Beilage zur Serie 4

Problemstellung

Zu gegebenen $n+1$ Stützpunkten (x_i, y_i), $i = 0, 1, \ldots, n$, suchen wir das eindeutig bestimmte Polynom p von maximalem Grad n durch diese Stützpunkte. Das heisst: $p(x_i) = y_i$ für $i = 0, 1, \ldots, n$ (vergleiche “Einführung in die Numerik”).
Das Polynom p kann man auf verschiedene Weise darstellen. Diese Schreibweisen sehen im Allgemeinen folgendermassen aus:

$$p(z) = \sum_{j=0}^{n} w_j \, p_j(z).$$

Dabei sind $w_j \in \mathbb{R}$ die sogenannten Gewichte, p_j bestimmte Polynome und $z \in \mathbb{R}$ die Stelle, an welcher wir das Interpolationspolynom auswerten wollen.
Damit alle Polynome von Grad kleiner oder gleich n auf diese Weise dargestellt werden können, müssen die $\{p_j\}_{j=1}^{n}$ eine Basis vom Raum der Polynome von maximalem Grad n sein (vergleiche “Lineare Algebra”).
Im Folgenden betrachten wir zwei mögliche Darstellungen.

Monomiale Interpolation und Horner-Schema

Bei der monomialen Interpolation sind die $p_j(z)$ Monome, also $p_j(z) = z^j$, und es folgt

$$p(z) = \sum_{j=0}^{n} a_j \, p_j(z) = \sum_{j=0}^{n} a_j \, z^j.$$

Um das Interpolationspolynom zu bestimmen müssen wir die Koeffizienten a_j, $j = 0, 1, \ldots, n$, berechnen. Da p durch die Punkte (x_i, y_i) für $i = 0, 1, \ldots, n$ gehen muss, erhalten wir $n+1$ Gleichungen, nämlich:

$$\begin{aligned}
a_0 \, x_0^0 + a_1 \, x_0^1 + \ldots + a_n x_0^n &= y_0, \\
a_0 \, x_1^0 + a_1 \, x_1^0 + \ldots + a_n x_1^n &= y_1, \\
&\vdots \\
a_0 \, x_n^0 + a_1 \, x_n^0 + \ldots + a_n x_n^n &= y_n.
\end{aligned}$$

In Matrixschreibenweise entspricht dies dem Gleichungssystem

$$\underbrace{\begin{bmatrix} x_0^0 & x_0^1 & \ldots & x_0^n \\ x_1^0 & x_1^1 & \ldots & x_1^n \\ \vdots & & & \vdots \\ x_n^0 & x_n^1 & \ldots & x_n^n \end{bmatrix}}_{\mathbf{X}} \underbrace{\begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix}}_{\mathbf{a}} = \underbrace{\begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix}}_{\mathbf{y}}.$$

Somit ist der Koeffizientenvektor $\mathbf{a}$ gegeben durch $\mathbf{a} = \mathbf{X}^{-1}\mathbf{y}$ (MATLAB: `a = X\y`).
Die Matrix $\mathbf{X}$ kann in MATLAB spaltenweise aufgebaut werden. Dazu ist es am einfachsten, wenn die Stützstellen x_i als ein Spaltenvektor $\mathbf{x}$ dargestellt werden. Das Codestück zum Aufbau der Matrix könnte dann folgende Struktur haben:

```
X = zeros(n+1);
for j=0:n;
    X(:,j+1) = ...
end
```

Um ein Polynom, das in monomialer Darstellung vorliegt, an einer bestimmten Stelle auszuwerten, schreibt man das Polynom am Besten zuerst einmal um. Es gilt:

$$
\begin{aligned}
p(z) &= a_n z^n + a_{n-1} z^{n-1} + \ldots + a_1 z + a_0 \\
&= (\ldots((a_n z + a_{n-1})z + a_{n-2})z + \ldots)z + a_0.
\end{aligned}
$$

Die zweite Form hat den Vorteil, dass sie mit einem Minimum an Multiplikationen auskommt. Daher ist eine Auswertung in dieser Form schneller. In MATLAB könnte man dies mit Hilfe einer `for`-Schleife umsetzen und damit diesen Ausdruck von innen her berechnen. In jedem Schritt wird dadurch eine weitere Klammer verrechnet. Dies ergäbe dann z.B. folgende Struktur:

```
p = a(n+1);
for j = n-1 : -1 : 0
    p = ...
end
```

Dieses Vorgehen zum Auswerten eines Polynoms wird *Horner-Schema* genannt.

Lagrange-Interpolation

Für die Lagrange-Interpolation ist

$$
p(z) = \sum_{j=0}^{n} b_j \, L_j(z),
$$

wobei L_j das j-te *Lagrange-Polynom* bezeichnet:

$$
L_j(z) = \prod_{k \neq j} \frac{z - x_k}{x_j - x_k}.
$$

Auch hier können die Gewichte b_j durch die Bedingung $p(x_i) = y_i$ für $i = 0, 1, \ldots, n$ bestimmt werden. Da aber $L_i(x_j) = \delta_{ij}$ (Kronecker-Symbol) gilt, gibt es jedoch eine grosse Vereinfachung, denn es gilt $b_j = y_j$ für $j = 0, 1, \ldots, n$.
Die Auswertung des Integrationspolynoms ist jedoch schwieriger. Der Hauptteil des Programms ist eine geschachtelte `for`-Schleife. Die äussere Schleife ist für die Summe und die innere Schleife für das Produkt, welches bei jedem Summanden neu berechnet werden muss. Der Hauptteil des Programms `lagrange_interpol` könnte somit wie folgt aussehen:

```
p = 0;
for j = 0:n % berechne Summe über alle Lagrange-Polynome
    L = 1;
    for k = 0:n % berechne das Produkt eines einzelnen Lagrange-Polynoms
        if k~=j
            L = ...
        end
    end
    p = ...
end
```