III. 10 Optimierungsprobleme

Im Zusammenhang mit Problemen in der Quantenmechanik stellen sich öfters Optimierungsprobleme z. B:

(i) Strukturoptimierung (Born Oppenheimer Näherung)

Für den Hamiltonoperator der Moleküle:

$$H_e = \sum_i^{N_e} -\frac{\hbar^2 \Delta_i}{2 m_e} - \sum_a^{N_K} \sum_i^{N_e} \frac{1}{4\pi\varepsilon_0} \frac{Z_a}{|\underline{r}_i - \underline{R}_a|} + \sum_{i \neq j}^{N_e} \frac{1}{8\pi\varepsilon_0} \frac{1}{|\underline{r}_i - \underline{r}_j|}$$

$$+ \sum_{a b}^{N_K} \frac{1}{8\pi\varepsilon_0} \frac{Z_a Z_b}{|\underline{R}_a - \underline{R}_b|}$$

bisher fixiert.

wurden die Kernposition angenommen!

Begründung: Born – Oppenheimer – Näherung:

Die Masse der Kerne ist sehr viel größer, als die der Elektronen $\Rightarrow$ langsame Bewegung.

- Daher wird die Kernposition $\underline{R}_a$ fixiert, an den Punkt fixiert, an dem es die minimale Gesamtenergie gibt! (Adiabatische Behandlung Elektronen folgen den Kern schnell)
- Bestimmung der Kernposition $\Rightarrow$ Optimierungsproblem bzgl. Kernpositionen $\Rightarrow$ Benötigt wird $E(\underline{R}_a)$ und Algorithmus.
- Taylorentwicklung um Gleichgewichtsposition $\underline{R}_a^0$

$$E(\underline{R}_a^0 + \underline{\Delta R}_a^0) = \sum_{i,j} M_{ij} (\Delta R_a^0)_i (\Delta R_a^0)_j$$

ergibt ein Quadratische Potential $\Rightarrow$ harmonische Oszilator

⇒ Phonon. M wird über die Hess'sche Matrix bestimmt

(kann durch finite Differenzen $E(\_)$ berechnet werden)

(ii) Basissatzoptimierung

Zur Erinnerung, STO oder GTO enthalten Parameter

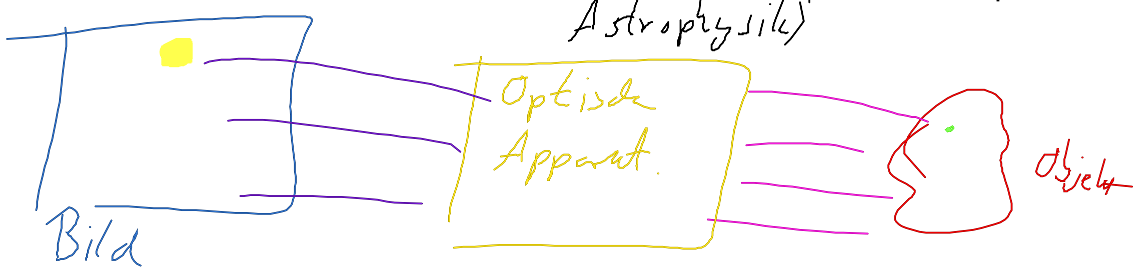
STO: $\chi_{\zeta,n,l,m}(r,\vartheta,\varphi) = N Y_{l,m}(\vartheta,\varphi) r^{n-1} e^{-\zeta r}$
Zeta

Dieses Zeta ζ wurde für den Ansatz der Wellenfunktion optimiert und zwar

$$\langle E(\zeta_1, \dots, \zeta_n)\rangle$$

Diese Funktion wird numerisch berechnet und dann optimiert.

(iii) Beispiel aus der normalen Physik (Exp. Physik, Imaging, Astrophysik)



$$B(x,y) = \int dx' \int dy' \underbrace{PSF(x-x', y-y')}_{\text{Point Spread Funktion}} \; O(x',y')$$

beschreibt wie ein Punkt des Objekts im Bild abgebildet wird.

PSF macht Bild unscharf.

Fassen wir $O(x',y')$ als Vektor auf, so kann als

$$E(O) = \left\| \int dx' \int dy' \, PSF(x-x', y-y') \, O(x',y') - B_{exp}(x,y) \right\|_2$$

$\Rightarrow$ wird $O(x,y)$ variiert und dabei $E(O)$ minimiert

Kann man in gewissen Grenzen Unschärfe reduzieren.

(Eine von vielen Möglichkeiten der Deconvolution Algorithmen)

$\Longrightarrow$ Eine Übersicht über Optimierungsalgorithmen wäre schön!

Algorithmen für Optimierungsprobleme

Bevor es los geht, eine Bemerkung

Quadratische Optimierungsprobleme

z.B. $$Err(\underline{a}) = \sum_j \Big(y_j - \sum_i^N a_i \, x_{ij}\Big)^2$$

wird gesucht

können über ein lineares Gleichungssystem gelöst werden.

Denn

$$\frac{\partial Err(\underline{a})}{\partial a_i} = -2 \sum_j x_{ij} \Big(y_j - \sum_{i'} a_{i'} x_{i'j}\Big) \overset{!}{=} 0$$

$$\Longrightarrow \sum_{i',j} x_{ij} \, x_{i'j} \, a_{i'} = \sum_j x_{ij} \, y_j$$

$$\sum_j x_{ij} \sum_{i'} x_{i'j} \, a_{i'} = \sum_j x_{ij} \, y_j$$

$$\Longrightarrow \sum_i x_{ij} \, a_i = y_j$$

$$\Rightarrow \underline{\underline{X}} \cdot \underline{a} = \underline{y}$$

Kann über $\underline{a} = \underline{\underline{X}}^{-1} \cdot \underline{y}$ gelöst werden.
falls $\underline{\underline{X}}$ invertierbar ist

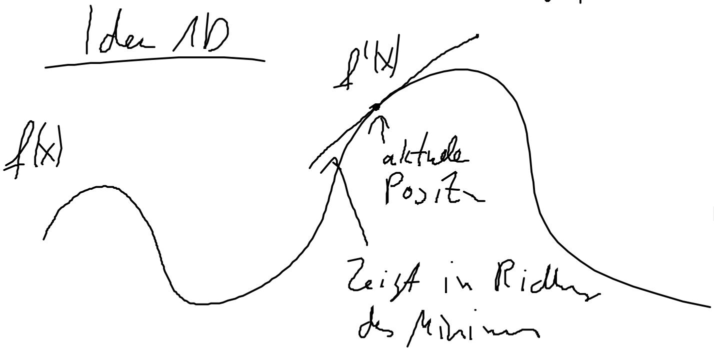
Kann man auch lösen ohne Invertierung, wichtig bei großen Matrizen.

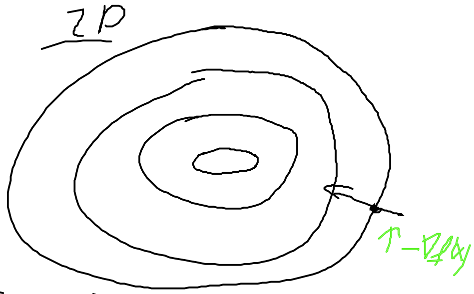
Methoden mit Gradienten:

Steepest descent (einfachste Methode)

Bemerkung: Wir suchen im folgenden Minimum einer Funktion $f(\underline{x})$, die folgenden Methoden finden in der Regel nur lokale Minimum!
(Zusätzliche Strategien für globale Minima notwendig)

Idee 1D





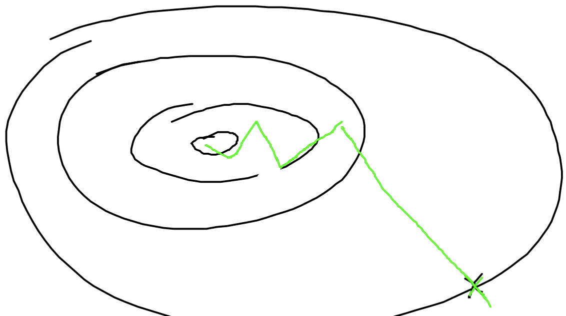
Der negative Gradient zeigt in Richtung des Minimum.

Idee für das Vorgehen:

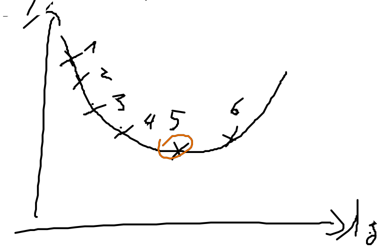
1) Starte auf Punkt x_0
2) Berechne $\underline{g} = -\nabla f(\underline{x}_0)$, das wird unsere Suchrichtung
3) Linesearch berechne: $f(x_0 + \lambda g)$ für verschiedene λ, solange f abnimmt erhöhe λ weiter, sobald f steigt höre auf.

Setze $\underline{x}_0^{(n)} = \underline{x}_0^{(n-1)} + \lambda_{(n)} \underline{g}_{(n)}$

Illustration



f! – 1D Linesearch



<u>Vorteile</u> · Bei guten Linesearch wird steepest descent immer den Funktionswert reduzieren (Garantie sich dem Minimum zu nähern)

· sehr einfach

· Nur ein Gradient muss bestimmt werden.

<u>Nachteile</u> : · Zwei aufeinander folgende Linesearch stehen senkrecht aufeinander

$\Rightarrow$ teilweise wird die erreichte Optimierung in einer Dimension wieder zunichte gemacht.

· Der Algorithmus oszilliert entlang des optimalen Weges.

· Nahe des Minimums nimmt die Konvergenzrate ab (flacher)

<u>Hauptanwendung</u>: Schnell nahe an ein Minimum zu kommen, dann bessere Methode.

Conjugate gradient Methode

Hauptproblem von Steepest Decent, wir unsere Optimierung im nächsten Schritt teilweise rückgängig gemacht haben.

Hier wird versucht, dass die Richtung sich nicht zu abrupt ändert:

$$\underset{\Uparrow}{\underset{-}{d}_i} = \underset{\Uparrow}{\underset{-}{g}_i} + \beta_i \underset{\uparrow}{\underset{-}{d}_{i-1}}$$

Richtung — Aktueller negativer Gradient — letzte Richtung.

Das ist so konstruiert, dass für quadratische Oberfläche, keine Gradientenrichtung entlang vorheriger Richtung entstehen.

Verschiedene Möglichkeiten für β:

Fletcher-Reeves: $\beta_i^{FR} = \dfrac{g_i \cdot g_i}{g_{i-1} \cdot g_{i-1}}$

Polak-Ribiere (PR) $\beta_i^{PR} = \dfrac{g_i \cdot (g_i - g_{i-1})}{g_{i-1} \cdot g_{i-1}}$

Hestenes-Stiefel $\beta_i^{HS} = \dfrac{g_i \cdot (g_i - g_{i-1})}{\underline{d}_{i-1}^t \cdot (g_i - g_{i-1})}$

Wurde anhand einer quadratischen Fkt entwickelt, so dass $\underline{d}_i^t \cdot \underline{\underline{H}} \cdot g_i = 0$, also $\underline{d}$ zu g konjugiert.
Dort sind alle Verfahren äquivalent, im allgemeinen unterschiedliche Vektor.

Komplett super funktioniert CG bei nicht-quadratischen Oberflächen nicht, daher ist ein Neustart notwendig.
(z.B. misst man inwieweit die aufeinanderfolgenden Gradienten nicht orthogonal sind (PR funktioniert besser nach Restart)

Bemerkung:

- Sehr viel bessere Konvergenz als steepest descent.
- Preconditioning auf die Variablen (Koordinatentransformation) kann helfen.
- Nur zwei Vektoren müssen gespeichert werden (einer mehr als Steepest Descent)
- Der Gradient muss bei beiden Methoden berechnet werden, nicht immer ist das analytisch oder in schneller Zeit möglich.