1.5.1 Analiza zaokrožitvenih napak za produkt $n+1$ števil

Računamo produkt $p = x_0 x_1 \cdots x_n$ predstavljivih števil $x_0, x_1, \ldots, x_n$.

Eksaktni algoritem:

$p_0 = x_0$
$i = 1, \ldots, n$
$\quad p_i = p_{i-1} x_i$
$p = p_n$

Dejanski algoritem:

$\widehat{p}_0 = x_0$
$i = 1, \ldots, n$
$\quad \widehat{p}_i = \widehat{p}_{i-1} x_i (1+\delta_i), \quad |\delta_i| \leq u$
$\widehat{p} = \widehat{p}_n$

Dobimo

$$\widehat{p} = p(1+\gamma) = p(1+\delta_1)\cdots(1+\delta_n).$$

Velja

$$(1-u)^n \leq 1+\gamma \leq (1+u)^n.$$

Ocenimo

$$(1+u)^n = 1 + \binom{n}{1}u + \binom{n}{2}u^2 + \cdots = 1 + nu + \mathcal{O}(u^2),$$

$$(1-u)^n \geq 1 - nu. \quad \text{(indukcija)}$$

Če je $nu \ll 1$ ocenimo $|\gamma| \leq nu$. Računanje produkta $n+1$ števil je direktno in obratno stabilno.

1.5.2 Analiza zaokrožitvenih napak za skalarni produkt

Imamo dva vektorja predstavljivih števil $x = (x_1 \ \cdots \ x_n)^T$ in $y = (y_1 \ \cdots \ y_n)^T$, računamo pa $s = y^T x = \sum_{i=1}^{n} x_i y_i$.

Eksaktni algoritem:

$s_0 = 0$

$i = 1, \ldots, n$

$\quad p_i = x_i y_i$

$\quad s_i = s_{i-1} + p_i$

$s = s_n$

Dejanski algoritem:

$\widehat{s}_0 = 0$

$i = 1, \ldots, n$

$\quad \widehat{p}_i = x_i y_i (1 + \alpha_i), \ |\alpha_i| \leq u$

$\quad \widehat{s}_i = (\widehat{s}_{i-1} + \widehat{p}_i)(1 + \beta_i), \ |\beta_i| \leq u$

$\widehat{s} = \widehat{s}_n$

Obratna analiza vrne $\widehat{s} = \sum_{i=1}^{n} x_i y_i (1 + \gamma_i)$, kjer je

$$1 + \gamma_1 = (1 + \alpha_1)(1 + \beta_2) \cdots (1 + \beta_n)$$

in

$$1 + \gamma_i = (1 + \alpha_i)(1 + \beta_i) \cdots (1 + \beta_n), \ i = 2, \ldots, n.$$

Tako dobimo ocene $|\gamma_1| \leq nu$ in $|\gamma_i| \leq (n - i + 2)u$ za $i = 2, \ldots, n$. To pomeni, da je $\widehat{s}$ točni skalarni produkt relativno malo zmotenih vektorjev x in y. Računanje skalarnega produkta je obratno stabilno.

Direktna analiza za izračun skalarnega produkta

Pri direktni analizi najprej izračunamo absolutno napako:

$$\widehat{s} - s = \sum_{i=1}^{n} x_i y_i \gamma_i,$$

torej

$$|\widehat{s} - s| \leq \sum_{i=1}^{n} |x_i| \cdot |y_i| \cdot |\gamma_i| \leq nu \sum_{i=1}^{n} |x_i| \cdot |y_i| = nu|y|^T|x|.$$

Dobimo

$$\left|\frac{\widehat{s} - s}{s}\right| \leq \frac{|y|^T|x|}{|y^T x|} nu.$$

Če so vsi $x_i y_i$ enakega predznaka, dobimo $\left|\frac{\widehat{s}-s}{s}\right| \leq nu$ in računanje je direktno stabilno, sicer pa imamo v primeru, ko sta vektorja skoraj pravokotna, lahko veliko relativno napako.

V splošnem torej računanje skalarnega produkta ni direktno stabilno.

1.5.3 Računanje vrednosti polinoma po Hornerju

Računamo vrednost polinoma $p(t) = a_0 t^n + a_1 t^{n-1} + \cdots + a_n$ v točki x.

Eksaktni algoritem je:

$p_0 = a_0$
$i = 1, \ldots, n$
$\quad p_i = p_{i-1} x + a_i$
$p = p_n$

Dejanski algoritem pa:

$\widehat{p}_0 = a_0$
$i = 1, \ldots, n$
$\quad \widehat{p}_i = (\widehat{p}_{i-1} x(1+\alpha_i) + a_i)(1+\beta_i)$
$\widehat{p} = \widehat{p}_n$

Dobimo

$$\widehat{p} = a_0 x^n (1+\gamma_0) + a_1 x^{n-1}(1+\gamma_1) + \cdots + a_n (1+\gamma_n),$$

kjer lahko ocenimo $|\gamma_i| \leq 2nu$ za $i = 0, \ldots, n$. Računanje vrednosti polinoma po Hornerju je obratno stabilno, saj dobimo vrednost bližnjega polinoma s koeficienti $a_i(1+\gamma_i)$ namesto a_i v točki x.

Iz absolutne napake $\widehat{p} - p = a_0 x^n \gamma_0 + a_1 x^{n-1} \gamma_1 + \cdots + a_n \gamma_n$ sledi

$$\frac{|\widehat{p} - p|}{|p|} \leq \frac{2nu(|a_0||x^n| + |a_1||x^{n-1}| + \cdots + |a_n|)}{|a_0 x^n + \cdots + a_n|}.$$

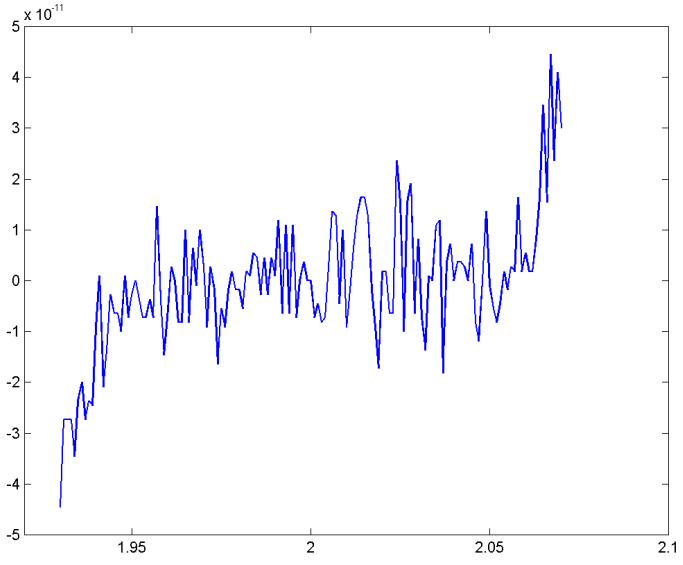
Računanje vrednosti polinoma po Hornerju ni direktno stabilno.

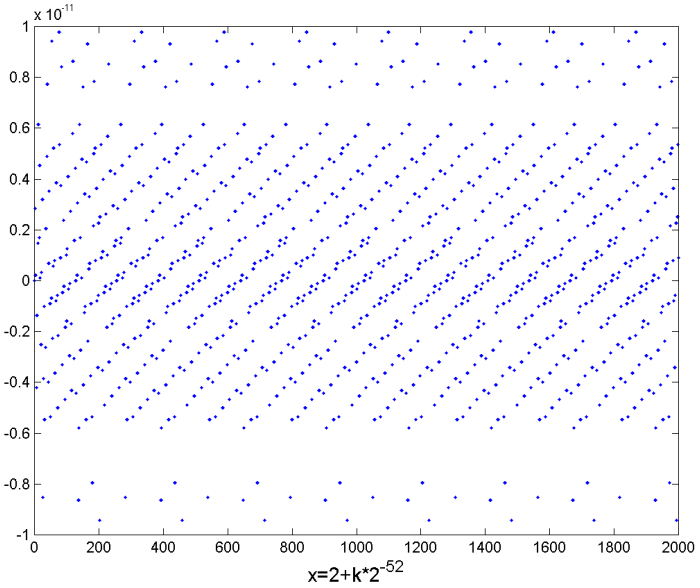
Računanje vrednosti polinoma po Hornerju - 2.del

Za predstavo o dobljenih ocenah si poglejmo računanje polinoma $(x-2)^9$ oz.

$$x^9 - 18x^8 + 144x^7 - 672x^6 + 2016x^5 - 4032x^4 + 5376x^3 - 4608x^2 + 2304x - 512$$

v okolici točke 2. Z Matlabom dobimo





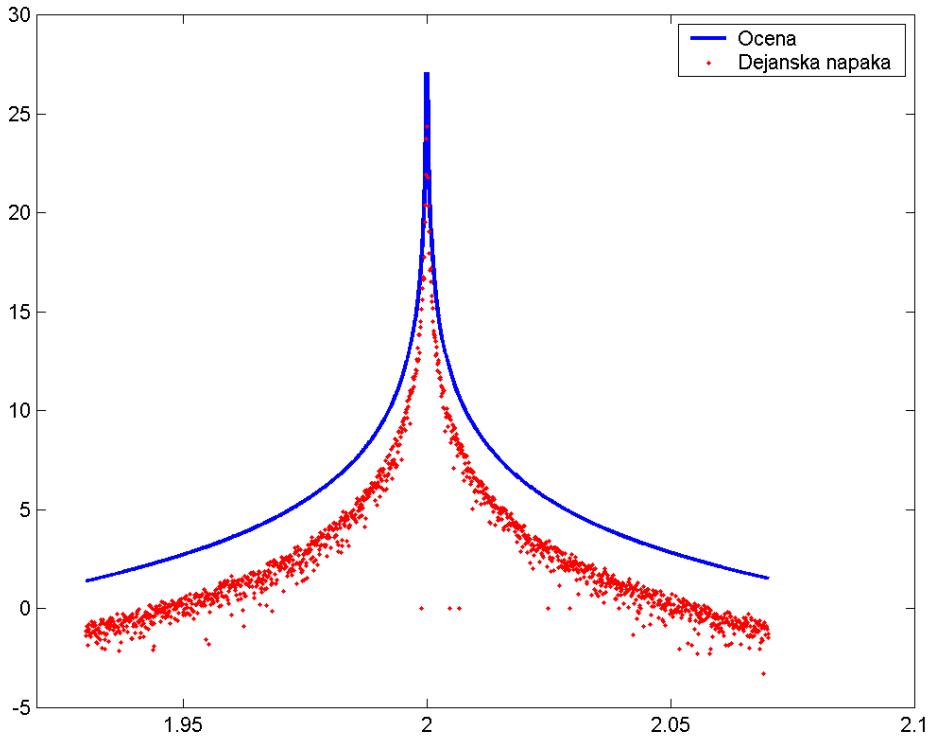
Napake niso povsem naključne, saj v bližini 2 odkrijemo vzorec.

Računanje vrednosti polinoma po Hornerju - 3.del

V algoritem lahko vstavimo sprotno računanje ocene:

$p_0 = a_0$
$e_0 = |a_0|$
$i = 1, \ldots, n$
$\quad p_i = p_{i-1}x + a_i$
$\quad e_i = e_{i-1}|x| + |a_i|$
$p = p_n$
$e = 2nu\frac{e_n}{|p|}$

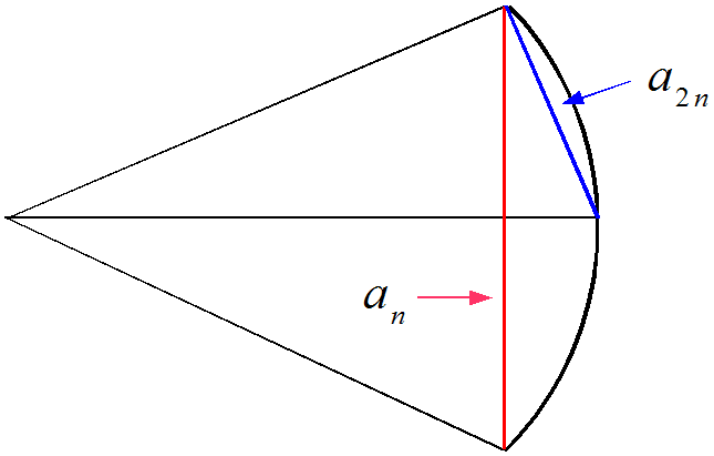
e je ocena za relativno napako.



Graf prikazuje resnično napako in oceno za napako v logaritemski skali. Ocena je res zgornja meja in dejanska napaka je manjša ali enaka, vendar zgled kaže, da so v bližini točke 2 napake res večje in tudi obnašajo se tako, kot napoveduje ocena.

1.6.1 Poučni primeri - računanje števila π

π je limita obsega S_n pravilnega mnogokotnika, včrtanega v krog s polmerom $r = \frac{1}{2}$. Naj bo a_n stranica pravilnega n-kotnika. Poiščimo zvezo med a_n in a_{2n}:



Velja

$$a_{2n} = \sqrt{\left(\frac{a_n}{2}\right)^2 + \left(\frac{1}{2} - \sqrt{\frac{1}{4} - \left(\frac{a_n}{2}\right)^2}\right)^2} = \sqrt{\frac{1 - \sqrt{1 - a_n^2}}{2}},$$

od tod pa iz $S_n = na_n$ sledi

$$S_{2n} = 2na_{2n} = 2n\sqrt{\frac{1 - \sqrt{1 - \left(\frac{S_n}{n}\right)^2}}{2}}.$$

Računanje števila π, 2.del

Začnemo pri $S_6 = 3$ in uporabljamo formulo $S_{2n} = 2n\sqrt{\frac{1-\sqrt{1-\left(\frac{S_n}{n}\right)^2}}{2}}$.

n	Sn	n	Sn
6	3.0000000	768	3.1430728
12	3.1058285	1536	3.1486604
24	3.1326292	3072	3.1374750
48	3.1393456	6144	3.1819806
96	3.1410186	12288	3.0000000
192	3.1414995	24576	4.2426405
384	3.1416743	49152	0.0000000

Formula odpove, saj pride do odštevanja skoraj enako velikih števil, napaka pa se množi z $2n$. V zgornji tabeli (enojna natančnost) so napačne decimalke rdeče.

Kadar imamo nestabilen postopek, nam ne pomaga niti računanje z večjo natančnostjo. Prava rešitev je preurediti postopek tako, da se med računanjem ne izgublja natančnost.

Računanje števila π, 3.del

Za stabilno računanje je potrebno formulo preurediti. Stabilna oblika je

$$S_{2n} = 2n \sqrt{\frac{\left(1 - \sqrt{1 - \left(\frac{S_n}{n}\right)^2}\right)\left(1 + \sqrt{1 - \left(\frac{S_n}{n}\right)^2}\right)}{2\left(1 + \sqrt{1 - \left(\frac{S_n}{n}\right)^2}\right)}} = S_n \sqrt{\frac{2}{1 + \sqrt{1 - \left(\frac{S_n}{n}\right)^2}}}\,.$$

Sedaj dobimo pravilne rezultate:

n	Sn	n	Sn
6	3.0000000	768	3.1415839
12	3.1058285	1536	3.1375904
24	3.1326287	3072	3.1415920
48	3.1393502	6144	3.1415925
96	3.1410320	12288	3.1415925
192	3.1414526	24576	3.1415925
384	3.1415577	49152	3.1415925

1.6.2 Seštevanje Taylorjeve vrste za e^{-x}

Vemo, da je

$$e^{-x} = \sum_{n=0}^{\infty} (-1)^n \frac{x^n}{n!}$$

in da vrsta konvergira za vsak $x \in \mathbb{C}$. Če pa to vrsto seštevamo numerično po vrsti, potem za $x > 0$ ne dobimo najboljših rezultatov.

x	e^{-x}	vrsta	relativna napaka
1	$3.678795 \cdot 10^{-1}$	$3.678794 \cdot 10^{-1}$	$1.4 \cdot 10^{-7}$
2	$1.353353 \cdot 10^{-1}$	$1.353353 \cdot 10^{-1}$	$2.3 \cdot 10^{-7}$
3	$4.978707 \cdot 10^{-2}$	$4.978702 \cdot 10^{-2}$	$9.3 \cdot 10^{-7}$
4	$1.831564 \cdot 10^{-2}$	$1.831531 \cdot 10^{-2}$	$1.8 \cdot 10^{-5}$
5	$6.737947 \cdot 10^{-3}$	$6.737477 \cdot 10^{-3}$	$7.0 \cdot 10^{-5}$
6	$2.478752 \cdot 10^{-3}$	$2.477039 \cdot 10^{-3}$	$6.9 \cdot 10^{-4}$
7	$9.118820 \cdot 10^{-4}$	$9.139248 \cdot 10^{-4}$	$2.2 \cdot 10^{-3}$
8	$3.354626 \cdot 10^{-4}$	$3.485951 \cdot 10^{-4}$	$3.9 \cdot 10^{-2}$
9	$1.234098 \cdot 10^{-4}$	$1.799276 \cdot 10^{-4}$	$4.6 \cdot 10^{-1}$
10	$4.539992 \cdot 10^{-5}$	$-7.266693 \cdot 10^{-5}$	$2.6 \cdot 10^{-0}$

Seštevanje Taylorjeve vrste za e^{-x}, 2. del

Razlog je, da zaporedje členov vrste alternira, poleg tega pa po absolutni vrednosti nekaj časa naraščajo, preden začnejo padati proti 0. Ko so členi največji, se zameglijo majhne decimalke, ki ostanejo netočne do konca računanja.

n	a_n	s_n	n	a_n	s_n
0	1.000000	1.000000	20	41.103188	13.396751
1	−10.000000	−9.000000	21	−19.572947	−6.176195
2	50.000000	41.000000	22	8.896794	2.720599
3	−166.666672	−125.666672	23	−3.868171	−1.147572
4	416.666687	291.000000	24	1.611738	0.464166
5	−833.333374	−542.333374	25	−0.644695	−0.180529
6	1388.888916	846.555542	26	0.247960	0.067430
7	−1984.127075	−1137.571533	27	−0.091837	−0.024407
8	2480.158936	1342.587402	28	0.032799	0.008392
9	−2755.732178	−1413.144775	29	−0.011310	−0.002918
10	2755.732178	1342.587402	30	0.000380	0.000852
11	−2505.211182	−1162.623779	31	−0.001216	−0.000364
12	2087.676025	925.052246	32	0.000380	0.000016
13	−1605.904663	−680.852417	33	−0.000115	−0.000099
14	1147.074707	466.222290	34	0.000034	−0.000065
15	−764.716492	−298.494202	35	−0.000010	−0.000075
16	477.947815	179.453613	36	0.000003	−0.000072
17	−281.145782	−101.692169	37	−0.000001	−0.000073
18	156.192108	54.499939	38	0.000000	−0.000073
19	−82.206375	−27.706436	39	−0.000000	−0.000073

1.6.3 Računanje I_{10}

Integrale $I_n = \int_0^1 x^n e^{x-1} dx$, $n = 0, 1, \ldots$, lahko računamo rekurzivno preko formule

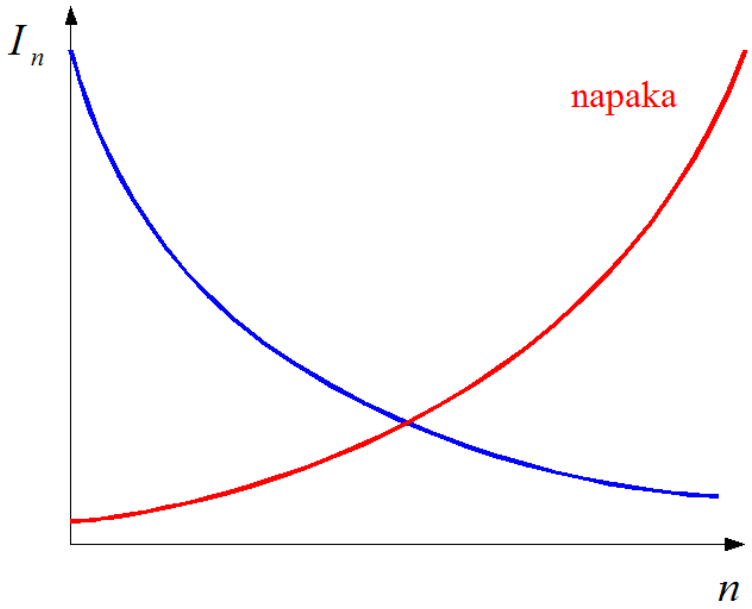
$$I_n = x^n e^{x-1}\Big|_0^1 - n \int_0^1 x^{n-1} e^{x-1} dx = 1 - nI_{n-1},$$

saj poznamo začetno vrednost $I_0 = 1 - e^{-1}$. V enojni natančnosti dobimo

n	I_n	n	I_n
0	0.6321205	7	0.1124296
1	0.3678795	8	0.1005630
2	0.2642411	9	0.0949326
3	0.2072767	10	0.0506744
4	0.1708932	11	0.4425812
5	0.1455340	12	−4.3109741
6	0.1267958	13	57.0426636

Razlog je v formuli $I_n = 1 - nI_{n-1}$. Napaka pri členu I_{n-1} se pomnoži z n in torej po absolutni vrednosti hitro narašča, točne vrednosti I_n pa padajo.

Računanje I_{10}, 2. del



$$I_n = 1 - nI_{n-1}$$

Če računamo v obratni smeri: $I_{n-1} = \frac{1-I_n}{n}$, se napaka v vsakem koraku deli z n. Če začnemo pri nekem dovolj velikem členu, lahko z začetnim $I_n = 0$ izračunamo vse začetne člene dovolj natančno. Če začnemo z $I_{26} = 0$ tako dobimo (v enojni natančnosti) vse člene od I_{12} do I_0 na vse decimalke točno.

n	I_n	n	I_n
0	0.6321205	8	0.1009320
1	0.3678795	9	0.0916123
2	0.2642411	10	0.0838771
3	0.2072766	11	0.0773522
4	0.1708934	12	0.0717733
5	0.1455329	13	0.0669477
6	0.1268024	⋮	⋮
7	0.1123835	26	0.0000000

1.6.4 Zaporedno korenjenje in kvadriranje

Vzamemo število $x > 0$, ga 80-krat korenimo in nato 80-krat kvadriramo. Za $x \geq 1$ dobimo 1, za $0 < x < 1$ pa dobimo 0! Dogajanje razložimo na modelu kalkulatorja HP 48G, kjer je baza desetiška, dolžina mantise pa je 12.

Za $0 < x < 1$ velja $\sqrt{x} > x$. Največje predstavljivo število, ki je še manjše od 1, je $1 - 10^{-12}$ oziroma $0.\underbrace{9\ldots9}_{12}$. Zaradi $\sqrt{1+x} = 1 + \frac{1}{2}x - \frac{1}{8}x^2 + \cdots$ velja

$$\sqrt{1 - 10^{-12}} = 1 - \frac{1}{2}10^{-12} - \frac{1}{8}10^{-24} + \cdots = 0.\underbrace{9\ldots9}_{12}4\underbrace{9\ldots9}_{11}87\ldots,$$

in število se zaokroži na $0.\underbrace{9\ldots9}_{12}$. Tako s korenjenjem nikoli ne pridemo do 1, ko pa kvadriramo, je število vedno manjše, dokler ne pride do podkoračitve.

Za $x > 1$ je $1 + 10^{-11}$ prvo predstavljivo število, ki je večje od 1. Tu dobimo

$$\sqrt{1 + 10^{-11}} = 1 + \frac{1}{2}10^{-11} - \frac{1}{8}10^{-22} + \cdots = 1.\underbrace{0\ldots0}_{11}4\underbrace{9\ldots9}_{10}87\ldots,$$

kar se zaokroži na 1. Tako s korenjenjem pridemo do 1, s kvadriranjem pa se to ne spremeni.

1.6.5 Seštevanje številske vrste

Znano je, da velja

$$\sum_{k=1}^{\infty} \frac{1}{k^2} = \frac{\pi^2}{6} = 1.644934066848\ldots$$

Kako bi to sešteli, če tega ne bi vedeli?

- Prištevamo člene, dokler se vsota ne spreminja več. V enojni natančnosti tako dobimo $1.64472532\ldots$, vrednost pa dosežemo pri $k = 4096$. V tem koraku delni vsoti ≈ 1.6 prištevamo 2^{-24}.

- Seštevamo v obratnem vrstnem redu od malih cifer proti velikim. Izkaže se, da bi za približek 1.64493406 z 8 točnimi decimalkami morali sešteti 10^9 členov!

Seveda nobeden izmed zgornjih dveh načinov ni primeren, se pa da z ustrezno numerično metodo dobiti dovolj natančen približek z uporabo relativno malo členov.

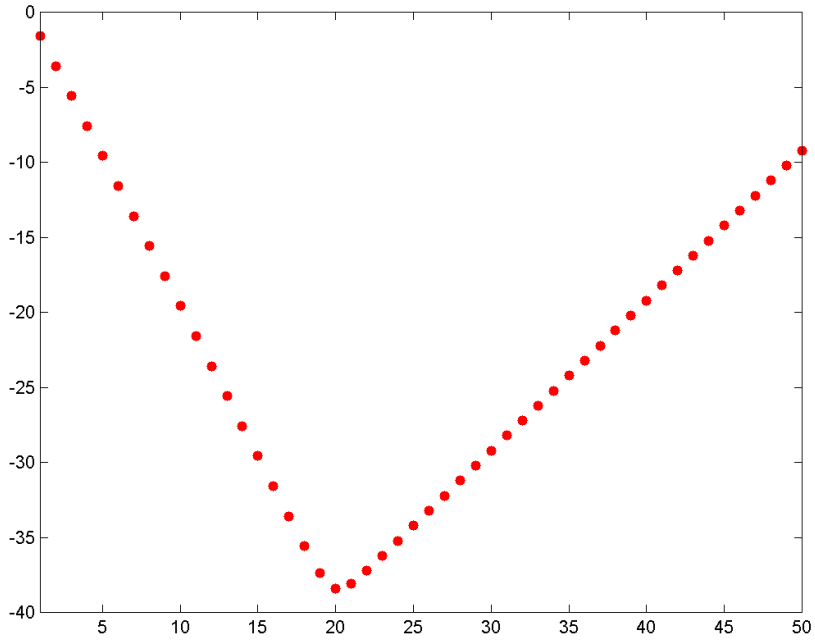
1.6.6 Tričlenska rekurzivna formula

Dana je tričlenska rekurzivna formula

$$x_{k+1} = 2.25x_k - 0.5x_{k-1}$$

in začetni dve vrednosti $x_1 = 1/3$ in $x_2 = 1/12$. Pri teh pogojih je splošna rešitev $x_k = \frac{4}{3} \cdot 4^{-k}$.

Numerično dobimo naslednje točke (logaritemska skala)



Tričlenska rekurzivna formula, 2.del

Odgovor se skriva v dejstvu, da je splošna rešitev dane diferenčne enačbe

$$x_k = \alpha \left(\frac{1}{4}\right)^k + \beta 2^k.$$

Zaradi zaokrožitvenih napak je $\beta \neq 0$ in prej ali slej drugi člen prevlada.

Zaradi zaokrožitvenih napak imamo na začetku

$$\begin{aligned} x_1 &= \frac{1}{3}(4^0 + 2^{-56}) \\ x_2 &= \frac{1}{3}(4^{-1} + 2^{-55}) \end{aligned}$$

Splošna (točna) rešitev za zgornja x_1, x_2 je

$$x_k = \frac{1}{3}(4^{1-k} + 2^{k-57}),$$

do preobrata pride ravno ko je $4^{1-k} = 2^{k-57}$ oziroma $k \approx 20$.

1.6.7 Reševanje kvadratne enačbe

Rešitvi kvadratne enačbe $ax^2 + bx + c = 0$ sta podani s formulo

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$$

$a = 1.2345678$, $b = 76543210.5$, $c = 0.1122334455$, dvojna natančnost $\Longrightarrow$

$$x_1 = -6.200000558900046 \cdot 10^7, \quad x_2 = -6.034970778375905 \cdot 10^{-9},$$

točni ničli pa sta

$$\widetilde{x}_1 = -6.200000558900046 \cdot 10^7, \quad \widetilde{x}_2 = -1.466275647008561 \cdot 10^{-9}.$$

Težava se je pojavila pri odštevanje približno enako velikih števil pri računanju x_2. Rešitev je, da eno rešitev izračunamo po formuli (odvisno od predznaka b), drugo pa dobimo iz Vietove formule $x_1 x_2 = c/a$.

Tako iz $x_2 = c/(ax_1)$ dobimo pravilno $x_2 = -1.466275647008561 \cdot 10^{-9}$.

1.6.8 Računanje $(e^x - 1)/x$

Računamo $f(x) = (e^x - 1)/x$ v bližini $x = 0$, kjer je $\lim_{x \to 0} f(x) = 1$.

Algoritem 1

if $x = 0$
    $y = 1$
else
    $y = (e^x - 1)/x$
end

Algoritem 2

$z = e^x$
if $z = 1$
    $y = 1$
else
    $y = (z - 1)/\ln z$
end

x	Algoritem 1	Algoritem 2
10^{-7}	1.000000049433680	1.000000050000002
10^{-8}	$9.999999939225290 \cdot 10^{-1}$	1.000000005000000
10^{-9}	1.000000082740371	1.000000000500000
10^{-10}	1.000000082740371	1.000000000050000
10^{-11}	1.000000082740371	1.000000000005000
10^{-12}	1.000088900582341	1.000000000000500
10^{-13}	$9.992007221626408 \cdot 10^{-1}$	1.000000000000050
10^{-14}	$9.992007221626408 \cdot 10^{-1}$	1.000000000000005
10^{-15}	1.110223024625156	1.000000000000000
10^{-16}	0.000000000000000	1.000000000000000

Primerjava algoritmov za računanje $(e^x - 1)/x$ pri $x = 7 \cdot 10^{-14}$

Pri prvem algoritmu dobimo

$$y = fl\left(\frac{e^x - 1}{x}\right) = fl\left(\frac{6.994405055138486 \cdot 10^{-14}}{7.000000000000001 \cdot 10^{-14}}\right) = 0.9992007221626 4.$$

Pri drugem algoritmu pa

$$y = fl\left(\frac{e^x - 1}{\ln(e^x)}\right) = fl\left(\frac{6.994405055138486 \cdot 10^{-14}}{6.994405055138241 \cdot 10^{-14}}\right) = 1.000000000000004.$$

To je sicer lepo, vendar! Točne vrednosti, ki bi morale nastopati v drugem algoritmu so

$$y = fl\left(\frac{e^x - 1}{\ln(e^x)}\right) = fl\left(\frac{7.000000000000245 \cdot 10^{-14}}{7.000000000000001 \cdot 10^{-14}}\right) = 1.000000000000004.$$

Kako je možno, da iz dveh netočnih vrednosti dobimo točno? Pokažimo, da je drugi algoritem stabilen.

Analiza stabilnega algoritma za računanje $(e^x - 1)/x$

Najprej izračunamo $\widehat{z} = e^x(1+\delta)$, kjer je $|\delta| \leq u$.

Denimo, da je $\widehat{z} = 1$. *To pomeni* $e^x(1+\delta) = 1$, *torej*

$$x = -\ln(1+\delta) = -\delta + \delta^2/2 - \delta^3/3 + \cdots,$$

torej je pravilno zaokrožena vrednost $f(x)$ *na* 1, *saj je* $f(x) = 1 + x/2 + x^2/6 + \cdots$.

Če je $\widehat{z} \neq 1$, potem računamo $y = (z-1)/\ln z$. Numerično velja

$$\widehat{y} = \frac{(\widehat{z}-1)(1+\epsilon_1)}{\ln \widehat{z}(1+\epsilon_2)}(1+\epsilon_3), \qquad |\epsilon_1|, |\epsilon_2|, |\epsilon_3| \leq u.$$

Če vpeljemo $w = \widehat{z} - 1$, dobimo

$$g(\widehat{z}) := \frac{\widehat{z}-1}{\ln \widehat{z}} = \frac{w}{\ln(1+w)} = 1 + \frac{w}{2} + \mathcal{O}(w^2).$$

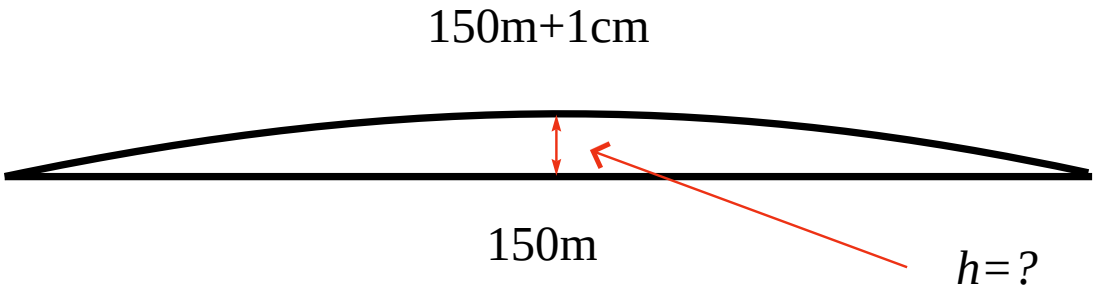
To pomeni, da za majhne x, ko je $z \approx 1$, velja

$$g(\widehat{z}) - g(z) \approx \frac{\widehat{z}-z}{2} \approx \frac{e^x\delta}{2} \approx \frac{\delta}{2}.$$

Ker je $g(z) \approx 1$, to pomeni, da je $\widehat{y} = y(1+\alpha)$, kjer je $\alpha \leq 3.5u$.

Zgled za nelinearno enačbo

Imamo $150m$ dolgo tračnico, ki je na obeh koncih trdno vpeta. Zaradi velike vročine se tračnica raztegne za $1cm$ in se dvigne v obliki krožnega loka. Na 8 decimalk točno izračunaj, kolikšna je maksimalna oddaljenost od tal.



Točen rezultat je $75.00074999cm$.