UPPSALA
UNIVERSITET

U.U.D.M. Project Report 2014:40

Miniräknarmania

eller hur att få ut så mycket som möjligt från en enkel reklamräknare

Christer Blomqvist

Examensarbete i matematik, 15 hp
Handledare och examinator: Vera Koponen
Maj 2014

Department of Mathematics
Uppsala University

.

Sammanfattning

I detta arbete undersöks möjliga algoritmer för den naturliga logaritmen och motsvarande exponentialfunktion givet de begränsningar som en enkel åtta siffrors reklamräknare med fyra räknesätt och roten ur har. Ett antal någorlunda praktiska algoritmer undersöktes med avseende på noggrannhet och enkelhet. För logaritmer hittas en mycket enkel algoritm som ger tre á fyra korrekta siffror. Även mer komplexa algoritmer som ger runt sex korrekta siffror undersöks. För e^x är kanske de mest praktiska algoritmerna, algoritmer som baserar sig på serieutveckling, och som ger sex á sju korrekta siffror.

.

Innehåll

0 Introduktion

Här sitter författaren med en enkel gratisräknare och vill, naturligtvis, beräkna lite mer än $+$, $-$,$\times$, $\div$ och roten ur, vilket i stort är vad räknaren i fråga klarar av. Kanske lite logaritmer, eller varför inte e^x? Räknaren har ett minne och en åtta siffrors display elegant skyddad av ett sådant där hightecklock som öppnar sig så där soft som portarna på skurkarnas rymdskepp i filmen Moonraker. Förutom den begränsade funktionsrepertoaren lider räknaren av att ha rak prioritetsordning samt att svaren trunkeras till åtta siffror.

Syftet med detta arbete är dels att hitta enkla algoritmer för sådana räknare, och då algoritmer som allra helst inte kräver att man skriver ner några mellanresultat. Algoritmerna skall också vara enkla att komma ihåg.

Avsnitten börjar ofta med en beskrivning av algoritmerna i stort på det sätt författaren kom på dem, följt av mer formell analys. En hel del av undersökningarna sker genom praktiska försök med en programvara som simulerar den enkla räknarens funktion. I möjligaste mån försöks även det som då upptäcks analyseras med mer rigorösa metoder.

Syftet är också i mångt och mycket att undersöka roande matematik, men även att undersöka hur man kan analysera numeriska algoritmer under diverse begränsningar.

Syftet är inte att söka finna något "bästa möjliga" algoritm — även om försök görs att optimera de undersökta algoritmerna. Syftet är inte heller att skapa en praktisk samling algoritmer, eftersom ganska få människor torde ha ett överväldigande behov av att beräkna logaritmer på en enkel reklamräknare.

Detta är inte skrivet i klassisk form med lemman, teorem och så vidare, utan är skrivet i en mindre formell form och i ordning mer eller mindre så som idéerna dök upp. Detta för att arbetet i sig inte handlar om någon direkt avancerad ny matematik, utan istället för att det delvis handlar om just hur man kan använda olika verktyg, mestadels från analysen, för att lösa problem av praktisk (eller i detta fall snarare ganska opraktisk) art, och för att den klassiska formen ofta kan dölja hur tankegångarna har gått.

Platsen i texten där en algoritm introduceras indikeras med en not i marginalen (L1, L2 och så vidare). Hur man i praktiken skall trycka in algoritmerna på en enkel räknare finns beskrivet i appendix 2.

1 Logaritmer

1.1 Logaritmer via integraler

Denna metod att beräkna logaritmer bygger på att den primitiva funktionen till $1/x$ är $\ln(x) + C$. Närmre bestämt har vi:

$$\int_1^x \frac{\mathrm{d}x}{x} = [\ln x]_1^x = \ln x \tag{1.1}$$

Detta hjälper dock föga. Dock skulle vi kunna välja en annan exponent än -1. Om exponenten är nära -1 torde också integralen vara nära $\ln x$. För andra exponenter än -1 har vi

$$\int_1^x x^{-a}\,dx = \left[\frac{1}{1-a}x^{1-a}\right]_1^x = \frac{1}{1-a}(x^{1-a} - 1) \tag{1.2}$$

Talet $1 - 1/p$ närmar sig 1 om p går mot oändligheten. Vi kan alltså välja detta som vårt a. Detta skulle ge

$$\begin{aligned}\int_1^x x^{-a}\,dx = \int_1^x x^{1/p-1}\,dx &= \left[\frac{1}{(1/p-1)+1}x^{(1/p-1)+1}\right]_1^x \\ &= [px^{1/p}]_1^x = p(x^{1/p} - 1)\end{aligned} \tag{1.3}$$

Detta skulle förhoppningsvis, för stora p, ge

$$\ln x \approx p(x^{1/p} - 1) \tag{1.4}$$

En praktisk test med till exempel $x = 10$ och $p = 10000$, på något lite kraftfullare än vår gratisräknare, ger oss

$$32768(10^{1/32768} - 1) \approx 2,30285020824672$$

Om vi räknar ut $\ln 10$ direkt så får vi med samma antal siffror 2,30258509299405. Dessa tal skiljer sig från varandra med c:a 0,0003. Inte alltför illa, men inte heller något revolutionerande.

Det var i detta steg tanken slog författaren att kvadratroten ur x är lika med $x^{1/2}$ och att upprepade kvadratrötter därmed är lika med $x^{1/2^n}$, där n är antalet gånger man drar kvadratroten ur. Vi skulle alltså kunna välja $p = 2^n$ och få

$$\ln x \approx 2^n(x^{1/2^n} - 1) \tag{1.5}$$

I praktiken skulle vi alltså kunna först dra roten ur ett antal gånger och därefter subtrahera ett och sedan multiplicera med två lika många gånger

som vi drog roten ur, eller direkt med en lämplig potens av 2 om vi kan den utantill. Detta innebär att vi kan beräkna logaritmer med hjälp av vår mycket enkla miniräknare med $+$, $-$,$\times$, $\div$ och $\sqrt{}$.

Hur många gånger skall man då dra roten ur? Rimligen torde fler successiva kvadratrötter ur ge ett allt bättre resultat, om det inte vore för räknarens begränsade noggrannhet. Räknaren visar, och räknar med, åtta siffror, och dessutom trunkerar den resultatet av varje beräkning. Så å ena sidan vill vi välja ett så stort värde på p som möjlig, men å andra sidan tappar vi då signifikanta siffror eftersom roten ur något efter ett antal steg blir något i formen 1,0001234, där antalet nollor i mitten av talet ökar efterhand. En gissning är att resultatet blir som bäst när rotutdragningarna ger något i stil med just 1,0001234, d.v.s. med hälften av siffrorna kvar efter nollorna. Vid lite försök att beräkna ln(10) för olika antal roten ur fås de svar som redovisas i tabell 1.

n	p	Mellanresultat	Svar	Fel
8	256	1,0090349	2,3129344	-0,0103
9	512	1,0045072	2,3076864	-0,0051
10	1024	1,0022510	2,3050240	-0,0024
11	2048	1,0011248	2,3035904	-0,0010
12	4096	1,0005622	2,3027712	-0,0002
13	8192	1,0002810	2,3019520	0,0006
14	16384	1,0001404	2,3003136	0,0023

Tabell 1: Första algoritmen, $ln10$ för olika antal steg.

Bäst resultat blev det alltså då vi drog roten ur tolv gånger. Mellanresultatet var då 1,0005622, alltså det första tal då antalet siffror efter nollorna är lika med hälften av antalet siffror, eller med andra ord, när vi har tre nollor efter decimalkommat. Vi kan kalla denna första algoritm, L1. L1

Detta är dock inte det bästa vi kan göra. Vi valde att låta $a = 1 - 1/p$, där $p > 0$, men vi skulle lika bra kunna ha valt $a = 1 + 1/p$. Även detta går mot 1 då p går mot oändligheten. Gör vi samma analys av detta som ovan får vi att L2

$$\int_1^x x^{-1/p-1}\,dx = p(1 - \frac{1}{x^{1/p}}) \tag{1.6}$$

som ger

$$\ln\ x \approx 2^n(1 - 1/x^{1/2^n}) \tag{1.7}$$

Vi kan kalla en algoritm baserat på detta för algoritm L2.

För $x = 10$ och $n = 15$ så får vi (på en mer avancerad räknare) för den första metoden,

$$\ln 10 = 32768(10^{1/32768} - 1) \approx 2,3026659950592$$

med ett fel på ungefär 0,00008 och för den andra,

$$\ln 10 = 32768(1 - 1/10^{1/32768}) \approx 2,3025041938842$$

med ett fel på ungefär $-0,00008$. Snittet av dessa bägge tal måste alltså bli bättre än endera talet. Vi får nu 2,3025850944717, som har ett fel på c:a $1,5 \cdot 10^{-9}$. En klar förbättring.

Om de bägge resultaten är A respektive B så har vi det bättre resultatet,

$$\begin{aligned} C = \frac{A+B}{2} &= ((x^{1/2^n} - 1)2^n + (1 - 1/x^{1/2^n})2^n)/2 \\ &= (x^{1/2^n} - 1/x^{1/2^n})2^{n-1}. \end{aligned} \tag{1.8}$$

I tabell 2 redovisas resultatet av en algoritm baserad på ovanstående ekvation, för den enkla, ack så idoga räknaren. Det verkar som bästa resultatet fås vid åtta rotutdragningar, då vi har två nollor efter decimalkommat, och då antalet kvarvarande signifikanta siffror blir strax över hälften av siffrorna.

p	n	Mellanresultat	Svar	Fel
6	64	1,0366328	2,3030725	-0,00049
7	128	1,0181516	2,3026944	-0,00011
8	256	1,0090349	2,302592	-0,00001
9	512	1,0045072	2,3025152	0,00007

Tabell 2: Tredje algoritmen, $ln10$ för olika antal steg.

L3 f

Denna tredje algoritm, medelvärdesalgoritmen, kan alltså sammanfattas som:

Algoritm, L3 f, första versionen: Beräkning av logaritmen av x, då x>1:
Dra roten ur talet tills reultatet <1,01
det vill säga att och antalet nollor efter decimalkommat är två.
Subtrahera de multiplikativa inversen av resultatet från resultatet.
Multiplicera detta med 2 upphöjt till ett mindre
än antalet rotutdragningar.

Hur gör man nu detta i praktiken? För vår räknare blir nog den enklaste algoritmen:

Skriv in talet som skall tas logaritmen ur.
Tag $\sqrt{}\ \sqrt{}\ \cdots \sqrt{}$,
så många gånger som behövs tills man får
ett resultat <1 och två nollor efter första 1:an.
M+
$1\ \div\ $ MR
$-\ $ MR
$\times$ 128, eller motsvarande 2 upphöjt till ett
mindre än antalet $\sqrt{}$, följt av $=$.

(Här antogs minnet vara nollställt från början.) Svaret blir negativt, men denna räknare har en tangent för teckenbyte, så det problemet är enkelt löst. Om man inte har en sådan tangent får man kanske vackert klara av att se det positiva i det. Om man envisas med att ha resultatet positivt kan man ju alltid lösa detta med sekvensen:

M+ + MR = M− MR.

Om vi struntar i denna eleganta men kanske ack så onödiga utvidgning så behöver vi in alles runt 20 tangenttryckningar för att få fram att $\ln 10 \approx 2.3026$.

Som enkel metod att finna närmevärden för logaritmer kan algoritmen duga redan i första steget, för $\sqrt{x} - 1/\sqrt{x}$ ger ett fel på c:a 1% eller mindre då $0,6 < x < 1,7$.

Nämnas kan också att på räknaren ifråga är MR en knapp märkt M$\frac{R}{C}$. Om man trycker en gång på knappen återkallas minnet, och trycker man två gånger så återkallas minnet, och samtidigt så raderas det.

1.1.1 Lite mer analys av algoritmerna

1.1.1.1 Några satser utifrån föregående resonemang. Vi kan alltså anta att

$$\ln x = \lim_{p\to\infty} p(x^{1/p} - 1) \tag{1.9}$$

respektive

$$\ln x = \lim_{p\to\infty} p(1 - \frac{1}{x^{1/p}}) \tag{1.10}$$

I efterhand inser författaren att dessa kan härledas direkt ur

$$e^x = \lim_{n\to\infty} \left(1 + \frac{x}{n}\right)^n \tag{1.11}$$

Om vi sätter $y = \left(1 + \frac{x}{n}\right)^n$, bryter ut x och sedan återinför gränsvärdet så fås något som (1.9). Lite mer manipulerande ger (1.10). Dessa härledningar är dock inga bevis utan bara formella förfaranden som – så ofta – kan visa

oss en väg till mer stringenta bevis. Vi återkommer till detta när vi skall titta på exponentialfunktionen. Vi skall dock strax titta på ett annat bevis.

Tar vi medelvärdet av (1.9) och (1.10), som vi tidigare gjorde, så får vi

$$\ln x = \lim_{p\to\infty} \frac{p}{2}(x^{1/p} - x^{-1/p}) \tag{1.12}$$

eller, om vi skriver som algoritmen är formulerad

$$\ln x = \lim_{n\to\infty} 2^{n-1}(x^{1/2^n} - x^{-1/2^n}) \tag{1.13}$$

Ett kanske inte alltför självklart resultat.

1.1.1.2 Bevis av (1.10)

Bevis. I detta bevis skriver vi om (1.10) till

$$\ln x = \lim_{q\to 0} \frac{1}{q}(1 - \frac{1}{x^q}) + \varepsilon \tag{1.14}$$

där $q = 1/p$ och $\varepsilon = 0$ om (1.10) gäller. Vi skriver om detta till

$$\varepsilon = \ln x - \lim_{q\to 0} \frac{1}{q}(1 - \frac{1}{x^q}) = \lim_{q\to 0}(\ln x - \frac{1}{q}(1 - \frac{1}{x^q})) \tag{1.15}$$

Utifrån hur vi fick fram (1.6) får vi[1]

$$\ln x - \frac{1}{q}(1 - \frac{1}{x^q}) = \int_1^x (\frac{1}{w} - \frac{1}{w^{1+q}})\, dw = \int_1^x \frac{1}{w}(1 - \frac{1}{w^q})\, dw \tag{1.16}$$

Om vi nu tittar på derivatan av $1 - 1/w^q$ så får vi $qw^{(-q-1)} = q/w^{q+1}$, men eftersom $w^{q+1} > w^1 > 1$, då $w > 1$ och $q > 0$, så gäller att $q/w^{q+1} < q$. Det innebär att

$$1 - \frac{1}{w^q} < qw \tag{1.17}$$

Det i sin tur innebär att

$$\int_1^x \frac{1}{w}(1 - \frac{1}{w^q})\, dw \leq \int_1^x \frac{1}{w} qw\, dw = \int_1^x q\, dw = q(x-1) \tag{1.18}$$

och att

$$\varepsilon = \lim_{q\to 0}(\ln x - \frac{1}{q}(1 - \frac{1}{x^q})) = \lim_{q\to 0}(q(x-1)) = 0 \tag{1.19}$$

vilket skulle bevisas. □

[1] Vi byter här variabelnamn på variabeln i integranden från x till w, för att lättare se bevisets struktur.

1.1.1.3 Bevis av (1.9)

Bevis. Givet föregående bevis har vi att

$$\ln x = \lim_{p\to\infty} p(1 - \frac{1}{x^{1/p}}) = \lim_{p\to\infty} p(\frac{x^{1/p} - 1}{x^{1/p}}) \tag{1.20}$$

Men eftersom $\lim_{p\to\infty} x^{1/p} = 1$ så har vi

$$\lim_{p\to\infty} p(\frac{x^{1/p} - 1}{x^{1/p}}) = \frac{\lim_{p\to\infty} p(x^{1/p} - 1)}{\lim_{p\to\infty} x^{1/p}} = \lim_{p\to\infty} p(x^{1/p} - 1) = \ln x \tag{1.21}$$

□

1.1.1.4 Bevis av (1.12)

Bevis. Eftersom (1.12), för ett givet värde på p, är ett medelvärde av uttrycken innanför limesoperatorn på (1.9) och (1.10) så blir gränsvärdet det samma som dessa har i och med att vi har en inneslutning. □

1.1.1.5 Ett andra bevis av (1.9)

Bevis. Om vi låter $q = 1/p$ och $x = 1 + z$ kan vi skriva om (1.9) till[2]

$$y = \lim_{p\to\infty} p(x^{1/p} - 1) = \lim_{q\to 0} \frac{1}{q}((1+z)^q - 1) \tag{1.22}$$

Binomialserien[3] [5, s. 231] ger oss

$$\begin{aligned} y &= \lim_{q\to 0} \frac{1}{q}((1+z)^q - 1) \\ &= \lim_{q\to 0} \frac{1}{q}(\sum_{n=0}^{\infty} \binom{q}{n} z^n - 1) \\ &= \lim_{q\to 0} \sum_{n=1}^{\infty} \frac{1}{q}\binom{q}{n} z^n \end{aligned} \tag{1.23}$$

[2]I detta arbete används variabelnamnet z i allmänhet för $1 - x$ eller $1 + z$, inte för en komplexvärd variabel.

[3]Biominalserien är en utvidgning av biominalteoremet till den komplexa domänen. I detta fall är $0 < q < 1$

För koefficienten för varje term gäller

$$\begin{aligned}\lim_{q\to 0}\frac{1}{q}\binom{q}{n} &= \lim_{q\to 0}\frac{(q-0)(q-1)...(q-(n+1))}{q\cdot 1\cdot 2\cdot ...\cdot n}\\ &= \lim_{q\to 0}\frac{q}{q}\cdot\lim_{q\to 0}\frac{q-1}{1}\cdot ...\cdot\lim_{q\to 0}\frac{q-(n+1)}{n+1}\cdot\frac{1}{n}\\ &= \frac{(-1)^{n-1}}{n}\end{aligned} \tag{1.24}$$

Här utnyttjar vi att $\lim_{q\to 0} f(p)g(p) = \lim_{q\to 0} f(p)\cdot\lim_{q\to 0} g(p)$ [2, s. 136] och att vi har $n-1$ negativa faktorer. Det ger oss

$$y = \lim_{q\to 0}\sum_{n=1}^{\infty}\frac{1}{q}\binom{q}{n}z^n = \sum_{n=1}^{\infty}\frac{(-1)^{n-1}}{n}z^n = z - \frac{z^2}{2} + \frac{z^3}{3} - \dots \tag{1.25}$$

Alltså samma maclaurinutveckling som för $\ln(1+z) = \ln x$, vilket innebär att $y = \ln x$ — åtminstone då $0 < x < 2$, eftersom detta är inom konvergensradien för ovanstående serie. □

1.1.2 Noggrannhet

1.1.2.1 Noggrannhet utan trunkeringsfel. Hur bra är då ovanstående algoritmer? Vi tar först inte trunkeringsfel[4] i beaktande.

För att lösa detta kan vi jämföra serieutvecklingarna av de olika uttrycken. Vi har ingen maclaurinutveckling av $\ln x$ eftersom denna funktion är divergent då x går mot 0. Men som nyss nämndes har vi däremot att

$$\ln(1+z) = z - \frac{z^2}{2} + \frac{z^3}{3} + ... \tag{1.26}$$

Om vi åter igen låter $x = 1+z$ och $q = 1/p$ så har vi av binominalteoremet att

$$\begin{aligned}x^{1/p} &= (1+z)^q\\ &= 1 + qz + \frac{q(q-1)}{2!}z^2 + \frac{q(q-1)(q-2)}{3!}z^3 + ...\end{aligned} \tag{1.27}$$

[4] I denna text menas med trunkeringsfel, de fel som uppkommer i och med de trunkeringar som sker eftersom räknaren bara kan hantera maximalt åtta signifikanta siffror, inte trunkering av en serieutveckling.

Detta ger oss

$$\begin{aligned} p(x^{1/p}-1) &= \frac{1}{q}((1+z)^q-1) \\ &= z + \frac{q-1}{2!}z^2 + \frac{(q-1)(q-2)}{3!} + O(z^4) \\ &= z + \left(-\frac{1}{2}+\frac{q}{2}\right)z^2 + \left(\frac{1}{3}-\frac{q}{2}+\frac{q^2}{6}\right)z^3 + O(z^4) \end{aligned} \tag{1.28}$$

Vi kan nu subtrahera med serien för $\ln(1+z)$ för att få

$$p(x^{1/p}-1) - \ln x = \frac{q}{2}z^2 + \left(-\frac{q}{2}+\frac{q^2}{6}\right)z^3 + O(z^4) \tag{1.29}$$

På samma sätt fås

$$p(1-\frac{1}{x^{1/p}}) - \ln x = -\frac{q}{2}z^2 + \left(\frac{q}{2}+\frac{q^2}{6}\right)z^3 + O(z^4) \tag{1.30}$$

Detta innebär att både algoritm ett och algoritm två har ett fel i storleksordningen

$$\varepsilon \approx \frac{q}{2}z^2 = \frac{1}{2}2^{-n}z^2 \tag{1.31}$$

— oaktat trunkeringsfel. Det vill säga att felet torde vara ungefär omvänt proportionerligt mot p, och därmed omvänt proportionerligt mot 2^n.

Felet i beräkningen enligt (1.12), alltså enligt tredje algoritmen, torde nu bli hälften av summan av ovanstående felen, eftersom vi tar medelvärdet av värdena. Termerna för z^2 tar ut varandra. Kvar har z^3-termerna i serieutvecklingarna. I dessa tar $-q/2$ och $q/2$ ut varandra. Kvar blir alltså i bägge serierna $q^2z^3/6$. Medelvärdet av dessa två blir då just

$$\varepsilon \approx \frac{q^2}{6}z^3 = \frac{1}{6}2^{-2n}z^3 \tag{1.32}$$

Vi skall strax underöka hur pass bra dessa ekvationer stämmer.

1.1.2.2 Trunkeringsfel. Eftersom beräkningarna här sker på den enkla räknaren kommer vi att få trunkeringsfel förutom de fel som beror på algoritmen i sig. Låt oss börja med trunkeringsfelet på svaret, givet att beräkningarna fram till sista roten ur vore exakta, och att vi därefter får en trunkering av mellanresultatet. Med andra ord att vi har en trunkering då svaret är i formen 1,00...*någonting*. Vi skall till att börja med titta på $p(x^{1/p}-1)$, där $p = 2^n$. Låt oss till att börja med anta att x är ganska nära 1 och vi därför har $x = 1+z$, där z är ett litet tal.

Binominalserien ger oss att

$$x^{1/p} - 1 = (1+z)^{1/p} - 1 = 1 + \frac{z}{p} + O(z^2) - 1 = \frac{z}{p} + O(z^2) \tag{1.33}$$

Detta är då ett tal nära noll. För att hitta första signifikanta siffran kan vi ta tiologaritmen (lg) av ovanstående.[5] Det betyder att den första signifikanta siffran är vid decimalen med exponenten närmast

$$-\lg(|z|/p) = \lg p - \lg|z| = \lg 2^n - \lg r = n\lg 2 - \lg|z| \tag{1.34}$$

Vi har till exempel då $z = 0.1$ och $p = 2^8$ att $x^{1/p} - 1 = (1+z)^{1/p} - 1 \approx 0.0008 \approx 10^{-3}$, och $n\lg 2 - \lg|z| = 8\lg 2 - lg0.2 \approx 3$, och trunkeringsfelet är alltså i storleksordningen 10^{-3}. Om talet är i formen $1 + z$, och vi har tillgång till d decimaler, och om vi subtraherar ett så har vi alltså kvar

$$d - (n\lg 2 - \lg|z|) \tag{1.35}$$

decimaler. Om svaret är i storleksordningen 1 blir alltså felet i storleksordningen 10 upphöjt till minus detta. Om x är signifikant skiljt från 1 så behövs först ett antal rotutdragningar för att ovanstående resonemang skall gälla. Eftersom dessa rotutdragningar inte förändrar antalet signifikanta siffror, såsom subtraktionen med 1 gör, så torde det absoluta felet förre multiplikationen med p vara ungefär lika vid detta steg. Multiplikationen ökar sedan felet i proportion till p, så det relativa felet torde vara relativt oberoende av x, och i storleksordning lika stort som de vi finner här eftersom $x \approx 1$.

Eftersom alla tre algoritmerna väsentligen har sin första signifikanta siffra vid samma decimal (annars så skulle inte en multiplikation med 2^n respektive 2^{n-1} ge svaret), så får trunkeringsfelet i stort samma effekt för de tre algoritmerna.

1.1.2.3 Praktisk undersökning av felet. För att undersöka hur felet i praktiken beter sig så skrevs lite program[6] där Texas Instruments Nspire® användes som verktyg. Eftersom den hanterar fler siffror än åtta så skrevs ett enkelt program som trunkerade svaret till åtta siffror efter varje beräkning. Detta eftersom den enkla räknaren visade sig trunkera och inte avrunda sina resultat. Resultaten enligt första tabellen och resultaten enligt programmen jämfördes, och de gav samma resultat.

[5] Vi kommer framleds, i diagram, att visa felet som lg av absolutvärdet av felet. Dels för att det är lättare att se felet om det kan variera över flera magnituder, dels för att det kommer att visa sig att logaritmen av absolutvärdet av felet kommer att vara i stort linjärt.

[6] Se appendix 1.

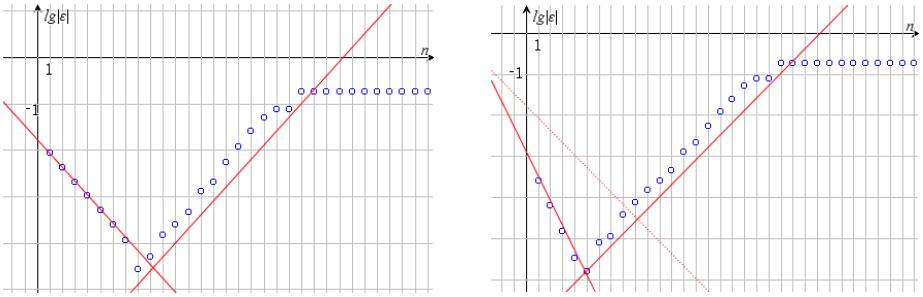
Enligt (1.31) så har vi för $p(x^{1/p} - 1)$,

$$|\varepsilon| \approx \frac{1}{2} z^2 p^{-1} = \frac{1}{2} z^2 2^{-n} \tag{1.36}$$

Tiologaritmen av bägge sidor ger oss

$$\lg |\varepsilon| \approx 2 \lg z - n \lg 2 - \lg 2 \tag{1.37}$$

I en lin/lg graf över felet, skulle alltså en linje anpassat till felen ha en lutning på $-\lg(2)$, eller ungefär $-0,3$.



(a) Gällande L1. **(b)** Gällande L3.

Figur 1: Logaritmen av felet, $\lg |\varepsilon|$, kontra n för den den första, L1, och den tredje algoritmen, L3. Även linjer enligt teori är inlagda.Den streckade linjen i delfigur b motsvarar den heldragna nedåtriktade linjen i delfigur a.

Detta var också vad man i praktiken finner. I figur 1a motsvarar cirklarna logaritmen (bas 10) av felen vid beräkning av ln(1,2) enligt algoritm ett då man låter n anta ett antal olika värden. Talet 1,2 valdes som ett någorlunda godtyckligt tal nära 1.

Den nedåtgående linjen i figur 1a är enligt ekvation (1.37). Som man kan se är passningen i stort perfekt. Den uppåtgående linjen är enligt ekvation (1.35). Här passar kurvan inte lika bra. Man kan anta att detta beror på att vi i ekvation (1.35) antog att den enda trunkeringen var i mellanresultatet och inte vid varje steg. Vid försök där trunkeringen just skedde endast vid mellanresultatet så blev passningen åter igen nästan perfekt.

I L2, motsvarande $p(1 - x^{-1/p})$, blir resultatet, som väntat, ungefär som för L1. I den tredje algoritmen, då medelvärdet beräknas får vi från (1.32) att

$$\lg |\varepsilon| \approx 3 \lg z - 2n \lg 2 - \lg 6 \tag{1.38}$$

I figur 1b har vi det teoretiska felet enligt algoritm 1 inlagt som en streckad linje. Det teoretiska felet enligt ekvation (1.38) är den nedåtgående linjen, och den uppåtgående motsvar trunkeringsfelet enligt (1.35), eftersom, enligt

tidigare resonemang, detta fel i stort torde vara samma för de tre algoritmerna. Som man kan se passar linjerna åter igen väldigt väl.

Man kan också se att den första algoritmen gav bäst svar då $n = 8$ och den tredje då $n = 5$, samt att den senare gav ett noggrannare resultat.

1.1.3 Bästa möjliga svar

1.1.3.1 Pratisk undersökning över ett större intervall . Bästa möjliga svar blir alltså där trunkeringsfelet ännu inte växt sig större än noggrannheten man får genom att använda större värde på n, alltså där de bägge linjerna enligt respektive ekvation för felet och för trunkeringsfelet korsar varandra. För första algoritmen ger detta

$$\lg|\varepsilon| \approx \frac{\lg z - \lg 2 - 8}{2} \tag{1.39}$$

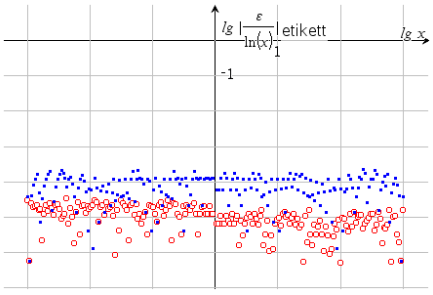
Om $1 < x < 3$ och alltså $0 < z < 2$ så blir $\lg|\varepsilon| < -4$.

Motsvarande för tredje algoritmen ger oss

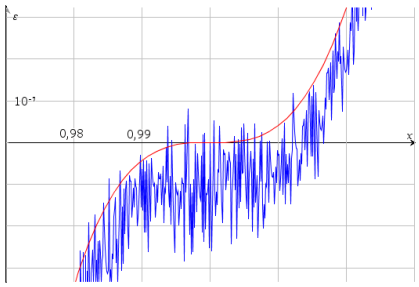
$$\lg|\varepsilon| \approx \frac{\lg z - \lg 6 - 16}{3} \tag{1.40}$$

Om $0 < z < 6$ så blir $\lg|\varepsilon| < -5$. Vi får alltså runt en signifikant siffra mer.

I figur 2a visas det faktiska resultatet av ett antal tal i området 10^{-3} till 10^3. Ett litet program fick testa att beräkna logaritmen med hjälp av



(a) Ett log/log diagram av relativa fel för första algoritmen (punkter) och tredje algoritmen (cirklar) i området området 10^{-3} till 10^3.



(b) Felet i tredje algoritmen runt $x = 1$ då $n = 1$ med trunkering (ojämn linje) och utan trunkering (jämn linje).

Figur 2: Två figurer gällande bästa svar.

första respektive tredje algoritmen för olika värden på n. Det som redovisas är logaritmen av det minsta möjliga relativa felet.

Som synes är felet för den första algoritmen (punkter) i stort en tiopotens större än felet för den tredje algoritmen (cirklar) då $x > 1$. Då $x < 1$ har den tredje algoritmen en aning större fel än då $x > 1$.

Varför nu detta resultat? Vid inspektion visar sig $p(x^{1/p} - 1) \leq \ln x$ för alla värden på x då $p \geq 1$. Om vi stället undersöker $\frac{p}{2}(x^{1/p} - x^{-1/p})$ så är värdet större än $\ln x$ då $x > 1$ och mindre än $\ln x$ då $x < 1$.

Eftersom en trunkering av positiva tal är en avrundning nedåt, så kommer felet orsakat av trunkering gå åt samma håll för algoritm ett, som bygger på $p(x^{1/p} - 1)$, oberoende av värdet på x, och därmed är de relativa felen i stort lika stora för värden på x oberoende av om de är större eller mindre än 1.

För den tredje algoritmen, som bygger på $\frac{p}{2}(x^{1/p} - x^{-1/p})$, så är situationen annorlunda. Både minuenden och subtrahenden avrundas nedåt, vilket i sig torde ta ut varandra. Men ytterliga trunkeringar, och alltså avrundningar nedåt, sker vid subtraktionen och vid multiplikationen med $p/2$. Detta innebär att man, då $x > 1$, avrundar mot ett bättre värde, medan man då $x < 1$, avrundar mot ett sämre värde.

I figur 2b kan man se detta. I figuren motsvarar den heldragna linjen resultatet av $\frac{p}{2}(x^{1/p} - x^{-1/p}) - \ln x$, utan trunkering efter åtta siffror. Den kraftigt hackiga linjen motsvarar differensen mellan reultatet enligt algoritm 3, inklusive trunkering, och $\ln x$. I bägge fallen är $p = 2$ och alltså $n = 1$.

Man kan se att kurvan med felet inklusive trunkering oftast ligger under den exklusive trunkering. Att så inte alltid är fallet beror på att trunkeringarna man har före subtraktionen kan slumpartat ta ut eller förstärka varandra.

1.1.3.2 Bevis av några olikheter. Återstår att bevisa de bägge påstådda olikheterna. Vi börjar med påståendet att $p(x^{1/p} - 1) \geq \ln x$.

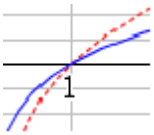
Bevis. Vi kan koppla till vår ursprungliga definition enligt ekvation (1.3) Integralen sker där över $x^{-1+1/p}$, där $p \geq 1$, medan integranden i ekvation (1.1), som ger $\ln x$, är $1/x$. Eftersom $x^{-1+1/p} \geq 1/x$, och eftersom integrationsgränserna är lika så blir integralen över $x^{-1+1/p} \geq \ln x$. □

Man kan även se detta som ett resonemang om derivatan av $p(x^{1/p} - 1)$, respektive $\ln x$, och det är den metod vi skall använda för det andra beviset.

Vi har alltså påståendet att för $\frac{p}{2}(x^{1/p} - x^{-1/p})$ så är värdet större än $\ln x$ då $x > 1$ och mindre än $\ln x$ då $x < 1$.

Bevis. Vi har att

$$\begin{aligned} \frac{d}{dx}\frac{p}{2}(x^{1/p} - x^{-1/p}) &= \frac{p}{2}(\frac{1}{p}x^{1/p-1} + \frac{1}{p}x^{-1/p-1}) \\ &= \frac{1}{2x}(x^{1/p} + x^{-1/p}) \end{aligned} \tag{1.41}$$

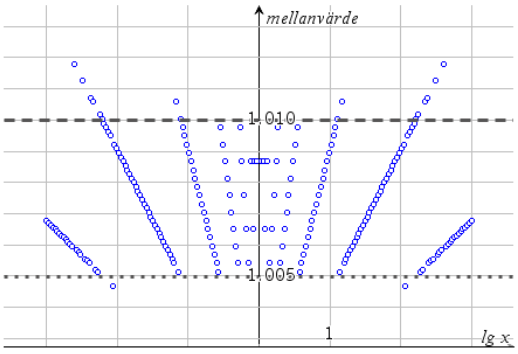


Men ett uttryck av formen $u + 1/u$ har, då $u > 0$, ett minimum vid $u = 2$ och alltså är värdet av ekvation (1.41) mindre än $1/x$ förutom då $x = 1$ då de är lika, och eftersom värdet av ekvation (1.41) $= 1/x = 1$ då $x = 1$ så är $\frac{p}{2}(x^{1/p} - x^{-1/p}) < \ln x$ då $x < 1$ och $> \ln x$ då $x < 1$

□

1.1.3.3 En liten förbättring. Vi kan utnyttja skillnaden i noggrannhet då x är mindre respektive större än 1 genom att beräkna $-\ln(1/x)$ då $0 < x < 1$. Visserligen får vi en extra trunkering, men detta fel drunknar snabbt i det fel som blir i och med varje kvadratrot som beräknas då. Vid praktiskt försök visar sig felet för denna modifierade algoritm vara i stort en spegling av felen då $x > 1$.

1.1.3.4 Bästa värde på n. I ovanstående resonemang söktes bästa möjliga svar för algoritm ett och algoritm tre, men i praktiken vill man inte behöva undersöka för vilket antal kvadratrötter som svaret blir bäst. I första varianten av algoritm tre antogs att man skulle dra rötter ur tills vi har två nollor efter decimalkommat. Är då detta en rimlig strategi? Ett praktiskt försök visar att värdet efter rotutdragningarna vid bästa möjliga svar i stort ligger mellan 1,01 och 1,005 som är väldigt nära $\sqrt{1,01}$



Figur 3: Mellanvärde kontra $\lg x$ för den tredje algoritmen vid bästa möjliga svar.I figuren är en horisontell streckad linje inlagd vid 1,010 och punktlinje vid 1,005.

Vi kan alltså se att vi i stort får bästa möjliga svar om man inverterar värden mindre än 1 och om man drar roten ur tills svaret blir mindre än 1,01.

Men om talet ligger mellan 1 och 1,01, skall man då dra roten ur en gång eller inte? Utifrån serieutvecklingen (1.26) har vi ett fel på i storleksordningen $z^2/2$, där $x = 1 + z$, och enligt (1.32) är felet i tredje algoritmen i storleksordningen $z^3/24$ då $n = 1$. Den tredje algoritmen kommer, trunkeringsfel obeaktat, alltid vara bättre än den linjära aproximationen $x - 1$. Vid

ett praktiskt försök visar sig detta, i stort, alltid vara sant, förutom för några få slumpartade värden extremt nära 1 där trunkeringsfelen råkar avrunda åt rätt håll.

1.1.3.5 Tredje algoritmen — sista versionen. Vi kan nu sammanfatta den förbättrade tredje algoritmen som: L3

Algoritm , L3 : Beräkning av logaritmen av x, då $x > 0$:
Om talet <1 beräkna 1/talet.
Dra roten ur talet minst en gång tills svaret blir <1.01
Subtrahera inversen av resultatet från resultatet.
Multiplicera detta med 2 upphöjt till ett mindre
än antalet rotutdragningar.
Om talet till att börja med var <1 så skall svaret göras negativt.

Denna algoritm ger oss i stort fem korrekta siffror över hela dess definitionsmängd, $0,0000001 \leq x \leq 99999999$.

1.1.4 Liknande algoritmer

Vis sökning på Internet hittades en algoritm [6] som omskrivet till en uttryck blir motsvarande

$$\frac{x^{1/p} - 1}{e^{1/p} - 1} \tag{1.42}$$

Här motsvarar, som tidigare, $x^{1/p}$ och $e^{1/p}$, via $1/p = 1/2^n$, beräkningen ett antal kvadraträtter ur. (I [6] angavs n till ett fixt tal, 12.) Om man låter $p \to \infty$ så kan vi utifrån maclaurinutvecklingen av e^x att $e^{1/p} \to 1 + 1/p$ och därmed att $e^{1/p} - 1 \to p$. Detta innebär att (1.42) går mot samma resultat som första algoritmen för stora p. Vid ett praktiskt försök visade sig algoritmen i stort alltid ge ett sämre resultat än algoritm ett, även då man drog roten ur ett optimalt antal gånger. Undantaget ett mindre område kring $x = 3$ där algoritmen gav nära fem signifikanta siffror.

1.2 Logaritmer via serieutveckling

Vi kan även beräkna logaritmer via serieutveckling av $\ln x = \ln(1 + z)$ enligt ekvation (1.26). I denna serieutveckling är konvergensen mycket långsam, men för värden på x nära ett kan det dock vara ett alternativ. För exempelvis $x = 1,1$ når man en bättre noggrannhet än algoritm tre efter endast tre termer, och efter sex termer är alla utom den sista siffran korrekt.

För exempelvis tre termer har vi

$$\begin{aligned}\ln(1-z) &= z - \frac{z^2}{2} + \frac{z^3}{3} - ... \\ &\approx \left(\left(\frac{z}{3} - \frac{1}{2}\right)z + 1\right)z\end{aligned} \tag{1.43}$$

Ovanstående visar också på hur man i praktiken kan räkna ut värdet på ett förhållandevis enkelt sätt.

Eftersom konvergensradien för serien är 1, och eftersom antalet nödvändiga termer växer mycket snabbt med avståndet från $x = 1$ så måste vi först reducera invärdet, och vad vore inte lämpligare än att utnyttja kvadratroten ur. Vi har då att

$$\ln x = 2^n \ln x^{1/2^n} \tag{1.44}$$

Vi kan alltså skala invärdet enligt ovanstående och sedan beräkna logaritmen enligt någon väl vald metod. I princip kan vi se de tidigare algoritmerna som specialfall av detta.

För att inte tappa signifikanta siffror vill vi ha en algoritm som fungerar inom ett intervall där vi ännu inte tappat signifikanta siffror på grund av rotutdragningarna—det vill säga, vi vill helst inte ha ett tal i formen $1,0...$ eller $0,9....$

För ekvation (1.43) kan man se att restermen i stort är lika med den första överhoppade termen, åtminstone då $z \approx 0$. För att få ett fel på mindre än 10^{-7} får vi alltså att $z^4/4 = 10^{-7}$ vilket ger att $z = 1 \pm 0,025$. I praktiken är felet ungefär lika med 10^{-7} inom intervallet $1 \pm 0,035$. Denna algoritm skulle alltså inte duga eftersom intervallet den täcker är för smalt.

1.2.1 En "förbättrad" algoritm

En idé vore att undersöka huruvida man kan finna ett värde eller uttryck i stället för koefficienten 1/3 som ger ett bättre svar. Vi skall alltså söka finna ett uttryck a sådant att

$$\ln(1+z) \approx z - \frac{z^2}{2} + \frac{z^3}{a} \tag{1.45}$$

med ett så litet fel som möjligt. Bryter vi ut a så får vi

$$a = \frac{z^3}{\ln(1+z) - z + \frac{z^2}{2}} \tag{1.46}$$

som, om vi ersätter logaritmen med (1.43), ger oss at a skall ha värdet 3. Om vi går ett steg till och ersätter logaritmen med ett fjärdegradspolynom så får vi ett utryck för $a = 12/(4 - 3z)$ som inte ger särskillt bra resultat.

Om vi roar oss med att rita en graf över a kontra z för (1.46) så får vi något som ser förhållandevis linjärt ut kring $z = 0$, så en ny ansats är att ersätta a med ett linjärt uttryck i z. Som vi tidigare konstaterade så blir uttrycket för $a = 3$, om man ersätter logaritmen med (1.43), och detta blir då gränsvärdet för a då $z \to 0$. Det är kanske lättast att se det genom att beräkna

$$\begin{aligned}\lim_{z\to 0}\frac{1}{a} &= \lim_{z\to 0}\frac{z - \frac{z^2}{2} + \frac{z^3}{3} + O(z^4) - z + \frac{z^2}{2}}{z^3} \\ &= \lim_{z\to 0}\frac{\frac{z^3}{3} + O(z^4)}{z^3} = \frac{1}{3}\end{aligned} \tag{1.47}$$

Vidare behöver vi lutningen. Lättast är kanske att utgå ifrån att

$$\ln(1+z) = z - \frac{z^2}{2} + \frac{z^3}{3} + \frac{z^4}{4} + O(z^5) \tag{1.48}$$

och föra in detta i 1.46, fast i formen $1/a$. Vi får då

$$\frac{1}{a} = \frac{\frac{z^3}{3} - \frac{z^4}{4} + O(z^5)}{z^3} = \frac{1}{3} - \frac{z}{4} + O(z^2) \tag{1.49}$$

och

$$\frac{d(a^{-1})}{dz} = -\frac{1}{4} + O(z) \tag{1.50}$$

och eftersom vi, via kedjeregeln, har

$$(f^{-1})' = -\frac{f'}{f^2} \tag{1.51}$$

så får vi

$$\left.\frac{da}{dz}\right|_{z=0} = \frac{9}{4} \tag{1.52}$$

Detta ger oss $a = 3 + \frac{9}{4}z = 3 + 2,25x$. Om vi nu använder detta i (1.45) så får vi något som fungerar bra mellan $0,9 < z < 1,07$. Genom prövning kan man dock finna att

$$a = 3 + 2.23x \tag{1.53}$$

ger ett fel nära 10^{-7} i intervallet $0,92 < z < 1,2$. Detta för att den senare approximationen ligger tillräckligt nära (1.46) över ett större intervall. Kombinerar vi detta med att invertera om $x < 1$ och att då $x > 1,2$ dra kvadratroten ur tills man får ett tal mellan 1 och 1,2 så får man en algoritm, L4, som ge sex á sju signifikanta siffror. L4

Problemet med denna algoritm är att vi, så vitt författaren kan se, måste komma ihåg ett mellanresultat, till exempel värdet på a. Dessutom så är algoritmen inte speciellt regelbunden, och därmed svår att komma ihåg.

1.2.2 Back to the roots

För att skapa en algoritm som går att använda utan att behöva lagra mellanresultat skall vi åter undersöka serieutvekling enligt (1.43). Åter igen kan vi dra roten ur tills resultatet är tillräckligt nära 1. Eftersom felet, då $x \approx 1$ och därmed $z \approx 0$, är nära den sist utlämnade termen så skall vi lösa

$$\frac{z^n}{n} = \varepsilon \tag{1.54}$$

Om vi vill att $x < 1,2$, $z < 0,2$ och $\varepsilon < 10^{-7}$ så behöver vi åtta termer.[7] I praktiken kan vi dock (som det kommer att visa sig) klara oss med sju termer.

Kan det vara så att vi kan få bra värden utanför konvergensområdet? Trunkerar vi serien vid någon term får vi ett polynom som alltid kommer att ge ändliga värden, och som nära $z = 1$ kommer att ha värden nära $\ln(1 + z)$.Det innebär att, som vi i praktiken ser, vi kan få någorlunda bra värden för logaritmen, även då $z > 1$, om vi väljer att trunkera serien vid någon lämplig term.

Eftersom räknaren har rak prioritetsordning så måste vi skriva om serieutvecklingen till en mer användbar form. Dessutom kan vi välja att beräkna $\ln(1 - z)$ för att slippa teckenväxlingar mellan termerna. För fyra termer skulle vi till exempel kunna använda

$$\begin{aligned} \ln(1-z) &\approx -\left(z + \frac{z^2}{2} + \frac{z^3}{3} + \frac{z^4}{4}\right) \\ &= -\left(\left(\left(\frac{3z}{4} + 1\right)\frac{2z}{3} + 1\right)\frac{z}{2} + 1\right)z \end{aligned} \tag{1.55}$$

Om vi provar detta för sju termer så är svaret runt 10^{-7} inom intervallet $0,84 < x < 1,19$, och för åtta termer inom intervallet $0,8 < x < 1,2$.

Om vi modifierar serieutvecklingen för sju termer så att dividerar z^7-termen med 6 om $x < 1$ och med 8 om $x > 1$ så kan vi utvidga intervallet till $0,8 < x < 1,23$

L5 Hurusom så kan vi använda endera serieutveckling i kombination med (1.44) för att få i stort samma noggrannhet som för algoritmen beskriven i föregående avsnitt. Omskrivet till en fungerande algoritm, L5, har vi för en serie med åtta termer:

[7] Vi kan exempelvis lösa detta genom prövning eller med hjälp av Lamberts W-funktion. Se appendix 3.

Skriv in talet som skall tas logaritmen ur.
Tag $\sqrt{}\ \sqrt{}\ \cdots \sqrt{}$,
så många gånger som behövs tills man får ett tal inom intervallet 0,8 till 1,2.
$-$ 1 $\pm$ M+
$\times$ MR $\div$ 8 $\times$ 7 $+$ 1
$\times$ MR $\div$ 7 $\times$ 6 $+$ 1
$\times$ MR $\div$ 6 $\times$ 5 $+$ 1
$\times$ MR $\div$ 5 $\times$ 4 $+$ 1
$\times$ MR $\div$ 4 $\times$ 3 $+$ 1
$\times$ MR $\div$ 3 $\times$ 2 $+$ 1
$\times$ MR $\div$ 2 $+$ 1
$\times$ MR $=$
$\times$ 128, eller motsvarande 2 upphöjt till antalet $\sqrt{}$.

1.3 Jämförelse av algoritmerna

Om vi skall beräkna $\ln 10 \approx 2,30258509299405$ så gav algoritm L1 svaret 2,3027712$\approx$ 2,303 efter knappsekvensen

$\sqrt{}\sqrt{}\sqrt{}\sqrt{}\sqrt{}\sqrt{}\sqrt{}\sqrt{}\sqrt{}\sqrt{}\sqrt{}\sqrt{}$
$-$ 1
$\times$ 4096 $=$

Alltså fyra signifikanta siffror efter 20 tryckningar — som var enligt ett väldigt enkelt mönster. L2 ger i stort samma noggrannhet.

Algoritm L3 ger svaret 2,302592$\approx$2,3026 efter knappsekvensen

$\sqrt{}\sqrt{}\sqrt{}\sqrt{}\sqrt{}\sqrt{}\sqrt{}\sqrt{}$
M+
1 $\div$ MR
$-$ MR
$\times$ 256 $=$ $\pm$.

Alltså fem signifikanta siffror efter 20 tryckningar — som var enligt ett ganska enkelt mönster. Algoritm L5, ger 2,3025824, vilket i stort ger sex signifikanta siffror. För att räkna ut detta behöver man dra kvadratroten ur fyra gånger, och därefter den sekvens som beskrivs i föregående sektion. Det ger inalles c:a 70 tryckningar, och dessutom enligt ett relative komplicerat schema. Om författarens egna intensiva men ack så korta minne får stå som modell, så är det nog så att man i praktiken med enkelhet kommer ihåg

algoritm L1, medan de andra algoritmerna kräver en hel del funderande innan man kan återkalla dem.

2 Exponentialfunktionen e^x

Hur att beräkna e^x? Här kan man direkt se två strategier. Antingen att försöka med

$$e^x = \lim_{n\to\infty} \left(1 + \frac{x}{n}\right)^n \tag{2.1}$$

eller med

$$e^x = 1 + \frac{x}{1} + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + ... \tag{2.2}$$

2.1 Algoritmer baserade på $(1 + x/n)^n$ och liknande

Låt oss börja med (2.1). För att få höga potenser av något kan vi på en sådan enkel miniräknare räkna $\times =$ om och om igen. Varje gång vi trycker $\times =$ så kvadrerar räknaren det som står i displayen. Vi kan alltså lätt räkna potenser i form av något (säg w) upphöjt till 2 upphöjt till något, alltså w^2, w^4, w^8, och så vidare. Vi skulle alltså kunna sätta $n = 2^p$ och sedan beräkna
E0 enligt följande algoritm, E0:

$x \;\div\; n$
$\times = \;\ldots\; \times =$, upprepat p gånger.

Detta är helt enkelt, som påpekades tidigare, algoritm L1, för logaritmer, baklänges. Fördelen med att hitta algoritm L1 utifrån integraler är dock att det då föll sig naturligt att hitta algoritm L2, och därmed algoritm L3, vilket kanske inte skulle vara lika självklart om vi direkt utgått ifrån (2.1).

Algoritm L2 utgår från att gränsvärdet av $p(x^{1/p} - 1)$, då $p \to \infty$. Om vi tar fram inversen av detta och återinför gränsvärdet skulle det eventuellt ge oss

$$e^x = \lim_{n\to\infty} \left(\frac{1}{1 - x/n}\right)^n \tag{2.3}$$

Att detta stämmer kan vi se genom att studera kvoten av $(1 + x/n)^n$ och $(1/(1 - x/n))^n$. Om denna går mot ett då $n \to \infty$ så går de mot samma

gränsvärde. Vi får

$$\begin{aligned}\frac{(1+x/n)^n}{(1/(1-x/n))^n} &= \left(\left(1+\frac{x}{n}\right)\left(1-\frac{x}{n}\right)\right)^n \\ &= \left(1-\frac{x^2}{n^2}\right)^n = \left(1-\frac{x^2}{n^2}\right)^{n^2/n} \\ &= \sqrt[n]{\left(1-\frac{x^2}{n^2}\right)^{n^2}}\end{aligned} \tag{2.4}$$

Inför vi gränsvärdesoperatorn så får vi

$$\lim_{n\to\infty} \sqrt[n]{\left(1-\frac{x^2}{n^2}\right)^{n^2}} = \lim_{n\to\infty} \sqrt[n]{e^{-x^2}} = 1 \tag{2.5}$$

Motsvarande omvändning av algoritm L3 skulle ge oss

$$e^x = \lim_{n\to\infty} \left(\frac{x+\sqrt{x^2+n^2}}{n}\right)^n \tag{2.6}$$

Att det sistnämnda stämmer är lätt att se, eftersom utrycket under kvadratroten asymptotiskt går mot n då $n \to \infty$. Uttrycket kan sedan förenklas till (2.1). Om man jämför (2.1) med (2.6) så kan man se att eftersom bägge uttrycken innanför limesoperatorn är monotont växande då $x > 0$, och eftersom $\sqrt{x^2+n^2} > x$ så kommer (2.6) att konvergera fortare än (2.1) då $x > 0$. Detta kan eventuellt utnyttjas i en algoritm för e^x.

2.1.1 Analys av algoritm baserat på $(1+x/n)^n$

Vilken potens skall man välja? Som för logaritmalgoritmerna hamnar man i en situation där högre värden på n teoretisk ger bättre noggrannhet, men där man, då man dividerar med stora n, förlorar signifikant siffror på grund av räknarens begränsade precision.

För att utröna detta börjar vi med en enkel analys trunkeringen oaktat. Hur stort blir då felet för ett givet n då $x \approx 0$? Analysen förenklas om vi undersöker logaritmen av $(1+x/n)^n$. Vi har

$$\begin{aligned}\ln\left(1+\frac{x}{n}\right)^n &= n\ln\left(1+\frac{x}{n}\right) \\ &= n\left(\frac{x}{n}+\frac{x^2}{2n^2}+O\left(x^3/n^3\right)\right) \\ &= x+\frac{x^2}{2n}+O\left(x^3/n^2\right)\end{aligned} \tag{2.7}$$

Här utnyttjade vi taylorutvecklingen av $\ln(1+z)$. Vi får sedan

$$\begin{aligned}\left(1+\frac{x}{n}\right)^n &= e^{x+\frac{x^2}{2n}+O\left(x^3/n^2\right)} = e^x e^{\frac{x^2}{2n}} e^{O\left(x^3/n^2\right)} \\ &= e^n\left(1+\frac{x^2}{2n}+O\left(x^3/n^2\right)\right)\left(1+O\left(x^3/n^2\right)\right) \\ &= e^n\left(1+\frac{x^2}{2n}+O\left(x^3/n^2\right)\right)\end{aligned} \tag{2.8}$$

Här utnyttjas att $e^x = 1 + x + O(x^2)$ ett par gånger.

Det relativa felet torde alltså vara i storleksordningen $x^2/2n$. Om $x = 1$ så behöver vi, för att få ett fel på 10^{-7}, eller mindre, att $n > 1/(2 \cdot 10^{-7}) = 5000000$. Provar vi med ett lite kraftfullare verktyg så finner vi att för $n = 5000000$ så ger $(1 + 1/n)^n$ ett fel som är extremt nära 10^{-7}. Detta gör i praktiken denna algoritm tämligen värdelös på vår enklare räknare eftersom man dividerar med n i första steget, och det gör att vi tappar signifikanta siffror. En rimlig gissning är att bästa svar blir då n är så stort att vi i stort dividerat bort hälften av siffrorna, vilket innebär att vi torde få kvar tre eller fyra signifikanta siffror. Det är också vad vi i praktiken finner i intervallet [-2,2]. För värden på x utanför detta intervall så räcker inte antalet kvadreringar till för att få ett någorlunda bra svar, innan divisionen i början tar bort allt för många signifikanta siffror, vilket ger oss två eller tre signifikanta siffror kvar i svaret.

E1 Så algoritm E1 skulle alltså vara algoritm E0, med n varandes den minsta potens av två (2^p) sådan att $|x|/n < 0,001$ alltså då vi i har hälften av siffrorna kvar.

2.1.1.1 Förbättring genom skalning.

Vi skulle kunna utnyttja att

$$e^x = e^h e^d \tag{2.9}$$

där h är heltalsdelen och d är decimaltalen av x.[8] Vi behöver då bara kunna beräkna e^d och sedan multiplicera med $e \approx 2.7182818$, h gånger.

Vilket värde på n, för en metod enligt $(1 + x/n)^n$, skulle då ge bästa resultat? Vi har å ena sidan ett fel i storleksordningen $x^2/2n$, och å andra sidan får vi kvar, då vi dividerar något som är strax under ett, $7 - \lg(n)$ siffror, eftersom trunkeringen gör så att en division med ett tal tar bort så många korrekta siffror som siffror i talet. Vi får alltså bästa möjliga resultat då

$$\lg\left(\frac{x^2}{2n}\right) = -7 + \lg n \tag{2.10}$$

[8] Detta är ett relativt standardmässigt sätt att beräkna e^x, förutom att e upphöjt till heltalsdelen beräknas på ett mycket effektivare sätt än vi gör här.

eller med andra ord för

$$n = \frac{x}{\sqrt{2}} \cdot 10^{7/2} \tag{2.11}$$

Om $x = 1$ så får vi $n \approx 2236$ som har som närmsta potens av två, $2^{11} = 2048$. Detta skulle ge oss $7 - \lg 2048 \approx 3,7$ korrekta siffror.

Vid praktiska försök får vi att man i stort alltid får mellan tre och fyra korrekta siffror, och att 2048 är den bästa divisorn som är en jämn potens av två. Algoritm E2 innebär alltså att vi beräknar $(1 + d/n)^n$, där d är decimaldelen av x och $n = 2048$. Vi multiplicerar sedan med e, $\lfloor x \rfloor$ gånger. I princip får man samma resultat som för E1, mellan 0 och 1, upprepat om och om igen. (Se figur 4a.) E2

2.1.2 Analys av algoritm baserad på algoritm L3 omvänt

För att få denna att fungera utan att behöva komma ihåg mellanresultat behöver vi arrangera om den lite. Vi skulle till exempel kunna räkna enligt

$$\left(\frac{x + \sqrt{x^2 + n^2}}{n}\right)^n = \left(\frac{\sqrt{\left(\frac{x^2}{n} + n\right) n + x}}{n}\right)^n \tag{2.12}$$

vilket ger:

x M+
$\times = \div n + n \times n \sqrt{\ } + \mathrm{MR} \div \mathrm{n}$
följt av p stycken $\times =$

Här är $n = 2^p$. Åter igen behöver vi veta hur felet påverkas av värdet på n, givet att $x \approx 0$. Vi kan i princip kopiera analysen från sektion 2.1.1. Om vi kallar uttrycket innanför limesoperatorn för I så får vi:

$$\ln I = \ln \left(\frac{x + \sqrt{x^2 + n^2}}{n}\right)^n = n \ln \left(\frac{x}{n} + \sqrt{1 + \frac{x^2}{n^2}}\right) \tag{2.13}$$

Eftersom $\sqrt{1 + z} = 1 + z/2 - z^2/8 + O(z^3)$ så får vi

$$\ln I = n \ln \left(\frac{x}{n} + 1 + \frac{x^2}{2n^2} + O(x^4)\right) \tag{2.14}$$

Vi utnyttjar att $\ln(1+z) = z - z^2/2 + z^3/3 + O(z^4)$, där vi har

$$z = \frac{x}{n} + \frac{x^2}{2n^2} + O(x^4/n^4) \tag{2.15}$$

$$-\frac{z^2}{2} = -\frac{x^2}{2n^2} - \frac{x^3}{2n^3} + O(x^4/n^4) \tag{2.16}$$

$$\frac{z^3}{3} = \frac{x^3}{3n^3} + O(x^4/n^4) \tag{2.17}$$

Detta ger oss

$$\ln I = x - \frac{x^3}{6n^2} - O(x^4/n^3) \tag{2.18}$$

Eftersom $e^z = 1 + z + O(z^2)$ så får vi

$$I = e^x \left(1 - \frac{x^3}{6n^2} - O(x^4/n^3)\right) \tag{2.19}$$

Samma sak som tidigare gäller för trunkeringen, så vi får att bästa möjliga resultat är då

$$\lg \frac{x^3}{6n^2} = -7 + \lg n \tag{2.20}$$

eller

$$n = \frac{x}{\sqrt[6]{3}} \cdot 10^{7/3} \tag{2.21}$$

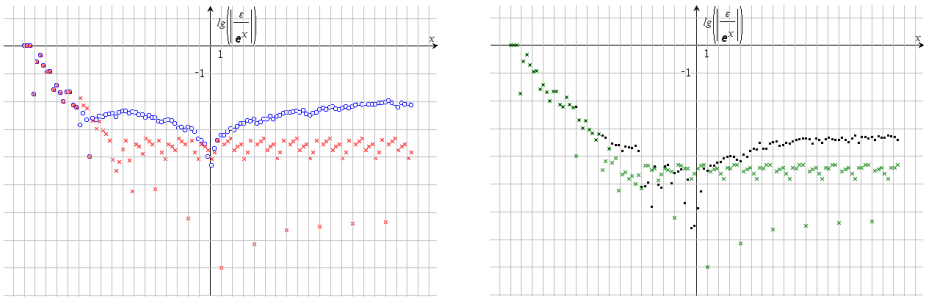
För $x = 1$ får vi då ≈ 119, och att $x/n \approx 0,01$. Närmaste potens av två är då 128, och torde alltså få runt en faktor $16 \approx 10^{1,2}$ bättre noggrannhet
E3 än algoritm E1 ger. Algoritm E3 skulle nu innebära att vi, för n väljer den minsta potens av 2 sådant att $|x|/n < 0,01$, och därefter beräknar enligt (2.13)

Vid försök så ger (i området $-10 < x < 18$) E3 i snitt ett fel som är ungefär $18 \approx 10^{1,26}$ gånger mindre än E1. Denna algoritm ger alltså i stort nästan exakt en korrekt siffra mer än E1.

Vi kan också använda metoden vi använde i algoritm E2, och beräkna $e^x = e^h e^d$, där vi beräknar e^d med samma metod som i algoritm E3, och ett
E4 fixt värde $n = 128$. Detta skulle ge oss algoritm E4. Denna algoritm ger oss i stort alltid en signifikant siffra mer än E2.

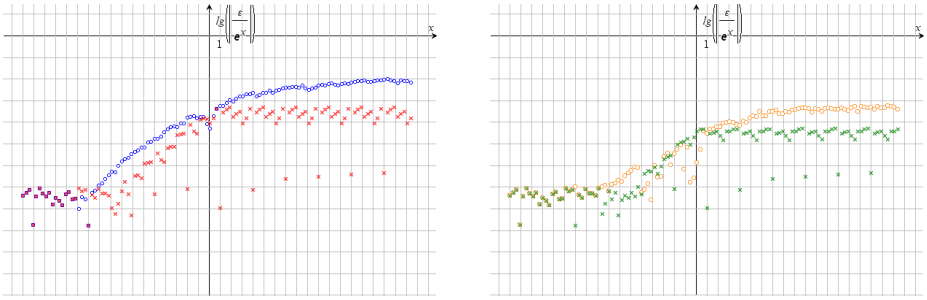
2.1.3 Vidare analys av E1 till E4

För ovanstående metoder kan vi alltså se att skalning genom att dela upp i en heltalsdel och en decimaltal ger bättre resultat över ett större område än att försöka optimera med olika värden på n.



(a) **(b)**

Figur 4: Logaritmen av de relativa felen kontra x. I delfigur a för E1 (cirklar) och E2 (kryss) och i delfigur b, E3 (prickar) och E4 (kryss)



(a) **(b)**

Figur 5: i denna figur visas den relativa positionen av den felet kontra positionen av den första visade siffran. I delfigur a för E1 (cirklar) och E2 (kryss) och i delfigur b, E3 (cirklar) och E4 (kryss)

Att, som man kan se i figur 4, det relativa felet går upp då $x < 0$ beror på förlusten av signifikanta siffror då $x < 0$. För till exempel $x = -12$ har vi att $e^x \approx 6,144 \cdot 10^{-6}$. Vår första algoritm ger oss här $6,1 \cdot 10^{-6}$, så det relativa felet blir förhållandevis stort, även om vi har sex korrekta siffror — nollorna inräknat.

Hur skulle vi kunna visa antalet korrekta siffror istället för relativa felet? Då $x > 0$ så ger det tio-logaritmen av det relativa felet i princip antalet korrekta siffror (-6,4 betyder att vi har sex korrekta siffror). Då $x < 0$ så får vi ett svar i formen $0,$*någonting*, och vi har vi i princip första felet vid siffran motsvarande $\lg|\varepsilon|$, där ε är det absoluta felet. Dessa två uttryck ger alltså i princip den relativa positionen på felet kontra den första visade siffran. För $x > 0$ sammanfaller graferna med de över de relativa felen. För $x < 0$ så är kontrasten stor. (Se figur 5.) Då x minskar får vi å ena sidan allt fler korrekta siffror, om vi räknar med de inledande nollorna, men å andra sidan får vi allt färre signifikanta siffror kvar på grund av trunkeringen efter åtta siffror.

2.2 Algoritmer baserade på serieutveckling

Vi kan också försöka oss på att använda (2.2). Denna har fördelarna mot serieutvecklingen för logaritmer att dels konvergera mycket hastigare, och dels att lättare låta sig omformas till en form lämplig att räkna med.

För att få en snabb konverteringstakt krävs dock att x är någorlunda nära noll. Om $x \approx 1$ så blir felet ungefär lika stort som den första utlämnade termen, åtminstone om felet skall vara litet, och antalet termer därför någorlunda stort. För ett fel på 10^{-7} får vi att $(n+1)! > 10^7$, viket ger $n = 10$, vilket ger elva termer. För 11 termer och $x = 1$ så ger serieutvecklingen enligt (2.2) ett fel på ungefär $3 \cdot 10^{-7}$. I praktiken räcker tio termer utom då x är väldigt nära 1. Värden på x längre bort från noll kräver snabbt många fler termer. För $x = 10$ krävs till exempel 39 termer.

2.2.1 Skalning av exponenten

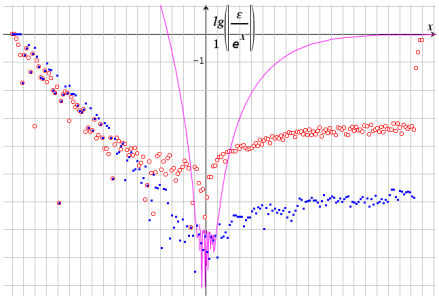
Vad vi kan göra är något liknande som vad vi gjorde med logaritmer. Vi har att

$$e^x = (e^{\frac{x}{2^n}})^{2^n} \tag{2.22}$$

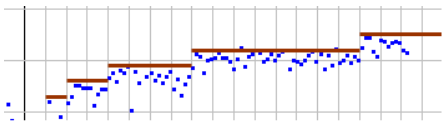
Det vill säga, vi dividerar först x med någon lämplig potens av 2 för att få exponenten tillräckligt litet. Sedan beräknar vi e upphöjt till detta, varefter vi upphöjer till 2^n igen genom n upprepade $\times =$. Om vi till exempel vill minska exponenten till mindre än 1 behöver vi aldrig dividera med mer än 32, eftersom $\ln 10^8 \approx e^{18,4}$ och vi därmed, med en åtta siffrors display, inte
E5 kan ha en exponent större än detta. I praktiken skulle en algoritm, E5, bli som

Om $|x| > 1$ dividera med det minsta av talen
2, 4, 8, 16 eller 32 för att få ett tal inom (-1,1).
M+ ÷ 9 + 1
× MR ÷ 8 + 1
× MR ÷ 7 + 1
× MR ÷ 6 + 1
× MR ÷ 5 + 1
× MR ÷ 4 + 1
× MR ÷ 3 + 1
× MR ÷ 2 + 1
× MR + 1
Upprepa sedan × = så många gånger som motsvarar
den eventuella divisionen i början (2⇒ en gång,
4⇒ två gånger, och så vidare)

I figur 6a ses logaritmen av det relativa felet för denna algoritm. Som synes får man, för $x > 0$ alltid minst fem korrekta siffror, och i intervallet [1,8] får man minst sex korrekta siffror.



(a) Logaritmen av de relativa felen kontra x.



(b) Delfigur med linje enligt teori.

Figur 6: I delfigur a ses logaritmen av de relativa felen kontra x, för algoritm E2, (cirklar) och för algoritmen beskriven i denna sektion, E5, (prickar). Den heldragna linjen motsvarar serieutvecklingen av e^x beräknat till x^9-termen. I delfigur b ses en del av figur a, men nu med en linje enligt teori inlagd.

I figuren kan man se ganska tydliga språng för denna algoritm då x är lika med 2, 4, 8, 16 och 32, och vid varje steg förloras signifikanta siffror. Eftersom en division med två motsvarar en förskjutning av siffrorna i ett decimaltal med $\lg 2 \approx 0,3$ steg åt höger, så kommer detta motsvara förlusten av signifikanta siffror. En praktisk test visar att $-6,7 + \lfloor \log_2 x \rfloor \cdot \lg 2$, för $x > 1$ passar de övre felen väldigt väl.

2.2.2 Separat beräkning av heltalsdelen och decimaldelen

E6 Självklart kan vi använda samma knep som för algoritm E2 och E4 och beräkna $e^x = e^h e^d$. I denna algoritm, E6, får vi i jämförelse med E5 i stort en halv signifikant siffra till då $x > 0$ och i stort samma svar då $x < 0$

2.3 Jämförelse av algoritmerna

A:↓ x: →	3,45		-3,45	
e^x	31,500392	ε	0,031746	ε
E1	31,439713	-0,0606790	0,031807	0,0000613
E2	31,492846	-0,0075460	0,031753	0,0000076
E3	31,497754	-0,0026380	0,031744	-0,0000017
E4	31,499422	-0,0009700	0,031747	0,0000010
E5	31,500381	-0,0000110	0,031746	0,0000000
E6	31,500387	-0,0000050	0,031746	0,0000000

Tabell 4: Resultatet av algoritm E1 till E6 för två värden. Vidare redovisas de absoluta felen, ε.

Algoritm E1 är otroligt enkel, men ger till och med sämre resultat än L1. Algoritm E6 är kanske den som är lättast att komma ihåg av de andra eftersom den i stort följer hur vi normalt skulle beräkna e^x, och dessutom har den fördelen att vara den mest exakta.

2.3.0.1 Ett litet knep då $x < 0$. Om man är villig att inte få det korrekta svaret presenterat i displayen kan man i praktiken få lika god noggrannhet då $x < 0$ som då $x > 0$. Man börjar med att beräkna $e^{|x|} = e^{-x}$ enligt väl vald algoritm. Säg att vi använder algoritm E5. Vi har då att $e^x = 1/e^{-x}$, men istället för att invertera direkt dividerar man först med någon lämplig potens av 10. Låt oss till exempel beräkna e^{-12}. Vi beräknar först e^{12} till 162754,59. Vi dividerar med 100000 för att få 1,6275459. Den multiplikativa inversen av detta blir nu 0,6144219, som vi då vet motsvarar $6,144219 \cdot 10^{-6}$. Vi har att $e^{-12} \approx 6,144212 \cdot 10^{-6}$, så noggrannheten är, inför omständigheterna, mycket god. Självklart kan samma knep användas i de andra algoritmerna för e^x.

3 Lite utvidgning till komplexa argument

Om man mot all (och då menas all) förmodan skulle stöta på en räknare av detta slag som skulle klara av att hantera komplexa tal, så skulle våra

algoritmer i princip fungera. Vi kan till exempel titta på algoritm ett för logaritmer. Om vi låter $x = re^{i\theta}$, där $\theta = \alpha + 2\pi n$, $n \in \mathbb{Z}$,och α är det värde i intervallet $(-\pi, \pi]$ för vilket $x = re^{i\alpha}$, så har vi

$$\ln\left(re^{i\theta}\right) = \ln r + i\theta \tag{3.1}$$

Detta innebär att vi enkelt kan räkna ut logaritmen för komplexa tal givet argumentet. Är talet givet i rektangulär form så har vi dock inte argumentet, och givet räknarens begränsningar så har vi inte heller de nödvändiga trigonometriska funktionerna för att finna vinkeln — så vi behöver undersöka om våra algoritmer fungerar så som de är givna. För (1.3) får vi

$$\begin{aligned} p(x^{1/p-1}) &= p\left(\left(re^{i\theta}\right)^{1/p} - 1\right) = p\left(r^{1/p}e^{i\theta/p} - 1\right) \\ &= p\left(r^{1/p}\left(\cos\frac{\alpha}{p} + i\sin\frac{\alpha}{p}\right) - 1\right) \\ &= p\left(r^{1/p}\left(1 + i\frac{\alpha}{p} + O(1/p^2)\right) - 1\right) \\ &= p\left(r^{1/p} - 1\right) + r^{1/p}\left(i\alpha + O(1/p)\right) \end{aligned} \tag{3.2}$$

Detta ger oss

$$\begin{aligned} \lim_{p\to\infty} p(x^{1/p} - 1) &= \lim_{p\to\infty}\left(p\left(r^{1/p} - 1\right) + r^{1/p}\left(i\alpha + O(1/p)\right)\right) \\ &= \ln r + i\alpha \end{aligned} \tag{3.3}$$

Alltså gäller (1.9), och därmed första algoritmen, L1, även för komplexa tal, och svaren vi får är då principalvärdet av logaritmen.

På liknande sätt kan visas att den andra algoritmen, L2, går mot samma svar. Dock skall vi här göra det på ett annat sätt.

Deriverar vi (3.1) med avssende på r så får vi $1/r$, och deriverar vi med θ så får vi i. Deriverar vi

$$p(1 - x^{-1/p}) = p(1 - r^{-1/p}e^{-i\theta/p}) \tag{3.4}$$

med avssende på r så får vi $r^{-1-1/p}$, som går mot $1/r$ då $p \to \infty$. Deriverar vi med avseende på θ så får vi $ie^{-i\theta/p}$, som går mot i då $p \to \infty$. Logaritmen och (1.10) har alltså i alla punkter där derivatan är definierad, samma derivata. Dessutom sammanfaller de i åtminstone en punkt, vilket visar, via entydighetssatsen för analytiska funktioner att funktionerna lika.

Detta innebär därmed att även tredje algoritmen, L3, fungerar för komplexa tal eftersom den helt enkelt innebär beräknande av medelvärdet av de två nämnda algoritmerna. För att i praktiken få det att fungera måste man

dock modifiera skalningen i första steget, till exempel genom att dra roten ur tills både realdelen och imaginärdelen är mindre än 1.01. (Kvarstår dock problemet om realdelen och imaginärdelen är väldigt olika stora, men vi kan nog anse att det problemet är faller en bit utanför syftet med detta arbete — eftersom räknaren nu inte klarar av komplexa tal.)

Angående algoritmerna baserad på serieutveckling så har serieutvecklingen en konvergensradie på 1 i det komplexa talplanet, och våra algoritmer fungerat åter igen.

Gällande algoritmerna för exponentialfunktionen, kan man i princip föra samma resonemang som för logaritmfunktionerna. Vi kan också derivera

$$u = \lim_{n\to\infty} \left(1 + \frac{x}{n}\right)^n \tag{3.5}$$

med avseende på a respektive b för att få

$$\frac{\partial u}{\partial a} = \lim_{n\to\infty} \left(1 + \frac{a+bi}{n}\right)^{n-1} = u \tag{3.6}$$

respektive

$$\frac{\partial u}{\partial b} = \lim_{n\to\infty} i\left(1 + \frac{a+bi}{n}\right)^{n-1} = iu \tag{3.7}$$

Det vill säga

$$i\frac{\partial u}{\partial a} = \frac{\partial u}{\partial b} \tag{3.8}$$

och därmed, via Cauchy–Riemanns ekvation [1, s. 483 ff] ovan att u är en analytisk funktion, som dessutom sammanfaller med e^x för reellvärda invärden. Via entydighetsteoremet för analytiska funktioner [1, s. 484] kan vi alltså dra slutsatsen att funktionerna är lika över hela komplexa planet.

Att serieutvecklingen för e^x gäller för hela komplexa talplanet kan väl anses vara allmänt känt.

Skalningarna via manipuleringar av potenserna (divisioner, kvadratrötter ur, kvadreringar) följer de vanliga potensreglerna, som gäller även för komplexa tal.

4 Slutord

I detta arbete undersöktes ett antal algoritmer med avseende på användbarhet och noggrannhet. Ett par mycket enkla algoritmer som ger runt tre korrekta siffror undersöktes. Vidare fann vi algoritmer, baserade på serieutvecklingar, som kan ge fem á sju korrekta siffror, men kanske är algoritm L3, som bygger på att

$$\ln x = \lim_{p\to\infty} \frac{p}{2}(x^{1/p} - x^{-1/p}) \tag{4.1}$$

den teoretisk mest intressanta algoritmen.

Under arbetet med detta undersökte författaren även diverse modifieringar av algoritmerna som inte redovisats här. Till exempel med att modifiera nämnaren i $(1 + x/n)^n$ på liknande sätt som vi provade för logaritmalgoritmerna. Dessa försök ledde dock vanligtvis till förbättringar som inte motsvarade hur mycket mer komplicerade algoritmerna blev. Samma sak gäller diverse försök till att reducera gradtalet på de använda serieutvecklingarna genom att till exempel använda Chebyshev polynom och teleskopering [3, s. 184 ff.], men det ger polynom med koefficienter som inte är särdeles lätta att komma ihåg.

Referensdelen till detta arbete är mycket kort, mest beroende på att de flesta verktyg som används nog kan betraktas som matematiska allmängods, såsom taylorutveckling, binominalutvekling ellller relativt enkla derivator eller integraler. Om inte annat kan allt detta refereras till [1].

Möjlig vidare utvidgning av detta arbete vore till andra funktioner, och då främst till de trigonometriska funktionerna. Här torde serieutveklingarna av dessa vara en bra startpunkt.

På det personliga planet gav detta arbete mig ett ypperligt tillfälle att lära mig LaTeX. Jag vill också passa på att tacka min handledare Vera Koponen på Uppsala Universitet.

5 Appendix

5.1 Appendix 1: Några av progammen.

Koden är skriven för Texas Instruments Nspire® programvara.

Nedanstående kod trunkerar ett tal till max åtta signifikanta siffror. Är absolutvärdet av talet mindre än 10^{-7} så trunkeras det till 0, och är det större än 99999999 så är svaret ogiltigt. I Nspire indikeras ett ickevärde med "_".

```
Define r8(x)=
  Func
    Local s,p
    s:=sign(x)
    x:=abs(x)
    If x<1.E-7 Then
      0
    ElseIf x≥100000000 Then
      _
    Else
      If x<1 Then
        p:=10000000.
      Else
        p:=10^(7-floor(log(x,10)))
      EndIf
      x:=s*floor(x*p)/p
    EndIf
  EndFunc
```

Den andra listningen är över tredje algoritmen för logaritmer, L3, inklusive invers om $x < 1$.

```
Define log8ti(x,n)=
  Func
    Local i,y,s
    x:=r8(x)
    If x<1 Then
      s:=-1
      x:=r8(1/x))
    Else
      s:=1
    EndIf
    For i,1,n
      x:=r8(√(x))
```

```
    EndFor
    y:=r8(1/x))
    x:=r8(x-y)
    r8(x*2^(n-1))*s
EndFunc
```

5.2 Appendix 2: Algoritmerna.

I dessa algoritmer antas att minnet är nollställt i början.

5.2.1 Logaritmer.

Algoritm L1:

> Skriv in talet som skall tas logaritmen ur.
> Tag $\sqrt{}\ \sqrt{}\ \cdots\sqrt{}$,
> så många gånger som behövs tills svaret är inom området
> (0,999; 1,001), det vill säga att man får tre nollor
> eller tre nior efter decimalkommat.
> $-\ 1$
> $\times$ 1024, eller motsvarande 2 upphöjt till ett
> mindre än antalet $\sqrt{}$, följt av $=$.

Algoritmen ger tre á fyra korrekta siffror.

Algoritm L2:

> Skriv in talet som skall tas logaritmen ur.
> Tag $\sqrt{}\ \sqrt{}\ \cdots\sqrt{}$,
> så många gånger som behövs tills svaret är inom området
> (0,999; 1,001), det vill säga att man får tre nollor
> eller tre nior efter decimalkommat.
> M+
> $1\ \div$ MR
> $-\ 1$
> $\times$ 128, eller motsvarande 2 upphöjt till ett
> mindre än antalet $\sqrt{}$, följt av $=$.

Algoritm L2 ger inte bättre svar än L1, men är aningen krångligare.

Algoritm L3 är i blir praktiken nästa samma som L2, förutom den möjliga divisionen i början och att man subtraherar med innehållet i minnet istället

för med 1. Algoritmen ger dock avsevärt bättre resultat.

Algoritm L3:

> Om talet <1 skriv in 1 $\div$
> Skriv in talet som skall tas logaritmen ur.
> Tag $\sqrt{}\ \sqrt{}\ \cdots \sqrt{}$,
> så många gånger som behövs tills man får
> ett tal <1,01
> M+
> 1 $\div$ MR
> $-$ MR
> $\times$ 128, eller motsvarande 2 upphöjt till ett
> mindre än antalet $\sqrt{}$, följt av $=$.

Om talet från början <1 så byt tecken på svaret.

Algoritmen ger runt fem korrekta siffror.

Algoritm L4 kräver att man kommer ihåg mellanresultat, så den uppfyller inte riktigt kraven. Algoritm L5 finns beskriven på sidan 19.

5.2.2 Exponentialfunktionen e^x.

Algoritm E1:

> Välj det minsta värde på p sådant att $|x|/n < 0,001$,
> där $n = 2^p$. Beräkna sedan
> x $\div$ n
> $\times =$ ... $\times =$, upprepat p gånger.

Denna algoritm ger minst tre signifikanta siffror i intervallet [2,2].

Algoritm E2.

> Skriv in 2.7182818 M+
> Om talet är negativt, byt tecken.
> Dela upp talet i en heltalsdel h och en decimaldel, d.
> Beräkna sedan
> d $\div$ 2048
> $\times =$ ... $\times =$, upprepat 11 gånger.
> Tag sedan $\times$ MR, h gånger, $=$
> Var talet negativt till att börja med, beräkna
> $-$ MR $=$ M+ 1 $\div$ MR $=$

Algoritm E3:

> Välj det minsta värde på p sådant att $|x|/n < 0,01$,
> där $n = 2^p$. Beräkna sedan
> x M+ $\times$ = $\div$ n + n $\times$ n $\sqrt{}$ + MR $\div$ n
> följt av p stycken $\times$ =

Algoritmen ger i stort en signifikant siffra mer än E1.

Algoritm E4

> Skriv in 2.7182818 M+
> Om talet är negativt, byt tecken.
> Dela upp talet i en heltalsdel h och en decimaldel, d.
> Beräkna sedan
> x M+ $\times$ = $\div$ n + n $\times$ 128 $\sqrt{}$ + MR $\div$ n
> följt av 7 stycken $\times$ =
> Var talet negativt till att börja med, beräkna
> $-$ MR = M+ 1 $\div$ MR =

Algoritmen ger i stort en signifikant siffra mer än E2.

Algoritm E5 finns beskriven på sidan 27. Algoritmen ger alltid minst fem korrekta siffror, och i intervallet [1,8] får man minst sex korrekta siffror.

Algoritm E6

> Skriv in 2.7182818 M+
> Om talet är negativt, byt tecken.
> Dela upp talet i en heltalsdel h och en decimaldel, d.
> Beräkna sedan
> M+ $\div$ 9 + 1
> $\times$ MR $\div$ 8 + 1
> $\times$ MR $\div$ 7 + 1
> $\times$ MR $\div$ 6 + 1
> $\times$ MR $\div$ 5 + 1
> $\times$ MR $\div$ 4 + 1
> $\times$ MR $\div$ 3 + 1
> $\times$ MR $\div$ 2 + 1
> $\times$ MR + 1
> Tag sedan $\times$ MR, h gånger, =

Var talet negativt till att börja med, beräkna
− MR = M+ 1 ÷ MR =

Algoritmen ger, då $x > 0$ runt en halv signifikant siffra mer än E5, annars i stort samma.

5.3 Appendix 3: Lamberts W-funktion

Vid ett par tillfällen under arbetets gång behövdes Lamberts W-funktion. Denna är definierad så att om

$$y = we^w \tag{5.1}$$

så är

$$w = W(y) \tag{5.2}$$

På grund av att inversen av (5.1) inte är entydig, så har vi egentligen, för vissa reella värden, två funktioner, en övre, W_0, och en nedre, W_{-1}. Vi använder här den övre funktionen. W-funktionen tillåter oss att lösa många olika funktioner, till exempel av typen

$$nu^n = v \tag{5.3}$$

Vi kan skriva om denna till

$$ne^{n \ln u} = v \tag{5.4}$$

Vi förlänger sedan detta till

$$n \ln(u) e^{n \ln u} = v \ln u \tag{5.5}$$

varvid vi har fått en ekvation i rätt form för att lösas med W-funktionen. Det tillåter oss även att lösa funktionen av formen (1.54), det vill säga $z^n/n = \varepsilon$. Vi tar bara den multiplikativ inversen av bägge sidor för att sedan, i princip, fortsätta som med (5.3).

W-funktionen finns inte implementerad i Nspire, så en funktion skrevs för att kunna beräkna denna.

```
Define w0(x)=
  Func
    Local v,vo,i,ei,l1,l2
    vo:=-1.E99
    i:=0
    x:=approx(x)
    ei:=e^(-1)
```

```
If x<-ei Then

Else
  If x>8 Then
    l1:=ln(x)
    l2:=ln(l1)
    v:=l1-l2+l2/l1
  ElseIf x>0 Then
     v:=1.22*(1.28*x+0.26)^(0.3)-0.82
  ElseIf x>-0.25 Then
     v:=x-x^2+2/3*x^3
  Else
     v:=2*√(x-ei)-1
  EndIf
  While abs(v-vo)>1.E-10 and i<20
       vo:=v
       v:=v-(v-x*e^(-v))/(1+v)
       i:=i+1
    EndWhile
    v
  EndIf
EndFunc
```

Detta är en implementation som använder sig av Newton-Raphsons metod [1, s. 90 ff.]. Som första steg väljs de tre första termerna av asymptotisk approximation nämnd i [7]. För $x \leq 8$ används två uttryck anpassade för hand, utom i det mittersta området $-0.25 < x < 0$, där de första tre termerna i maclaurinutvecklingen av W_0 används.

Referenser

[1] Bronshtein, I. N. och Semendyayev, K. A., *Handbook of mathematics*. Verlag Harri Deutsch, Thun and Frankfurt/Main, tredje upplagan, 1985.

[2] Böiers Lars-Christer och Persson Arne *Analys i en variabel*. Studentlitteratur, Lund, andra upplagan, 2001.

[3] Pennington, Ralph H. *Introductory Computer Methods and Numerical Analysis* The MacMillan company Collier-MacMillan Canada Ltd., Toronto, andra upplagan, 1970.

[4] Simmons, George F. *Differential equations whith applications and historical notes*. McGraw-Hill, Inc., Singapore, andra upplagan, 1991.

[5] Spiegel, Murray R. *Theory and problems of Advanced Calculus*. McGraw-Hill, Book Company (UK) Limited, London, metric edition, 1974.

[6] Pathan, Tehsinkhan. *How to find log by using simple calculator*. http://ebookbrowsee.net/461361-729372-how-to-find-log-by-using-simple-calculator-doc-d331611519 (2012-12-28).

[7] Wolfram Mathworld, uppslagsord *Lambert W-Function*. http://mathworld.wolfram.com/LambertW-Function.html (2014-01-11).