APC 523/MAE 507/AST 523
Numerical Algorithms for Scientific Computing
Problem Set 1

Jessica Flores

March 12, 2019

1 Error in (symmetric) rounding vs. chopping

Prove

$$\left|\frac{x - \mathrm{rd}(x)}{x}\right| \leq 2^{-p}.$$

Proof. In $\mathbb{R}(p, q)$ let

$$x = \pm\left(\sum_{\ell=1}^{\infty} b_{-\ell} 2^{-\ell}\right) \times 2^e$$

$$\mathrm{rd}(x) = \begin{cases} \pm\left(\sum_{\ell=1}^{p} b_{-\ell} 2^{-\ell}\right) \times 2^e & \text{if } b_{p+1} = 0 \\ \pm\left(\left(\sum_{\ell=1}^{p} b_{-\ell} 2^{-\ell}\right) + 2^{-p}\right) \times 2^e & \text{otherwise} \end{cases}$$

Consider the case in which the first discarded bit b_{p+1} is 0, then

$$\begin{aligned} |x - \mathrm{rd}(x)| &= \left(\sum_{\ell=p+2}^{\infty} b_{-\ell} 2^{-\ell}\right) \times 2^e \\ &\leq \left(\sum_{\ell=p+2}^{\infty} 2^{-\ell}\right) \times 2^e \\ &= (2^{-(p+2)} + 2^{-(p+3)} + 2^{-(p+4)} + \cdots) \times 2^e \\ &= 2^{-(p+2)}(1 + 2^{-1} + 2^{-2} + \cdots) \times 2^e \\ &= 2^{-(p+2)}\left(\frac{1}{1 - 1/2}\right) \times 2^e \\ &= 2^{-(p+1)} \times 2^e \\ &= 2^{e-p-1}. \end{aligned}$$

Thus

$$\frac{|x - \mathrm{rd}(x)|}{|x|} \leq \frac{2^{e-p-1}}{2^{e-1}} = 2^{-p}.$$

Now, if $b_{p+1} = 1$, then

$$\begin{aligned}
|x - \mathrm{rd}(x)| &= -\left(\sum_{\ell=p+1}^{\infty} b_{-\ell}2^{-\ell}\right) \times 2^e + 2^{e-p} \\
&= -\left(2^{-(p+1)} + \sum_{\ell=p+2}^{\infty} b_{-\ell}2^{-\ell}\right) \times 2^e + 2^{e-p} \\
&\leq -\left(2^{-(p+1)}\right) \times 2^e + 2^{e-p} \\
&= -2^{e-p-1} + 2^{e-p} \\
&= 2^{e-p}(1 - 2^{-1}) \\
&= 2^{e-p}(2^{-1}) \\
&= 2^{e-p-1}.
\end{aligned}$$

Thus

$$\frac{|x - \mathrm{rd}(x)|}{|x|} \leq \frac{2^{e-p-1}}{2^{e-1}} = 2^{-p}.$$

$\square$

2 An accurate implementation of e^x

(a) The first 31 terms in order are:

term 1 = 1.0
term 2 = 5.5
term 3 = 15.125
term 4 = 27.73
term 5 = 38.129
term 6 = 41.942
term 7 = 38.447
term 8 = 30.208
term 9 = 20.768
term 10 = 12.692
term 11 = 6.9805
term 12 = 3.4902
term 13 = 1.5997
term 14 = 0.67679
term 15 = 0.26588
term 16 = 0.097484
term 17 = 0.03351
term 18 = 0.010842
term 19 = 0.0033128
term 20 = 0.00095898
term 21 = 0.00026372
term 22 = 6.907e-05
term 23 = 1.7269e-05
term 24 = 4.1297e-06
term 25 = 9.4638e-07
term 26 = 2.0821e-07
term 27 = 4.4043e-08

term 28 = 8.9715e-09
term 29 = 1.7623e-09
term 30 = 3.3422e-10
term 31 = 6.1274e-11

(b) Adding from left to right:

$S_0 = 1.0$
$S_1 = 6.5$
$S_2 = 21.625$
$S_3 = 49.355$
$S_4 = 87.484$
$S_5 = 129.43$
$S_6 = 167.88$
$S_7 = 198.09$
$S_8 = 218.86$
$S_9 = 231.55$
$S_{10} = 238.53$
$S_{11} = 242.02$
$S_{12} = 243.62$
$S_{13} = 244.3$
$S_{14} = 244.57$
$S_{15} = 244.67$
$S_{16} = 244.7$
$S_{17} = 244.71$
$S_{18} = 244.71$
$S_{19} = 244.71$
$S_{20} = 244.71$
$S_{21} = 244.71$
$S_{22} = 244.71$
$S_{23} = 244.71$
$S_{24} = 244.71$
$S_{25} = 244.71$
$S_{26} = 244.71$
$S_{27} = 244.71$
$S_{28} = 244.71$
$S_{29} = 244.71$
$S_{30} = 244.71$

At $k = 17$ the value of $e^{5.5}$ converges to 5 significant figures (244.71).

(c) Adding from right to left:

$S_0 = 1.0$
$S_1 = 6.5$
$S_2 = 21.625$
$S_3 = 49.355$
$S_4 = 87.484$
$S_5 = 129.42$
$S_6 = 167.88$
$S_7 = 198.09$
$S_8 = 218.85$
$S_9 = 231.54$
$S_{10} = 238.51$
$S_{11} = 242.01$
$S_{12} = 243.62$
$S_{13} = 244.28$
$S_{14} = 244.56$
$S_{15} = 244.66$
$S_{16} = 244.69$
$S_{17} = 244.69$
$S_{18} = 244.69$
$S_{19} = 244.69$
$S_{20} = 244.69$
$S_{21} = 244.71$
$S_{22} = 244.71$
$S_{23} = 244.71$
$S_{24} = 244.71$
$S_{25} = 244.71$
$S_{26} = 244.71$
$S_{27} = 244.71$
$S_{28} = 244.71$
$S_{29} = 244.71$
$S_{30} = 244.71$

At $k = 21$ the value of $e^{5.5}$ converges to 5 significant figures (244.71).

The value of $e^{5.5}$ computed directly is 244.69193226422038.

The converged value is the same for both methods (244.71), thus the magnitude of the relative error is the same in both cases:

$$rel\ err = \left| \frac{244.71 - 244.69193226422038}{244.69193226422038} \right| = 7.383870654188337\text{e} - 05$$

(d) For $e^{-5.5}$:

(i) Left to right:	(ii) Right to left:	(iii) Left to right (+,-):	(iv) Right to left (+,-):
$S_0 = 1.0$	$S_0 = 1.0$	$S_0 = 1.0$	$S_0 = 1.0$
$S_1 = -4.5$	$S_1 = -4.5$	$S_1 = -4.5$	$S_1 = -4.5$
$S_2 = 10.625$	$S_2 = 10.625$	$S_2 = 10.625$	$S_2 = 10.625$
$S_3 = -17.105$	$S_3 = -17.105$	$S_3 = -17.105$	$S_3 = -17.105$
$S_4 = 21.024$	$S_4 = 21.024$	$S_4 = 21.024$	$S_4 = 21.024$
$S_5 = -20.918$	$S_5 = -20.918$	$S_5 = -20.918$	$S_5 = -20.918$
$S_6 = 17.529$	$S_6 = 17.529$	$S_6 = 17.529$	$S_6 = 17.529$
$S_7 = -12.679$	$S_7 = -12.679$	$S_7 = -12.679$	$S_7 = -12.679$
$S_8 = 8.089$	$S_8 = 8.089$	$S_8 = 8.09$	$S_8 = 8.09$
$S_9 = -4.603$	$S_9 = -4.603$	$S_9 = -4.6$	$S_9 = -4.6$
$S_{10} = 2.3775$	$S_{10} = 2.377$	$S_{10} = 2.38$	$S_{10} = 2.37$
$S_{11} = -1.1127$	$S_{11} = -1.113$	$S_{11} = -1.11$	$S_{11} = -1.12$
$S_{12} = 0.487$	$S_{12} = 0.487$	$S_{12} = 0.49$	$S_{12} = 0.49$
$S_{13} = -0.18979$	$S_{13} = -0.19$	$S_{13} = -0.19$	$S_{13} = -0.19$
$S_{14} = 0.07609$	$S_{14} = 0.076$	$S_{14} = 0.08$	$S_{14} = 0.07$
$S_{15} = -0.021394$	$S_{15} = -0.021$	$S_{15} = -0.02$	$S_{15} = -0.03$
$S_{16} = 0.012116$	$S_{16} = 0.012$	$S_{16} = 0.01$	$S_{16} = 0.0$
$S_{17} = 0.001274$	$S_{17} = 0.001$	$S_{17} = 0.0$	$S_{17} = -0.01$
$S_{18} = 0.0045868$	$S_{18} = 0.005$	$S_{18} = 0.0$	$S_{18} = 0.01$
$S_{19} = 0.0036278$	$S_{19} = 0.004$	$S_{19} = 0.0$	$S_{19} = 0.01$
$S_{20} = 0.0038915$	$S_{20} = 0.004$	$S_{20} = 0.0$	$S_{20} = 0.01$
$S_{21} = 0.0038224$	$S_{21} = 0.004$	$S_{21} = 0.0$	$S_{21} = 0.01$
$S_{22} = 0.0038397$	$S_{22} = 0.004$	$S_{22} = 0.0$	$S_{22} = 0.01$
$S_{23} = 0.0038356$	$S_{23} = 0.004$	$S_{23} = 0.0$	$S_{23} = 0.01$
$S_{24} = 0.0038365$	$S_{24} = 0.004$	$S_{24} = 0.0$	$S_{24} = 0.01$
$S_{25} = 0.0038363$	$S_{25} = 0.004$	$S_{25} = 0.0$	$S_{25} = 0.01$
$S_{26} = 0.0038363$	$S_{26} = 0.004$	$S_{26} = 0.0$	$S_{26} = 0.01$
$S_{27} = 0.0038363$	$S_{27} = 0.004$	$S_{27} = 0.0$	$S_{27} = 0.01$
$S_{28} = 0.0038363$	$S_{28} = 0.004$	$S_{28} = 0.0$	$S_{28} = 0.01$
$S_{29} = 0.0038363$	$S_{29} = 0.004$	$S_{29} = 0.0$	$S_{29} = 0.01$
$S_{30} = 0.0038363$	$S_{30} = 0.004$	$S_{30} = 0.0$	$S_{30} = 0.01$
$k = 25$	$k = 19$	$k = 17$	$k = 18$
rel err = 6.12883403e − 02	*rel err* = 2.12322709e − 02	*rel err* = 1.00000000e + 00	*rel err* = 1.44691932e + 00

The value of $e^{-5.5}$ computed directly is 0.004086771438464067.

The third method converges the most quickly, but the second method has the lowest error. When the argument of the exponential was positive, adding from left to right converged more quickly, but when the argument of the exponential was negative, adding from left to right was the method that took the longest to converge.

(e) A way to compute $e^{-5.5}$ more accurately is to first compute $e^{5.5}$ and then invert it by performing the operation $\frac{1}{e^{5.5}} = e^{-5.5}$ to avoid subtraction. This method converges to the value 0.0040865, which results in a relative error of 7.38332548e-05, which is much smaller than the relative error of the four methods used above.

3 Error propagation in exponentiation

(a) (i) If x is an exact machine number:

Computing x^2 :

$$\begin{aligned}\mathrm{fl}(x \cdot x) &= (x \cdot x)(1+\epsilon_{multip}) \\ &= x^2(1+\epsilon_{multip})\end{aligned}$$

Computing x^3:

$$\begin{aligned}\mathrm{fl}(\mathrm{fl}(x \cdot x) \cdot x) &= (x^2(1+\epsilon_{multip}) \cdot x)(1+\epsilon_{multip2}) \\ &= x^3(1+\epsilon_{multip})(1+\epsilon_{multip2}) \\ &= x^3(1+\epsilon_{multip}+\epsilon_{multip2})\end{aligned}$$

Therefore, the maximum bound for the relative error is (n-1)eps.

(ii) If x is an exact machine number:

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

$$\begin{aligned}\mathrm{fl}(n \cdot \mathrm{fl}(\ln x)) &= n \ln x(1+\epsilon_{ln})(1+\epsilon_{mult\ by\ n}) \\ &= n \ln x(1+\epsilon_{ln}+\epsilon_{mult\ by\ n})\end{aligned}$$

$$\begin{aligned}\mathrm{fl}(\exp(\mathrm{fl}(n \cdot \mathrm{fl}(\ln x))) &= (\exp(n \ln x(1+\epsilon_{ln}+\epsilon_{mult\ by\ n})))(1+\epsilon_{\exp}) \\ &= (e^{n \ln x} \cdot e^{\epsilon_{ln} n \ln x} \cdot e^{\epsilon_{mult\ by\ n} n \ln x})(1+\epsilon_{\exp}) \\ &= e^{n \ln x}(1+\epsilon_{ln} n \ln x)(1+\epsilon_{mult\ by\ n} n \ln x)(1+\epsilon_{\exp}) \\ &= e^{n \ln x}(1+\epsilon_{ln} n \ln x+\epsilon_{mult\ by\ n} n \ln x)(1+\epsilon_{\exp}) \\ &= e^{n \ln x}(1+\epsilon_{ln} n \ln x+\epsilon_{mult\ by\ n} n \ln x+\epsilon_{\exp}) \\ &\leq e^{n \ln x}(1+n \ln x(\mathrm{eps})+n \ln x(\mathrm{eps})+\mathrm{eps}) \\ &= e^{n \ln x}(1+(2n \ln x+1)\mathrm{eps})\end{aligned}$$

Therefore, the maximum bound for the relative error is ($2n \ln x$+1)eps.

Exponentiating via repeated multiplication is more accurate when:

$$\begin{aligned}n-1 &< 2n \ln x+1 \\ n &< 2(n \ln x+1) \\ \frac{n}{2}-n \ln x &< 1 \\ n\left(\frac{1}{2}-\ln x\right) &< 1\end{aligned}$$

Since n is positive, the above relation is satisfied when $(\frac{1}{2}-\ln x) \leq 0$. Thus, repeated multiplication should be used when $x \geq e^{1/2}$.

(b)(i) If x is an exact machine number and $a = a(1+\epsilon_a)$:

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

$$\begin{aligned}
\text{fl}(\text{fl}(a)\cdot\text{fl}(\ln x)) &= (a(1+\epsilon_a)\ln x(1+\epsilon_{ln}))(1+\epsilon_{mult\ by\ a}) \\
&= (a+a\epsilon_a)(\ln x+\epsilon_{ln}\ln x)(1+\epsilon_{mult\ by\ a}) \\
&= (a\ln x+\epsilon_a a\ln x+\epsilon_{ln}a\ln x)(1+\epsilon_{mult\ by\ a}) \\
&= a\ln x+\epsilon_a a\ln x+\epsilon_{ln}a\ln x+\epsilon_{mult\ by\ a}a\ln x \\
&= a\ln x(1+\epsilon_a+\epsilon_{ln}+\epsilon_{mult\ by\ a})
\end{aligned}$$

$$\begin{aligned}
\text{fl}(\exp(\text{fl}(\text{fl}(a)\cdot\text{fl}(\ln x)))) &= (\exp(a\ln x(1+\epsilon_a+\epsilon_{ln}+\epsilon_{mult\ by\ a})))(1+\epsilon_{\exp}) \\
&= e^{a\ln x(1+\epsilon_a+\epsilon_{ln}+\epsilon_{mult\ by\ a})}(1+\epsilon_{\exp}) \\
&= e^{a\ln x}\cdot e^{a\ln x\epsilon_a}\cdot e^{a\ln x\epsilon_{ln}}\cdot e^{a\ln x\epsilon_{mult\ by\ a}}(1+\epsilon_{\exp}) \\
&= e^{a\ln x}(1+a\ln x\epsilon_a)(1+a\ln x\epsilon_{ln})(1+a\ln x\epsilon_{mult\ by\ a})(1+\epsilon_{\exp}) \\
&= e^{a\ln x}(1+a\ln x\epsilon_a)(1+a\ln x\epsilon_{ln}+a\ln x\epsilon_{mult\ by\ a}+\epsilon_{\exp}) \\
&= e^{a\ln x}(1+a\ln x\epsilon_a)(1+a\ln x\epsilon_{ln})(1+a\ln x\epsilon_{mult\ by\ a})(1+\epsilon_{\exp}) \\
&= e^{a\ln x}(1+a\ln x\epsilon_a+a\ln x\epsilon_{ln}+a\ln x\epsilon_{mult\ by\ a}+\epsilon_{\exp}) \\
&\le e^{a\ln x}(1+a\ln x\ \epsilon_a+2a\ln x\ \text{eps}+\text{eps})
\end{aligned}$$

Therefore, the maximum bound for the relative error is $a\ln x\ \epsilon_a + 2a\ln x\ \text{eps} + \text{eps}$. The propagated error can become substantial if $a\ln x$ is large.

(ii) If a is an exact machine number and $x = x(1+\epsilon_x)$:

$$\begin{aligned}
\text{fl}(\ln\text{fl}(x)) &= (\ln[x(1+\epsilon_x)])(1+\epsilon_{ln}) \\
&= (\ln x+\ln[1+\epsilon_x])(1+\epsilon_{ln}) \\
&= (\ln x+\epsilon_x)(1+\epsilon_{ln}) \\
&= \ln x+\epsilon_x+\ln x\epsilon_{ln} \\
&= \ln x(1+\epsilon_x/\ln x+\epsilon_{ln})
\end{aligned}$$

$$\begin{aligned}
\text{fl}(a\cdot\text{fl}(\ln\text{fl}(x))) &= a\ln x(1+\epsilon_x/\ln x+\epsilon_{ln})(1+\epsilon_{mult\ by\ a}) \\
&= a\ln x(1+\epsilon_x/\ln x+\epsilon_{ln}+\epsilon_{mult\ by\ a})
\end{aligned}$$

$$\begin{aligned}
\text{fl}(\exp(\text{fl}(a\cdot\text{fl}(\ln\text{fl}(x))))) &= \exp(a\ln x(1+\epsilon_x/\ln x+\epsilon_{ln}+\epsilon_{mult\ by\ a}))(1+\epsilon_{\exp}) \\
&= e^{a\ln x(1+\epsilon_x/\ln x+\epsilon_{ln}+\epsilon_{mult\ by\ a})}(1+\epsilon_{\exp}) \\
&= e^{a\ln x}(1+a\epsilon_x)(1+a\ln x\epsilon_{ln})(1+a\ln x\epsilon_{mult\ by\ a})(1+\epsilon_{\exp}) \\
&= e^{a\ln x}(1+a\epsilon_x+a\ln x\epsilon_{ln}+a\ln x\epsilon_{mult\ by\ a}+\epsilon_{\exp}) \\
&\le e^{a\ln x}(1+a\epsilon_x+2a\ln x\ \text{eps}+\text{eps}) \\
&\le e^{a\ln x}(1+a\epsilon_x+2a\ln x\ \text{eps}+\text{eps})
\end{aligned}$$

Therefore, the maximum bound for the relative error is $a\epsilon_x + 2a\ln x\ \text{eps} + \text{eps}$. The propagated error can become substantial if a or $a\ln x$ is large.

4 Conditioning

(a)

$$
\begin{aligned}
f(x) &= 1 - e^{-x} \\
f'(x) &= e^{-x} \\
(\text{cond } f)(x) &= \left| \frac{xf'(x)}{f(x)} \right| = \frac{xe^{-x}}{1-e^{-x}} = \frac{x}{e^x - 1}
\end{aligned}
$$

Since in the interval from [0,1] the condition of f is positive, it is less than or equal to 1 if the reciprocal is greater than 1:

$$
\begin{aligned}
\text{Assuming } \frac{x}{e^x-1} &\leq 1 \\
1 &\leq \frac{e^x - 1}{x} \\
&= \frac{(1 + x + \frac{1}{2}x^2 + \frac{1}{6}x^3 + \cdots) - 1}{x} \\
&= \frac{x(1 + \frac{1}{2}x + \frac{1}{6}x^2 + \cdots)}{x} \\
&= 1 + \frac{1}{2}x + \frac{1}{6}x^2 + \cdots
\end{aligned}
$$

which is always greater than or equal to 1 in [0,1]. In fact, it is true for any positive x.

(b)

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

$$
\begin{aligned}
1 - (\text{fl}(e^{-x})) &= 1 - (e^{-x})(1 + \epsilon_{\text{exp}}) \\
&= 1 - e^{-x} - e^{-x}\epsilon_{\text{exp}} \\
&= (1 - e^{-x}) \left(1 - \frac{e^{-x}}{1 - e^{-x}} \epsilon_{\text{exp}} \right)
\end{aligned}
$$

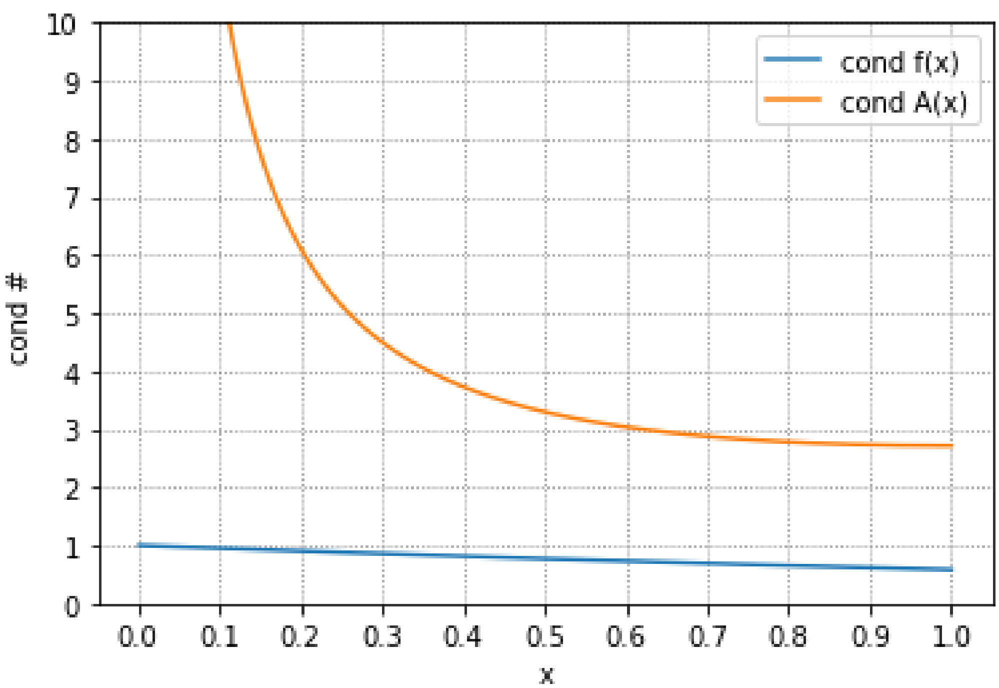
$$
\begin{aligned}
\text{fl}(1 - (\text{fl}(e^{-x}))) &= \left((1 - e^{-x}) \left(1 - \frac{e^{-x}}{1 - e^{-x}} \epsilon_{\text{exp}} \right) \right) (1 + \epsilon_{rd}) \\
&= (1 - e^{-x} - e^{-x}\epsilon_{\text{exp}})(1 + \epsilon_{rd}) \\
&= 1 - e^{-x} - e^{-x}\epsilon_{\text{exp}} + (1 - e^{-x})\epsilon_{rd}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

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

$$
\begin{aligned}
f(x_A) = (1 - e^{-x}) \left(1 - \frac{e^{-x}}{1 - e^{-x}} \epsilon_{\exp} + \epsilon_{rd}\right) &= 1 - e^{-x_A} \\
e^{-x_A} &= 1 - (1 - e^{-x}) \left(1 - \frac{e^{-x}}{1 - e^{-x}} \epsilon_{\exp} + \epsilon_{rd}\right) \\
e^{-x_A} &= 1 - (1 - e^{-x} - e^{-x}\epsilon_{\exp} + (1 - e^{-x})\epsilon_{rd}) \\
e^{-x_A} &= e^{-x} + e^{-x}\epsilon_{\exp} - \epsilon_{rd} + e^{-x}\epsilon_{rd} \\
e^{-x_A} &= e^{-x}(1 + \epsilon_{\exp} - e^{x}\epsilon_{rd} + \epsilon_{rd}) \\
-x_A &= -x + \ln(1 + \epsilon_{\exp} - e^{x}\epsilon_{rd} + \epsilon_{rd}) \\
-x_A &= -x + \epsilon_{\exp} - e^{x}\epsilon_{rd} + \epsilon_{rd} \\
x_A - x &= -\epsilon_{\exp} - e^{x}\epsilon_{rd} + \epsilon_{rd} \\
|x_A - x| &\leq -|\epsilon_{\exp}| + (1 - e^{x})|\epsilon_{rd}| \\
&\leq |-e^{x}|\text{eps} \\
&= e^{x}\text{eps}
\end{aligned}
$$

$$
\begin{aligned}
\text{cond A}(x) &\leq \frac{1}{\text{eps}} \frac{|x_A - x|}{|x|} \\
&= \frac{e^{x}}{x}
\end{aligned}
$$

(c)



The root cause of the poor conditioning of the algorithm is that in the limit as x approaches 0, $e^{-x} \approx 1$, therefore the subtraction of 1 minus something close to 1 has catastrophic cancellation. Also, as you can see the equation for the condition of the algorithm is divided by x, thus it will blow up at $x = 0$.

(d) The number of bits you lose in performing $1 - e^{-x}$:

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

The least value of x that ensures at most 1 bit of significance is lost while subtracting is:

$$\begin{aligned} 2^{-1} &\leq 1 - e^{-x} \\ \frac{1}{2} &\leq 1 - \frac{1}{e^x} \\ \frac{1}{e^x} &\leq 1 - \frac{1}{2} \\ -x &\leq \ln\left(\frac{1}{2}\right) \\ x &\geq 0.69314718056 \end{aligned}$$

For 2 bits:

$$\begin{aligned} 2^{-2} &\leq 1 - e^{-x} \\ \frac{1}{4} &\leq 1 - \frac{1}{e^x} \\ \frac{1}{e^x} &\leq 1 - \frac{1}{4} \\ -x &\leq \ln\left(\frac{3}{4}\right) \\ x &\geq 0.28768207245 \end{aligned}$$

For 3 bits:

$$\begin{aligned} 2^{-3} &\leq 1 - e^{-x} \\ \frac{1}{8} &\leq 1 - \frac{1}{e^x} \\ \frac{1}{e^x} &\leq 1 - \frac{1}{8} \\ -x &\leq \ln\left(\frac{7}{8}\right) \\ x &\geq 0.13353139262 \end{aligned}$$

For 4 bits:

$$\begin{aligned} 2^{-4} &\leq 1 - e^{-x} \\ \frac{1}{16} &\leq 1 - \frac{1}{e^x} \\ \frac{1}{e^x} &\leq 1 - \frac{1}{16} \\ -x &\leq \ln\left(\frac{15}{16}\right) \\ x &\geq 0.06453852113 \end{aligned}$$

(e) An upper bound on the relative error in the output:

$$
\begin{aligned}
\frac{|f_a(x)-f(x)|}{|f(x)|} &\leq (\mathrm{cond} f)(x)\,[\epsilon + (\mathrm{cond}\ \mathrm{A})(x^*)\cdot \mathrm{eps}] \\
\text{say } \epsilon &= 0 \text{ assuming } x \text{ is a machine number:} \\
&= \frac{x}{e^x-1}\left(\frac{e^x}{x}\right) \\
&= \frac{e^x}{e^x-1} \\
x &= -\ln\frac{1}{2} \\
\frac{|f_a(x)-f(x)|}{|f(x)|} &\approx \frac{e^{-\ln\frac{1}{2}}}{e^{-\ln\frac{1}{2}}-1} \\
&= \frac{2}{2-1} \\
&= 2 \\
x &= -\ln\frac{3}{4} \\
\frac{|f_a(x)-f(x)|}{|f(x)|} &\approx \frac{e^{-\ln\frac{3}{4}}}{e^{-\ln\frac{3}{4}}-1} \\
&= \frac{4/3}{4/3-1} \\
&= 4 \\
x &= -\ln\frac{7}{8} \\
\frac{|f_a(x)-f(x)|}{|f(x)|} &\approx \frac{e^{-\ln\frac{7}{8}}}{e^{-\ln\frac{7}{8}}-1} \\
&= \frac{8/7}{8/7-1} \\
&= 8 \\
x &= -\ln\frac{15}{16} \\
\frac{|f_a(x)-f(x)|}{|f(x)|} &\approx \frac{e^{-\ln\frac{15}{16}}}{e^{-\ln\frac{15}{16}}-1} \\
&= \frac{16/15}{16/15-1} \\
&= 16
\end{aligned}
$$

(f) An idea for an alternate algorithm for small x is:

$$
\begin{aligned}
f(x) &= 1-e^{-x} \\
&= \frac{e^x-1}{e^x} \\
&= \frac{\sum_{n=1}^{\infty}\frac{x^n}{n!}}{\sum_{n=0}^{\infty}\frac{x^n}{n!}}
\end{aligned}
$$

This method will avoid subtraction completely.

5 Limits in $\mathbb{R}(p,q)$

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

In [1]:
```
import numpy as np

def seq(n):
    value=(1+(1/n))**n
    return value

exp=np.arange(0.0000,20.0,1)
val=np.zeros(len(exp))

for idx,i in enumerate(exp):
        n=10.000**i
        val[idx]=seq(n)
        print('$10^{',i,'}$ & ',val[idx],' \\\ ')
        if idx!=0 and abs(val[idx]-val[idx-1])<1e-13:
            print('n_{stop} = 10^{',i,'}')
            print('value converged =',val[idx])
            break
```

n	value
$10^{0.0}$	2.0
$10^{1.0}$	2.5937424601000023
$10^{2.0}$	2.7048138294215285
$10^{3.0}$	2.7169239322355936
$10^{4.0}$	2.7181459268249255
$10^{5.0}$	2.7182682371922975
$10^{6.0}$	2.7182804690957534
$10^{7.0}$	2.7182816941320818
$10^{8.0}$	2.7182817983473577
$10^{9.0}$	2.7182820520115603
$10^{10.0}$	2.7182820532347876
$10^{11.0}$	2.71828205335711
$10^{12.0}$	2.7185234960372378
$10^{13.0}$	2.716110034086901
$10^{14.0}$	2.716110034087023
$10^{15.0}$	3.035035206549262
$10^{16.0}$	1.0
$10^{17.0}$	1.0

$n_{stop} = 10^{17.0}$

value converged = 1.0

The code converged at this value because at double precision, the 53-bit significand precision gives from 15 significant decimal digits precision ($2^{-52} \approx 2.22 \times 10^{-16}$). So if a decimal string with at most 15 significant digits is converted to IEEE double-precision representation, and then converted back to a decimal string with the same number of digits, the final result should match the original string. Once the code needed to calculate 10^{-16}, it rounded the result to zero and thus the solution is 1.0.

6 Fun with square roots

$$(((x^{1/2})^{1/2})^{1/2})^{\cdots} = x^{(1/2)^n} = x^{2^{-n}}$$
$$(((x^2)^2)^2)^{\cdots} = x^{2^n}$$

In performing these operations, the results from the square roots are rounded to machine numbers after each operation. The final result from the square roots for values greater than 1 + ϵ, where machine epsilon $(\epsilon) = 2^{-53}$ with double precision are mapped to the smallest binary number greater than 1, which is $+2^0 \times (1 + 2^{-52}) \approx 1.0000000000000002$. The first jump on the graph when the operations are performed 52 times is at $1.0000000000000002^{2^{52}} = 2.718281828459045 = e$. It can be seen that:

$$1.0000000000000002^{2^{50}} = 1.2840254166877414 = e^{1/4}$$
$$1.0000000000000002^{2^{51}} = 1.648721270700128 = e^{1/2}$$
$$1.0000000000000002^{2^{52}} = 2.718281828459045 = e$$
$$1.0000000000000002^{2^{53}} = 7.389056098930649 = e^2$$
$$1.0000000000000002^{2^{54}} = 54.598150033144215 = e^4$$

The same can be seen with the next smallest binary number greater than 1, which is 1.0000000000000004 :

$$1.0000000000000004^{2^{50}} = 1.648721270700128 = e^{1/2}$$
$$1.0000000000000004^{2^{51}} = 2.718281828459045 = e$$
$$1.0000000000000004^{2^{52}} = 7.389056098930649 = e^2$$
$$1.0000000000000004^{2^{53}} = 54.598150033144215 = e^4$$
$$1.0000000000000004^{2^{54}} = 2980.957987041723 = e^8$$

These values correspond to the start of the second jump, and so on.

So for $n = 50$, the jumps are at $e^{0.25}, e^{0.5}, e^{0.75}, e, e^{1.25}, e^{1.5}, e^{1.75}, e^2, e^{2.25}, \cdots$.

For $n = 51$, the jumps are at $e^{0.5}, e, e^{1.5}, e^2, \cdots$.

For $n = 52$, the jumps are at $e, e^2, \cdots$.

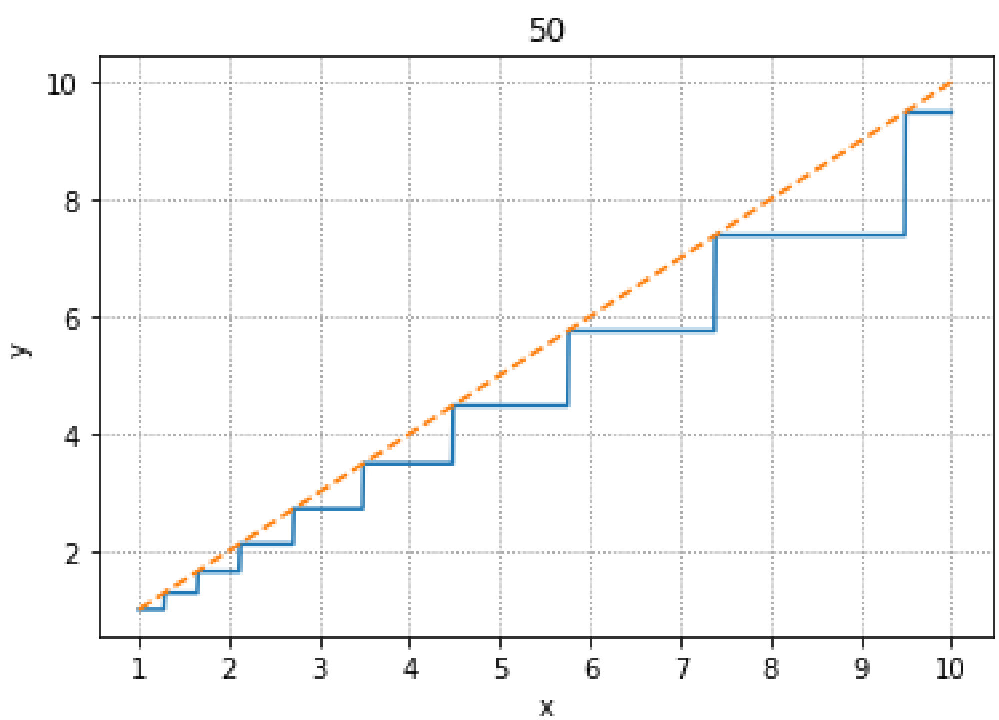
For $n = 53$, the jumps are at $e^2, e^4 \cdots$.

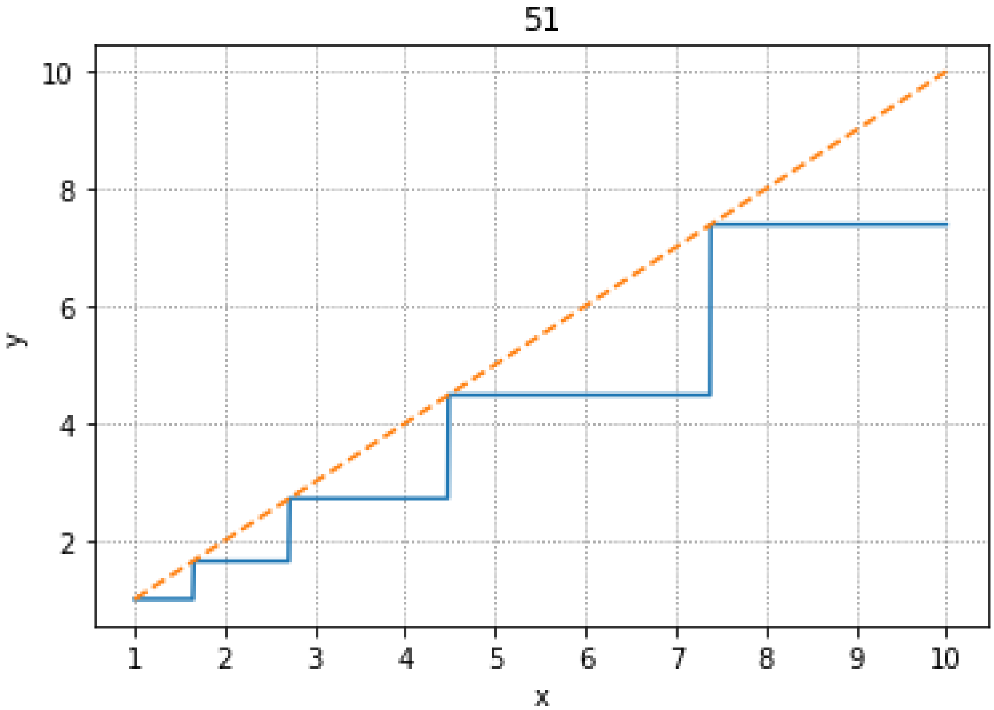
For $n = 54$, the jumps are at $e^4, e^8 \cdots$.

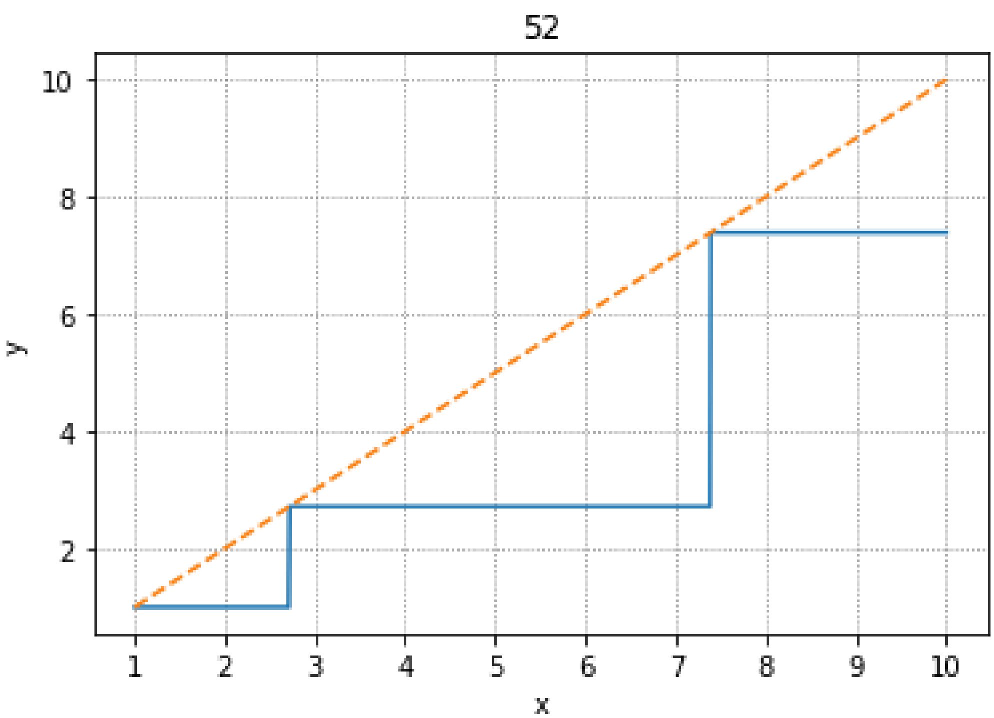
These values can be seen in the following plots:

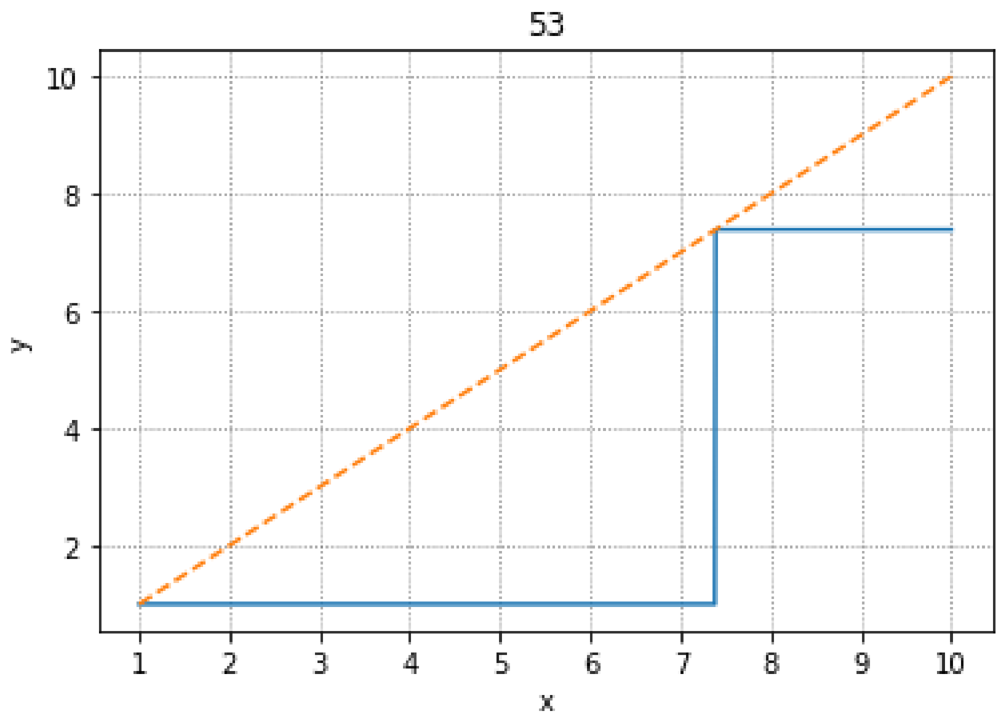
```
In [1]: import numpy as np
        import matplotlib.pyplot as pl

        num=1001
        end=10.
        y=np.linspace(1.0,end,num)
        stop=np.arange(0,5,1)
        stop=stop+50
        i=np.zeros(len(stop))
        j=np.zeros(len(stop))
```

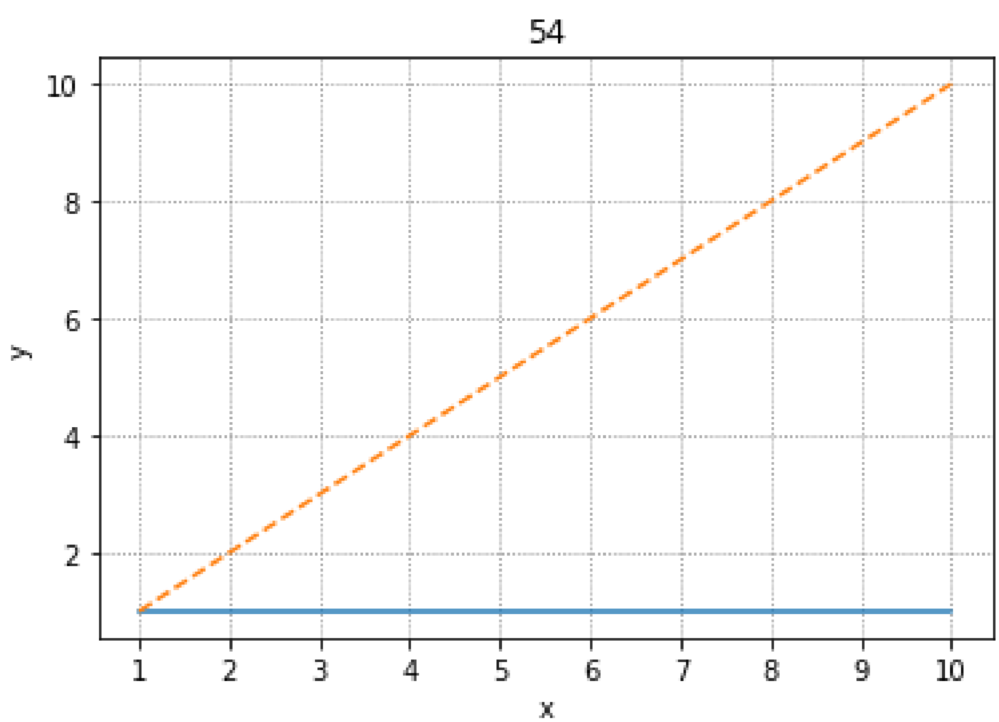
```
for idx,val in enumerate(stop):
    i[idx]=0
    while i[idx] < val:
        x=y
        y=np.sqrt(x)
        i[idx]=i[idx]+1
    j[idx]=0
    while j[idx] < val:
        x2=y
        y=np.square(x2)
        j[idx]=j[idx]+1

    x=np.linspace(1.0,end,num)
    fig,ax = pl.subplots()
    ax.plot(x, y,'-')
    ax.plot(x,x,'--')
    ax.set_title(val)
    ax.set_xlabel('x')
    ax.set_ylabel('y')
    ax.set_xticks([1,2,3,4,5,6,7,8,9,10])
    pl.rc('grid', linestyle=":", color='grey')
    pl.grid(True)
```




51
10
8
6
4
2
y
1
2
3
4
5
6
7
8
9
10
x


52
10
8
6
4
2
y
1
2
3
4
5
6
7
8
9
10
x

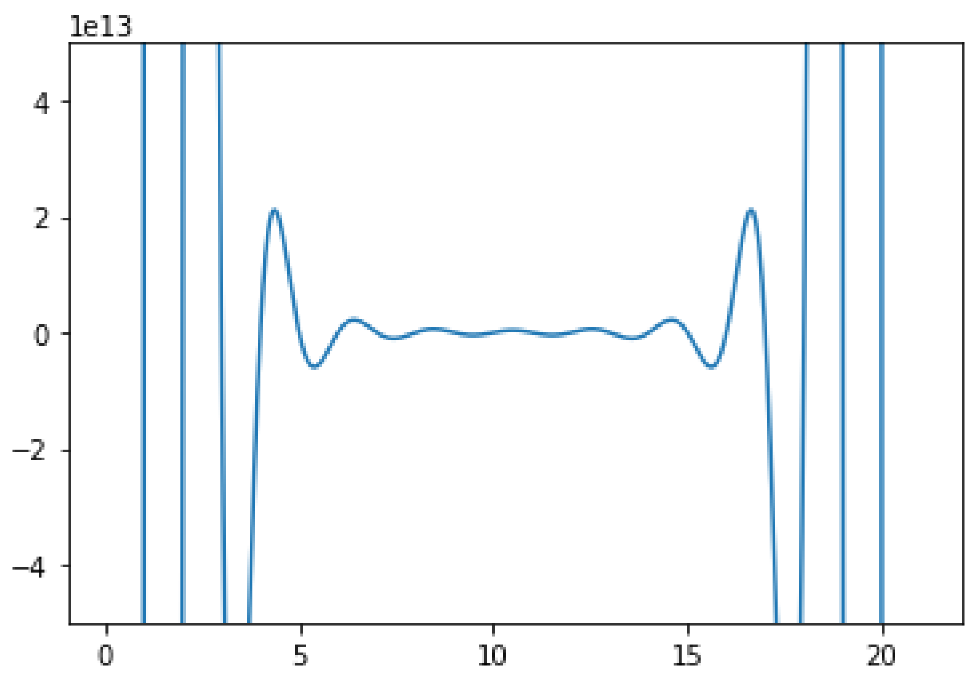

53
10
8
6
4
2
y
1
2
3
4
5
6
7
8
9
10
x


54
10
8
6
4
2
y
1
2
3
4
5
6
7
8
9
10
x

7 The issue with polynomial roots

(a) The coefficients are:

$$\begin{aligned}
a_{20} &= 1 \\
a_{19} &= -210 \\
a_{18} &= 20615 \\
a_{17} &= -1256850 \\
a_{16} &= 53327946 \\
a_{15} &= -1672280820 \\
a_{14} &= 40171771630 \\
a_{13} &= -756111184500 \\
a_{12} &= 11310276995381 \\
a_{11} &= -135585182899530 \\
a_{10} &= 1307535010540395 \\
a_9 &= -10142299865511450 \\
a_8 &= 63030812099294896 \\
a_7 &= -311333643161390640 \\
a_6 &= 1206647803780373360 \\
a_5 &= -3599979517947607200 \\
a_4 &= 8037811822645051776 \\
a_3 &= -12870931245150988800 \\
a_2 &= 13803759753640704000 \\
a_1 &= -8752948036761600000 \\
a_0 &= 2432902008176640000
\end{aligned}$$



(b) The value of the root with an initial guess of 21 is : 20.0000120683109

The polynomial evaluated at this value: -943362789888.000

The value of the largest root from numpy: 19.999853724604495

The polynomial evaluated at this value: -18556675925504.0

The values converge to numbers close to 20, but not exactly.

Numpy's built in method was not any better.

(c) The new coefficient a_{20}: 1.00000001000000

The value of the root with an initial guess of 21 is : 9.58422935762262

Here is the root from numpy: (20.647584104252793+1.1869264407682594j)

The new coefficient a_{20}: 1.00000100000000

The value of the root with an initial guess of 21 is : 7.75263279661436

Here is the root from numpy: (23.149016178734218+2.7409846232256965j)

The new coefficient a_{20}: 1.00010000000000

The value of the root with an initial guess of 21 is : 5.96933087555109

Here is the root from numpy: (28.40021241167642+6.510434219089854j)

The new coefficient a_{20}: 1.01000000000000

The value of the root with an initial guess of 21 is : 5.46959208048127

Here is the root from numpy: (38.47818361714738+20.8343235871312j)

When using the Newton-Raphson method, the largest root gets progressively smaller and smaller, further away from the actual value of 20. When using the built in root() method from numpy, the roots were driven into the complex plane.

(d) Roots 16 and 17 collide into a double root which turns into a pair of complex conjugate roots at $x \approx 16.73074487979267 \pm 2.812624896721978j$. The same occurred for roots 18 and 19, 17 and 16, 15 and 14, 13 and 12, and 11 and 10. The larger roots are greatly displaced, even though the change to the coefficient is tiny and the original roots seem widely spaced. Wilkinson showed that this behavior is related to the fact that some roots α (such as $\alpha = 15$) have many roots β that are "close" in the sense that $|\alpha - \beta| < |\alpha|$.

(e)(i)

$$
\begin{aligned}
(\text{cond } \Omega_k)(\vec{a}) &\equiv \sum_{\ell=0}^{n-1} \left| (\Gamma_{k\ell})(\vec{a}) \right| \\
&= \sum_{\ell=0}^{n-1} \left| \frac{a_\ell \frac{\partial \Omega_k}{\partial a_\ell}}{\Omega_k} \right|
\end{aligned}
$$

$$
\begin{aligned}
p(x) &= a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1} x^{n-1} + x^n \\
p'(x) &= a_1 + 2a_2 x + 3a_3 x^3 + \cdots + (n-1) a_{n-1} x^{n-2} + n x^{n-1}
\end{aligned}
$$

$$
\begin{aligned}
\text{Knowing } p(\Omega_3) = 0 \\
\text{Say } a_2 \text{ is slightly perturbed: } a_2 + \delta a_2 \\
\text{and } \Omega_3 \text{ is slightly perturbed: } \Omega_3 + \delta \Omega_3
\end{aligned}
$$

$$
\begin{aligned}
a_0 + a_1(\Omega_3 + \delta\Omega_3) + (a_2 + \delta a_2)(\Omega_3 + \delta\Omega_3)^2 + \cdots + a_{n-1}(\Omega_3 + \delta\Omega_3)^{n-1} + (\Omega_3 + \delta\Omega_3)^n = 0 \\
(a_0 + a_1\Omega_3 + \cdots + a_{n-1}\Omega_3^{n-1} + \Omega_3^n) + (a_1\delta\Omega_3 + 2a_2\delta\Omega_3\Omega_3 + \cdots + (n-1)a_{n-1}\delta\Omega_3\Omega_3^{n-2} + n a_n \delta\Omega_3\Omega_3^{n-1}) + \delta a_2 \Omega_3 = 0 \\
\cancel{p(\Omega_3)} + \delta\Omega_3 p'(\Omega_3) + \delta a_2 \Omega_3 = 0
\end{aligned}
$$

More generally:

$$
\begin{aligned}
\delta\Omega_k p'(\Omega_k) + \delta a_\ell \Omega_k &= 0 \\
\delta a_\ell \Omega_k &= -\delta\Omega_k p'(\Omega_k) \\
\frac{\Omega_k}{-p'(\Omega_k)} &= \frac{\delta\Omega_k}{\delta a_\ell}
\end{aligned}
$$

Thus

$$
\begin{aligned}
(\text{cond } \Omega_k)(\vec{a}) &= \sum_{\ell=0}^{n-1} \left| \frac{a_\ell \frac{\Omega_k}{-p'(\Omega_k)}}{\Omega_k} \right| \\
&= \sum_{\ell=0}^{n-1} \left| \frac{a_\ell}{-p'(\Omega_k)} \right| \\
&= \frac{1}{|p'(\Omega_k)|} \sum_{\ell=0}^{n-1} |a_\ell|
\end{aligned}
$$

(ii) cond 14.0 = 11334002.3222513
cond 16.0 = 1625631.97782680
cond 17.0 = 407129.794534419
cond 20.0 = 419.998958378639

The higher roots are more stable since the derivative at those points are much larger, thus they have a lower condition number.

(iii) The Wilkinson polynomial expressed in the monomial basis as we have been looking at is ill conditioned. Even the most clever algorithm will not help us here. However, if it is expressed in a Lagrange basis, then the Wilkinson polynomial is well conditioned.

8 Recurrence in reverse

(a)

$$\begin{aligned} y_n &= e - ny_{n-1} \\ y_{n-1} &= e - (n-1)y_{n-2} \\ y_{n-2} &= e - (n-2)y_{n-3} \end{aligned}$$

$$\begin{aligned} y_n &= e - ny_{n-1} \\ y_{n-1} &= \frac{e - y_n}{n} \end{aligned}$$

$$\begin{aligned} y_n &= e - n(e - (n-1)y_{n-2}) \\ y_n &= e - ne + n(n-1)y_{n-2} \\ y_n &= e(1-n) + n(n-1)y_{n-2} \\ y_{n-2} &= \frac{-e(1-n) + y_n}{n(n-1)} \\ y_{n-2} &= \frac{-(e(1-n) - y_n)}{n(n-1)} \end{aligned}$$

$$\begin{aligned} y_n &= e - ne + n(n-1)(e - (n-2)y_{n-3}) \\ y_n &= e - ne + (n^2 - n)e - (n^2 - n)(n-2)y_{n-3} \\ y_n &= e(1 - 2n + n^2) - n(n-1)(n-2)y_{n-3} \\ y_{n-3} &= \frac{e(1-n)^2 - y_n}{n(n-1)(n-2)} \end{aligned}$$

$$\begin{aligned} y_{n-m} &= (n-m)!\frac{(-1)^{m-1}(e(1-n)^{m-1} - y_n)}{n!} \\ &= (n-m)!\frac{e(n-1)^{m-1} + (-1)^m y_n}{n!} \\ y_k &= \frac{k!}{n!}\left(e(n-1)^{n-k-1} + (-1)^{n-k}y_n\right) \\ g_k(y_n) &= \frac{k!}{n!}\left(e(n-1)^{n-k-1} + (-1)^{n-k}y_n\right) \end{aligned}$$

$$\begin{aligned} |(\text{cond } g_k)(y_n)| &= \left| g'(y_n)\frac{y_n}{y_k} \right| \\ &\leq \left| \frac{k!}{n!}\frac{y_n}{y_k} \right| \\ &\leq \frac{k!}{n!}\left| \frac{y_n}{y_k} \right| \\ &\leq \frac{k!}{n!} \end{aligned}$$

(b)

$$
\begin{aligned}
\text{rel err } y_k &= (\text{cond } g_k)(y_n) * \text{rel err } y_n \\
\text{rel err } y_n &= 1 \\
\text{rel err } y_k &< \epsilon
\end{aligned}
$$

$$
\begin{aligned}
\epsilon &\geq |(\text{cond } g_k)(y_n)| \\
\epsilon &\geq \frac{k!}{n!} \\
n! &\geq \frac{k!}{\epsilon}
\end{aligned}
$$

(c)

$$
\begin{aligned}
n! &\geq \frac{20!}{\text{eps}} \\
n &= 32 \text{ is the minimum n for this value of } k \text{ and } \epsilon \\
2.631308369336935e + 35 &> 1.0956816577453247e + 34
\end{aligned}
$$

(d) Using the recursion relation:

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

and starting from a seed value of 0 at the value N=32 gives an extremely accurate solution to the integral after simply 12 iterations.

$$
\begin{aligned}
y_{32} &= 0.0 \\
y_{31} &= 0.08494630713934516 \\
y_{30} &= 0.08494630713934516 \\
y_{29} &= 0.08777785071065666 \\
y_{28} &= 0.09070703371546167 \\
y_{27} &= 0.09384195695512798 \\
y_{26} &= 0.0972014767223673 \\
y_{25} &= 0.10081078275910299 \\
y_{24} &= 0.10469884182799769 \\
y_{23} &= 0.10889929110962698 \\
y_{22} &= 0.113451414667366 \\
y_{21} &= 0.11840138244507631 \\
y_{20} &= 0.12380383076256993
\end{aligned}
$$

Compared to Wolfram Alpha:

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