Programmierblatt 1.

Bearbeiten bis: **Sonntag, 19.03.2023**

Auf diesem ersten Programmierblatt wollen wir uns mit der Rekonstruktion von geschlossenen, orientierbaren Oberflächen befassen: Gegeben seien gestreute Punkte, welche auf einer unbekannten Oberfläche liegen. Die Punkte genügen dabei keiner spezifischen Anordnung – sie sind daher *gitterfrei*. Zwei praktische Anwendungen hiervon sind etwa die Rekonstruktion einer Gletscherobefläche, ausgehend von diskreten Höhenlinen oder Punkten, oder der Laserscan eines Körpers, was bei der Herstellung von Prothesen hilfreich ist.

Ausgehend von gegebenen Punkten auf der besagten Oberfläche generieren wir Punkte innerhalb und ausserhalb des von der Oberfläche umschlossenen Volumens und versuchen anschliessend, eine stetige Funktion $f : \mathbb{R}^s \to \mathbb{R}$ zu lernen, welche die Bedingung

$$f(\mathbf{x}) \begin{cases} < 0, & \mathbf{x} \text{ liegt im Innern,} \\ = 0, & \mathbf{x} \text{ befindet sich auf der Oberfläche,} \\ > 0, & \mathbf{x} \text{ liegt ausserhalb des Körpers,} \end{cases}$$

erfüllen soll. Eine solche Funktion wird *Levelset-Funktion* genannt.

Ansatzfunktion

Gegeben seien die Punkte $\mathbf{p}_1, \dots, \mathbf{p}_N$ auf einer Oberfläche $\Gamma \subset \mathbb{R}^s$ sowie eine Kernfunktion $k : \mathbb{R}^s \times \mathbb{R}^s \to \mathbb{R}$. Wir wollen dabei annehmen, dass k eine radiale Funktion sei, das bedeutet, dass $k(\mathbf{x}, \mathbf{y}) = k(r)$ mit $r := \|\mathbf{x} - \mathbf{y}\|_2$ gilt. Hierdurch wird die resultierende Kernmatrix symmetrisch. Ein wichtiges Beispiel hierfür ist die Klasse der *Matérn*-Kerne, welche exemplarisch durch

$$\begin{aligned} k_{1/2}(\mathbf{x}, \mathbf{y}) &= \frac{1}{\sigma} e^{-\frac{\|\mathbf{x}-\mathbf{y}\|_2}{\sigma}}, \\ k_{3/2}(\mathbf{x}, \mathbf{y}) &= \frac{1}{\sigma} \left(1 + \sqrt{3} \frac{\|\mathbf{x} - \mathbf{y}\|_2}{\sigma} \right) e^{-\sqrt{3}\frac{\|\mathbf{x}-\mathbf{y}\|_2}{\sigma}}, \\ k_{\infty}(\mathbf{x}, \mathbf{y}) &= \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{\|\mathbf{x}-\mathbf{y}\|_2^2}{2\sigma^2}}, \end{aligned}$$

gegeben sind, wobei $\sigma > 0$ die Korrelationslänge bezeichne. Hierbei wird $k_{1/2}$ auch als *Exponentialkern* und k_∞ als *Gauss-Kern* bezeichnet.

Kennen wir zusätzlich noch Punkte $\mathbf{p}_{N+1}, \dots, \mathbf{p}_{N+N_i}$ und $\mathbf{p}_{N+N_i+1}, \dots, \mathbf{p}_{N+N_i+N_o}$, welche innerhalb beziehungsweise ausserhalb des von der Oberfläche Γ umschlossenen Volumens liegen, so setzen wir die Funktion f als

$$f(\mathbf{x}) := \sum_{\ell=1}^{N+N_i+N_o} c_\ell k(\mathbf{x}, \mathbf{p}_\ell) \tag{1}$$

an. Genügen die Koeffizienten dem Gleichungssystem

$$\mathbf{K} \cdot \begin{bmatrix} c_1 \\ \vdots \\ c_N \\ c_{N+1} \\ \vdots \\ c_{N+N_i} \\ c_{N+N_i+1} \\ \vdots \\ v_{N+N_i+N_o} \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ -1 \\ \vdots \\ -1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}, \qquad \mathbf{K} := \left[k(\mathbf{p}_i, \mathbf{p}_j)\right]_{i,j=1}^{N+N_i+N_o},$$

so definiert f näherungsweise eine Funktion, welche wie gewünscht auf der Obefläche den Wert 0, innerhalb einen negativen sowie ausserhalb einen positiven Wert annimmt. Falls $\mathbf{p}_\ell \approx \mathbf{p}_{\ell'}$ ist, dann stimmen die ℓ-ten und die ℓ'-ten Zeilen und Spalten nahezu überein und die Kondition verschlechtert sich extrem. Daher müssen wir das System regularisieren: Wir lösen ersatzweise wir das System

$$(\mathbf{K} + \alpha\mathbf{I})\mathbf{c} = [\mathbf{0}, -\mathbf{1}, \mathbf{1}]^\intercal$$

mit einem hinreichend kleinen Raglarisierungsparameter $\alpha > 0$. Diese Prozedur wird auch *Tichonow*-Regularisierung genannt.

Aufgabe 1. Schreiben Sie eine Funktion

```
function cs = get_coefficients(k, p_surf, p_in, p_out, alpha),
```

welche für ein `function handle` k die Kernmatrix $\mathbf{K}$ aufstellt und anschliessend die Koeffizienten über das mit α regularisierte Gleichungssystem berechnet.

Generierung der Punkte

Um die Punkte inner- und ausserhalb der Oberfläche zu generieren, gibt es verschieden Möglichkeiten. Wir möchten gerne reproduzierbare Ergebnisse, aber trotzdem keine uniformen Gitter. Einen Kompromiss bieten die *Halton-Punkte*, welche quasi-uniform verteilte Punkte auf dem Intervall $[0, 1]$ definieren.

Im Folgenden bezeichne p eine Primzahl. Es ist bekannt, dass jedes $n \in \mathbb{N}$ eine eindeutige p-adische Darstellung

$$n = \sum_{j=0}^{J} a_j p^j$$

besitzt. Durch diese Darstellung definieren wir die Halton-Folge

$$H_p^N = \{h_p(n) \,:\, 0 \le n \le N\}, \qquad h_p(n) := \sum_{j=0}^{J} \frac{a_j}{p^{j+1}}.$$

Beispielsweise ist

$$H_3^{10} = \left\{0, \frac{1}{3}, \frac{2}{3}, \frac{1}{9}, \frac{4}{9}, \frac{7}{9}, \frac{2}{9}, \frac{5}{9}, \frac{8}{9}, \frac{1}{27}, \frac{10}{27}\right\}.$$

Auf $[0, 1]^s$ definieren wir die Halton-Punkte für Primzahlen $p_1, \dots, p_s$ als

$$H_{\mathbf{p}}^N = \left\{ \left(h_{p_1}(n), \dots, h_{p_s}(n)\right)^\intercal \,:\, 0 \le n \le N \right\}.$$

Um Eckpunkte zu vermeiden, entfernen wir zum Schluss noch den Ursprung.

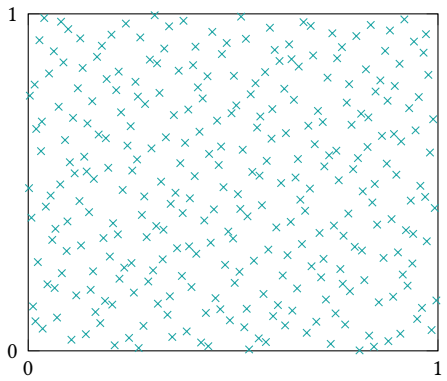

Abbildung 1: 300 Halton Punkte auf dem Einheitsquadrat bezüglich der Primzahlen 2 und 3.

Aufgabe 2. Schreiben Sie eine Funktion

```
function hs = haltonseq(s, ps, N),
```

welche N Halton-Punkte auf $[0,1]^s$ bezüglich der Primzahlen `ps` generiert.

Hinweis. *Schreiben Sie zuerst eine Funktion, welche die Halton-Punkte bezüglich einer Primzahl p in einer Dimension berechnet und fügen Sie die Punkte erst anschliessend zusammen.*

Aufgabe 3. Schreiben Sie eine Funktion

```
function [ps_in, ps_out] = generate_points(geometry, N_in, N_out),
```

welche für einen gegebenen String `geometry` Punkte innerhalb und ausserhalb generiert. Übergeben werden können sollen dabei `sphere`, `pyramid`, `cat` und `pig`, welche sich auf die Geometrien in der Beilage beziehen. Die Punkte innerhalb sollen dabei dreidimensionale Halton-Punkte in einem Quader sein, die Punkte ausserhalb zweidimensionale Halton-Punkte, welche auf eine Quaderoberfläche transformiert wurden.

Visualisierung der Oberfläche

Mit den obigen Funktionen haben wir das Werkzeug, um die Funktion f gemäss (1) zu berechnen. Nun möchten wir noch die Oberfläche visualisieren, welche durch die Menge

$$\{\mathbf{x} \in \mathbb{R}^3 \,:\, f(\mathbf{x}) = 0\} \tag{2}$$

gegeben ist. Eine Möglichkeit dazu ist die Verwendung der Matlab-Funktion `isosurface`. Ist $[\mathbf{X}, \mathbf{Y}, \mathbf{Z}]$ ein dreidimensionales `meshgrid`, und enthält $\mathbf{V}$ die Werte von f an (x, y, z), d.h.,

```
V(i,j,k) = f([X(i,j,k), Y(i,j,k), Z(i,j,k)]),
```

visualisiert der Befehl `isosurface(X,Y,Z,V,0)` näherungsweise die Oberfläche (2).

Aufgabe 4. Schreiben Sie eine Funktion

```
function V = evaluate_f(X,Y,Z,ps,cs,sigma,ker),
```

welche die Funktion f aus (1) auf dem Meshgrid $[\mathbf{X}, \mathbf{Y}, \mathbf{Z}]$ auswertet. Die Variable `ker` soll dabei als `String` übergeben werden. Achten Sie darauf, die Summe vektorisiert auszuwerten. Beachten Sie auch, dass Sie auch auf ein dreidimensionales Array mit nur einem Index zugreifen können. Somit sollten Sie für jeden Kern nur eine einzige for-Schleife benötigen.

Hinweis. *Die Funktion* `sum` *ist hierbei hilfreich.*

Aufgabe 5. Schreiben Sie ein Skript, welches eine gegebene Geometrien mit dem Befehl `isosurface` visualisiert. Plotten Sie die Oberfläche in dem Quader, den Sie für die äusseren Punkte gewählt haben. Verwenden Sie dazu ein `meshgrid` mit $N_{\text{eval}} \geq 50$ Punkten in jeder Raumdimension. Probieren Sie die verschiedenen Kerne auf den verschiedenen Geometrien sowie verschiedene Korrelationslängen σ aus. Was fällt Ihnen auf?

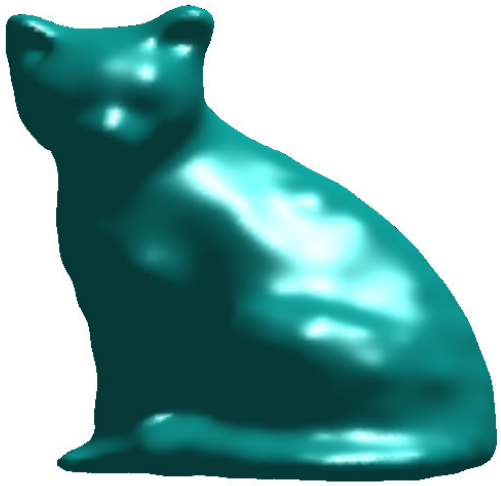

Abbildung 2: Die `cat`-Oberfläche mit dem Matérn-$\frac{3}{2}$-Kern, $\sigma = 0.7$, $\texttt{tol} = 10^{-3}$, $N_{\text{eval}} = 100$.

Mex-Files*

Für grössere N_{eval} beansprucht die Auswertung der Kernfunktion viel Zeit. Diese Auswertung kann mit einer effizienten Implementierung beschleunigt werden. Eine Möglichkeit dazu bieten `mex`-Files[1]. Diese Dateien werden in einer schnellen Programmiersprache, beispielsweise `C` geschrieben, danach kompiliert und können so aus `MATLAB` aufgerufen werden.

Aufgabe 6 (Freiwillig). Kompilieren Sie die Datei `f_exp.c` in Matlab mit dem Befehl

```
mex f_exp.c.
```

Dafür muss ein `C`-compiler, beispielsweise `gcc` auf Ihrem Computer installiert sein. Sie erhalten als Output eine Datei `f_exp.mex***`, welche den `C`-Code ausführt. Die Endung `***` hängt dabei von Ihrem Betriebssystem ab. Die fertige `mex`-Funktion kann dann von `MATLAB` aus wie ein `m`-File aufgerufen werden:

```
V = f_exp(x,y,z,ps,cs,sigma);  V = reshape(Vm,Neval,Neval,Neval);
```

Dabei sind `x`, `y`, `z` die Vektoren, welche Sie auch dem `meshgrid`-Befehl übergeben. Vervollständigen Sie anschliessend die anderen Kerndateien und kompilieren Sie diese. Damit sind Sie in der Lage, die Funktion f aus (1) schneller auszuwerten.

[1] vgl. `https://www.mathworks.com/help/matlab/ref/mex.html`