Seminar Numerische Verfahren: Nichtlineare Ausgleichsprobleme

Kazimir Menzel kazimir.menzel@me.com
Markus Pawellek markuspawellek@gmail.com

1 Mathematische Grundlagen

1.1 Idee und Problemstellung

Seien $n \in \mathbb{N}$ und $x, y \in \mathbb{R}^n$, sodass die folgenden Terme Messwerte eines Versuches darstellen.

$$(x_i, y_i) \qquad \text{für alle } i \in \mathbb{N}, i \leq n$$

Weiterhin sei nun $m \in \mathbb{N}$ mit $m \leq n$. Man gehe nun davon aus, dass der Zusammenhang zwischen y und x näherungsweise für mindestens einen Parametervektor $\lambda \in \mathbb{R}^m$ durch die nachstehende Funktion bestimmt ist.

$$p : \mathbb{R} \times \mathbb{R}^m \longrightarrow \mathbb{R}, \qquad y_i \approx p(x_i, \lambda) \quad \text{für alle } i \in \mathbb{N}, i \leq n$$

Das Ziel ist es nun einen Parametervektor $\lambda^\star \in \mathbb{R}^m$ zu finden, der die Messwerte „am besten“ mit der gegebenen Parameterfunktion p approximiert. Hierfür könnte man die Summe der Quadrate der Differenzen minimieren (auch Methode der kleinsten Quadrate).

$$\min \left\{ \sum_{i=1}^{n} \left[y_i - p(x_i, \lambda) \right]^2 \;\middle|\; \lambda \in \mathbb{R}^m \right\}$$

Der dann bestimmbare Zusammenhang

$$f : \mathbb{R} \longrightarrow \mathbb{R}, \qquad f(x) := p(x, \lambda^\star)$$

beschreibt im Sinne der Methode der kleinsten Quadrate gerade die beste Approximation der Messwerte im Bezug auf die Parameterfunktion p. Ein Beispiel ist gerade in Abbildung 1 gegeben.

1.2 Das Ausgleichsproblem

Betrachtet man die obige Problemstellung, ist es hilfreich eine etwas allgemeinere Definition des Ausgleichsproblems zu geben. Denn es lassen sich viele Problemstellungen auf ein Solches zurückführen. Die Verallgemeinerung soll jedoch darauf beschränkt werden, dass eine beste Approximation gerade im Sinne der Methode der kleinsten Quadrate zu verstehen ist.

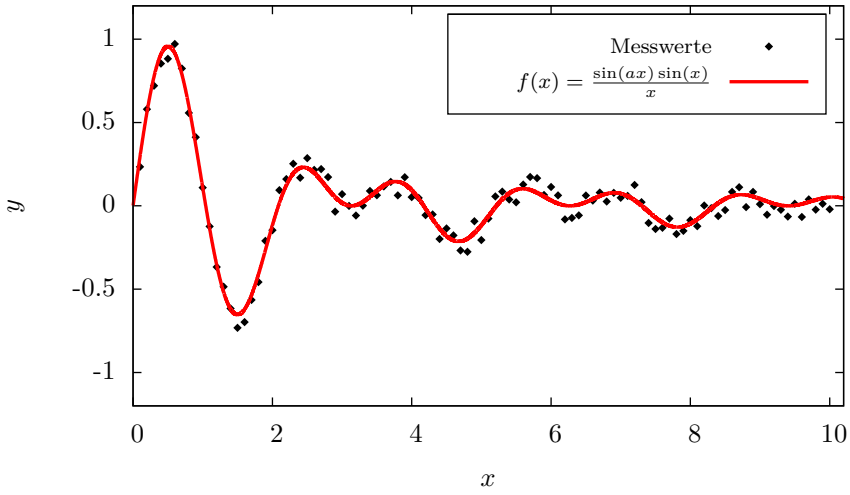

Abbildung 1: Beispiel eines nichtlinearen Ausgleichsproblems mit einem freiem Parameter a, gegebenen Messwerten und der besten Approximation f mit $a = 3.01134$

Definition: (Ausgleichsproblem, Methode der kleinsten Quadrate)

Für $n, m \in \mathbb{N}, m \leq n$, eine gegebene Parametermenge $U \subset \mathbb{R}^m$ und eine gegebene Residuen-Funktion $r : U \longrightarrow \mathbb{R}^n$ ist das Ausgleichsproblem definiert durch

$$\min \left\{ \|r(\lambda)\|_2^2 \;\middle|\; \lambda \in U \right\}$$

2 Lösungsverfahren

Im Folgenden seien nun immer $n, m \in \mathbb{N}, m \leq n, U \subset \mathbb{R}^m$ die gegebene Parametermenge und $r : U \longrightarrow \mathbb{R}^n$ die Residuen-Funktion, wenn nicht anders behauptet. Da wir hier, wie bereits gesagt, nur die Methode der kleinsten Quadrate anwenden, ist es durchaus sinnvoll noch die folgende Funktion zu definieren.

$$s : U \longrightarrow [0, \infty), \qquad s := \tfrac{1}{2} \|r\|_2^2 = \tfrac{1}{2} r^{\mathrm{T}} r$$

Ist s zweimal stetig differenzierbar, folgt durch Ausrechnen

$$\nabla s = (\mathrm{D}\, r)^{\mathrm{T}} r, \qquad \mathrm{D}\,(\nabla s) = (\mathrm{D}\, r)^{\mathrm{T}}\, \mathrm{D}\, r + \sum_{i=1}^{n} r_i\, \mathrm{D}^2\, r_i$$

2.1 Gauß-Newton-Verfahren

Das Gauß-Newton-Verfahren ist ein sogenanntes Quasi-Newton-Verfahren. Die grundsätzliche Idee besteht darin, das gegebene Ausgleichsproblem in ein nichtlineares Gleichungssystem umzuformulieren. Dieses würde sich dann mit dem bereits

bekannten Newton-Verfahren lösen lassen.

Nun zu den Details. Wir betrachten ein zwangloses System $U = \mathbb{R}^m$ von Parametern. r sei zweimal stetig differenzierbar. Die Lösung des Ausgleichsproblems soll hier gerade durch das Auffinden eines lokalen Minimums $\lambda^\star \in U$ mit den folgenden hinreichenden Bedingungen gegeben sein.

Bedingungen für Minimum

$$\nabla s(\lambda^\star) = 0 \tag{1}$$

$$\mathrm{D}\left(\nabla s\right)(\lambda^\star) \text{ ist positiv definit} \tag{2}$$

Durch Einsetzen von s in Bedingung (1) ergibt sich für das gerade definierte $\lambda^\star \in U$

$$\left[\left(\mathrm{D}\, r\right)^{\mathrm{T}} r\right](\lambda^\star) = 0$$

Dies ist nun ein nichtlineares Gleichungssystem, welches mithilfe des Newton-Verfahren gelöst werden kann. Für den k-ten Iterationsschritt mit $k \in \mathbb{N}$ folgt also

$$\lambda^{(k+1)} = \lambda^{(k)} + \xi^{(k)}$$

$$\left[\left(\mathrm{D}\, r\right)^{\mathrm{T}} \mathrm{D}\, r + \sum_{i=1}^{n} r_i\, \mathrm{D}^2\, r_i\right]\left(\lambda^{(k)}\right)\xi^{(k)} = -\left[\left(\mathrm{D}\, r\right)^{\mathrm{T}} r\right]\left(\lambda^{(k)}\right)$$

Um sich die aufwendige Berechnung der zweiten Ableitung zu sparen, geht man davon aus, dass $r\left(\lambda^\star\right) \approx 0$. Damit kann also die Berechnung von $\xi^{(k)}$ in einer geeigneten Umgebung von $\lambda^\star$ durch folgendes ersetzt werden.

$$\left[\left(\mathrm{D}\, r\right)^{\mathrm{T}} \mathrm{D}\, r\right]\left(\lambda^{(k)}\right)\xi^{(k)} = -\left[\left(\mathrm{D}\, r\right)^{\mathrm{T}} r\right]\left(\lambda^{(k)}\right)$$

Algorithmus: Gauß-Newton-Verfahren

Eingabe: $r : \mathbb{R}^m \longrightarrow \mathbb{R}^n, \quad \lambda^{(0)} \in \mathbb{R}^m$

(1) Setze $k = 0$.

(2) Berechne $r\left(\lambda^{(k)}\right), \mathrm{D}\, r\left(\lambda^{(k)}\right)$.

(3) Bestimme den Korrekturvektor $\xi^{(k)}$ gemäß

$$\left[\left(\mathrm{D}\, r\right)^{\mathrm{T}} \mathrm{D}\, r\right]\left(\lambda^{(k)}\right)\xi^{(k)} = -\left[\left(\mathrm{D}\, r\right)^{\mathrm{T}} r\right]\left(\lambda^{(k)}\right)$$

(4) Setze $\lambda^{(k+1)} = \lambda^{(k)} + \xi^{(k)}$.

(5) Setze $k = k + 1$.

(6) Gehe zu Schritt (2), wenn

$$k < k_{\max} \quad \wedge \quad \left\|\xi^{(k)}\right\|_2^2 > \delta_{\min}$$

2.2 Levenberg-Marquardt-Verfahren

Das Levenberg-Marquardt-Verfahren stellt eine Erweiterung des Gauß-Newton-Verfahrens dar.

Algorithmus: Levenberg-Marquardt-Verfahren

Eingabe: $r : \mathbb{R}^m \longrightarrow \mathbb{R}^n, \quad \lambda^{(0)} \in \mathbb{R}^m$

(1) Setze $k = 0$.

(2) Berechne $r\left(\lambda^{(k)}\right), \mathrm{D}\, r\left(\lambda^{(k)}\right)$.

(3) Bestimme den Korrekturvektor $\xi^{(k)}$ gemäß

$$\left[\left(\mathrm{D}\, r\right)^{\mathrm{T}} \mathrm{D}\, r + \mu^2 \mathrm{I}\right]\left(\lambda^{(k)}\right) \xi^{(k)} = -\left[\left(\mathrm{D}\, r\right)^{\mathrm{T}} r\right]\left(\lambda^{(k)}\right)$$

(4) Berechne ε_μ und teste, ob die Korrektur akzeptabel ist.

$$\varepsilon_\mu = \frac{\left\|r\left(\lambda^{(k)}\right)\right\|_2^2 - \left\|r\left(\lambda^{(k)} + \xi^{(k)}\right)\right\|_2^2}{\left\|r\left(\lambda^{(k)}\right)\right\|_2^2 - \left\|r\left(\lambda^{(k)}\right) + \mathrm{D}\, r\left(\lambda^{(k)}\right)\xi^{(k)}\right\|_2^2}$$

(i) Fall $\varepsilon_\mu \leq \beta_0$: Setze $\mu = 2\mu$ und gehe zu (3).

(ii) Fall $\varepsilon_\mu \geq \beta_1$: Setze $\mu = \frac{\mu}{2}$.

(5) Setze $\lambda^{(k+1)} = \lambda^{(k)} + \xi^{(k)}$.

(6) Setze $k = k + 1$.

(7) Gehe zu Schritt (2), wenn

$$k < k_{\max} \quad \wedge \quad \left\|\xi^{(k)}\right\|_2^2 > \delta_{\min}$$

2.3 Simulated Annealing

Simulated Annealing stellt ein Verfahren dar, welches im Vergleich zu den zuvor genannten Verfahren ein anderen Ansatz verfolgt. Anstatt sich auf das lokale Minimum zu beschränken, versucht es mithilfe von Zufallszahlen das globale Minimum ausfindig zu machen.

Algorithmus: Simulated Annealing

Eingabe: $r : \mathbb{R}^m \longrightarrow \mathbb{R}^n, \quad \lambda_0 \in \mathbb{R}^m$

(1) Berechne $c_0 = \|r\left(\lambda_0\right)\|_2^2$.

(2) Setze $T = T_0$ und $i = 0$.

(3) Bestimme einen zufälligen Parameter λ_1.

(4) Berechne $c_1 = \|r\left(\lambda_1\right)\|_2^2$.

(5)
- Fall $c_1 < c_2$: Setze $\lambda_0 = \lambda_1$.

- Fall $c_1 \geq c_2$: Berechne

$$p = \exp\left(\frac{c_0 - c_1}{T}\right)$$

und setze $\lambda_0 = \lambda_1$ mit Wahrscheinlichkeit p.

(6) Setze $i = i + 1$ und gehe zu Schritt (3), wenn $i < i_{\max}$.

(7) Setze $T = \alpha T$ und gehe zu Schritt (3), wenn $T > T_{\min}$.

C++-Implementierung: Curve Fitting mit Simulated Annealing

```
#include "sa_fit.h"


void sa_fit(float *sample_x, float *sample_y, uint sample_count,
    paramff param_func, uint param_count, float *param){
    float old_cost = sqnorm(sample_x, sample_y, sample_count,
        param_func, param);
    float temp = sa_temp_max;

    while (temp > sa_temp_min){
        for (uint i = 0; i < sa_it_max; i++){

            const uint rand_param_idx = rand()%param_count;
            const float tmp_param = param[rand_param_idx];
            param[rand_param_idx] += (1.0f - 2.0f*(float)rand()/(
                float)RAND_MAX);

            float new_cost = sqnorm(sample_x, sample_y,
                sample_count, param_func, param);

            if (new_cost > old_cost){
                const float prob = exp((old_cost - new_cost)/temp);

                if (prob < (float)rand()/(float)RAND_MAX){
                    param[rand_param_idx] = tmp_param;
                }else{
                    old_cost = new_cost;
                }
            }else{
                old_cost = new_cost;
            }

        }

        temp *= sa_temp_factor;
    }
}
```

3 Referenzen

- Hermann, *Numerische Mathematik*, 3.Auflage
- `http://en.wikipedia.org/wiki/Gauss-Newton_algorithm`

- http://en.wikipedia.org/wiki/Levenberg-Marquardt_algorithm
- http://en.wikipedia.org/wiki/Simulated_annealing
- Funken, *Numerik III*, Skript Universität Ulm, 2012/2013
- katrinaeg.com/simulated-annealing.html
- Kincaid und Cheney, *Numerical Analysis*, 3.Edition