Programmierblatt 2.

Besprechungswoche: **06.11. – 10.11.2023**

Wir wollen uns in diesem Blatt der numerischen Lösung von Ausgleichsproblemen widmen, welche beispielsweise bei inversen Problemen auftreten. In solchen weist ein Modell unbekannte Parameter auf. Die Auslenkung eines gedämpften Federpendels etwa genügt der Beziehung

$$x(t) = e^{-\beta t}\left(x_0 \cos \omega t + (x_0\beta + \dot{x}_0)\frac{\sin \omega t}{\omega}\right),$$

wobei die x_0 die Anfangsauslenkung, $\dot{x}_0$ die Anfangsgeschwindigkeit, β den Dämpfungsfaktor sowie $\omega = \sqrt{\omega_0^2 - \beta^2}$ die gedämpfte Schwingungsfrequenz bezeichne. Dabei hängt der Dämpfungsfaktor β vom Medium ab, in welchem die Masse schwingt. Ist die Grösse ω_0 sowie die Anfangsgrössen x_0, $\dot{x}_0$ bekannt, so kann der Dämpfungsfaktor aus experimentellen Messungen $x(t_j) = x_j$, $j = 1, \dots, N$, rekonstruiert werden.

Etwas allgemeiner bezeichne $f(\mathbf{x}, \boldsymbol{\theta})$ ein Modell, $\mathbf{x}_j$ Eingaben, $\boldsymbol{\theta}$ unbekannte Parameter, und $\mathbf{y}_j$ Messwerte, welche der Beziehung $f(\mathbf{x}_j, \boldsymbol{\theta}) = \mathbf{y}_j$, $j = 1, \dots, N$ genügen sollen. Dabei sollen die Parameter $\boldsymbol{\theta} \in \mathbb{R}^p$ so bestimmt werden, dass

$$\left\| \begin{bmatrix} f(\mathbf{x}_1, \boldsymbol{\theta}) - \mathbf{y}_1 \\ \vdots \\ f(\mathbf{x}_N, \boldsymbol{\theta}) - \mathbf{y}_N \end{bmatrix} \right\|_2 \to \min. \tag{1}$$

Für das obige Modell des gedämpften, harmonischen Oszillators wäre $\mathbf{x}_j = t_j$, $\mathbf{y}_j = x_j$ sowie $\boldsymbol{\theta} = \beta$.

Lineares Ausgleichsproblem

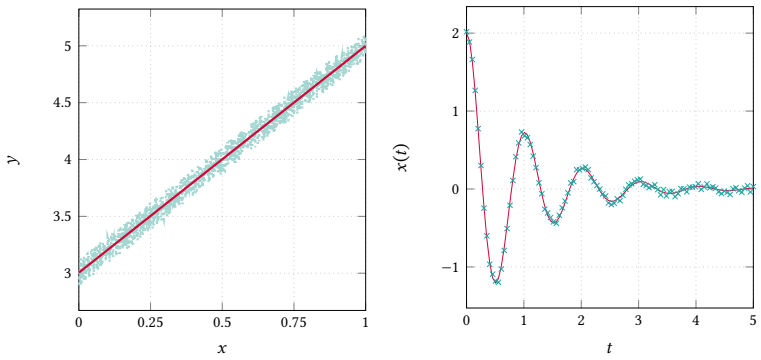
Ein lineares Ausgleichsproblem lässt sich leicht mit dem *CGLS-Verfahren* lösen. Voraussetzung dafür ist, dass das Modell f in (1) linear ist, das heisst, es existiert eine Matrix $\mathbf{A}_{\boldsymbol{\theta}} \in \mathbb{R}^{d_2 \times d_1}$ sowie ein Vektor $\mathbf{b}_{\boldsymbol{\theta}} \in \mathbb{R}^{d_2}$, so dass für $\mathbf{x} \in \mathbb{R}^{d_1}$ gilt $f(\mathbf{x}_j, \boldsymbol{\theta}) = \mathbf{A}_{\boldsymbol{\theta}}\mathbf{x} + \mathbf{b}_{\boldsymbol{\theta}}$.

Aufgabe 1. Schreiben Sie eine Methode

```
function thetas = cgls(A, y, theta0, tol, maxIter),
```

welche das CGLS-Verfahren gemäss Skript implementiert.

Aufgabe 2. Laden Sie die Datei `Data.zip` herunter und testen Sie Ihre Implementierung mit der Datei `test.mat`. Gesucht ist der Vektor $\boldsymbol{\theta} = (a, b)$, so dass $f(\mathbf{x}; a, b) = ax + b$ ist. Die Arrays `x`, `y` $\in \mathbb{R}^N$ enthalten dabei die Ein- respektive die Ausgabewerte. Stellen Sie die Punkte (x_j, y_j) sowie ihre Annäherung an f graphisch dar.



Nichtlineares Ausgleichsproblem

Während ein lineares Ausgleichsproblem mittels einer QR-Zerlegung noch direkt gelöst werden könnte, ist das für nichtlineare Ausgleichsprobleme, welche im Allgemeinen nur iterativ gelöst werden können, nicht mehr der Fall. In der Vorlesung behandeln wir dafür das *Gauß-Newton*- sowie das *Levenberg-Marquardt-Verfahren*.

Im Levenberg-Marquardt-Verfahren müssen möglicherweise Optimierungsprobleme mit Nebenbedingungen gelöst werden. Ist $\mathbf{r}_k = \mathbf{y} - \mathbf{f}(\mathbf{x}_k)$ das Residuum im k-ten Schritt, dann suchen wir $\mathbf{d} \in \mathbb{R}^n$, so dass

$$\left\|\mathbf{r}_k - \mathbf{f}'(\mathbf{x}_k)\mathbf{d}\right\|_2^2 = \min_{\substack{\mathbf{h}\in\mathbb{R}^n \\ \|\mathbf{h}\|_2 \le \Delta}} \left\|\mathbf{r}_k - \mathbf{f}'(\mathbf{x}_k)\mathbf{h}\right\|_2^2.$$

Liegt dabei das Minimum des Optimierungsproblems ausserhalb eines Kreises mit Radius Δ, muss das Minimum aufgrund der Linearität des Problems auf dem Rand des Kreises liegen. Solche Optimierungsprobleme mit Gleichheitsnebenbedingungen lassen sich mit der Methode der *Lagrange-Multiplikatoren* lösen. Iterativ lassen sich die Näherungen an den Lagrange-Multiplikator im k-ten Schritt mit dem *Hebden-Verfahren* berechnen. Für eine detailierte Herleitung siehe [1].

Algorithmus Hebden-Verfahren

Input: Startwert $\lambda^{(0)} \in \mathbb{R}$, Nebenbedingung $\rho \in \mathbb{R}$, Genauigkeit `tol`
Output: Näherungen an den Lagrange-Multiplikator λ_k

1: Setze $\mathbf{A} := \mathbf{f}'(\mathbf{x}_k)$, $\mathbf{b} := \mathbf{y} - \mathbf{f}(\mathbf{x}_k)$, $j := 0$
2: Setze $\mathbf{h} := (\mathbf{A}^\intercal\mathbf{A} + \lambda^{(j)}\mathbf{I})^{-1}\mathbf{A}^\intercal\mathbf{b}$
3: Setze $\mathbf{g} := (\mathbf{A}^\intercal\mathbf{A} + \lambda^{(j)}\mathbf{I})^{-1}\mathbf{h}$
4: Setze $\lambda^{(j+1)} := \lambda^{(j)} - \|\mathbf{h}\|_2^2 \frac{1-\|\mathbf{h}\|_2/\rho}{\mathbf{h}^\intercal\mathbf{g}}$
5: **if** $\left|\lambda^{(j+1)} - \lambda^{(j)}\right| <$ `tol` **then**
6: setze $\lambda_k := \lambda^{(j+1)}$
7: `break;`
8: **else**
9: erhöhe $j := j + 1$ und gehe nach 2.
10: **end if**

Aufgabe 3. Schreiben Sie zwei Methoden

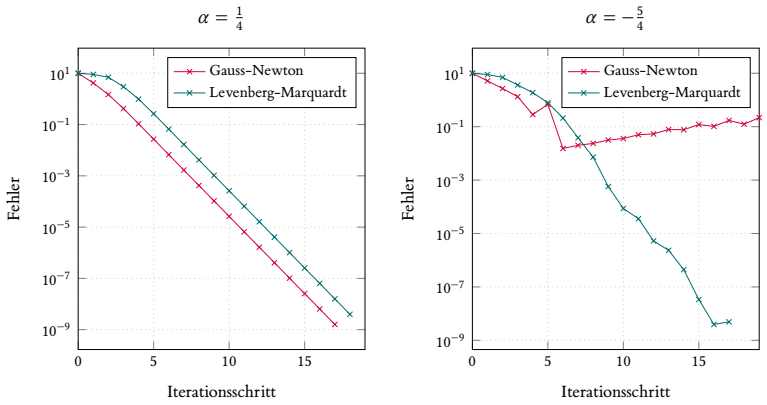
```
function thetas = GaußNewton(f, Df, theta0, y, tol, maxIter),
function thetas = levenbergMarquardt(f, Df, mus, theta0, y, ...
                    Delta0, tol, maxIter),
```

welche das Gauß-Newton- respektive das Levenberg-Marquardt-Verfahren implementieren. Die Lagrange-Multiplikatoren λ_k des Levenberg-Marquardt-Verfahrens sollen dabei mit dem

Hebden-Verfahren berechnet werden. Testen Sie Ihre Implementierungen mit den Parametern `tol` $= \sqrt{\texttt{eps}}$, `maxIter` $= 25$ sowie (für das Levenberg-Marquardt-Verfahren) $\mu^- = \frac{1}{4}$, $\mu^+ = \frac{1}{2}$, $\Delta_0 = 1$ wie folgt:

(a) In der Datei `Oscillator.mat` befinden sich Zeiten `ts` und Auslenkungen `xs` eines gedämpften, harmonischen Oszillators. Bekannt sind die Werte $\omega_0 = 2\pi$, $x_0 = 2$ sowie $\dot{x}_0 = 0$. Berechnen Sie den Dämpfungsfaktor β, wählen Sie als Startwert $\beta_0 = 2$. Stellen Sie die Punkte sowie die Regressionskurve graphisch dar.

(b) Verwenden Sie die Funktion $\mathbf{f}(x) = \begin{bmatrix} x+1, & \alpha x^2 + x - 1 \end{bmatrix}^\mathsf{T}$, für ein $\alpha \in \mathbb{R}$. Testen Sie Ihre Implementierung einmal für $\alpha_1 = \frac{1}{4}$ sowie für $\alpha_2 = -\frac{5}{4}$. Als Startwert wählen Sie $x_0 = 10$. Plotten Sie Ihren Fehler zur analytischen Lösung $x = 0$ gegen den Iterationsschritt.



SIR-Modell

Eine einfache Möglichkeit zur Modellierung einer Epidemie bietet das SIR-Modell. Die Population wird hierbei in die drei Klassen der gesunden, aber noch nie erkrankten Personen (susceptible), der erkrankten Personen (infectious) und der genesenen Personen (recovered) eingeteilt. Dabei nehmen wir an, dass der zeitabhängige Datenvektor $\mathbf{y}(t) = \big(S(t), I(t), R(t)\big)^\mathsf{T}$ der gewöhnlichen Differentialgleichung

$$\mathbf{y}'(t) = \mathbf{f}\big(\mathbf{y}(t)\big), \quad \mathbf{f}(\mathbf{y}) = \begin{bmatrix} -\beta SI \\ \beta SI - \gamma I \\ \gamma I \end{bmatrix} = \begin{bmatrix} -\beta y_1 y_2 \\ \beta y_1 y_2 - \gamma y_2 \\ \gamma y_2 \end{bmatrix}, \tag{2}$$

für unbekannte, zu bestimmende Parameter $\beta, \gamma > 0$ genüge. Möglicherweise ungenaue Messungen der Populationsgrössen zu bestimmten Zeitpunkten $t_0, \dots, t_N$ finden Sie in der Datei `SIR.mat`.

Die Differentialgleichung (2) kann näherungsweise gelöst werden. Konkret benutzen wir die Trapezregel, um das Integral

$$\mathbf{y}(t_{j+1}) = \mathbf{y}(t_j) + \int_{t_j}^{t_{j+1}} \mathbf{y}'(s)\,\mathrm{d}s \approx \mathbf{y}(t_j) + \frac{t_{j+1} - t_j}{2}\big(\mathbf{y}'(t_{j+1}) + \mathbf{y}'(t_j)\big)$$

anzunähern. Rekursiv ergibt sich dann

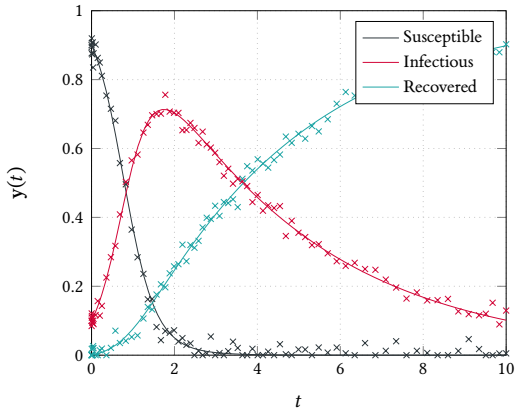
$$\begin{aligned}
\mathbf{y}(t_k) &\approx \mathbf{y}(t_{k-1}) + \frac{t_k - t_{k-1}}{2}\left(\mathbf{y}'(t_k) + \mathbf{y}'(t_{k-1})\right) \\
&\approx \mathbf{y}(t_{k-2}) + \frac{t_{k-1} - t_{k-2}}{2}\left(\mathbf{y}'(t_{k-1}) + \mathbf{y}'(t_{k-2})\right) + \frac{t_k - t_{k-1}}{2}\left(\mathbf{y}'(t_k) + \mathbf{y}'(t_{k-1})\right) \\
&\vdots \\
&\approx \mathbf{y}(t_0) + \frac{1}{2}\sum_{j=1}^{k}(t_j - t_{j-1})\left(\mathbf{y}'(t_j) + \mathbf{y}'(t_{j-1})\right). \qquad (3)
\end{aligned}$$

Setzen wir nun (2) in (3) ein, so ist es naheliegend, folgende Beziehung zu fordern:

$$\begin{aligned}
\begin{bmatrix} y_1(t_k) - y_1(t_0) \\ y_2(t_k) - y_2(t_0) \\ y_3(t_k) - y_3(t_0) \end{bmatrix} &= \frac{1}{2}\sum_{j=1}^{k}(t_j - t_{j-1}) \begin{bmatrix} -\beta y_1(t_j) y_2(t_j) \\ \beta y_1(t_j) y_2(t_j) - \gamma y_2(t_j) \\ \gamma y_2(t_j) \end{bmatrix} \\
&\quad + (t_j - t_{j-1}) \begin{bmatrix} -\beta y_1(t_{j-1}) y_2(t_{j-1}) \\ \beta y_1(t_{j-1}) y_2(t_{j-1}) - \gamma y_2(t_{j-1}) \\ \gamma y_2(t_{j-1}) \end{bmatrix}, \qquad k = 1, \dots, N. \qquad (4)
\end{aligned}$$

Aufgabe 4. Berechnen Sie die Parameter mit einem Verfahren Ihrer Wahl. Visualisieren Sie die Daten sowie ihre berechnete Lösung beispielsweise wie folgt mit der Matlab-Funktion `ode45`.

```
load('SIR.mat');
% TODO: Implementieren
f = @(t, y) [-beta*y(1)*y(2); beta*y(1)*y(2) - gamma*y(2);
        gamma*y(2)];
[ts_app, ys_app] = ode45(f, [ts(1), ts(end)], ys(:, 1));
ts_app = ts_app.';
ys_app = ys_app.';
plot(ts, ys, 'x', ts_app, ys_app);
```



Aufgabe 5. Experimentieren Sie mit den Parametern. Nutzen Sie die Funktion `ode45`, um sich selbst Daten zu generieren. Wählen Sie sich ein `delta` > 0 und setzen Sie `ys = ys + delta * randn(size(ys))`. Lösen Sie danach das Ausgleichsproblem. Stellen Sie die Daten sowie die berechnete Lösung graphisch dar.

Literatur

[1] Martin Hanke-Bourgeois. *Grundlagen der Numerischen Mathematik und des Wissenschaftlichen Rechnens*. Vieweg + Teubner Verlag, Wiesbaden, dritte Edition, 2009.