Karlsruher Institut für Technologie

Angewandte Geometrie (CAGD)
Prof. Dr. Hartmut Prautzsch
Dipl.-Ing. Yijun Xu
B.Sc. Louis Lutzweiler

MARS-Basispraktikum **WS 2019/20**

Versuch 3

PARALLELKURVEN

Inhaltsverzeichnis

1 Allgemeines

Dieser Versuch liefert eine Anwendung der bisher kennen gelernten Kurvenerzeugungsverfahren. Zu einer gegebenen Spline-Kurve sollen Parallelkurven approximiert werden.
Im Versuch werden dazu einige Prozeduren implementiert. Zu Punkten auf der Spline-Kurve werden die exakten Punkte auf der Parallelkurve ermittelt und mit den Interpolationsalgorithmen der ersten Aufgaben interpoliert. Eine Fehlerfunktion bestimmt die Abweichung des Approximanden von der exakten Parallelkurve, der gegebenenfalls durch Hinzunahme weiterer Stützpunkte verbessert wird.

Durchführung

1. Lesen Sie sich diese Versuchsbeschreibung gründlich durch.
2. Programmieren Sie die Methode *insert_knot* der Klasse *spline*.

3. Erstellen Sie eine Eingabedatei mit einer einfachen kubischen B-Spline-Kurve. Fügen Sie einige Knoten in etwa gleichen Abständen ein. Lassen Sie sich beide Kontrollpolygone und die Kurve anzeigen. Fügen Sie weitere Knoten ein. Was fällt Ihnen auf, wenn Sie die Kontrollpolygone und die Kurve miteinander vergleichen?

4. Programmieren Sie die Methode *tangent* der Klasse *spline*. Orientieren sie sich dabei an der Implementierung der Methode *evaluate*.

5. Programmieren Sie die Methode *generate_parallel*.

6. Führen Sie die Ergebnisse dem Tutor vor.

2 Theoretische Grundlagen

Zur Modellierung von Gegenständen mit realistischer Materialdicke sowie bei der Mustererstellung im Designbereich werden so genannte Parallelkurven und -flächen eingesetzt. Sie treten auch beim NC-(numerisch gesteuerten)-Fräsen als Bahnkurven des Fräsers parallel zur ausgefrästen Fläche auf.
Unter einer Parallelkurve versteht man anschaulich eine Kurve mit konstantem Abstand zu einer Ausgangskurve. So erhält man beispielsweise als Parallelen eines Kreises wiederum Kreise mit gleichem Mittelpunkt. Parallelkurven wurden bereits in der Mitte des letzten Jahrhunderts von Bertrand untersucht und finden erneut Interesse durch oben erwähnte technische Anwendungen.
Leider sind Parallelkurven zu polynomialen Kurven im Allgemeinen keine polynomialen Kurven mehr, so dass für eine B-Spline-Darstellung auf Approximations- und Interpolationsverfahren zurückgegriffen werden muss.

2.1 Parallelkurven

Eine Parallelkurve ist eine Kurve, die die Normalen einer gegebenen Kurve in konstantem Abstand schneidet. Zu einer Stammkurve

$$\mathbf{s}(t) \;=\; \begin{pmatrix} x(t) \\ y(t) \end{pmatrix} \tag{1}$$

erhält man die Parallelkurve im Abstand d als

$$\mathbf{s}_d(t) \;=\; \mathbf{s}(t) \pm d\mathbf{n}(t), \quad (d \neq 0). \tag{2}$$

Im Fall ebener Kurven ergibt sich $\mathbf{n}$ durch Drehung des Einheitstangentenvektors um $\frac{\pi}{2}$:

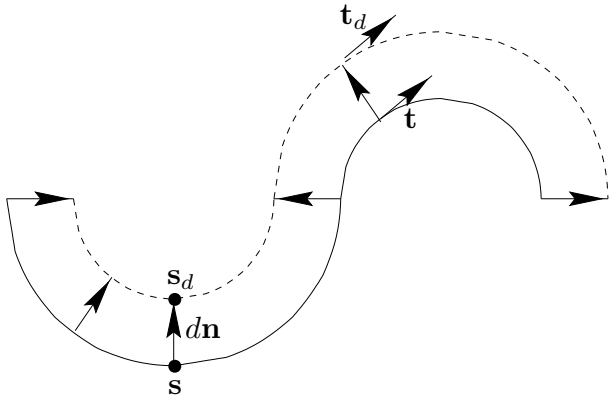
$$\mathbf{n}(t) \;=\; D\,\frac{\mathbf{s}'(t)}{\|\mathbf{s}'(t)\|} \;=\; \frac{1}{\sqrt{x'^2(t)+y'^2(t)}} \begin{pmatrix} -y'(t) \\ x'(t) \end{pmatrix}; \quad D := \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}.$$

Da hier im Nenner Wurzelterme auftreten, die sich im Allgemeinen nicht polynomial darstellen lassen, ist die Parallelkurve einer polynomialen Spline-Kurve nur in Ausnahmefällen wieder bezüglich der B-Spline-Basis darstellbar.

Aufgabe: *Die Ableitung der Parameterdarstellung der Kurve definiert einen Vektor in Tangentenrichtung. Normiert man seine Länge zu Eins, erhält man den Tangentenvektor. Berechnen Sie den Tangentenvektor der Parallelkurve* $\mathbf{t}_d(t)$*. Wo stimmt er mit* $\mathbf{t}(t)$ *überein, wo nicht?*

Im Allgemeinen überträgt sich die Gestalt der Stammkurve auf die Parallelkurve, weswegen Offsets auch bei der Vergrößerung und Verkleinerung von Bildern Einsatz finden. Charakteristische Punkte (Punkte mit vertikaler bzw. horizontaler Tangente, Wendepunkte und Scheitel) von Stamm- und Parallelkurve entsprechen einander in regulären Punkten (Nachweis erfolgt durch

einfaches Nachrechnen oder Nachschlagen in der entsprechenden Literatur).



2.2 Approximation von Parallelkurven

Die einfachste Möglichkeit, die Parallelkurve zu einer gegebenen Spline-Kurve

$$\mathbf{s}(t) \;=\; \sum_{i=0}^{m} \mathbf{d}_i N_i^3(t)$$

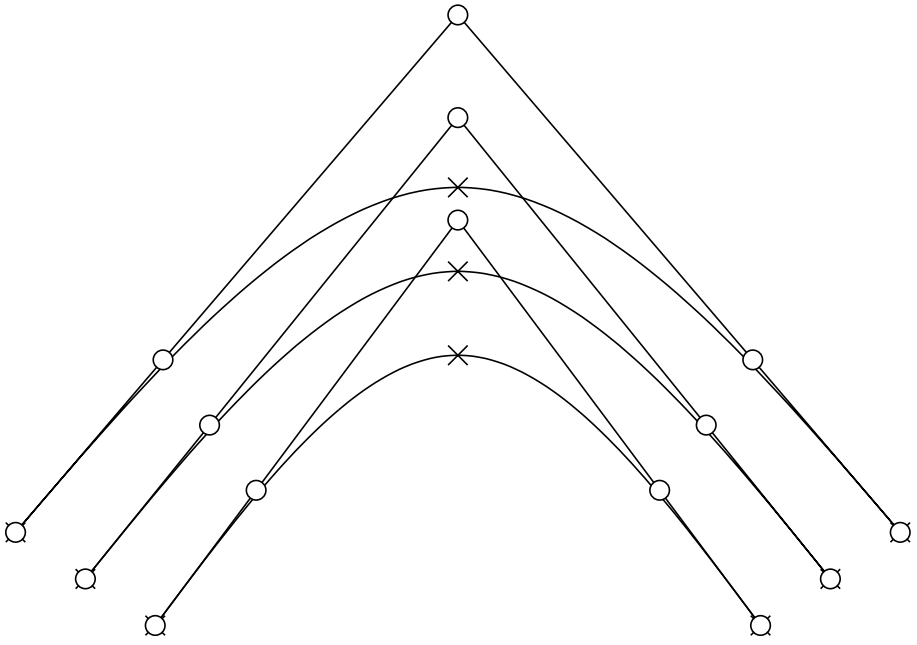
zu bestimmen, ist die Interpolation einiger exakt bestimmter Punkte der Parallelkurve. Hierzu kann der in Aufgabe 1 entwickelte Interpolationsalgorithmus verwendet werden.
Berechnen Sie die exakten Kurvenpunkte $\mathbf{p}_i = \mathbf{s}(t_i)$ in den Knotenpunkten t_i der B-Spline-Kurve. Die zugehörigen exakten Punkte der Parallelkurve ergeben sich dann als

$$\mathbf{p}_i^d \;=\; \mathbf{p}_i \;+\; d\mathbf{n}_i.$$

Mit diesen Punkten und zugehörigen Parameterwerten kann nun interpoliert werden.

Aufgabe: *Berechnen Sie die Einheitsnormalenvektoren* $\mathbf{n}_i$ *der Spline-Kurve* $\mathbf{s}(t)$ *(Formeln für die erste Ableitung siehe Literatur oder Vorlesung) in den Knotenpunkten und ermitteln damit die Punkte* $\mathbf{p}_i^d$*. Hierzu wird nicht die exakte Ableitung benötigt, es genügt bereits eine Tangentenrichtung* $\mathbf{t}_i$*. Diese wird durch die vorletzte Spalte des De-Boor-Schemas bestimmt, die zwei Punkte auf der Tangenten enthält:*

$$\mathbf{t}_i \parallel \mathbf{d}_{i-1}^2 \;-\; \mathbf{d}_{i-2}^2.$$



2.3 Approximationsfehler

Die Qualität der Approximation wird mit Hilfe einer Fehlerfunktion gemessen, die diskret in einigen Kurvenpunkten die Abweichung der exakten Parallelen von der approximierten bestimmt. Als Parameterwerte für die Auswertung bieten sich die Mittelpunkte der Knotenintervalle an (in den Knotenpunkten wurden die Punkte der Parallelen exakt ermittelt). Zu diesen Parameterwerten sind die Abstände der zugehörigen Punkte der approximierten Kurve von der exakten Parallelen zu bestimmen. Überschreitet ein solcher Abstand eine vorgegebene Fehlertoleranz, so ist das entsprechende Knotenintervall durch Einfügen des Mittelpunktes zu unterteilen und die Approximation der Parallelen mit der neuen B-Spline-Darstellung vorzunehmen. (Aus Effizienzgründen wird man gegebenenfalls mehrere Unterteilungsschritte gleichzeitig vornehmen und erst danach wieder approximieren).

Aufgabe: *Stellen Sie eine Gleichung für die Fehlerfunktion auf. Formulieren Sie damit einen Algorithmus, der zu einer gegebenen B-Spline-Kurve die Parallelkurve innerhalb einer gewissen Fehlertoleranz approximiert. Wo könnten dabei Probleme auftreten?*

Bei der Interpolation hat man wenig Kontrolle über den Parameterverlauf im Inneren der Knotenintervalle. Insbesondere kann der Abstand zwischen den zum Knotenintervallmittelpunkt gehörenden Punkten der approximierten und exakten Parallelen (der sich auf den ersten Blick als Bewertungskriterium anbietet) wesentlich größer oder kleiner sein als der tatsächliche maximale Kurvenabstand. Korrigieren Sie Ihre Fehlerfunktion entsprechend, indem sie beispielsweise den Abstand ihres Testpunktes auf der approximierten Parallelen von der Ausgangskurve bestimmen und mit dem Offsetabstand vergleichen. Spielen Sie ein wenig mit den verschiedenen Parametrisierungsmöglichkeiten herum.

2.4 Knoteneinfügen bei Spline-Kurven

Jede über einem Intervall polynomiale Kurve ist trivialerweise auch über einem Teilintervall polynomial darstellbar. Deshalb hat jeder Spline über den Knoten $t_i,\ i = 0, \ldots, m+n+1$ auch eine B-Spline-Darstellung über den Knoten $t_0, \ldots, t_l, \hat{t}, t_{l+1}, \ldots, t_{m+n+1}$.

Da über einem Knotenintervall $[t_{i+n}, t_{i+n+1})$ nur die $n+1$ B-Splines $N_i^n(t), \ldots, N_{i+n}^n(t)$ verschieden von Null sein können (s. a. Versuch 1), müssen auch nur diese ersetzt werden, wenn in dieses Segment ein Knoten $\hat{t}$ eingefügt wird, und zwar durch die über dem so modifizierten Knotenvektor definierten $\hat{N}_i^n(t), \ldots, \hat{N}_{i+n+1}^n(t)$, deren Träger $\hat{t}$ enthält.

Die neuen Knoten für $\hat{t} \in [t_{i+n}, t_{i+n+1})$ sind

$$\hat{t_j} = \begin{cases} t_j & j \leq i+n \\ \hat{t} \quad \text{für} & j = i+n+1 \\ t_{j-1} & j > i+n+1 \end{cases} .$$

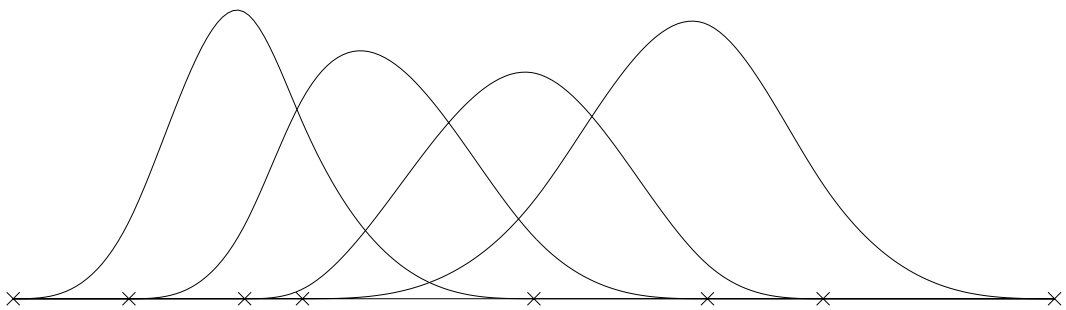


Abbildung 1: B-Spline-Basis ($N_i^3(t)$) zu einem gegebenen Knotenvektor

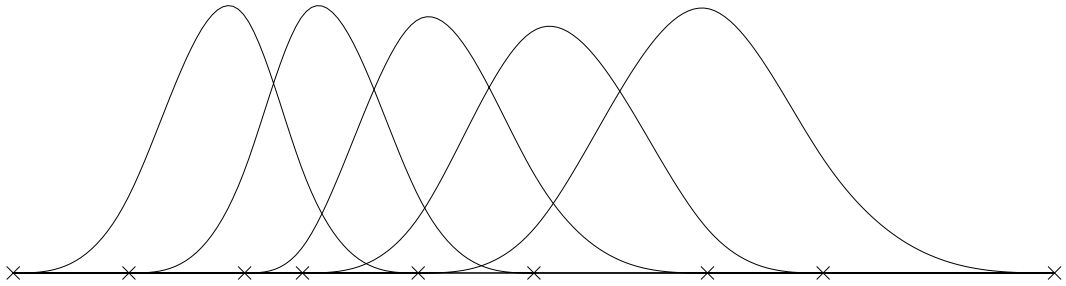


Abbildung 2: B-Spline-Basis ($\hat{N}_i^3(t)$) nach Einfügen eines Knotens

$$
\begin{array}{ccccc}
\mathbf{d}_i^0 & & & & \\
& \mathbf{d}_i^1 & & & \\
\mathbf{d}_{i+1}^0 & & \mathbf{d}_i^2 & & \\
& \mathbf{d}_{i+1}^1 & & \ddots & \\
\mathbf{d}_{i+2}^0 & & \vdots & & \mathbf{d}_i^n = \mathbf{s}(t) \\
& \vdots & & & \\
\vdots & & \mathbf{d}_{i+n-2}^2 & & \\
& \mathbf{d}_{i+n-1}^1 & & & \\
\mathbf{d}_{i+n}^0 & & & &
\end{array}
$$

$$
\begin{aligned}
\mathbf{d}_j^0 &= \mathbf{d}_j \\
\mathbf{d}_j^k &= (1-\alpha_j^k)\,\mathbf{d}_j^{k-1} + \alpha_j^k\,\mathbf{d}_{j+1}^{k-1} \\
\alpha_j^k &= \frac{t - t_{j+k}}{t_{j+n+1} - t_{j+k}}
\end{aligned}
$$

Abbildung 3: De-Boor-Schema für Grad n

Abbildung 1 zeigt für den Grad $n = 3$ die vier B-Splines, die nach Einfügen eines Knotens in das mittlere Segment durch die fünf B-Splines gleichen Grades in Abb. 2 ersetzt werden müssen.

Im Kontrollpolygon müssen dann die mit den ersetzten B-Splines gewichteten Kontrollpunkte durch zu den neuen B-Splines gehörende Kontrollpunkte ersetzt werden, also $\mathbf{d}_i, \ldots, \mathbf{d}_{i+n}$ durch $\hat{\mathbf{d}}_i, \ldots, \hat{\mathbf{d}}_{i+n+1}$, wobei die ersten bzw. letzten jeweils sogar noch übereinstimmen. Man erhält das neue Kontrollpolygon schließlich aus

$$\hat{\mathbf{d}}_j = \begin{cases} \mathbf{d}_j & & j \leq i \\ (1 - \alpha_{j-1}^1)\mathbf{d}_{j-1} + \alpha_{j-1}^1 \mathbf{d}_j & \text{für} & i < j < i+n+1 \\ \mathbf{d}_{j-1} & & j \geq i+n+1 \end{cases} .$$

Für $i < j < i + n + 1$ gilt also $\hat{\mathbf{d}}_j = \mathbf{d}_{j-1}^1$, wobei die $\mathbf{d}_{j-1}^1$ die Elemente aus der 1. Spalte des De-Boor-Schemas (s. Abb. 3) sind. Man spricht daher auch von einem „Umweg über die 1. Spalte“.

Falls ein Knoten r-fach eingefügt werden soll, kann man dieses Verfahren entsprechend oft auf das jeweils neue Kontrollpolygon anwenden oder einfacher einen „Umweg über die r-te Spalte“ gehen, d. h. von allen Spalten bis zur $(r-1)$-ten werden nur das erste und letzte Element, von der r-ten alle Elemente übernommen (s. [PRA], [FAR] oder [BO1]).

Bemerkung: Der Knoteneinfüge-Algorithmus ist hier nur für Knoten $\hat{t}$ aus dem Trägerintervall $[t_{i+n}, t_{i+n+1})$ formuliert. Damit kann er problemlos für die in Aufgabe 1 gewonnenen Spline-Kurven verwendet werden. Für die periodischen Splines aus Aufgabe 2 ist eine Modifikation des Algorithmus nötig (vergleiche Flächenaufgaben bzw. [BO2]).

Zur Approximation von Parallelkurven sollen sukzessive nur die Mittelpunkte der Knotenintervalle in den Knotenvektor eingefügt werden, bis die approximierte Parallele genügend hohe Genauigkeit erreicht hat.

Aufgabe: *Stellen Sie den Knoteneinfüge-Algorithmus speziell für den Fall uniformer Knoten und das Einfügen sämtlicher Knotenmittelpunkte auf. Was fällt auf?*

3 Praktischer Teil

3.1 Programmierung

In `spline.py` fehlt die Methode *insert_knot*. Berechnen Sie mithilfe des de-Boor-Schemas die neuen Kontrollpunkte und fügen Sie den neuen Knoten in den Knotenvektor ein. Um den Knoten sortiert einzufügen, können sie die Methode *insert* der Klasse *knots* verwenden.

Außerdem soll die Methode *tangent* der Klasse *spline* implementiert werden. Orientieren Sie sich dazu an der Implementierung der Methode *evaluate*.

Bei der Programmierung der Methode *generate_parallel* sollen nun einige der von Ihnen implementierten Methoden angewandt werden.

Dazu soll zunächst der gegebene Spline an den Knoten abgetastet werden und daraus Interpolationspunkte für den parallelen Spline berechnet werden.

Als nächstes soll mit chordaler Interpolation daraus ein Spline berechnet werden. Um die Güte der Approximation bestimmen zu können, ist es wichtig, dass beide Splines den gleichen Knotenvektor haben. Setzen Sie daher die Knoten des parallelen Splines auf die Knoten des Eingabesplines. Jetzt soll in der Mitte zwischen zwei Knoten die Distanz der Splines berechnet werden. Bei einer Abweichung von mehr als *eps* von der geforderten Distanz soll an dieser Stelle ein neuer Knoten in den originalen Spline eingefügt werden um eine bessere Approximation zu gewinnen.

Wenn Knoten eingefügt wurden, soll die Prozedur mit dem nun feiner parametrisierten Spline wiederholt werden.

Literatur

[BO1] *W. Boehm, G. Farin, J. Kahmann*, A survey of curve and surface methods in CAGD, Computer Aided Geometric Design 1, July 1984.

[BO2] *W. Boehm, H. Prautzsch*, The insertion algorithm, CAD 17 (2), 1985, S. 58–59.

[FAR] *G. Farin*, Curves and Surfaces for Computer Aided Design, Second Edition 1990, Academic Press.

[HOS] *J. Hoschek, D. Lasser*, Grundlagen der geometrischen Datenverarbeitung, 2. Auflage 1992, Teubner Verlag, Stuttgart.

[PRA] *H. Prautzsch, W. Boehm, M. Paluszny*, Bézier- and B-Spline Techniques, 2002, Springer-Verlag, Berlin.