Numerisches Programmieren

Übung #04

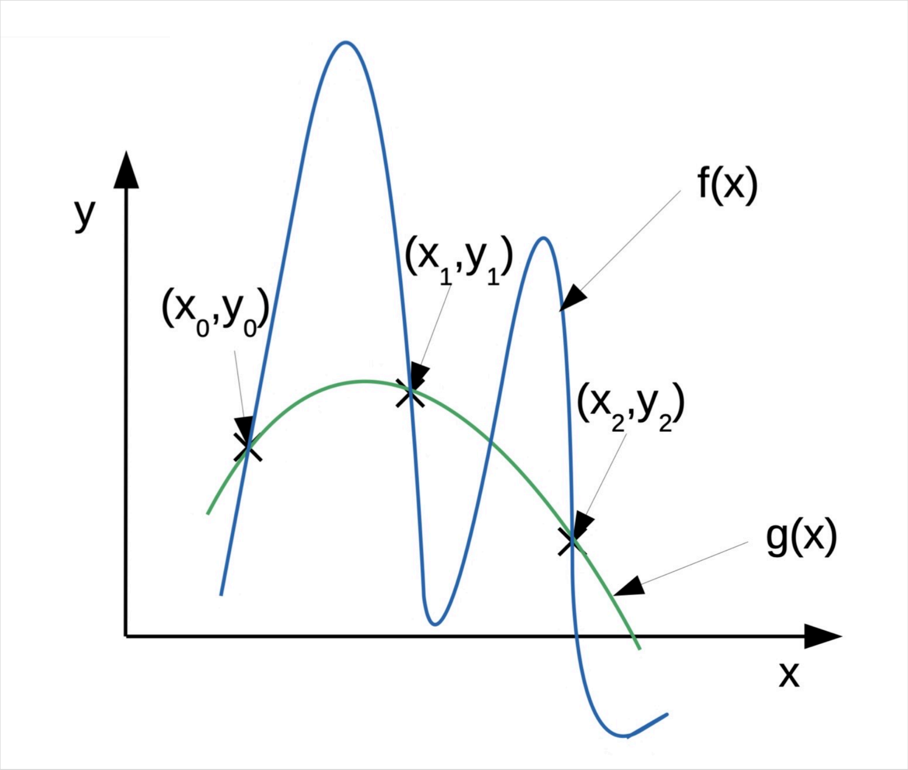
geheimtippmuenchen.de/content/uploads/2023/02/geheimtipp-muenchen-thiersch-turm-1.jpg

Interpolation #2

Recap: **Interpolation**

- Abschätzung einer unbekannten Funktion $f(x)$ für gegebene Stützpunkte
- Probleme:
  - Keine Vorgaben für die Ableitung
  - Runge-Effekt

→ Stückweise Interpolation



Hermite Interpolation

- Ansatz: Kubisches Polynom für jedes Intervall (je 2 Stützpunkte)
  - Teilpolynome an Stützpunkten „aneinandergeklebt“
  - Übereinstimmung mit Stützwerten & Ableitungen von $f(x)$ (Stetige Differenzierbarkeit $p \in \mathcal{C}^1$)

- Für jedes Intervall: Stützwerte y_0, y_1; Ableitungen y'_0, y'_1

  → Kubisches Polynom eindeutig festgelegt

- Mithilfe von **Hermite-Basispolynomen**

Aufgabe 1)

1) Stückweise Hermite-Interpolation

Ziel der Interpolation nach Hermite ist es, eine Funktion $p(x)$ zu erhalten, die überall stetig differenzierbar ist ($p \in \mathcal{C}^1 =$ keine Knicke). Um dies zu erreichen, müssen zusätzlich zu Stützpunkten $(x_0, y_0), \ldots, (x_n, y_n)$ auch noch die Werte der ersten Ableitung $y_0', \ldots, y_n'$ an den Stützstellen vorgegeben werden. Damit ist es möglich, für $p(x)$ auf jedem Teilintervall $[x_i; x_{i+1}]$, $i = 0, \ldots, n-1$, ein kubisches Polynom zu erzeugen und diese Einzeldarstellungen stetig differenzierbar an den x_i zu »verkleben«.

a) Betrachten Sie den einfachen Fall nur eines Teilintervalls mit $x_0 = 0$ und $x_1 = 1$. Zusätzlich zu den Funktionswerten y_0, y_1 seien auch die ersten Ableitungen y_0', y_1' bei x_0 und x_1 gegeben.
Bestimmen Sie das kubische Polynom $p(x)$, das durch die Vorgabe dieser Werte

$$\begin{aligned} p(x_0) = y_0, \quad p(x_1) &= y_1 \\ p'(x_0) = y_0', \quad p'(x_1) &= y_1' \end{aligned}$$

festgelegt ist! Nützen Sie hierfür den allgemeinen Ansatz für kubische Polynome

$$p(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3, \quad t \in [0; 1] = [x_0; x_1].$$

Bestimmen Sie anschließend mittels Koeffizientenvergleich die kubischen Basispolynome $H_0, \ldots, H_3$ des Hermite-Ansatzes

$$p(t) = y_0 \cdot H_0(t) + y_1 \cdot H_1(t) + y_0' \cdot H_2(t) + y_1' \cdot H_3(t), \quad t \in [0; 1] = [x_0; x_1].$$

Aufgabe 1)

b) Um die stückweise Hermite-Interpolation durchführen zu können, muss der Spezialfall $x_0 = 0$ und $x_1 = 1$ von Teilaufgabe a) auf ein beliebiges Teilintervall $[x_i, x_{i+1}]$ übertragen werden. Geben Sie dazu zuerst die passenden vier Bedingungen für das lokale kubische Polynom an!
Überlegen Sie dann, wie mit Hilfe einer Transformationsfunktion $t_i(x)$ sowie der bereits in a) berechneten Basispolynome $H_0(t)$ bis $H_3(t)$ eine passende Interpolationsfunktion $p_i(t_i(x))$ auf dem Teilintervall $[x_i, x_{i+1}]$ konstruiert werden kann! Insgesamt gilt dann:

$$p(x)|_{[x_i, x_{i+1}]} = p_i(t_i(x)).$$

Hermite Interpolation

Hermite Basispolynome:

$$H_0(t) = 1 - 3t^2 + 2t^3$$
$$H_1(t) = 3t^2 - 2t^3$$
$$H_2(t) = t - 2t^2 + t^3$$
$$H_3(t) = -t^2 + t^3$$

Transformation $\boldsymbol{t_i(x)}$: $[x_i, x_{i+1}] \to [0, 1]$

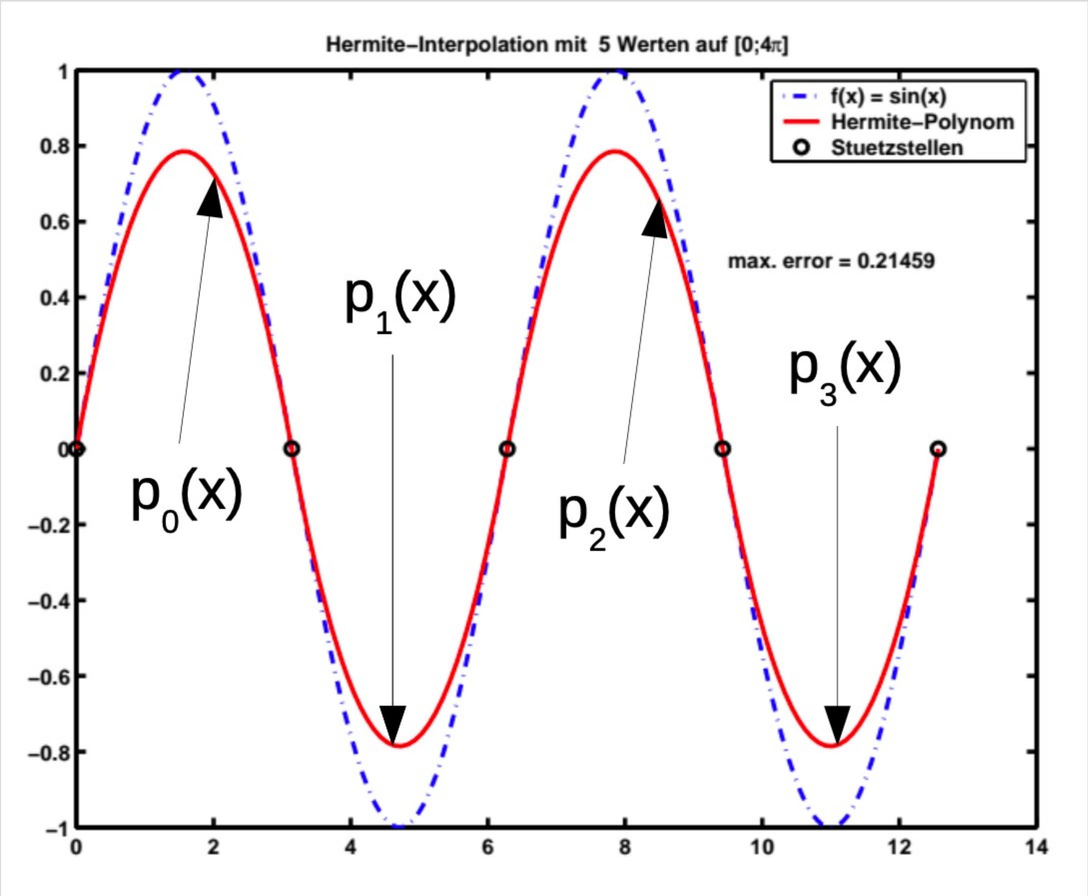
$$t_i(x) = \frac{x - x_i}{h_i}$$

Kubisches Teilpolynom:

$$p_i(t_i(x)) = y_i \cdot H_0(t_i(x)) + y_{i+1} \cdot H_1(t_i(x))$$
$$+ h_i \cdot y'_i \cdot H_2(t_i(x)) + h_i \cdot y'_{i+1} \cdot H_3(t_i(x))$$

- Für beliebige Intervalle anwendbar
  - ggf. Transformation $t_i(x)$ notwendig
- Laufzeit von $\mathcal{O}(n^3)$ relativ kostenaufwendig

Hermite Interpolation



Kubische Splines

- Kubisches Polynom („**Spline**") für jedes Intervall
- Keine Ableitungen gegeben
  - (Ableitung an den Rändern y_0', y_n' aber schon)

→ $\mathcal{C}^2$ Stetigkeit an Klebestellen (Übereinstimmung der Teilpolynome bis zur 2. Ableitung)

Kubische Splines

Algorithmus:

- gegeben: äquidistante (!) Stützstellen, Ableitungen *an den Rändern*

1. Fehlende Ableitungen durch LGS ermitteln:

$$\begin{bmatrix} 4 & 1 & & \\ 1 & 4 & \ddots & \\ & \ddots & \ddots & 1 \\ & & 1 & 4 \end{bmatrix} \cdot \boxed{\begin{pmatrix} y_1' \\ y_2' \\ \vdots \\ y_{n-2}' \\ y_{n-1}' \end{pmatrix}} = \frac{3}{h} \cdot \begin{pmatrix} y_2 - y_0 \\ y_3 - y_1 \\ \vdots \\ y_{n-1} - y_{n-3} \\ y_n - y_{n-2} \end{pmatrix} - \begin{pmatrix} y_0' \\ 0 \\ \vdots \\ 0 \\ y_n' \end{pmatrix}$$

Kubische Splines

2. Splines $p_i(t)$ ermitteln (ähnlich wie bei Hermite)
   - Transformationen $t_i(x)$ der jeweiligen Splines ermitteln
   - Stützwerte & Ableitungen in $p_i(t)$ einsetzen

→ Beispiel Ergebnis:

$$s(x) = \begin{cases} p_0(t_0(x)) & \text{für } x \in [-1, 0[\text{ mit } t_0(x) = x + 1 \\ p_1(t_1(x)) & \text{für } x \in [0, 1[\text{ mit } t_1(x) = x \\ p_2(t_2(x)) & \text{für } x \in [1, 2] \text{ mit } t_2(x) = x - 1 \end{cases}$$

$$\begin{aligned} p_0(t) &= 2 \cdot H_0(t) && && + \; 9 \cdot H_2(t) && - \; 3 \cdot H_3(t) \\ p_1(t) &= && 2 \cdot H_1(t) && - \; 3 \cdot H_2(t) && + \; 3 \cdot H_3(t) \\ p_2(t) &= 2 \cdot H_0(t) && + \; 3 \cdot H_1(t) && + \; 3 \cdot H_2(t). && \end{aligned}$$

Aufgabe 2)

2) Interpolation mit kubischen Splines

Bei der kubischen Spline-Interpolation möchte man eine interpolierende Funktion $s(x)$ erhalten, die ähnlich wie bei der Hermite-Interpolation aus kubischen Teilpolynomen $p_i(x)$ auf den Teilintervallen $[x_i, x_{i+1}]$ besteht. Allerdings soll die Splinefunktion $s(x)$ überall zweimal stetig differenzierbar sein ($s \in \mathcal{C}^2$) und dafür keine anderen Informationen benötigen als die $n+1$ zu interpolierenden Stützpunkte $(x_0, y_0), \ldots, (x_n, y_n)$.

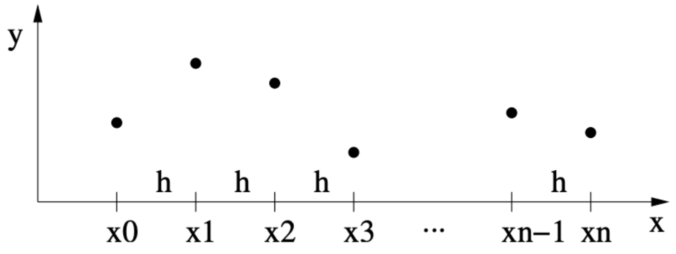


Abbildung 1: Visualisierung der äquidistanten Stützstellen x_i.

In dieser Aufgabe beschränken wir uns auf den Fall äquidistanter Stützstellen x_i, d.h. alle Teilintervalle haben dieselbe Länge (vgl. Abb. 1):

$$x_{i+1} - x_i \quad = \quad h \ := \ \frac{x_n - x_0}{n}, \quad i = 0, 1, \ldots, n-1.$$

Aufgabe 2)

a) Geben Sie die Bedingungen an, die aus der Interpolation und der $\mathcal{C}^2$-Stetigkeit resultieren!

b) Berechnen Sie die zweiten Ableitungen $H_i''(t), i = 0, \dots, 3$, der Hermite-Basispolynome $H_i(t)$ aus Aufgabe 1) und werten Sie sie an den Stellen $t = 0$ und $t = 1$ aus!

c) Zeigen Sie, dass $s(x)$ mit Hilfe der Stützwerte y_i und folgendem linearen Gleichungssystem konstruiert werden kann:

$$\begin{pmatrix} 4 & 1 & & \\ 1 & 4 & \ddots & \\ & \ddots & \ddots & 1 \\ & & 1 & 4 \end{pmatrix} \begin{pmatrix} y_1' \\ y_2' \\ \vdots \\ y_{n-2}' \\ y_{n-1}' \end{pmatrix} = \frac{3}{h} \begin{pmatrix} y_2 - y_0 - \frac{h}{3} y_0' \\ y_3 - y_1 \\ \vdots \\ y_{n-1} - y_{n-3} \\ y_n - y_{n-2} - \frac{h}{3} y_n' \end{pmatrix}.$$

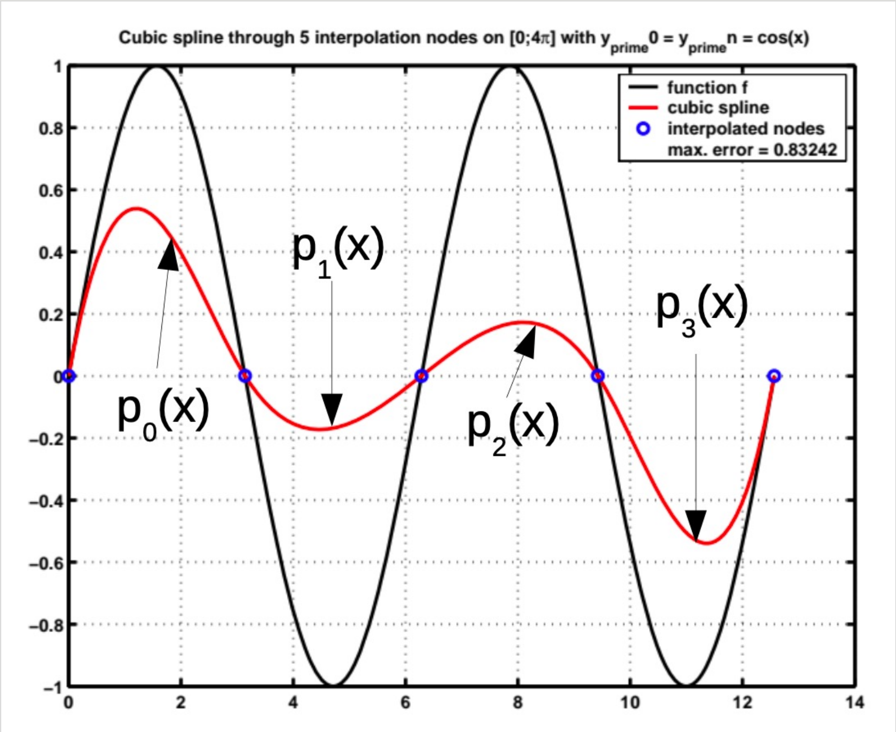
d) Bestimmen Sie die Spline-Funktion $s(x)$ für die Stützpunkte

$$P_0 = (-1, 2), \quad P_1 = (0, 0), \quad P_2 = (1, 2), \quad P_3 = (2, 3)$$

und die Randbedingungen

$$s'(-1) = 9, \quad s'(2) = 0 \quad .$$

Kubische Splines



- In $\mathcal{O}(n)$, wegen der Tri-diagonalmatrix
- Verlaufen durch die Stützstellen mit wenig Krümmung → wenig Oszillation

Aufgabe 3)

3) Wiederholung: Interpolation

Betrachten Sie im Folgenden die Funktion

$$f : [0,2] \to [0,1], \quad f(x) = \sin\left(\frac{\pi x}{2}\right).$$

a) Bestimmen Sie einen Polynom-Interpolanten $p(x)$, der die Funktion f interpoliert und die drei Stützpunkte $P_0 = (x_0, y_0)$, $P_1 = (x_1, y_1)$, and $P_2 = (x_2, y_2)$ mit den zugehörigen Stützstellen

$$x_0 = 0, \quad x_1 = 1, \quad x_2 = 2$$

besitzt.

Berechnen Sie dazu die dividierten Differenzen und geben Sie eine geschlossene Darstellung des Interpolanten $p(x)$ an, indem Sie die Newton'sche Interpolationsformel verwenden.

b) Um eine bessere Approximation zu erzielen, werden die Stützpunkte um einen neuen Punkt $P_3 = (x_3, y_3)$ mit $x_3 = \frac{1}{3}$ erweitert. Bestimmen Sie den neuen Interpolanten, der die vier Punkte P_0, P_1, P_2 und P_3 interpoliert.

Aufgabe 3)

c) Sei $p_1(u)$ ein Interpolant zu beliebig gegebenen Stützstellen $\{u_0, u_1, u_2, u_3\}$ und $p_2(u)$ ein Interpolant zu Stützstellen $\{u_1, u_2, u_3, u_4\}$. Bestimmen Sie einen Interpolanten $p_3(u)$ zu den Stützstellen $\{u_0, u_1, u_2, u_3, u_4\}$, basierend auf $p_1(u)$ und $p_2(u)$!

d) Welche Methode eignet sich zur Interpolation von ca. 100 äquidistanten Stützpunkten? Geben Sie zwei Vorteile gegenüber der Polynom-Interpolation an!

e) Die Methode **aitken** des folgenden Java-Codefragments soll die Polynominterpolation mit dem Schema nach Aitken-Neville berechnen. Dabei bezeichnen **xs** den Vektor der x-Koordinaten der Stützstellen, **f** den Vektor der Stützwerte und **x** die gewünschte Auswertungsstelle des Interpolationspolynoms. Zeigen Sie 2 Fehler auf und korrigieren Sie diese.

```java
public static double aitken(double xs[], double f[], double x) {
  int n = f.length;
  assert n==xs.length;

  for(int k = 1; k <= n; k++)
    for(int i = 0; i < n-k; i++)
      f[i] = ((x-xs[i])*f[i+1] - (x-xs[i+k])*f[i])/(xs[i]-xs[i+k]);

  return f[n-1];
}
```

Aufgabe 3)

f) Erklären Sie kurz, was die Methode `foo()` des folgenden Java-Codefragments unter Verwendung der Methode **`aitken`** aus Teilaufgabe e) berechnet.

```java
public static double foo(double xs[], double ys[], double f[][], double x,
    double y) {
   assert xs.length == ys.length;
   assert xs.length == f.length;
   assert ys.length == f[0].length;

   double pHat[] = new double[xs.length];

   for(int i=0; i < xs.length; i++) pHat[i]=aitken(ys,f[i],y);
   return aitken(xs,pHat,x);
}
```

Danke fürs Kommen!
Bis nächste Woche! 😊

geheimtippmuenchen.de/content/uploads/2023/02/geheimtipp-muenchen-thiersch-turm-1.jpg