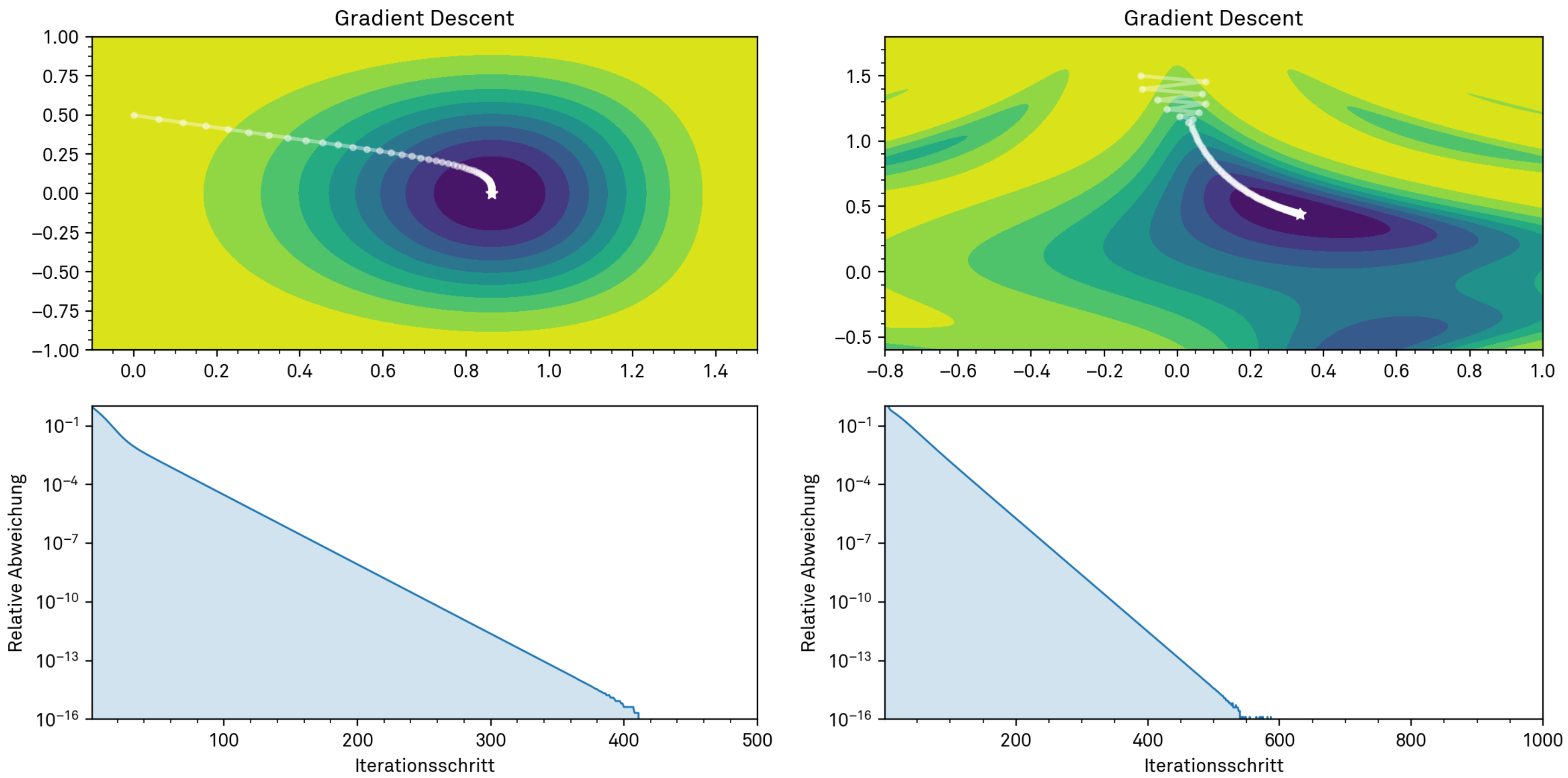
→ Gradient Descent

Mehrdimensionale Minimierung – Gradient Descent

- Setze Initialwert $x^{(0)}$
- Aktualisiere den Iterationswert gemäß

$$x^{(k+1)} = x^{(k)} - \alpha \nabla f(x^{(k)})$$

Mehrdimensionale Minimierung – Gradient Descent



Mehrdimensionale Minimierung – Adam

- Idee: Iteration erhält “Schwung” (*Momentum*) wenn sie entlang eines steilen Gradienten abfällt

- *Momentum* sorgt für eine Art adaptive Schrittweitenanpassung

- Analogie: Kugel die mit Reibung eine Landschaft hinunterrollt

→ Adam (**Ada**ptive **M**oment Estimation)

Mehrdimensionale Minimierung – Adam

- Setze Initialwerte $x^{(0)}$, $m^{(0)} = 0$, $v^{(0)} = 0$
- Aktualisiere den Iterationswert gemäß

$$m^{(k+1)} = \beta_1 m^{(k)} + (1-\beta_1)\nabla f(x^{(k)})$$

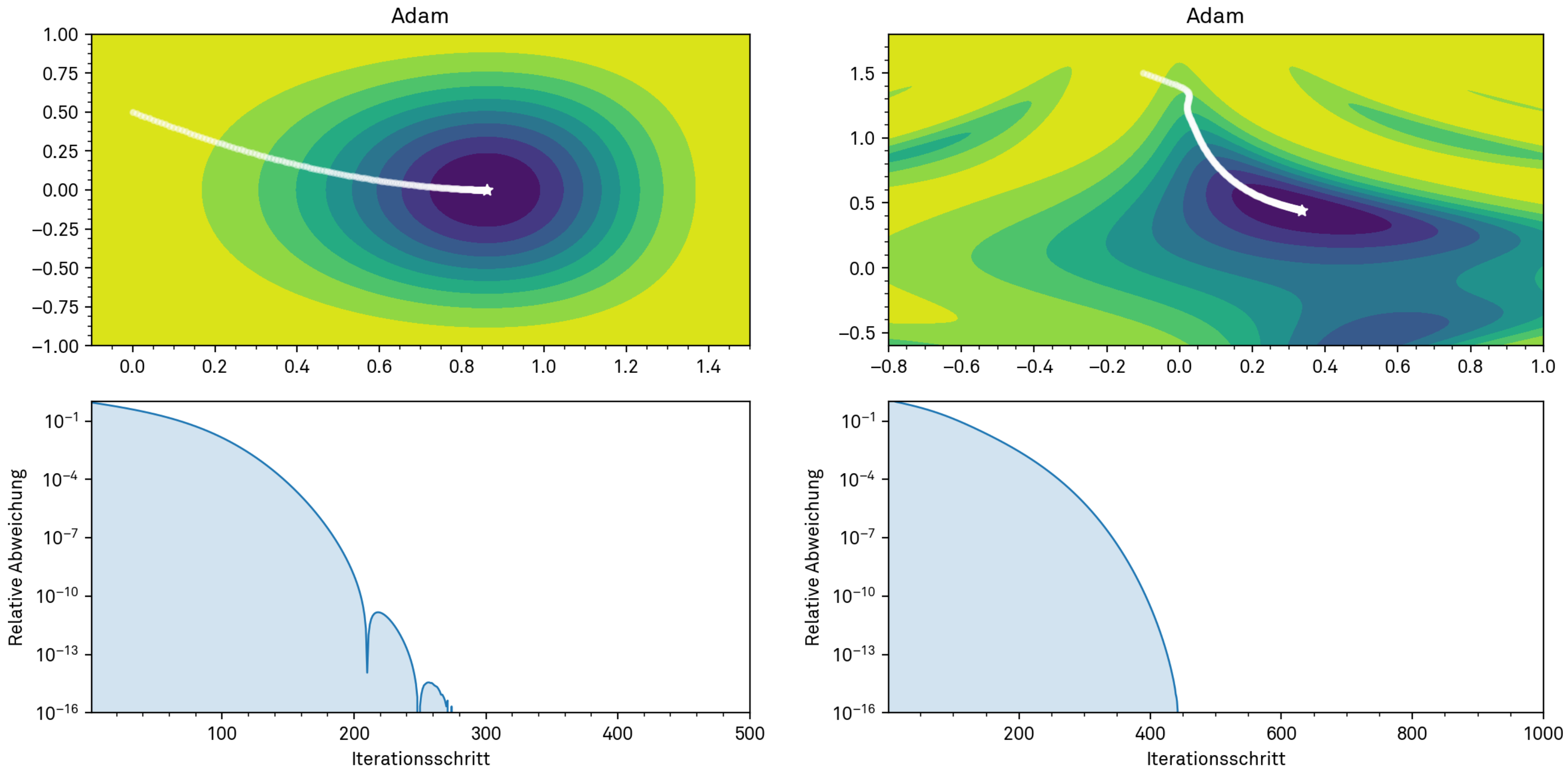
$$v^{(k+1)} = \beta_2 v^{(k)} + (1-\beta_2)\left(\nabla f(x^{(k)})\right)^2$$

$$\tilde{m} = \frac{m^{(t+1)}}{1-\beta_1^{(k+1)}}$$

$$\tilde{v} = \frac{v^{(t+1)}}{1-\beta_2^{(k+1)}}$$

$$x^{(k+1)} = x^{(k)} - \alpha\frac{\tilde{m}}{\sqrt{\tilde{v}}+\epsilon}$$

Mehrdimensionale Minimierung – Adam

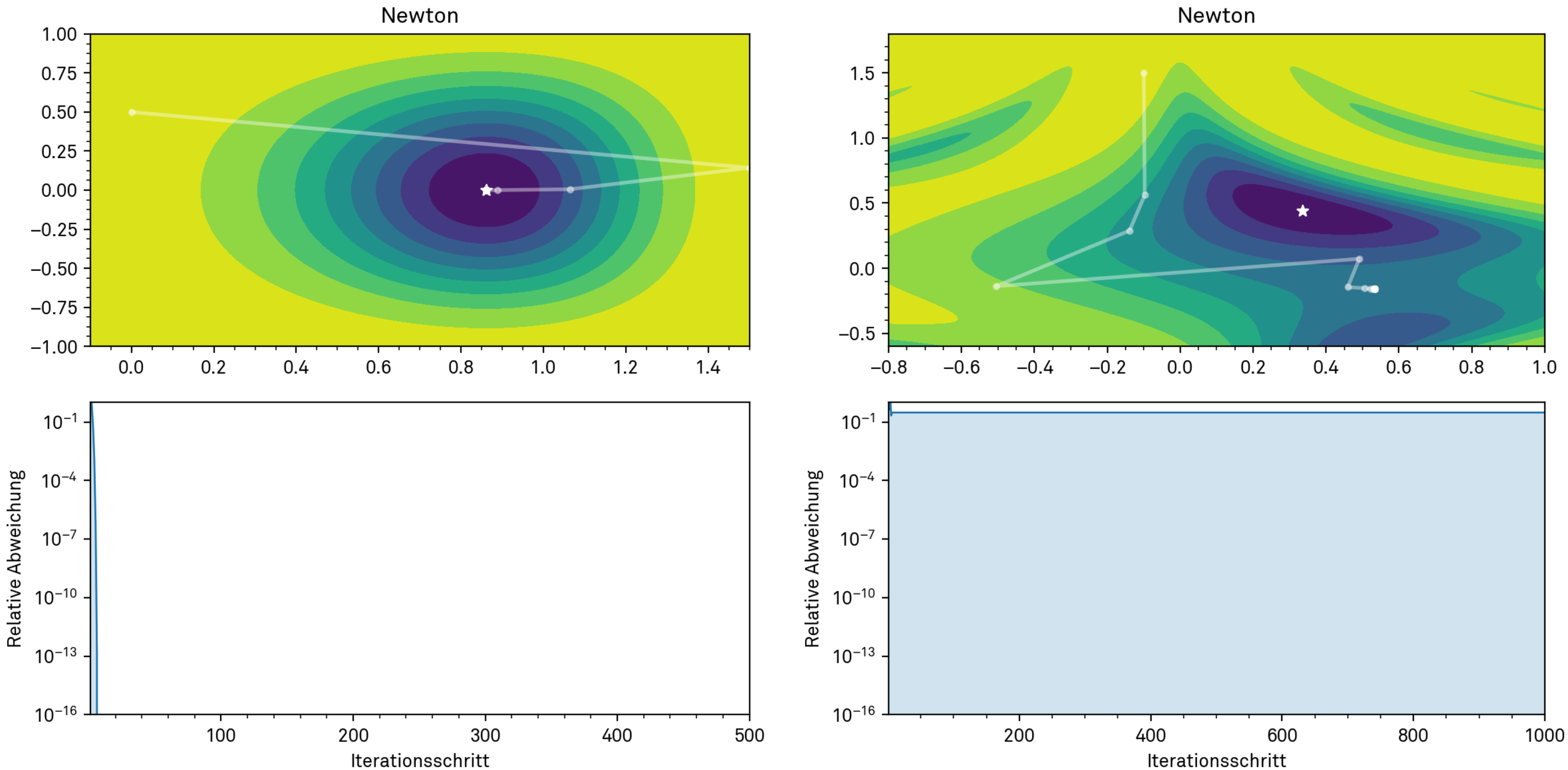


Mehrdimensionale Minimierung – Newton-Verfahren

- Analog zum eindimensionalen Verfahren kann auch in *n* Dimensionen die Information der zweiten Ableitung genutzt werden. Hierbei wird die erste Ableitung durch den Gradienten ersetzt und die zweite Ableitung durch die Hesse-Matrix.

$$\frac{f'(x)}{f''(x)} \longrightarrow \mathrm{H}_f{}^{-1} \cdot \nabla f$$

Mehrdimensionale Minimierung – Newton-Verfahren



Mehrdimensionale Minimierung

- *Random Descent* nähert sich erwartungsgemäß nur langsam dem Minimum → Dennoch sehr robust und funktioniert auch bei nicht-differenzierbaren Funktionen
- *Gradient Descent* weist oftmals oszillierendes Verhalten auf → Vorsichtiges Wählen der Schrittweite oder Ausweichen auf alternative Algorithmen wie Adam
- *Das Newton-Verfahren* findet recht zuverlässig ein Extremum, aber eben auch Maxima und Sattelpunkte → Ergebnis hängt empfindlich vom Startwert ab → Verwendung zusätzlicher Heuristiken

Numerische Minimierung in Python: 1-dimensional

```python
from scipy.optimize import minimize_scalar


def function(x):
    return x ** 5 - x - 1.0


minimize_scalar(function, bracket=(0.0, 1.0))
```

```
     fun: -1.5349922439811376
    nfev: 15
     nit: 11
 success: True
       x: 0.6687403059883172
```

Numerische Minimierung in Python: n-dimensional

```python
from scipy.optimize import minimize


def function(x, y):
    return np.cos(x * y + 1.0) + x ** 2 + y ** 2 + x


minimize(lambda p: function(*p), x0=[1.0, 1.0])
```

```
      fun: 0.2293885002366689
 hess_inv: array([[0.70123043, 0.37691145],
       [0.37691145, 0.74614311]])
      jac: array([-9.68575478e-08,  8.94069672e-08])
  message: 'Optimization terminated successfully.'
     nfev: 32
      nit: 7
     njev: 8
   status: 0
  success: True
        x: array([-0.63707889, -0.29551653])
```