1 Lösningsförslag till övningarna

1.1 Lösningar till kapitel 1

1: Absoluta felet ≤ 0.0005 (en halv enhet i sista decimalen). Relativa felet $\leq 0.0005/(24.516-0.0005) \approx 2\cdot 10^{-5}$.

2: När vi härledde uppskattningen antog vi att koefficienterna utsattes för små störningar. Det är inte helt uppenbart vad som är litet, utan vi får testa lite. $p(x) = x^2-3x+2$ har nollställena 1 och 2 varför uppskattningarna av konditionstalen blir:

$$\frac{|-3|+|2/1|}{2-1} = 5, \quad \frac{|-3|+|2/2|}{2-1} = 4$$

Tag $\delta_a = 3\cdot 10^{-4}$, $\delta_b = 2\cdot 10^{-4}$. Då är $|\delta_a|/|a| = |\delta_b|/|b| = 10^{-4}$.

Vi får då $|\delta_r|/|r| \lessapprox 5\cdot 10^{-4}$ respektive $4\cdot 10^{-4}$.

```
r =   1.0005e+00  % beräknade rötter
      1.9992e+00

rel = 5.0040e-04  % relativa fel; stämmer bra!
      4.0020e-04
```

$p(x) = x^2-1.99x+0.99$ har nollställena 0.99 och 1 så att uppskattningarna av konditionstalen blir 298 respektive 299. Med $\delta_a = 2\cdot 10^{-4}$, $\delta_b = 10^{-4}$ blir $|\delta_r|/|r| \lessapprox 0.03$.

```
r =   9.9490e-01+  1.6553e-02i
      9.9490e-01-  1.6553e-02i

rel = 1.7321e-02    % relativt fel, jämfört med 1
```

Detta är för stora störningar för att satsen skall fungera bra. Om vi stör mindre och byter tecken på δ_a, stämmer det dock bra. Med $\delta_a = -2\cdot 10^{-6}$, $\delta_b = 10^{-6}$ blir $|\delta_r|/|r| \lessapprox 3\cdot 10^{-4}$.

```
r =   9.9969e-01
      9.9031e-01

rel = 3.0739e-04
     -3.0848e-04
```

3: Vi vet att $\hat{x}-\delta \leq x \leq \hat{x}+\delta$. $f(x) = f(\hat{x}+x-\hat{x}) = f(\hat{x}) + (x-\hat{x})f'(\xi), \xi \in (x,\hat{x})$, så $|f(x)-f(\hat{x})| \leq \delta \max_{\xi\in(\hat{x}-\delta,\hat{x}+\delta)} |f'(\xi)|$.
Om f är linjär (första fallet) så är derivatan konstant, 7 i exemplet, varför absoluta felet begränsas av 7δ. I andra fallet får vi begränsningen $2\delta(|\hat{x}|+\delta)$ eftersom derivatan, $2x$, är strängt växande.

4:

$$\sum_{k=1}^{n}\frac{1}{1.1R_k} \leq \frac{1}{R} \leq \sum_{k=1}^{n}\frac{1}{0.9R_k} \Rightarrow \frac{0.9}{\sum_{k=1}^{n}\frac{1}{R_k}} \leq R \leq \frac{1.1}{\sum_{k=1}^{n}\frac{1}{R_k}}$$

För seriekoppling gäller:

$$\sum_{k=1}^{n} 0.9R_k \leq R \leq \sum_{k=1}^{n} 1.1R_k$$

5: Frågan är alltså: hur ändras $f(x)$ när vi ändrar x lite? Taylors formel ger $f(x+\delta_x) = f(x) + \delta_x f'(x) + \cdots$. Den absoluta förändringen är: $f(x+\delta_x) - f(x) \approx \delta_x f'(x)$. Om x och $f(x)$ är skilda från noll kan vi studera relativa förändringar:

$$\frac{f(x+\delta_x) - f(x)}{f(x)} \approx \frac{xf'(x)}{f(x)} \frac{\delta_x}{x}$$

eller

$$\left| \frac{f(x+\delta_x) - f(x)}{f(x)} \right| \approx \underbrace{\left| \frac{xf'(x)}{f(x)} \right|}_{\kappa} \left| \frac{\delta_x}{x} \right|$$

där κ är en uppskattning av konditionstalet.

Då $f(x) = \cos x$ får vi

$$\kappa = \left| \frac{xf'(x)}{f(x)} \right| = \left| \frac{x \sin x}{\cos x} \right|$$

När $x = \delta > 0$ ($x = 0$ ger division med noll; dessutom är $\sin 0 = 0$, så för att få en uppskattning får man titta på nästa term i Taylorutvecklingen) kan vi gör följande approximation, för att lättare kunna analysera vad som händer:

$$\kappa \approx \delta^2$$

ty $\sin x \approx x$ och $\cos x \approx 1$ för $x \approx 0$. När $x = \pi/2 - \delta$ får vi

$$\kappa = \left| \frac{(\pi/2 - \delta)\sin(\pi/2 - \delta)}{\cos(\pi/2 - \delta)} \right| \approx \left| \frac{\pi/2 \cdot 1}{\delta} \right|$$

Detta κ växer alltså som $1/\delta$ och kan bli mycket stort. Här följer en numerisk kontroll, $x = \delta = 10^{-3}$ och $\delta_x = 10^{-6}$ ger:

```
>> x = 1e-3; delta_x = 1e-6;
>> kappa = abs(((cos(x + delta_x) - cos(x)) / cos(x)) / (delta_x / x))
kappa = 1.0005e-06
```

vilket stämmer bra med x^2.

```
>> x = pi / 2 - 1e-3; delta_x = 1e-6;
>> kappa = abs(((cos(x + delta_x) - cos(x)) / cos(x)) / (delta_x / x))
kappa = 1.5698e+03
```

vilket stämmer bra med $\pi/(2\delta)$.

6: Derivatan av en funktion behöver inte vara begränsad även om funktionen är det. Tag t.ex. $\delta(x) = \epsilon \cos(\omega x)$. Funktionen är tydligen begränsad, men derivatan kan vara godtyckligt stor om vi väljer ett stort ω (hög frekvens medför stor derivata). En liten störning av f kan alltså ändra derivatan godtyckligt mycket. Detta gäller inte integralen. Vi har ju:

$$\left| \int_a^b f(x) + \delta(x) dx - \int_a^b f(x) dx \right| = \left| \int_a^b \delta(x) dx \right| \leq (b-a)\epsilon$$

7: Vi kan beräkna $\mathbf{x}$ explicit genom $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$ förutsatt att inversen existerar, dvs. då $|\alpha| \neq 1$. Så:

$$\mathbf{x} = \frac{1}{1-\alpha^2} \begin{bmatrix} 1 & -\alpha \\ -\alpha & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \frac{1}{1-\alpha^2} \begin{bmatrix} 1 \\ -\alpha \end{bmatrix}$$

Om $|\alpha| \approx 1$ kommer små variationer i α att ge upphov till stora förändringar i $\mathbf{x}$. Låt, t.ex. $\alpha = 1-\epsilon$, då blir $\mathbf{x}$

$$\mathbf{x} = \frac{1}{1-(1-\epsilon)^2}\begin{bmatrix} 1 \\ -(1-\epsilon) \end{bmatrix} \approx \frac{1}{2\epsilon}\begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

Små variationer i ϵ ger upphov till stora variationer i $\mathbf{x}$. Beräkningen av $\mathbf{x}$ är således illakonditionerad då $|\alpha| \approx 1$.

En alternativ härledning är att beräkna derivatan av $\mathbf{x}$ som funktion av α. Vi utgår från $\mathbf{A}(\alpha)\mathbf{x}(\alpha) = \mathbf{b}$, där jag skrivit ut (α) för att betona beroendet av denna parameter ($\mathbf{b}$ beror inte av α utan är en konstant vektor). Om prim får beteckna elementvis derivering med avseende på α kan man visa att följande operationer är tillåtna. $[\mathbf{A}(\alpha)\mathbf{x}(\alpha)]' = \mathbf{b}'$ ger

$$\mathbf{A}'(\alpha)\mathbf{x}(\alpha) + \mathbf{A}(\alpha)\mathbf{x}'(\alpha) = \mathbf{0}$$

så att

$$\mathbf{x}'(\alpha) = -\mathbf{A}^{-1}(\alpha)\mathbf{A}'(\alpha)\mathbf{x}(\alpha) = -\mathbf{A}^{-1}(\alpha)\mathbf{A}'(\alpha)\mathbf{A}^{-1}(\alpha)\mathbf{b}$$

Derivatan blir stor när $\mathbf{A}^{-1}(\alpha)$ innehåller stora element, dvs. när $|\alpha| \approx 1$. Det går givetvis att derivera $x_1(\alpha)$ och $x_2(\alpha)$ på vanligt sätt (vi har ju uttryck för $x_1(\alpha)$ och $x_2(\alpha)$).

8: Denna övning är snarlik den föregående. Vi kan återigen derivera, men nu beror $\mathbf{x}$ både på α och β så vi får använda partiella derivator.

$$\mathbf{x}(\alpha,\beta) = \frac{1}{\alpha^2-\beta^2}\begin{bmatrix} \alpha \\ -\beta \end{bmatrix}$$

som är känslig för störningar då $|\alpha| \approx |\beta|$.

9: Lösningen är $y(t) = e^{ct}y_0$. Låt $z(t)$ vara lösningen till det störda problemet: $z'(t) = cz(t), z(0) = y_0 + \delta$. Tydligen är $z(t) = e^{ct}(y_0+\delta)$ så att $z(t) - y(t) = e^{ct}\delta$. Om $c > 0$ kommer $|z(t) - y(t)|$ att växa; större c ger snabbare tillväxt. Om $c < 0$ så kommer skillnaden att avta och om $c = 0$ så är den konstant.

10: Denna uppgift kräver kunskap om hur man löser linjära differensekvationer (jag tar inte upp teorin här). Karakteristiska polynomet är $r^2 = r+1$ så att rötterna är $r_1 = (1+\sqrt{5})/2 \approx 1.62$ och $r_2 = (1-\sqrt{5})/2 \approx -0.62$. Den allmänna lösningen har formen $a_k = c_1 r_1^k + c_2 r_2^k$. Vi bestämmer c_1 och c_2 med begynnelsevärdena, a_0 och a_1, och får slutligen:

$$a_k = \frac{r_2 a_0 - a_1}{r_2 - r_1} r_1^k + \frac{-r_1 a_0 + a_1}{r_2 - r_1} r_2^k$$

Låt nu b_k vara en störd talföljd definierad genom: $b_{k+2} = b_{k+1} + b_k$, $b_0 = a_0 + \delta$ och $b_1 = a_1 + \epsilon$. Eftersom a_0 och a_1 ingår linjärt i a_k, så ges skillnaden, $b_k - a_k$, av:

$$b_k - a_k = \frac{1}{r_2 - r_1}\left[(r_2\delta - \epsilon)r_1^k + (-r_1\delta + \epsilon)r_2^k\right]$$

Den relativa förändringen ges av:

$$\frac{b_k - a_k}{a_k} = \frac{\frac{1}{r_2-r_1}\left[(r_2\delta-\epsilon)r_1^k + (-r_1\delta+\epsilon)r_2^k\right]}{\frac{r_2a_0-a_1}{r_2-r_1}r_1^k + \frac{-r_1a_0+a_1}{r_2-r_1}r_2^k}$$

Om $r_2 a_0 - a_1 \neq 0$ så kommer $(b_k - a_k)/a_k$ att konvergera mot $(r_2\delta - \epsilon)/(r_2 a_0 - a_1)$ då $k \to \infty$ (eftersom $r_1 > |r_2|$). Det relativa felet stannar alltså på en viss nivå. Om däremot $r_2 a_0 - a_1 = 0$ så avtar $|a_k|$ eftersom $|r_2| < 1$. Är då koefficienten, framför r_1 i uttrycket för b_k, skild från noll så kommer felet att växa. Dvs. om $r_2(a_0+\delta) - a_1 - \epsilon \neq 0$ så kommer $|b_k| \to \infty$ men $|a_k| \to 0$.

11: Låt $p(x) = a_n x^n + \cdots + a_1 x + a_0$ med distinkta nollställen $r_1, \ldots, r_n$. Vi vill veta hur känslig ett nollställe, r_k, är för små ändringar i koefficienterna, dvs. vi vill studera $\partial r_k(a_0, a_1, \ldots, a_n)/\partial a_j, j = 0, 1, \ldots, n, k = 1, \ldots, n$. För att få enklare formler låter vi r beteckna ett av nollställena (vi får samma härledning oberoende av vilket

---

nollställe vi arbetar med). $\partial r(a_0, a_1, \ldots, a_n)/\partial a_j$ kommer jag att beteckna med r' (vi måste då komma ihåg att vi deriverar med avseende på just a_j). Observera dock att olika nollställen kan vara olika känsliga och att det spelar stor roll vilken koefficient vi stör.

Låt oss först se på specialfallet $n = 3$. Eftersom r är ett nollställe, $p(r) = 0$, så gäller:

$$a_3 r^3 + a_2 r^2 + a_1 r + a_0 = 0$$

Vi deriverar nu detta uttryck med avseende på a_2 och får (med våra beteckningar):

$$3a_3 r^2 r' + r^2 + 2a_2 r r' + a_1 r' = 0$$

Om vi låter $p'(x)$ beteckna derivatan av $p(x)$ med avseende på x, dvs. $p'(x) = 3a_3 x^2 + 2a_2 x + a_1 x$ så ser vi att

$$3a_3 r^2 r' + r^2 + 2a_2 r r' + a_1 r' = r' p'(r) + r^2$$

Uppgift: övertyga Dig om att när vi deriverar $p(r) = 0$ med avseende på a_j för allmänt n så får vi sambandet: $r' p'(r) + r^j = 0$. Detta ger omedelbart följande uttryck för derivatan, $r' = -r^j/p'(r)$. $p'(r)$ är lite svårtolkat, så vi gör följande omskrivning (åter specialfallet $n = 3$):

$$p(x) = a_3(x - r_1)(x - r_2)(x - r_3) \;\Rightarrow\; p'(x) = a_3\left[(x - r_2)(x - r_3) + (x - r_1)(x - r_3) + (x - r_2)(x - r_3)\right]$$

varför (r är ju en av r_1, r_2 eller r_3)

$$p'(r) = a_3 \prod_{s=1,\ldots,3, r_s \neq r} (r - r_s)$$

I det allmänna fallet gäller väsentligen samma formel, dvs.

$$p'(r) = a_n \prod_{s=1,\ldots,n, r_s \neq r} (r - r_s)$$

Vi vill slutligen få en relativ feluppskattning och sammanställer ovanstående. Låt ändringarna i r_k och a_j betecknas med δ_{r_k} respektive δ_{a_j}. Vi utnyttjar (som vanligt) att $r' \approx \delta_{r_k}/\delta_{a_j}$ och får (givet att $r_k \neq 0$):

$$\left|\frac{\delta_{r_k}}{r_k}\right| \approx \underbrace{\left|\frac{a_j r_k^{j-1}}{a_n \prod_{s\neq k}(r_k - r_s)}\right|}_{\kappa} \left|\frac{\delta_{a_j}}{a_j}\right|$$

När $n = 2$ känner vi igen resultaten från föreläsningen. Produkten består då bara av gapet, $|r_1 - r_2|$. När $n > 2$ spelar gapet också en viktig roll. Låt oss se på två exempel: låt nollställena vara $1, 1 \pm \epsilon, 1 \pm 2\epsilon, \ldots, 1 \pm p\epsilon$ (så att $n = 2p + 1$). Tag $r = 1$, produkten kommer då att bli $(\epsilon^p p!)^2$. Genom att hålla p fixt och låta $\epsilon \to 0$ kan vi göra gapen godtyckligt små. $\kappa \to \infty$ då $\epsilon \to 0$. Ett, eller flera, små gap leder till ett stort konditionstal. Om r har långt till alla övriga nollställen, stora gap, så blir κ normalt inte så stort. Låt nollställena vara $1, \lambda + 1 + \epsilon, \lambda + 1 + 2\epsilon, \ldots, \lambda + 1 + (n-1)\epsilon$, där λ är mycket större än ett och $\epsilon > 0$. Produkten blir då $|(\lambda+\epsilon)(\lambda+2\epsilon)\cdots(\lambda+(n-1)\epsilon)|$. Produkten står i nämnaren, så ett genom produkten kan begränsas av $1/\lambda^{n-1}$. Detta blir ett litet tal och r är ett välkonditionerat nollställe.
Ibland bjuds man dock på överraskningar. Antag att vi har nollställena $r_k = k, k = 1, \ldots, n$ med $n = 20$. Gapen mellan angränsande nollställen är då ett, vilket verkar vara betryggande, alla nollställen borde vara välkonditionerade. Låt oss nu studera hur känsligt nollstället $r_n = n$ är för störningar i $a_n = 1$. Vår uppskattning ger:

$$\left|\frac{\delta_{r_n}}{r_n}\right| \approx \left|\frac{a_n r_n^{n-1}}{a_n \prod_{s\neq n}(r_n - r_s)}\right| \left|\frac{\delta_{a_n}}{a_n}\right| = \frac{n^{n-1}}{(n-1)!}\left|\frac{\delta_{a_n}}{a_n}\right| \approx 4.3 \cdot 10^7 \left|\frac{\delta_{a_n}}{a_n}\right|$$

Några av de andra nollställena är ännu känsligare.

12: Får Du göra själv, se dock diskussionen i slutet av föregående övning.

13: Vi kan klara av alla tre fallen på en gång genom att låta p, nedan, anta värdena -10, 0 respektive 10. Det exakta värdet blir: $6.278 \cdot 10^p + 4.039 \cdot 10^p = 1.0317 \cdot 10^{p+1}$ och det lagras, korrekt avrundat och i normaliserad form, som $1.032 \cdot 10^{p+1}$. Det absoluta felet blir: $1.032 \cdot 10^{p+1} - 1.0317 \cdot 10^{p+1} = 3 \cdot 10^{p-3}$. Det relativa felet blir $3 \cdot 10^{p-3}/1.0317 \cdot 10^{p+1} \approx 3 \cdot 10^{-4}$. Notera att de tre absoluta felen blir: $3 \cdot 10^{-13}$, $3 \cdot 10^{-3}$ respektive $3 \cdot 10^7$. Det relativa felet blir, i alla tre fallen, $\approx 3 \cdot 10^{-4}$.

14: Vi vet att $fl(a+b) = (a+b)(1+\epsilon)$, $|\epsilon| \leq \epsilon_{mach}$. Alltså gäller att $fl(a+b) = \tilde{a} + \tilde{b}$ där $\tilde{a} = a(1+\epsilon)$, $|\tilde{a} - a| \leq |a|\epsilon_{mach}$, analogt för b. $fl(a+b)$ är således den exakta summan av två något störda tal.

15: Eftersom matrisaddition utförs elementvis kan vi tillämpa resultatet från föregående övning på varje summa. Således gäller att $fl(\mathbf{A}+\mathbf{B}) = \tilde{\mathbf{A}} + \tilde{\mathbf{B}}$ där $\tilde{\mathbf{A}}$ ligger nära $\mathbf{A}$ och $\tilde{\mathbf{B}}$ ligger nära $\mathbf{B}$.

16: Låt oss betrakta specialfallet då $n = 4$. För att få mindre oläsliga formler inför vi $\sigma_k = 1 + \delta_k$, $\pi_k = 1 + \epsilon_k$ (σ för summa och π för produkt) där alla $|\epsilon_k|$ och $|\delta_k|$ är mindre än ϵ_{mach}. $fl(x_1y_1) = x_1y_1\pi_1$.
$fl(x_1y_1 + x_2y_2) = [x_1y_1\pi_1 + x_2y_2\pi_2]\sigma_2$, vi använder inte σ_1
$fl(x_1y_1 + x_2y_2 + x_3y_3) = ([x_1y_1\pi_1 + x_2y_2\pi_2]\sigma_2 + x_3y_3\pi_3)\sigma_3$.
$fl(x_1y_1 + x_2y_2 + x_3y_3 + x_4y_4) = [([x_1y_1\pi_1 + x_2y_2\pi_2]\sigma_2 + x_3y_3\pi_3)\sigma_3 + x_4y_4\pi_4]\sigma_4$.
Allmänt gäller tydligen, om vi inför $\sigma_{j:k} = \sigma_j\sigma_{j+1}\cdots\sigma_k$, att:

$$fl(x_1y_1 + x_2y_2 + \ldots + x_ny_n) = x_1y_1\pi_1\sigma_{2:n} + x_2y_2\pi_2\sigma_{2:n} + x_3y_3\pi_3\sigma_{3:n} + \ldots + x_ky_k\pi_k\sigma_{k:n} + \ldots + x_ny_n\pi_n\sigma_n$$

Med våra antaganden om flyttalsaritmetik kan vi skriva detta

$$fl\left(\sum_{k=1}^{n} x_ky_k\right) = x_1y_1(1+n\epsilon_1') + \sum_{k=2}^{n} x_ky_k(1+(n-k+2)\epsilon_k')$$

där alla $|\epsilon_k'| \leq \epsilon_{mach}$. Om vi t.ex. sätter $\tilde{x}_k = x_k\sqrt{1+(n-k+2)\epsilon_k'}$ och $\tilde{y}_k = y_k\sqrt{1+(n-k+2)\epsilon_k'}$ (analogt för x_1 och y_1) så gäller tydligen att:

$$fl\left(\sum_{k=1}^{n} x_ky_k\right) = \sum_{k=1}^{n} \tilde{x}_k\tilde{y}_k$$

Om då $n\epsilon_{mach}$ är **tillräckligt litet** så är den beräknade skalärprodukten en exakt skalärprodukt av näraliggande värden och algoritmen är stabil.
Ovanstående analys är onödigt pessimistisk. Det visar sig nämligen att många FPUer (Floating Point Unit) kan beräkna `sum = sum + x(k) * y(k)` med endast ett avrundningsfel (en s.k. FMA, floating multiply add; även kallad MADD). Det kan också vara så att FPUn internt har flera bitar (än 64) i flyttalsregistren. Här följer ett experiment. Jag har skrivit ett innerproduktsprogram, i enkel precision, i Fortran90. Vektorerna innehåller 100000 slumptal valda så att innerprodukten skall bli noll. Koden har kompilerats med olika optimeringsoptioner, allt på en IBM-dator. Först tvingade jag kompilatorn att generera kod som följer IEEE. Det står i manualbladet för kompilatorn att:

```
-qfloat=<suboption1>[:<suboption2>[:...:<suboptionN>]]
 Specifies various floating-point suboptions. Suboptions include:

 hssngl  - Rounds single-precision expressions only when the results are
           stored into REAL(4) memory locations.

 rndsngl - Ensures strict adherence to IEEE standard.  That is, all operations on
           single-precision values produce results that remain in single precision.
```

Därefter kompilerade jag utan optioner och i det sista fallet slog jag på aggressiv optimering. Jag fick följande resultat i de tre fallen:

```
Innerprodukten =  3.85E-02
Innerprodukten =  1.89E-04
Innerprodukten = -1.07E-14
```

I tredje fallet räknas allt i dubbel precision (minst).

17: Största talet i båda fallen är $9.999 \cdot 10^{10}$. Minsta normaliserade talet är $1.000 \cdot 10^{-10}$ och det minsta denormaliserade talet är $0.001 \cdot 10^{-10}$. Denormaliserade flyttal används bara (i IEEE-standarden) kring nollan och inte kring största/minsta tal. Det är därför det största talet i b)-uppgiften är samma som i a)-uppgiften. De största talet är i båda deluppgifterna normaliserat.

18: $100^{20} + 100^{15}$ pga utskiftning, så `1.000000000100000e+40`.

19: a) 2365.27 normaliserat blir $2.36527 \cdot 10^3$ med $t = 6$ och $U = 3$.
$0.0000512 = 5.12 \cdot 10^{-5}$ så $L = -5$.
b) Med $t = 6$ får vi $0.0000512 = 0.00512 \cdot 10^{-2}$ så $L = -2$.

20: Två logaritmberäkningar torde ta mer tid än en logaritm och en division.
Problem med kancellation då $x \approx y$. Sätt $x = (1+\delta)y$, där $|\delta|$ är litet och antag att $fl(\log z) = (1+\epsilon_1)\log z$ med $|\epsilon_1| \leq \epsilon_{mach}$ (dvs. logaritmfunktionen returnerar ett korrekt avrundat värde). Via Taylorutveckling kan vi skaffa oss en uppfattning om det exakta värdet:

$$\log x - \log y = \log(1+\delta)y - \log y = \delta - \delta^2/2 + \cdots$$

Nu till felen:

$$fl(\log x - \log y) = ((1+\epsilon_1)\log x - (1+\epsilon_2)\log y)(1+\epsilon_3) =$$

$$\underbrace{\log x - \log y}_{exakt} + \underbrace{\epsilon_3(\log x - \log y)}_{\approx \delta\epsilon_3} + \epsilon_1 \log x - \epsilon_2 \log y + \underbrace{\epsilon_3(\epsilon_1 \log x - \epsilon_2 \log y)}_{\approx \epsilon_{mach}^2}$$

Så

$$|fl(\log x - \log y) - (\log x - \log y)| \lesssim (|\delta| + 2|\log x|)\epsilon_{mach}$$

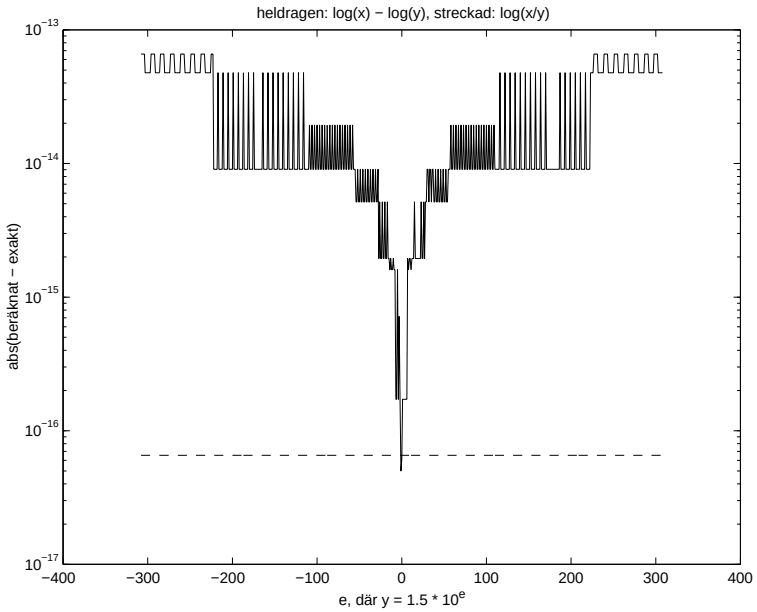
$|\log x|$ kan bli stor (men inte väldigt stor).
Motsvarande för $\log x/y$:

$$fl(\log x/y) = (1+\epsilon_2)\log[(x/y)(1+\epsilon_1)] = (1+\epsilon_2)(\log x/y + \log(1+\epsilon_1)) = \log x/y + \log(1+\epsilon_1) + \epsilon_2 \log x/y + \epsilon_2 \log(1+\epsilon_1)$$

så

$$|fl(\log x/y) - (\log x/y)| \lesssim (1 + |\delta| + \epsilon_{mach})\epsilon_{mach}$$

(ty $\log(1+\epsilon_1) = \epsilon_1 - \epsilon_1^2/2 + \cdots \approx \epsilon_1$). Metod två ger, med andra ord, bättre resultat utom då $x \approx 1$ (när $2|\log x| < 1$). Detta faktum illustreras i nedanstående plot (gjort i Fortran och för $\delta = 10^{-5}$).



21: Kancellation om $x \approx y$.

$$fl(x^2 - y^2) = (x^2(1+\epsilon_1) - y^2(1+\epsilon_2))(1+\epsilon_3) = x^2(1+\epsilon_1+\epsilon_3+\epsilon_1\epsilon_3) - y^2(1+\epsilon_2+\epsilon_3+\epsilon_2\epsilon_3)$$

Så

$$\left|fl(x^2-y^2) - (x^2-y^2)\right| \leq (|x^2+y^2|(1+\epsilon_{mach}) + |x^2-y^2|)\epsilon_{mach}$$

vilket ger ett stort relativt fel om $x \approx y$, ty

$$\frac{\left|fl(x^2-y^2) - (x^2-y^2)\right|}{|x^2-y^2|} \leq \left[\frac{x^2+y^2}{|x^2-y^2|}(1+\epsilon_{mach}) + 1\right]\epsilon_{mach}$$

Alternativ formel:

$$fl((x-y)(x+y)) = (x-y)(1+\epsilon_1)(x+y)(1+\epsilon_2)(1+\epsilon_3) = (x^2-y^2)(1+\epsilon_1)(1+\epsilon_2)(1+\epsilon_3)$$

så

$$\left|fl((x-y)(x+y)) - (x^2-y^2)\right| \lesssim 3\epsilon_{mach}|x^2-y^2|$$

vilket alltid ger ett litet relativt fel, ty

$$\frac{\left|fl((x-y)(x+y)) - (x^2-y^2)\right|}{|x^2-y^2|} \lesssim 3\epsilon_{mach}$$

22: Låt oss kalla den stora (negativa) roten R och den lilla, nära noll, r. Standardformeln och Taylorutveckling ger:

$$R = -\frac{a}{2} - \sqrt{\frac{a^2}{4} - b} = -\frac{a}{2}\left[1 + \sqrt{1 - \frac{4b}{a^2}}\right] = -\frac{a}{2}\left[2 - \frac{2b}{a^2} - \frac{2b^2}{a^4} - \cdots\right] \approx -a$$

$$r = -\frac{a}{2} + \sqrt{\frac{a^2}{4} - b} = \frac{a}{2}\left[-1 + \sqrt{1 - \frac{4b}{a^2}}\right] = \frac{a}{2}\left[-\frac{2b}{a^2} - \frac{2b^2}{a^4} - \cdots\right] \approx -\frac{b}{a}$$

Här följer ett litet Matlabexempel:

```
>> format long e
>> a = 1e4;   b = 1;
>> roots([1 a b])
ans =
    -9.999999899999999e+03
    -1.000000010000000e-04
```

Vi kan uppskatta konditionstalen enligt:

$$\kappa_R \approx \frac{|a| + |b/R|}{|R - r|} \approx \frac{a + b/a}{a} \approx 1$$

och

$$\kappa_r \approx \frac{|a| + |b/r|}{|R - r|} \approx \frac{a + b/(b/a)}{a} \approx 2$$

När vi beräknar $r = -\frac{a}{2} + \sqrt{\frac{a^2}{4} - b}$ kommer att få utskiftning av b. I det mest extrema fallet kommer inte b alls med och approximationen blir noll. I vårt exempel får vi approximationen `-1.000000011117663e-04`. Hur skall vi beräkna r? Ett sätt är att använda utvecklingen ovan:

$$r = -\frac{b}{a} - \frac{b^2}{a^3} - 2\frac{b^3}{a^5} \cdots$$

Hur många termer skall vi ta med? Det är enklare att analysera om vi skriver om utvecklingen som:

$$r = -\frac{b}{a}\left[1 + \frac{b}{a^2} + 2\frac{b^2}{a^4} \cdots\right]$$

Det är ingen vits att ta med termer, inom hakparenteserna, som är mindre än ϵ_{mach} eftersom de skiftas ut. I exemplet ovan får vi approximationen:

```
>> r = -(b / a) * (1 + b / a^2)
r = -1.000000010000000e-04
```

Den exakta roten är `-0.000100000001000000020000000500000...`.
Ett standardtrick är att förlänga med konjugatet, sålunda:

$$r = \frac{\left[-\frac{a}{2} + \sqrt{\left(\frac{a}{2}\right)^2 - b}\right]\left[-\frac{a}{2} - \sqrt{\left(\frac{a}{2}\right)^2 - b}\right]}{-\frac{a}{2} - \sqrt{\left(\frac{a}{2}\right)^2 - b}} = \frac{b}{-\frac{a}{2} - \sqrt{\left(\frac{a}{2}\right)^2 - b}}$$

Vi får utskiftning av b i nämnaren, men b kommer med i täljaren. Ytterligare ett sätt, är att göra en transformation så att r blir en dominant rot i det transformerade problemet. Sätt $y = 1/x$ (så att $r \to 1/r$). Ekvationen $x^2 + ax + b = 0$ övergår då till $y^2 + (a/b)y + 1/b = 0$. Om vi använder standardformeln får vi för den sökta roten:

$$\frac{1}{r} = -\frac{a}{2b} - \sqrt{\frac{a^2}{4b^2} - \frac{1}{b}}$$

23: Problemet är att normen kan vara representerbar men att delresultat ger under- eller overflow. I dubbel precision är det minsta normaliserade positiva talet $\approx 2.2 \cdot 10^{-308}$ och det största $\approx 1.8 \cdot 10^{308}$. Den naiva implementationen `nrm = sqrt(sum(x.^2))` fungerar inte bra, t.ex.:

```
>> x = [1e300 2e300]; sqrt(sum(x.^2))
ans =
   Inf
>> norm(x)     % Matlabs normkommando
ans =
  2.2361e+300

>> x = [1e-300 2e-300]; sqrt(sum(x.^2)), norm(x)
ans =
     0
ans =
  2.2361e-300
```

Så här ser, en förenklad och förkortad version av, Fortran77-rutinen `dnrm2` från BLAS ut (jag har skrivit rutinen i ett pseudo-språk). BLAS = Basic Linear Algebra Subprograms, en uppsättning standardrutiner för enkla vektor- och matrisoperationer. (I ett riktigt program räknar man inte om $|x_k|$ utan man lagrar absolutbeloppet i en variabel.)

if $n == 1$ **then**
  $norm = |x_1|$
else
  $\sigma = 0$
  $s = 1$
  for $k = 1$ **to** n **do**
    if $x_k \neq 0$ **then**
      if $\sigma < |x_k|$ **then**
        $s = 1 + s(\sigma/x_k)^2$
        $\sigma = |x_k|$
      else
        $s = s + (x_k/\sigma)^2$
      end if
    end if
  end for
end if
$norm = \sigma\sqrt{s}$

Låt oss analysera ovanstående algoritm. σ är en skalfaktor som ser till att vi slipper under- overflow. σ kommer (pga if-satsen) att vara det hittills största $|x_k|$-värdet. Låt oss se hur det fungerar i ett specialfall, då $n = 5$. När $k = 1$ blir $s = 1$ och $\sigma = |x_1|$. Antag nu att både $|x_2|$ och $|x_3|$ är $\leq |x_1|$ men att $|x_4| \geq |x_1|$. Efter tredje loopen ($k = 3$) har s värdet $s = 1+(x_2/x_1)^2+(x_3/x_4)^2$. Efter nästa varv är $s = 1+[1+(x_2/x_1)^2+(x_3/x_1)^2](x_1/x_4)^2 = 1 + (x_1/x_4)^2 + (x_2/x_4)^2 + (x_3/x_4)^2$. Om $|x_5| \leq |x_4|$ får vi slutligen

$$norm = |x_4|\sqrt{1 + (x_1/x_4)^2 + (x_2/x_4)^2 + (x_3/x_4)^2 + (x_5/x_4)^2} = \sqrt{x_4^2 + x_1^2 + x_2^2 + x_3^2 + x_5^2}$$

dvs. rätt uttryck. Vi kan notera att $|\sigma/x_k| < 1$ i **then**-delen och att $|x_k/\sigma| \leq 1$ i **else**-delen. Vi bildar inga farliga kvoter med andra ord. Detta innebär givetvis inte att vi slipper utskiftning. Normen av $[1, 10^{-10}]^T$ kommer att bli ett oavsett hur vi räknar, ty det exakta värdet på normen är 1.00000000000000000000499999999999999999999875000 ... och den första decimalen skild från noll kommer inte med om vi räknar med 16 siffror. Däremot klarar algoritmen de två exemplen i början.
Man kan kritisera algoritmen på två punkter. Det görs fler operationer än i den vanliga algoritmen (roten ur kvadratsumman) så vi kan förvänta oss flera avrundningsfel (inte nödvändigtvis större). Ett sätt att få ner felen är att skala med tvåpotenser (det svarar ju mot att flytta binärpunkten, vilket är en exakt operation). Detta är nu lite knöligt att göra på ett portabelt sätt, men om man skriver program för en speciell dator så går det bra.

Ett värre problem är att ovanstående program går väldigt långsamt på en modern RISC-dator (en faktor 10-30 långsammare än motsvarande naiva implementation (beroende på dator och kompilator)). Alla if-satserna förstör pipeliningen i CPUn. Ett sätt att komma förbi detta är att utnyttja att IEEE-standarden föreskriver att man skall kunna avläsa statusflaggor som visar om en beräkning har orsakat under- eller overflow. Man kan då alltid chansa och utföra den vanliga, snabba beräkningen (som torde gå bra i de flesta fall). Om statusflaggorna säger att den beräkningen inte gick bra använder man den långsamma men säkra algoritmen.

24: Lite teori. Låt oss studera trunkeringsfelet. Dvs. vad är skillnaden mellan gränsvärdet 1 och $(e^x - 1)/x$ för $x \approx 0$. Taylorutveckling ger:

$$\frac{e^x - 1}{x} = \frac{1 + x + x^2/2 + x^3/3! + \cdots - 1}{x} = 1 + \frac{x}{2} + \cdots$$

Så trunkeringsfelet är ungefär $x/2$ för små $|x|$. Nu till avrundningsfelet. Vi antar att $fl(e^x) = e^x(1+\epsilon)$ med $|\epsilon| \leq \epsilon_{mach}$. Då gäller

$$fl\left[\frac{e^x - 1}{x}\right] = \frac{e^x(1+\epsilon_1) - 1}{x}(1+\epsilon_2)(1+\epsilon_3) = \left[\frac{e^x - 1}{x} + \epsilon_1\frac{e^x}{x}\right](1+\epsilon_2)(1+\epsilon_3)$$

så (då $x \to 0$)

$$\left| fl\left[\frac{e^x - 1}{x}\right] - \frac{e^x - 1}{x}\right| \lesssim \left[2 + \frac{1}{|x|}\right]\epsilon_{mach}$$

Notera $|x|$ i nämnaren! Följande tabell visar approximationen, felet och uppskattningen (diskretiseringsfel+avrundningsfel).

```
>> x = 10.^-(1:16)';
>> limit = (exp(x) - 1) ./ x;
>> sprintf('%22.16e %8.1e %8.1e\n', [limit, limit - 1, x / 2 + (2 + 1 ./ x) * eps]')
ans =
1.0517091807564771e+00  5.2e-02  5.0e-02
1.0050167084167949e+00  5.0e-03  5.0e-03
1.0005001667083846e+00  5.0e-04  5.0e-04
1.0000500016671410e+00  5.0e-05  5.0e-05
1.0000050000069649e+00  5.0e-06  5.0e-06
1.0000004999621837e+00  5.0e-07  5.0e-07
1.0000000494336803e+00  4.9e-08  5.2e-08
9.9999999392252903e-01 -6.1e-09  2.7e-08
1.0000000827403710e+00  8.3e-08  2.2e-07
1.0000000827403710e+00  8.3e-08  2.2e-06
1.0000000827403710e+00  8.3e-08  2.2e-05
1.0000889005823410e+00  8.9e-05  2.2e-04
9.9920072216264089e-01 -8.0e-04  2.2e-03
9.9920072216264089e-01 -8.0e-04  2.2e-02
1.1102230246251565e+00  1.1e-01  2.2e-01
0.0000000000000000e+00 -1.0e+00  2.2e+00
```

Om man vill beräkna $e^x - 1$ på ett bra sätt kan man utnyttja C-rutinen `expm1` t.ex. Här är ett förkortat utdrag ur manualsidan:

```
Mathematical Library                                    expm1(3M)

NAME: expm1 - computes exponential functions
```

```
SYNOPSIS
     cc [ flag ... ] file ...  -lm [ library ... ]
     #include <math.h>
     double expm1(double x);

DESCRIPTION: The expm1() function computes e**x-1.0.

USAGE: The value of expm1(x) may be more accurate  than  exp(x)-1.0 for small
  values of x. The expm1() and log1p(3M) functions are useful for financial
  calculations of (((1+x)**n)-1)/x namely:

          expm1(n * log1p(x))/x

  when x is very small (for example, when performing calculations  with a small daily
  interest rate). These functions also simplify writin accurate inverse hyperbolic functions.
```

25: Diskretiseringsfelet erhålles via Taylorutveckling:

$$\frac{f(x+h)-f(x)}{h} = \frac{f(x)+hf'(x)+h^2f''(x)/2+\cdots-f(x)}{h} = f'(x)+\frac{h}{2}f''(x)+\cdots$$

När vi uppskattar avrundningsfelet antar vi (för att förenkla) att $x+h$ beräknas exakt (se dock nedan). Dessutom antar vi att $fl(f(x)) = f(x)(1+\epsilon_k), |\epsilon_k| \leq \epsilon_{mach}$ (vilket kan vara orealistiskt om f är en komplicerad funktion).

$$fl\left[\frac{f(x+h)-f(x)}{h}\right] = \frac{f(x+h)(1+\epsilon_1)-f(x)(1+\epsilon_2)}{h}(1+\epsilon_3)(1+\epsilon_4) =$$

$$\frac{f(x+h)(1+\epsilon_1)(1+\epsilon_3)(1+\epsilon_4)-f(x)(1+\epsilon_2)(1+\epsilon_3)(1+\epsilon_4)}{h} =$$

$$\frac{f(x+h)(1+\epsilon_5)-f(x)(1+\epsilon_6)}{h} = \frac{f(x+h)-f(x)}{h}+\frac{f(x+h)\epsilon_5-f(x)\epsilon_6}{h}$$

Antar vi dessutom att $f(x) \approx f(x+h)$ får vi uppskattningen:

$$\left|fl\left[\frac{f(x+h)-f(x)}{h}\right]-\frac{f(x+h)-f(x)}{h}\right| \lesssim \left|\frac{f(x)}{h}\right| 6\epsilon_{mach}$$

Det totala felet får vi om vi adderar de två felen:

$$\left|fl\left[\frac{f(x+h)-f(x)}{h}\right]-f'(x)\right| = \left|fl\left[\frac{f(x+h)-f(x)}{h}\right]-\frac{f(x+h)-f(x)}{h}+\left[\frac{f(x+h)-f(x)}{h}-f'(x)\right]\right| \leq$$

$$\left|fl\left[\frac{f(x+h)-f(x)}{h}\right]-\frac{f(x+h)-f(x)}{h}\right|+\left|\frac{f(x+h)-f(x)}{h}-f'(x)\right| \lesssim \left|\frac{f(x)}{h}\right| 6\epsilon_{mach}+\left|\frac{h}{2}f''(x)\right|$$

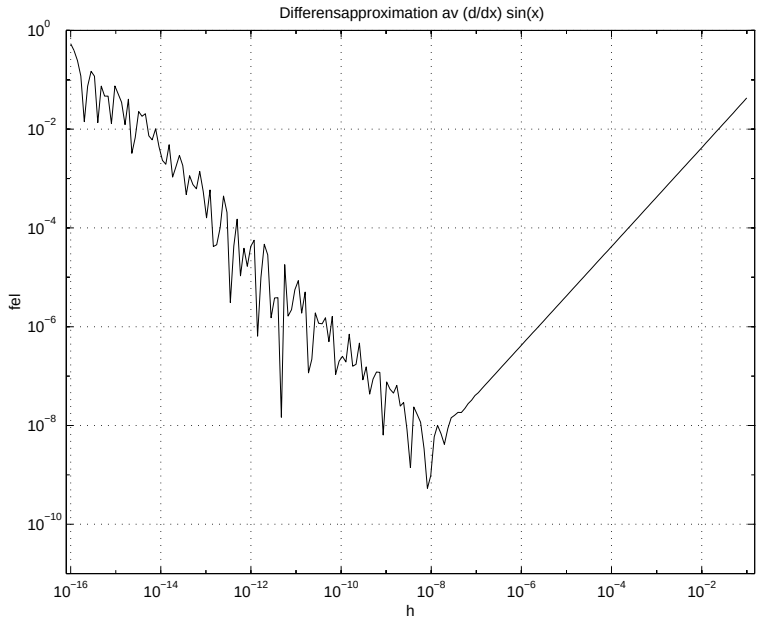
Tar vi $|h|$ för litet dominerar avrundningsfelet och om vi tar ett för stort $|h|$ dominerar diskretiseringsfelet. Genom att derivera begränsningen av felet med avseende på h får vi det optimala värdet:

$$h_{opt} = \sqrt{12\epsilon_{mach}\left|\frac{f(x)}{f''(x)}\right|}$$

Hur stort blir det totala felet, e_T, med h_{opt}?

$$e_T = \left|\frac{f(x)}{h_{opt}}\right| 6\epsilon_{mach}+\left|\frac{h_{opt}}{2}f''(x)\right| = \cdots = \sqrt{12\,|f(x)f''(x)|}\,\sqrt{\epsilon_{mach}}$$

Här följer ett exempel ($f(x) = \sin x, x = 1$, $h_{opt} \approx 5\cdot 10^{-8}$ vilket stämmer mycket bra).



Vi hade antagit att $x+h$ beräknas exakt. Om detta inte är fallet är det risk att vår feluppskattning stämmer dåligt. Man kan lätt skapa ett problem där vi får utskiftning i $x+h$. Rimligtvis borde h_{opt} skalas med x. Det borde göra stor skillnad om $x = 10^{-10}$, 1 resp. 10^{10}. Här följer en rimligare härledning där vi tar hänsyn till beräkningsfelet i $x+h$. Får att kunna återanvända det vi redan gjort inför vi ϵ_5 och antar att $fl(x+h) = (x+h)(1+\epsilon_5)$. Taylorutveckling ger $f((x+h)(1+\epsilon_5)) = f((x+h)+\epsilon_5(x+h)) \approx f(x+h)+\epsilon_5(x+h)f'(x+h) \approx f(x)+\epsilon_5(x+h)f'(x)$. Vi kan även approximera $x+h \approx x$ eftersom h-termen ändå försvinner när vi deriverar. Vi får

$$fl\left[\frac{f(x+h)-f(x)}{h}\right] = \frac{(f(x)+\epsilon_5 x f'(x))(1+\epsilon_1)-f(x)(1+\epsilon_2)}{h}(1+\epsilon_3)(1+\epsilon_4)$$

$$\left|fl\left[\frac{f(x+h)-f(x)}{h}\right] - \frac{f(x+h)-f(x)}{h}\right| \lesssim \left|\frac{f(x)}{h}\right| 6\epsilon_{mach} + \left|\frac{xf'(x)}{h}\right| \epsilon_{mach}$$

Detta ger följande uppskattning av h_{opt}

$$h_{opt} = \sqrt{2\epsilon_{mach}\frac{6|f(x)|+|xf'(x)|}{|f''(x)|}}$$

Denna uppskattning verkar rimligare eftersom h_{opt} skalas med x. I vårt exempel ger detta dock en liten förändring av h_{opt}. Det totala felet blir

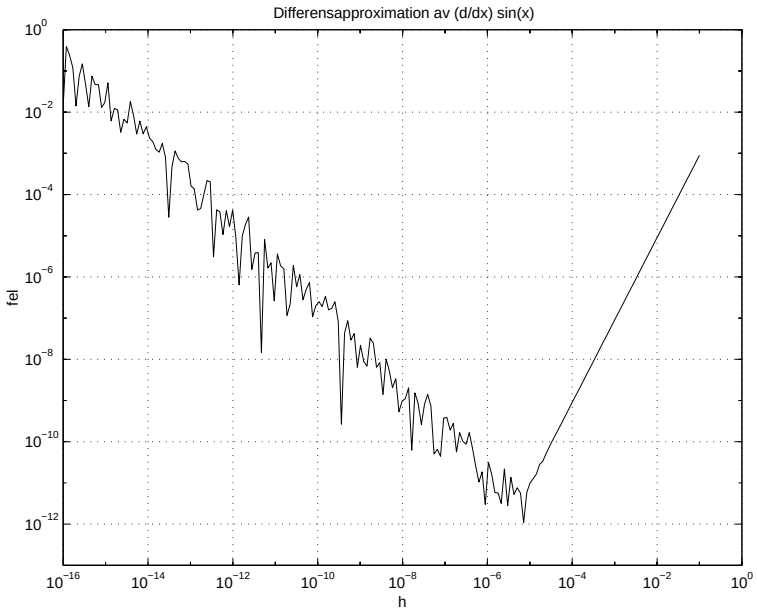
$$e_T = \sqrt{(6|f(x)|+|xf'(x)|)\ |f''(x)|/2}\ \sqrt{\epsilon_{mach}}$$

b) Trunkeringsfelet blir $h^2 f'''(x)/6 + \cdots$ och avrundningsfelet som i a-uppgiften (om vi antar att multiplikation med 2 görs exakt). Så det totala felet och h_{opt} blir ungefär

$$\frac{h^2}{6}|f'''(x)| + \left|\frac{f(x)}{h}\right| 6\epsilon_{mach}, \quad h_{opt} = \left[18\epsilon_{mach}\left|\frac{f(x)}{f'''(x)}\right|\right]^{1/3}$$

Här följer ett exempel ($f(x) = \sin x, x = 1$, $h_{opt} \approx 1.4 \cdot 10^{-5}$ vilket stämmer mycket bra). Totala felet böir mindre i detta fall:

$$e_T = \frac{h_{opt}^2}{6}|f'''(x)| + \left|\frac{f(x)}{h_{opt}}\right| 6\epsilon_{mach} \approx 2.7\left|f^2(x)f'''(x)\right|^{1/3}\ \epsilon_{mach}^{2/3}$$



Vi kan även i denna uppgift ta hänsyn till felet i beräkningen av $x + h$.

26: Den harmoniska serien är divergent, men om vi räknar med ändligt många siffror får vi utskiftning och för något N kommer det att gälla att:

$$fl\left[\sum_{n=1}^{N}\frac{1}{n}\right] = fl\left[\sum_{n=1}^{N+1}\frac{1}{n}\right]$$

Låt oss **uppskatta** N. Vi vet att

$$\sum_{n=1}^{N}\frac{1}{n} \approx \log N + \gamma$$

där $\gamma = 0.5772156649015328606065 1\ldots$ (Eulers konstant) Vi får utskiftning då

$$(1/N)/(\log N + \gamma) \approx \epsilon_{mach}$$

Dvs. i enkel precision då $N \approx 2.1 \cdot 10^6$ och i dubbel då $N \approx 1.2 \cdot 10^{14}$. Division är ett mycket långsam operation (22 klockcykler på en Sun), så att testa detta tar enormt mycket tid i dubbel precision (tusentals timmar) men i enkel är det möjligt (tar mindre än en sekund). Här följer ett Fortran90-program:

```
program harmonisk
  real    :: summa, gammal_summa = 0.0
  integer :: n = 1

  do
    summa = gammal_summa + 1.0 / n
    if ( gammal_summa == summa ) exit
    gammal_summa = summa
    n = n + 1
  end do

  print*, n, summa
end program harmonisk
```

```
Körning:
% a.out
 2097152  15.4036827
```

Det stämmer rätt bra med andra ord.

27: Den andra formeln kräver att vi går igenom x-vektorn en gång, men den första formeln kräver två pass, vilken ur prestandasynpunkt är en nackdel. Ur felsynpunkt är den första formeln dock bättre, eftersom subtraktionen av medelvärdet har en utjämnande verkan. Vi kan få stor skillnad mellan värdena genom att sett till att det blir under- eller overflow i $\sum_{i=1}^{n} x_i^2$-beräkningen. T.ex.

```
>> x  = [2e154 2e153];
>> n  = length(x);
>> mx = sum(x) / n;
>> [sum((x - mx).^2) sum(x.^2) - n * mx^2]
ans =
  1.6200e+308          NaN

>> x = [0.1 ones(1, 50)] * 1e-161; % ger
ans =
  7.9051e-323 -9.8813e-323   % OBS: -9.8813e-323 < 0
```