Prof. Dr. B. Haasdonk
Dr. Y. Cheng
Dr. M. Nottoli

Mathematische Programmierung für LA

Universität Stuttgart
IANS

03.06.2024

Phase 1

Kurveninterpolation

1 Einleitung

In diesem Projekt soll eine beliebige, geschlossene, zwei-dimensionale Kurve $c : I \subset \mathbb{R} \to \mathbb{R}^2$ interpoliert werden. Anwendung finden solche Verfahren z.B. bei Nutzereingaben auf Touchscreens. Das Problem ist, dass die Nutzereingabe nur an diskreten Punkten gespeichert werden kann. Um diese Punkte dann zu einer stetigen oder sogar differenzierbaren Kurve zu verbinden, wird Interpolation genutzt. In diesem Projekt soll als Verfahren die trigonometrische Interpolation auf dem Intervall $I = [0, 2\pi]$ genutzt werden.

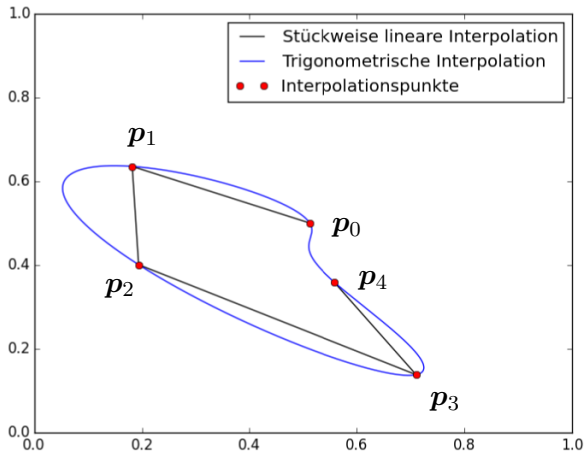

Abbildung 1: Stückweise lineare (schwarz) und trigonometrische (blau) Interpolation einer zwei-dimensionalen Kurve für 5 gegebene Interpolationspunkte $\boldsymbol{p}_i$, $0 \leq i \leq 4$ (rot) in geschlossener oder offene Weise.

Wir nehmen $n + 1$ diskrete Punkte $\boldsymbol{p}_i := (p_{i,x}, p_{i,y})^T \in \mathbb{R}^2$, $0 \leq i \leq n$, als gegeben an. Es soll eine stetige Kurve $c : [0, 2\pi] \to \mathbb{R}^2$ gefunden werden, sodass für jeden Punkt $\boldsymbol{p}_i$ ein $t_i \in [0, 2\pi]$ existiert, sodass

$$c(t_i) = \boldsymbol{p}_i \qquad \text{für alle } 0 \leq i \leq n \tag{1}$$

gilt (siehe Abbildung 1). Das heißt, die Kurve verläuft durch die Punkte $\boldsymbol{p}_i$. Für unsere Anwendung fordern wir zusätzlich, dass die Punkte $\boldsymbol{p}_i$ in der Reihenfolge $i = 0, 1, \ldots, n$ durchlaufen werden. D.h. es gilt zusätzlich zu Gleichung (1) noch $t_0 < t_1 < \cdots < t_n$. Die Kurveninterpolation führt auf ein Interpolationsproblem für die Komponentenfunktionen $c_x, c_y : [0, 2\pi] \to \mathbb{R}$. Genauer wird gefordert, dass die interpolierende Kurve

$$c(t) = \begin{bmatrix} c_x(t) \\ c_y(t) \end{bmatrix} \qquad \text{und} \qquad \begin{aligned} c_x(t_i) &= p_{i,x}, \\ c_y(t_i) &= p_{i,y} \end{aligned}$$

für alle Punkte erfüllt. In diesem Projekt soll nun zunächst die trigonometrische Interpolation verwendet werden, die im Nachfolgenden vorgestellt wird.

Es seien die Interpolationspunkte $(t_i, z_i) \in [0, 2\pi] \times \mathbb{R}$ mit $t_i = \frac{2\pi}{n+1} i$ für $0 \leq i \leq n$ gegeben. Die 2π-periodische Funktion $f : [0, 2\pi] \to \mathbb{R}$

$$f(t) = \frac{a_0}{2} + \sum_{k=1}^{m} \Big(a_k \cos(kt) + b_k \sin(kt) \Big) + \frac{\theta}{2} a_{m+1} \cos((m+1)t),$$

mit den Koeffizienten $a_k, b_k \in \mathbb{R}$ und $\theta \in \{0, 1\}$, welche durch

$$m := \begin{cases} \frac{n}{2}, & \text{falls } n \text{ gerade} \\ \frac{n-1}{2}, & \text{falls } n \text{ ungerade} \end{cases}, \qquad a_k = \frac{2}{n+1} \sum_{j=0}^{n} z_j \cos(jt_k),$$

$$\theta = \begin{cases} 0, & \text{falls } n \text{ gerade} \\ 1, & \text{falls } n \text{ ungerade} \end{cases}, \qquad b_k = \frac{2}{n+1} \sum_{j=0}^{n} z_j \sin(jt_k)$$

gegeben sind, heißt die trigonometrische Interpolierende zu den Interpolationspunkten (t_i, z_i) und erfüllt die Interpolationseigenschaft $f(t_i) = z_i$ für alle $0 \leq i \leq n$.
Wir nutzen die trigonometrische Interpolierende zu $(\frac{2\pi}{n+1} i, p_{i,x})$ und $(\frac{2\pi}{n+1} i, p_{i,y})$ für $0 \leq i \leq n$, um die Komponentenfunktionen $c_x(t)$ und $c_y(t)$ der oben beschriebenen Kurveninterpolation darzustellen.

2 Projektinhalt

Grundanforderungen:

a) Visualisierung der stückweise linearen Interpolation

b) Bestimmung und Visualisierung der trigonometrischen Interpolierenden zu einem vorgegebenen Satz an Punkten

c) Interaktives Abfragen von Punktdaten im Terminal inklusive des Verarbeitens von nicht zulässigen Eingaben

d) Einfügen von weiteren Punkten mit Neuberechnung und Darstellung der Veränderung

e) Testen Sie Ihre Methode zunächst für einfache Testfälle wie z.B. die Interpolation eines Kreises

$$p_{i,x}(t) = \cos(t), \qquad p_{i,y}(t) = \sin(t).$$

Mögliche Erweiterungen:

- Realisieren der Interpolation für sowohl offene als auch geschlossene Kurven.
- Interaktive Generierung von Punkten durch Anklicken im Ausgabefenster
- Verschieben von Punkten durch Drag-and-drop

- Alternative Interpolationstechniken
- Berechnung und Darstellung der konvexen Hülle (`https://de.wikipedia.org/wiki/Konvexe_H%C3%BClle`)

Tipp: Diese Interaktivität kann zum Beispiel mit Hilfe der matplotlib-Routine `mpl_connect` und onclick-Events realisiert werden. Unter Umständen muss dabei die Aktualisierung des Plots forciert werden. Für einen Plot `fig` kann diese durch `fig.canvas.draw()` ausgeführt werden.