APC 523

EINARA ZAHN

1) Error in (symmetric) rounding vs. chopping

Machine number $\mathbb{R}(p,q)$

Symmetric rounding:

- Round down when first discarded bit b_{p+1} is zero
- Round up when b_{p+1} is 1

$$x = \sum_{i=1}^{\infty} b_i 2^{-i} 2^e$$

- If round down:

ex.: $\underbrace{\boxed{1|\cdots|0|1|1|0}}_{p}\overset{p+1\ \rightarrow}{\boxed{0|1|0}}$

$$x = rd(x) + Err \Rightarrow x = \sum_{i=1}^{p} b_i 2^{-i} 2^e + \sum_{i=p+2}^{\infty} b_i 2^{-i} 2^e$$

Absolute error: $|x - rd(x)| = \left| \sum_{i=p+2}^{\infty} b_i 2^{-i} 2^e \right|$

The maximum value: consider all b_i's equal to 1.

$$\max|x - rd(x)| = \left| \sum_{i=p+2}^{\infty} 2^{-i} 2^e \right| = \left| \left(2^{-p-2} + 2^{-p-3} + \cdots\right) 2^e \right| = 2^{-p-1}\left(2^{-1} + 2^{-2} + \cdots\right) 2^e$$

$$\max|x - rd(x)| = 2^{-p-1} S_n 2^e, \quad \text{with } S_n = \frac{a_1}{1-r} = \frac{1/2}{1-1/2} = 1$$

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

The relative error: $\max \dfrac{|x - rd(x)|}{\min|x|}$, with $\min|x| = 2^{-1} 2^e$

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

• If round up: $\underbrace{\boxed{1}\boxed{0}\boxed{\cdots}\boxed{0}\boxed{1}\boxed{1}}_{p}\overset{p+1}{\boxed{1}}\boxed{0}$

b_{p+1} is $1 \to$ add one in $p \Rightarrow 2^{-p}2^{e}$

$$x = rd(x) + \text{Error}$$

$$x = \sum_{i=1}^{p} b_i 2^{-i} 2^{e} \underbrace{- 2^{-p} 2^{e}}_{\substack{\text{added}\\ \text{value due}\\ \text{to rounding}}} + \underbrace{\sum_{i=p+1}^{\infty} b_i 2^{-i} 2^{e}}_{\text{Error}}$$

The absolute error:

$$\left| x - rd(x) \right| = \left| \sum_{i=p+1}^{\infty} b_i 2^{-i} 2^{e} - 2^{-p} 2^{e} \right| = \left| 2^{-p-1} 2^{e} + \sum_{i=p+2}^{\infty} b_i 2^{-i} 2^{e} - 2^{-p} 2^{e} \right| ,$$

since the first digit of the residual is 1.

$$\left| \left(\frac{2^{-p}}{2} + \sum_{i=p+2}^{\infty} b_i 2^{-i} - 2^{-p} \right) 2^{e} \right| = \left| \left(-\frac{2^{-p}}{2} + \underbrace{\sum_{i=p+2}^{\infty} b_i 2^{-i}}_{0 \leq \sum_{p+2} b_i 2^{-i} \leq 1} \right) 2^{e} \right|$$

To get the maximum value, we can set the second term to zero. We get:

$$\left| \frac{x - rd(x)}{x} \right| = \frac{2^{-p-1} 2^{e}}{2^{-1} 2^{e}} = 2^{-p}$$

Therefore, in both cases of rounding (up or down) the upper bound for the error is 2^{-p}.

(2)

(a) Approximate $e^{5.5}$ by working out the terms in this series up to $n = 30$.

Numerator	Denominator
1.00000000000E+00	1.00000000000E+00
5.50000000000E+00	1.00000000000E+00
3.02500000000E+01	2.00000000000E+00
1.66380000000E+02	6.00000000000E+00
9.15090000000E+02	2.40000000000E+01
5.03300000000E+03	1.20000000000E+02
2.76820000000E+04	7.20000000000E+02
1.52250000000E+05	5.04000000000E+03
8.37380000000E+05	4.03200000000E+04
4.60560000000E+06	3.62880000000E+05
2.53310000000E+07	3.62880000000E+06
1.39320000000E+08	3.99170000000E+07
7.66260000000E+08	4.79000000000E+08
4.21440000000E+09	6.22700000000E+09
2.31790000000E+10	8.71780000000E+10
1.27480000000E+11	1.30770000000E+12
7.01140000000E+11	2.09230000000E+13
3.85630000000E+12	3.55690000000E+14
2.12100000000E+13	6.40240000000E+15
1.16660000000E+14	1.21650000000E+17
6.41630000000E+14	2.43300000000E+18
3.52900000000E+15	5.10930000000E+19
1.94100000000E+16	1.12400000000E+21
1.06760000000E+17	2.58520000000E+22
5.87180000000E+17	6.20450000000E+23
3.22950000000E+18	1.55110000000E+25
1.77620000000E+19	4.03290000000E+26
9.76910000000E+19	1.08890000000E+28
5.37300000000E+20	3.04890000000E+29
2.95510000000E+21	8.84180000000E+30
1.62530000000E+22	2.65250000000E+32

(b) $S_K = \sum_{n=0}^{K} \frac{x^n}{n!}$

For $K = 17$, we converged to 244.71

K	S_K
15	244.67
16	244.70
17	244.71
18	244.71

Relative error: $\left| \frac{e^{5.5} - S_{17}}{e^{5.5}} \right| \approx 0.007\%$

(c) Repeat part (b), now add from right to left.

If we add the sum backwards, we start adding small numbers first. Just when we add bigger numbers it starting converging. With this method, the value converged to 244.71 in the last iteration.

(d) Approximate $e^{-5.5}$

(i) Always add left to right

The sum converged to 0.0038363 in $K = 25$

relative error: $\left| \dfrac{e^{-5.5} - S_{25}}{e^{-5.5}} \right| \approx 6.13\%$

(ii) Within each partial sum, add right to left

The sum converged to 0.004 at $K = 19$.

error: $\left| \dfrac{e^{-5.5} - S_{19}}{e^{-5.5}} \right| \approx 2.2\%$

(iii) positive: left to right
Negative: left to right

The sum converged to 0.00000 at $K=17$

error: $\left| \dfrac{e^{-5.5} - 0.0}{e^{-5.5}} \right| = 100\%$

(iv) positive: right to left
negative: right to left

The sum converged to 0.01000 at $K = 18$

error: $\left| \dfrac{e^{-5.5} - 0.01}{e^{-5.5}} \right| \approx 60\%$

The method (iii) converged faster; however, it was the method with the larger error (100%).

(e) In order to avoid the subtractions in the Taylor Series, $e^{5.5}$ should be calculated as $\frac{1}{e^{5.5}}$. First find the value for $e^{5.5}$ (which has smaller error). The division of 1 for this value also has smaller error than subtraction. Implementing the method, we get a value equal $\frac{1}{244.71} = 0.0040865$, error $\approx 0.036\%$.

(3) Error propagation in exponentiation

(a) x^n

(i) Repeatedly multiplying by x (Assume x is a perfect machine number)

x^2: $fl(x \cdot x) = xx(1+\epsilon_{xx}) = xx(1+\epsilon)$

x^3: $fl((x\cdot x)\cdot x) = xxx(1+\epsilon)(1+\epsilon)$
$= xxx(1+2\epsilon+\epsilon^2)$
$= xxx(1+2\epsilon)$

x^4: $fl((x\cdot x\cdot x)\cdot x) = xxxx(1+2\epsilon)(1+\epsilon)$
$= xxxx(1+\epsilon+2\epsilon+2\epsilon^2)$
$= xxxx(1+3\epsilon)$

x^5: $fl((x\cdot x\cdot x\cdot x)\cdot x) = xxxxx(1+3\epsilon)(1+\epsilon)$
$= xxxxx(1+\epsilon+3\epsilon+3\epsilon^2)$
$= xxxxx(1+4\epsilon)$

Therefore, the pattern shows that the error of repetatedly multiply by x is $\epsilon_{x^n} = (n-1)|\epsilon|$

Upper bound: $\epsilon_{x^n} = \mathcal{E}(n-1)$

$1+\epsilon_2+\epsilon_1+\epsilon_1\epsilon_2$

(ii) $e^{n\ln x}$ Assume x has no rounding error

$A = fl(\ln x) = \ln x(1+\epsilon_1)$

$B = fl(n\cdot A) = \; = n\ln x(1+\epsilon_1)(1+\epsilon_2)$
$= n\ln x(1+\epsilon_2+\epsilon_1+\epsilon_1\epsilon_2)$
$= n\ln x(1+\epsilon_2+\epsilon_1) = n\ln x + n\epsilon_2\ln x + n\epsilon_1\ln x$

$C = fl(\exp(B)) = \exp(n\ln x)\exp(n\epsilon_2\ln x)\exp(n\epsilon_1\ln x)(1+\epsilon_3)$
$= \exp(n\ln x)(1+n\epsilon_2\ln x)(1+n\epsilon_1\ln x)(1+\epsilon_3)$
$= \exp(n\ln x)(1+n\epsilon_1\ln x + n\epsilon_2\ln x + n^2\epsilon_2\epsilon_1\ln x)(1+\epsilon_3)$
$= \exp(n\ln x)(1+n\epsilon_1\ln x+n\epsilon_2\ln x)(1+\epsilon_3)$
$= \exp(n\ln x)(1+\epsilon_3+n\epsilon_1\ln x+n\epsilon_2\ln x+O(\epsilon^2))$
$= \exp(n\ln x)(1+\epsilon_3+n\epsilon_1\ln x+n\epsilon_2\ln x)$

Upper bound: $|\epsilon_1|=|\epsilon_2|=|\epsilon_3|=\mathcal{E}$ $\quad \epsilon_{e^{n\ln x}} = \mathcal{E}(1+2n\ln x)$

- We need to find when ϵ_{x^n} is smaller than $\epsilon_{e^{n\ln x}}$

$\mathcal{E}(n-1) \leq \mathcal{E}(1+2n\ln x)$
$n-1 \leq 1+2n\ln x$
$n-2 \leq 2n\ln x$
$\ln x \geq \frac{n-2}{2n}$ $\rightarrow$ $x \geq e^{\frac{n-2}{2n}}$

(b) Arbitrary exponents

x is positive and a is nonzero.

(i) x is an exact machine number but a is subject to a relative error ε_a:

$fl(x) = x$ no error of rounding

$fl(a) = a(1+\varepsilon_a)$

Assuming a perfect algorithm:

$$fl(x^a) = x^{a(1+\varepsilon_a)} = x^a x^{a\varepsilon_a}$$
$$= x^a(1 + a\varepsilon_a \log x + o(\varepsilon^2))$$
$$= x^a(1 + a\varepsilon_a \log x)$$

$$x^a = \sum_{n=0} \frac{a^n \log^n(x)}{n!}$$

Upper bound: $\varepsilon_a = |\varepsilon|$

$$= x^a(1 + a\,\varepsilon \log x)$$

The error could become substantial if both a and $\log x$ are very big.

(ii) a is an exact machine number, but not x

$$fl(x^a) = [x(1+\epsilon_x)]^a = x^a(1+\epsilon_x)^a$$
$$= x^a(1 + a\epsilon_x + \tfrac{1}{2}(a-1)a\epsilon^2 \cdots)$$
$$= x^a(1+a\epsilon_x)$$

Upper bound: $fl(x^a) = x^a(1+a\varepsilon)$

In this case, the higher the value of a, the higher would be the error.

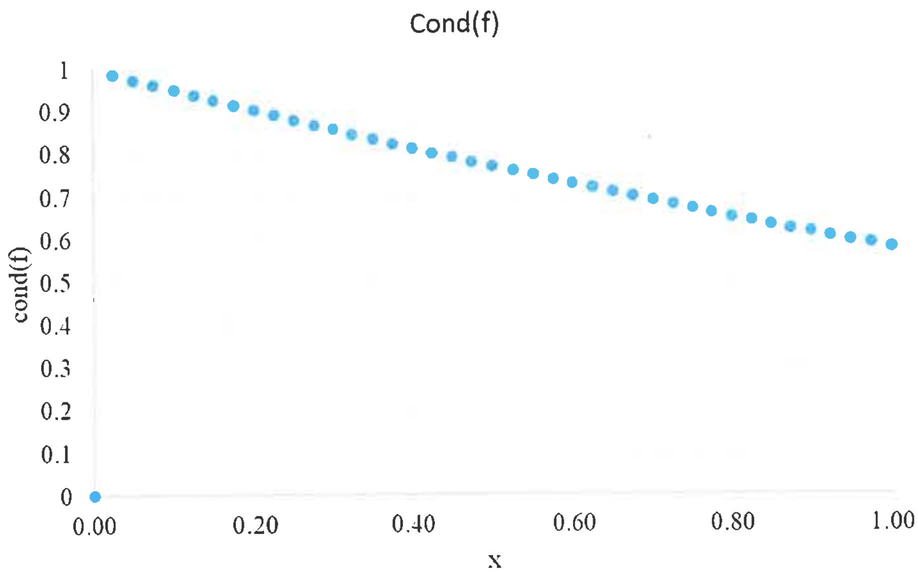
(4) Conditioning

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

(a) Determine $(\text{cond}\, f)(x)$ in terms of x

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

The graph is shown below, where it can be seen that $(\text{cond} f)(x) < 1$ on the interval $]0, 1]$.



(b) Find $(\text{cond} A)(x)$

x is a machine number

$$fl(-x) = x$$

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

$$fl(1 - fl(\exp(fl(-x)))) = (1 - e^{-x}) \underset{\substack{\uparrow \\ \text{subtraction}}}{(1 + \epsilon_{subtr})} \underset{\substack{\uparrow \\ \text{rounding}}}{(1 + \epsilon_{rd})}$$

we know:

$$\epsilon_{x-y} = \frac{x}{x-y} \underbrace{\cancel{\epsilon_x}}_{=0\ (1 \text{ has no error})} + \frac{y}{x-y} \underbrace{\cancel{\epsilon_y}}_{= \epsilon_{exp}}$$

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

Expanding to 1st order: $= (1-e^{-x})\left(1 + \epsilon_{rd} + \frac{e^{-x}}{1-e^{-x}}\epsilon_{exp}\right)$

we want to compare this with our output $f(x_A) = 1 - e^{x_A}$

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

$$\cancel{1}-e^{-x_A} = \cancel{1}+\epsilon_{rd} - e^{-x} - \epsilon_{rd}e^{-x} + e^{-x}\epsilon_{exp}$$

$$x_A = -\ln(-\epsilon_{rd} + e^{-x} + \epsilon_{rd}e^{-x} - e^{-x}\epsilon_{exp})$$

$$x_A = -\ln(e^{-x}(-e^{-x}\epsilon_{rd} + 1 + \epsilon_{rd} - \epsilon_{exp}))$$

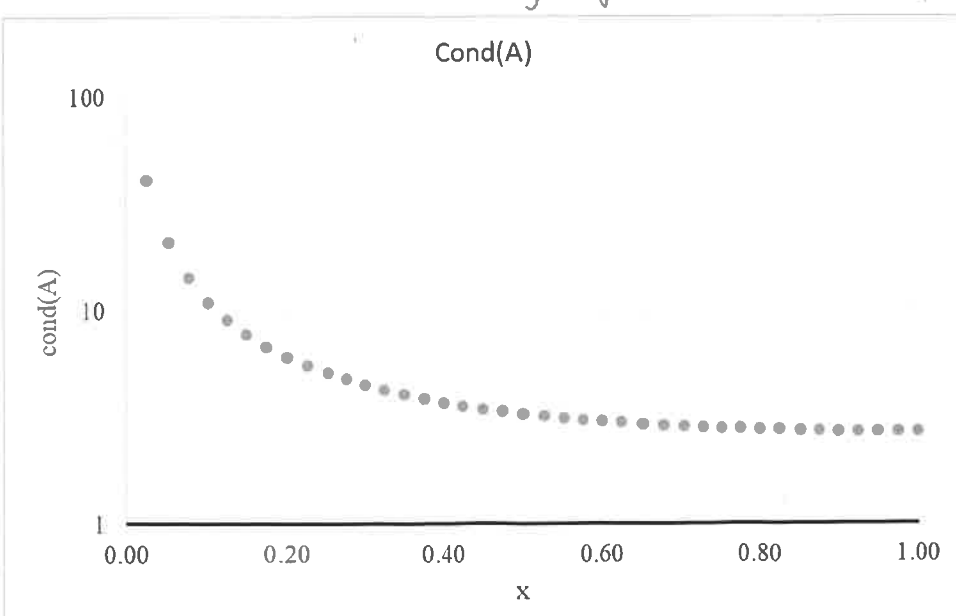
$$x_A = -\ln(e^{-x}) + \ln(1 - e^{x}\epsilon_{rd} + \epsilon_{rd} - \epsilon_{exp})$$ Taylor expansion

$$x_A - x = -(-e^{x}\epsilon_{rd} + \epsilon_{rd} - \epsilon_{exp}) = ((e^{x}-1)\epsilon_{rd} + \epsilon_{exp})$$ $\ln(1+x) \approx x + \dots$

$$\text{Cond}(A) = \frac{\|x_A - x\|}{x\varepsilon}$$ For upper band: assume $|\epsilon_{rd}| = |\epsilon_{exp}| = |\varepsilon|$

$$\text{Cond}(A) = \frac{\|(e^{x}-1)\varepsilon+\varepsilon\|}{x\varepsilon} = \frac{\|e^{x}-1+1\|}{x} = \frac{e^{x}}{x}$$

The graph is shown below (in logscale on y axis), where we can see that the condition number is always greater then 1 in this interval.



(d) $2^{-a} \leq \left| \frac{x-y}{x} \right| < 2^{-b}$

a- at most
b- at least

$$\frac{1-e^{-x}}{1} < 2^{-b} \Rightarrow 1-e^{-x} \geqslant 2^{-a}$$

$$-e^{-x} \geqslant 2^{-a}-1$$

$$e^{-x} \leqslant 1-2^{-a}$$

$$-x \leqslant \ln(1-2^{-b})$$

$$x \geqslant -\ln(1-2^{-b})$$

$b=1$ $\quad x \geqslant 0.693147$

$b=2$ $\quad x \geqslant 0.287682$

$b=3$ $\quad x \geqslant 0.133531$

$b=4$ $\quad x \geqslant 0.06453$

(e)

(f) Use Taylor expansion:

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

$$e^{x} = 1 + x + \frac{x^2}{2}$$

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

→ starting from n=1, the first term already cancel the -1. With this algorithm, no subtraction is done.

5

$$e \equiv \lim_{m \to \infty} \left(1 + \frac{1}{m}\right)^m$$

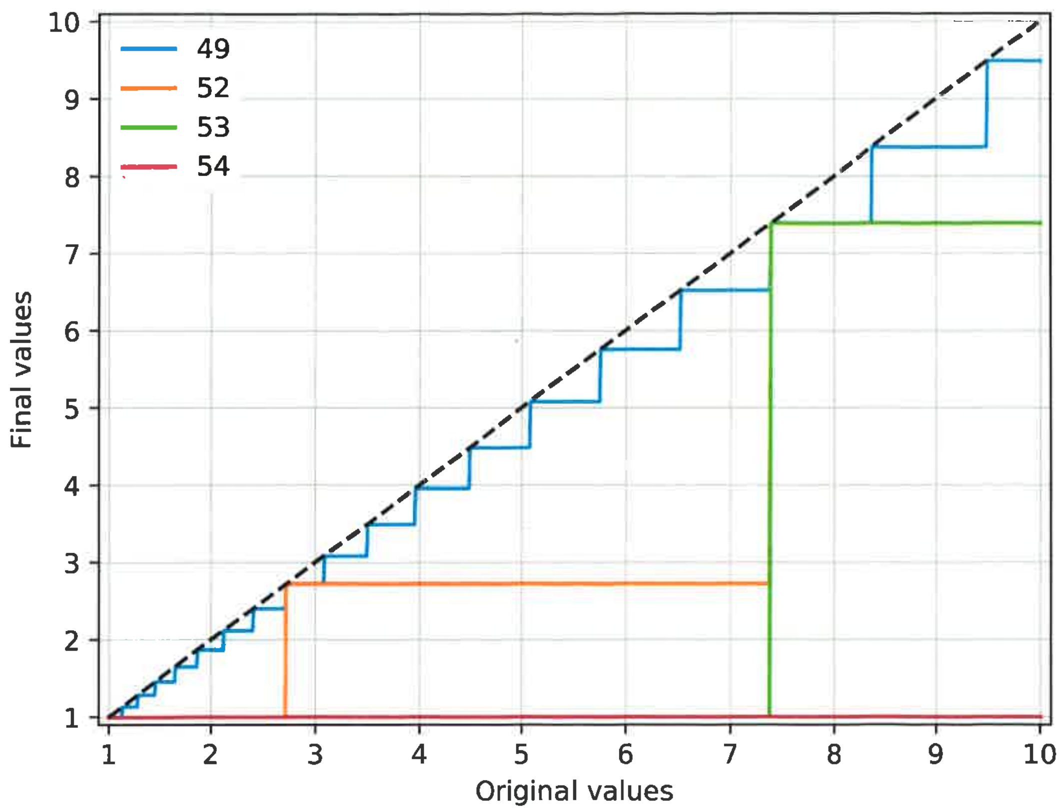
it	n	e
1	1.0E+00	2.000000000000
2	1.0E+01	2.593742460100
3	1.0E+02	2.704813829421
4	1.0E+03	2.716923932235
5	1.0E+04	2.718145926824
6	1.0E+05	2.718268237192
7	1.0E+06	2.718280469095
8	1.0E+07	2.718281694132
9	1.0E+08	2.718281798347
10	1.0E+09	2.718282052011
11	1.0E+10	2.718282053234
12	1.0E+11	2.718282053357
13	1.0E+12	2.718523496037
14	1.0E+13	2.716110034086
15	1.0E+14	2.716110034087
16	1.0E+15	3.035035206549
17	1.0E+16	1.000000000000
18	1.0E+17	1.000000000000

The code only converged to 13 sig-fig (12 decimal places, with no rounding) for $m = 10^{16}$. For this m, the value of e converged to $e = 1.000000000000$.

For $n \geq 10^{16}$, we are summing to 1 a value that is of the same order of the machine epsilon ($1/10^{16} = 10^{-16}$) in double precision. In this sense, we are doing $(1 + \epsilon_x + \epsilon_{ps})^n$, and the result is the same as if we were raising just the first term to the power m. Therefore, this digits are lost when taking the power.

(5) Make an array of 1001 floats from 1 to 10.
- Write a loop to square-root 52 times;
- write another loop to square ~~root~~ 52 times;

The result is show im the plot below:



When we square root a number between 1 to 10 n times, the final value tends to 1. As we square-root, im the binary representation of the computer, we are losing significative digits. In this sense, as we approach 52 (the number of bits for the mantissa), we here just a few digits left. After taking the square-root many times, the final value looks like $(1+d)$, where d is a small value.

If, now, we take the square 52 times, it is the same as $e^x = \lim_{n\to\infty}\left(1+\frac{x}{m}\right)^m$.

Only if $d > 10^{-16}$ the square operation is going to return a value different from 1. If we square root 54, for example, all the significant digits of the mantissa are lost (only zeros are left), and the square can't recover these values. If we take the square root 52 times, some numbers are left with digits left in the mantissa.

7. The issue with polynomial roots

$$w(x) = \prod_{k=1}^{20} (x-k) = (x-1)(x-2)\cdots(x-20)$$

(a) Work out all the (integer) coefficients exactly.

$a_{20} = 1$
$a_{19} = -210$
$a_{18} = 20615$
$a_{17} = -1256850$
$a_{16} = 53327946$
$a_{15} = -1672280820$
$a_{14} = 40171771630$
$a_{13} = -756111184500$
$a_{12} = 11310276995381$
$a_{11} = -135585182899530$
$a_{10} = 1307535010540395$
$a_{9} = -10142299865511450$
$a_{8} = 63030812099294896$
$a_{7} = -311333643161390640$
$a_{6} = 1206647803780373360$
$a_{5} = -3599979517947607200$
$a_{4} = 8037811822645051776$
$a_{3} = -12870931245150988800$
$a_{2} = 13803759753640704000$
$a_{1} = -8752948036761600000$
$a_{0} = 2432902008176640000$

(b) Using the built in function newton from scipy.optimize (python), we get 20.000011760259119. Numpy.roots gives 19.99980929

(c) Change the coefficient a_{20} from 1 to $1+\delta$

δ	Newton (scipy)	roots (numpy)
10^{-8}	9.584921	20.647582 + 1.186926i
10^{-6}	7.752693	23.149016 + 2.74098i
10^{-4}	5.969335	28.400212 + 6.5164i
10^{-2}	5.46959310	38.47818362 + 20.8343i

The two methods give very different results; in general, the largest root gets far from the real root; The smallest roots depart less.

(d) Using numpy.roots:

Root 16: 16.73074488 + 2.8126249i

Root 17: 16.73074488 − 2.8126249i

(e)

$$p(x) = \sum_{k=0}^{m} a_k x^k = a_0 + a_1 x + a_2 x^2 + \dots + a_{n-1}x^{n-1} + x^n$$

$\Omega = \Omega(a_0, a_1, \dots, a_{n-1})$, Ω_k = the kth root of $p(x)$, $k = 1, 2, \dots, n$

$$(\text{cond } \Omega_k)(\vec{a}) \equiv \sum_{l=0}^{m-1} (\gamma_{kl})(\vec{a})$$

(i) Find an expression for $(\text{cond } \Omega_k)(\vec{a})$

$$(\text{cond } \Omega_k)(\vec{a}) = \sum_{i=0}^{\infty} \left| \frac{a_i \frac{\partial \Omega_k}{\partial x_i}}{\Omega_k} \right| = \sum_{l=0}^{m-1} \frac{a_l \Omega_k^{l-1}}{p'(\Omega_k)}$$

A small perturbation in coefficient l brings perturbation to the root l.

$P_{\vec{a}+\delta a}(\Omega_K + \delta\Omega_K) = 0 \Rightarrow$ expand this polinomio

$$0 = a_0 + a_1(\Omega_K + \delta\Omega_K) + a_2(\Omega_K + \delta\Omega_K)^2 + \dots + (a_l + \delta a_l)(\Omega_K + \delta\Omega_K)^l + \dots a_n(\Omega_K + \delta\Omega_K)^m$$

$$0 = a_0 + a_1\Omega_K\left(1 + \frac{\delta\Omega_K}{\Omega_K}\right) + a_2\Omega_K^2\left(1 + \frac{\delta\Omega_K}{\Omega_K}\right)^2 + \dots + a_n\Omega_K^n\left(1 + \frac{\delta\Omega_K}{\Omega_K}\right)^m + (a_l + \delta a_l)\Omega_K^l\left(1 + \frac{\delta\Omega_K}{\Omega_K}\right)^l$$

Expand and keep juster 1st order.

$$0 \neq a_0 + a_1\Omega_K\left(1 + \frac{\delta\Omega_K}{\Omega_K}\right) + a_2\Omega_K^2\left(1 + 2\frac{\delta\Omega_K}{\Omega_K}\right) + \dots + a_n\Omega_K^n\left(1 + n\frac{\delta\Omega_K}{\Omega_K}\right) + (a_l + \delta a_l)\Omega_K^l\left(1 + l\frac{\delta\Omega_K}{\Omega_K}\right)$$

$$0 = \underbrace{a_0 + a_1\Omega_K + a_2\Omega_K^2 + \dots + a_n\Omega_K^n + a_l\Omega_K^l}_{= p(\Omega_K) = 0} + \delta\Omega_K\underbrace{\left(a_1 + 2a_2\Omega_K + \dots n a_n \Omega_K^{m-1}\right)}_{= p'(\Omega_K)}$$

$$+ \delta a_l \Omega_K^l \left(1 + l\frac{\delta\Omega_K}{\Omega_K}\right)^{0} = 0 \text{ (2nd order)}$$

$$0 = \delta\Omega_K p'(\Omega_K) + \delta a_l \Omega_K^l$$

$$\frac{\delta\Omega_K}{\delta a_l} = \frac{-\Omega_K^l}{p'(\Omega_K)} \Rightarrow (\text{cond } \Omega_K)(\vec{a}) = \sum_{l=0}^{m-1} \left| \frac{a_l \frac{\partial \Omega_K}{\partial x_l}}{\Omega_K} \right| = \sum_{l=0}^{m-1} \frac{a_l \Omega_K^{l-1}}{p'(\Omega_K)}$$

(ii) Evaluate condition number

$r = 14, 16, 17, 20$

$$\text{Cond}(A) = \sum_{l=0}^{n-1} \frac{a_l \Omega_k^{l-1}}{p'(\Omega_k)}$$

Root 14 : $\text{cond}(A) = -1332794061.8645$

Root 16: $\text{cond}(A) = -240981121 8.8491$

Root 17: $\text{cond}(A) = 1902076584.7781$

Root 20: $\text{cond}(A) = -43100882.3056$

As can be seen, the absolute value of the condition number for all cases is much bigger than one, highligting that a small perturbation in the coefficient leads to a big error in the root.

(iii) The problem with this polinomium is that when we store the coefficients as floats, we already lose significant digits.
One solution could be to change the basis (for example, in lagrange form).

(8) $y_m \equiv \int_0^1 x^m e^x \, dx \qquad (n \geq 0)$

(a)

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

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

$$y_{m-1} = \frac{e - y_n}{m}$$

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

$$y_{n-3} = \frac{e - y_{n-2}}{m-2} = \frac{1}{m-2}\left(e - \frac{1}{m(n-1)}\left(ne - e + y_n\right)\right)$$

$$= \frac{1}{m(n-1)(n-2)}\left(em(m-1) - ne + e - y_n\right)$$

$$y_{n-4} = \frac{e - y_{n-3}}{m-3} = \frac{1}{m-3}\left(e - \frac{1}{m(m-1)(m-2)}\left(em(m-1) - ne + e - y_n\right)\right)$$

$$= \frac{1}{(m-3)(n-2)(m-1)m}\left(en(n-1)(n-2) - en(m-1) + ne - e + y_n\right)$$

$$\vdots \qquad \qquad \vdots$$

$$y_K = \frac{e - y_{K+1}}{K+1} = \frac{1}{m(m-1)\cdots(m-K+1)}\left(\left(\text{terms of } e \text{ and } m\right) + y_n\right)$$

or minus does not matter!

$$y_K = \frac{1}{(m-K+1)!}\left(\text{constant} + y_n\right) = \frac{K!}{m!}\left(\text{constant} + y_n\right) \qquad g'_K = \frac{K!}{m!}$$

$$\text{Cond}\left(g_K(y_N)\right) = \left|\frac{y_N \, g'(y_K)}{y_K}\right| = \left|\frac{y_N}{y_K}\frac{K!}{m!}\right|$$

upper bound: when $\left|\frac{y_N}{y_K}\right| = 1$

$$\text{cond}\left(g_K(y_N)\right) = \left|\frac{K!}{m!}\right|$$

(b) 100% relative error

$$|\mathcal{E}_K| \leq \operatorname{cond} g_{(K)}(y_N)\, \epsilon_N$$

$$|\mathcal{E}_K| \leq \left|\frac{K!}{m!}\right| \epsilon_N$$ With 100% relative error, we have $\epsilon_N = 1$

$$|\mathcal{E}_K| \leq \left|\frac{K!}{m!}\right|$$

(c) $|\mathcal{E}_K| = 2\cdot 10^{-16}$, $K=20$

$$|2\cdot 10^{-16}| \leq \left|\frac{20!}{m!}\right| \Rightarrow m! \leq \frac{20!}{2\cdot 10^{-16}} \sim O(10^{34})$$

After some iterations, we get $m \geq 32$.

(d) Given answer (c), lets use $m=33$.

start from $m=33$ and $y_{33}=0$.

From Wolfram Alpha:

$$\int_0^1 x^{20} e^x \, dx \simeq 0.123804$$

From code: $y_{20} \approx 0.12380383$

Therefore, we see that the values are very close, and the method is well conditioned. In addition, the initial guess for y_{33} does not matter, and the same result was obtained for any initial value (even for much bigger orders of magnitude).