Peter DEUFLHARD, Berlin

Bessel'scher Irrgarten

Einleitung

Dieser Artikel behandelt das allgemeine Thema "Rundungsfehler müssen nicht klein sein" exemplarisch an einem in Leistungskursen am Gymnasium realisierbaren Beispiel, der Drei-Term-Rekursion für die Bessel-Funktionen. Eine ausführliche Darstellung des Themas findet sich in [2]. Das hier beschriebene Vorgehen ist seit vielen Jahren bei Besuchen von Schülergruppen am ZIB erprobt und verfeinert worden.

Bessel-Rekursion

Zentrales Hilfsmittel für diese Unterrichtseinheit ist die Eigenschaft, dass Bessel-Funktionen einer speziellen sogenannten *Drei-Term-Rekursion* genügen:

$$J_{k+1} = \frac{2k}{x} J_k - J_{k-1}, \quad k = 1, 2, \dots \ . \tag{1}$$

Falls, etwa für das Argument $x = 2.13$, die beiden Werte

$$J_0 = 0.149\ 606\ 770\ 448\ 844, \quad J_1 = 0.564\ 996\ 980\ 564\ 127 \tag{2}$$

auf 15 Stellen genau gegeben sind, so lassen sich aus der Beziehung (1) rekursiv die Folgewerte

$$\begin{aligned} J_2 &= \frac{2 * 1}{2.13} * J_1 - J_0 = 0.380\ 906\ 826\ 324\ 984 \\ J_3 &= \frac{2 * 2}{2.13} * J_2 - J_1 = 0.150\ 321\ 003\ 144\ 763 \\ &\vdots \end{aligned}$$

bis hin zu einem Zielwert J_N (für $k = N - 1$ in (1)) berechnen.
Falls Werte J_N, J_{N-1} vorgegeben sind, so lässt sich daraus durch "Umkehrung" der Drei-Term-Rekursion gemäß

$$J_{k-1} = \frac{2k}{x} J_k - J_{k+1}, \quad k = N - 1, N - 2, \dots, 1 \ , \tag{3}$$

der Zielwert J_0 berechnen. Die Rekursion (1) wird als *Vorwärtsrekursion* bezeichnet, entsprechend (3) als *Rückwärtsrekursion.*
Darüberhinaus genügen die Besselfunktionen noch einer *Summenbeziehung* der Form:

$$J_0 + 2\sum_{k=1}^{\infty} J_{2k} = 1 \ . \tag{4}$$

Ausdrücklich sei noch darauf hingewiesen, dass auch die sogenannten *Neumann-Funktionen* $\{Y_k(x)\}$ der Drei-Term-Rekursion (1) genügen. Um sie aus der Vorwärtsrekursion zu erhalten, sind natürlich Startwerte Y_0, Y_1 vorzugeben, während die Rückwärtsrekursion Startwerte Y_N, Y_{N-1} benötigt.
Merke: Erst durch *zwei* Werte ist festgelegt, welche dieser speziellen Funktionen man aus der Drei-Term-Rekursion erhält.
Die oben dargestellte Rekursion (1) eignet sich hervorragend zur Berechnung sämtlicher Bessel-Funktionen, wenn zwei Startwerte bekannt sind. Wir gehen zunächst von den Werten aus (2) für J_0, J_1 zu $x = 2.13$ aus und rechnen bis J_{23}. In kleinen Schülergruppen (3-4), die sich selbst zusammenfinden, oftmals Schülerinnen gegen Schüler, wird dann "um die Wette" gerechnet. Die Schüler schreiben entweder ein kleines Programm oder sie lösen das Problem in der Gruppe durch sukzessives Eintippen und Aufschreiben der Zwischenresultate. Gut ist es, wenn verschiedene Gruppen verschiedene Taschenrechner haben. Die ersten beiden fertigen Gruppen dürfen dann ihre Resultate an die Tafel schreiben. In aller Regel sind diese Resultate verschieden, oftmals sehr verschieden. In Tabelle 1 geben wir die Resultate an, die wir durch Vergleich eines Schulrechners mit dem "Supercomputer" des ZIB erzielt haben.
Idee: Kontrolle durch Rückwärtsrekursion. Seien die numerisch erhaltenen Resultate mit $\tilde{J}_{23}, \tilde{J}_{22}$ bezeichnet. Sie werden in die Rekursion (3) für $N = 23$ eingesetzt, um daraus Werte $\tilde{J}_0, \tilde{J}_1$ zu berechnen. Wenn die Zahlen stimmen, sollte approximativ gelten: $\tilde{J}_0 \approx J_0, \ \tilde{J}_1 \approx J_1$. Dies zeigt sich aber keineswegs! Zur Illustration des Effektes stellen wir in Abbildung 1 die Resultate einer Wiederholung des Paares "Vorwärtsrekursion-Rückwärtsrekursion" graphisch dar (logarithmische Skala). Offenbar gibt es keine Chance, zu den richtigen Werten zurückzukehren: wir sind im "Bessel'schen Irrgarten" gelandet.

k	TI-30Xa Solar	IBM pSeries 690
0	1.49606770 E–01	1.49606770448844 E–01
1	5.64996981 E–01	5.64996980564127 E–01
2	3.80906827 E–01	3.80906826324984 E–01
3	1.50321004 E–01	1.50321003144764 E–01
4	4.2532622* E–02	4.25326191532232 E–02
5	9.425932** E–03	9.42592325231867 E–03
6	1.720581** E–03	1.72054165578469 E–03
7	2.67483*** E–04	2.67269174637329 E–04
8	3.7525**** E–05	3.61571446484618 E–05
9	1.4396**** E–05	4.33378985815840 E–06
10	8.4135**** E–05	4.66431617665503 E–07
11	7.75608*** E–04	4.58497443345861 E–08
12	7.926841** E–03	7.13381677623191 E–09
13	8.8540907* E–02	3.45312897638016 E–08
14	1.072854187E+00	4.14374884565947 E–07
15	1.401470663E+01	5.41265029138480 E–06
16	1.963173800E+02	7.58201362616988 E–05
17	2.935354382E+03	1.13366920903930 E–03
18	4.665910469E+04	1.80203080831450 E–02
19	7.586692319E+05	3.03434918111721 E–01
20	1.396997508E+07	5.39537259719639 E+00
21	2.615882806E+08	1.01018116202947 E+02
22	5.144108798E+09	1.98651114408063 E+03
23	1.060016920E+11	4.09348928413313 E+04

Tabelle 1: Taschenrechner gegen Supercomputer (* wegen Festkomma-Darstellung auf TI)

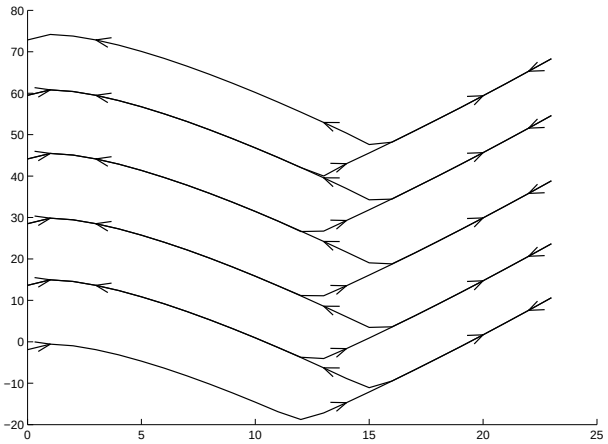

Abbildung 1: $N = 23$: Bessel'scher Irrgarten (logarithmische Skala)

In Tabelle 2 vergleichen wir abschließend die richtigen Zahlen mit den Zahlen von Tabelle 1, die wir mit den beiden unterschiedlichen Computern in Vorwärtsrekursion berechnet hatten. Das Resultat ist klar: beide Rechner lagen falsch, auch der Supercomputer war nicht super!
Nachfolgend werden zwei alternative Algorithmen angegeben, mit denen die "richtigen" Werte berechnet werden können.

k	TI-30Xa Solar	IBM pSeries 690	exakt
0	1.49606770 E–01	1.49606770448844 E–01	1.49606770448844E–01
1	5.6499698**1** E–01	5.64996980564127 E–01	5.64996980564127E–01
2	3.8090682**7** E–01	3.80906826324984 E–01	3.80906826324984E–01
3	1.5032100**4** E–01	1.50321003144764 E–01	1.50321003144763E–01
4	4.2532622* E–02	4.25326191532232 E–02	4.25326191532221E–02
5	9.425932** E–03	9.42592325231867 E–03	9.42592325231519E–03
6	1.720581** E–03	1.72054165578469 E–03	1.72054165576939E–03
7	2.67483*** E–04	2.67269174637329 E–04	2.67269174554612E–04
8	3.7525**** E–05	3.61571446484618 E–05	3.61571441200797E–05
9	1.4396**** E–05	4.33378985815840 E–06	4.33378597180825E–06
10	8.4135**** E–05	4.66431617665503 E–07	4.66399303652013E–07
11	7.75608*** E–04	4.58497443345861 E–08	4.55502127176888E–08
12	7.926841** E–03	7.13381677623191 E–09	4.07237700017248E–09
13	8.8540907* E–02	3.45312897638016 E–08	3.35725312423605E–10
14	1.072854187E+00	4.14374884565947 E–07	2.56784566414815E–11
15	1.401470663E+01	5.41265029138480 E–06	1.83186408413325E–12
16	1.963173800E+02	7.58201362616988 E–05	1.22445951944503E–13
17	2.935354382E+03	1.13366920903930 E–03	7.69951315506101E–15
18	4.665910469E+04	1.80203080831450 E–02	4.57074943794664E–16
19	7.586692319E+05	3.03434918111721 E–01	2.56971625952894E–17
20	1.396997508E+07	5.39537259719639E+00	1.37208842176626E–18
21	2.615882806E+08	1.01018116202947E+02	6.97561233257824E–20
22	5.144108798E+09	1.98651114408063E+03	3.38443254494399E–21
23	1.060016920E+11	4.09348928413313E+04	1.57037227051201E–22

Tabelle 2: *Vorwärtsrekursion:* Vergleich der berechneten Zahlen mit den "exakten" Zahlen, d.h. hier auf 15 Stellen genau.

Klassischer Algorithmus

Der englische Mathematiker J. C. P. Miller (1906–1981) machte 1952 die Beobachtung, dass Bessel-Funktionen sehr wohl in Rückwärtsrichtung korrekt berechenbar sind.

Eine Erklärung dieses Phänomens ist, dass die Bessel-Rekursion einen zweidimensionalen Lösungsraum hat, mit Basis etwa die Bessel-Funktionen $\{J_k\}$ und die Neumann-Funktionen $\{Y_k\}$. Asymptotisch gilt:

$$\lim_{k\to\infty} J_k = 0 \ , \quad \lim_{k\to\infty} Y_k = \infty \ . \tag{5}$$

Auf der Basis dieser Einsicht entwickelte er einen Algorithmus, der heute seinen Namen trägt. Er sei hier kurz informell angegeben:

Algorithmus 1: Miller-Algorithmus

1. Wähle einen *Abbrechindex* N und einen "kleinen" Wert $\sigma > 0$. Das Argument x sei vorgegeben.
2. Wähle Startwerte $\bar{J}_N = 0 \ , \bar{J}_{N-1} = \sigma$.
3. Berechne Werte $\bar{J}_k$ aus der Rückwärtsrekursion (3).
4. Berechne den Skalierungsfaktor (vergleiche (4))
$$\kappa_N = \bar{J}_0 + 2 \sum_{k=1}^{[N/2]} \bar{J}_{2k} \ .$$
5. Berechne Approximationen der Bessel-Funktionen gemäß
$$J_k^{(N)} = \frac{\bar{J}_k}{\kappa_N} \ , \quad k = 0, \ldots, n < N \ .$$

Als Approximation der Summenbeziehung (4) gilt hierbei offenbar

$$J_0^{(N)} + 2 \sum_{k=1}^{N} J_{2k}^{(N)} = 1 \ .$$

Es bleibt die Frage: Wie groß ist N bei beliebig vorgegebenem x zu wählen? Als Strategie wird man die Rechnungen mit wachsendem N so oft wiederholen, bis die Resultate ausreichend genau sind, also hinreichend viele Stellen stehen bleiben. Dabei ist es störend, dass beim Übergang von N nach $N+1$ immer die gesamte Rückwärtsrekursion mit etwa $4N$ Operationen durchgerechnet werden muß.

Ein Horner-artiger Algorithmus

Der Horner-Algorithmus zur ökonomischen Auswertung von Polynomen gehört an vielen Schulen zum Standardstoff. Der hier vorgelegte Algorithmus für die Auswertung von Bessel-Reihen baut darauf unmittelbar auf.
Die Auswertung von Polynomen verlangt die Berechnung von Summen der Form

$$S_n(x) = \sum_{k=0}^{n} A_k \; x^k \; .$$

Geeignetes Ausklammern führt dabei auf den Horner-Algorithmus, bei dem die Summe "von hinten her" ausgewertet wird:

$$\begin{aligned} S_n(x) &= A_0 + A_1 \; x + \ldots + A_{n-1} \; x^{n-1} + A_n \; x^n \\ &= A_0 + x \; (A_1 + \ldots + x \; (A_{n-1} + x \; A_n)) \; . \end{aligned}$$

Hier wollen wir den Algorithmus etwas anders herleiten. Wir gehen aus von der Tatsache, dass die Polynome $P_k = x^k$ einer *Zwei-Term-Rekursion* genügen:

$$P_{k+1} = x \; P_k \; .$$

Mit dem bequemen Startwert $P_0 = 1$ ist dies eine (homogene) *Vorwärtsrekursion*. Der Effekt des Ausklammerns lässt sich darstellen als eine inhomogene Zwei-Term-Rekursion der Form

$$\begin{aligned} U_n &= A_n \\ U_{n-1} &= A_{n-1} + x \; U_n \\ &\vdots \\ U_0 &= A_0 + x \; U_1 \\ S_n &= U_0 \; . \end{aligned}$$

Dies ist offenbar eine Rückwärtsrekursion, die in der Literatur als "adjungierte" Rückwärtsrekursion bezeichnet wird. Der Horner-Algorithmus lässt sich demnach auch als "adjungierte Summation" [1] bezeichnen.
Diese Sichtweise liefert den Schlüssel zur Erweiterung des Horner-Algorithmus auf allgemeine Drei-Term-Rekursionen, die 1977 von P. Deuflhard geleistet

worden ist. Der Algorithmus eignet sich allgemein zur Berechnung von Summen der Form

$$S_n^{(N)}(x) = \sum_{k=0}^{n} A_k \; J_k^{(N)}(x) \; ,$$

wobei der Abbrechindex N adaptiv so gewählt wird, dass $S_n^{(N)}$ korrekt auf vorgegebene Genauigkeit ϵ ist.

Algorithmus 2: Adjungierte Summation von Bessel-Funktionen

$g := \Delta u := 0; \bar{f} := -1; u := A_0; N := 1$
repeat
  $\bar{m} := 2\mathbf{mod}(N+1, 2)\bar{f};$
  $\Delta u := \bar{f} A_N - g\Delta u - \bar{m} u;$
  $g := \bar{m} - 2N/x + g;$
  $\Delta u := \Delta u / g;$
  $u := u + \Delta u;$
  if $(N > n$ **and** $|\Delta u| \leq |u| \cdot \epsilon)$ **then exit**; (Lösung $S_n \approx u$)
  $g := -1/g;$
  $\bar{f} := \bar{f} g;$
  $N := N + 1;$
until $(k > n_{\max})$

In Tabelle 3 sind einige Beispiele angegeben. Wie schön zu sehen ist, hängt das "Stehenbleiben gültiger Ziffern" sowohl vom Argument x also auch von den Koeffizienten in der Summe ab. Beim Übergang von N nach $N+1$ benötigt dieser Algorithmus nur $O(1)$ Operationen.

N	$S = J_0^{(N)}(1024.13) + \sum_{k=1}^{1024} k^2 J_k^{(N)}(1024.13)$	N	$J_{10}^{(N)}(10.13)$
1000	9.873630161365381E+05	10	1.722923265030349E-01
1010	1.009291691691664E+06	12	2.273511331914537E-01
1020	1.028158041678334E+06	14	2.203454815320414E-01
1030	9.073090336047271E+05	16	2.185693492939857E-01
1040	7.435725532242977E+05	18	2.183830486762933E-01
1050	7.150012371538359E+05	20	2.183689776345255E-01
1060	7.123942230060821E+05	22	2.183681224273101E-01
1070	7.122257868444083E+05	24	2.183680793897036E-01
1080	7.122176541637876E+05	26	2.183680775681829E-01
1090	7.122173576457674E+05	28	2.183680775024694E-01
1100	7.122173493609600E+05	30	2.183680775004246E-01
1110	7.122173491807199E+05	32	2.183680775003691E-01
1120	7.122173491776222E+05	34	2.183680775003678E-01
1130	7.122173491775796E+05	36	2.183680775003678E-01
1140	7.122173491775794E+05		

Tabelle 3: Adjungierte Summation.
Links: spezielle Bessel-Reihe zu $A_0 = 1, A_k = k^2$, $x = 1024.13$ und $n = 1024$.
Rechts: Bessel-Funktion J_{10} zu $x = 10.13$.

Literatur

[1] P. Deuflhard, A. Hohmann. Numerische Mathematik I: Eine algorithmisch orientierte Einführung. 3. überarb. u. erw. Aufl., de Gruyter, Berlin, New York, 2002

[2] P. Deuflhard, U. Nowak, B. Lutz-Westphal: Bessel'scher Irrgarten. ZIB Report 07-09, 2007