PRESEK
List za mlade matematike, fizike, astronome in računalnikarje

ISSN 0351-6652
Letnik **14** (1986/1987)
Številka 2
Strani 116–124

Sandi Klavžar in Matija Lokar:

RAČUNANJE VREDNOSTI NEKATERIH MATEMATIČNIH FUNKCIJ

Ključne besede: matematika, računalništvo, algoritmi, elementarne funkcije, kalkulator HP.

Elektronska verzija: http://www.presek.si/14/826-Lokar.pdf

RAČUNANJE VREDNOSTI NEKATERIH MATEMATIČNIH FUNKCIJ

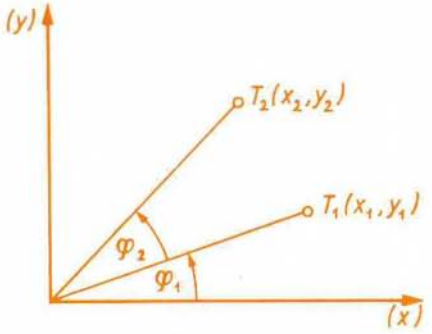
Uvod

Verjetno ste se že večkrat vprašali, kako kalkulatorji ali pa tudi vaši domači računalniki računajo vrednosti različnih matematičnih funkcij. Če je funkcija dovolj "lepa", in take so prektično vse funkcije, ki jih srečamo v srednji šoli, potem lahko vrednost funkcije zapišemo v obliki neskončne vrste. Učeno bi lahko dejali, da funkcijo razvijemo v Taylorjevo vrsto. Tako na primer lahko zapišemo identiteto za funkcijo $\sin(x)$:

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots$$

Neskončnega računa seveda ne moremo opraviti, toda če vzamemo zadostno število členov vrste, lahko dobimo poljubno natančnen približek. Na prvi pogled smo tako problem računanja funkcij rešili, in to celo zelo splošno. Žal pa je v računalništvu, za razliko od matematike, ponavadi hitrost algoritma važnejši kriterij kot splošnost rešitve. Problem pri računanju vrednosti funkcij je s pomočjo Taylorjeve vrste je v tem, da moramo pri velikem argumentu x za dosego željene natančnosti upoštevati precej členov. To pa nasprotuje želji po učinkovitosti (hitrosti).

Dobri algoritmi ponavadi zahtevajo globlje poznavanje problema, ki ga re-



šujemo. Še preden si ogledamo, kako večina kalkulatorjev firme Hewlett–Packard računa kvadratni koren in trigonometrične funkcije, izpeljimo formuli, ki ju bomo potrebovali kasneje.

Naj bo T_1 poljubna točka v ravnini s koordinatama (x_1, y_1). V polarnih koordinatah se koordinati točke T_1 glasita:

$$x_1 = r.\cos(\varphi_1) \qquad y_1 = r.\sin(\varphi_1) \tag{1}$$

kjer je r razdalja točke T_1 od koordinatnega izhodišča in φ_1 kot, ki ga poltrak iz izhodišča od T_1 oklepa z osjo x. Zavrtimo točko T_1 za kot φ_2 v pozitivni smeri okrog izhodišča v točko T_2 (slika 1).

Vprašajmo po koordinatah točke T_2. Najprej je:

$$x_2 = r.\cos(\varphi_1 + \varphi_2) \qquad y_2 = r.\sin(\varphi_1 + \varphi_2)$$

Če uporabimo za sin in cos adicijska teorema in upoštevamo (1), dobimo formuli:

$$\begin{aligned} x_2 &= x_1.\cos(\varphi_2) - y_1.\sin(\varphi_2) \\ y_2 &= y_1.\cos(\varphi_2) + x_1.\sin(\varphi_2) \end{aligned} \tag{2}$$

Povejmo še, da kalkulatorji HP uporabljajo posebne mikroprocesorje, pri katerih množenje z 10^n ne pomeni nič drugega kot premik decimalne pike. Prednost desetiške aritmetike je v tem, da ni potrebno pretvarjanje med dvojiškim in desetiškim sistemom. Če bi bila aritmetika dvojiška, bi računali dvojiške cifre, ki bi jih potem pomnožili z ustreznimi potencami 2^n.

Kvadratni koren

Kako bi izračunali kvadratni koren, če bi bili brez kalkulatorja, v usmerjeni šoli pa vas tega niso naučili? Izberimo neki približek, ki se nam zdi primeren, in preverimo, koliko sreče smo imeli, tako da izračunamo kvadrat približka. Če smo dobili premalo, smo izbrali premajhen približek, če smo dobili preveč, je bil približek prevelik, sicer pa smo imeli srečo in zadeli rešitev. Ker je verjetnost zadetka majhna, postopek nadaljujmo tako, da približek izboljšamo, To delamo toliko časa, da smo z rezultatom zadovoljni. Zapišimo opisani postopek še v algoritmičnem jeziku.

Vzemi začetni približek za koren a

ponavljaj
izračunaj a^2
$R \leftarrow x - a^2$
če je R dovolj majhen **potem končaj**
sicer popravi a glede na predznak R
do tod

Seveda ta postopek še ni dober. Kalkulatorju je treba povedati, za koliko naj popravi a, razen tega pa zahteva vsak korak kar precej časa: izračun a^2 in ostanka R. Algoritem torej spremenimo tako, da računanje a^2 in R opravimo kar najlažje, ko spremenimo približek a. To naredimo tako, da postopoma računamo približek: najprej določimo pravilne tisočice, nato stotice, desetice, enice, ... Zdaj spoznamo v njem starega znanca (ali pa tudi ne). Tako vendar računamo kvadratni koren "peš" s papirjem in svinčnikom.

Vpeljimo nekaj oznak, ki nam bodo olajšale nadaljnje delo:

x . . . . število, katerega koren računamo
a . . . . približek za koren
b . . . . naslednja cifra $x^{1/2}$, ki jo iščemo
j . . . . eksponent potence števila 10
R_a . . . $x - a^2$ tekoči ostanek
a_j . . . . novi a, ko dodamo b na ustrezno mesto: $a_j = a + b \cdot 10^j$
in R_b naj bo enak $a_j^2 - a^2$

Število $x^{1/2}$ bomo poiskali tako, da se bomo pravi vrednosti bližali od spodaj. Torej mora v vsakem trenutku veljati pogoj:

$$a \leqslant x^{1/2}$$

in s tem

$$R_a \geqslant 0$$

Ostanek moramo karseda zmanjšati, kljub temu pa še ostati pod $x^{1/2}$. Zato mora biti b največje tako število, a bo veljalo:

$$R_a - R_b \geqslant 0$$

Če upoštevamo definicijo R_b

$$R_b = (a + (b \,.\, 10^j))^2 - a^2 = 2a\,b \,.\, 10^j + (b \,.\, 10^j)^2$$

je b tisto največje število, da velja:

$$2ab \,.\, 10^j + (b \,.\, 10^j)^2 \leqslant R_a$$

Ko najdemo največje naravno število b, ki tej neenačbi zadošča, izračunamo nov približek

$$a \leftarrow a + b \,.\, 10^j$$
$$R_a \leftarrow R_a - R_b$$
$$j \leftarrow j - 1$$

Oglejmo si primer: $x = 54756, j = 1, a = 200, R_a = 14756$.
Zaporedoma računamo nove ostanke:

b	R_b	$R_a - R_b$
0	0	14756
1	4100	10656
2	8400	6356
3	12900	1856
4	17600	−2844

Torej moramo za b vzeti 3, ker je $b = 4$ že preveč. Novi približek je tedaj $a = 200 + 3.\,101 = 230$, novi ostanek $R_a = 1856$ in $j = 0$.
Postopek nato ponovimo tolikokrat, da dosežemo željeno natančnost.

Algoritem pa lahko tudi izboljšamo. Prva taka izboljšava je način, kako izrazimo $(b \,.\, 10^j)^2$. Pri tem upoštevamo, da je b^2 vsota prvih b lihih števil. Od tod:

$$(b \,.\, 10^j)^2 = b^2 \,.\, 10^{2j} = \Sigma\,(2i - 1).\,10^{2j}$$

prištejmo $2a\,b \,.\, 10^j$ in dobimo

$$2a\,b \,.\, 10^j + (b \,.\, 10^j)^2 = \Sigma\,(2a \,.\, 10^j + (2i - 1) \,.\, 10^{2j})$$

Leva stran je novi ostanek R_b. Namesto da gledamo R_a in R_b, opazujmo $5R_a$ in $5R_b$, saj če velja $R_b \leqslant R_a$, velja tudi $5R_b \leqslant 5R_a$. Ostanek R_b ima obliko

$$5R_b = \Sigma\,10a \,.\, 10^j + (10i - 5).\,10^{2j}$$

postopek pa ostane nespremenjen, le da iščemo tisto največje naravno število b, za katerega velja $5R_b \leqslant R_a$, novi ostanek pa je $5R_a = 5R_a - 5R_b$. Transformacija je na prvi pogled nesmiselna. Če pa si ogledamo zaporedne R_b in upoštevamo, da zaradi BCD mikroprocesorja množenje z 10 pomeni le rotacijo v desno, dobi transformacija smisel.

$b = 1 \qquad R_b = 10a \,.\, 10^j + 05 \,.\, 10^{2j}$
$b = 2 \qquad R_b = 10a \,.\, 10^j + 15 \,.\, 10^{2j}$
$b = 3 \qquad R_b = 10a \,.\, 10^j + 25 \,.\, 10^{2j}$

Vidimo, da imajo ostanki R_b obliko $10a \,.\, 10^j + (b-1)|5 \,.\, 10^{2j}$, kjer oznaka $|5$ pomeni dopisovanje petice. Poglejmo tako popravljeni postopek na našem primeru ($x = 54756, j = 1, a = 200, 5R_a = 73780$).
Zaporedoma računamo nove ostanke:

b	$10a.\,10^j + (b-1)\|5.10^{2j}$	$R_a - R_b$	
1	20500	53280	
2	21500	31780	
3	22500	9280	(novi $5R_a$)
4	23500	−14220	prekoračitev

in naslednji korak ($j = 0, a = 230, 5R_a = 9280$):

b	$10a.10^j + (b-1)\|5.10^{2j}$	$R_a - R_b$	
1	2305	6975	
2	2315	4660	
3	2325	2335	
4	2335	0	
5	2345	−2345	prekoračitev

Če pogledamo obe tabeli, opazimo, da vrednosti v srednjem stolpcu ni treba računati, saj jih dobimo enostavno s spajanjem naračunanih cifer približka, petice in še $2j$ ničel. Ničlo potem nadomeščamo zaporedoma s ciframi 1, 2, ..., tako da je edino delo, ki ga še mora opraviti računalnik, računanje razlike $5R_a - 5R_b$. Pri ogledu algoritma smo vzeli kot primer veliko število. Dejansko pa je v kalkulatorjih tipa HP število predstavljeno v eksponentnem zapisu z mantiso M in eksponentom exp:

$$x = M \,.\, 10^{\exp}$$

kjer je $1 \leqslant M < 10$. Poleg mantise je treba koreniti tudi potenco 10^{exp}. S sodim eksponentom ni težav:

$$(10^{exp})^{1/2} = 10^{exp/2}$$

Če je eksponent lih, pa ga zmanjšamo za 1. S tem smo seveda povečali mantiso, tako da je v mejah $1 \leqslant M < 100$, a po korenjenju bo mantisa M v pravih mejah. Ves postopek bi bil torej takšen:

1. izračunaj eksponent odgovora
2. pomnoži mantiso s 5, dobiš $5R_a$
3. z začetnim približkom $a = 0$ uporabi opisano metodo in poišči 12 cifer odgovora
4. zaokroži mantiso, dodaj eksponent
5. izpiši rezultat

Če ste postopek razumeli, ga sprogramirajte v nekem višjem programskem jeziku, recimo v basicu na vašem domačem računalniku.

Trigonometrične funkcije: sin(φ), cos(φ), tg(φ), ctg(φ)

Za izračun vseh štirih trigonometričnih funkcij bomo uporabili isti algoritem, kar vsekakor pomeni bistven prihranek ROM–a. V vsakem primeru najprej izračunamo tg(φ) ali pa ctg(φ), nato pa po potrebi iz njiju sin(φ) in cos(φ) s formulama:

$$\sin(\varphi) = \frac{\pm \operatorname{tg}(\varphi)}{1 + \operatorname{tg}^2(\varphi)} \tag{3}$$

$$\cos(\varphi) = \frac{\pm \operatorname{ctg}(\varphi)}{1 + \operatorname{ctg}^2(\varphi)} \tag{4}$$

V primerih, ko dobimo kot φ v drugih kotnih enotah kot radianih, ga najprej pretvorimo v radiane. Ker so trigonometrične funkcije periodične s periodo 2π, reduciramo poljuben kot na intervalu $[0, 2\pi)$. To lahko naredimo tako, da odštevamo 2π od kota, dokler ne pridemo v ustrezni interval. Vendar bi bil tak postopek za velike kote sila neučinkovit. Zato kot najprej zapišimo v eksponentnem zapisu in nato od njega odštevajmo velike večkratnike kota 2π, dokler ne dobimo negativnega kota. Nato ta večkratnik enkrat prištejemo in postopek ponovimo pri manjšem večkratniku kota 2π. Zapišimo ta postopek algoritmično:

kot φ zapiši v obliki $\varphi = a_1.a_2a_3 \ldots . 10^n$
za $k = n, n-1, \ldots, 0$ **ponovi**
$y \leftarrow 2.\pi.10^k$
ponavljaj $\varphi \leftarrow \varphi - y$ **dokler ni** $\varphi < 0$
$\varphi \leftarrow \varphi + y$

Na ta način velike kote zelo hitro spravimo v ustrezni integral. Sam premisli, kaj moramo opraviti v gornjem postopku, če imamo za podatek negativen kot. S tem algoritmom smo spravili kot na interval $[0, 2\pi)$ in od tu dalje nas bodo zanimali le še ustrezni koti. Glavna ideja, ki nas pripelje do končnega algoritma, je naslednja. Če poznamo poljubno točko na poltraku iz izhodišča, ki oklepa z osjo x kot φ, znamo $\mathrm{tg}(\varphi)$ in $\mathrm{ctg}(\varphi)$ izračunati s srednješolskima formulama:

$$\mathrm{tg}(\varphi) = \frac{y}{x} \qquad \text{in} \qquad \mathrm{ctg}(\varphi) = \frac{x}{y} \tag{5}$$

in odtod s formulama (1) in (2) tudi $\sin(\varphi)$ in $\cos(\varphi)$. Seveda pa še ne vemo, kako naj pridelamo ustrezno točko (x, y) na poltraku, ki oklepa z osjo x kot φ. Postopajmo takole: začetno točko, ki je dovolj blizu osi x, zavrtimo nekajkrat v pozitivni smeri za ustrezne kote φ, dokler ne dobimo zaželjene točke. Najprej zapišimo formuli (2), ki zavrtita točko (x_1, y_1) v ravnini za kot φ_2:

$$\begin{aligned} x_2 &= x_1 \,.\, \cos(\varphi_2) - y_1 \,.\, \sin(\varphi_2) \\ y_2 &= y_1 \,.\, \cos(\varphi_2) + x_1 \,.\, \sin(\varphi_2) \end{aligned} \tag{6}$$

Če obe enačbi (6) delimo s $\cos(\varphi_2)$, dobimo identiteto:

$$\begin{aligned} x_2/\cos(\varphi_2) &= x_1 - y_1 \,.\, \mathrm{tg}(\varphi_2) \\ y_2/\cos(\varphi_2) &= y_1 + x_1 \,.\, \mathrm{tg}(\varphi_2) \end{aligned} \tag{7}$$

Označimo prvo desno stran v (7) z x_2', drugo desno stran pa z y_2'. Če enačbi (7) zapišemo v teh novih oznakah, dobimo razmerje:

$$y_2'/x_2' = y_2/x_2 \tag{8}$$

Ker je $y_2/x_2 = \mathrm{tg}(\varphi_1 + \varphi_2)$, lahko tangens kota $\varphi_1 + \varphi_2$ dobimo tako, da izračunamo x_2' in y_2'. Torej lahko izračunamo tangens za φ_2 večjega kota, če le poznamo $\mathrm{tg}(\varphi_2)$, x_2' in y_2. To in pa seveda dejstvo, da postopek lahko večkrat ponovimo, uporabimo v našem algoritmu. Za izračun $\mathrm{tg}(\varphi_1 + \varphi_2)$ v (8) moramo izračunati x_2' in y_2', ki ju dobimo v formuli (7).

Ker smo pri izbiri kota φ_2 še svobodni, ga izberimo tako, da bo izračun v (7) čim preprostejši. Kot smo že omenili, uporabljajo HP kalkulatorji posebne procesorje, pri katerih predstavlja množenje s potencami 10 le premik decimalne pike. Zato zahtevajmo, da je $\mathrm{tg}(\varphi_2)$ oblike 10^k. Množenje v (7) namreč tedaj ni nič drugega kot premik decimalne pike za k mest. Kot φ torej zapišimo v obliki:

$$\varphi = a_0 . \mathrm{tg}^{-1}(1) + a_1 . \mathrm{tg}^{-1}(0.1) + a_2 . \mathrm{tg}^{-1}(0.01) + \ldots + r \qquad (9)$$

Pri tem smo s tg^{-1} zapisali inverzno funkcijo k funkciji tangens. Vse konstante $a_0, a_1, \ldots$ so naravna števila, manjša ali enaka 10, tako da za vsako od njih potrebujemo en sam štiribitni zapis. Ustrezne približne kote zapišimo v radianih in stopinjah.

$\mathrm{tg}^{-1}(1)$	=	0.785398163	$\mathrm{tg}^{-1}(1)$	=	45
$\mathrm{tg}^{-1}(0.1)$	=	0.099668652	$\mathrm{tg}^{-1}(0.1)$	=	5.710593137
$\mathrm{tg}^{-1}(0.01)$	=	0.009999667	$\mathrm{tg}^{-1}(0.01)$	=	0.572938698
$\mathrm{tg}^{-1}(0.001)$	=	0.001000000	$\mathrm{tg}^{-1}(0.001)$	=	0.057295604
. . . . .			. . . . .		

Koti v radianih vsebujejo v svojih cifrah več pravilnosti. Vsi ti koti so seveda stalno zapisani v ROM–u, večja pravilnost pa pomeni manjšo porabo prostora, kar daje radianom prednost pred stopinjami. Zato smo tudi takoj na začetku pretvorili kot v radiane. Razcep opravimo s preprostim algorimnom. Od kota odštevamo ustrezni kot, dokler kot ni negativen, potem pa mu še enkrat prištejemo isti kot. Ves postopek nato ponovimo na manjšem kotu. Zapišimo ta postopek zopet v obliki algoritma:

```
za i = 0, 1, 2, ... ponovi
    a_i ← 0
odštevaj:
    ponavljaj
        φ ← φ − tg^-1(10^-i)
        a_i ← a_i + 1
        če je φ < 0 potem
            φ ← φ + tg^-1(10^-i)
            a_i ← a_i − 1
            zapusti odštevaj
    do tod
do tod
```

Kako majhen naj bo ostanek r pri razcepu (9), je odvisno od natančnosti aritmetike. V večini kalkulatorjev HP se napravi razcep do kota $\mathrm{tg}^{-1}(0.0001)$. Tedaj je namreč ostanek že tako majhen, da ne vpliva niti na zadnjo decimalko v končnem izračunu. Da pa lahko začnemo celoten postopek, moramo določiti še začetno točko. Ker je ostanek r zelo majhen, je, če vzamemo $x = 1$, $\mathrm{tg}(r)$ približno enak r in za začetno točko vzamemo točko $(1, r)$. Napišimo še algoritem, s katerim izračunamo končno točko na poltraku, ki oklepa z osjo x kot φ.

$x \leftarrow 1$
$y \leftarrow r$
za $k = 0, 1, \ldots, m$ **ponovi**
  za $i = 1, 2, \ldots, a_k$ **ponovi**
    $x' \leftarrow x - y \,.\, 10^{-k}$
    $y' \leftarrow y + x \,.\, 10^{-k}$
    $x \leftarrow x'$
    $y \leftarrow y'$
  do tod
do tod

Povzemimo celotni postopek:

1. pretvori kot v ekvivalentnega na $[0, 2\pi)$
2. razbij kot v linearno kombinacijo (9)
3. z rotacijami izračunaj točko (x, y)
4. iz (x, y) izračunaj, kar potrebuješ, in rezultat izpiši

Tudi za trigonometrične funkcije je za razumevanje postopka priporočljivo, če postopek sprogramiramo v kakem višjem programskem jeziku.

LITERATURA

[1] M.Abramowitz, I.A. Stegun, Handbook of Mathematical Functions, Dover Publications, New York, 1964

[2] W.E. Egbert, Personal Calculator Algorithms I: Square Roots, HP Journal, May 1977

[3] W.E. Egbert, Personal Calculator Algorithms II: Trigonometric Functions, HP Journal, June 1977

[4] W.E. Egbert, Personal Calculator Algorithms III: Inverse Trigonometric Functions, HP Journal, November 1977

[5] W.E. Egbert, Personal Calculator Algorithms IV: Logarithmic Functions, HP Journal, April 1978