b) Suche Lösung zu: $3x_1 + 4{,}127\,x_2 = 15{,}41$ (I)

$x_1 + 1{,}374\,x_2 = 5{,}147$ (II)

mit 4-stelliger Genauigkeit

Nehme : Gauss Elimination

$$\mathrm{II} \to \mathrm{II} - \frac{1}{3{,}000}\,\mathrm{I} \Rightarrow \left(1{,}374 - \frac{4{,}127}{3{,}000}\right)x_2 = 5{,}147 - \frac{15{,}41}{3{,}000}$$

$$\Leftrightarrow (1{,}374 - 1{,}376)\,x_2 = (5{,}147 - 5{,}140)$$

$$\Rightarrow \quad -0{,}002000\,x_2 = 0{,}007000$$

$$\Rightarrow \quad \tilde{x}_2 = -3{,}500$$

wobei $x_2 = -6{,}2$ das korrekte Ergebnis wäre!

Problem in beiden Fällen:

Subtraktion nahezu gleichgroßer Zahlen, was in jedem Fall vermieden werden sollte! (Auslöschung)
Die Fehlerursache ist in beiden Fällen sehr unterschiedlich:

a) der Algorithmus ist schlecht konditioniert
b) das Problem ist schlecht konditioniert, da es zu dem unlösbaren Problem benachbart ist.

$$3x_1 + 4{,}122\,x_2 = 15{,}41$$
$$x_1 + 1{,}374\,x_2 = 5{,}147$$

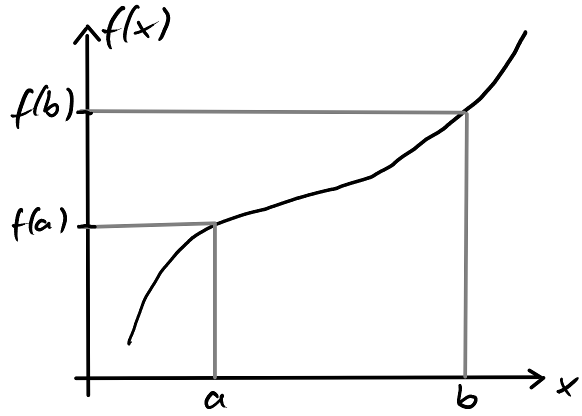
Für a) hilft ein besserer Algorithmus, wohingegen für b) nur höhere Genauigkeit hilft („brute force“)

2. Approximation von Funktionen

Gegeben sei ein diskreter Datensatz:

$f_i = f(x_i) \qquad i = 0, \dots, n \qquad x_i \in [a, b]$

$x_0 = a \qquad x_n = b$

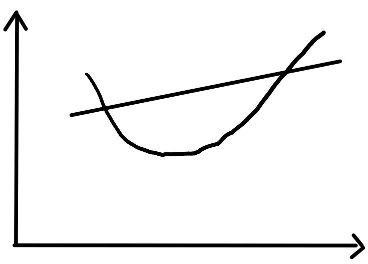


Die Menge $\{x_i\}$ bezeichnet man als Stützstellen und die Menge $\{f_i\}$ als Stützwerte.

2.1 Lineare Interpolation

Im einfachsten Fall approximiert man f linear für $x \in [x_0, x_1]$. D.h.:

$$f(x) = f_0 + \frac{x - x_0}{x_1 - x_0}(f_1 - f_0) + \underbrace{\Delta_f(x)}_{\text{Fehler der Interpolation}}$$



Dies ist i.A. keine besonders gute Approximation der Funktion.

2.2 Das Interpolationspolynom

Weierstraßscher Approximationssatz:

Ist die Funktion $f(x)$ auf dem Intervall $[a,b]$ definiert und stetig, dann existiert zu einer vorgegebenen Zahl $\varepsilon > 0$ ein Polynom $P(x)$ auf diesem Intervall, das der folgenden Beziehung genügt.

$$|f(x) - P(x)| < \varepsilon \quad \text{für alle } x \in [a,b]$$

Gesucht wird ein Polynom $P_n(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^2$ vom Grad höchstens n derart, dass es durch die $n+1$ vorgegebenen Punkte $(x_0, f_0), (x_1, f_1), \ldots, (x_n, f_n)$ verläuft.

$$P_n(x_k) = f_k \quad \text{für } k = 0, 1, \ldots, n$$

Konstruktion mit Lagrangepolynomen:

Nochmal lineare Interpolation: $P(x) = \underbrace{\frac{x - x_1}{x_0 - x_1}}_{:= L_0} f_0 - \underbrace{\frac{x - x_0}{x_1 - x_0}}_{:= L_1} f_1$

Quotienten L_0 und L_1 erfüllen:

$x = x_0 : L_0(x_0) = 1 \quad , \quad L_1(x_0) = 0$

$x = x_1 : L_0(x_1) = 0 \quad , \quad L_1(x_1) = 1$

Allgemein muss für jedes $k = 0, 1, \dots, n$ ein Quotient $L_{n,k}(x)$ bestimmt werden, der folgende Eigenschaft besitzt: $L_{n,k}(x_i) = \delta_{ki}$

Betrachte $\omega_{n+1}(x) = (x - x_0)(x - x_1) \dots (x - x_n)$

$$= x^{n+1} + \dots$$

$$\Rightarrow L_{n,k}(x) = \frac{\omega_{n+1}(x)}{\omega_{n+1}(x_k)} = \prod_{\substack{i=0 \\ i \neq k}}^{n} \frac{x - x_i}{x_k - x_i}$$

Das Polynom $P_n(x)$ kann nun aus den Lagrangefaktoren $L_{n,k}(x)$ konstruiert werden

$$P_n(x) = \sum_{k=0}^{n} L_{n,k}(x) f_k$$

Man kann zeigen, dass dieses Polynom eindeutig ist.

2.3 Newtonsche Form des Interpolationspolynoms

Def. dividierte Differenzen

$$f[y] = f(y)$$

$$f[x,y] = \frac{f[x]-f[y]}{x-y} := \frac{f(x)-f(y)}{x-y} \quad \hat{=} \text{ Steigung der Sekante}$$

$$\vdots$$

$$f[x,y,\dots,v,w] = \frac{f[x,y,\dots,v]-f[x,y,\dots,v,w]}{x-y}$$

Bemerkung: Nach dem Mittelwertsatz der Differentialrechnung existiert ein $\tau \in [x,y]$ mit
$f[x,y] = f'(\tau)$

genauso: $f[x_1,\dots,x_k] = \frac{1}{(k-1)!} f^{(k-1)}(\tau)$, $\tau \in \langle x_1,\dots,x_k \rangle$

Herleitung der Newtonschen Formel

Wir verfahren schrittweise und nehmen mehr und mehr Stützstellen hinzu, so dass $P_n(x_i) = f_i$ für alle Stützstellen x_i

0. - Näherung

Mit $P_0(x)$ approximieren wir $f(x)$ durch eine Konstante, nämlich den Wert $f(x_0)$

$$f(x) = P_0(x) + \Delta^{(0)}(x) \qquad \Delta^{(0)}(x): \text{Approximationsfehler}$$

$$= f[x_0] + \Delta^{(0)}(x)$$

$$\Delta^{(0)}(x) = f(x) - f[x_0] = (x-x_0)\frac{f(x)-f(x_0)}{x-x_0} = (x-x_0) f[x,x_0]$$

1. - Näherung

Wir gewinnen die 1. Näherung aus der 0. Näherung, indem wir im Restgliedterm die x-Abhängigkeit in der dividierten Differenz durch den festen Wert x_1 ersetzen:

$$f(x) = P_1(x) + \Delta^{(1)}(x) \qquad P_1(x) = f[x_0] + (x-x_0)f[x_0,x_1]$$

und $\Delta^{(1)}(x) = \Delta^{(0)}(x) - (x-x_0)f[x_0,x_1]$

Bemerkungen: $P_1(x_0) = f[x_0] = f(x_0)$

zu $P_1(x)$ $\quad P_1(x_1) = f[x_1] = f(x_1)$

Bemerkung zu dividierter Differenz

$$f[x,y] = f[y,x]$$

allgemein: $f[\ldots]$ ist total symmetrisch unter Vertauschung seiner Argumente.

$$\Delta^{(1)}(x) = (x-x_0)f[x,x_0] - (x-x_0)f[x_0,x_1]$$
$$= (x-x_0)(x-x_1)f[x,x_0,x_1]$$

Dies lässt sich fortsetzen (n-te Näherung):

$$f(x) = P_n(x) + \Delta^{(n)}(x)$$

mit $P_n(x) = f[x_0] + (x-x_0)f[x_0,x_1] + (x-x_0)(x-x_1)f[x_0,x_1,x_2] + \ldots +$

$$\underbrace{(x-x_0)(x-x_1)\ldots(x-x_{n-1})}_{w_n(x)}\underbrace{f[x_0,x_1,\ldots,x_n]}_{\frac{1}{n!}f^{(n)}(\tau)}$$

$$\Delta^{(n)}(x) = \underbrace{(x-x_0)(x-x_1)\ldots(x-x_n)}_{w_{n+1}(x)}\underbrace{f[x_0,x_1,\ldots,x_n,x]}_{\frac{1}{(n+1)!}f^{(n+1)}(\tau)}$$

Beispiel: 1. Näherung: $\Delta^{(1)}(x) = \frac{f''(\tau)}{2}(x-x_0)(x-x_1)$

Abschätzung des Fehlers: mit $d = \max_{x \in [x_0, x_1]} |f''(x)|$

und $\frac{d\,\Delta^{(1)}(x)}{dx} \overset{!}{=} 0$

$\Rightarrow |\Delta^{(1)}(x)| \leq \frac{d}{8}(x_1 - x_0)^2$

Konkret: $\sin(x) = f(x)$ bei $x = \frac{3\pi}{8}$ mit Stützstellen

$$x_0 = \frac{\pi}{4} \rightarrow f_0 = 0{,}707$$

$$x_1 = \frac{\pi}{2} \rightarrow f_0 = 1{,}0$$

$\Rightarrow f(x) \approx 0{,}854$ wobei $f(x) = \sin(x) = 0{,}924$

Damit folgt: $|\Delta^{(1)}(x)| = 0{,}070 < \frac{d}{8}(x_1 - x_0)^2 = 0{,}077$

Praktische Verwendung

1) Berechne die Newtonkoeffizienten aus den Stützstellen
2) Werte das Polynom mit einem geeigneten Algorithmus aus.

zu 1) Wir müssen die Hauptdiagonale der Matrix

$$A = \begin{pmatrix} f[x_0] & & & \\ f[x_1] & f[x_0, x_1] & & \\ \vdots & \vdots & & \\ f[x_n] & \dots & \dots & f[x_0, x_1, \dots, x_n] \end{pmatrix}$$

bestimmen,

Algorithmus: Newton-Koeffizienten

Input: $(x_0, f_0)(x_1, f_1) \dots (x_n, f_n)$

$A_{00} = f_0$
for $i = 1, \dots, n$
  $A_{i0} = f_i$
  for $j = 1, \dots, i$

$$A_{ij} = \frac{A_{i,j-1} - A_{i-1,j-1}}{x_i - x_{i-j}}$$

output $A_{00}, \dots, A_{nn} \rightarrow A_0, \dots, A_n$

zu 2) Horner Schema

$$11 + 7x - 5x^2 - 4x^3 + 2x^4$$
$$= 11 + x(7 + x(-5 + x(-4 + x \cdot 2)))$$

<u>Algorithmus:</u> Newton-Horner Schema

Input: A_i, x_i, x

$B = 0$

for $i = n, .., 0$

  $B = (x - x_i) B + A_i$

output $B = P_n(x)$

<u>Bemerkung:</u> Dieses Schema lässt sich leicht um die gleichzeitige Auswertung von $P'(x)$ erweitern.