

IZDAJA DRUŠTVO MATEMATIKOV, FIZIKOV IN ASTRONOMOV SR SLOVENIJE

OBZORNIK ZA MATEMATIKO IN FIZIKO

1972

Letnik 19

3

XIX. LETNIK — ŠTEVILKA 3

OBZORNIK ZA MATEMATIKO IN FIZIKO

LJUBLJANA, AVGUST 1972

Uredniški odbor: Franc Avsec, gimnazija Kranj; Robert Blinc, FNT univerze v Ljubljani; Franc Kvaternik, tehnični in odgovorni urednik, gimnazija Poljane, Ljubljana; Jože Lep, VTŠ, Maribor; Anton Moljk, FNT; Jože Pahor, inštitut »J. Štefan«; Mitja Rosina, FNT; Tomaž Skulj, gimnazija Moste, Ljubljana; Janez Strnad, FNT; Anton Suhadolc, FNT; Ivan Vidav, FNT.

Izdaja: Društvo matematikov, fizikov in astronomov SRS 4-krat letno.

Izdajo revije sofinancira Kulturna skupnost Slovenije.

Naročnina: za člane 20 din, za nečlane 30 din, za dijake in študente 10 din, za ustanove in podjetja 40 din, za inozemstvo 51 din (3 \$), posamezna številka 8 din.

Dopise pošiljajte in list naročajte na naslov: Komisija za tisk DMFA SRS, Ljubljana, Jadranska c. 19, pp. 227, telefon: 61-564/22, tek. račun št. 501-8-240/1.

Člani društva naj nakazujejo na tekoči račun Komisije za tisk 20 din naročnine in 10 din članarine, skupaj 30 din.

Tiska tiskarna Ljudske pravice v Ljubljani, Kopitarjeva ul. 2.

VSEBINA

Članki	Stran
Numerična matematika (Z. Bohte)	97
Analiza napak (Z. Bohte)	103
Pregled elementarne matematike s stališča numeričnega računanja (Z. Bohte)	108
Elementi programskega jezika <i>ALGOL</i> 60 (E. Zakrajšek)	128
Numerologija? (A. Suhadolc)	141
Analiza algoritmov (A. Kmet)	143
Stabilnost numeričnih procesov (J. Kozak)	149
Literatura za dopolnilno branje	152
Domače vesti	
IV. seminar iz matematike — »Novo računstvo« (C. Velkovrh)	153
Naši seminarji (D. Modic)	154
Delo podružnic Društva matematikov, fizikov in astronomov SRS v letu 1971 (I. Pavliha) ..	157
Poročilo o delu oddelka za matematiko inštituta za matematiko fiziko in mehaniko univerze v Ljubljani v šolskem letu 1971/72 (C. Velkovrh)	158
Iz naših aktivov	
Aktiv matematikov in fizikov dne 17. maja 1972 v Novem mestu (D. Modic)	159

CONTENTS

Articles	Page
Elementary exposition on the subject of numerical mathematics (Z. Bohte)	97
Error analysis (Z. Bohte)	103
Survey of elementary mathematics from the computational standpoint (Z. Bohte)	108
Elements of the programming language (E. Zakrajšek)	128
Numerology? (A. Suhadolc)	141
Analysis of algorithms (A. Kmet)	143
The stability of numerical processes (J. Kozak)	149
Home News	153
From our Actives	159

NUMERIČNA MATEMATIKA

ZVONIMIR BOHTE

MOS 65-01

Elementaren prikaz predmeta numerične matematike.

NUMERICAL MATHEMATICS

Elementary exposition on the subject of numerical mathematics.

Numerična matematika je veja matematike, ki razvija metode za numerično reševanje matematičnih problemov. Pri numeričnem reševanju problemov izhajamo od danih številskih podatkov in iščemo številske rezultate. Če iščemo rešitev kake naloge z analitičnimi metodami, potem navadno poiščemo rešitev nekega razreda sorodnih problemov v obliki formule ali analitičnega izraza, v katerega vstavimo konkretne vrednosti parametrov in izračunamo konkretne vrednosti rezultatov. Analitične metode imajo mnogo odličnih lastnosti, ki jih ni potrebno naštevati. Toda le malo je takih praktičnih problemov, ki se dajo rešiti analitično. V času pred dobo elektronskih računalnikov so matematiki ponavadi iskali rešitve preveč kompliciranih problemov tako, da so zanemarili vpliv nekaterih količin in z analitičnimi metodami poiskali rešitev poenostavljenega problema, ki je bila tako le približna rešitev dejanskega problema. Pri numeričnem reševanju pa se trudimo približno rešiti dejanski problem, toda le pri danih vrednostih podatkov.

Kdaj torej uporabljamo numerične metode? Predvsem takrat, kadar problema ne znamo rešiti analitično ali kako drugače rezultat zadovoljivo aproksimirati. Včasih pa tudi takrat, ko je analitična rešitev sicer znana, a je numerično reševanje udobnejše. Zgodi se namreč lahko, da je enostavneje poiskati rezultat z numerično metodo, kot pa izračunati vrednost analitičnega izraza, ki predstavlja analitično rešitev, pri konkretnih vrednostih parametrov.

Oglejmo si te trditve na dveh preprostih primerih!

Vzemimo najprej problem reševanja algebraične enačbe

$$a_0x^n + a_1x^{n-1} + \dots + a_n = 0, \quad a_0 \neq 0$$

Če je $n = 2$, je enačba kvadratna in poznamo formuli za oba korena te enačbe. Kakršnikoli so koeficienti kvadratne enačbe, moremo iz teh formul izračunati oba korena poljubno natančno, če le izračunamo dovolj natančno kvadratni koren iz diskriminante. Nobenega razloga ni, da bi se za reševanje kvadratnih enačb posluževali drugih metod.

Pri $n = 3$ ali $n = 4$ je stanje nekoliko drugačno. Znane so formule za analitično rešitev (Cardanove in Ferrarijeve formule iz 16. stoletja), toda te so za izračun pri danih koeficientih zelo neudobne. Pri $n = 3$ je treba enačbo najprej predelati v obliko

$$x^3 + px + q = 0$$

izračunati števili

$$y = -\frac{1}{2}q + \sqrt{D}, \quad z = -\frac{1}{2}q - \sqrt{D}$$

kjer je

$$D = \left(\frac{1}{2}q\right)^2 + \left(\frac{1}{3}p\right)^3$$

nato pa sestaviti tri korene takole:

$$x_1 = \sqrt[3]{y} + \sqrt[3]{z}$$

$$x_2 = \zeta \sqrt[3]{y} + \zeta^2 \sqrt[3]{z}$$

$$x_3 = \zeta^2 \sqrt[3]{y} + \zeta \sqrt[3]{z}$$

kjer je

$$\zeta = \frac{1}{2} (-1 + i\sqrt{3})$$

kompleksni tretji koren enote. Razen za posebno lepe primere koeficientov so te formule neudobne, saj je v primeru, ko je $D < 0$, potrebno izračunati kubični koren iz kompleksnega števila.

Pri enačbi

$$x^3 - 5x + 1 = 0$$

dobimo

$$y = -\frac{1}{2} + i\sqrt{\frac{473}{108}}, \quad z = -\frac{1}{2} - i\sqrt{\frac{473}{108}}$$

in nadaljnji račun je očitno zamuden. Pri tej enačbi se da dobiti en koren zelo hitro z naslednjo iteracijo:

$$x_0 = 0, \quad x_{r+1} = (1 + x_r^3)/5, \quad r = 0, 1, 2, \dots$$

Prvi trije približki so:

$$x_1 = 0.2, \quad x_2 = 0.2016, \quad x_3 = 0.201639$$

Zadnji je zaokrožen na 6 decimalk. Na 6 decimalk natančen koren pa je 0.201640.

Takih numeričnih postopkov je znanih cela vrsta. Zanimivo je, da je že leta 1225 Leonardo iz Pise poiskal rešitev enačbe

$$x^3 + 2x^2 + 10x - 20 = 0$$

na 10 decimalnih mest

$$x = 1.368808107$$

Ni znano, po kateri poti je prišel do tega rezultata, a zelo verjetno je bila to kaka numerična metoda.

Podobno velja za enačbe četrte stopnje. Pri $n > 4$ pa v splošnem ni mogoče dobiti analitične rešitve in nam kot glavno sredstvo za reševanje preostanejo le numerične metode. Pri teh pa seveda ni bistveno, ali je enačba pete ali šeste stopnje, ali so koeficienti cela števila ali ne, pač pa nastopajo težave drugačne narave.

Kot naslednji primer vzemimo rešitev konkretne diferencialne enačbe

$$y' = 1 - 2xy, \quad y(0) = 1$$

z začetnim pogojem! Enačba je linearna in splošna rešitev je

$$y = e^{-x^2} \int_0^x e^{t^2} dt + Ce^{-x^2}$$

Če upoštevamo začetni pogoj, dobimo $C = 1$. Integral ni elementarna funkcija. Vzemimo, da želimo tabelirati vrednosti rešitve na intervalu $[0, 1]$! Izračunati je treba več vrednosti integrala, zato je pametno razviti rešitev v vrsto:

$$y = \left(1 - x^2 + \frac{x^4}{2!} - \frac{x^6}{3!} + \dots\right) \left(1 + x + \frac{x^3}{3} + \frac{x^5}{5 \cdot 2!} + \dots\right) =$$

$$= 1 + x - x^2 - \frac{2x^3}{3} + \frac{x^4}{2} + \frac{4x^5}{15} - \frac{x^6}{6} - \dots$$

Vrsto za rešitev moremo dobiti tudi direktno iz diferencialne enačbe. Z odvajanjem diferencialne enačbe dobimo rekurzivno formulo za odvode rešitve pri $x = 0$:

$$y(0) = y'(0) = 1, y^{(n+1)}(0) = -2ny^{(n-1)}(0), n = 1, 2, \dots$$

Dobljena vrsta ni posebno dobro konvergentna. Pri absolutno majhnih argumentih je treba za določeno natančnost upoštevati manj členov kot pri absolutno večjih argumentih. Za izračun vrednosti $y(1)$ na 4 decimalke natančno je treba upoštevati 18 členov vrste. Pri drugem argumentu je treba začeti znova.

Sedaj pa primerjajmo to analitično pot z neko numerično metodo! Osnovna ideja pri tej metodi je ta, da odvod funkcije v neki točki aproksimiramo z odvodom parabole, ki poteka skozi tri sosednje enako oddaljene točke:

$$f'(x) \doteq \frac{f(x+h) - f(x-h)}{2h}$$

Geometrično je nazorno, da je za gladke krivulje diferenčni kvocient na desni dobra aproksimacija za odvod. To aproksimacijo uporabimo takole: Interval $[0, 1]$ razdelimo na n enakih delov in naj bo $h = 1/n$. Delilne točke označimo z $x_i = ih, i = 0, 1, \dots, n$. Namesto $f(x_i)$ pa na kratko pišimo f_i pri vsaki funkciji $f(x)$! Diferencialno enačbo v točki x_i

$$y'(x_i) = 1 - 2x_i y(x_i)$$

nadomestimo z enačbo

$$(y_{i+1} - y_{i-1})/(2h) = y'_i = 1 - 2x_i y_i$$

ali

$$y_{i+1} = y_{i-1} + 2hy'_i = y_{i-1} + 2h(1 - 2x_i y_i)$$

Začetno vrednost $y_0 = 1$ poznamo, potrebujemo še vrednost $y_1 = y(h)$. Ker je h navadno majhno število, dobimo to vrednost zelo hitro iz prejšnje vrste. Npr. pri $h = 0.1$ zadošča upoštevati 5 členov vrste za natančnost 4 decimalke. Računanje zaporednih približnih vrednosti rešitve poteka sedaj zelo preprosto. Rešitev sproti tabeliramo, na vsakem koraku je izredno malo računskih operacij. Natančnost izračunanih vrednosti je seveda odvisna od velikosti koraka h . Na primer, pri $h = 0.05$ ($n = 20$), dobimo $y(1)$ na 3 decimalke natančno. Izračunane vrednosti zapišimo v naslednji tabeli:

i	x_i	y_i	i	x_i	y_i
0	0.00	1.0000	10	0.50	1.2032
1	0.05	1.0474	11	0.55	1.1902
2	0.10	1.0895	12	0.60	1.1723
3	0.15	1.1256	13	0.65	1.1496
4	0.20	1.1557	14	0.70	1.1229
5	0.25	1.1793	15	0.75	1.0924
6	0.30	1.1967	16	0.80	1.0591
7	0.35	1.2075	17	0.85	1.0230
8	0.40	1.2121	18	0.90	0.9852
9	0.45	1.2105	19	0.95	0.9457
10	0.50	1.2032	20	1.00	0.9056

Vrednosti so bile izračunane na 4 decimalke z rezanjem (brez zaokrožanja).

Ta primer ne kaže izrazite prednosti numerične metode pred analitično. Opaziti pa je, da je računski postopek enostaven. Na vsakem koraku je v bistvu treba izračunati le vrednost odvoda.

Mnoge numerične metode so že dolgo znane. Še danes stalno uporabljamo nekatere metode, ki so jih odkrili Newton, Gauss, Jacobi, Euler, Lagrange, Laguerre in drugi znameniti matematiki. Večina teh starih metod je bila prirejena za računanje »peš«, saj računskih pripomočkov, razen morda logaritemskih tablic, ni bilo. Težnja je bila, da bi čimveč množenj in deljenj nadomestili s seštevanji in odštevanji. Počasen način računanja je zelo omejeval obsežnost računskih problemov.

Osnova mnogih metod so bile včasih interpolacija in končne difference. Oglejmo si primer računanja vrednosti polinoma! Danes vedno računamo vrednost polinoma po Hornerjevi shemi, pri kateri je potrebno n množenj in n seštevanj za eno vrednost polinoma stopnje n . Včasih pa so za tabeliranje polinoma v ekvidistantnih točkah uporabljali postopek, po katerem najprej izračunamo $n + 1$ vrednost polinoma, tvorimo diferenčno tabelo, nato pa računamo nadaljnje vrednosti s samimi seštevanji. Za eno novo vrednost je potrebno samo n seštevanj.

Kot zgled vzemimo kar polinom tretje stopnje:

$$f(x) = a_0x^3 + a_1x^2 + a_2x + a_3$$

Po Hornerjevi metodi izračunamo vrednost tega polinoma pri poljubnem x takole:

$$f(x) = ((a_0x + a_1)x + a_2)x + a_3$$

Po drugi metodi pa izračunamo najprej $f_i = f(x_i)$, $i = 0, 1, \dots, n$, kjer je $x_i = x_0 + ih$, kakor vemo in znamo, nato pa tvorimo naslednjo diferenčno tabelo:

x_0	f_0			
		$\Delta f_0 = f_1 - f_0$		
x_1	f_1		$\Delta^2 f_0 = \Delta f_1 - \Delta f_0$	
		$\Delta f_1 = f_2 - f_1$		$\Delta^3 f_0 = \Delta^2 f_1 - \Delta^2 f_0$
x_2	f_2		$\Delta^2 f_1 = \Delta f_2 - \Delta f_1$	
		$\Delta f_2 = f_3 - f_2$		
x_3	f_3			

Znano je, da so n -te difference polinoma n -te stopnje konstantne (podobno kot n -ti odvod). Zato v naši tabeli izpolnimo zadnji stolpec s samimi $\Delta^3 f_0$, stolpce na levi pa postopoma dopolnjujemo s samimi seštevanji:

$$\begin{aligned} \Delta^2 f_2 &= \Delta^2 f_1 + \Delta^3 f_0, & \Delta f_3 &= \Delta f_2 + \Delta^2 f_2, & f_4 &= f_3 + \Delta f_3 \\ \Delta^2 f_3 &= \Delta^2 f_2 + \Delta^3 f_0, & \Delta f_4 &= \Delta f_3 + \Delta^2 f_3, & f_5 &= f_4 + \Delta f_4 \\ \text{itd.} \end{aligned}$$

Pri računanju »peš« in pri velikem številu ekvidistantnih argumentov so prednosti druge metode očitne. Pri sodobnih računalnikih pa čas, potreben za tvorbo vsote dveh števil, ni bistveno krajši od časa, potrebnega za produkt dveh števil, torej druga metoda skoraj ne pride več v poštev.

Pred dobrim desetletjem, ko so prešli hitri računalniki v vsakdanjo rabo, je numerična matematika začela doživljati močan pretres. Mnoge metode, ki so bile prej v čislih, so se izkazale za neuporabne pri obsežnejših problemih; mnoge metode, ki so bile namenjene računanju »peš«, so se pokazale kot neudobne za programiranje. Nasprotno pa so nekatere

metode, ki so jih prej zavrgli zaradi neudobnosti, doživele ponovno odkritje in pogosto uporabo. Večina metod, ki jih danes uporabljamo, pa je bila odkrita šele nedavno.

Ker se računalniki hitro razvijajo, se hitro spreminjajo tudi kriteriji za primerjavo učinkovitosti posameznih metod za reševanje istih problemov. Z vse večjim spominom in z vse večjo hitrostjo računanja omogočajo današnji računalniki numerično reševanje vse obsežnejših problemov. Z naraščajočim obsegom računanja pa nastajajo novi problemi. V zadnjem času je pri raziskovanju velik poudarek na študiju stabilnosti računskih postopkov in analizi zaokrožitvenih napak. Šele podroben študij teh vprašanj pokaže pravo vrednost posameznih metod. Ilustrirajmo te trditve na nekaterih primerih!

Vzemimo problem lastnih vrednosti matrik

$$Ax = \lambda x, x \neq 0$$

kjer je A dana kvadratna matrika reda n . Vemo, da so lastne vrednosti λ ničle karakterističnega polinoma

$$f(\lambda) = \det(A - \lambda I)$$

Še kakih 15 let nazaj so bile skoraj vse numerične metode osnovane na določitvi koeficientov karakterističnega polinoma in na reševanju ustrezne algebraične enačbe. Pred uporabo sodobnih računalnikov so na ta način reševali problem lastnih vrednosti za matrike največ reda 6. Pri tako majhnih redih navadno ni prišlo do kakih posebnih težav. Z računalnikom so po isti poti začeli reševati probleme lastnih vrednosti do $n = 50$ in takoj se je pokazalo, da so bile lastne vrednosti večinoma čisto napačno izračunane. Tudi računanje na 20 decimalk in več ni dosti pomagalo. Ko je Wilkinson l. 1959 pokazal, da so ničle polinomov lahko zelo občutljive funkcije koeficientov, je postalo jasno, da se je treba eksplcitnemu računanju koeficientov karakterističnega polinoma izogniti. Kar naenkrat je postala za simetrične matrike najudobnejša Jacobijeva metoda, ki je bila že skoraj pozabljena. Na osnovi idej pri Jacobijevi metodi se je v zadnjem času razvila vrsta odličnih metod, pri katerih so odpravljene vse odpravljive težave.

Omenimo še problem reševanja sistemov linearnih enačb

$$Ax = b, \quad \det A \neq 0$$

Na prvi pogled ne opazimo nobenih načelnih težav. Že dolgo je znano, da je za reševanje sistemov linearnih enačb najbolj ekonomična Gaussova eliminacijska metoda. Za rešitev sistema reda n je potrebno izvršiti približno $n^3/3$ množenj in prav toliko seštevanj. Šele z uporabo računalnikov je postalo možno reševati sisteme reda 100 in več. In šele pri večjih sistemih se je pokazalo, da ni vseeno, v katerem vrstnem redu eliminiramo neznanke. L. 1961 je Wilkinson izdelal podrobno analizo zaokrožitvenih napak pri Gaussovi metodi, ki je pokazala, kako je treba to metodo uporabiti, da dobimo čimbolj natančne rezultate. Pravilno prirejena Gaussova metoda ima še danes prednost pred vsemi drugimi metodami. To seveda velja za splošne sisteme, ki nimajo kake posebne oblike.

Kot zadnji primer si oglejmo problem aproksimacije funkcije, ki je podana v obliki tabele: (x_i, y_i) , $i = 1, 2, \dots, N$. Vrednosti y_i naj bodo rezultati opazovanj, torej utegnejo vsebovati razne napake. Želimo dobiti preprost analitičen izraz $g(x)$, ki bi v celoti dobro aproksimiral našo funkcijo. V praksi zelo pogosto uporabljamo princip najmanjših kvadratov, ki pravi, da je med funkcijami iz nekega razreda najugodnejša tista, pri kateri doseže izraz

$$\sum_{i=1}^N [g(x_i) - y_i]^2$$

najmanjšo vrednost. Če je funkcija $g(x)$ linearno odvisna od nekih parametrov, dobimo za iskane vrednosti teh parametrov sistem linearnih enačb.

Pogosto iščemo $g(x)$ kot polinom določene stopnje:

$$g(x) = a_0 + a_1x + \dots + a_mx^m$$

Ekstrem izraza

$$F(a_0, a_1, \dots, a_m) = \sum_{i=1}^N [a_0 + a_1x_i + \dots + a_mx_i^m - y_i]^2$$

dobimo pri tistih koeficientih a_k , ki zadoščajo sistemu linearnih enačb

$$\frac{\partial F}{\partial a_k} = 0, \quad k = 0, 1, \dots, m$$

ali

$$s_0a_0 + s_1a_1 + \dots + s_ma_m = t_0$$

$$s_1a_0 + s_2a_1 + \dots + s_{m+1}a_m = t_1$$

$$- - - - -$$

$$s_ma_0 + s_{m+1}a_1 + \dots + s_{2m}a_m = t_m$$

kjer je

$$s_k = \sum_{i=1}^N x_i^k, \quad t_k = \sum_{i=1}^N y_i x_i^k$$

Navadno se izkaže, da ta sistem slabo določa rešitev, to se pravi, da majhne napake v koeficientih povzročijo velike napake v rešitvi. Pri stopnjah $m > 3$ je pametno izraziti polinom $g(x)$ kako drugače.

Za zelo ugodno pot se izkaže iskati polinom $g(x)$ v obliki

$$g(x) = b_0p_0(x) + b_1p_1(x) + \dots + b_mp_m(x)$$

kjer so $p_k(x)$ ortogonalni polinomi na našem sistemu točk:

$$\sum_{i=1}^N p_r(x_i)p_s(x_i) = 0, \quad r \neq s$$

V tem primeru dobimo za koeficiente preproste izraze

$$b_k = \left[\sum_{i=1}^N y_i p_k(x_i) \right] / \left[\sum_{i=1}^N p_k^2(x_i) \right]$$

in ni nobenih problemov v zvezi z natančnostjo izračunanih koeficientov. Kot je pokazal Forsythe l. 1957, je take polinome zelo enostavno konstruirati s pomočjo tročlenske rekurzivne relacije.

Našteli smo samo nekaj numeričnih metod in njihov razvoj. Več o teh in drugih metodah si bomo ogledali v nadaljnjih prispevkih.

Za zaključek naštejmo, katera poglavja navadno vsebujejo učbeniki za numerično matematiko: Sistemi linearnih enačb. Nelinearne enačbe. Problem lastnih vrednosti matrik. Interpolacija. Odvajanje. Integriranje. Navadne diferencialne enačbe. Parcialne diferencialne enačbe. Aproksimacija.

ANALIZA NAPAK*

ZVONIMIR BOHTE

MOS 65-01

Elementaren uvod v analizo napak.

ERROR ANALYSIS

Elementary introduction to error analysis.

Pri numeričnem reševanju problemov moremo le redkokdaj izračunati eksaktni rezultat. Navadno izračunani rezultati vsebujejo napake, ki nastanejo zaradi raznih vzrokov. Če želimo numerično matematiko obravnavati rigorozno kot eksaktno znanost, moramo vse napake, ki nastanejo pri numeričnem računanju, analizirati in oceniti. Numerični rezultat skupaj z realistično ocenjeno napako, ki ne presega dopustne napake, more v večini praktičnih primerov popolnoma nadomestiti eksaktni rezultat.

Ogledali si bomo glavne vire napak in na nekaterih zgledih ilustrirali osnovne pojme.

1. Neodstranljiva napaka

Če so podatki numeričnega problema rezultati meritev ali prejšnjih računov, nosijo s seboj napake, ki onemogočajo izračun eksaktnega rezultata. Sem spadajo tudi iracionalni podatki, ki jih je treba v računu nadomestiti z racionalnimi približki (npr. število π), ali pa podatki, ki jih čitamo v računalnik in jih je pri tem treba aproksimirati (npr. 0.1 v binarnem računalniku).

Zaradi napak v podatkih nastane torej v rezultatu napaka, ki se ji ne moremo izogniti in jo zato imenujemo neodstranljiva napaka.

Kako napake v podatkih vplivajo na natančnost rezultata, je seveda odvisno od problema. Če je ta napaka velika v primerjavi z velikostjo napak v podatkih, pravimo, da je problem slabo pogojen. Takim problemom se po možnosti izogibamo. Če pa majhne napake v podatkih povzročijo le majhno napako v rezultatu, pravimo, da je problem dobro pogojen. Neodstranljive napake študira teorija perturbacij.

Oglejmo si nekaj primerov!

a) Vrednost polinoma.

Naj bo

$$f(x) = a_0x^n + a_1x^{n-1} + \dots + a_n$$

polinom z realnimi koeficienti. Ocenimo, koliko se utegne spremeniti vrednost polinoma pri danem realnem argumentu x , če spremenimo koeficiente polinoma za δa_i , $i = 0, 1, \dots, n$! Ta sprememba je očitno enaka

$$\Delta_a f(x) = |\delta a_0x^n + \delta a_1x^{n-1} + \dots + \delta a_n|$$

* Bralcu priporočamo, da si ogleda članek Z. Bohteta: Napake pri numeričnem računanju, OMF 18 (1971), 10—14.

Če velja

$$|\delta a_i| \leq \delta a, \quad i = 0, 1, \dots, n$$

potem imamo oceno

$$\Delta_a f(x) \leq \delta a (|x|^n + |x|^{n-1} + \dots + 1) = \delta a \frac{|x|^{n+1} - 1}{|x| - 1}$$

ki pravi, da se vrednost polinoma ne spremeni dosti, če le ni $|x|^n$ veliko število. Pri velikem argumentu x je seveda absolutna napaka v vrednosti polinoma lahko velika. Toda če si ogledamo relativno napako in upoštevamo, da je vrednost polinoma tedaj približno $a_0 x^n$, dobimo, da je

$$\frac{\Delta_a f(x)}{|f(x)|} \approx \frac{\delta a_0}{|a_0|}$$

Od tod moremo sklepati, da je računanje vrednosti polinoma dobro pogojen problem glede na spremembe koeficientov.

Do drugačne ugotovitve pridemo, če študiramo spremembo vrednosti polinoma glede na spremembo argumenta x . Če spremenimo x za δx , dobimo

$$\Delta_x f(x) = f(x + \delta x) - f(x) = f'(x) \delta x + O(|\delta x|^2)$$

Simbol $O(|\delta x|^2)$ pomeni, da imajo vsi drugi členi δx najmanj v drugi potenci. To se pravi, da je pri majhnem δx glavni del izraza na desni $f'(x) \delta x$. Od tod sledi, da je pri majhnih δx sprememba v vrednosti polinoma majhna, če le ni odvod $f'(x)$ absolutno veliko število. Ta rezultat je geometrično popolnoma nazoren. Računanje vrednosti strme funkcije je slabo, položne pa dobro pogojen problem.

b) Ničle polinoma.

Drugačni zaključki veljajo, če iščemo ničle polinoma $f(x)$ ali korene enačbe

$$f(x) = 0$$

Intuitivno bi sklepali, da bo rezultat ravno obraten kot pri računanju vrednosti polinoma: če poteka polinom skozi ničlo strmo, je ničla dobro pogojena, če pa položno, je lahko slabo pogojena. Drugi del trditve je geometrično nazoren: Če poteka polinom skozi ničlo zelo položno, potem utegne majhna sprememba polinoma povzročiti veliko spremembo v ničli. Prvi del trditve pa ne velja vedno, kot bomo videli na primeru.

Oglejmo si nekoliko spremenjeno enačbo

$$F(x) = f(x) + \varepsilon g(x) = 0$$

kjer je $g(x)$ tudi polinom, ki ni višje stopnje kot $f(x)$. Zanima nas, v kakšni zvezi so ničle polinoma $F(x)$ z ničlami polinoma $f(x)$ pri majhnih ε . Velja naslednji izrek:

Če je x_r enostavna ničla polinoma $f(x)$, potem ima pri majhnem ε polinom $F(x)$ eno ničlo $x_r(\varepsilon)$, ki se izraža kot potenčna vrsta po potencah ε -a in velja

$$x_r(\varepsilon) = x_r - \varepsilon g(x_r)/f'(x_r) + O(\varepsilon^2)$$

Če pa je x_r m -kratna ničla polinoma $f(x)$, potem ima pri majhnem ε polinom $F(x)$ m ničel, ki se izražajo kot potenčne vrste po potencah $\varepsilon^{1/m}$ in velja

$$x_r(\varepsilon) = x_r + \varepsilon^{1/m} (-m! g(x_r)/f^{(m)}(x_r))^{1/m} + O(\varepsilon^{2/m})$$

kjer dobimo za vsako vrednost m -tega korena eno ničlo.

Ta izrek nam pove, da so večkratne ničle nujno slabo pogojene, saj je glavni del spremembe v taki ničli sorazmeren $\varepsilon^{1/m}$. Enostavne ničle so tudi slabo pogojene, če je $|g(x_r)/f'(x_r)|$ veliko število. Intuicija nam je povedala, da se to lahko zgodi, če je $|f'(x_r)|$ majhno število, ne pa, da tudi v primeru, ko je $|f'(x_r)|$ sicer znatno število, pa $|g(x_r)|$ znatno prekaša to vrednost. L. 1959 je Wilkinson navedel naslednji primer:

$$\begin{aligned} f(x) &= (x-1)(x-2)\dots(x-20) \\ g(x) &= x^{19}, \quad \varepsilon = -2^{-23} \doteq -10^{-7} \end{aligned}$$

Ničle polinoma $F(x) = f(x) + \varepsilon g(x)$ so na 1 mesto natančno: 1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 8.9, 20.8, $10.1 \pm 0.6i$, $11.8 \pm 1.7i$, $14.0 \pm 2.5i$, $16.7 \pm 2.8i$, $19.5 \pm 1.9i$. Očitno so torej večje ničle polinoma $f(x)$ slabo pogojene. Vzrok je v velikem $|g(x_r)|$, ne pa v majhnem $|f'(x_r)|$. V tem primeru je $|g(16)| = 16^{19} \doteq 10^{23}$, $|f'(16)| = 15! \doteq 10^{13}$, $|g(16)/f'(16)| \doteq 10^{10}$ in $|g(1)| = 1$, $|f'(1)| = 19! \doteq 10^{17}$, $|g(1)/f'(1)| \doteq 10^{-17}$.

c) Odvajanje in integriranje.

Oglejmo si še problema odvajanja in integriranja dane funkcije! Izkaže se, da je odvajanje lahko slabo pogojen problem, medtem ko je integriranje vedno dobro pogojen problem.

Vzemimo funkciji $f(x)$ in

$$F(x) = f(x) + \varepsilon g(x)$$

kjer sta $f(x)$ in $g(x)$ zvezno odvedljivi funkciji in naj velja na nekem intervalu $[a, b]$, da je $|g(x)| \leq 1$. Odvoda funkcij $f(x)$ in $F(x)$ se razlikujeta za $\varepsilon g'(x)$. Samo pri pogoju $|g(x)| \leq 1$ odvoda $g'(x)$ ni mogoče oceniti in neodstranljiva napaka je lahko velika. Vzemimo npr. $g(x) = \sin nx$. Tedaj je $g'(x) = n \cos nx$ in odvod je lahko pri velikem n zelo velik.

Drugače je pri integriranju. Brez težav dobimo zvezo

$$\left| \int_a^b F(x) dx - \int_a^b f(x) dx \right| = \varepsilon \left| \int_a^b g(x) dx \right| \leq \varepsilon(b-a)$$

Če ni integracijski interval prevelik, potem se pri majhni spremembi funkcije integral ne more dosti spremeniti.

Obe ugotovitvi sta geometrično popolnoma nazorni.

2. Napake metode.

Le redki so problemi, pri katerih moremo izračunati eksaktni rezultat s končnim številom aritmetičnih operacij. Ponavadi numerična metoda nadomesti kak neskončni proces (limito zaporedja, odvod, integral, ...) s končnim (pozni člen, razlika, vsota, ...). Pri tem nastane napaka metode, ki je seveda odvisna od konkretne numerične metode.

Oglejmo si nekaj primerov ocen za napako metode!

a) Računanje vrednosti funkcije iz neskončne vrste:

Približek za vrednost funkcije je končni odsek vrste. Napako metode ocenimo, če znamo oceniti ostanek vrste. Npr.

$$\left| \sin x - \left(x - \frac{x^3}{3!} + \frac{x^5}{5!} \right) \right| < \frac{|x|^7}{7!}, \quad |x| \leq \frac{\pi}{4}$$

Za vsak x , ki je med $-\pi/4$ in $\pi/4$, je izraz $x - x^3/3! + x^5/5!$ približek za $\sin x$ z napako, ki ne presega $5 \cdot 10^{-5}$.

b) Računanje korenov enačb z iteracijo

Če iščemo koren enačbe

$$x = g(x)$$

kjer je $|g'(x)| \leq m < 1$ za vse x v primerni okolici korena a , je ugodna metoda navadna iteracija:

$$x_{r+1} = g(x_r), \quad r = 0, 1, 2, \dots$$

kjer mora biti x_0 dovolj dober približek za koren. Tedaj je limita zaporedja približkov enaka korenu a zgornje enačbe:

$$a = \lim_{r \rightarrow \infty} x_r$$

Pri praktičnem računanju se seveda ustavimo po končnem številu korakov. Brez težav se da izpeljati ocena za napako približka x_r :

$$|x_r - a| \leq \frac{m}{1 - m} |x_r - x_{r-1}|$$

Če poznamo število m , moremo na vsakem koraku oceniti napako približka s pomočjo razlike zadnjih dveh približkov. S to oceno moremo tudi ugotoviti, kdaj prenehati z iteracijo. Če je m blizu 1, zaporedje počasi konvergira h korenu in čeprav se dva zaporedna približka že dobro ujemata, je napaka lahko še vedno znatna.

c) Interpolacija s polinomom.

Naj bo $f(x)$ zvezna in $(n + 1)$ -krat zvezno odvedljiva funkcija in naj bodo znane vrednosti $f(x_i)$ za $i = 0, 1, \dots, n$. Interpolacijski polinom $I_n(x)$ te funkcije je polinom stopnje n , ki se v točkah x_i , $i = 0, 1, \dots, n$, ujema z našo funkcijo:

$$I_n(x_i) = f(x_i), \quad i = 0, 1, \dots, n$$

Če je x neko število, ki je različno od vseh danih x_i , moremo neznano vrednost $f(x)$ nadomestiti z vrednostjo $I_n(x)$. Pri tem storimo napako, ki se da izraziti takole

$$f(x) - I_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} (x - x_0)(x - x_1) \dots (x - x_n)$$

kjer je ξ neko število, ki je na intervalu

$$\min(x, x_0, \dots, x_n) < \xi < \max(x, x_0, \dots, x_n)$$

To oceno moremo praktično izkoristiti le, če znamo oceniti $(n + 1)$ -ti odvod funkcije, ker števila ξ ne poznamo. Dodatno pa lahko ugotovimo, da je pričakovati, da bo napaka manjša, če bo x nekje v sredi med $x_0, \dots, x_n$, če so le ti med seboj približno enako oddaljeni. To praktično pomeni, da je smiselno pri interpolaciji v obširni tabeli pri danem x upoštevati funkcijske vrednosti na obeh straneh.

d) Numerično odvajanje.

Če je funkcija podana v obliki tabele, je naravno odvod te funkcije aproksimirati z odvodom interpolacijskega polinoma. Pri tem se izkaže, da je enostavneje oceniti napako, če odvode računamo v točkah, v katerih so podane funkcijske vrednosti. Pri $n = 2$ in ekvidistantnih točkah dobimo naslednjo zvezo:

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} + \frac{1}{6} h^2 f'''(\xi)$$

kjer je h razmak v tabeli in ξ neko število med $x - h$ in $x + h$.

Odvod torej lahko dobro aproksimiramo s simetričnim diferenčnim kvocientom. Napaka metode je tem manjša, čim manjši je h . Formula je eksaktna za polinome druge stopnje in primerna za funkcije, ki imajo omejen tretji odvod.

e) Numerično integriranje.

Podobno moremo integral funkcije, ki je podana v obliki tabele, nadomestiti z integralom interpolacijskega polinoma. Pri $n = 1$ in $n = 2$ pridemo do dveh dobro znanih formul:

$$\int_x^{x+h} f(t) dt = \frac{h}{2} (f(x) + f(x+h)) - \frac{h^3}{12} f''(\xi)$$

$$\int_{x-h}^{x+h} f(t) dt = \frac{h}{3} (f(x-h) + 4f(x) + f(x+h)) - \frac{h^5}{90} f^{iv}(\xi)$$

Prva je trapezna formula in je eksaktna za vse polinome prve stopnje, druga je Simpsonova in je eksaktna za vse polinome tretje stopnje. Glede ocene za napako veljajo podobne ugotovitve kot pri odvajanju. Napake se dajo praktično oceniti tudi tako, da iste integrale računamo pri različnih h -jih. Iz dobljenih razlik se da potem sklepati, kolika je napaka.

3. Zaokrožitvena napaka

Z numerično metodo prevedemo reševanje problema na končno zaporedje aritmetičnih operacij. Teh operacij iz praktičnih razlogov ne izvršujemo eksaktno, ampak v okviru neke diskretne množice števil. Pri tem prihaja do zaokrožitvenih napak, ki jih je dokaj težko ocenjevati. Pri ocenjevanju moramo upoštevati vse podrobnosti pri izvrševanju štirih aritmetičnih operacij in natančen vrstni red teh operacij pri konkretnem računu. Te ocene so torej bistveno odvisne od načina računanja. Zaokrožitveno napako moremo pri danem procesu zmanjšati le, če povečamo natančnost računanja.

Analiza zaokrožitvenih napak je pomemben del numerične matematike. Le redki procesi so popolnoma analizirani. Pri nekaterih procesih se da ugotoviti, da je izračunani rezultat eksaktni rezultat pri nekoliko spremenjenih podatkih in dobiti ocene za spremembe podatkov. V teh primerih se zaokrožitvene napake nekako spojijo z neodstranljivo napako.

Če je zaokrožitvena napaka smiselno omejena, pravimo, da je računski proces stabilen. Nestabilni računski procesi so praktično neuporabni.

4. Celotna napaka

Absolutno vrednost razlike med eksaktno rešitvijo teoretičnega problema in izračunano rešitvijo pri danih podatkih, z uporabo numerične metode in z nujnim zaokroževanjem, imenujemo celotna napaka. Oceniti se da z vsoto neodstranljive napake, napake metode in zaokrožitvene napake. Smiselno računanje je tako, da so vsi trije prispevki k celotni napaki istega velikostnega reda. Vložiti veliko truda v to, da zmanjšamo napako metode ali zaokrožitveno napako znatno pod neodstranljivo napako, nima ustreznega učinka.

Naloge sodobnega računarja pri reševanju praktičnih problemov moremo formulirati takole: Probleme je treba formulirati tako, da se izognemo slabi pogojenosti; izbrati je treba najprimernejšo metodo reševanja glede na numerično stabilnost in na ekonomičnost računanja; izvršiti račune s primernim računskim pripomočkom; analizirati napake; tolmačiti rezultate. Dostikrat so te naloge za današnjo znanstveno stopnjo pretežke in spreten računar je včasih bolj umetnik kot znanstvenik.

PREGLED ELEMENTARNE MATEMATIKE S STALIŠČA NUMERIČNEGA RAČUNANJA

ZVONIMIR BOHTE

MOS 65-01

Kratek pregled problemov iz elementarne matematike, pri katerih bi bilo koristno ob obravnavanju omeniti tudi numerično reševanje.

SURVEY OF ELEMENTARY MATHEMATICS FROM THE COMPUTATIONAL STANDPOINT

Short survey of the problems of elementary mathematics in the treatment of which it would be useful to mention also the numerical solution.

Na kratko si bomo ogledali nekatera poglavja iz elementarne matematike, pri katerih se da brez posebnih težav dodati nekaj napotkov v zvezi z numeričnim računanjem. Pri tem bomo izpustili vse, kar se nanaša na računanje z logaritmi. Vse zglede računajmo na 4 decimalna mesta za piko z rezanjem (brez zaokrožanja).

I. razred

1. Linearna interpolacija

Pri računanju »peš« dostikrat uporabljamo razne matematične tabele. Te tabele so navadno tako goste, da je dovolj, če pri argumentih, ki niso v tabeli, poiščemo iskano vrednost z linearno interpolacijo.

Vzemimo, da imamo tabelo za dvakrat zvezno odvedljivo funkcijo $y = f(x)$. Tabela naj bo ekvidistantna, to se pravi, dane naj bodo vrednosti y_i , $i = 0, 1, \dots, N$ pri argumentih $x_0 < x_1 < \dots < x_N$, za katere velja $x_{i+1} - x_i = h$. Vrednosti funkcije $f(x_i)$ naj bodo zaokrožene na t mest za decimalno piko. Velja naj torej

$$|y_i - f(x_i)| \leq \frac{1}{2} 10^{-t} = \delta$$

Naj bo x argument, ki je različen od vseh x_i in naj leži med x_k in x_{k+1} :

$$x_k < x < x_{k+1}$$

Približno vrednost za $y = f(x)$ moremo dobiti z linearno interpolacijo:

$$f(x) \doteq I(x) = f(x_k) + \frac{f(x_{k+1}) - f(x_k)}{x_{k+1} - x_k} (x - x_k)$$

Pri tem je

$$I(x_k) = f(x_k), \quad I(x_{k+1}) = f(x_{k+1})$$

Najprej izpeljimo izraz za napako metode in skušajmo dobiti zanjo primerno oceno! Ker vemo, da je napaka pri $x = x_k$ in $x = x_{k+1}$ enaka nič, je smiselno pričakovati, da bo napaka imela faktor $(x - x_k)(x - x_{k+1})$. Pa si oglejmo izraz

$$F(x) = f(x) - I(x) - R(x - x_k)(x_{k+1} - x)$$

Očitno je $F(x_k) = F(x_{k+1}) = 0$. Svobodni faktor R izberimo tako, da bo $F(x) = 0$ tudi pri našem izbranem x , ki je različen od x_k in x_{k+1} ! Funkcija $F(x)$ ima torej tri različne ničle x_k , x , x_{k+1} . Tako kot $f(x)$ je tudi $F(x)$ dvakrat zvezno odvedljiva funkcija, zato eksistira po Rollejevem izreku števili ξ_1 in ξ_2

$$x_k < \xi_1 < x < \xi_2 < x_{k+1}$$

za kateri je $F'(\xi_1) = F'(\xi_2) = 0$. Če še enkrat uporabimo ta izrek, dobimo rezultat, da eksistira med ξ_1 in ξ_2 tako število ξ , da je

$$F''(\xi) = 0$$

Od tod pa sledi

$$F''(\xi) = f''(\xi) + 2R = 0$$

Od tod dobimo $R = -\frac{1}{2}f''(\xi)$ in

$$f(x) - I(x) = -\frac{1}{2}f''(\xi)(x - x_k)(x_{k+1} - x)$$

V izrazu za napako nastopa poleg faktorja, ki smo ga uganili, še drugi odvod funkcije v neki točki med x_k in x_{k+1} , ki pa je odvisna od x . Napako moremo oceniti, če znamo oceniti drugi odvod tabelirane funkcije na ustreznem intervalu. Če je npr.

$$|f''(x)| \leq M, \quad x_k \leq x \leq x_{k+1}$$

je

$$|f(x) - I(x)| \leq \frac{1}{8} Mh^2,$$

saj zavzame izraz $(x - x_k)(x_{k+1} - x)$ največjo vrednost $h^2/4$ pri $x = (x_k + x_{k+1})/2$.

Poiščimo še oceno za neodstranljivo napako! Ker eksaktnih funkcijskih vrednosti ne poznamo, ne bomo mogli izračunati vrednosti $I(x)$, temveč bomo skušali izračunati vrednost

$$\bar{I}(x) = y_k + \frac{y_{k+1} - y_k}{x_{k+1} - x_k} (x - x_k) = \frac{x_{k+1} - x}{x_{k+1} - x_k} y_k + \frac{x - x_k}{x_{k+1} - x_k} y_{k+1}$$

Od tod sledi izraz za neodstranljivo napako:

$$I(x) - \bar{I}(x) = \frac{x_{k+1} - x}{x_{k+1} - x_k} (f(x_k) - y_k) + \frac{x - x_k}{x_{k+1} - x_k} (f(x_{k+1}) - y_{k+1})$$

in ocena je

$$|I(x) - \bar{I}(x)| \leq \frac{x_{k+1} - x}{x_{k+1} - x_k} \delta + \frac{x - x_k}{x_{k+1} - x_k} \delta = \delta$$

Rezultat je torej ugoden. Neodstranljiva napaka je kvečjemu tolikšna, kot je maksimalna napaka v funkcijskih vrednostih. To je geometrično nazorno. Vendar velja pripomniti, da je to res le, če je x med x_k in x_{k+1} , kajti le tedaj sta oba ulomka pozitivna.

Nazadnje moramo analizirati še zaokrožitveno napako. Vzemimo, da računamo izraz $\bar{I}(x)$ na p decimalah za piko natančno, torej, da pri množenju ali deljenju storimo kvečjemu napako $\eta = \frac{1}{2} 10^{-p}$. Če označimo izračunano vrednost z $\tilde{I}(x)$, dobimo:

$$\tilde{I}(x) = y_k + \left(\frac{y_{k+1} - y_k}{x_{k+1} - x_k} + \varepsilon_1 \right) (x - x_k) + \varepsilon_2 = \bar{I}(x) + \varepsilon_1(x - x_k) + \varepsilon_2,$$

pri čemer je ε_1 napaka, ki smo jo storili pri deljenju, ε_2 pa napaka pri množenju. Za obe velja

$$|\varepsilon_1| \leq \eta, \quad |\varepsilon_2| \leq \eta$$

Od tod hitro ocenimo zaokrožitveno napako:

$$|\bar{I}(x) - \bar{\bar{I}}(x)| \leq |\varepsilon_1|(x - x_k) + |\varepsilon_2| \leq \eta(1 + x - x_k) \leq \eta(1 + h)$$

Če h ni veliko število, utegne biti zaokrožitvena napaka le malo večja kot napaka pri eni računski operaciji.

Če združimo vse tri ocene, moremo dobiti oceno za celotno napako:

$$\begin{aligned} |f(x) - \bar{\bar{I}}(x)| &= |f(x) - I(x) + I(x) - \bar{I}(x) + \bar{I}(x) - \bar{\bar{I}}(x)| \leq \\ &\leq |f(x) - I(x)| + |I(x) - \bar{I}(x)| + |\bar{I}(x) - \bar{\bar{I}}(x)| \leq \\ &\leq \frac{1}{8} Mh^2 + \delta + \eta(1 + h) \end{aligned}$$

Iz te ocene moremo napraviti nekaj zaključkov. Če je h majhno število (npr. $h < 1$), potem je smiselno računati na t decimalk ($\eta = \delta$), ali kvečjemu na eno mesto več ($\eta = \delta/10$). Če računamo mnogo bolj natančno, celotna napaka ne bo dosti manjša. Linearna interpolacija je še smiselna, če je pri dani funkciji h tako izbran, da je

$$\frac{1}{8} Mh^2 \leq \delta$$

To seveda velja le, če dopuščamo napako istega velikostnega reda kot δ . Tedaj je celotna napaka ocenjena v najslabšem primeru s 4δ . Narobe je pričakovati, da bo izračunana vrednost imela — tako kot funkcijske vrednosti — kvečjemu napako δ .

Poskusimo še odgovoriti na vprašanje, kako pri dani tabeli spoznamo, da je linearna interpolacija dopustna! Problem je v tem, kako dobiti uporabno oceno za drugi odvod funkcije.

Definirajmo prve in druge difference funkcije pri razmaku h :

$$\begin{aligned} \Delta f(x) &= f(x + h) - f(x) \\ \Delta^2 f(x) &= \Delta f(x + h) - \Delta f(x) = \\ &= f(x + 2h) - 2f(x + h) + f(x) \end{aligned}$$

Prve in druge difference je pri dani ekvidistantni tabeli zelo enostavno izračunati. Med drugo diferenco in drugim odvodom dvakrat zvezno odvedljive funkcije je zelo enostavna zveza. Z razvojem izrazov $f(x + 2h)$ in $f(x + h)$ v Taylorjevo vrsto okrog točke $x + h$ dobimo:

$$\begin{aligned} \Delta^2 f(x) &= f(x + h) + hf'(x + h) + \frac{1}{2} h^2 f''(\xi_1) - \\ &\quad - 2f(x + h) + \\ &\quad + f(x + h) - hf'(x + h) + \frac{1}{2} h^2 f''(\xi_2) = \\ &= \frac{1}{2} h^2 (f''(\xi_1) + f''(\xi_2)) = h^2 f''(\xi) \end{aligned}$$

Tu je ξ_1 med $x + h$ in $x + 2h$, ξ_2 med x in $x + h$, ξ pa med ξ_1 in ξ_2 , torej nekje med x in $x + 2h$.

Pri funkcijah, pri katerih se drugi odvod močno ne spreminja, moremo ta odvod aproksimirati z izrazom $\Delta^2 f(x)/h^2$.

Pogoj za primernost linearne interpolacije moremo tako izraziti z drugo diferenco: Če je

$$\frac{1}{8} \Delta^2 f(x) = \frac{1}{8} h^2 f''(\xi) \leq \delta = \frac{1}{2} 10^{-t}$$

ali

$$\Delta^2 f(x) \leq 4 \cdot 10^{-t}$$

povsod na nekem delu tabele, potem je tam linearna interpolacija dopustna. Druge difference ne smejo torej presegati štirih enot na zadnjem mestu.

Za zgled vzemimo funkcijo $f(x) = \sin x$ in $t = 4$! Kako gosto je treba tabelirati funkcijo $\sin x$ na 4 decimalke natančno, da bo povsod zadoščala linearna interpolacija? Ocena za drugi odvod je očitno 1, torej $M = 1$. Če je h tako izbran, da je

$$h \leq \sqrt[4]{4 \cdot 10^{-4}} = 2 \cdot 10^{-2} = 0.02,$$

bo linearna interpolacija vedno zadoščala. Maksimalni korak, izražen v stopinjah, je 1.14° . Razmak 1° je torej povsod dovolj majhen.

Oglejmo si dva odseka tabele za $\sin x$ v bližini $x = \pi/2$! Prva tabela naj bo za kote v stopinjah z razmakom 1° , druga pa za kote v radianih z razmakom 0.02.

x	$\sin x$	Δ	Δ^2	x	$\sin x$	Δ	Δ^2
85°	0.9962			1.48	0.9959		
		14				16	
86°	0.9976		-4	1.50	0.9975		-4
		10				12	
87°	0.9986		-2	1.52	0.9987		-4
		8				8	
88°	0.9994		-4	1.54	0.9995		-4
		4				4	
89°	0.9998		-2	1.56	0.9999		-3
		2				1	
90°	1.0000			1.58	1.0000		

Če izračunamo z linearno interpolacijo $\sin 88^\circ 30'$, dobimo vrednost 0.9996, in za $\sin 1.55$ dobimo 0.9997. Obe vrednosti sta za 1 na zadnjem mestu premajhni, kar je praktično največja napaka, ki jo utegnemo narediti pri tej tabeli. Druge difference so pri manjših x kvečjemu manjše in zato je manjša tudi napaka pri linearni interpolaciji.

Druge probleme si bomo ogledali bolj na kratko.

2. Sistemi linearnih enačb

a) Sistemi linearnih enačb so lahko slabo pogojeni, kar pomeni, da utegnejo majhne spremembe v podatkih povzročiti velike spremembe v rešitvi. Oglejmo si naslednji sistem dveh enačb:

$$100x + 99y = 199$$

$$99x + 98y = 197$$

Rešitev tega sistema je $x = y = 1$. Če desne strani malo spremenimo, npr.

$$100x + 99y = 198.99$$

$$99x + 98y = 197.01$$

dobimo sistem, ki ima rešitev $x = 2.97$, $y = -0.99$. Ta se znatno razlikuje od prejšnje. Relativni spremembi v desnih straneh sta približno $2 \cdot 10^{-5}$, relativni spremembi v rešitvi pa skoraj 2. Če prvo enačbo delimo s 100, jo nato pomnožimo z 99 in odštejemo od druge enačbe, dobimo navedeno eksaktno rešitev, saj nismo napravili nobene zaokrožitvene napake. Če pa delimo drugo enačbo z 99, se deljenje ne izide. Po četrti decimalki koeficient pri y in desno stran odrežimo, nato to enačbo pomnožimo s 100 in jo odštejemo od prve! Na ta način dobimo popolnoma drugačno »rešitev«: $y = -0.5$, $x = 2.4849$.

Sistem enačb, pri katerem majhna sprememba v desnih straneh ali v koeficientih povzroči veliko spremembo v rešitvi, je slabo pogojen. To ima za posledico tudi veliko spremembo v izračunani rešitvi, če spremenimo vrstni red aritmetičnih operacij, saj se takrat spremenijo tudi zaokrožitvene napake.

Geometrično je nazorno, zakaj je naš primer slabo pogojen. Sistem enačb določa presečišče dveh skoraj vzporednih premic, ki zato slabo določata presečišče. Majhen premik ene od obeh premic povzroči, da se premakne presečišče daleč stran.

Zapišimo sistem linearnih enačb v matrični obliki

$$Ax = b$$

Pri tem predpostavimo, da je $\det A \neq 0$, torej, da eksistira inverzna matrika A^{-1} . Če spremenimo desne strani za vektor δb , naj ima sistem rešitev $\bar{x}$: $A\bar{x} = b + \delta b$.

Od tod sledi:

$$\bar{x} = A^{-1}b + A^{-1}\delta b = x + A^{-1}\delta b$$

Rešitev se torej spremeni za vektor $A^{-1}\delta b$. Očitno je tudi, da velikost elementov inverzne matrike odloča o velikosti sprememb v rešitvi.

V našem primeru je

$$A = \begin{bmatrix} 100 & 99 \\ 99 & 98 \end{bmatrix}, \quad A^{-1} = \begin{bmatrix} -98 & 99 \\ 99 & -100 \end{bmatrix}, \quad \delta b = \begin{bmatrix} -0.01 \\ 0.01 \end{bmatrix}$$

$$A^{-1}\delta b = \begin{bmatrix} 1.97 \\ -1.99 \end{bmatrix}$$

kar je v skladu z našimi rezultati. Velikost elementov inverzne matrike postane bolj očitna, če obe enačbi najprej delimo s 100.

Tedaj je

$$A = \begin{bmatrix} 1.00 & 0.99 \\ 0.99 & 0.98 \end{bmatrix}, \quad A^{-1} = \begin{bmatrix} -9800 & 9900 \\ 9900 & -10000 \end{bmatrix}, \quad \delta b = \begin{bmatrix} -0.0001 \\ 0.0001 \end{bmatrix}$$

Produkt $A^{-1}\delta b$ ostane pri tem seveda nespremenjen.

Če želimo natančno rešiti slabo pogojen sistem, moramo računati na več decimalk. Tudi pri dobro pogojenih sistemih je včasih važno, v katerem vrstnem redu eliminiramo neznanke. Obravnavanje podrobnosti v zvezi z reševanjem sistemov linearnih enačb bi nas zaneslo predaleč.

b) Sisteme linearnih enačb moremo reševati tudi iterativno. Čeprav je pri Gaussovi metodi potrebno izvršiti le končno število aritmetičnih operacij, da pridemo do rešitve, moremo pri sistemih posebne oblike hitreje priti do rešitve drugače. Oglejmo si naslednji primer sistema treh enačb:

$$\begin{aligned} 100x + y + z &= 102 \\ x + 100y + 2z &= 103 \\ 2x + 2y + 100z &= 104 \end{aligned}$$

Ta sistem, ki ima to posebnost, da diagonalni elementi močno prekašajo vse ostale, najprej zapišimo v obliki

$$\begin{aligned} x &= (102 - y - z)/100 \\ y &= (103 - x - 2z)/100 \\ z &= (104 - 2x - 2y)/100 \end{aligned}$$

Sedaj moremo tvoriti zaporedje približkov takole (Jacobijeva iteracija):

$$\begin{aligned} \text{(i)} \quad & x_{r+1} = (102 - y_r - z_r)/100 \\ & y_{r+1} = (103 - x_r - 2z_r)/100 \\ & z_{r+1} = (104 - 2x_r - 2y_r)/100 \quad r = 0, 1, \dots \end{aligned}$$

ali pa takole (Seidelova iteracija):

$$\begin{aligned} \text{(ii)} \quad & x_{r+1} = (102 - y_r - z_r)/100 \\ & y_{r+1} = (103 - x_{r+1} - 2z_r)/100 \\ & z_{r+1} = (104 - 2x_{r+1} - 2y_{r+1})/100 \quad r = 0, 1, \dots \end{aligned}$$

Začnemo s poljubnim začetnim približkom. Ponavadi vzamemo kar $x_0 = y_0 = z_0 = 0$. Rezultate obeh iteracij zapišimo v naslednji tabeli:

(i)	r	x_r	y_r	z_r
	0	0.0000	0.0000	0.0000
	1	1.0200	1.0300	1.0400
	2	0.9993	0.9990	0.9988
	3	1.0000	1.0000	1.0000
(ii)	r	x_r	y_r	z_r
	0	0.0000	0.0000	0.0000
	1	1.0200	1.0198	0.9992
	2	0.9998	1.0000	1.0000
	3	1.0000	1.0000	1.0000

Da se dokazati, da obe zaporedji približkov konvergirata k rešitvi sistema pri poljubnem začetnem približku, če v vsaki vrstici matrike diagonalni element po absolutni vrednosti prekaša vsoto absolutnih vrednosti drugih elementov v isti vrstici. Čim večja je dominantnost diagonalnih elementov, tem večja je hitrost konvergence.

II. razred

3. Kvadratni koren iz kompleksnega števila

Naj bo $a = a + bi$ in $\xi = \sqrt{a} = x + iy$. Kvadratni koren ξ iz kompleksnega števila a dobimo iz sistema enačb

$$x^2 - y^2 = a, \quad 2xy = b$$

Najprej izračunamo absolutno vrednost števila a :

$$|a| = \sqrt{a^2 + b^2}$$

nato pa določimo x in y s korenjenjem iz enačb

$$x^2 = \frac{1}{2} (|a| + a), \quad y^2 = \frac{1}{2} (|a| - a)$$

Nastopi vprašanje, ali je ta pot primerna za računanje pri vseh a in b . Oglejmo si dva primera:

$$\begin{aligned} a &= 1 + 0.04i, & \xi &= 1.0002 + 0.0200i \\ a &= 1 + 0.008i, & \xi &= 1.0000 \end{aligned}$$

Prvi kvadratni koren je na 4 decimalna mesta pravilno izračunan, drugi pa ne. V drugem primeru je na 4 mesta pravilen rezultat

$$\xi = 1.0000 + 0.0040i$$

ki ga dobimo npr. tako, da imaginarno komponento y izračunamo z deljenjem: $y = b/(2x)$.

Pri uporabi zgornjih formul utegne nastopiti težava samo v primeru, ko je $|b|$ majhno število v primeri z $|a|$. Tedaj je $|a| \doteq a$, če je $a > 0$, in $|a| \doteq -a$, če je $a < 0$. Zato je bodisi y , bodisi x , narobe izračunan, saj se pri odštevanju skoraj vsa mesta odštejejo in absolutna napaka kvadratnega korena je precej velika, relativna pa še večja. Če le moremo, se pri računanju izogibajmo odštevanju skoraj enakih števil!

Pravilen način računanja je naslednji: Če je $a > 0$, izračunamo ξ po formulah

$$x^2 = \frac{1}{2} (|a| + a), \quad y = b/(2x)$$

Če pa je $a < 0$, pa po formulah

$$y^2 = \frac{1}{2} (|a| - a), \quad x = b/(2y)$$

Vedno izračunamo poleg $|a|$ le tisti kvadratni koren, pri katerem se v radikandu sumanda res seštejeta. Drugo komponento izračunamo iz enačbe $2xy = b$. Tako si prihranimo računanje enega kvadratnega korena in se izognemo nepotrebno velikim napakam v posebnih primerih. Pri izdelavi dobrega podprograma za računanje kvadratnega korena iz kompleksnega števila moramo posebej upoštevati tudi primera, ko je $a = 0$ ali $b = 0$.

4. Kvadratna enačba

Vzemimo kvadratno enačbo v poenostavljeni obliki:

$$x^2 + 2px + q = 0$$

Njena korena x_1 in x_2 sta:

$$x_1 = -p + \sqrt{D}, \quad x_2 = -p - \sqrt{D}$$

kjer je

$$D = p^2 - q$$

Mislimo si, da koeficienta p in q malo spremenimo. Zanima nas, koliko se pri tem spremenita oba korena enačbe. Poglejmo torej, kakšna korena ima enačba:

$$x^2 + 2(p + \varepsilon r)x + (q + \varepsilon s) = 0$$

pri majhnih ε ! Pri tem predpostavimo, da velja

$$\max(|r|, |s|) \leq \max(|p|, |q|)$$

tako, da r in s nista večjega reda velikosti kot p ali q .

Če zapišemo oba korena spremenjene enačbe v obliki

$$\begin{aligned} x(\varepsilon) &= -(p + \varepsilon r) \pm \sqrt{(p + \varepsilon r)^2 - (q + \varepsilon s)} = \\ &= -p - \varepsilon r \pm \sqrt{D + (2pr - s)\varepsilon + \varepsilon^2 r^2} \end{aligned}$$

vidimo, da moramo ločiti dva primera.

a) $D \neq 0$

Enačba ima pri $\varepsilon = 0$ dva različna korena: $x_1 \neq x_2$. Tedaj je

$$x(\varepsilon) = -p - \varepsilon r \pm \sqrt{D} \sqrt{1 + u\varepsilon + v\varepsilon^2}$$

kjer je

$$u = (2pr - s)/D, \quad v = r^2/D$$

Pri majhnih ε je

$$\sqrt{1 + u\varepsilon + v\varepsilon^2} = 1 + \frac{1}{2} u\varepsilon + O(\varepsilon^2)$$

Zato ima pri majhnih ε enačba dva korena:

$$x_1(\varepsilon) = x_1 + (-r + (2pr - s)/(2\sqrt{D}))\varepsilon + O(\varepsilon^2)$$

$$x_2(\varepsilon) = x_2 + (-r - (2pr - s)/(2\sqrt{D}))\varepsilon + O(\varepsilon^2)$$

Pri majhnih ε je glavni del spremembe v korenih kvadratne enačbe enak izrazu

$$(-r \pm (2pr - s)/(2\sqrt{D}))\varepsilon$$

ki je tem večji, čim manjša je absolutna vrednost diskriminante.

b) $D = 0$

Enačba ima pri $\varepsilon = 0$ dvojni koren $x_1 = x_2$. Pri majhnem ε pa ima enačba dva korena, ki ju lahko zapišemo v obliki:

$$x(\varepsilon) = -p - \varepsilon r \pm \sqrt{2pr - s} \sqrt{\varepsilon} \sqrt{1 + r^2 \varepsilon / (2pr - s)} = -p \pm \sqrt{2pr - s} \sqrt{\varepsilon} + O(\varepsilon)$$

Če je $2pr - s \neq 0$, ima enačba pri majhnem ε dva korena, ki se od dvojnega korena $x_1 = -p$ razlikujeta približno za

$$\pm \sqrt{2pr - s} \sqrt{\varepsilon}$$

Ker je $\sqrt{\varepsilon}$ znatno več kot ε , moremo trditi, da so dvojni koreni slabo pogojeni. To pomeni, da se v splošnem dvojni koren spremeni za količino reda velikosti 10^{-t} , če koeficiente v enačbi spremenimo za količino reda velikosti 10^{-2t} . Le v posebnem primeru $2pr - s = 0$ ima enačba dva korena:

$$x_1(\varepsilon) = -p, \quad x_2(\varepsilon) = -p - 2r\varepsilon$$

V tem zelo specialnem primeru ne moremo trditi, da je dvojni koren slabo pogojen.

Pri računanju slabo pogojenih ničel že majhne zaokrožitvene napake zelo poslabšajo natančnost rezultatov.

Oglejmo si nekaj primerov!

$$\begin{aligned} \text{(i)} \quad & x^2 - 0.63x + 0.0991 = 0 \\ & x = 0.315 \pm \sqrt{0.0001} = 0.3150 \pm 0.0100 \\ & x = (0.63 \pm \sqrt{0.0005})/2 = 0.3150 \pm 0.0112 \end{aligned}$$

Pri prvem računu smo pri računanju diskriminante odrezali zadnji dve mesti. Pri korenjenju se ta napaka zelo poveča. Pri drugem računu je diskriminanta eksaktno izračunana in zato sta korena na vsa 4 mesta prav izračunana. To je primer dveh bližnjih realnih korenov, ki sta slabo pogojena. Zaokrožitvena napaka $25 \cdot 10^{-6}$ pri računu diskriminante je povzročila napako $12 \cdot 10^{-4}$ v korenih.

$$\begin{aligned} \text{(ii)} \quad & x^2 - 0.6319x + 0.0998 = 0 \\ & x = 0.3159 \pm \sqrt{-0.0001} = 0.3159 \pm 0.0100i \\ & x = 0.3160 \pm \sqrt{0.0000} = 0.3160 \pm 0.0000 \end{aligned}$$

Tudi tu gre za dva bližnja realna korena. Če namesto $0.6319/2$ vzamemo 0.3159 , dobimo par konjugirano kompleksnih korenov, če pa 0.3160 , en dvojni realni koren. Na 4 mesta natančna korena sta 0.3209 in 0.3110 . Če želimo natančno izračunati slabo pogojene korene, moramo računati na več decimalk.

$$(iii) \quad x^2 - 92.45x + 0.1005 = 0$$

Če računamo samo s štirimestnimi števili z zaokrožanjem, dobimo

$$x = (92.45 \pm \sqrt{8547})/2 = (92.45 \pm 92.45)/2 \\ x_1 = 92.45, \quad x_2 = 0.0000$$

Na 4 mesta natančna korena sta 92.45 in 0.001087 . Večji koren je pravilno izračunan, manjši pa popolnoma napačno, saj se pri odštevanju vsa mesta uničijo. Tu ne gre za slabo pogojenost korenov, ampak za nepravilen način računanja. Če izračunamo večji koren kot prej, manjšega pa z deljenjem $0.1005/92.45$, dobimo vsa 4 mesta prav.

Kadar kaže, da se utegnejo pri enem izmed korenov točna mesta izgubiti, je najbolje, da po klasični formuli izračunamo le absolutno večji koren, to je tistega, pri katerem se oba člena seštejeta. Drugi koren, manjši po absolutni vrednosti, pa moremo vedno natančno izračunati iz svobodnega člena v enačbi, ki je produkt korenov. Seveda pride ta nasvet v poštev le pri realnih korenih. Pri enačbi

$$x^2 + 2px + q = 0,$$

pri kateri je $D = p^2 - q > 0$, izračunamo

$$x_1 = -p - \sqrt{D}, \quad x_2 = q/x_1$$

če je $p \geq 0$ in

$$x_1 = -p + \sqrt{D}, \quad x_2 = q/x_1$$

če je $p < 0$.

Tak način računanja je potreben samo v primeru, ko imamo dva realna korena, ki sta močno različna po absolutni vrednosti.

III. razred

5. Nelinearne enačbe

a) Navadna iteracija

Vzemimo, da iščemo koren α enačbe

$$f(x) = 0,$$

kjer je $f(x)$ zvezna in zvezno odvedljiva funkcija na nekem intervalu okrog korena. Včasih je zelo uspešna numerična metoda za reševanje takih enačb navadna iteracija. Enačbo najprej zapišemo v ekvivalentni obliki:

$$x = g(x)$$

nato si izberemo primeren začetni približek x_0 in tvorimo zaporedje približkov po pravilu:

$$x_{r+1} = g(x_r), \quad r = 0, 1, 2, \dots$$

Geometrično nazorno je in da se tudi dokazati, da zaporedje x_r konvergira proti korenu α , če je

$$|g'(a)| < 1$$

in je x_0 zadosti dober približek za koren. Čim manjša je absolutna vrednost odvoda, tem hitreje zaporedje konvergira k rešitvi. Oglejmo si primer!

Enačbo

$$x^3 - 5x + 1 = 0$$

moremo na nešteto načinov zapisati v obliki $x = g(x)$. Eno od teh smo že spoznali v uvodnem sestavku ($x = (1 + x^3)/5$). Navedimo še dve preprosti obliki:

$$x = 1/(5 - x^2), \quad x = \sqrt[3]{5x - 1}$$

Če pri prvi začnemo z $x_0 = 0$, dobimo $x_1 = 0.2$ in $x_2 = 0.2016$, kar je že na 4 mesta pravi rezultat. Pri drugi obliki pa dobimo pri vsakem začetnem približku enega od preostalih dveh korenov, npr. $x_0 = 2$, $x_1 = 2.08$, $x_2 = 2.11$, $x_3 = 2.121$, $x_4 = 2.125$, itd. (koren je 2.1284...); $x_0 = -2$, $x_1 = -2.224$, $x_2 = -2.297$, $x_3 = -2.319$, $x_4 = -2.327$, itd. (koren je -2.3300...).

Ta metoda ni zelo univerzalna. V posebnih primerih utegne biti uspešna in je zaradi preprostega poteka računanja včasih bolj ekonomična kot druge.

b) Računanje kvadratnega korena

Pri računanju »peš« je korenjenje števil zelo neudobna operacija. Če nimamo pri roki logaritmskih tablic, ali če potrebujemo kvadratni koren na več mest, kot pa dopuščajo tabele, si moramo pomagati kako drugače. Računanje kvadratnega korena po klasični metodi, ko ugotavljamo cifro za cifro tega korena, pride komajda v poštev. Če imamo na razpolago namizni računski strojček, je zelo ugodna naslednja metoda za računanje kvadratnega korena:

Vzemimo, da želimo izračunati $\sqrt{a}$. To število je koren enačbe $f(x) = x^2 - a = 0$. Koren te enačbe moremo računati s tangentno metodo:

$$x_{r+1} = x_r - f(x_r)/f'(x_r) = x_r - (x_r^2 - a)/(2x_r) = \frac{1}{2}(x_r + a/x_r), \quad r = 0, 1, \dots$$

Za začetni približek x_0 vzamemo čimboljši približek za $\sqrt{a}$, ki ga uganemo na pamet. Na vsakem koraku iteracije je treba izračunati eno deljenje in eno aritmetično sredino dveh števil. To zaporedje konvergira h $\sqrt{a}$ pri vsakem začetnem približku $x_0 > 0$, kar je geometrično popolnoma nazorno, saj je x_{r+1} presečišče tangente na parabolo $y = x^2 - a$ pri $x = x_r$ z osjo x . Če je $x_0 > \sqrt{a}$, je zaporedje približkov monotono padajoče proti $\sqrt{a}$. Če pa je $0 < x_0 < \sqrt{a}$, potem je $x_1 > \sqrt{a}$ in za nadaljnje približke velja prejšnja ugotovitev. Konvergenca približkov je kvadratična, kar pomeni, da se število točnih decimalk na vsakem koraku iteracije približno podvoji.

Izračunajmo na ta način $\sqrt{5}$!

$$\begin{array}{llll} x_0 = 2 & x_1 = 2.25 & x_2 = 2.236 & x_3 = 2.236068 \\ 5/x_0 = 2.5 & 5/x_1 = 2.222 & 5/x_2 = 2.236136 & \end{array}$$

Tretji približek je natančen na vseh 6 mest.

Na podoben način izpeljemo iterativni postopek za računanje $\sqrt[n]{a}$:

$$x_{r+1} = x_r - (x_r^n - a)/(nx_r^{n-1}) = \frac{1}{n}((n-1)x_r + a/x_r^{n-1}), \quad r = 0, 1, \dots$$

Izračunajmo po tem postopku $\sqrt[3]{9}$!

$$\begin{array}{lll} x_0 = 2 & x_1 = 2.08 & x_2 = 2.08008 \\ 2x_0 = 4 & 2x_1 = 4.16 & \\ 9/x_0^2 = 2.25 & 9/x_1^2 = 2.08025 & \\ \underline{6.25} & \underline{6.24025} & \end{array}$$

Drugi približek je natančen na vseh pet decimalk.

c) Računanje kompleksnih korenov

Če iščemo tudi kompleksne korene enačbe

$$f(x) = 0,$$

moremo prav tako kot za realne korene uporabiti Newtonovo (tangentno) metodo:

$$x_{r+1} = x_r - f(x_r)/f'(x_r), \quad r = 0, 1, \dots$$

Toda, če je $f(x)$ realna funkcija, ne bomo mogli z realnim začetnim približkom izračunati kompleksnega korena. Zato je v takem primeru potrebno začeti s kompleksnim začetnim približkom.

d) »Izboljšana« sekantna metoda

Kadar želimo izračunati realni koren enačbe

$$f(x) = 0$$

s sekantno metodo, potem moramo najprej določiti interval $[a, b]$, na krajiščih katerega ima $f(x)$ nasproten predznak. Nato zvežemo točki $(a, f(a))$ in $(b, f(b))$ s premico in izračunamo presečišče te sekante z osjo x . Od dveh delnih intervalov $[a, c]$ in $[c, b]$ izberemo nato tistega, ki ima lastnost, da ima $f(x)$ na krajiščih nasproten znak. Na ta način dobimo zaporedje intervalov, ki vedno vsebujejo koren enačbe in katerih širine gredo proti nič. Toda ta konvergenca je lahko zelo počasna.

Hitrost konvergence lahko občutno povečamo, ne da bi povečali število računskih operacij, takole: Če že imamo dva približka za koren, npr. x_{r-1} in x_r , potem dobimo naslednji približek x_{r+1} kot presečišče sekante skozi točki $(x_{r-1}, f(x_{r-1}))$ in $(x_r, f(x_r))$ z osjo x :

$$x_{r+1} = x_r - f(x_r) (x_r - x_{r-1}) / (f(x_r) - f(x_{r-1})), \quad r = 1, 2, \dots$$

Pri tem se nič ne oziramo na znake števil $f(x_{r-1})$ in $f(x_r)$. Če delamo tako, izgubimo ugodno lastnost, ki jo ima sekantna metoda, namreč to, da zaporedje približkov vedno konvergira h korenu. Zgodi se namreč lahko, da novo zaporedje sploh ne konvergira, podobno, kot se to lahko zgodi pri tangentni metodi. Toda, če vemo, da je potek funkcije $f(x)$ tak, da se to ne bo zgodilo (npr. da v bližini korena $f'(x)$ in $f''(x)$ ne spremenita znaka), pridobimo na hitrosti konvergence. Dokazati se da, da zaporedje približkov, ki ga tvorimo po tej izboljšani metodi, konvergira v bližini enostavnega korena enačbe s hitrostjo 1.62 . To pomeni, da se na vsakem koraku število točnih decimalk poveča približno za faktor 1.62 .

Kot primer računajmo koren enačbe:

$$x^3 + 2x^2 + 10x - 20 = 0$$

ki leži med 1 in 2, s sekantno, izboljšano sekantno in tangentno metodo na 4 decimalna mesta.

r	$x_r^{(1)}$	$x_r^{(2)}$	$x_r^{(3)}$
0	1.0000	1.0000	1.0000
1	1.3043	2.0000	1.4667
2	1.3579	1.3043	1.3716
3	1.3669	1.3579	1.3689
4	1.3684	1.3690	1.3688
5	1.3687	1.3688	
6	1.3688		

Če računamo vrednost polinoma po Hornerjevi metodi, je pri navadni sekantni metodi potrebno izvršiti v zgornji tabeli 24, pri izboljšani sekantni metodi 16 in pri tangentni metodi 20 množenj. Če bi hoteli računati koren na več decimalk natančno, ali če bi imeli opravka z enačbo višje stopnje, bi bilo še bolj očitno, da je izboljšana sekantna metoda bolj ekonomična od tangentne, ta pa od navadne sekantne metode. To seveda velja samo v primeru, ko vse tri metode dajo konvergentno zaporedje.

IV. razred

6. Vsota geometričnega zaporedja

Pri obrestno obrestnem računu in tudi drugod večkrat naletimo na vsoto geometričnega zaporedja

$$1 + a + a^2 + \dots + a^{n-1} = (a^n - 1)/(a - 1)$$

Zastavimo si vprašanje, katera stran v zgornji identiteti je primernejša za računanje! Če ne računamo števila a^n z logaritmi, ampak z računskim strojem, je število množenj na levi in desni približno enako, le seštevanj je na levi več. Glede števila aritmetičnih operacij med obema možnostma torej ni odločilne razlike. Pogledati pa moramo še, kako je z računsko napako. Izkaže se, da je pri številih a , ki so blizu 1, kot je to navadno pri obrestno obrestnem računu, ocena za zaokrožitveno napako pri računanju izraza $1 + a + \dots + a^{n-1}$ manjša od ocene pri izrazu $(a^n - 1)/(a - 1)$.

Izračunajmo najprej en primer, nato pa ocenimo zaokrožitveni napaki! Vzemimo $n = 10$ in $a = 1.025$! Tedaj izračunamo

$$1 + a + \dots + a^9 = 11.2029$$

$$(a^{10} - 1)/(a - 1) = 11.1840$$

Računali smo na 4 decimalke z rezanjem. Na 4 decimalke natančen rezultat je 11.2034. Torej je v prvem primeru dejanska napaka $5 \cdot 10^{-4}$, v drugem pa $194 \cdot 10^{-4}$.

Pri analizi zaokrožitvenih napak upoštevajmo, da pri seštevanju in odštevanju ne napravimo nobene napake, pri množenju in deljenju pa kvečjemu napako 10^{-t} , če računamo na t mest za decimalno piko z rezanjem!

Analizirajmo najprej račun izraza:

$$s = 1 + a + a^2 + \dots + a^{n-1}$$

Edine napake, ki jih storimo, so napake pri računanju zaporednih potenc števila a . Izračunano vrednost i -te potence označimo z $\bar{a}^i$, izračunano vrednost vsote pa z $\bar{s}$. Analiza poteka takole:

Izračunano vrednost i -te potence dobimo tako, da s številom a pomnožimo izračunano vrednost $(i - 1)$ -te potence in ta produkt okrajšamo na t mest. Torej velja:

$$\bar{a}^i = a \bar{a}^{i-1} + \varepsilon_i, \quad |\varepsilon_i| \leq 10^{-t}$$

Če označimo napako i -te potence z η_i

$$\bar{a}^i = a^i + \eta_i, \quad i = 2, 3, \dots, n - 1,$$

dobimo brez težav zvezo

$$\eta_i = a^{i-2} \varepsilon_2 + a^{i-3} \varepsilon_3 + \dots + a \varepsilon_{i-1} + \varepsilon_i$$

in oceno

$$|\eta_i| \leq 10^{-t} (a^{i-1} - 1)/(a - 1)$$

Od tod potem sledi

$$\bar{s} = s + \eta,$$

pri čemer je

$$\eta = \eta_2 + \eta_3 + \dots + \eta_{n-1}$$

in velja ocena

$$\begin{aligned} |\eta| &\leq 10^{-t}(a - 1 + a^2 - 1 + \dots + a^{n-2} - 1)/(a - 1) = \\ &= \frac{10^{-t}}{a - 1} \left(\frac{a^{n-1} - 1}{a - 1} - (n - 1) \right) \end{aligned}$$

V našem primeru dobimo oceno za napako

$$|\eta| \leq 38 \cdot 10^{-4}$$

kar je skoraj 8-krat več kot dejanska napaka.

Sedaj pa analizirajmo računski postopek po drugi poti! Izračunano vrednost imenujmo sedaj $\bar{s}$! Očitno velja

$$\bar{s} = (\bar{a}^n - 1)/(a - 1) + \varepsilon_1, \quad |\varepsilon_1| \leq 10^{-t}$$

kjer je ε_1 napaka pri zaokrožitvi kvocienta na t mest. Če označimo napako te vrednosti z η_1 , je

$$\bar{s} = s + \eta_1$$

pri čemer velja

$$\eta_1 = \eta_n/(a - 1) + \varepsilon_1$$

Tu smo upoštevali, da je $\bar{a}^n = a^n + \eta_n$ in $s = (a^n - 1)/(a - 1)$. Oceno za $|\eta_n|$ že poznamo. Torej je

$$|\eta_1| \leq 10^{-t} (|a^{n-1} - 1|/(a - 1)^2 + 1)$$

V našem primeru dobimo

$$|\eta_1| \leq 399 \cdot 10^{-4},$$

kar je dvakrat več od dejanske napake. Ocena je v drugem primeru približno 10-krat slabša od ocene v prvem in zato je pametneje računati po prvi poti. Če si ogledamo razliko obeh ocen za zaokrožitveno napako, vidimo, da se razlikujeta pri $a > 1$ za izraz $10^{-t}((n - 1)/(a - 1) + 1)$, ki je tem večji, čim bližje k 1 je število a . Tudi če a ni blizu 1, je druga ocena večja.

7. Numerično odvajanje

Naj bo funkcija $y = f(x)$ podana v obliki ekvidistantne tabele: (x_i, y_i) , $i = 0, 1, \dots, N$. Že večkrat smo omenili, da moremo izračunati približno vrednost odvoda s simetričnim diferenčnim kvocientom:

$$f'(x_i) \doteq y_i' = \frac{1}{2h} (y_{i+1} - y_{i-1})$$

kjer je $h = x_{k+1} - x_k$, razmak v tabeli. Do te približne formule pridemo, če poiščemo skozi tri sosednje točke (x_{i-1}, y_{i-1}) , (x_i, y_i) , (x_{i+1}, y_{i+1}) parabolo in odvod naše funkcije nadomestimo z odvodom te kvadratne funkcije pri $x = x_i$. Enostavneje je ta izraz izračunati, če vpeljemo novo spremenljivko s z izrazom

$$s = (x - x_i)/h$$

Tedaj zapišemo iskano parabolo kot kvadratno funkcijo v spremenljivki s :

$$y = ax^2 + bx + c = as^2 + \beta s + \gamma$$

Odvod na x izrazimo z odvodom na s :

$$\frac{dy}{dx} = \frac{dy}{ds} \frac{ds}{dx} = \frac{1}{h} \frac{dy}{ds}$$

Odvod na x pri $x = x_i$ se izraža torej z odvodom na s pri $s = 0$. Tako dobimo

$$y'(x_i) = y_i' = \left. \frac{1}{h} \frac{dy}{ds} \right|_{s=0} = \frac{1}{h} \beta$$

Koeficient β dobimo iz naslednjih pogojev, ki povedo, naj kvadratna funkcija v s poteka skozi točke $(-1, y_{i-1})$, $(0, y_i)$, $(1, y_{i+1})$:

$$a - \beta + \gamma = y_{i-1}$$

$$\gamma = y_i$$

$$a + \beta + \gamma = y_{i+1}$$

Od tod dobimo

$$\beta = \frac{1}{2} (y_{i+1} - y_{i-1}).$$

In naša formula je izpeljana.

Sedaj hočemo oceniti napako. Vzemimo, da je funkcija $f(x)$ trikrat zvezno odvedljiva na intervalu $[x_0, x_N]$! Vrednosti y_{i+1} in y_{i-1} razvijmo v Taylorjevo vrsto okrog točke x_i :

$$y_{i+1} = f(x_{i+1}) = f(x_i + h) = f(x_i) + hf'(x_i) + \frac{1}{2} h^2 f''(x_i) + \frac{1}{6} h^3 f'''(\xi_1)$$

$$y_{i-1} = f(x_{i-1}) = f(x_i - h) = f(x_i) - hf'(x_i) + \frac{1}{2} h^2 f''(x_i) - \frac{1}{6} h^3 f'''(\xi_2)$$

Tu je ξ_1 neko število med x_i in x_{i+1} , ξ_2 pa med x_{i-1} in x_i . Če oba izraza odštejemo, dobimo

$$y_{i+1} - y_{i-1} = 2hf'(x_i) + \frac{1}{6} h^3 (f'''(\xi_1) + f'''(\xi_2))$$

Če je $f'''(x)$ zvezna funkcija, potem obstaja med x_{i-1} in x_{i+1} tako število ξ , da je $\frac{1}{2} (f'''(\xi_1) + f'''(\xi_2)) = f'''(\xi)$, saj zvezna funkcija zavzame vsako vrednost med dvema svojima vrednostma. Od tod dobimo formulo za približno vrednost odvoda z napako vred:

$$f'(x_i) = \frac{1}{2h} (y_{i+1} - y_{i-1}) - \frac{1}{6} h^2 f'''(\xi)$$

Napaka metode je torej sorazmerna kvadratu razmaka h in tretjemu odvodu funkcije v neki vmesni točki. Simetrični diferenčni kvocient nam da eksaktno vrednost odvoda v srednji točki, če je funkcija $f(x)$ kvadratna funkcija.

Oglejmo si še, kakšna je pri tej formuli neodstranljiva napaka! Mislimo si, da so y_i približki za $f(x_i)$ z maksimalno napako δ . Očitno se utegne tedaj simetrični diferenčni kvocient razlikovati od pravega za največ δ/h . Od tod sledi: čim manjši je h , tem večja utegne biti neodstranljiva napaka pri računanju simetričnega diferenčnega kvocienta. Če zanemarimo zaokrožitveno napako, se da tedaj celotna napaka oceniti z izrazom

$$\frac{\delta}{h} + \frac{1}{6} h^2 M_3,$$

kjer je M_3 ocena za absolutno vrednost tretjega odvoda naše funkcije. Če gre h proti 0, gre le napaka metode proti nič, neodstranljiva napaka pa raste prek vsake meje. Če h raste, pa raste napaka metode. Zato eksistira neka vrednost za h , pri kateri je ocena za napako minimalna.

Naslednji tabeli naj ilustrirata, kako se manjša natančnost odvodov pri numeričnem odvajanju. Druge odvode računamo po isti poti iz prvih odvodov.

i	x	e^x	$(e^x)'$	$(e^x)''$
0	0.00	1.0000		
1	0.01	1.0101	1.0100	
2	0.02	1.0202	1.0200	1.0000
3	0.03	1.0305	1.0300	
4	0.04	1.0408		
0	9.80	18034		
1	9.85	18958	18960	
2	9.90	19930	19940	20000
3	9.95	20952	20960	
4	10.00	22026		

Izračunane vrednosti v drugi tabeli so relativno bolj natančne kot vrednosti v prvi tabeli, ker je korak h večji.

Zanimiva je tudi naslednja tabela, ki kaže dejansko in ocenjeno napako pri računanju odvoda funkcije e^x pri $x = 1$ za razne h iz štirimestnih tabel. V oceni smo vzeli $M_3 = 3$ in ocenjene vrednosti zaokrožili navzgor:

h	napaka $\cdot 10^4$	ocena $\cdot 10^4$
0.01	17	51
0.02	8	27
0.03	17	22
0.04	4	21
0.05	17	23
0.06	17	27
0.07	24	32
0.08	29	39
0.09	39	47
0.10	47	55

Pri argumentu $x = 1$ je med navedenimi razmaki najugodnejši $h = 0.04$ tako pri dejanski kot pri ocenjeni napaki.

8. Numerično integriranje

Vzemimo, da iščemo določeni integral

$$I = \int_a^b f(x) dx,$$

kjer je funkcija $f(x)$ podana z ekvidistantno tabelo (x_i, y_i) , $i = 0, 1, \dots, n$ z razmakom h , ali pa je integrand analitično podan, toda ne poznamo nedoločenega integrala. V drugem primeru si mislimo, da smo funkcijo $f(x)$ sami tabelirali v izbranih točkah x_i .

Oglejmo si dve različni metodi za izpeljavo formul za numerično integriranje (kvadraturenih formul), ki jih že poznamo: trapezne in Simpsonove. Pri tem bomo tudi izpeljali izraze za napako obeh formul in omenili, kako praktično napake ocenimo.

a) Trapezna formula

Iščemo formulo tipa

$$\int_{x_0}^{x_n} f(x) dx = \sum_{k=0}^n A_k y_k + R_n,$$

ki jo dobimo tako, da funkcijo $f(x)$ na vsakem delnem intervalu (x_{k-1}, x_k) nadomestimo z linearno funkcijo skozi točki (x_{k-1}, y_{k-1}) in (x_k, y_k) , to je, z interpolacijskim polinomom prve stopnje. Zahtevajmo, naj bo funkcija $f(x)$ dvakrat zvezno odvedljiva! V prvem razdelku smo izpeljali zvezo:

$$f(x) = \frac{x - x_k}{x_{k-1} - x_k} y_{k-1} + \frac{x - x_{k-1}}{x_k - x_{k-1}} y_k - \frac{1}{2} f''(\xi) (x - x_{k-1})(x_k - x)$$

Če integriramo to enačbo od x_{k-1} do x_k in vpeljemo novo spremenljivko s z izrazom $x = x_{k-1} + hs$, $dx = hds$, dobimo

$$\int_{x_{k-1}}^{x_k} f(x) dx = h \int_0^1 ((1-s)y_{k-1} + sy_k) ds - \frac{1}{2} h^3 \int_0^1 f''(\xi) s(1-s) ds$$

Če upoštevamo, da je izraz $s(1-s)$ na intervalu $(0, 1)$ ves čas pozitiven, moremo pri zadnjem integralu uporabiti izrek o povprečni vrednosti. Ta pravi, da smemo integral produkta dveh funkcij zapisati v obliki

$$\int_a^b f(x) g(x) dx = f(\xi) \int_a^b g(x) dx,$$

kjer je ξ neko število med a in b , če sta le $f(x)$ in $g(x)$ zvezni in $g(x)$ nenegativna funkcija na intervalu $[a, b]$.

Torej smemo izraz $f''(\xi)$ postaviti pred integral kot $f''(\xi_k)$, kjer sedaj ξ_k pomeni neko novo točko med x_{k-1} in x_k . Od tod dobimo

$$\int_{x_{k-1}}^{x_k} f(x) dx = \frac{h}{2} (y_{k-1} + y_k) - \frac{h^3}{12} f''(\xi_k)$$

Če seštejemo te integrale od $k = 1$ do $k = n$, dobimo trapezno formulo

$$\int_{x_0}^{x_n} f(x) dx = h \left(\frac{1}{2} y_0 + y_1 + \dots + y_{n-1} + \frac{1}{2} y_n \right) + R_n,$$

kjer je napaka R_n

$$R_n = - \frac{h^3}{12} \sum_{k=1}^n f''(\xi_k)$$

Ker je drugi odvod zvezna funkcija, obstaja med x_0 in x_n taka točka ξ , da je

$$\frac{1}{n} \sum_{k=1}^n f''(\xi_k) = f''(\xi)$$

Aritmetična sredina števil je namreč vedno med najmanjšim in največjim številom, zvezna funkcija pa zavzame vse vrednosti med najmanjšo in največjo vrednostjo na nekem intervalu. Od tod sledi, da se napaka trapezne kvadrature formule izraža s formulo

$$R_n = - n \frac{h^3}{12} f''(\xi),$$

kjer je ξ neko število med x_0 in x_n . Če znamo oceniti drugi odvod integranda, moremo to formulo uporabiti za direktno oceno napake.

b) Simpsonova formula

Če funkcijo $f(x)$ nadomestimo s parabolo skozi 3 sosednje točke (z interpolacijskim polinomom druge stopnje), pridemo do Simpsonove formule:

$$\int_{x_{2k-2}}^{x_{2k}} f(x) dx \doteq \frac{h}{3} (y_{2k-2} + 4y_{2k-1} + y_{2k})$$

To formulo izpeljimo sedaj drugače, z metodo nedoločenih koeficientov! Nastavimo si formulo z nedoločenimi koeficienti, nato pa te koeficiente določimo tako, da je formula eksaktna za polinome čim višjih stopenj!

V našem primeru iščemo formulo oblike:

$$\int_{x_{2k-2}}^{x_{2k}} f(x) dx = ay_{2k-2} + by_{2k-1} + cy_{2k} + R$$

Za napako R predvidimo le to, da bo imela kot faktor tretji odvod integranda, saj bo mogoče števila a, b, c izbrati tako, da bo formula eksaktna za vse polinome druge stopnje. Naj bo torej

$$R = df'''(\xi_k),$$

kjer je ξ_k neko število med x_{2k-2} in x_{2k} . Če izberemo števila a, b, c tako, da bo formula eksaktna za $f(x) = 1$, $f(x) = x - x_{2k-1}$ in za $f(x) = (x - x_{2k-1})^2$, bo eksaktna tudi za vse polinome druge stopnje. Od tod dobimo 3 enačbe za 3 neznanke:

$$\begin{aligned} 2h &= a + b + c \\ 0 &= -ha + hc \\ \frac{2}{3}h^3 &= h^2a + h^2c \end{aligned}$$

Rešitev tega sistema je

$$a = c = \frac{h}{3}, \quad b = \frac{4h}{3}$$

Če postavimo v formulo za $f(x)$ še izraz $(x - x_{2k-1})^3$, bomo ugotovili, kolikšen je faktor d v napaki, saj je tedaj tretji odvod konstanta. Iz enačbe:

$$0 = -h^3 a + h^3 c + 6d$$

sledi $d = 0$. To seveda ne pomeni, da je formula eksaktna za vse funkcije, le da je eksaktna tudi — nekoliko nepričakovano — za vse polinome tretje stopnje. Torej bo napaka najbrž vsebovala četrti odvod integranda. Če postavimo

$$R = ef^{iv}(\xi_k),$$

dobimo iz enačbe z nedoločenimi koeficienti pri $f(x) = (x - x_{2k-1})^4$

$$\frac{2h^5}{5} = h^4a + h^4c + 24e$$

konstanto e :

$$e = -\frac{h^5}{90}$$

Torej smo izpeljali formulo

$$\int_{x_{2k-2}}^{x_{2k}} f(x) dx = \frac{h}{3} (y_{2k-2} + 4y_{2k-1} + y_{2k}) - \frac{h^5}{90} f^{iv}(\xi_k)$$

Naj bo sedaj $n = 2m$ sodo število; seštejmo zgornje enačbe za $k = 1$ do $k = m$. Tako dobimo Simpsonovo kvadraturno formulo:

$$\int_{x_0}^{x_{2m}} f(x) dx = \frac{h}{3} (y_0 + 4y_1 + 2y_2 + 4y_3 + \dots + 2y_{2m-2} + 4y_{2m-1} + y_{2m}) - m \frac{h^5}{90} f^{iv}(\xi)$$

kjer je obstoj števila ξ med x_0 in x_{2m} zagotovljen, če je funkcija $f(x)$ štirikrat zvezno odvedljiva na $[x_0, x_{2m}]$. Če želimo oceniti napako v Simpsonovi formuli, moramo poznati oceno za četrti odvod integranda.

c) Praktično ocenjevanje napake

Postavimo se na stališče, da je tisti odvod, ki nastopa v napaki kvadrature formule konstanta, da torej neznano število ξ izgine. Tedaj je mogoče napako izračunati, če izračunamo približno vrednost integrala pri dveh različnih korakih h .

Naj bo eksaktna vrednost integrala I , približna vrednost po kvadraturi formuli s korakom h naj bo I_h in napaka v formuli naj bo proporcionalna potenci h^{n+1} . Pri trapezni formuli je $n = 2$, pri Simpsonovi pa $n = 4$.

Najprej izračunamo I_{2h} , nato pa še I_h . Pri navedenih predpostavkah torej velja:

$$\begin{aligned} I_{2h} &= I + C(2h)^{n+1} \\ I_h &= I + Ch^{n+1} \end{aligned}$$

Dvojka v drugi formuli pride zaradi podvojitve števila delnih intervalov. Od tod izračunamo

$$2Ch^{n+1} = (I_{2h} - I_h)/(2^n - 1)$$

Torej se napaka vrednosti I_h izraža z razliko $I_{2h} - I_h$. To je izračunljiva količina in jo sprejmemo za približno vrednost napake, ki pa je dober približek za napako le, če se ustrezni odvod, za katerega smo vzeli, da je konstanten, na integracijskem intervalu le malo spreminja.

Na kratko pomislimo še na to, kakšna je ocena za neodstranljivo napako pri obeh naših kvadraturnih formulah. Izkaže se, da ni večja od širine integracijskega intervala, pomnožene z največjo napako v funkcijskih vrednostih. To je posledica ugodnega dejstva, da so koeficienti v formulah vsi pozitivni in da je na koncu ustrezna vsota pomnožena s h .

Vzemimo kot primer integral

$$\int_0^{1.2} \frac{dx}{1+x} = \ln 2.2 = 0.788457 \dots$$

Izračunajmo približno vrednost tega integrala po trapezni in Simpsonovi formuli pri $h = 0.1$ in ocenimo napako po formuli in po zgornjem nasvetu! Zapišimo najprej tabelirane vrednosti:

k	x_k	$1/(1+x_k)$
0	0.0	1.00000
1	0.1	0.90909
2	0.2	0.83333
3	0.3	0.76923
4	0.4	0.71429
5	0.5	0.66667
6	0.6	0.62500
7	0.7	0.58824
8	0.8	0.55556
9	0.9	0.52632
10	1.0	0.50000
11	1.1	0.47619
12	1.2	0.45455

Iz te formule hitro izračunamo po trapezni formuli:

$$I_{2h} = 0.79109, \quad I_h = 0.78912$$

$$(I_{2h} - I_h)/3 = 66 \cdot 10^{-5}, \quad R_{12} \leq 200 \cdot 10^{-5}$$

Dejanska napaka vrednosti I_h je $66.3 \cdot 10^{-5}$. Pri oceni R_{12} smo ocenili, da je drugi odvod funkcije povsod manjši ali kvečjemu enak 2. V tem primeru je napaka mnogo bližja praktični oceni.

Po Simpsonovi formuli pa dobimo:

$$I_{2h} = 0.78850, \quad I_h = 0.78846$$

$$(I_{2h} - I_h)/15 = 3 \cdot 10^{-6}, \quad R_{12} \leq 16 \cdot 10^{-6}$$

Vrednost I_h je na vseh 5 mest pravilna. Četrty odvod smo ocenili s 24. To vrednost četrty odvod zavzame pri $x = 0$, nekje med 0 in 1.2 je seveda manjši, zato je tudi tu R_{12} ocenjen navzgor.

Velika natančnost Simpsonove formule nas ne sme zapeljati v misel, da bi z njo računali tudi integrale, pri katerih že integrand sam ali kak nizek odvod ni zvezen. Takrat je približek za integral lahko samo slučajno dober, omenjeni način za oceno napake pa sploh nima nobenega opravičila.

9. Konstrukcija empiričnih formul

Denimo, da pridemo z meritvami pri nekem eksperimentu do tabele (x_i, y_i) , $i = 1, \dots, n$ za neko funkcijo! V praksi pogostokrat želimo dobiti preprosto formulo, po kateri bi se izračunane vrednosti malo razlikovale od merjenih in po kateri bi lahko računali vrednosti funkcije pri drugih argumentih in si s tem prihranili nadaljnje meritve. Ker so merjene vrednosti y_i navadno podvržene raznim (tudi slučajnim) napakam, nam ni do tega, da bi vrednosti naše iskane funkcije pri $x = x_i$ bile eksaktno enake $y = y_i$, pač pa, da bi ta funkcija merjene vrednosti v celoti dobro aproksimirala.

Ponavadi gre postopek pri konstrukciji empiričnih formul tako, da si merjenje vrednosti najprej grafično predočimo; iz slike skušamo uganiti obliko funkcije, ki ima podoben potek, nato pa določiti najugodnejše vrednosti parametrov, ki nastopajo v funkciji, glede na posebno izbran kriterij.

Kriterij, po katerem sodimo o prilaganju funkcije podatkom, je navadno princip najmanjših kvadratov.

Naj bo

$$y = g(x)$$

funkcija, ki po našem mnenju ustreza merjenim podatkom. Ta funkcija naj bo preprosta in naj vsebuje nekaj parametrov. Po principu najmanjših kvadratov so najugodnejše vrednosti parametrov tiste vrednosti, pri katerih doseže izraz

$$\sum_{i=1}^n (g(x_i) - y_i)^2$$

najmanjšo vrednost. To se pravi, da želimo, da bo vsota kvadratov odstopov izračunanih vrednosti od merjenih čim manjša. Če v funkciji $g(x)$ parametri nastopajo linearno, dobimo za optimalne vrednosti teh parametrov sistem linearnih enačb.

Najpreprostejši primer je, če merjene vrednosti ležijo približno na neki premici. Takrat je smiselno, če podatke aproksimiramo z linearno funkcijo. Za $g(x)$ vzamemo torej $a + bx$ in določimo a in b tako, da bo izraz

$$F(a, b) = \sum_{i=1}^n (a + bx_i - y_i)^2$$

minimalen. Z odvajanjem na a in b dobimo enačbi:

$$\begin{aligned} \frac{1}{2} \frac{\partial F}{\partial a} &= \sum_{i=1}^n (a + bx_i - y_i) = 0 \\ \frac{1}{2} \frac{\partial F}{\partial b} &= \sum_{i=1}^n (a + bx_i - y_i) x_i = 0 \end{aligned}$$

Od tod dobimo za a in b sistem dveh linearnih enačb

$$s_0 a + s_1 b = t_0$$

$$s_1 a + s_2 b = t_1$$

kjer je

$$s_k = \sum_{i=1}^n x_i^k, \quad k = 0, 1, 2; \quad t_k = \sum_{i=1}^n x_i^k y_i, \quad k = 0, 1$$

Da se pokazati, da ima ta sistem enolično določeno rešitev, če le niso vsi x_i med seboj enaki, in da pri tako določenih a in b ekstrem res nastopi.

Teže je določiti empirično formulo, če parametri v njej ne nastopajo linearno. Z odvajanjem vsote kvadratov odstopov na parametre dobimo potem nelinearni sistem enačb, ki ga navadno rešujemo s kako iterativno metodo. Eksistirajo tudi druge metode za nelinearno aproksimacijo.

Včasih si v posebnih primerih lahko pomagamo s kako transformacijo, s katero prevedemo funkcijo v obliko z linearno nastopajočimi parametri.

Oglejmo si nekaj primerov empiričnih formul z dvema parametroma! V vseh teh primerih moremo uvesti nove spremenljivke tako, da je v teh spremenljivkah funkcija linearna.

$y = ab^x$	$\log y = \log a + \log b x$
$y = ax^b$	$\log y = \log a + b \log x$
$y = 1/(a + bx)$	$1/y = a + bx$
$y = x/(a + bx)$	$x/y = a + bx$
$y = a + b/x$	$y = a + b(1/x)$

Če sumimo, da se funkcija obnaša npr. kot eksponentna funkcija ab^x , potem si grafično predočimo vrednosti $(x_i, \log y_i)$. Če te točke ležijo približno na neki premici, izračunamo koeficienta te linearne funkcije po metodi najmanjših kvadratov in dobimo od tod a in b . To seveda nista vrednosti, ki dasta minimum izrazu

$$\sum_{i=1}^n (ab^{x_i} - y_i)^2,$$

toda vseeno dobimo na ta način morda dovolj dobro ujemanje. Če je ujemanje preslabo, poskusimo s kako drugo funkcijo.

LITERATURA

F. Križanič: Aritmetika, algebra in analiza, I., II., III., IV. del, Ljubljana, 1970.

ELEMENTI PROGRAMSKEGA JEZIKA *ALGOL 60*

EGON ZAKRAJŠEK

MOS 68-01

Sestavek prikazuje na preprostih in poučnih primerih osnovne pojme programskega jezika *Algol 60*.

ELEMENTS OF THE PROGRAMMING LANGUAGE *ALGOL 60*

The fundamentals of the programming language *Algol 60* are explained by means of simple and useful examples.

Uvod

Namen tega sestavka je prikazati v kar se da skopi obliki, kaj je programski jezik *Algol 60*. Prostor ne dopušča, da bi lahko jezik razložili v celoti. Vsekakor pa je mogoče prikazati mnogo zanimivih podrobnosti. Pot, ki smo jo ubrali, je nekoliko nenavadna. *Algol 60* je opisan s primeri, saj je bil naš cilj, da jezik na razumljiv in zanimiv način opišemo, ne pa rigorozno definiramo. Taka pot ima še neko posebnost. Pri vsakem primeru smo najprej povedali, kaj želimo storiti, nato smo napisali program, ki mu sledi potreben komentar. Bravca naj zato ne moti, če bo v tekstu naletel na navidezno nerazumljiv program, saj mu sledi kar se da popoln opis posameznih prijemov.

Ideja računalnika

Z nekaj besedami je težko povedati, kaj je računalnik. Za silo velja naslednji opis.

Računalnik je naprava, ki zmore opravljati nekaj preprostih operacij. Računalnik ima *spomin*, kjer so zapisana števila, s katerimi računa. V tem spominu je zapisano tudi zaporedje *ukazov*, ki mu povedo, katere operacije in v kakšnem vrstnem redu naj jih izvršuje. Tako zaporedje ukazov imenujemo *program*.

Tipični ukazi, ki so lahko zapisani v spominu računalnika, so na primer:

- Prinesi število iz določenega dela spomina v *register A*;
- prištej število iz določenega dela spomina k številu, ki je zapisano v registru *A*;
- shrani število iz registra *A* v določeni del spomina!

»Del spomina«, o katerem govorimo, je določen takole: Spomin je razdeljen na posamezne celice. V vsaki celici je lahko zapisano po eno število, celice same pa so numerirane. Zgoraj navedeni ukazi se na primer glase

- prinesi število iz celice 24854 v register *A*

in podobno za ostale ukaze.

Če sedaj želimo, da nam bo računalnik opravil neko računsko delo, moramo sestaviti program. V ta namen se moramo odločiti, v katerih celicah bodo posamezna števila in posamezni ukazi. Tak način sestavljanja programov je silno zamuden. Zato se je že zelo zgodaj, sredi petdesetih let, pojavila težnja, da bi to delo poenostavili.

V ta namen si je treba izmisliti poseben jezik, v katerem bomo sestavljali programe. Tak jezik mora biti po eni strani kar se da razumljiv in bogat, da se more v njem čim lažje izražati, po drugi strani pa mora biti kar se da preprost. Sestaviti je namreč treba poseben program, *prevajalnik*, s katerim bo računalnik brez naše pomoči prevajal programe iz *izvornega jezika* v zaporedje ukazov, ki ga bo računalnik končno res izvrševal.

Takih programskih jezikov si lahko veliko izmislimo in veliko jih je tudi izmišljenih. V nadaljevanju si bomo ogledali enega od njih, *Algol 60*.

Programski jezik *Algol 60* je sestavila posebna komisija, ki sta jo imenovali *Gesellschaft für Angewandte Mathematik und Mechanik (GAMM)* in *Association for Computing Machinery (ACM)*. Komisija je svoje delo opravila v letih 1958–1962.

Elementi jezika *Algol 60*

V nobenem programskem jeziku nimamo več opraviti s številkami posameznih celic. Denimo, da želimo sešteti dve števili! V *Algolu 60* napišemo na primer

$$c := a + b$$

Tak *stavek* je treba razumeti takole: V računalnikovem spominu sta zapisani dve števili, ki ju mi imenujemo *a* in *b*. Pod *a* in *b* je pravzaprav treba razumeti številki dveh celic, v katerih sta ti dve števili shranjeni. Napisani stavek zahteva, naj računalnik ti dve števili sešteje in vsoto zapiše v spomin pod imenom *c*. Število, ki je bilo prej zapisano v celici *c*, je izgubljeno, na tem mestu je sedaj novo število, vsota števil *a* in *b*. Kadarkoli sedaj napišemo ime *c*, pomeni to zgoraj izračunano vsoto, seveda le toliko časa, dokler *spremenljivki c* ne *priredimo* nove vrednosti. V tem duhu lahko zapišemo celo zaporedje stavkov:

$$\begin{aligned} c &:= a + b \times x; & y &:= (a + b)/c; & c &:= (x - y) \uparrow 3; \\ u2c &:= (peter + lojze)/marjan \uparrow xxx; \\ w &:= ((a + b) \times (b + c)) \uparrow 4 \end{aligned}$$

Iz navedenih primerov vidimo naslednje:

- Za imena spremenljivk lahko izbiramo poljubne kombinacije črk in števil pri pogoju, da se ime začne s črko;
- za deljenje uporabljamo poševno črto, saj si ne moremo privoščiti, da bi formule pisali v več nadstropjih;
- iz istega razloga uporabljamo za potenciranje navpično puščico, saj eksponentov ne moremo pisati nad črto;
- na voljo nam je samo ena vrsta oklepajev in to okrogli oklepaji, zato pa jih lahko uporabljamo v poljubno globino. *Algol 60* pozna sicer še oglate oklepaje, a ti so rezervirani za nekaj drugega.

Za tvorjenje bolj kompliciranih izrazov nam je na voljo še zbirka standardnih funkcij, na primer

$$\begin{aligned} y &:= \sin(x + z) \times \cos(x \times w) \uparrow 2; \\ u &:= \ln(a) \times \exp(a + \arctan(y)); \\ w &:= \text{sqrt}(\text{abs}(m)); \\ hk &:= \text{entier}(p) - \text{sign}(st) \end{aligned}$$

Navedena imena pri tem pomenijo

<i>sin</i>	sinus kota, merjenega vadianih,
<i>cos</i>	cosinus kota, merjenega vadianih,

ln naravni logaritem,
exp eksponentna funkcija,
arctan glavna vrednost funkcije arctg, izražena v radianih,
sqrt nenegativni kvadratni koren,
abs absolutna vrednost,
entier celi del števila,
sign predznak števila, funkcija signum.

Vse te funkcije lahko uporabljamo brez predsodkov in kot argumente smemo pisati poljubno zamotane izraze.

Primer:

Izračunati želimo ploščino trikotnika po Heronovi formuli.

```

.....
s := (a + b + c)/2;
p := sqrt(s × (s - a) × (s - b) × (s - c));
.....

```

Pikice pred napisanima stavkoma ponazarjajo del programa, s katerim definiramo stranice *a*, *b* in *c*, pikice za njima pa del programa, kjer moramo povedati, kaj se bo z izračunano ploščino zgodilo. Vrednosti vseh treh stranic bi lahko definirali takole:

$$a := 17.3; \quad b := 16.82; \quad c := 0.187_{10} 2;$$

Ta primer nam obenem pokaže, kako pišemo necela števila. Namesto decimalne vejice uporabljamo decimalno piko, števila pa lahko pišemo tudi v pollogaritmični obliki (stranica *c*), kjer majhna desетка pomeni

»krat deset na«.

Namesto tega bi lahko napisali tudi

$$c := 18.7;$$

Tipičen primer, kaj bi želeli z izračunano ploščino, bi bila zahteva, naj računalnik to število zapiše na papir. V ta namen napišemo v programu:

$$\text{out real } (1, p)$$

out real je pri tem ime posebnega *podprograma*, ki zna pisati realna števila na papir. Prvi argument, *1*, je pri tem številka enote, kamor naj se rezultat napiše. Računalnik namreč zmore pisati rezultate na različne enote (tiskarski stroj, luknjalik kartice, teleprinter in podobno). Domenili se bomo, da nosi tiskarski stroj številko *1*. Ta dogovor velja za računalnik *IBM 1130*. Na drugih računalnikih imajo morda tiskarski stroji drugačne številke, pa tudi podprogram za pisanje se utegne drugače imenovati. V splošnem velja naslednje. *Algol 60* je kot jezik nedvoumno definiran brez posebnih podprogramov za pisanje rezultatov in čitanje podatkov. Sestavljalci so bili v začetku mnenja, naj se pri vsakem računalniku uredi čitanje podatkov in pisanje rezultatov po lastni presoji. Kasneje je več komisij predlagalo nekaj predlogov, kako naj bodo ti podprogrami urejeni, vendar se enotnosti ni dalo več doseči.

Sestavimo sedaj popoln program za računanje ploščine trikotnika po Heronovi formuli.

```
begin real a, b, c, s, p;  
    in real (0, a); in real (0, b); in real (0, c);  
    s := (a + b + c)/2;  
    p := sqrt (s × (s - a) × (s - b) × (s - c));  
    out real (1, p)  
end
```

Ob popolnem programu je na mestu brez dvoma še nekaj pojasnil.

Ves program je napisan kot zaporedje *simbolov*. Ti simboli so treh vrst: *črke*, *številke* in *specialni znaki*. Standardnih specialnih znakov kot so pika, vejica, podpičje in podobno, ni dovolj. Zato so si sestavljalci jezika izmislili še nove, in to na preprost način. Mastno odtisnjena beseda* pomeni nov specialni znak. Tako je na primer **begin** simbol jezika, ki ima popolnoma določen pomen, medtem ko *begin* pomeni ime, ki smo si ga mi izmislili, da nam pomeni na primer neko število.

Vsak program je napisan kot zaporedje stavkov, ki so *med seboj* ločeni s podpičjem, vse skupaj pa je vklenjeno med simbola **begin** in **end**. Oblikovno pišemo program tako, da je kar se da pregleden. Presledki med posameznimi simboli in pisanje v novo vrstico nimajo nikakršnega pomena.

Prvi stavek,

```
real a, b, c, s, p
```

pove naslednje: V programu bomo uporabljali navedena imena kot imena realnih količin. Računalnik more namreč računati s celimi in necelimi števili. Necela števila imenujemo *realna* števila. Navedena imena predstavljajo tako imena količin, ki lahko zavzamejo (skoraj) poljubne realne vrednosti. Za vsa imena, ki se v programu pojavijo, moramo na začetku povedati, kakšne količine predstavljajo. Takemu informativnemu stavku pravimo *deklaracija*.

V drugi vrsti smo trikrat uporabili podprogram *in real*. To je podoben podprogram kot *out real*. S tem podprogramom zmore računalnik prečitati eno realno število. Podprogram *in real* ima spet dva parametra. Prvi parameter pove, s katere enote naj računalnik število prečita. Domenili se bomo, da številka 0 pomeni čitalnik kartic. Drugi parameter pove, kam v spomin se prečitano število zapiše. V našem programu bo računalnik prečital tri števila, *a, b, c*, to je, vse tri stranice trikotnika.

Količine *a, b* in *c* imajo tako definirano vrednost in ploščino lahko izračunamo. Dobljeni rezultat ukažemo napisati na papir s podprogramom *out real*.

Primer 1.

Naj bodo znane vse tri stranice trikotnika. Izračunati želimo vse tri notranje kote, izražene v stopinjah, minutah in sekundah.

Uporabili bomo izrek o tangensih polovičnih kotov

$$\operatorname{tg} \frac{a}{2} = \sqrt{\frac{(s-b)(s-c)}{s(s-a)}}$$

* V rokopisu si mastnega tiska ne moremo privoščiti, zato tam take besede enostavno podčrtamo.

in podobno za ostale kote. Hitro lahko ugotovimo, da moramo velik del programa skoraj dobesedno trikrat prepisati, za vsak kot enkrat. Da bi se temu izognili, definiramo nekaj podprogramov, analogno podprogramoma *in real* in *out real*, seveda s pomembno razliko, da sta podprograma *in real* in *out real* že definirana, naše podprograme pa moramo definirati sami. Popoln program lahko sestavimo takole.

```

begin real a, b, c;
  integer a1, a2, a3, b1, b2, b3, c1, c2, c3, s1, s2, s3;

  procedure reduc (m, n); integer m, n;
  begin integer q; q := m ÷ 60;
    m := m - q × 60; n := n + q
  end;

  procedure print (a, b, c); integer a, b, c;
  begin out integer (1, a); out integer (1, b);
    out integer (1, c); sys act (1, 14, 1)
  end;

  procedure kot (a, b, c, st, mi, se); real a, b, c;
    integer st, mi, se;
  begin real s, al;
    s := (a + b + c)/2;
    al := 2 × arctan (sqrt ((s - b) × (s - c)/s/(s - a)));
    al := al × 180/3.14159265;
    st := entier (al); al := (al - entier (al)) × 60;
    mi := entier (al); al := (al - entier (al)) × 60;
    se := entier (al + 0.5);
    reduc (se, mi); reduc (mi, st);
    print (st, mi, se)
  end;

  in real (0, a); in real (0, b); in real (0, c);
  sys act (1, 15, 1);
  out real (1, a); out real (1, b); out real (1, c);
  sys act (1, 14, 1);
  kot (a, b, c, a1, a2, a3);
  kot (b, c, a, b1, b2, b3);
  kot (c, a, b, c1, c2, c3);
  s1 := a1 + b1 + c1; s2 := a2 + b2 + c2; s3 := a3 + b3 + c3;
  reduc (s3, s2); reduc (s2, s1);
  print (s1, s2, s3)
end

```

Opišimo ta program nekoliko podrobneje!

Najprej smo definirali tri realne količine, *a, b* in *c*. To bodo stranice našega trikotnika. Nadalje smo deklarirali dvanajst celoštevilčnih količin. Potrebujemo namreč tri notranje kote, vsakega izraženega s stopinjami, minutami in sekundami. To nam da devet celoštevilčnih količin. *a1* nam pomeni stopinje kota *a*, *b2* minute kota *β* in *c3* sekunde kota *γ*

ter podobno ostala imena. Na kraju želimo napraviti preizkus: vsota vseh notranjih kotov mora znašati 180 stopinj. Za to vsoto smo uporabili imena *s1*, *s2* in *s3*.

Med računanjem je treba predvideti, da se nam nabere več kot šestdeset sekund ali minut, ki jih moramo pretvoriti v minute oziroma v stopinje. To delo je treba opraviti večkrat. Da ne bi vsakokrat pisali ustreznega dela programa, deklariramo poseben podprogram, **procedure** *reduc*, ki opravlja to delo. *reduc* ima dva parametra, ki smo ju imenovali *m* in *n*. To sta dve celoštevilčni količini, kar smo vestno zapisali. Ti dve količini nam predstavljata sekunde in minute nekega kota. Če je sekund 60 ali več, naj ta podprogram število sekund zmanjša in ustrezno poveča število minut. V ta namen smo deklarirali pomožno celoštevilčno količino *q*. Najprej izračunamo

$$q := m \div 60$$

Znak $\div$ pomeni deljenje v okviru celih števil, torej celi del kvocienta. Toliko celih minut je v *m* sekundah. Nato ustrezno korigiramo število sekund in minut. Razume se, da lahko isti podprogram uporabimo tudi za pretvarjanje minut v stopinje.

Trojico števil — stopinje, minute in sekunde — moramo večkrat izpisovati. Zato deklariramo nov podprogram — *print*, ki nam trojico števil izpiše in poskrbi za premik tiskarskega stroja v novo vrstico. Podprogram *print* ima tri parametre, tri cela števila, ki jih je treba napisati. Spet smo vestno napisali, da so ti parametri celoštevilčne spremenljivke. Za izpisovanje posameznega števila uporabimo že definiran podprogram *out integer*, ki deluje tako kot *out real*, le da je namenjen za pisanje celih števil. Na kraju uporabimo še podprogram *sys act*: *sys act* je podprogram, ki opravlja vrsto najrazličnejših poslov ob izpisovanju in čitanju. Podprogram *sys act* ima tri parametre. Prvi pove, s katero enoto imamo opraviti. Razume se, da bo pri nas to enota številka 1. Drugi parameter pove vrsto dela, ki naj ga računalnik opravi. Operacija številka 14 pomeni premik tiskarskega stroja v novo vrsto. Tretji parameter v tem primeru pove, kolikokrat je treba valj premakniti v novo vrsto. Ukaz

sys act (1, 14, 1)

pomeni torej premik v novo vrsto na tiskarskem stroju in to enkrat.

Končno definiramo še tretji podprogram, ki se imenuje *kot*. Ta podprogram ima šest parametrov. Prvi trije so realna števila in pomenijo stranice trikotnika, drugi trije so tri cela števila in pomenijo število stopinj, minut in sekund za en izračunani kot. Podprogram naj ta tri števila izračuna in mimogrede še napiše na papir. Če pri uporabi tega podprograma ciklično menjamo stranice, dobimo vse tri kote.

Za izračun kota najprej izračunamo polovičen obseg *s*, nato pa s pomočjo funkcije *arctan* še kot sam. Seveda je ta kot izražen v radianih, zato ga najprej pretvorimo v stopinje, nato pa postopoma v cele stopinje, minute in sekunde. Ker sekunde na kraju zaokrožimo, se lahko zgodi, da dobimo 60 sekund, zato uporabimo še podprogram *reduc*, s katerim se najprej znebimo odvečnih sekund, nato pa še odvečnih minut. Dobljena tri števila izpišemo s podprogramom *print*.

Kar nam je dela še ostalo, je v glavnem uporaba že definiranih podprogramov. S podprogramom *in real* prečitamo vse tri stranice in jih s podprogramom *out real* takoj spet napišemo na papir. Pred tem pa še s podprogramom *sys act* prestavimo papir na začetek nove strani. To je namreč pomen operacije številka 15. Nadalje s pomočjo podprograma *kot* izračunamo vse tri kote tako, da ciklično menjamo stranice. Kot smo omenili, podprogram *kot* kote tudi izpiše. Končno vse tri kote še seštejemo. Ker lahko pri tem dobimo več kot 59 sekund ali minut, s podprogramom *reduc* reduciramo odvečne sekunde v minute in odvečne minute v stopinje.

Primer 2.

Izračunati je treba prostornino telesa, ki nastane, če se polovica krožnega odseka zavrti okrog tetive. Za znane vrednosti r in h (glej skico!) je treba sestaviti primerno tabelo. Kratek račun pokaže

$$V = \pi \left\{ R^2 h - \frac{h^3}{3} - 2R^2 (R - r) \times \arctg \frac{r}{h} \right\}$$

kjer je V iskana prostornina in

$$R = \frac{r^2 + h^2}{2r}$$

Iskana prostornina je odvisna od dveh količin, r in h . Da bi skrčili obseg potrebne tabele, napravimo naslednje: Naj bo V_0 prostornina stožca s polmerom osnovne ploskve r in višino h , torej $V_0 = \pi r^2 h / 3$

Označimo še $x = r/h$

Po krajšem računu dobimo:

$$\frac{V}{V_0} = \frac{1}{4x^4} \left(3(1 + x^2)^2 \left\{ 1 - \frac{1 - x^2}{x} \arctg x \right\} - 4x^2 \right)$$

Napisana formula je za majhne vrednosti x neuporabna. Očitno je, da ostane razmerje V/V_0 omejeno, ko teži x proti nič, čeprav bi po napisani formuli sodili, da to razmerje raste kot $1/x^4$. To pomeni, da se pri majhnih vrednostih spremenljivke x vrednosti v oklepajih skoraj popolnoma uničijo med seboj, in to tako, da ostane končen ulomek omejen. Pri računanju na končno število decimalok pa to seveda pomeni veliko numerično napako.

Da bi se tej nerodnosti izognili, definiramo:

$$\arctg x = x - \frac{x^3}{3} + x^5 g(x)$$

$$g(x) = \frac{1}{5} - \frac{x^2}{7} + \frac{x^4}{9} - \frac{x^6}{11} + \dots$$

Po krajšem računu dobimo

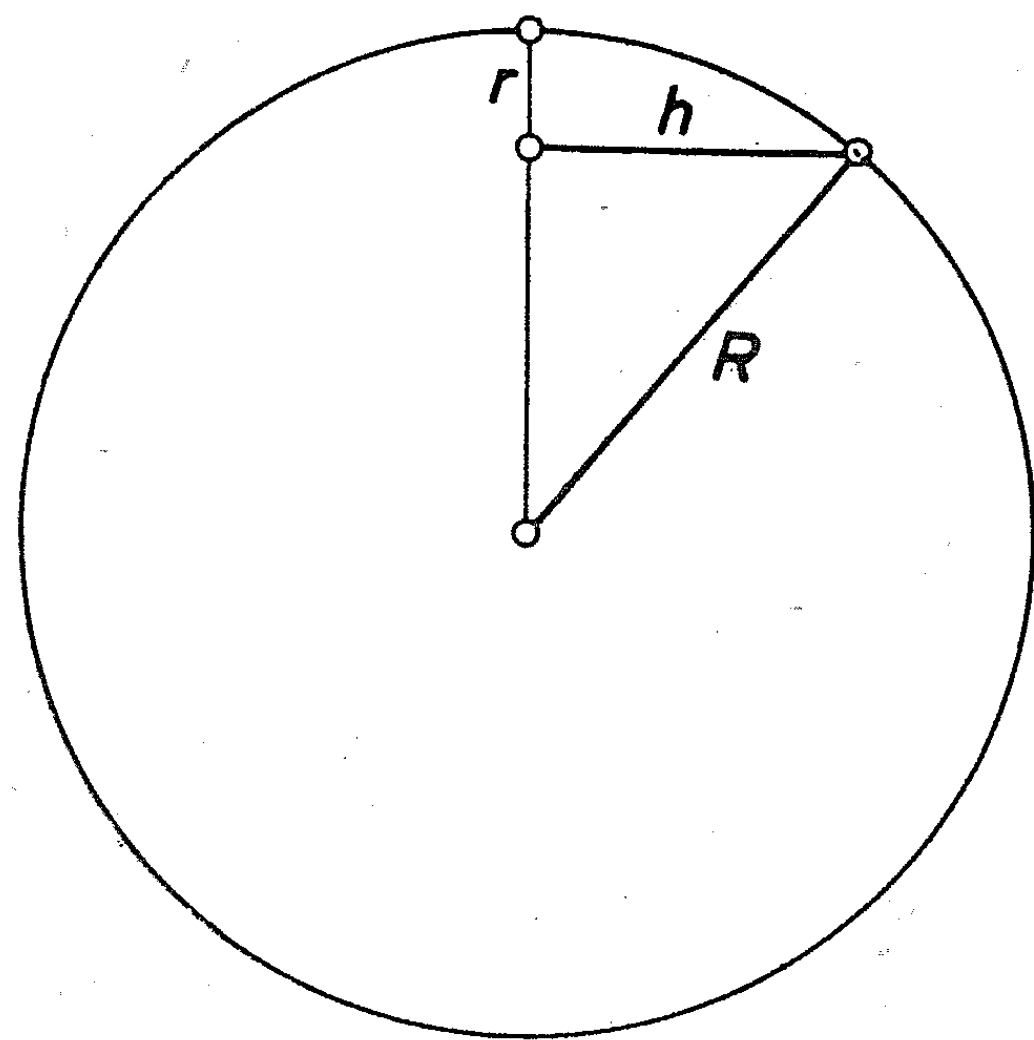
$$\frac{V}{V_0} = \frac{1}{4} (7 + 2x^2 - x^4 - 3(1 - x^2)(1 + x^2)^2 g(x))$$

Vse numerične nerodnosti so sedaj skrite v funkciji $g(x)$. Poskrbeti moramo le, da bomo to vrednost natančno izračunali. Vrednost te funkcije pa lahko računamo na dva načina:

$$g(x) = \frac{1}{x^5} \left(\arctg x - x + \frac{x^3}{3} \right) = \frac{1}{5} - \frac{x^2}{7} + \frac{x^4}{9} - \dots$$

Za vrednosti x , večje od 0.25, bomo uporabili prvo formulo, za vrednosti, ki so manjše od 0.25, pa drugo formulo. Preprost premislek pokaže, da so zanimive samo tiste vrednosti spremenljivke x , ki leže med nič in ena. Zato bomo sestavili tabelo razmerij prostornin za vrednosti

$$x = 0(0.05)1.0$$



```

begin real x;

  real procedure g(x); real x;
  if x > 0.25 then g := (arctan(x) - x + x ↑ 3/3)/x ↑ 5 else
    begin real s, a, z; integer k;
      s := 0; k := 5; z := x ↑ 2; a := 1;
      ponovi: s := s + a/k; a := -a × z; k := k + 2;
      if abs(a) > 10 -6 then go to ponovi;
      g := s
    end;

  sys act (1, 15, 1);
  for x := 0 step 0.05 until 1.01 do
    begin real y, z; z := x ↑ 2;
      y := 0.25 × (7 + 2 × z - z ↑ 2 -
        3 × (1 - z) × (1 + z) ↑ 2 × g(x));
      out real (1, x); out real (1, y);
      sys act (1, 14, 1)
    end
  end

```

V tem programu je spet cela vrsta novosti.

Najprej smo definirali funkcijo $g(x)$. Postopek je podoben definiciji podprogramov, vendar je med obema nekaj pomembnih razlik. Podprograme kličemo kot samostojne stavke, funkcije uporabljamo v aritmetičnih izrazih. Pri podprogramih ime podprograma ne predstavlja nikakršne vrednosti, pri funkcijah pa nam ime funkcije predstavlja funkcijsko vrednost. Ta vrednost je nekega tipa, v našem primeru realno število. Zato smo deklarirali g kot **real procedure**. Funkcija g ima en realen argument, kar smo spet vestno zapisali.

Funkcijska vrednost se v našem primeru izračuna na precej zamotan način. Najprej moramo ločiti, ali je argument večji ali manjši od 0.25. Če je vrednost argumenta večja od 0.25, se funkcijska vrednost preprosto izračuna, sicer pa (**else!**) je potreben daljši postopek, ki smo ga zapisali kot zaporedje stavkov, vključeno med simbola **begin** in **end**. V drugem primeru najprej definiramo nekaj pomožnih količin. Realna spremenljivka a nam bo predstavljala števec vsakega člena vrste skupaj z ustreznim predznakom. Ta števec računamo rekurzivno. V začetku je enak 1, v vsakem nadaljnjem koraku pa ga množimo z $-x^2$. Da ne bi vsakokrat pisali izraza x^2 , smo definirali pomožno količino z , ki ima ravno to vrednost. Vsoto neskončne vrste akumuliramo pod imenom s . V začetku je s enak nič, nato vsakokrat prištejemo po en člen. Imenovalce posameznih členov smo imenovali k . Spremenljivka k dobi v začetku vrednost 5, nato jo vsakokrat povečamo za 2.

Postopek računanja in prištevanja posameznih členov moramo ponavljati toliko časa, dokler členi ne postanejo zanemarljivo majhni. Kot kriterij si izberemo naslednje: členi niso zanemarljivo majhni, če je števec člena po absolutni vrednosti večji od 10^{-6} . Ponovno uporabimo pogojni stavek: če je absolutna vrednost števca večja od 10^{-6} , naj se program nadaljuje (**go to!**) pri stavku, ki nosi oznako *ponovi*. To je tisti stavek, kjer se k delni vsoti prišteje nov člen. Ime stavka *ponovi* je od stavka samega ločeno z dvopičjem. Šele ko je absolutna vrednost števca dovolj majhna, se proces prekine in funkcijska vrednost postane enaka nabrani vsoti s .

Z definicijo funkcije g smo rešili težji del naloge. Postopek je sedaj silno preprost. Najprej ukažemo, naj se papir na tiskarskem stroju premakne na začetek nove strani (podprogram *sys act*). Nato želimo del programa ponoviti za vse potrebne vrednosti spremen-

ljivke x . V ta namen uporabimo **for** stavek: spremenljivka x naj se avtomatično spreminja od začetne vrednosti 0 s korakom 0.05 , dokler ne doseže vrednosti 1.01 . Namenoma smo zapisali 1.01 namesto 1.0 . Zaradi numeričnih napak, ki so pri računanju z realnimi števili neizogibne, se lahko zgodi, da bo računalnik namesto vrednosti 0.95 naračunal vrednost 0.950000001 . Na natančnost rezultatov to odstopanje kaj malo vpliva, vendar bi bila lahko naslednja vrednost spremenljivke x na primer 1.000000001 . Ta vrednost pa je večja, čeprav komaj opazno, od vrednosti 1.0 in računalnik bi v takem primeru delo prezgodaj prekinil. Vrednost 1.01 je kot zgornja meja izbrana zelo premišljeno: numerične napake se ne more nabrati toliko, da bi namesto vrednosti 1.0 dobili več kot 1.01 , po drugi strani pa je vrednost 1.01 dosti manjša od naslednje vrednosti 1.05 , tako da ni nevarnosti, da bi računalnik izračunal eno vrednost preveč.

Medtem ko se spremenljivka x avtomatično spreminja, se vsakokrat izvrši stavek, ki neposredno sledi simbolu **do**. Ker želimo, da se vsakokrat izvrši več kot en stavek, vse potrebno zaporedje stavkov vklenemo med simbola **begin** in **end**. Tako zaporedje stavkov med simboloma **begin** in **end** velja za en sam stavek in se torej pri vsaki vrednosti spremenljivke x izvrši v celoti. Isto konstrukcijo smo uporabili že pri definiciji funkcije g . Pri *if-else* stavku lahko za vsako alternativo napišemo po en stavek. Pri prvi alternativni, če je vrednost spremenljivke x večja od 0.25 , nam en stavek zadošča. V drugi alternativni potrebujemo za opis več stavkov. Zato smo potrebno zaporedje stavkov vklenili med simbola **begin** in **end**.

Kaj je treba storiti pri vsaki vrednosti spremenljivke x , je očitno. Najprej definiramo dve pomožni količini, z in y . Spremenljivka z bo imela vrednost x^2 , da ne bo treba vsakokrat pisati tega izraza, spremenljivka y pa bo imela vrednost iskanega razmerja prostornin. Ker je funkcija g že definirana, je račun skrajno preprost. Obe vrednosti x in y napišemo še na papir in s podprogramom *sys act* poskrbimo za novo vrstico.

Primer 3.

Iz fizike vemo, da je jakost svetlobnega toka na enoto valovne dolžine pri sevanju črnega telesa podana z izrazom

$$q = \frac{2\pi hc^2}{\lambda^5 \left\{ \exp \left(\frac{hc}{\lambda kT} \right) - 1 \right\}}$$

kjer je

- h Planckova konstanta
- c svetlobna hitrost
- λ valovna dolžina
- k Boltzmanova konstanta
- T absolutna temperatura

Zanima nas, pri kateri valovni dolžini je q največji.

Najprej uvedemo

$$x = hc/\lambda kT$$

in dobimo

$$q = a x^5 / (e^x - 1)$$

$$a = 2\pi k^5 T^5 / h^4 c^3$$

Količina a je od valovne dolžine λ neodvisna in zadošča, da poiščemo maksimum funkcije f

$$f(x) = \frac{x^5}{e^x - 1}$$

Če postavimo, da je odvod funkcije f enak nič, dobimo naslednjo enačbo:

$$x = 5(1 - e^{-x})$$

To transcendentno enačbo lahko rešimo z iteriranjem. Zaporedne približke tvorimo po formuli

$$x_{k+1} = 5(1 - \exp(-x_k))$$

Če vzamemo za začetni približek vrednost $x_0 = 5$, se lahko prepričamo, da bodo zaporedni približki hitro monotonno padali k rešitvi enačbe. Tako si lahko sestavimo naslednji program:

```
begin real h, c, la, k, t, x, xx;  
  x := 5;  
  for xx := 5 × (1 - exp(-x)) while x > xx do x := xx;  
  
  in real (0, h); in real (0, c);  
  in real (0, k); in real (0, t);  
  
  out string (1, 'h = ');    out real (1, h);    sys act (1, 14, 1);  
  out string (1, 'c = ');    out real (1, c);    sys act (1, 14, 1);  
  out string (1, 'k = ');    out real (1, k);    sys act (1, 14, 1);  
  out string (1, 't = ');    out real (1, t);    sys act (1, 14, 1);  
  la := h × c / (k × t × x);  
  out string (1, 'l = ');    out real (1, la);    sys act (1, 14, 1)  
  
end
```

Napisani program je zelo preprost. Ker lahko enačbo rešimo, ne da bi poznali vrednosti fizikalnih parametrov, to tudi storimo. Poudariti velja, kako preprosto se da zapisati reševanje take enačbe v *Algolu 60*. Uporabili smo naslednje oznake: x nam pomeni stari približek, xx pa novega. Kot smo obljubili, si za začetni približek izberemo vrednost 5. Nato dodeljujemo vrednost

$$5 \times (1 - \exp(-x))$$

spremenljivki xx toliko časa, dokler je tako dobljeno zaporedje res monotonno padajoče, to je, dokler je novi približek xx res strogo manjši od starega. Teoretično se kajpada približki manjšajo v nedogled, čedalje počasneje. Ker pa računalnik računa samo s končnim številom decimalk, se prej ali slej zgodi, da se zaporedni približki začno ponavljati ali pa celo oscilirati okrog neke vrednosti. V enem in drugem primeru ne padajo več monotonno in s tem pogoj, ki smo ga zapisali pod **while**, ni več izpolnjen in iteriranje se prekine. Razume se, da moramo vsakokrat še stari približek x nadomestiti z novim, to pa smo zapisali kot poseben stavek za simbolom **do**.

Preostanek programa je preprost.

Najprej s podprogramom *in real* prečitamo vse potrebne podatke, nakar jih takoj zapišemo na papir. Izpisane podatke smo opremili še s tekstom. To delo nam opravlja podprogram *out string*. Prvi parameter tega podprograma je številka enote, kamor želimo tekst napisati, drugi parameter pa je ustrezen tekst. Ta tekst mora biti vklenjen med posebne narekovaje, kot lahko vidimo zapisano v programu. Za vsak podatek najprej izpišemo ustrezen tekst, nato vrednost podatka in končno poskrbimo še za novo vrsto. Na koncu izračunamo še valovno dolžino, ki jo izpišemo na isti način.

Primer 4.

Želimo izračunati, kako visoka mora biti navpična palica, da se pod lastno težo ukloni.

Iz mehanike vemo, da velja za upogib navpične palice pod lastno težo naslednja diferencialna enačba:

$$\frac{d^2}{dx^2} EI \frac{d^2 y}{dx^2} + \frac{d}{dx} gwx \frac{dy}{dx} = 0$$

kjer je

E modul elastičnosti

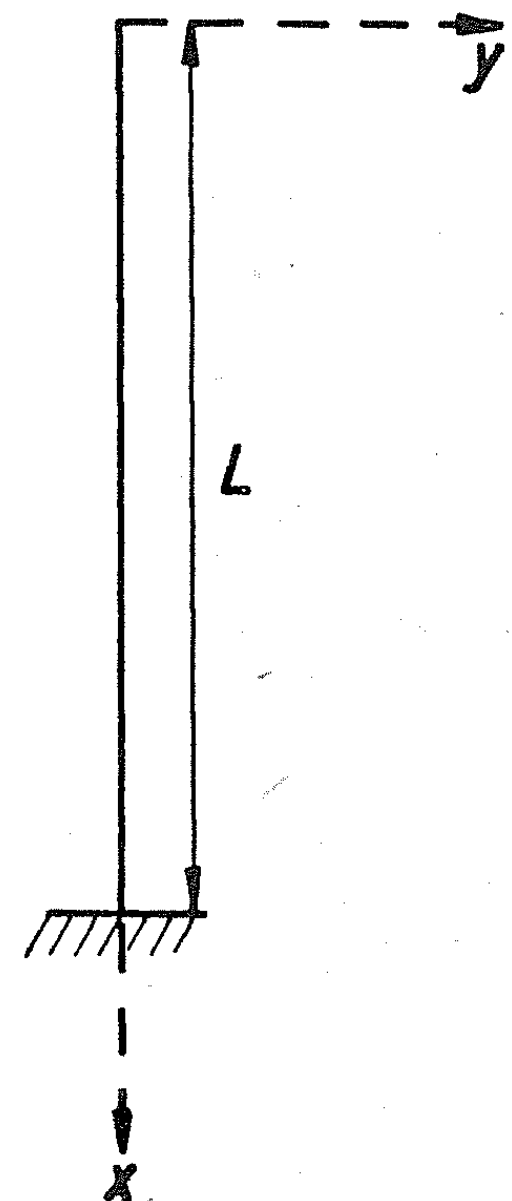
I vztrajnostni moment preseka

g pospešek prostega pada

w linearna gostota

y odmik palice od navpične lege

x koordinata, merjena vzdolž palice



Če je palica na spodnjem koncu tego vpeta, na zgornjem koncu pa prosta, sodijo k diferencialni enačbi še naslednji robni pogoji:

$$y(L) = y'(L) = y''(0) = y'''(0) = 0$$

kjer je L dolžina palice. To dolžino je treba določiti tako, da bo imela naloga netrivialno rešitev.

Z uvedbo novih spremenljivk

$$x = L\xi, \quad \frac{dy}{dx} = u(\xi)$$

dobimo

$$\frac{d^3 u}{d\xi^3} + 3\beta \frac{d}{d\xi} (\xi u) = 0$$

$$u(1) = u'(0) = u''(0) = 0$$

kjer je

$$3\beta = \frac{L^3 gw}{EI}$$

Rešitev iščemo v obliki vrste

$$u(\xi) = \sum_{k=0}^{\infty} c_k \xi^k$$

Iz primerjave koeficientov pri istih potencah spremenljivke ξ dobimo

$$c_{k+3} = -\frac{3\beta}{(k+2)(k+3)} c_k, \quad k = 0, 1, 2, \dots$$

kjer so koeficienti c_0 , c_1 in c_2 poljubni. Robna pogoja v točki $\xi = 0$ povesta, da velja

$$c_1 = c_2 = 0$$

Tako so od nič različni samo koeficienti

$$c_{3k}, \quad k = 0, 1, 2, \dots$$

Če uvedemo

$$c_{3k} = \beta^k b_k$$

sledi

$$u(\xi) = \sum_{k=0}^{\infty} b_k \beta^k \xi^{3k}$$

kjer je

$$b_{k+1} = -\frac{b_k}{(k+1)(3k+2)}, \quad k = 0, 1, 2, \dots$$

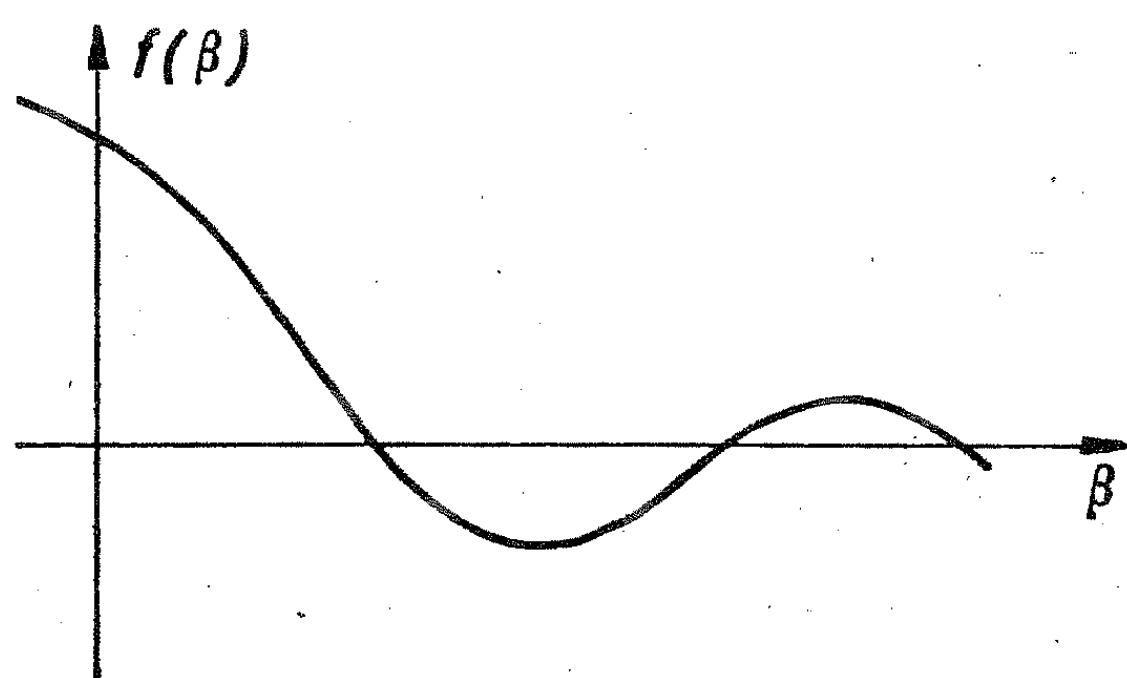
Koeficient b_0 si lahko izberemo poljuben, na primer

$$b_0 = 1$$

Preostali robni pogoj nam da

$$u(1) = \sum_{k=0}^{\infty} b_k \beta^k = 0$$

To je transcendentna enačba, katere najmanjši pozitivni koren moramo poiskati. Označimo



$$f(\beta) = \sum_{k=0}^{\infty} b_k \beta^k$$

$$f'(\beta) = \sum_{k=0}^{\infty} (k+1) b_{k+1} \beta^k = \sum_{k=0}^{\infty} d_k \beta^k$$

Velja

$$d_0 = -1/2$$

$$d_{k+1} = -\frac{d_k}{(k+1)(3k+5)}, \quad k = 0, 1, 2, \dots$$

Koren enačbe

$$f(\beta) = 0$$

bomo iskali po tangentni metodi

$$\beta_{k+1} = \beta_k - \frac{f(\beta_k)}{f'(\beta_k)}, \quad \beta_0 = 0$$

Skrbna analiza pokaže, da ima funkcija obliko, prikazano na sliki. Da ne bi pomotoma izračunali napačnega korena, bomo v program vgradili pogoj, razviden iz slike

$$f'(\beta) < 0$$

Tako lahko sestavimo naslednji program.

```

begin real e, i, w, g, l, beta, f, ff, db;
  integer j;

  procedure vrsta (be, f, ff); real be, f, ff;
  begin real c, cc; integer k;
    f := ff := 0; c := 1; cc := -0.5;
    k := 0;
  nov : f := f + c; ff := ff + cc;
    c := -c × be/(k+1)/(3 × k + 2);
    cc := -cc × be/(k+1)/(3 × k + 5);
    k := k + 1;
    if abs(c) > 10-6 then go to nov
  end;

```

```

    in real (0, e); in real (0, i); in real (0, w);
    in real (0, g); beta := 0; j := 0;
nov : vrsta (beta, f, ff);
    if ff ≥ 0 then
        begin db := db/2; beta := beta + db;
            go to nov
        end;
    db := f/ff; j := j + 1; beta := beta - db;
    if j > 50 then out string (1, 'ne konvergira') else
        begin if abs (db) >  $10^{-6}$  then go to nov;
            l := (3 × beta × e × i/g/w) ↑ (1/3);
            out real (1, e); out real (1, i);
            out real (1, w); out real (1, g);
            out real (1, l); sys act (1, 14, 1)
        end
    end
end

```

V tem programu najprej najdemo podprogram *vrsta*, ki nam izračuna vrednosti obeh vrst *f* in *ff* pri danem argumentu *be*. Z imenoma *c* in *cc* smo označili splošni člen vsake vrste. Člene seštevamo toliko časa, dokler so členi prve vrste po absolutni vrednosti večji od 10^{-6} .

Kot posebnost velja omeniti naslednje: Iteracijske korake smo to pot šteli. Če iteracijski proces v petdesetih korakih ni končan, se nadaljnje računanje prekine in računalnik izpiše tekst »ne konvergira«. Poleg tega pazimo na to, da ne bi med iteriranjem dobili nenegativne vrednosti odvoda. Če se to zgodi, takoj razpolovimo prejšnji popravek *db* in ponovno izračunamo vrednosti obeh vrst. Nevarnost, da bi izračunali napačen koren ali pa da proces sploh ne bi konvergiral, je močno zmanjšana, nikakor pa ne odstranjena.

Zaključek

Primeri, ki smo jih obravnavali, nikakor ne pokrivajo polnega jezika *Algol 60*. *Algol 60* skriva v sebi še celo vrsto možnosti, pa tudi uporabljenih prijemov žal nismo mogli razložiti z vsemi podrobnostmi.

Bravcu, ki ga ta snov dovolj zanima, priporočam, naj si podrobneje ogleda katero od spodaj navedenih del.

Jezik *Algol 60* je uradno definiran s poročilom

Backus J. W., Bauer F. L., Green J., Katz C., McCarthy J., Naur P., Perlis A. J., Rutishauser H., Samelson K., Vauquois B., Wegstein J. H., van Wijngaarden A., Woodger M., Revised Report on the Algorithmic Language ALGOL 60, Edited by P. Naur, International Federation for Information Processing, 1962.

Preostala navedena dela opisujejo jezik *Algol 60*.

Лавров С. С. Универсальный язык программирования, Москва, 1964

Rutishauser H. Description of Algol 60, Berlin 1967

Wooldridge R., Ractliffe J. F. An Introduction to Algol Programming, London 1965.

Herschel R. Anleitung zum praktischen Gebrauch von Algol 60, München, 1969.

McCracken D. D. A Guide to Algol Programming, New York, 1962.

Zakrajšek E. Programiranje v Algolu, Ljubljana, 1965.

Vsa navedena dela ima matematična knjižnica v Ljubljani.

NUMEROLOGIJA?

ANTON SUHADOLC

MOS 65-01

Pri računanju z računalnikom, pa tudi ročno, navadno ne računamo z vsemi realnimi števili, ampak le z njihovimi približki, ki imajo končno mnogo decimalk. V računalniku so realna števila upodobljena z neko končno množico takih števil. Narava te množice ni dobro proučena. Ogledali si bomo nekaj problemov v tej zvezi. Vsi imajo skupno lastnost: odgovor nanje ni tak, kot se nam na prvi pogled vsiljuje.

NUMEROLOGY?

In numerical calculations only real numbers with finitely many digits are used. In computers real numbers are approximated by a finite set of such numbers. The nature of such sets are not yet well-understood. In the following we shall consider some questions connected to such sets. All questions have a common property: the answer to them is not the expected, »obvious« answer.

1. Porazdelitev prve decimalke. Ogledali si bomo tale problem. Vzemimo neko množico števil, zapisanih npr. v dekadичnem sistemu. Decimalno vejico si odmislimo. Za prvo številko danega števila bomo proglasili prvo od nič različno številko. Mislimo si, da z računalnikom rešujemo kak obširnejši računski problem. Pri računanju nastopa mnogo realnih števil. Popolnoma naravna se zdi predpostavka, da je v taki množici števil porazdelitev prve decimalke približno enakomerna: približno enaki deli števil se začnejo s številko 1, 2, ..., 9. Obširni poskusi kažejo, da to ni res!

Napravili so naslednje statistične poskuse. Sestavili so med drugim tabelo površin, ki jih napaja 335 rek; tabelo hišnih števil večjega števila znanih osebnosti; tabelo molekularnih tež večjega števila organskih spojin in druge. Pri vseh poskusih se je izkazalo, da porazdelitev prve številke ni enakomerna, ampak logaritmična: del števil, ki se začne s številom manjšim ali enakim p , ($p = 1, 2, \dots, 9$), je približno enak $\log(p + 1)$. Torej se začne z enojko približno $\log 2 \doteq 0,30$ vseh števil, z dvojko približno $\log 3 - \log 2 \doteq 0,17$ vseh števil, z devetko pa le $\log 9 - \log 8 \doteq 0,05$ vseh števil!

Podobne statistične izračune so napravili tudi s števili, ki nastopajo pri daljšem numeričnem računu v računalniku, in pri tabeli prvih nekaj praštevil. Rezultati so bili vselej približno isti: prva decimalka je porazdeljena logaritmično.

V članku [1], po katerem povzemamo obravnavano snov, so omenjeni drugi članki, v katerih skušajo avtorji razložiti opazovani pojav. Eden močnih argumentov, ki pojasnjuje logaritmično porazdelitev prve decimalke, je tale: Površine, ki jih napajajo reke, izrazimo z dvema merskima enotama, npr. s kvadratnimi čevlji in kvadratnimi metri. Pokazati se da: če je porazdelitev prve decimalke pri uporabi ene enote enakomerna, v drugi ni. Če pa je porazdelitev prve decimalke pri eni enoti logaritmična, je pri drugi tudi! Podoben argument je tale: če je porazdelitev prve decimalke v neki množici števil logaritmična, je v množici števil, recipročnih k številom prve množice, porazdelitev prve decimalke tudi logaritmična.

Zanimivo je, da tudi druga številka v številih dane množice ni enakomerno porazdeljena. Obširni statistični poskusi kažejo, da nastopa ničla kot druga številka števil v približno 12% primerov, devetica pa le še v približno 8,5% primerov. Za nadaljnja mesta pa poskus ne pokaže, da porazdelitev ni enakomerna.

Posledice porazdelitve prve številke se poznajo celo pri konstrukciji računalnikov.

2. Število mest in natančnost. Predpostavimo, da imamo v računalniku upodobljena realna števila s števili glede na osnovo b ($b \geq 2$) z n ($n \geq 1$) mesti! To množico realnih števil označimo z $S(b, n)$:

$$S(b, n) = \{x : x \text{ ali } -x \text{ je oblike } c_1 b^k + c_2 b^{k-1} + \dots + c_n b^{k-n+1}, \\ k \in \mathbb{Z}, c_i \in \mathbb{Z}^+, c_1 \neq 0, 0 \leq c_i \leq b-1\}$$

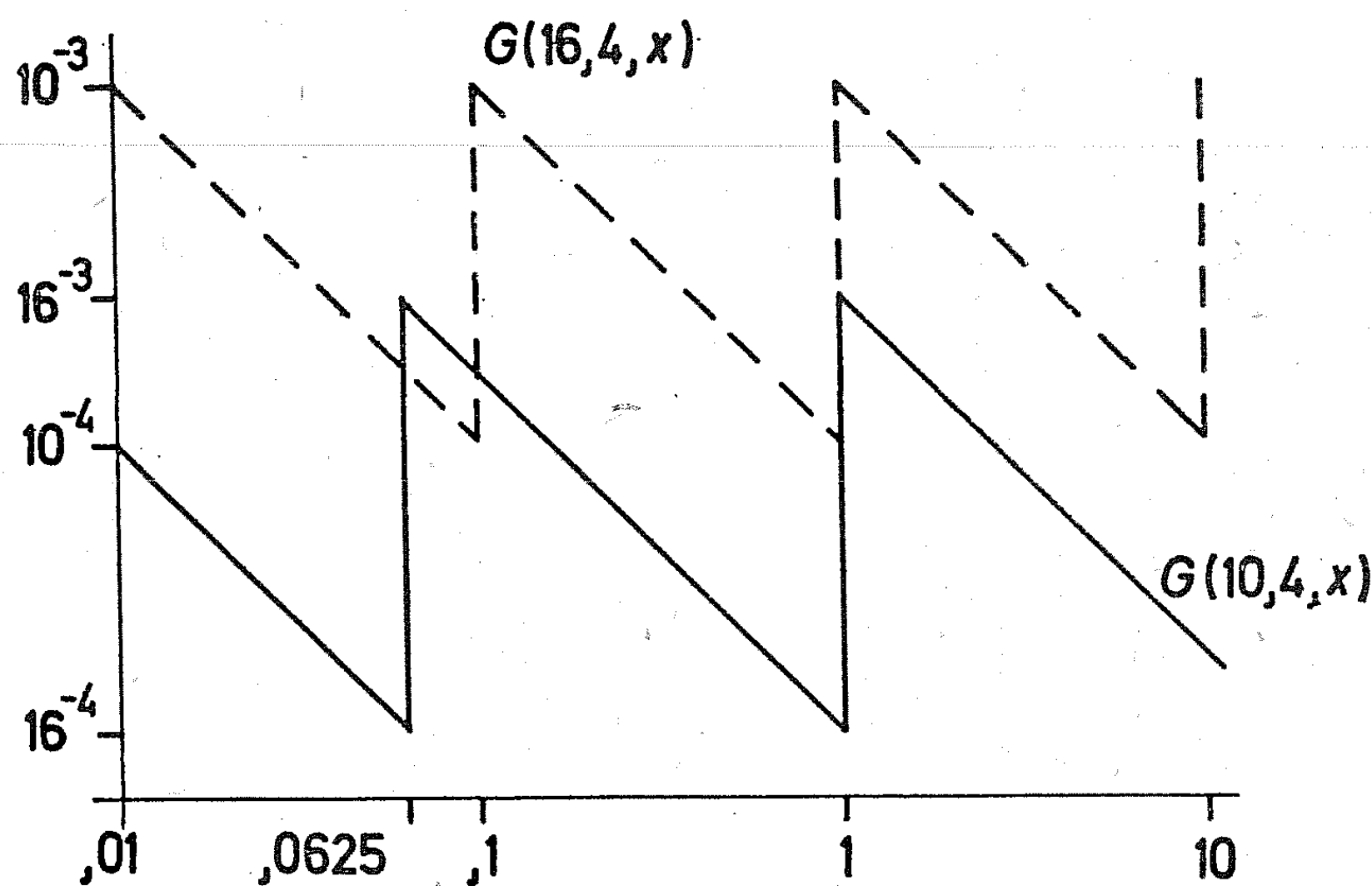
Relativna napaka, ki jo zagrešimo, ko dano realno število aproksimiramo s števili množice $S(b, n)$, ni za vsa realna števila ista. Prav gotovo je napaka manjša, če pri isti osnovi b uporabimo za aproksimacijo števil $n+1$ mest, namesto n mest. Število x smo torej upodobili s števili iz množice $S(b, n+1)$, namesto s števili množice $S(b, n)$. Bolj zamotana pa je situacija, če primerjamo napaki pri aproksimaciji števila x s števili iz množic $S(b, n)$ in $S(a, m)$, $a \neq b$.

Za merilo relativne napake si izberemo funkcijo $G(b, n, x)$, definirano takole: Pri danem številu $x > 0$ naj pomeni c najbližje število množice $S(b, n)$, ki je manjše ali enako številu x : $c = \max\{y : y \in S(b, n), y \leq x\}$; podobno naj pomeni d najbližjega desnega soseda: $d = \min\{y : y \in S(b, n), y > x\}$. Pri tako določenih številih c in d , sosedih števila x , naj pomeni

$$G(b, n, x) = (d - c)/x, \quad x > 0 \\ G(b, n, x) = G(b, n, -x), \quad x < 0$$

Funkcija G meri torej širino presledka med sosedoma iz množice $S(b, n)$ k številu x , relativno na velikost števila x .

Pri primerjavi relativne napake upodobitve števila x s števili iz množice $S(b, n)$ in $S(a, n)$, $a > b$, se nam vsiljuje domneva, da je relativna napaka upodobitve z n -mestnimi števili glede na večjo bazo a manjša od ustrezne napake glede na bazo b . Domneva je nasploh napačna, kot kaže tale primer. Vzemimo množici $S(10, 4)$ in $S(16, 4)$. Funkciji $G(10, 4, x)$



in $G(16, 4, x)$ sta narisani v logaritmičnem merilu v območju $0,01 \leq x \leq 10$. Iz slike je jasno, da je večji del realnih števil upodobljenih z manjšo napako v šestnajstiškem sistemu, kot v desetiškem, vsakič s 4 mesti. Realna števila iz intervala od 0,0625 do 1 pa so bolj natančno aproksimirana v bazi 10 kot v bazi 16!

Avtor članka [1] obravnava še druge probleme: število decimalk, ki ustreza številu binarnih mest pri zapisu danega realnega števila glede na bazo 10 in 2 in vpliv različnih načinov zaokrožanja na točnost rezultatov.

[1] D. W. Matula, Significant digits: Numerical analysis or numerology. IFIP Congress 71, Ljubljana, str. TA-I-33 do TA-I-37, 1971.

ANALIZA ALGORITMOV

ANDREJ KMET

MOS 68-01
68 A 10

»in this case, mathematics is being applied to computer problems instead of applying computers to mathematical problems«

D. E. Knuth

V članku je prikazanih nekaj metod, ki se uporabljajo v analizi algoritmov. Deloma je obdelan Evklidov algoritem ter algoritem za iskanje k-tega največjega elementa med n danimi elementi.

ANALYSIS OF ALGORITHMS

In this article we apply some techniques used in the analysis of algorithms to some particular problems, i. e. to Euclid algorithm and to algorithm for selecting the k -th largest of n given elements.

Da bo kak problem pripravljen za obdelavo na računalniku, mora biti razčlenjen na elementarne korake, ki so lahko osnovne, pa tudi sestavljene računske operacije. Zaporedju teh operacij in preprostih odločitev pravimo algoritem. Posamezne probleme v splošnem reši množica različnih algoritmov, ki so zato v teoretičnem smislu ekvivalentni. Pri konkretni uporabi na računalniku pa se pokaže, da so nekateri algoritmi nestabilni: — zaradi računanja s končnim številom decimalnih mest utegnejo biti rezultati zelo nenatančni ali pa celo popolnoma napačni. Po drugi strani se število operacij menja od enega algoritma do drugega.

Ena od vej matematike, ki se ukvarja s temi vprašanji, je analiza algoritmov.

Snov za članek sem črpal iz dveh virov: knjige Art of Programming D. E. Knutha ter članka Mathematical analysis of algorithms istega avtorja v IFIP-ovem biltenu.

Pomembno orodje pri analizi algoritmov so rekurzije. Ogledali si bomo preprosto rekurzivno relacijo

$$x_{n+1} - x_n = \frac{1}{n+1} \quad n = 0, 1, \dots$$

ker jo bomo kasneje večkrat srečali.

Rešitev rekurzije je zaporedje $\{x_n\}_{n=0}^{\infty}$, ki je natanko določeno z začetno vrednostjo x_0 . Zapišimo rekurzijo za nekaj zaporednih indeksov:

$$x_{n+1} - x_n = \frac{1}{n+1}$$

$$x_n - x_{n-1} = \frac{1}{n}$$

.....

$$x_2 - x_1 = \frac{1}{2}$$

$$x_1 - x_0 = 1$$

Seštejmo leve in desne strani zgornjih enačb!

$$x_{n+1} = x_0 + \sum_{k=1}^{n+1} \frac{1}{k}$$

Če si za x_0 izberemo vrednost 0, se rešitev tedaj glasi $x_n = H_n$, pri čemer smo s H_n označili n -to harmonično število

$$H_n = 1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n}$$

H_n je naraščajoče zaporedje, ki se asimptotično vede kot $\ln n$. Velja namreč $\lim (H_n - \ln n) = \gamma$, kjer je γ Eulerjeva konstanta in ima vrednost $\gamma = 0.57721\dots$

Zlahka preverimo preprosto sumacijsko pravilo za harmonična števila

$$\sum_{k=2}^n H_{k+t} = (n+t+1)H_{n+t} - (t+2)H_{t+1} - (n-1),$$

ki v primeru $t = 0$ preide v pravilo

$$\sum_{k=2}^n H_k = (n+1)H_n - (n+1)$$

Evklidov algoritem.

Naj bosta $x_0, x_1 \in \mathbb{N}$ naravni števili ter $x_0 \geq x_1$. Zaporedje $\{x_k\}_{k=0}^n$ tvorimo z znanim Evklidovim algoritmom:

$$x_0 = k_1 x_1 + x_2$$

$$x_1 = k_2 x_2 + x_3$$

.....

$$x_{n-2} = k_{n-1} x_{n-1} + x_n$$

$$x_{n-1} = k_n x_n$$

Pravimo, da se je Evklidov algoritem iztekel po n korakih. x_n je največji skupni delitelj števil x_0 in x_1 . Vprašajmo se, pri katerih najmanjših številih x_0 in x_1 se Evklidov algoritem izteče šele po n korakih! Zapišimo zgornji proces v malo drugačni obliki (spremenimo indekse)!

$$x_n = k_{n-1} x_{n-1} + x_{n-2}$$

$$x_{n-1} = k_{n-2} x_{n-2} + x_{n-3}$$

.....

$$x_2 = k_1 x_1 + x_0$$

$$x_1 = k_0 x_0$$

Očitno dosežemo minimum za x_n, x_{n-1} , če vzamemo za $x_0 = 1, k_0 = 2$ ter $k_1 = k_2 = \dots = k_{n-1} = 1$. Shema Evklidovega algoritma se za ta primer poenostavi:

$$x_{i+1} = x_i + x_{i-1} \quad i = 1, 2, \dots, n-1$$

$$x_0 = 1, x_1 = 2$$

I

Z relacijo (I) je zaporedje $\{x_k\}_{k=0}^n$ natanko določeno. Tvorijo ga števila

$$1, 2, 3, 5, 8, 13, 21, 34, \dots$$

t. im. Fibonaccijevo zaporedje.

Najmanjši števili, za kateri je potrebno n korakov Evklidovega algoritma, sta torej n -to in $n-1$ Fibonaccijevo število.

Izrek. Naj bo dano naravno število p . Za vsak par $u, v \leq p$ je potrebnih kvečjemu $[\log_a \sqrt{5} \cdot p] - 2$ korakov Evklidovega algoritma. Pri tem je $a = \frac{1 + \sqrt{5}}{2}$.

Največje število deljenj nastopi, kadar sta u in v zaporedni Fibonaccijevi števili F_n, F_{n-1} . Določiti moramo največji n , pri katerem je še $F_n \leq p$.

Rekurzivno relacijo (I) je mogoče analitično rešiti. Splošni člen x_n se izraža takole:

$$x_n = \frac{1}{\sqrt{5}} (a^{n+2} - (-a)^{-n-2})$$

Za vsak $n \geq 0$ prevladuje prvi člen

$$\left| x_n - \frac{1}{\sqrt{5}} a^{n+2} \right| < 0.5,$$

saj je $a^{-n-2} < a^{-2} < 0.5$.

x_n so torej cela števila, ki so najbližja členom geometrijskega zaporedja $\frac{1}{\sqrt{5}} a^{n+2}$, zato

$$a^{n+2} < \sqrt{5} p \text{ odnosno}$$

$$n + 2 < \log_a \sqrt{5} p \quad \text{ali}$$

$$n < [\log_a \sqrt{5} p] - 2$$

$$n < 4.78510 \log_{10} p$$

kar pomeni, da je število deljenj gotovo manjše od približno petkratnega števila desetiških mest.

Največje število korakov Evklidovega algoritma poznamo. Poskusimo ugotoviti še poprečno!

Označimo s $T(m, n)$ število deljenj, če opravimo Evklidov algoritem na številih m in n . Pri tem velja opozoriti, da vedno delimo prvo število (m) z drugim (n), ne glede na to, katero je večje. Pri prvem koraku torej dopuščamo kvocient o in ostanek m . Nadaljujemo tako, da prejšnji divizor delimo z ostankom. S tem smo dosegli, da je število $T(m, n)$ pri fiksnem n isto za vsa števila, ki so kongruentna po modulu n . Na primer:

$$T(3,5) = T(8,5) = T(13,5) = \dots$$

Definirajmo še

$$T(m, 0) = 0$$

Naj bo T_n poprečno število deljenj pri Evklidovem algoritmu nad števili n, m , pri čemer izbiramo m slučajno. Iz pravkar povedanega sledi:

$$T_n = \frac{1}{n} \sum_{m=0}^{n-1} T(m, n)$$

Saj je vsa števila $m > n$ mogoče pisati v obliki $m = p + kn$, kjer je $p < n$ in $T(m, n) = T(p, n)$.

Za primer izračunajmo T_5 :

$$T(0,5) = 1, T(1,5) = 2, T(2,5) = 3,$$

$$T(3,5) = 4, T(4,5) = 3$$

$$T_5 = \frac{1}{5} (1 + 2 + 3 + 4 + 3) = 2.6$$

Potrebujemo oceno za T_n pri velikih n . Pomagajmo si s tole aproksimacijo: Izberimo število n . Pri slučajno izbranem m imamo poprečno T_n operacij. Po prvem koraku dobimo enako verjetne ostanke $r = 0, 1, \dots, n - 1$. Torej imamo še bodisi $T_0, T_1, \dots, T_{n-1}$ operacij.

$$T_n \approx 1 + \frac{1}{n} (T_0 + T_1 + \dots + T_{n-1})$$

$$T_0 = 0$$

Pri tem smo tiho privzeli, da je po prvem deljenju vnaprej izbrano število n slučajno glede na prvi ostanek r , kar je seveda precej groba aproksimacija. Rešimo diferenčno enačbo:

$$x_{n+1} = 1 + \frac{1}{n+1} (x_0 + x_1 + \dots + x_n)$$

$$x_{n+1} = 1 + \frac{1}{n+1} ((x_n - 1)n + x_n) = 1 - \frac{n}{n+1} + x_n$$

$$x_{n+1} - x_n = \frac{1}{n+1}$$

Če upoštevamo začetni pogoj $x_0 = 0$, je rešitev $x_n = H_n$. V poprečju potrebujemo torej $T_n \approx \ln n$ operacij ali $T_n \approx 2.3 \log n$, kar je približno 2.3 krat število desetiških mest.

Oglejmo si še enega od algoritmov za določitev k -tega največjega števila med n števili! Opišimo najprej algoritem! Naj bodo $x_1, \dots, x_n$ števila ali v splošnem elementi, ki jih je mogoče urediti, ter naj bodo med seboj različni. Izberimo poljuben x_i ter primerjajmo po vrsti vse ostale člene zaporedja z njim! Vse večje uvrščajmo na začetna mesta $y_1, y_2, \dots$ vse manjše pa na zadnja $y_n, y_{n-1}, \dots$! Denimo, da naš element pristane na m -tem mestu! Če je $m = k$, je algoritem končan, sicer pa nadaljujemo tako, da v primeru $m > k$ iščemo k -ti največji element med prvimi m , v primeru $m < k$ pa $(k - m)$ -tega med zgornjo $(n - m)$ -torico. Preprost premislek nas vodi do zaključka, da moramo opraviti v najbolj neugodnem primeru $n(n - 1)/2$ primerjanj. To pa je tudi potrebno število primerjanj za popolno ureditev zaporedja. Zanimalo nas bo predvsem, kakšno je poprečno število primerjanj, saj ne smemo pričakovati, da se bo vedno pojavil najbolj neugoden primer.

Porabimo še nekaj besed o tem, kako naj definiramo poprečno število primerjanj $C_{n,k}$, da med n različnimi elementi določimo k -tega po velikosti!

Imenujmo zaporedje, kjer si členi sledijo po velikosti, osnovno! Očitno je število primerjanj za določitev k -tega elementa isto za vsa zaporedja, ki so ista permutacija osnovnega zaporedja. Zadostuje torej, da si ogledamo množico permutacij enega zaporedja ter definiramo poprečno število primerjanj kot

$$C_{n,k} = \frac{\sum_{p \in P_n} C(p, k)}{n!},$$

kjer smo s $C(p, k)$ označili število primerjanj, potrebnih, da z izbranim algoritmom določimo k -ti element v zaporedju, ki je p -ta permutacija osnovnega zaporedja. Seštevamo po vseh permutacijah P_n n elementov.

Poizkusimo dobiti rekurzivno relacijo med $C_{n,k}$ -ji! Za večjo nazornost si pomagajmo z zgledom! Denimo, da iščemo tretji največji element med petimi! Izberimo na primer prvega po vrsti ter ga primerjajmo z ostalim, kot smo opisali zgoraj! Po štirih primerjanjih dobimo naslednje enako verjetne izide:

$$0xxxx \quad x0xxx \quad xx0xx \quad xxx0x \quad xxxx0$$

Pri tem smo z 0 označili izbrani element; večji od njega so levo, manjši pa desno od njega ter označeni s skupnim znakom x . Očitno moramo v naslednjem koraku opraviti bodisi $C_{4,2}$, $C_{3,1}$, 0, $C_{3,3}$ in $C_{4,3}$ primerjanj. V splošnem je

$$C_{n,k} = (n-1) + \frac{1}{n}(C_{n-1,k-1} + C_{n-2,k-2} + \dots + C_{n-k+1,1} + C_{n-1,k} + C_{n-2,k} + \dots + C_{k,k})$$

Označimo prvi del vsote z $A_{n,k}$, drugega pa z $B_{n,k}$; torej

$$A_{n,k} = C_{n-1,k-1} + C_{n-2,k-2} + \dots + C_{n-k+1,1}$$

$$B_{n,k} = C_{n-1,k} + C_{n-2,k} + \dots + C_{k,k}$$

Za $A_{n,k}$ in $B_{n,k}$ veljajo rekurzije;

$$A_{n+1,k+1} = A_{n,k} + C_{n,k}$$

$$B_{n+1,k} = B_{n,k} + C_{n,k}$$

Poskusimo eliminirati $A_{n,k}$ in $B_{n,k}$, da dobimo rekurzijo le za $C_{n,k}$:

$$(n+1)C_{n+1,k+1} = n(n+1) + A_{n+1,k+1} + B_{n+1,k+1}$$

$$nC_{n,k+1} = n(n-1) + A_{n,k+1} + B_{n,k+1}$$

$$nC_{n,k} = n(n-1) + A_{n,k} + B_{n,k}$$

$$(n-1)C_{n-1,k} = (n-1)(n-2) + A_{n-1,k} + B_{n-1,k}$$

Seštejmo prvo in četrto enačbo ter odštejmo srednji dve!

$$\begin{aligned} (n+1)C_{n+1,k+1} - nC_{n,k+1} - nC_{n,k} + (n-1)C_{n-1,k} &= \\ &= 2 + C_{n,k} - C_{n-1,k} + C_{n,k+1} - C_{n-1,k} \end{aligned}$$

Če preuredimo, dobimo

$$C_{n+1,k+1} - C_{n,k+1} - C_{n,k} + C_{n-1,k} = \frac{2}{n+1} \quad \text{II}$$

Dobiti moramo še robne pogoje, to je $C_{n,1}$ in $C_{n,n}$.

$$C_{n,1} = (n-1) + \frac{1}{n}(C_{1,1} + C_{2,1} + \dots + C_{n-1,1}); \quad C_{1,1} = 0$$

Če zapišemo to relacijo še za $C_{n+1,1}$ ter ju odštejemo, dobimo:

$$(n+1)C_{n+1,1} - nC_{n,1} = 2n + C_{n,1}$$

$$C_{n+1,1} - C_{n,1} = \frac{2n}{n+1} = 2 - \frac{2}{n+1}$$

Rešitev te rekurzije pa se glasi:

$$C_{n,1} = 2n - 2H_n \text{ in zaradi simetrije } C_{n,n} = 2n - 2H_n$$

Če zapišemo rekurzijo (II) za več zaporednih n ter nato dobljene relacije seštejemo, dobimo:

$$C_{n+1, k+1} - C_{n, k} - (C_{n, k+1} - C_{n-1, k}) = \frac{2}{n+1}$$

$$C_{n, k+1} - C_{n-1, k} - (C_{n-1, k+1} - C_{n-2, k}) = \frac{2}{n}$$

...

$$C_{k+2, k+1} - C_{k+1, k} - (C_{k+1, k+1} - C_{k, k}) = \frac{2}{k+2}$$

Vsota je

$$C_{n+1, k+1} - C_{n, k} = \frac{2}{n+1} + \dots + \frac{2}{k+2} + (C_{k+1, k+1} - C_{k, k})$$

Če upoštevamo še robne pogoje, je

$$\begin{aligned} C_{n+1, k+1} - C_{n, k} &= 2H_{n+1} - 2H_{k+1} + 2(k+1) - 2H_{k+1} - 2k + 2H_k = \\ &= 2(H_{n+1} - H_{k+1}) + 2 - \frac{2}{k+1} \end{aligned}$$

Zgornjo relacijo zapišemo za (n, k) , $(n-1, k-1)$, ... ter seštejemo. Dobimo

$$C_{n, k} = 2 \sum_{i=2}^k \left(H_{n-k+i} - H_i + 1 - \frac{1}{i} \right) + C_{n+1-k, 1} = 2((n+1)H_n -$$

$$- (n-k+2)H_{n-k+1} - (k-1) - (k+1)H_k + (k+1) + (k-1) - H_k + 1 + \\ + (n+1-k) - H_{n+1-k}) = 2[(n+1)H_n - (n+3-k)H_{n-k+1} - (k+2)H_k + n+3]$$

V primeru, da je $n = 2k - 1$ (iščemo mediano n elementov), je poprečno število primerjanj

$$\begin{aligned} C_{n, k} &= 2[2kH_{2k-1} - 2(k+2)H_k + 2k + 2] \\ C_{n, k} &= 4k(H_{2k-1} - H_k) - 8H_k + 4k + 4 = \\ &= 4k(1 + \ln 2) - 8\ln k + o(1) = \\ &= 6.8k - 8\ln k + o(1) \end{aligned}$$

Kljub temu, da je poprečno število korakov pri iskanju mediane sorazmerno majhno (približno 3.4 krat število elementov), algoritem ni prikladen za iskanje na primer prvega med n elementi. Z našim algoritmom potrebujemo vsaj $n - 1$ primerjanj (v poprečju približno $2n$), medtem ko je za sledeči algoritem potrebnih natanko $n - 1$ primerjanj.

Prvi element x_1 primerjamo z naslednjimi, dokler ga eden ne preseže. Od tu naprej primerjamo tega z naslednjimi itd.

Za zaključek bi omenil, da postaja analiza algoritmov vse bolj »resna« matematična disciplina. Prebrodila je obdobje reševanja posameznih problemov s triki in vedno bolj uporablja sistematčne prijeme. Ne moremo ji torej oporekati, da ni perspektivna, saj jo čaka še mnogo zanimivih problemov, da jih reši.

STABILNOST NUMERIČNIH PROCESOV

JERNEJ KOZAK

MOS 65-01

V tem članku je razložen pomen stabilnosti numeričnih procesov ter ilustriran z nekaj numeričnimi zgledi.

THE STABILITY OF NUMERICAL PROCESSES

In this article the importance of stability of numerical processes is discussed, and some illustrative numerical examples given.

Pred uvedbo elektronskih računalnikov je predstavljalo število aritmetičnih operacij enega poglavitnih problemov pri obravnavanju numeričnih metod. Da bi si skrajšali računske muke, so si matematiki pomagali z vsemogočimi tabelami, ki so služile predvsem redukciji aritmetičnih operacij na sprejemljivo, a po pravilu majhno število. Z uporabo modernih računalnikov pa postane čas posamezne operacije zanemarljivo majhen. Število operacij, ki jih nalagamo računalniku, postaja vse večje, problemi vse bolj kompleksni.

Veliko število operacij odkriva probleme, ki jih pri računih s svinčnikom in papirjem nismo poznali. Končna diskretna upodobitev realnih števil v računalniku vodi k zaokrožitvenim napakam, tako da numeričnega procesa kot zaporedja aritmetičnih operacij ne moremo eksaktno realizirati. Zaokrožitvene napake se pojavljajo kot motnje in od narave procesa je odvisno, ali vplivajo bistveno na pravilnost rezultatov ali ne. Tako pridemo do pojma stabilnosti numeričnega procesa. Za numerični proces pravimo, da je stabilen, če zaokrožitvene napake, ki jih naredimo med izračunom, ne vplivajo mnogo na pravilnost rezultatov. Idealno stabilni procesi so redki in tako je stabilnost postopka eden najvažnejših kriterijev pri izbiri algoritma za numerično reševanje problemov. Poglejmo si nekaj preprostih zgledov, ki nazorno prikazujejo pomembnost numerične stabilnosti.

Vzemimo integral

$$I_n = e^{-1} \int_0^1 x^n e^x dx \quad (1)$$

ter izračunajmo njegovo vrednost za večji nabor naravnih števil $n!$. Z integracijo per partes ugotovimo, da vrednosti I_n zadoščajo rekurzivni relaciji

$$I_0 = 1 - e^{-1}, \quad I_n = 1 - nI_{n-1} \quad n = 1, 2, \dots \quad (2)$$

Ker I_0 poznamo, lahko z zaporednim vstavljanjem v rekurzivno formulo, vsaj teoretično, določimo vrednosti I_n . Ker v računalniku ne moremo popolnoma točno računati, dobimo namesto pravih vrednosti I_n približne, ki jih bomo označili z $\tilde{I}_n$. Denimo, da δ_n predstavlja zaokrožitveno napako na n -tem koraku. Vrednosti $\tilde{I}_n$ zadoščajo rekurzivni enačbi:

$$\tilde{I}_0 = I_0 + \delta_0, \quad \tilde{I}_n = 1 - n\tilde{I}_{n-1} + \delta_n \quad n = 1, 2, \dots \quad (3)$$

Poglejmo si, kako se obnaša napaka ε_n , razlika izračunane in prave vrednosti. Ko odštejemo (2) od (3), ugotovimo, da za ε_n velja

$$\varepsilon_0 = \delta_0, \quad \varepsilon_n = -n\varepsilon_{n-1} + \delta_n \quad n = 1, 2, \dots \quad (5)$$

Če zanemarimo vpliv zaokrožitvenih napak $\delta_1, \delta_2, \dots$, vidimo, da se ε_n obnaša kot

$$\varepsilon_n \approx (-1)^n n! \delta_0. \quad (6)$$

Prave vrednosti I_n z rastočim n konvergirajo proti nič, napaka ε_n pa po (6) narašča prek vseh meja in prave vrednosti integralov popolnoma prekrije.

Rekurzivni izračun (2) predstavlja torej nestabilni numerični proces, kar lahko praktično preverimo na katerem koli računalniku. Diferenčno enačbo (2) lahko obrnemo:

$$I_{n-1} = (1 - I_n)/n \quad n = N, N-1, \dots \quad (7)$$

Žal vrednosti I_N za $n = N$ ne poznamo in ne moremo začeti. Ker vemo, da I_n konvergira proti nič, si izberemo dovolj velik $n = N$, postavimo za približno vrednost $\tilde{I}_n = 0$ ter začnemo z rekurzivnim izračunom nazaj po formuli (7). V začetku zagrešimo napako, ki je precej večja kot zaokrožitvena napaka δ_0 v (3). Približne vrednosti $\tilde{I}_n$ zadoščajo enačbi:

$$\tilde{I}_n = 0 \quad \tilde{I}_{n-1} = (1 - \tilde{I}_n)/n + \delta_{n-1} \quad n = N, N-1, \dots \quad (8)$$

Podobno kot prej ugotovimo, da za napako ε_n velja

$$\varepsilon_n = \delta_n = I_n, \quad \varepsilon_{n-1} = -\varepsilon_n/n + \delta_{n-1} \quad n = N, N-1, \dots \quad (9)$$

Zaokrožitvene napake sproti izginjajo, saj se dele z vedno večjimi števili. Od začetne napake δ_N za $n < N$ preostane le še

$$\frac{\delta_N}{N(N-1) \dots (N-n)} \quad (10)$$

tako da tudi začetna napaka, čeprav relativno velika, po nekaj korakih izgine. Rekursivni izračun (7) predstavlja stabilni numerični proces.

Ogledali smo si problem stabilnosti pri enočlenski rekurzivni formuli. Pri veččlenskih naletimo na še večje težave. Denimo, da bi želeli ugnati enačbo

$$y_{r+1} - 10 \cdot 1y_r + y_{r-1} = 0 \quad (11)$$

Splošno rešitev te enačbe, ki jo poiščemo z nastavkom $y_r = \lambda^r$, dobimo kot linearno kombinacijo funkcij 10^r in 10^{-r} :

$$y_r = A_1 10^r + A_2 10^{-r} \quad (12)$$

Vzemimo, da iščemo rešitev, pri kateri je $A_2 = 0$! Uporabiti moramo rekurziven izračun naprej, z rastočim r . Zaokrožitvene napake, ki jih zagrešimo, povzročijo, da dobi izračunana rešitev y_r tudi komponento v smeri druge rešitve, funkcije 10^{-r} . Ker ta rešitev pada, prava pa raste, zaokrožitvene napake v smeri 10^{-r} sproti izginjajo. Če bi začeli pri velikem r , bi prava rešitev padala, motnja pa rastla, tako da bi zaokrožitvene napake, podobno kot prej, pojedle prave vrednosti.

Poglejmo si še, kako stabilno izračunati vrednosti funkcije e^{-x} za pozitivne argumente! Kot vemo, lahko uporabimo vrsto

$$e^{-x} = 1 - \frac{x}{1!} + \frac{x^2}{2!} - \dots \quad (13)$$

S trivialno spremembo dobimo še formulo:

$$e^{-x} = 1/e^x = 1/\left(1 + \frac{x}{1!} + \frac{x^2}{2!} + \dots\right) \quad (14)$$

Pokazali bomo, da je malenkostna sprememba v načinu izračuna e^{-x} numerično zelo pomembna.

Poglejmo si posamezne člene $x^n/n!$. Njihova vrednost za cele, pozitivne x narašča vse do $n = x$, kjer člen doseže velikost $x^n/x!$

Za velike x lahko $x!$ aproksimiramo s pomočjo Stirlingove formule

$$x! \approx \sqrt{2\pi x} x^x e^{-x} \quad (15)$$

kar nam da za velikost člena

$$\frac{x^x}{x!} \approx \frac{1}{\sqrt{2\pi x}} e^x \quad (16)$$

Za velike x je to spoštljivo velika vrednost. Do iste ugotovitve pridemo tudi za argumente x , ki niso nujno celi. V računalniku vrednosti posameznega člena ne moremo popolnoma točno določiti. Približno vrednost dobimo kot

$$\widetilde{\left(\frac{x^n}{n!}\right)} = \frac{x^n}{n!} + \delta_n \quad (17)$$

kjer je δ_n zaokrožitvena napaka. Če je velikost člena $x^n/n!$ velika, je velikost δ_n , ki je sicer relativno na velikost $x^n/n!$ majhna, absolutno velika.

Vzemimo, da sedaj izračunamo približno vrednost $\widetilde{(e^{-x})}$ takole:

$$\begin{aligned} \widetilde{(e^{-x})} &= \widetilde{(1)} - \widetilde{\left(\frac{x}{1!}\right)} + \widetilde{\left(\frac{x^2}{2!}\right)} - \dots \\ &= e^{-x} + \delta_0 - \delta_1 + \delta_2 - \dots \end{aligned} \quad (18)$$

Vrednost funkcije e^{-x} je za večje x zelo majhna, medtem ko so vrednosti zaokrožitvenih napak mnogo večje in približna vrednost iz formule (18) je popolnoma napačna. Če pa najprej določimo približno vrednost funkcije e^x iz

$$\widetilde{(e^x)} = e^x + \delta_0 + \delta_1 + \dots, \quad (19)$$

dobimo mnogo točnejši rezultat, saj v (19) prevlada e^x , ki je relativno mnogo večja od zaokrožitvenih napak.

N	$I(N) - \text{NAPREJ}$	ABS. NAPAKA	$I(N) - \text{NAZAJ}$	ABS. NAPAKA
0	0.632120E 00	0 000000E 00	0.632120E 00	0.000000E 00
1	0.367879E 00	0.596046E-07	0.367879E 00	0.000000E 00
2	0.264241E 00	0.119209E-06	0.264241E 00	0.000000E 00
3	0.207277E 00	0.417232E-06	0.207276E 00	0.000000E 00
4	0.170891E 00	0.163912E-05	0.170893E 00	0.000000E 00
5	0.145541E 00	0.825524E-05	0.145532E 00	0.000000E 00
6	0.126752E 00	0.494718E-04	0.126802E 00	0.000000E 00
7	0.112730E 00	0.346526E-03	0.112383E 00	0.000000E 00
8	0.981598E-01	0.277216E-02	0.100931E 00	0.000000E 00
9	0.116561E 00	0.249496E-01	0.916122E-01	0.000000E 00
10	—0.165618E 00	0.249495E 00	0.838770E-01	0.149011E-07
11	0.282180E 01	0.274445E 01	0.773523E-01	0.104308E-06
12	—0.328617E 02	0.329334E 02	0.717719E-01	0.128150E-05
13	0.428202E 03	0.428135E 03	0.669642E-01	0.165849E-04
14	—0.599382E 04	0.599389E 04	0.625000E-01	0.232160E-03
15	0.899084E 05	0.899083E 05	0.625000E-01	0.348246E-02

Tabela vrednosti I_n .

X	EXP ($-X$) — (1)	ABS. NAPAKA	EXP ($-X$) — (2)	ABS. NAPAKA
1.	0.367879E 00	0.238418E-06	0.367879E 00	0.000000E 00
2.	0.135335E 00	0.596046E-07	0.135335E 00	0.000000E 00
3.	0.497872E-01	0.216066E-06	0.497870E-01	0.000000E 00
4.	0.183159E-01	0.286847E-06	0.183156E-01	0.000000E 00
5.	0.673680E-02	0.114087E-05	0.673794E-02	0.000000E 00
6.	0.248469E-02	0.594183E-05	0.247875E-02	0.000000E 00
7.	0.941297E-03	0.294158E-04	0.911882E-03	0.000000E 00
8.	0.343467E-03	0.800512E-05	0.335462E-03	0.000000E 00
9.	0.174506E-03	0.510963E-04	0.123409E-03	0.000000E 00
10.	—0.232107E-03	0.277507E-03	0.453999E-04	0.000000E 00
11.	0.321628E-03	0.304926E-03	0.167017E-04	0.000000E 00
12.	—0.370067E-03	0.376212E-03	0.614421E-05	0.000000E 00
13.	0.107021E-02	0.106795E-02	0.226032E-05	0.000000E 00
14.	0.111699E-03	0.110867E-03	0.831528E-06	0.000000E 00
15.	0.153040E-01	0.153037E-01	0.305902E-06	0.000000E 00
16.	—0.153088E-01	0.153089E-01	0.112535E-06	0.000000E 00

Tabela vrednosti e^{-x} .

Za konec si pogledjmo v zgornjih tabelah še numerična zgleda, izračunana na računalniku IBM 1130 s točnostjo 23 binarnih decimalk! Primerjajmo vrednosti I_n , dobljene po formulah (3) in (7) za $n = 0, 1, \dots, 15$, ter vrednosti e^{-x} , izračunane iz izrazov (13) in (14) za $x = 1, 2, \dots, 16$!

LITERATURA

I. Babuška, M. Prager, E. Vitasek: NUMERICAL PROCESSES IN DIFFERENTIAL EQUATIONS. SNTL, A Division of JOHN WILEY & Sons, London, 1964.

L. Fox, D. F. Mayers: COMPUTING METHODS FOR SCIENTISTS AND ENGINEERS Clarendon press, Oxford 1968.

LITERATURA ZA DOPOLNILNO BRANJE

UČBENIKI:

1. I. S. Berezin, N. P. Židkov: Metody vyčislenij, Tom I, II, Nauka, Moskva, 1966.
2. B. P. Demidovič, I. A. Maron: Osnovy vyčislitel'noj matematiki, Nauka, Moskva, 1966.
3. B. P. Demidovič, I. A. Maron, E. Z. Šuvalova: Čislennye metody analiza, Fizmatgiz, Moskva, 1962.
4. E. Isaacson, H. B. Keller: Analysis of numerical methods, John Wiley, 1966.
5. W. E. Milne: Numerical calculus, Princeton, 1959.
6. A. Ralston: A first course in numerical analysis, Mc Graw-Hill, 1965.

SLOVENSKI TEKSTI:

1. Z. Bohte: Numerično reševanje enačb, Sigma, 1964.
2. Z. Bohte: Napake pri numeričnem računanju, OMF 18 (1971).
3. Z. Bohte: Števila s stalno in premično vejico, OMF 19 (1972).
4. A. Kmet: O preprostih diferenčnih enačbah, OMF 15 (1968).
5. J. Kozak: Zlepki in interpolacija, OMF 18 (1971).
6. S. Pahor: Reševanje enačb z iteracijo, OMF 6 (1957/58).
7. T. Skubic: Programiranje v simboličnem jeziku ALGOL 60, OMF 12 (1965).
8. A. Suhadolc: Cauchyjeva metoda reševanja algebrskih enačb, OMF 11 (1964).
9. E. Zakrajšek: Programiranje v ALGOLU, Ljubljana, 1966.
10. E. Zakrajšek: O numerični uporabi neke rekurzivne formule, OMF 15 (1968).

IV. SEMINAR IZ MATEMATIKE – NOVO RAČUNSTVO

V letošnjem letu je Društvo matematikov, fizikov in astronomov SRS v sodelovanju z oddelkom za matematiko pri Inštitutu za matematiko, fiziko in mehaniko Univerze v Ljubljani, odsekom za matematiko Fakultete za naravoslovje in tehnologijo ter zavodom za šolstvo SR Slovenije organiziralo že IV. seminar iz matematike, namenjen predavateljem matematike v srednjih šolah. Delovni naslov seminarja je bil NOVO RAČUNSTVO. Vabilo na seminar in program sta bila objavljena v OMF, 18., 1971, 4. štev.

Dopolnilno izobraževanje profesorjev srednjih šol je po zakonu o srednjem šolstvu obvezno. Zato se mora vsak profesor v petih letih vsaj enkrat udeležiti seminarja iz svoje stroke. Kakor lansko leto, je tudi letos zavod za šolstvo SRS predlagal, naj bo seminar obvezen za vse tiste, ki jih bo vabil zavod na seminar. Ker pa tako dopolnilno izobraževanje še ni do vseh podrobnosti urejeno, ker so na matematičnih in fizikalnih seminarjih večinoma skoraj vedno isti udeleženci (le redki profesorji poučujejo en sam predmet) in bi bilo preveč, če bi se morali obvezno udeleževati seminarjev pogostejše (naši seminarji so v zadnjih desetih letih redno vsako leto), ker imajo matematiki in fiziki v okviru Društva matematikov, fizikov in astronomov SRS redne mesečne sestanke, je upravni odbor društva predlagal zavodu za šolstvo, naj bo udeležba seminarja prostovoljna. Zavod za šolstvo je na osnovi naših argumentov dani predlog tudi sprejel. Močan razlog za svobodno udeležbo pa je tudi dejstvo, da seminar, ki ga organizira društvo, ne more biti obvezen. Ker je zavod za šolstvo naš program za seminar zelo pozitivno ocenil, je tudi prevzel plačilo večine finančnih stroškov seminarja. Naj se še na tem mestu zavodu v imenu organizatorjev in celotnega društva prav lepo zahvalim za razumevanje pri reševanju finančnih problemov.

Program za letošnji seminar je bil tak, da bi bil glede na popoldanske vaje najbolj primerno število udeležencev okrog 50. Zaradi izrednega zanimanja za očitno posrečeno izbran program se je do roka prijavilo 84 slušateljev, do začetka predavanj pa kar 113. 37 jih je bilo iz Ljubljane, 75 pa iz drugih krajev Slovenije. Seminarja se je udeležil tudi gost iz Zagreba, prof. ing. Vido Nikolić. 63 je bilo kolegov iz slovenskih gimnazij, 13 iz tehniških šol, 37 pa iz drugih šol, zaradi velike udeležbe in zaradi skrčenja programa s petih na štiri dni je trpel predvsem popoldanski praktični del.

Organizatorji so na tem seminarju vpeljali še nekaj drugih novosti, te so udeleženci pohvalili. Z vabilom na seminar, ki smo ga poslali na vse štiriletne srednje šole, smo poslali tudi ekspres informacijo o napovedanih predavanjih. Prijavljenci so se lahko že pred seminarjem seznanili z vsebino seminarja.

Po dopoldanskih predavanjih so dobili udeleženci problem za premislek ter so lahko do drugega dne samostojno izdelali program in ga v popoldanskem času tudi preskusili na elektronskem računalniku IBM 1130. Pri odpravljanju nujnih napak so jim pomagali sodelavci računskega centra. Udeležba pri vajah je bila tolikšna, da vaj niso nikdar zaključili ob napovedanem času, pač pa so jih morali organizirati tudi tretji dan istočasno z računskim praktikumom.

Kot tretje pa so imeli udeleženci med dopoldanskimi predavanji enourne odmori, med katerimi so se lahko pogovorili o strokovnih, stanovskih, pa tudi o privatnih zadevah. S tem smo skušali ustvariti res prijetno vzdušje in ne preveč naporen teden. Neformalni zaključek smo imeli že predzadnji dan pri skupni večerji, tako da so lahko kolegi iz oddaljenih krajev odpotovali že v četrtek popoldne.

Poleg osnovnega programa iz numerične matematike in programiranja smo pripravili v popoldanskem času še naslednje: V ponedeljek popoldan je predaval prof. dr. Fran Do-

minko o pulzarjih, zvečer pa smo si ogledali Astronomsko-geofizikalni observatorij na Golovcu, kamor nas je kljub zasneženi cesti pripeljal avtobus. V torek zvečer je namesto referata in razgovora o moderni matematiki v naših osnovnih šolah predaval o uvajanju nove matematike v osnovne šole v Belgiji profesor Ignac Smolec iz Zagreba. Žal nam je, da se tega predavanja ni udeležilo več kolegov iz osnovne šole pa morda tudi elementarcev, ker je bilo predavanje zares zanimivo.

Referata in razgovora o učnih načrtih za srednje šole prvi dan ni bilo ter smo ga na izrecno zahtevo mnogih udeležencev improvizirali zadnji dan seminarja. Zaradi večletnih izkušenj smo referat namenoma uvrstili že prvi dan, tako da bi se lahko kolegi v naslednjih dneh in v zato posebej napovedanih odmorih na osnovi dobljenih informacij pogovorili in pripravili na razgovor in sklepe v zadnjem dnevu. Tako pa smo slišali le nekaj spontanih zahtev posameznikov ter vprašanj, z odgovori nanje pa nismo bili povsem zadovoljni. Med najpomembnejše predloge sodi iniciativa, naj bi v prihodnje organizirali podobne strokovne seminarje kot doslej v organizaciji Društva matematikov, fizikov in astronomov SRS, metodične seminarje, ki pa so prav tako nadvse potrebni, pa naj organizira zavod za šolstvo SR Slovenije.

Sekretar seminarja
Ciril Velkovrh

NAŠI SEMINARJI

Po svojih pravilih mora Društvo matematikov, fizikov in astronomov skrbeti tudi za izboljševanje pouka na naših šolah. Zato je društvo že od nekdaj organiziralo strokovna in metodična predavanja, na katerih naj bi člani dopolnili svoje znanje.

Posameznim predavanjem so se že v letu 1956 pridružili seminarji. Pri tečajih in seminarjih so s predavatelji, prostori in opremo sodelovali odsek za fiziko FNT (nekdaj Fizikalni inštitut Univerze), inštitut Jožef Štefan, inštitut za matematiko, fiziko in mehaniko, odsek za matematiko FNT in višja pedagoška šola v Ljubljani. Del potrebnega denarja je preskrbel zavod za šolstvo (nekdaj Zavod za napredek šolstva in Pedagoški center pri SPK LRS).

V letih od 1956 do 1961 je bilo pet tečajev eksperimentalne fizike za učitelje osnovnih šol, med njimi leta 1957 tečaj eksperimentalne fizike za učitelje srednjih šol. Tečaji so bili po pouku v juniju ali juliju in so trajali do 14 dni. Bili so v prostorih fizikalnega inštituta Univerze. Vsak dan je bilo po eno strokovno metodično predavanje in štiri do pet ur dela v laboratoriju, popoldan pa so bile ekskurzije v ljubljanske inštitute. Na tečajih so predavali in vodili vaje: Miroslav Adlešič, France Ahlin, Joško Battestin, Evald Bračko, Silvo Breskvar, Rajko Korbar, Pavel Kunaver, Fran Dominko, Franc Kvaternik, Darko Jamnik, Ivan Molinaro, Milan Osredkar, Anton Peterlin, France Plevnik, Tomo Skubic, Ivan Štalec, France Šušteršič, Stanko Uršič, Albin Wedam, Črt Zupančič in Davorin Tomažič s sodelavci.

Udeležencev je bilo na srednješolskem tečaju 36, na zadnjem tečaju za nižjo stopnjo pa 55. Kot rezultat teh tečajev so izšla tudi skripta »Zbirka poskusov iz fizike« (1960).

Leta 1961 je bil 17. in 18. marca v Portorožu **prvi seminar izbranih poglavij moderne fizike**. 118 udeležencem so predavali Peter Gosar o polprevodnikih in Miodrag Mihailović in Bogdan Povh o razpadu β . Po predavanjih so udeležencem priredili še ekskurzijo v Umag s hidrogliserjem ter v tovarno TOMOS.

Drugi seminar iz fizike je bil 18. in 19. maja naslednjega leta v Štatenbergu. Tu so udeleženci poslušali naslednja predavanja: Robert Blinc: Osnove teorije relativnosti, Janez Strnad: Eksperimentalni temelji teorije relativnosti in Mitja Kregar: Jedrske reakcije v zvezdah.

Na željo udeležencev je Zavod za napredek šolstva SRS naknadno izdal prvi dve predavanji v obliki skript.

Seminarja se je udeležilo 80 poslušalcev. Oba seminarja so organizirali društvo matematikov, fizikov in astronomov SRS, zavod za napredek šolstva in inštitut J. Štefan.

Prvega seminarja iz matematike, ki ga je organiziral oddelek za matematiko inštituta za matematiko, fiziko in mehaniko univerze v Ljubljani v sodelovanju z zavodom za napredek šolstva se je udeležilo okrog 80 učiteljev srednjih šol. V dneh od 22. do 28. junija 1964 so bila tale predavanja: Niko Prijatelj: Uvod v teorijo množic in logiko, Ivan Vidav: Osnovni pojmi abstraktne algebre, France Križanič: O pojmu prostora v matematični analizi in Rajko Jamnik: Verjetnostni račun.

Po predavanjih je bil še razgovor o študiju matematike na univerzi in ogled računskega centra.

Tretji seminar iz fizike je bil od 27. do 29. maja 1965, v Ljubljani na Poljanski c. 28. Organizirali so ga inštitut za matematiko, fiziko in mehaniko, zavod za napredek šolstva in Društvo matematikov, fizikov in astronomov SRS. Na tem seminarju, kjer je bila udeležba sorazmerno slaba, so predavali Črt Zupančič o nekaterih vprašanjih kvantne mehanike, Mitja Rosina in Janez Strnad pa o osnovnih delcih. Pogovor o pouku fizike je vodil France Ahlin.

Četrty seminar iz fizike je bil v dneh od 28. in 29. oktobra 1966 v prostorih tehniške šole za elektrotehniško stroko. Udeležilo se ga je okrog 60 poslušalcev. Seminar, ki sta ga organizirala poleg društva še oba inštituta, je bil namenjen *supraprevodnosti*, nekaterim vprašanjem *statistične mehanike in novostim iz astrofizike*. Predavali so Ivan Kuščer o osnovnih idejah statistične mehanike, Peter Gosar o supraprevodnosti, Fran Dominko pa o novostih iz astrofizike. Predavanja so se zaključila z razgovorom o šolskih problemih in z rednim letnim občnim zborom društva.

Od tedaj se izmenjujeta vsako leto fizikalni in matematični seminar.

Drugi seminar iz matematike v letu 1967 (od 26. junija do 1. julija) je obravnaval *vektorske prostore in uporabo v elementarni matematiki*. Predavanje Ivana Puclja »Elementarna geometrija kot model evklidičnega vektorskega prostora« je izšlo v obliki skript, ki jih je dobil vsak od 50 udeležencev po seminarju.

Poleg tega sta predavala Niko Prijatelj o osnovnih pojmi splošne teorije vektorskih prostorov, Ivan Vidav pa o končno razsežnih vektorskih prostorih s skalarnim produktom. Organizatorji inštitut za matematiko, fiziko in mehaniko, Društvo matematikov, fizikov in astronomov SRS in zavod za šolstvo so pripravili udeležencem še razgovor o zakonu za šolstvo, ogled računskega centra in razstavo strokovnih knjig.

Medtem ko so bili prvi štirje seminarji v obliki predavanj, je peti seminar iz fizike, ki je bil v času od 26. do 29. junija 1968 v Ljubljani na odseku za fiziko FNT, namenjen *uporabi elektronike pri pouku fizike*, ponovno uvedel vaje za udeležence, predavanja pa so spremljali eksperimenti. Na programu so bila naslednja predavanja: Anton Moljk: Merjenje in upravljanje fizikalnih količin ter fizikalne osnove senzorjev, Jože Pahor: Digitalna in analogna obdelava podatkov, Analogna računska tehnika, Študij vsiljenega nihanja z analogno računsko tehniko in Franc Cvelbar: Zapisi količin.

Vsak udeleženec je izdelal tranzistorski ojačevalac, razen tega pa je mogel opraviti večino opisanih meritev, za kar je prejel ciklostirana navodila. Pri tem delu jim je bil v veliko pomoč Davorin Tomažič z sodelavci. Udeleženci seminarja, ki so ga pripravili DMFA SRS, IMFM, katedra za fiziko FNT, IJŠ in ZŽ, so si ogledali še novosti na inštitutu J. Štefan in razstavo učil tvrdke PHYWE na ljubljanskem učiteljišču.

Predavanja so izšla v brošurici z naslovom *O novih možnostih pri fizikalnih eksperimentih*.

Ta seminar je bil zadnji, ki je bil po končanem pouku. Naslednji seminarji so bili v zimskem času, v začetku februarja. Udeležba na teh je prekoračila število sto.

Tako je bilo 107 udeležencev na **tretjem seminarju iz matematike** od 2. do 4. febr. 1970, ki je bil posvečen *grupam v geometriji in fiziki*. Predavanje »Grupe v geometriji« je bilo razmnoženo v obliki lične brošure, ki jo je prejel vsak udeleženec ob prihodu. Po predavanjih pa so si udeleženci seminarja ogledali novi računski elektronski stroj IBM 1130. Organizatorji DMFA SRS, IMFM, odsek za matematiko FNT in ZŠ so pripravili predavanja: Niko Prijatelj: Osnovni pojmi teorije grup, Ivan Vidav: Grupe v geometriji in France Križanič: Grupe v fiziki.

Šestega seminarja iz fizike ki je obravnaval *poglavja iz fizikalne optike*, se je udeležilo doslej največ poslušalcev. Bilo jih je kar 130. Vsak dan je bilo po eno teoretično predavanje, ostala pa so spremljali lepi eksperimenti. Vsa predavanja so izšla v brošuri, ki jo je prejel vsak udeleženec ob začetku seminarja. En dan je bil posvečen šolskim problemom, kjer sta poročala Stanko Uršič in Aleksander Kregar.

V času od 1. do 4. febr. 1971 so predavali na odseku za fiziko FNT: Janez Strnad: Elektromagnetno valovanje (teorija), Gabrijel Kernel: Optični eksperimenti v mikrovalovnem področju (eksperimentalno), Darko Jamnik: Sevanje dipola (z eksperimenti), Jože Pahor: Fizikalne osnove detekcije in energijske pretvorbe svetlobe (z eksperimenti), Lovro Pičman: Stimulirana emisija (teorija), Anton Moljk: Laserji (z eksperimenti), France Ahlin: Eksperimenti s šolskim laserjem, Sergej Pahor: Holografija (z eksperimenti), Robert Blinc: Prehod elektromagnetnega valovanja skozi snov (teorija), Peter Gosar: Kerrov efekt (z eksperimenti), Rudi Kladnik: Sipanje svetlobe na majhnih delcih (z eksperimenti) in Franc Cvelbar: Fotoefekt v polprevodnikih (z eksperimenti).

Organizatorji (DMFA, IMFM, Odsek za fiziko FNT, IJS, ZŠ SRS) so pripravili še ogled razstave učil za vaje na tehniški šoli za razne stroke v Ljubljani, laboratorijev na oddelku za fiziko, elektronskega računalnika in reaktorja v Podgorici.

S tem seminarjem se je začela akcija za nabavo šolskega laserja na srednjih šolah, ki pa še ni uspešno dokončana.

Četrty seminar iz matematike v času od 31. januarja do 3. februarja 1972 z naslovom *Novo računstvo* je prav tako prinesel nekaj novosti: vsa predavanja so zbrana v pričujoči številki Obzornika. V želji, da bi se udeleženci pripravili na vaje, jim je bil posredovan hkrati z vabilom ustrezen material. Pri praktičnih vajah so udeleženci na računalniku in računskih strojih preverjali, če so svoje »domače naloge« dobro rešili. Med seminarjem je izhajal bilten, ki je obveščal o novostih in spremembah. Namesto pričakovanih 50 je bilo na seminarju kar 113 udeležencev.

Seminarju je bilo dodano tudi predavanje iz astronomije z ogledom Astronomsko-geofizikalnega observatorija na Golovcu. Zamisel, da bi med seminarjem razpravljali o šolskih problemih in na koncu sprejeli odločitve, pa se ni uresničila.

Če sedaj razmišljamo o že tradicionalnih seminarjih, spoznamo, da je vsak prinesel nekaj novega, boljšega. Obenem pa lahko ugotovimo, da so vsi dosegli svoj namen: osvežiti in poglobiti pridobljeno znanje in ga dopolniti z novimi spoznanji.

To je nedvomno zasluga številnih sodelavcev, pa tudi tečajnikov, ki so z discipliniranim sodelovanjem omogočili pravo delovno vzdušje.

Sklenimo z željo, da bi bili seminarji v prihodnje še boljši in še bolj uspešni.

Dušan Modic

DELO PODRUŽNIC DRUŠTVA MATEMATIKOV, FIZIKOV IN ASTRONOMOV SRS V LETU 1971

Društvo matematikov, fizikov in astronomov SR Slovenije ima že šest podružnic v večjih krajih Slovenije. Zastopniki le-teh so na letošnjem občnem zboru poročali o svojem delu v minulem letu ter nakazali nekaj najvažnejših načrtov za prihodnje obdobje.

Najstarejša podružnica je v *Mariboru*. Tu je že vrsto let tajnica kolegica Marija Munda. V zadnjem letu so organizirali naslednja predavanja: Dva študenta sta predavala dijakom o vektorjih in programiranju v FORTRAN, Janez Strnad je predaval o podaljšanju časa v specialni teoriji relativnosti in se z dijaki pogovarjal o študiju matematike in fizike. France Avsec je govoril kolegom o svojih izkušnjah pri pouku verjetnostnega računa. Na šolah so delovali krožki iz matematike in fizike, ki so jih vodili dijaki in profesorji. Name-njeni so bili poglobljanju znanja in pripravam za tekmovanja.

Podružnica v *Celju*, ki vključuje kolege iz 34 osnovnih šol ter 7 srednjih šol, je organizirala naslednja predavanja: Stane Pirnat je govoril o poskusih z leserjem, Zdenka Baša je predavala o novi matematiki v osnovni šoli. Franjo Jakhel iz Maribora je predaval o avdiovizualnih sredstvih pri pouku matematike in fizike. Za dijake srednjih šol pa je predaval Jože Vrabec o grafih. Odbor podružnice je organiziral tudi ekskurzijo v Ljubljano, kjer so si člani ogledali fizikalni praktikum na odseku za fiziko, matematično knjižnico, planetarij v centralni zgradbi univerze, astronomsko-geofizikalni observatorij na Golovcu ter rojstno hišo Jurija Vege v Zagorici.

Podružnica društva v *Novi Gorici* je najmlajša. Imeli so le dva delavna sestanka. Pomenili so se o problemih pri pouku matematike in fizike na osnovnih šolah. Kolegi iz osnovnih šol so potožili, da ravnatelji v težnji za čim manjšim osipom pri ocenjevanju preveč pritiskajo na predavatelje. V okviru podružnice je prišlo do večje povezave med učitelji na osnovnih in srednjih šolah. Alojz Kumar pa je predaval ločeno za profesorje in dijake o holografiji in o lesarju.

V okviru *koprške* podružnice so poslušali naslednja predavanja: Bogomila Kolenko je predavala o teoriji množic in uporabi le-te v višjih razredih osemletke in pri osnovah geometrije. Ivan Breclj je predaval o valovanju, Ciril Memon je pripravil nekaj poskusov o tlaku, Janez Ferbar pa je predaval o entropiji. Zadnjega predavanja se je udeležilo tudi več dijakov, ki se zanimajo za fiziko. V jeseni je Ciril Memon dvakrat predaval o molekularnih silah, združeno z številnimi eksperimenti. Na enem izmed rednih dvomesečnih sestankih so bili udeleženci seznanjeni s strokovno in nestrokovno zasedbo predavateljev matematike in fizike na Koprskem ter o politiki štipendiranja, ki premalo stimulira študij pedagoških smeri. V minulem letu so navezali tesne stike in se dogovorili za sodelovanje s kolegi iz Postojne, ki prav tako razmišljajo o ustanovitvi svoje podružnice. Člani koprške podružnice so precej pomagali pri organizaciji tekmovanj ter sodelovali pri matematičnih in fizikalnih krožkih na šolah.

Čeprav v *Novem mestu* deluje več zelo aktivnih članov našega društva, se podružnica zaradi prezaposlenosti njenih članov še ni docela uveljavila. Organizirali so le dve predavanji: Marija Glonar je govorila o dvoletnem eksperimentu modernizacije matematike v prvem razredu osnovne šole, Janez Ferbar pa je pokazal poučni film o entropiji. Pripravili so tudi drugi del maturitetnih nalog. Delo je z razumevanjem spremljala temeljna izobraževalna skupnost Novo mesto, zavod za zaposlovanje in občinski sindikalni svet.

Kolegi iz podružnice *Murska Sobota* pa so pripravili letošnji občni zbor.

Ustanovljene podružnice so precej prispevale k požitvi dela celotnega društva ter zboljšale stike med našimi člani.

Za prihodnja leta obstajajo možnosti, da bo ustanovljena podružnica v Postojni, na Jesenicah in v Ravnah na Koroškem.

Ivan Pavliha

POROČILO O DELU ODDELKA ZA MATEMATIKO INŠTITUTA ZA MATEMATIKO, FIZIKO IN MEHANIKO UNIVERZE V LJUBLJANI V ŠOLSKEM LETU 1971/72

Člani oddelka za matematiko so v tem šolskem letu zaključili naslednji raziskovalni nalogi, ki jih je financiral Sklad Borisa Kidriča:

I. Vidav in G. Tomšič: Homogeni operatorji in homogene spektralne mere.
N. Prijatelj: Študij univerzalnih algeber.

V delu so še naslednje naloge:

R. Jamnik: Ortogonalni sistemi slučajnih spremenljivk.
J. Grasselli: Adicijski izreki v teoriji grup in v teoriji števil.
E. Zakrajšek: Avtomatizacija faktorske analize.
S. Pahor in A. Suhadolc: Inverzni problemi albeda.
Z. Bohte: Stabilnost numeričnih procesov linearne algebre (nadaljevanje tri letne teme).

V letu 1972 je raziskovalna skupnost SR Slovenije odobrila naslednje raziskovalne naloge, ki so jih prijaviili naši člani:

T. Klinc: Razvoj po splošnih lastnih funkcijah sebi adjungiranih operatorjev in uporaba na problemih v transportnih teorijah nevtronov.

I. Vidav in J. Globevnik: Vektorske analitične funkcije s konstantno normo.

I. Vidav in P. Mizori-Oblak: Posplošenje Tamarkin-Smulyanovih izrekov na Banachove algebre.

J. Vrabec, A. Cedilnik in P. Legiša: Aciklične trirazsežne mnogoterosti.

Člani oddelka so poleg člankov v Obzorniku za matematiko in fiziko objavili še naslednje članke v zagrebškem Glasniku in drugih tujih revijah.

T. Klinc: Effect of lateral leakage on neutron waves. Transport theory and statistical physics. 1 (1), 1971, 41-58 str. (Skupaj z I. Kuščer).

J. Grad: Matrix balancing. The Computer Journal. 14, 1971, 280—283 str.

J. Globevnik: A note on $A^{1/2}$, where $-A$ generates a bounded semigroup. Glasnik matematički, 6 (26), 1971, 301—306 str.

V letošnjem postdiplomskem seminarju smo zopet poslušali več različnih predavanj, ki so jih obiskovali člani oddelka ter nekateri študenti matematike na fakulteti za naravoslovje in tehnologijo. Program je bil naslednji:

J. Vrabec: O ovojnem številu in njegovi uporabi v topologiji.*

J. Globevnik: Lomljene potence operatorjev.

A. Suhadolc: O neki kvadraturi.*

Z. Bohte: O redukciji algebraične enačbe.*

I. Vidav: O neki neenačbi za konveksne funkcije.

I. Kuščer: O matematičnih problemih pri sipanju molekul na površinah.

J. Vrabec: O Cantorjevi množici.*

T. Klinc: O razvoju po posplošenih lastnih funkcijah sebi adjungiranih operatorjev.

F. Križanič: O eliptičnih operatorjih.

Poleg centralnega seminarja je bil v letošnjem šolskem letu tudi občasni seminar računskega centra IMFM ter prvič tudi manjši specialni seminar. V prvem, katerega je vodil Z. Bohte, so bila naslednja predavanja:

E. Zakrajšek: Pseudoinverzna matrika in metoda najmanjših kvadratov.

J. Kozak: Potreben in zadosten pogoj za monotonost zaporedja približkov pri potenčni metodi.

E. Kramar: O operatorjih tipa (N).

A. Suhadolc: O Devidenkovi metodi.

E. Zakrajšek: O metodi končnih elementov.

J. Malešič: Reševanje cikličnega sistema linearnih enačb s konvolucijo.

Vodja drugega seminarja je bil J. Vrabec. Predavali pa so:

J. Vrabec: O snopih kolobarjev.

P. Legiša: Reprezentacija kolobarjev s prerezi snopov po Pierceu.

I. Vidav: Regularni kolobarji in moduli nad regularnimi kolobarji.

J. Rakovec: Vložitev regularnega kolobarja v regularen kolobar z enoto.

J. Vrabec: Telemanove harmonične algebre.

V gosteh smo imeli naslednje tuje predavatelje:

dr. V. Marić (Prirodno-matematički fakultet Novi Sad): O Bergmanovih integralnih operatorjih.

dr. K. Veselić (Prirodno-matematički fakultet Zagreb): O redukciji matrice na Jordanovu formu.

prof. T. R. Brahana (The University of Georgia): Application of cohomology to the isomorphism problem for certain groups.

prof. A. Crumeyrolle (Université Paul Sabatier, Toulouse): Cliffordove algebre in spinorji.

prof. L. Fox: (Oxford University): Singularities and free boundary problems in elliptic differential equations.

Profesor Fox je imel tudi ciklus šestnajstih predavanj v okviru meduniverzitetnega postdiplomskega študija z naslovom: Inherent and induced instability in numerical mathematics.

Nekateri predavatelji so svoja predavanja tudi napisali (označeno z*) in smo jih razmnožili v nekaj sto izvodih, katere lahko interesi dobijo v matematični knjižnici. Prav tako bo oddelek za matematiko IMFM nadaljeval z izdajanjem periodike PUBLICATIONS OF THE DEPARTMENT OF MATHEMATICS. Ker naši člani objavljajo svoja originalna znanstvena dela v zagrebškem Glasniku in drugih tujih revijah, bomo v omenjeni publikaciji objavljali le vse reprinte članov naših članov v tujih revijah. V letošnjem letu so izšle naslednje štiri številke za leta 1964/68, 1969, 1970 in 1971. Tiskali smo tudi magistrsko delo P. Petka z naslovom: Primarne kohomološke algebre. 1971. 126 str.

Člani našega oddelka so v zadnjem razdobju objavili še naslednje druge knjige in skripta:

V zbirki univerzitetnih učbenikov in monografij »MATEMATIKA« je izšla že četrta knjiga I. Vidav: Algebra. 1972. 340 str.

V knjižnici »SIGMA« so izhajali le ponatisi:

A. Vadnal: Funkcije I. (2. ponatis). 1971. 244 str.

N. Prijatelj: Matematične strukture I. 1971. 220 str.

I. Vidav: Rešeni in nerešeni problemi matematike. 1972. 136 str.

F. Križanič: Vektorji, matrike, tenzorji. (Vezano v dva dela.) 1972. 212 + 132 str.

J. Grad pa je soavtor knjige: Elektronski računalniki, 885 str., ki jo je leta 1971 izdala elektrotehniška zveza Slovenije.

Več članov oddelka se je tudi letos udeležilo rednega občnega zbora Društva matematikov, fizikov in astronomov SRS, ki je bil v Murski Soboti. V semestralnih počitnicah pa so pripravili v sodelovanju z društvom seminar za srednješolske profesorje z delovnim naslovom »Novo računstvo«.

V minulem šolskem letu je bil v inozemstvu le en član in sicer P. Petek na strokovnem izpopolnjevanju v Aarhusu s štipendijo danske vlade. Nova člana oddelka sta postala ing. T. Pisanski, programer RC ter ing. J. Zohil, asistent na elektro fakulteti. Oddelek šteje že 36 članov.

Ob koncu šolskega leta so člani oddelka po letnem obračunu izbrali tudi novega predstojnika. Dolžnost, ki jo je doslej opravljal N. Prijatelj, bo naslednji dve leti Z. Bohte.

Ciril Velkovrh

IZ NAŠIH AKTIVOV

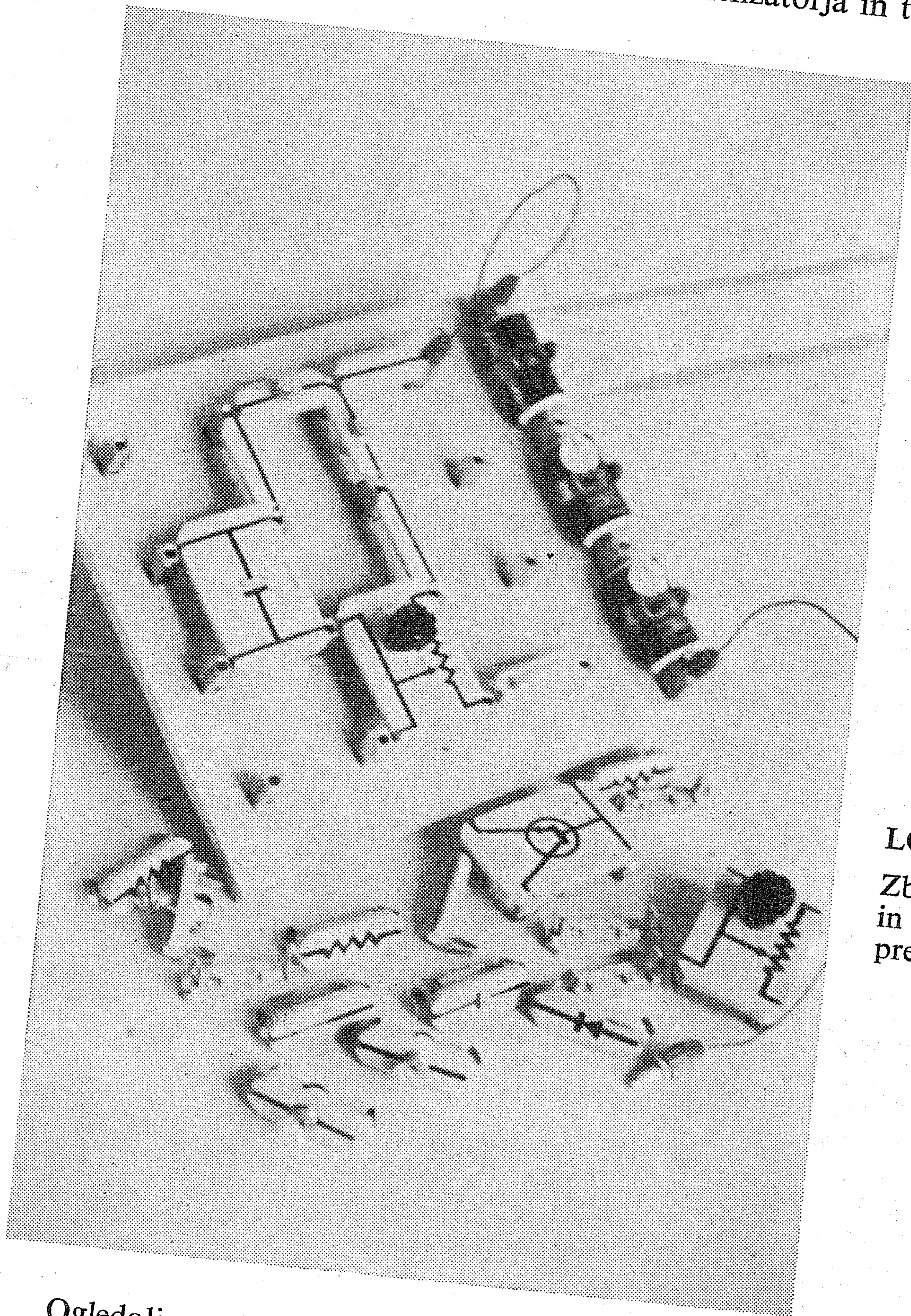
AKTIV MATEMATIKOV IN FIZIKOV

dne 17. maja 1972 v Novem mestu

Novomeška enota zavoda za šolstvo je ob sodelovanju podružnice DMFA iz Novega mesta organizirala 17. maja 1972 seminar za učitelje fizike na osnovnih šolah. Seminar je bil v fizikalnih laboratorijih na novomeški gimnaziji.

Udeleženci so se seznanili z učili (»LOCKTRONICS«) za pouk elektrike in elektronike angleške tovarne Lock, ki so namenjene vajam učencev. Pribor sestavlja podloga, na katero se zatikajo posamezni deli električnih tokokrogov: stikala, žarnice, upori, reostati, diode,

kondenzatorji, tranzistorji, fotoupori itd. Na podlogi so tudi trije členi po 1,5 V, kot izvir napetosti. Udeleženci so izvedli serijo poskusov s tem priborom: različne vezave električnih elementov, model varovalke, vloga kondenzatorja in tuljave v krogu izmeničnega toka itd.



LOCKTRONICS —

Zbirka za vaje učencev iz elektrike in elektronike v osnovni šoli: Fazni premik v krogu s kondenzatorjem.

Ogledali so si novo urejene prostore za fiziko in nekaj novih pridobitev fizikalnega kabineta: model očesa, van de Graaffov generator, zbirko za vaje iz optike, sestavljeno iz svetila in cilindričnih leč ter modelov iz pleksistekla, izvire napetosti za učence s svojimi izhodi, ki zadoščajo vsem zahtevam praktičnih vaj, valovno banjo, prav tako namenjeno vajam učencev idr.

Po razgovoru, ki je bil namenjen tekočim problemom, je bil seminar zaključen. Sodelovalo je 26 učiteljev.

Dušan Modic

V LETU 1971 SMO PREJELI NASLEDNJE REVIJE V ZAMENO
ZA OBZORNIK ZA MATEMATIKO IN FIZIKO

ACTA FAC. RERUM NATUR. UNIV. COMENIANAE. Bratislava, 1971.

1. Antropologia. Publ. 16, 17, 18. K
2. Botanica. Publ. 17, 19. K
3. Chimia. Publ. 15, 16. K
4. Mathematica. Publ. 25. K
5. Meteorologia. Publ. 4. K
6. Microbiologia. Publ. 1. K
7. Physica. Publ. 11. K
8. Physiologia plantarum. Publ. 3. K
9. Zoologia. Publ. 16. K
0. ACTA UNIV. CAROLINAE. Math. phys. Praha, 1969. Vol. 10. K 1970. Vol. 11. K
1. ALGEBRA I LOGIKA — seminar. Novosibirsk, 1970. T. 9. Vyp. 6. K 1971. T. 10. Vyp. 1—3.
2. ANALELE STI. UNIV. AL. I. CUZA IASI. Math. Iasi, 1971. T. 17. F. 1.
3. ANALELE UNIV. BUCARESTI. Acta logica. Bukaresti, 1963 — 1970. Nr. 6—13. K
4. ANALELE UNIV. BUCARESTI. Mat. mec. Bukaresti, 1970. Ann. 19. K.
5. ANALELE UNIV. TIMISOARA. Fiz. chem. Timisoara, 1970. T. 8. K.
6. ANALELE UNIV. TIMISOARA. Mat. Timisoara, 1970. T. 8. F. 1. K.
7. ANNALES ACAD. SCI. FENNICAE. Math. Helsinki, 1970. Nos 475, 478, 483. 1971. Nos 486—497, 499, 503.
8. ARKIV FOR MAT. Stockholm, 1952. Bd. 1. H. 6 1956. Bd. 3. H. 3—6. 1970. Bd. 8. H. 3. K 1971. Bd. 9. H. 1, 2.
9. ASTRONOMSKE EFEMERIDE. Ljubljana, 1971. K
0. BERICHTER DER GESELLSCHAFT FÜR MATH. UND DATENVERARBEITUNG. Bonn, 1971. Nrs 37—51. K.
1. BILTEN NA DRUŠTVATO NA MAT. FIZ. OD NR MAKEDONIJA. Skopje, 1969. T. 20. K.
2. BOLETIN ACAD. CIEN. FIS. MAT. NATUR. Caracas, 1970. Anno 30. Nos 87—89. K.
3. BOLETIN SOC. MAT. MEXICANA. Mexico, 1969. Vol. 14. Nu. 2. K 1970. Vol. 15. K 1971. Vol. 16. Nu. 1.
4. BONNER MATH. SCHRIFTEN. Bonn, 1958. Nr. 6. 1970. Nr. 10. 1963. Nrs 16, 21. 1969. Nrs 38—41. K 1970. Nrs 43—45. K.
5. BULLETIN ACAD. POLONAIS SCI. Math. astronom. phys. Varsovie, 1971. Vol. 19. K.
6. BULLETIN DISASTER PREVENTION RES. INST. Kyoto, 1969/70. Vol. 20, P. 1—5. 1971. Vol. 21. P. 1, 2.
7. BULLETIN SCIENTIFIQUE. Zagreb, 1971. T. 16. K.
8. COMMENTATIONES MATH. UNIV. CAROLINAE. Praha, 1971. T. 12. K.
9. DIFFERENCIAL'NYE URAVENIJA. Minsk, 1971. T. 7. K.
0. ECONOMIC COMPUTATION CYBERNETICS STUD. RES. Bucaresti, 1970. T. 4. K 1971. T. 1—4. K.
1. ELEKTROTEHNIŠKI VESTNIK. Ljubljana, 1971. Letnik 38. Štev. 1—10.
2. FARMACEVTSKI VESTNIK. Ljubljana, 1971. Letnik 22. K.
3. FIZIKA. Zagreb, 1971. Vol. 3. K.
4. FOTOKEMIJSKA INDUSTRIJA. Zagreb, 1971. g. 18. br. 1.
5. GLASNIK MATEMATIČKI. Zagreb, 1971. T. 6. K.
6. GODIŠEN ZBORNIK. Mat. fiz. hem. Skopje, 1970. knj. 20 K 1971. knj. 21. K.
7. GODIŠEN ZBORNIK. Skopje, 1968. knj. 2. K 1969. knj. 3. K.
8. GODIŠNIK VIŠŠ. TEHN. UČBENI ZAVED. Fizika. Sofija, 1967. T. 4. Liv. 1. 1968. T. 5. Liv. 1, 2. 1969. T. 6. Liv. 1, 2.
9. GODIŠNIK VIŠŠ. TEHN. UČEBNI ZAVED. Matematika. Sofija, 1967. Knj. 1, 2. K 1968. Knj. 3.
0. IDRIJSKI RAZGLEDI. Idrija, 1971. Letnik 16. K.
1. HIROSHIMA MATH. J. Hiroshima, 1971. Vol. 1. Nu. 1.
2. INTERNATIONAL MATH. NACHRICHTEN. Wien, 1971. Nrs 97, 98, 99. K.
3. IZVESTIJA MAT. INSTITUT. Sofija, 1959/60. T. 4. K.
4. JIPDEC REPORT. Tokyo, 1971. No. 10, 11.
5. JOURNAL FRANKLIN INSTITUTE. Philadelphia, 1970. Vol. 290. Nu. 6. K Vol. 291. K 1971. Vol. 292. K. Vol. 293. Nus 1—3.
6. JOURNAL SCI. HIROSHIMA UNIV. Hiroshima, 1961. Vol. 25. Nu. 2. K 1962. Vol. 26. K 1963. Vol. 27. K 1970. Vol. 34. Nu. 2. K.
7. LECTII GEOMETRIE DIFFERENTIALE. Bucaresti, 1968. Vol. 4. K.

48. MATEMATYCKI ČASOPIS. Bratislava, 1971. r. 21. K.
49. MATEMATIČKI LIST ZA UČENIKE OSNOVNE ŠKOLE. Beograd, 1970/71. g. 5. K.
50. MATEMATIČKI VESNIK. Beograd, 1971. Knj. 8. K.
51. MATEMATIČKO-FIZIČKI LIST ZA UČENIKE SREDNJIH ŠKOLA. Zagreb, 1970/71 g. 21. K.
52. MATEMATIKA A FYZIKA VE ŠKOLE. Praha, 1970/71. r. 1. K.
53. MATEMATIKA VE ŠKOLE. Praha, 1969/70. r. 20. č. 8, 10. K.
54. MATHEMATICAL GAZETTE. London, 1971. Vol. 55. Nus 391, 392, 393.
55. NATURA. Plovdiv, 1971. T. 3. F. 1. K.
56. NAUČNI TRUDOVE. Plovdiv, 1971. T. 9. K.
57. NOTICES AMER. MATH. SOC. Providence, 1970. Vol. 17. K 1971. Vol. 18. K.
58. OSAKČA J. MATH. Osaka, 1970. Vol. 7. No. 2. K 1971. Vol. 8. Nos. 1, 2, 3. K.
59. POROČILO — Inštitut »J. Štefan«. Ljubljana, 1971. Nos 266—275.
60. PROCEEDINGS AMER. MATH. SOC. Menasha, 1970. Vol. 24, 25, 26. K 1971. Vol. 27, 28, 29, 30. K.
61. PROCEEDINGS FAC. SCI. TOKAI UNIV. Tokyo, 1971. Vol. 6. K.
62. PROCEEDINGS MATH. PHYS. SOC. EGYPT. Cairo, 1968. Vol. 32. K.
63. PROTEUS. Ljubljana, 1970/71. Letnik 33. K.
64. PUBLICATIONS INST. MATH. Belgrade, 1971. T. 11. K.
65. PUBLIKACIJE ELEKTROTEHNIČKOG FAK. Beograd, 1970. Nos 302—337.
66. RAZPRAVE. Društvo meteorologov Slovenije. Ljubljana, 1971. Letnik 13. K.
67. REPORT. Inštitut »J. Štefan«. Ljubljana, 1971. Nos 595—604.
68. REVISTA MAT. HISPANO-AMER. Madrid, 1971. T. 31. K.
69. REVISTA UNION MAT. ARGENTINA Y ASOC. FIS. ARGENTINA. Buenos Aires, 1970. Vol. 25. Nu. 1/2.
70. REVUE ROUMANE MATH. PURES APPL. Bucarest, 1971. T. 16. K.
71. REVISTA MAT. UNIV. PARMA. Parma, 1960. Vol. 1. K 1961. Vol. 2. K 1962. Vol. 3. K 1969. Vol. 10. K.
72. ROZHLEDY MAT. FIZ. Praga, 1970/1971. T. 49. K.
73. RUDARSKO-METALURŠKI ZBORNIK. Ljubljana, 1971. Štev. 2/3.
74. SODOBNA PEDAGOGIKA. Ljubljana, 1971. Letnik 22. K.
75. SPISY. Brno, 1970. K 1969. K.
76. STUDIA UNIV. BABES-BOLYAI. Mat. phys. Cluj, 1970. T. 15. K.
77. STUDII CERC. MATH. Bucuresti, 1971. T. 23. K.
78. UČENYE ZAP. TARTUSKOGO GOS. UNIV. Math. meh. Tartu, 1970. T. 9. K.
79. UKRAINSKIJ MAT. ŽURNAL. Kiev, 1971. T. 23. K.
80. USPEHI MATEMATIČESKIH NAUK. Moskva, 1971. T. 26. K.
81. VARILNA TEHNIKA. Ljubljana, 1971. Letnik 20. K.
82. VESCI AN BSSR. Fiz. mat. Minsk, 1971. K.
83. VESTNIK LENINGRADSKOGO UNIV. Fiz. hem. Leningrad, 1971. Vyp. 1—3.
84. VESTNIK MOSKOVSKOGO UNIV. Mat. meh. Moskva, 1971. T. 26. No. 1, 2, 3, 5, 6.
85. VZGOJA IN IZOBRAŽEVANJE. Ljubljana, 1971. Letnik 2. Štev. 1—7.
86. ZBORNIK RADOVA PRIR. MAT. FAK. Fiz. Novi Sad, 1971. br. 1.
87. ZDRAVSTVENI VESTNIK. Ljubljana, 1971. Letnik 40. K.
88. ŽURNAL PRIKLADNOJ SPEKTROSKOPII. Moskva, 1971. T. 14. Vyp. 3, 4, 5.

Ciril Velkovrh

Pred kratkim je izšel ponatis matematičnega priročnika

Stanko Uršič:

ŠTIRIMESTNI LOGARITMI IN DRUGE TABELE

Če omenjene knjige ni v prodaji v knjigarnah, lahko dijaki, člani društva in šole pošljejo skupno naročilo za več izvodov na naslov: **KOMISIJA ZA TISK DMFA SRS, 61001 Ljubljana, Jadran-ska c. 19, pp. 227, tel. št. 61-564/22.** Cena v platno vezanega izvoda je 20.— din. Ustrezni zne-sek nakažite na ŽR št. 501-8-240/1.

Za komisijo za tisk DMFA SRS

Ciril Velkovrh