APC 523 - Pset #1

Cristian Lacey
Mar. 12

① W.L.O.G. consider $x > 0$ s.t. $x = \left(\sum_{n=1}^{\infty} b_n 2^{-n}\right) \cdot 2^e$

$\min \|x\| = 2^{-1} 2^e$

consider $b_{p+1} = 0$:

$$x - rd(x) = \sum_{n=1}^{\infty} b_n 2^{-n} \cdot 2^e - \sum_{n=1}^{p} b_n 2^{-n} \cdot 2^e$$

$$= \sum_{n=p+1}^{\infty} b_n 2^{-n} \cdot 2^e$$

$$\max \|x - rd(x)\| = \sum_{n=p+2}^{\infty} 2^{-n} \cdot 2^e = 2^{-p-2}\left(\frac{1}{1-\frac{1}{2}}\right) \cdot 2^e$$

$$= 2^{-1} \cdot 2^{-p} \cdot 2^e$$

$$\therefore \frac{\max \|x - rd(x)\|}{\min \|x\|} = \frac{2^{-1} \cdot 2^{-p} \cdot 2^e}{2^{-1} \cdot 2^e} = 2^{-p}$$

consider $b_{p+1} = 1$:

(round last digit up)

$$x - rd(x) = \sum_{n=1}^{\infty} b_n 2^{-n} \cdot 2^e - \left(\sum_{n=1}^{p} b_n 2^{-n} + 2^{-p}\right) \cdot 2^e$$

$$= \sum_{n=p+1}^{\infty} b_n 2^{-n} \cdot 2^e - 2^{-p} \cdot 2^e$$

$$\max \|x - rd(x)\| = 2^{-p-1} \cdot 2^e - 2^{-p} \cdot 2^e$$

$$\therefore \frac{\max \|x - rd(x)\|}{\min \|x\|} = \left| \frac{(2^{-p-1} - 2^{-p}) \cancel{2^e}}{2^{-1} \cdot \cancel{2^e}} \right|$$

$$= |2^{-p} - 2 \cdot 2^{-p}| = |-2^{-p}|$$

$$= 2^{-p}$$

$$\boxed{\therefore \left| \frac{x - rd(x)}{x} \right| \le 2^{-p}}$$

for worst case of rounding down and rounding up.

(2)

a) See part a) of code.

NOTE: ☆ %g kills trailing zeros in output, but everything is rounded to 5 sig. digs!

b) Converges to 244.71 for k = 18 (see code).

Comparison and rel. error to math.exp() in code.

c) Yes, summing right to left introduced less error, since we add the smallest values first, so that by the time we are adding the larger values, the sum of all the previous small values is of a similar order as the subsequent large values, and we don't lose information (as much) due to limited precision.

(Again, see code)

d) i)
ii)
iii)
iv)
(See code)

Quickest convergence: d) iii)

Lowest Error: d) ii)

Error is worse for $x < 0$ compared to $x > 0$.

e) Perform $e^{5.5}$, then take the reciprocal.

(See code) You get 0 error when rounding to 5 sig. digs.

③

a)

i) if x is a machine number, then:

$fl(x) = x$

and $fl(x \cdot x) = x^2(1+\epsilon_{x^2})$ where $\epsilon_{x^2} \le eps$

$fl(fl(x^2)\cdot x) = x^3(1+\epsilon_{x^2})(1+\epsilon_{x^3})$

$= x^3(1+\epsilon_{x^2}+\epsilon_{x^3})$

. . .

$fl(fl(x^{n-1})\cdot x) = x^n(1+\epsilon_{x^2}+\dots+\epsilon_{x^n})$

where $\epsilon_{x^2}, \dots, \epsilon_{x^n}$ are all bounded by eps

$$\boxed{\therefore \left|\epsilon_{x^2}+\dots+\epsilon_{x^n}\right| \le (n-1)\,eps}$$

if x is not a machine number, then:

$fl(x) = x(1+\epsilon_x)$ where $|\epsilon_x| \le eps$

so $fl(fl(x)\cdot fl(x)) = fl(x^2(1+2\epsilon_x)) = x^2(1+2\epsilon_x)(1+\epsilon_{x^2})$

$= x^2(1+2\epsilon_x+\epsilon_{x^2})$

$fl(fl(fl(x)\cdot fl(x))\cdot fl(x)) = x^2(1+2\epsilon_x+\epsilon_{x^2})\, x\,(1+\epsilon_x)(1+\epsilon_{x^3})$

$= x^3(1+3\epsilon_x+\epsilon_{x^2}+\epsilon_{x^3})$

so for x^n:

$= x^n(1+n\epsilon_x+\epsilon_{x^2}+\dots+\epsilon_{x^n})$

where $\epsilon_x, \epsilon_{x^2}, \dots, \epsilon_{x^n}$ all bounded by eps

$$\boxed{\therefore \left|n\epsilon_x+\epsilon_{x^2}+\dots+\epsilon_{x^n}\right| \le (2n-1)\,eps}$$

ii) $fl(\ln(x)) = \ln x(1+\epsilon_{ln})$ where $|\epsilon_{ln}| \leq eps$

assuming $n\ln x$ calculated through multiplication:

$fl(n) = n$ (assume machine number according to almighty Gabe, via Piazza)

$$\therefore \underbrace{fl(fl(n)fl(\ln x))}_{y} = n\ln x(1+\epsilon_{ln})(1+\epsilon_{mult})$$

$$= n\ln x(1+\epsilon_{ln}+\epsilon_{mult})$$

$$fl(\exp(y)) = e^{n\ln x(1+\epsilon_{ln}+\epsilon_{mult})}(1+\epsilon_{exp})$$

$$= e^{n\ln x}\, e^{n\ln x(\epsilon_{ln}+\epsilon_{mult})}(1+\epsilon_{exp})$$

$$= x^n(1+n\ln x(\epsilon_{ln}+\epsilon_{mult}))(1+\epsilon_{exp})$$

$$= x^n(1+\epsilon_{exp}+n\ln x(\epsilon_{ln}+\epsilon_{mult}))$$

where $|\epsilon_{exp}|, |\epsilon_{ln}|, |\epsilon_{mult}|$ all $\leq eps$

$$\boxed{\therefore |\epsilon_{exp}+n\ln x(\epsilon_{ln}+\epsilon_{mult})| \leq eps(1+2n\ln x)}$$

assume $n\ln x$ calc by adding $\ln x$, n times:

$$fl(fl(\ln x)+fl(\ln x)) = (\ln x(1+\epsilon_{ln}) + \ln x(1+\epsilon_{ln}))(1+\epsilon_1)$$

$$= 2\ln x(1+\epsilon_{ln}+\epsilon_1)$$

repeating sums:

$$n\ln x(1+\epsilon_{ln}+\epsilon_1+\dots+\epsilon_{n-1}) = y$$

$$fl(\exp(y)) = e^{n\ln x(1+\epsilon_{ln}+\epsilon_1+\dots+\epsilon_{n-1})}(1+\epsilon_{exp})$$

by same procedure:

$$= x^n(1+\epsilon_{exp}+n\ln x(\epsilon_{ln}+\epsilon_1+\dots+\epsilon_{n-1}))$$

$$\boxed{\therefore |\epsilon_{exp}+n\ln x(\epsilon_{ln}+\epsilon_1+\dots+\epsilon_{n-1})| \leq eps(1+n^2\ln x)}$$

Since error for repeated multiplication is bounded based on n, and assuming $n \ln x$ is calculated with multiplication, then the log-exponential method is bounded by a function of $\ln x$. That is repeated multiplication would be better for all x except where $x \approx 1$ and $|\ln x| < 1$. Otherwise the $\ln x$ term just increases the error bound for the log-exp method.

b)

i) $$x^{a(1+\epsilon_a)} = x^a x^{a\epsilon_a}$$

$$= x^a e^{a\epsilon_a \ln x}$$

$$= x^a(1 + a\epsilon_a \ln x)$$

$$\therefore fl(x^a) = x^a(1+\epsilon_p)$$

$$\boxed{\epsilon_p = a\epsilon_a \ln x}$$

← Becomes substantial when $x \approx 0$ since $\ln x$ blows up. Also, if x is very large.

ii) $$(x(1+\epsilon_x))^a = x^a(1+\epsilon_x)^a$$

$$= x^a(1 + a\epsilon_x)$$

So if $fl(x^a) = x^a(1+\epsilon_p)$

then $\boxed{\epsilon_p = a\epsilon_x}$

Propagated error ind. of x. But, gets large for large a, which is especially important for $x \approx 1$ where x^a doesn't overflow.

(4) $f(x) = 1 - e^{-x}$

a) $f'(x) = e^{-x}$

$$\epsilon_f = \left(\frac{x f'(x)}{f(x)}\right) \epsilon_x$$

$$\boxed{(\text{cond} f)(x) = \left| \frac{x e^{-x}}{1 - e^{-x}} \right|}$$

$$\lim_{x \to 0} (\text{cond} f)(x) = \lim_{x \to 0} \frac{e^{-x} - x e^{-x}}{e^{-x}} = \lim_{x \to 0} 1 - x = 1$$

and $$(\text{cond} f)'(x) = \frac{(e^{-x} - x e^{-x})(1 - e^{-x}) - (x e^{-x})(e^{-x})}{(1 - e^{-x})^2}$$

$$= \underbrace{\frac{e^{-x}(1 - x - e^{-x} \cancel{+ x e^{-x}} \cancel{- x e^{-x}})}{(1 - e^{-x})^2}}_{< 0 \text{ for } x \in [0, 1]}$$

$\therefore$ $(\text{cond} f)(x)$ decreasing on $[0, 1]$

and ≤ 1 at $x = 0$ so

$$\boxed{(\text{cond} f)(x) \leq 1 \text{ on } [0, 1]}$$

b) $f_A(x) \Rightarrow fl(-x) = -x$ (since x is machine #)

$$fl(e^{-x}) = e^{-x}(1 + \epsilon_{exp})$$

$$fl(1 - fl(e^{-x})) = (1 - e^{-x}(1 + \epsilon_{exp}))(1 + \epsilon_{sub})$$

$$= (1 + \epsilon_{sub}) - e^{-x}(1 + \epsilon_{exp} + \epsilon_{sub})$$

$$= (1 - e^{-x}) \left[\frac{1 + \epsilon_{sub} - e^{-x} - e^{-x}(\epsilon_{exp} + \epsilon_{sub})}{1 - e^{-x}} \right]$$

$$= (1 - e^{-x}) \left(1 + \frac{\epsilon_{sub} - e^{-x}(\epsilon_{exp} + \epsilon_{sub})}{1 - e^{-x}} \right)$$

and from part a), $\epsilon_f = \frac{x f'(x)}{f(x)} \epsilon_a$ where $\epsilon_a = \frac{|x_A - x|}{|x|}$

so ϵ_a is <u>not</u> bounded by eps

where $\epsilon_f = \frac{\epsilon_{sub} - e^{-x}(\epsilon_{exp} + \epsilon_{sub})}{1 - e^{-x}}$

since $f(x_A) = f_A(x)$

perfect algo — actual algo — actual input

$\frac{e^x - 1}{x}$

$\therefore \quad \epsilon_a = \frac{\epsilon_{sub} - e^{-x}(\epsilon_{exp} + \epsilon_{sub})}{x e^{-x}}$

and ϵ_{sub} and ϵ_{exp} both bounded by eps

- RHS expression max when $\epsilon_{sub} = -\epsilon_{exp} = \text{eps}$

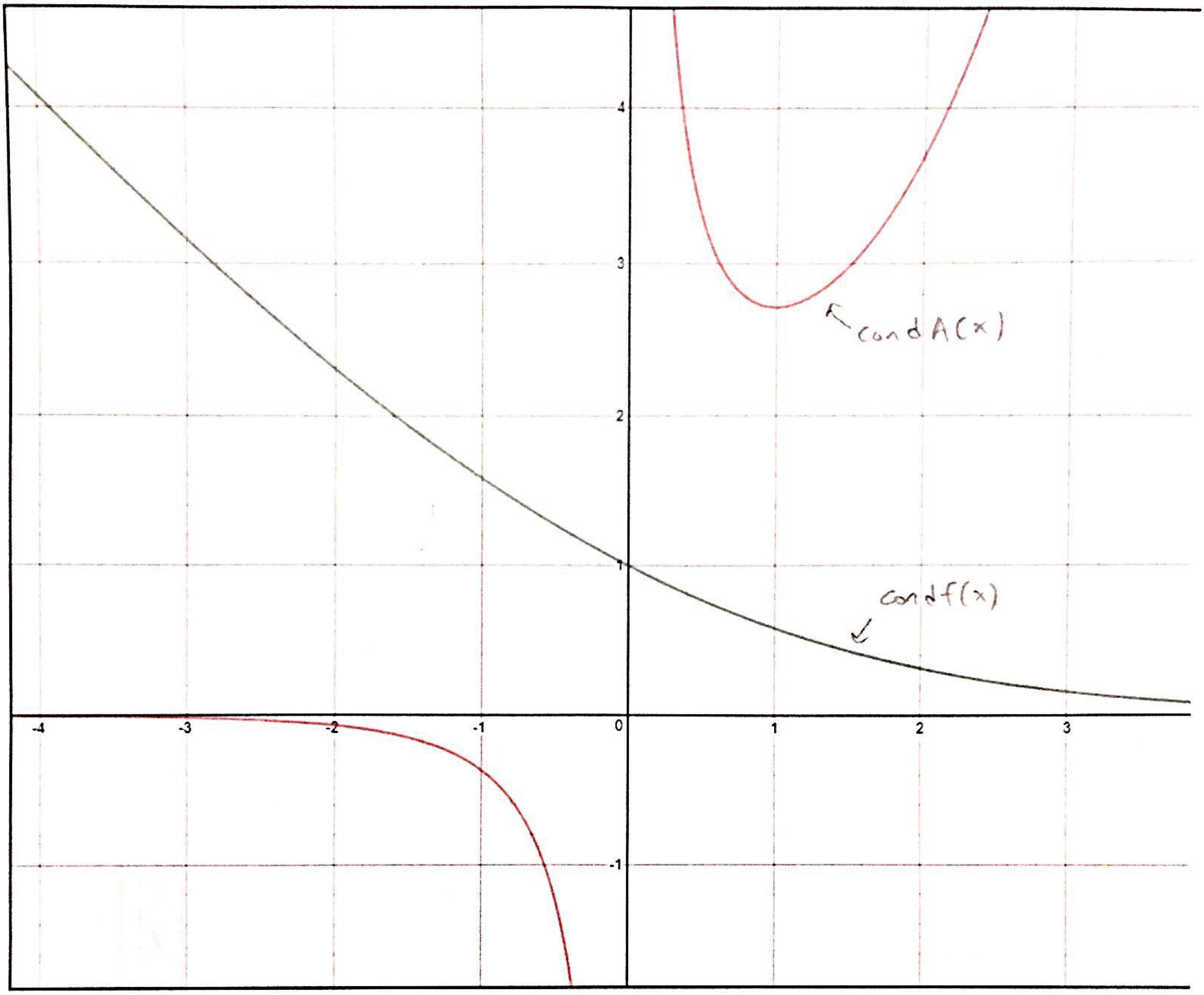
$$\therefore \quad \epsilon_a = \text{eps} \underbrace{\left(\frac{1}{x e^{-x}} \right)}_{\text{cond A}}$$

so $\boxed{\text{cond}A(x) = \frac{e^x}{x}}$

where $\text{cond}A(1) = e > 1$

and $\text{cond}A'(x) = e^x \left(\frac{x-1}{x^2} \right) < 0$ on $[0,1]$

$\boxed{\therefore \quad \text{cond}A(x) > 1 \text{ on } [0,1]}$



1 $\sim \frac{1}{xe^{-x}}$ = cond A (x)

2 $\sim \frac{(xe^{-x})}{1-e^{-x}}$ = cond f(x)

c) See graph attached.

The cause of the poor conditioning is the subtraction in $1-e^{-x}$. As x gets close to zero, e^{-x} approaches 1. As a result, two similar magnitude terms are subtracted, yielding catastrophic!!! error.

d) $2^{-b} \leq 1 - \frac{q}{p} \leq 2^{-a}$

where $p = 1$, $q = e^{-x}$

$\therefore \quad 2^{-b} \leq 1 - e^{-x}$

$2^{-b} = 1 - e^{-x_{min}}$

$e^{-x_{min}} = 1 - 2^{-b}$

$x_{min} = -\ln(1 - 2^{-b})$

For 1 bit loss: $b = 1$

$x_{min} = 0.69315$

2 bits: $b = 2$

$x_{min} = 0.28768$

3 bits: $b = 3$

$x_{min} = 0.13353$

4 bits: $b = 4$

$x_{min} = 0.0645385$

e) $\frac{|y_A - y|}{|y|} \le \text{cond} f(x) \left(\cancel{\epsilon_x}^{\,0} + \text{cond} A(x) \cdot eps \right)$

assuming x is machine #

$$\therefore \frac{|y_A - y|}{|y|} \le \left(\frac{x}{e^x - 1}\right)\left(\frac{e^x}{x}\right) \cdot eps$$

$$\frac{|y_A - y|}{|y|} \le \frac{e^x}{e^x - 1} eps$$

plugging in x_{min}'s from part d) and using $eps = 2^{-53}$ for double precision:

	$\frac{\vert y_A - y \vert}{\vert y \vert}$ max
1 bit	2 eps
2 bit	4 eps
3 bit	8 eps
4 bit	16 eps

f) Try to get rid of subtraction:

$$1 - e^{-x}\left(\frac{1+e^{-x}}{1+e^{-x}}\right) = \frac{1 - (e^{-x})^2}{1 + e^{-x}}$$ where $e^{-x} = \cosh x - \sinh x$

$$= \frac{2\sinh(x) e^{-x}}{1 + e^{-x}} = \boxed{\frac{2\sinh(x)}{e^x + 1}}$$

where $\sinh(x)$ can be represented by a Taylor series with all positive terms (no subtraction).

or, alternatively, take Taylor series for e^{-x}!

$$\cancel{1} - \left(\cancel{1} - x + \frac{x^2}{2} - \dots\right)$$

$$= x\left(1 - \frac{x}{2} + \frac{x^2}{6} - \dots\right)$$

$$= x\left(1 + x\left(\frac{1}{2} - (\dots)\right)\right)$$

where $1 - e^{-x}$ can be approx. by N terms. Subtraction introduced in the above method is no longer between similarly sized quantities (for $x \approx 0$).

⑤ (see code).

Converges to 1. This makes sense, since as n becomes larger and larger, $\frac{1}{n}$ becomes smaller. Eventually, $\frac{1}{n}$ becomes so small that $1 + \frac{1}{n}$ simply evaluates to 1 within the machine precision (that is, the difference in scales of 1 and $\frac{1}{n}$ exceeds the ability of the mantissa to capture them simultaneously). Of course, 1 to any exponent will then simply evaluate to 1, thus the loop converges to 1.

i.e. when $n = 10^{16}$, $e = 1$ b/c:

then $e = (1 + 10^{-16})^{10^{16}}$

but $10^{-16} = 2^x$

$-16 \ln 10 = x \ln 2$

$x = -16 \frac{\ln 10}{\ln 2}$

$x = -53.15$

so $e = (1 + 2^{-53.15})^{10^{16}}$

but $1 + 2^{-53.15}$ is just 1 with 53 bit mantissa, $\therefore e = 1$

(6) If we write $x^{\frac{1}{2^i}}$ as $e^{\frac{1}{2^i}\ln x}$ (see code)

then take the Taylor series:

$$1 + \ln x \frac{1}{2^i} + \ldots$$

we see that for $i = 52$:

$$x^{\frac{1}{2^{52}}} = 1 + \ln x \, 2^{52} + \ldots$$

If $x = e^n$ then

$$x^{\frac{1}{2^{52}}} = 1 + n 2^{-52} + \ldots$$

but in double precision, we only have a 53 bit mantissa, so if $n = 1$, then

$$= 1 + \underbrace{0.0\ldots01}_{52 \text{ dig.}} + \ldots$$

$\therefore$ if n is slightly larger than 1, but less than 2, the information is lost when the first two terms are added, due to limited machine precision. Since the result of the sum is then squared 52 times, a small deviation from 1 is important. y_f is the same for $e \leq x < e^2$ since the same number is squared 52 times. We $\therefore$ observe only integer powers of e as jumps.

Similarly, for $i = 51$:

$$x^{\frac{1}{2^{51}}} = 1 + \ln x \, 2^{-51} + \ldots$$

Since $\ln x$ is only shifted 51 places, the first binary decimal place for $n = \ln x$ will not be lost due to precision limitations. As a result, half integer powers of e are also shown as jumps. Similar arguments can be made for other i values shown.

⑦

a) (see code)

b) Yes, it converges to 20 (about).
The alternative method also converges to 20.

c) The largest root becomes imaginary in each case, but the Newton method root-finder incorrectly locates a nearby real root.
It cannot find imaginary roots.

d) Same issue. Roots 16 and 17 are imaginary, but the Newton method cannot accomodate this.

e) i) $p_i(\Omega_k + \delta\Omega_k) = \sum_{\substack{l=0 \\ l \neq i}}^{n-1} a_l(\Omega_k + \delta\Omega_k)^l + (a_i + \delta a_i)(\Omega_k + \delta\Omega_k)^i + (\Omega_k + \delta\Omega_k)^n$

$$= \sum_{\substack{l=0 \\ l \neq i}}^{n-1} a_l \Omega_k^l \left(1 + \frac{\delta\Omega_k}{\Omega_k}\right)^l + (a_i + \delta a_i)\Omega_k^i \left(1 + \frac{\delta\Omega_k}{\Omega_k}\right)^i + \Omega_k^n \left(1 + \frac{\delta\Omega_k}{\Omega_k}\right)^n$$

$$= \sum_{\substack{l=0 \\ l \neq i}}^{n-1} a_l \Omega_k^l \left(1 + l\frac{\delta\Omega_k}{\Omega_k}\right) + (a_i + \delta a_i)\Omega_k^i \left(1 + i\frac{\delta\Omega_k}{\Omega_k}\right) + \Omega_k^n \left(1 + n\frac{\delta\Omega_k}{\Omega_k}\right)$$

$$= \underbrace{\sum_{\substack{l=0 \\ l \neq i}}^{n-1} a_l \Omega_k^l + a_i \Omega_k^i + \Omega_k^n}_{p(\Omega_k) = 0} + \underbrace{\sum_{\substack{l=0 \\ l \neq i}}^{n-1} a_l l \Omega_k^{l-1} \delta\Omega_k + a_i i \Omega_k^{i-1}\delta\Omega_k + n\Omega_k^{n-1}\delta\Omega_k}_{P'(\Omega_k)\delta\Omega_k}$$

$$+ \delta a_i \Omega_k^i + \underbrace{i\Omega_k^{i-1}\cancel{\delta a_i}\,\delta\Omega_k}_{\to 0}$$

and $p_i(\Omega_k + \delta\Omega_k) = 0$ since $\Omega_k + \delta\Omega_k$ is also a root,

$$\therefore\ 0 = P'(\Omega_k)\delta\Omega_k + \delta a_i \Omega_k^i \Rightarrow \frac{\delta\Omega_k}{\delta a_i} = \frac{-\Omega_k^i}{P'(\Omega_k)}$$

and $T_{ki} = \dfrac{a_i \frac{\partial\Omega_k}{\partial a_i}}{\Omega_k}$

$$\therefore \quad \operatorname{cond} \Omega_k(\underline{a}) = \sum_{l=0}^{n-1} T_{kl}(\underline{a})$$

$$\boxed{\operatorname{cond} \Omega_k(a) = \sum_{l=0}^{n-1} \left| \frac{a_l \, \Omega_k^{\,l-1}}{p'(\Omega_k)} \right|}$$

ii) (See code).

Since the condition numbers are so large for Ω_k's for 14, 16, 17, 20, this means that small perturbations in the coefficients a_l lead to large changes in the resulting roots.

iii) Since the problem itself is ill-conditioned, we can't use a clever algorithm to fix it. There will always be certain cases that behave poorly.

(8)

a) $g_k(y_N) = y_k$

$$g_k(y_N(1+\epsilon_N)) = g_k(y_N + \Delta y_N)$$

$$= g_k(y_N) + \Delta y_N \, g_k'(y_N)$$

$$= y_k + y_N \epsilon_N \, g_k'(y_N)$$

$$= y_k \left(1 + \frac{g_k'(y_N)\, y_N}{y_k} \epsilon_n \right)$$

$$= y_k (1 + \epsilon_k)$$

$$\therefore \text{cond } g_k = \frac{|\epsilon_k|}{|\epsilon_N|}$$

$$\text{cond } g_k = \left| \frac{g_k'(y_N)\, y_N}{y_k} \right|$$

$$y_{n-2} = \frac{e - \left(\frac{e - y_n}{n}\right)}{n-1}$$

$$y_n = \frac{(e - y_{n+1})}{(n+1)}$$

$$\frac{e}{n-1} - \frac{e - y_n}{n(n-1)n-2}$$

reversed recurrence relation:

$$\boxed{y_{n-1} = \left(\frac{e - y_n}{n}\right)}$$

$$\therefore y_k = \frac{k!}{N!}\left(a - (b - (\dots (e - y_N))))\right)$$

so $\left|\frac{dy_k}{dy_n}\right| = \frac{k!}{N!} \Rightarrow \boxed{\text{cond } g_k = \left| \frac{k!\, y_N}{N!\, y_k} \right|}$

b) $$\epsilon_k = \frac{g'_k\, y_N}{y_k}\, \epsilon_N$$

with $\varepsilon = \epsilon_k$ and assuming $\epsilon_N = 1$:

$$\epsilon_k = \frac{k!}{N!}\, \frac{y_N}{y_k}\, \epsilon_n$$

well-conditioned if:

$$\frac{k!}{N!}\, \frac{y_N}{y_k} \leq 1$$

and from the definition of $y_N = \int_0^1 e^x x^n\, dx$

then $y_N < y_k$ for $N > k$

$$\therefore \left| \frac{y_N}{y_k} \right| \leq 1$$

$$\therefore \varepsilon \geq \frac{k!}{N!}$$

$$\boxed{N! \geq \frac{k!}{\varepsilon}}$$

c) For $k = 20$, $\varepsilon = 2^{-53}$

$$\frac{20!}{2^{-53}} = 2.19 \times 10^{34}$$

so, for $\boxed{N = 32}$ $\cdot$ $32! = 2.63 \times 10^{35} > 2.19 \times 10^{34}$

d) (see code and wolfram snippet)

WolframAlpha computational intelligence.

integral from 0 to 1 of x^20 * e^x

Browse Examples Surprise Me

Definite integral:

Fewer digits More digits Step-by-step solution

$$\int_0^1 x^{20} e^x \, dx = 209\,(4\,282\,366\,656\,425\,369\, e - 11\,640\,679\,464\,960\,000) \approx 0.123803830762570$$