

Programmieren für Physiker

Interfakultatives Institut für Anwendungen der Informatik
Institut für Theoretische Teilchenphysik

Prof. Dr. M. Steinhauser, Dr. A. Mildenerger
<http://comp.physik.kit.edu>

SS 2023 – Blatt 06
Bearbeitung bis 07. Juni MMXXIII

Aufgabe 15: Interpolation mit kubischen Splines

Pflichtaufgabe

Gegeben seien die Stützstellen $\Delta = \{a = x_0 < x_1 < \dots < x_n = b\}$ und die zugehörigen Funktionswerte $Y = \{y_0, y_1, \dots, y_n\}$. In dieser Aufgabe soll eine kubische Splinefunktion $S_\Delta(Y; x)$ durch diese Punkte bestimmt werden. Bei kubischen Splines werden abschnittsweise Polynome dritter Ordnung durch je zwei aufeinanderfolgende Punkte gelegt, so dass die Polynome selbst und ihre ersten beiden Ableitungen stetig an den Stellen x_i , $i = 1, \dots, n - 1$ anschließen. Das Problem wird eindeutig lösbar, wenn zusätzlich zwei Randbedingungen vereinbart werden: In dieser Aufgabe soll gefordert werden, dass die zweite Ableitung der Splinefunktion an den Rändern $x = a$ und $x = b$ verschwindet.

Zentral bei der Berechnung der kubischen Splinefunktion sind die sogenannten „Momente“, die zweiten Ableitungen an den Stützstellen $M_j := S''_\Delta(Y, x_j)$. Für diese gilt das folgende tridiagonale $(n + 1) \times (n + 1)$ Gleichungssystem

$$\begin{pmatrix} 2 & \lambda_0 & 0 & \dots & 0 \\ \mu_1 & 2 & \lambda_1 & \ddots & \vdots \\ 0 & \mu_2 & 2 & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \lambda_{n-1} \\ 0 & \dots & 0 & \mu_n & 2 \end{pmatrix} \cdot \begin{pmatrix} M_0 \\ M_1 \\ M_2 \\ \vdots \\ M_n \end{pmatrix} = \begin{pmatrix} d_0 \\ d_1 \\ d_2 \\ \vdots \\ d_n \end{pmatrix}$$

mit den Größen ($j = 1, \dots, n - 1$)

$$\lambda_j = \frac{x_{j+1} - x_j}{x_{j+1} - x_{j-1}}, \quad \mu_j = \frac{x_j - x_{j-1}}{x_{j+1} - x_{j-1}}, \quad d_j = \frac{6}{x_{j+1} - x_{j-1}} \left(\frac{y_{j+1} - y_j}{x_{j+1} - x_j} - \frac{y_j - y_{j-1}}{x_j - x_{j-1}} \right)$$

und den Randbedingungen $\lambda_0 = \mu_n = d_0 = d_n = 0$.

In diesem System wird zunächst die untere Nebendiagonale eliminiert. Setze $\mu_0 = 2$ und führe für $i = 1, 2, \dots, n$ nacheinander aus:

$$\begin{aligned} f &:= -\mu_i / \mu_{i-1}, \\ \mu_i &:= 2 + f \cdot \lambda_{i-1}, \\ d_i &:= d_i + f \cdot d_{i-1}. \end{aligned}$$

Die neu berechneten Werte μ_0, \dots, μ_n bilden dann die Diagonale des umgeformten Systems, die Nebendiagonale $\lambda_0, \dots, \lambda_{n-1}$ bleibt unverändert.

Nun die Rücksubstitution: $M_n := d_n / \mu_n$ und dann für $i = n - 1, \dots, 0$: $M_i = (d_i - \lambda_i M_{i+1}) / \mu_i$. Mit den M_i sind die Koeffizienten von $S_\Delta(Y; x)$ vollständig bekannt:

$$S_\Delta(Y; x) = \alpha_j + \beta_j(x - x_j) + \gamma_j(x - x_j)^2 + \delta_j(x - x_j)^3 \quad \text{für } x \in [x_j, x_{j+1}]$$

mit

$$\alpha_j = y_j, \quad \beta_j = \frac{y_{j+1} - y_j}{x_{j+1} - x_j} - \frac{2M_j + M_{j+1}}{6}(x_{j+1} - x_j), \quad \gamma_j = M_j/2, \quad \delta_j = \frac{M_{j+1} - M_j}{6(x_{j+1} - x_j)}.$$

Implementieren Sie dieses Verfahren in einem C++-Programm. Verwenden Sie zum Testen die Datei `a15-spline-input.txt`, die einige Wertepaare (x, y) enthält. Sie dürfen voraussetzen, dass die x -Werte aufsteigend geordnet sind.

Werten Sie die Splinefunktion an 300 äquidistanten x -Werten zwischen minimaler und maximaler Stützstelle aus und schreiben Sie jeweils x und den Wert der Splinefunktion zeilenweise in eine neue Ergebnisdatei. Die ursprüngliche Punktedatei und die von Ihnen erzeugte Ergebnisdatei schauen Sie sich bitte in einem geeigneten Plot-Werkzeug (z.B. `python/matplotlib`) an. Stellen Sie bitte die Ausgabe so ein, dass die gegebenen und die neu berechneten Funktionswerte durch Punkte und verschiedene Farben erkennbar sind.

Aufgabe XVI: Römisch → Arabisch

Pflichtaufgabe

Klasse, dass die Rom-Reise nun klappt. Aber jetzt fragt Sie Ihre altphilologische Reisebegleitung ständig, ob Sie mal eben die römische Jahreszahl an der oder jener Inschrift umrechnen könnten. Ach, was wäre es toll, ein Programm dafür zu haben. . .

Na denn. Es gibt sieben römische Zahlzeichen:

römisch	M	D	C	L	X	V	I
Wert	1000	500	100	50	10	5	1

Im Wert absteigende oder gleich große Zahlzeichen werden einfach addiert (Bsp. XVII \equiv 17). Falls aber irgendwo ein kleinerer Zahlwert direkt vor einem größeren Wert auftritt, wird die kleinere Zahl subtrahiert, die folgende größere wieder addiert (Bsp. XIV \equiv 14). Zwei unmittelbar aufeinanderfolgende Subtraktionen kommen in regulär gebildeten Zahlen nicht vor.

Lassen Sie den Benutzer ein `char`-Feld eingeben. Nach `#include <cstring>` können Sie mit `strlen` die Länge des Zahlentexts ermitteln. Rechnen Sie dann mit oben angegebener Regel (Addition oder Subtraktion der einzelnen Zeichenwerte) die Gesamtzahl aus und geben Sie diese aus. Sie brauchen nicht zu untersuchen, ob die eingegebene römische Zahl einen gültigen regulären Aufbau besitzt.

Schreiben und verwenden Sie bitte folgende Funktionen: Eine Funktion `char to_upper (char)`, die ein Buchstaben-Argument als Großbuchstaben zurück gibt. Weiterhin eine Funktion `int value_of_rmchar (char)`, die von einem einzelnen römischen Zahlzeichen den Wert nach obiger Tabelle ermittelt. Greifen Sie auf die Funktion `to_upper` zurück, so dass Ihr Programm für Groß- und Kleinbuchstaben funktioniert.

Zusatz (freiwillig): Die umgekehrte Fragestellung: Wie kann man eine eingegebene arabische Zahl in die römische Zahldarstellung umrechnen?

Zum Knobeln: Nim-Spiel

freiwillig

Beim Nim-Spiel in der hier betrachteten Variante gibt es zwei Spieler und zwei Häufen mit Streichhölzern auf dem Tisch. Die Spieler sind abwechselnd am Zug und haben dabei jeweils folgende zwei Möglichkeiten: entweder von genau einem Haufen ihrer Wahl eine beliebige Zahl (≥ 1) an Hölzchen zu entfernen oder aber von beiden Häufen genau die gleiche Zahl an Hölzchen zu entfernen. In jedem Zug muss also immer mindestens ein Streichholz weggenommen werden. Wer mit seinem Zug das letzte Streichholz (oder die letzten Streichhölzer) wegnehmen kann, hat gewonnen.

Schreiben Sie zunächst ein C++-Programm, welches Sie dieses Spiel mit zwei Spielern spielen lässt. In diesem Fall soll das Programm also die Streichholzhäufen verwalten und dabei von beiden Spielern abwechselnd gültige Züge erfragen, bis ein Spieler gewonnen hat.

Interessant ist es, sich zu überlegen, welche Strategie in diesem Spiel zum Gewinn führt. Gibt es Haufengrößen, von denen aus der Sieg erzwungen werden kann? Ersetzen Sie in Ihrem Programm einen Mitspieler durch den Computer und bringen Sie ihm Ihre Gewinnstrategie bei.

Hinweis: Es ist zweckmäßig, mit zwei zufälligen Stapeln von je etwa 5 bis 25 Streichhölzern zu beginnen. Um dies zu erreichen, können Sie vom Rechner ermittelte Zufallszahlen verwenden. Nach der Anweisung `#include <cstdlib>` und `#include <ctime>` im Kopf des Programms können Sie mit `rand()%a` ganzzahlige Zufallszahlen im Intervall $[0, a[$ erzeugen. Um den Generator auf einen nicht vorherbestimmten Anfangswert zu setzen, benötigen Sie noch ein einmaliges `srand((unsigned int)time(0))` am Anfang des Programmcodes.
