HOMEWORK 1

APC 523: Numerical Algorithm for Scientific Computing

Due: Oct.10

PROBLEM 1

Let $x = 2^k(1 + \sum_{i=1}^{\infty} 2^{-i} b_i)$, where $k = floor(\log_2 x)$
or $x = 1b_1 b_2 \cdots b_k.b_{k+1} b_{k+2} \cdots$

case 1: if $b_{p+1} = 1$, then $rd(x) = 2^k(1 + \sum_{i=1}^{p} 2^{-i} b_i + 2^{-p})$
$\Rightarrow |\frac{x - rd(x)}{x}| = \frac{2^k(-2^{-p} + \sum_{i=p+1}^{\infty} 2^{-i} b_i)}{2^k(1 + \sum_{i=1}^{\infty} 2^{-i} b_i)} \leq 2^{-p} - \sum_{i=p+1}^{\infty} 2^{-i} b_i \leq 2^{-p}$

case 2: if $b_{p+1} = 0$, then $rd(x) = 2^k(1 + \sum_{i=1}^{p} 2^{-i} b_i)$
$\Rightarrow |\frac{x - rd(x)}{x}| = \frac{2^k(\sum_{i=p+1}^{\infty} 2^{-i} b_i)}{2^k(1 + \sum_{i=1}^{\infty} 2^{-i} b_i)} \leq \sum_{i=p+1}^{\infty} 2^{-i} b_i \leq \sum_{i=p+1}^{\infty} 2^{-i} \leq 2^{-p}$

Therefore, $\Rightarrow |\frac{x - rd(x)}{x}| \leq 2^{-p}$

PROBLEM 2

(a) The result is 244.71

(b) Upto $k = 17$, the result does not change with 5 digits precision. The exact value is $e^{5.5} = 244.692$, so relative error = $7.36(10^{-5})$

(c) Upto $k = 17$, the result does not change with 5 digits precision, which is 244.70. The exact value is $e^{5.5} = 244.692$, so relative error = $3.27(10^{-5})$

(d)

	# of terms k	result	relative error ($e^{-5.5} = 0.00408677$)
i	25	0.038363	0.0613
ii	19	0.0040000	0.0212
iii	17	0.0000	1
iv	17	0.0000	1

Methods iii and iv converge most quickly, but least accurate. Method ii has lowest error.
Empirically, adding from right to left is better and results in less error.

(e) I propose algorithm:
(i) compute $e^{0.5}$ (adding from left to right)
(ii) divide $e^{0.5}$ by $e = 2.7183$ repeatedly for 6 times.

Validation: result = 0.004086
relative error = $1.88(10^{-4})$
# of terms k = 4

PROBLEM 3

(a)

(i)

$fl(x * x) = x^2(1 + \epsilon)$

$fl(x^2(1+\epsilon) * x) = x^3(1+2\epsilon)$
...
$fl(x^{n-1}(1+(n-2)\epsilon) * x) = x^n(1+(n-1)\epsilon)$

(ii)
$fl(\ln x) = \ln x(1+\epsilon)$
$fl(n * lnx(1+\epsilon)) = n\ln x(1+\epsilon)^2 \approx n\ln x(1+2\epsilon)$
$fl(e^{n\ln x(1+2\epsilon)}) = x^n e^{2\epsilon}(1+\epsilon) \approx x^n(1+3\epsilon)$
When $n-1 < 3$ or $n < 4$, repeated multiplication is more accurate than log-exponential method.

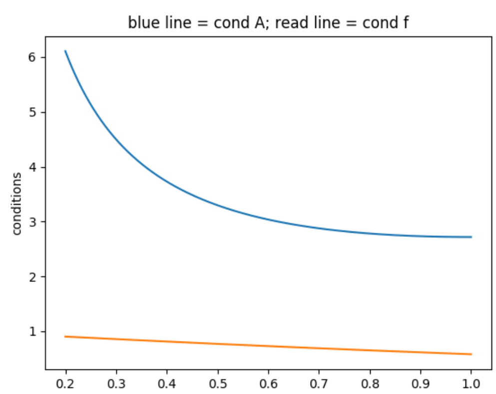
(b)
(i) $x^{a(1+\epsilon_a)} = x^a x^{a\epsilon_a} = x^1 e^{a\epsilon_a \ln x} \approx x^a(1+a\epsilon_a \ln x)$
(ii) $(x(1+\epsilon+x))^a = x^a(1+\epsilon_x)^a \approx x^a(1+a\epsilon_x)$
when $x \to 0, \infty$ or $|a|$ becomes large, the relative error is substantial.

PROBLEM 4

(a)
(cond $f)(x) = |\frac{xf'}{f}| = \frac{x}{e^x-1} \leq 1$ because $e^x - 1 > x$

(b)
$f_A(x) = [1-e^{-x}(1+EPS)](1+EPS) \approx (1-e^{-x})(1-\frac{e^{-x}}{1-e^{-x}}EPS)(1+EPS) \approx (1-e^{-x})(1+\frac{EPS}{1-e^{-x}})$
$f_A(x) = f(x_A) \Rightarrow |f_A(x) - f(x)| = |f(x_A) - f(x)| \approx f(x)\frac{EPS}{1-e^{-x}} \approx |f'(x)||x-x_A|$
$\Rightarrow |\frac{x-x_A}{x}| \approx |\frac{f(x)}{xf'(x)}\frac{EPS}{1-e^{-x}}|$
$\Rightarrow$ (cond $A)(x) \approx \frac{1}{EPS}|\frac{x-x_A}{x}| \approx \frac{1}{(1-e^{-x})(\text{cond } f)(x)} = \frac{e^x}{x}$

(c)



The root cause of ill conditioning is that when x is small, $1-e^{-x}$ introduces a large error.

(d)+(e)
$1-e^{-x} < 2^{-b}$ implies that less than less than b bits are lost, so the minimum allowed x should be

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

In addition, the relative error should be

$$\epsilon = \frac{EPS}{1-e^{-x}} = 2^b EPS$$

# bits lost	min x	relative error
1	0.693	2^{-51}
2	0.288	2^{-50}
3	0.134	2^{-49}
4	0.0645	2^{-48}

(f)
Yes, I propose the algorithm using Taylor expansion.
(i) Find $e^x - 1 = \sum_{n=1}^{20} \frac{x^n}{n!}$ (20 terms should be sufficiently convergent)
(ii) Find $f_A(x) = \frac{\sum_{n=1}^{20} \frac{x^n}{n!}}{1+\sum_{n=1}^{20} \frac{x^n}{n!}}$

PROBLEM 5

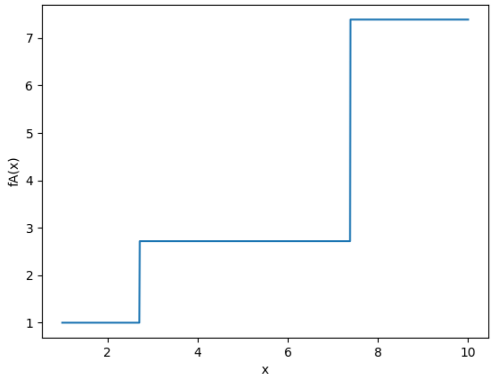
$n_{stop} = 17$, converges to 1.

n	result
n result 10^1	2.5937424601000
10^2	2.7048138294215
10^3	2.7169239322356
10^4	2.7181459268249
10^5	2.7182682371923
10^6	2.7182804690958
10^7	2.7182816941321
10^8	2.7182817983474
10^9	2.7182820520116
10^10	2.7182820532348
10^11	2.7182820533571
10^12	2.7185234960372
10^13	2.7161100340869
10^14	2.7161100340870
10^15	3.0350352065493
10^16	1.0000000000000
10^17	1.0000000000000

This is caused by the following reasons:

1. As n gets large, $(1 + \frac{1}{n}(1 + EPS))^n = (1 + \frac{1}{n})^n(1 + 10n\log(n)EPS$, and the relative error will increase with n.
2. When $n = 10^{16}$, $\frac{1}{n} < EPS$, $1 + \frac{1}{n} \approx 1 \Rightarrow (1 + \frac{1}{n})^n \approx 1$

PROBLEM 6



To explain this phenomenon let's define $f_1(x) = x^{2^{-52}}$, $f_2(x) = x^{2^{52}}$, such that our algorithm is $f_A(x) = f_2 \circ f_1(x) = x$

Let's first look at $f_1(x)$, it can be shown that if $x = e^y$, where $e^y << 2^{52}$, then

$$\begin{aligned} f_1(x) &= e^{y(2^{-52})} \\ &\approx 1 + y * 2^{-52} \\ &= 1 + y * EPS \end{aligned}$$

Therefore,

if $x \in [1, e)$, $f_1(x) < 1 + EPS \approx 1 \Rightarrow f_A(x) = 1$;

if $x \in [e, e^2)$, $f_1(x) \in [1 + EPS, 1 + 2 * EPS) \approx 1 + EPS \Rightarrow f_A(x) = (1 + EPS)^{2^{52}}$;

if $x \in [e^2, e^3)$, $f_1(x) \in [1 + 2 * EPS, 1 + 3 * EPS) \approx 1 + 2 * EPS \Rightarrow f_A(x) = (1 + 2 * EPS)^{2^{52}}$

PROBLEM 7

(a)

$w(x) = x^{20} - 210x^{19} + 20615x^{18} - 1256850x^{17} + 53327946x^{16} - 1672280820x^{15} + 40171771630x^{14} - 756111184500x^{13} + 11310276995381x^{12} - 135585182899530x^{11} + 1307535010540395x^{10} - 10142299865511450x^{9} + 63030812099294896x^{8} - 311333643161390640x^{7} + 1206647803780373360x^{6} - 3599979517947607200x^{5} + 8037811822645051776x^{4} + 5575812828558562816x^{3} - 4642984320068847616x^{2} - 8752948036761600000x + 2432902008176640000$

(b)

root = 19.99987405572419

(c)

δ	max root
10^{-8}	(20.647582887998496+1.1869261883090942j)
10^{-6}	(23.149016020150878+2.740984637982632j)
10^{-4}	(28.40021241591655+6.5104342165628175j)
10^{-2}	(38.478183617151515+20.83432358712749j)

(d) Let $a_{19} = -210 - 2^{-23}$, roots 16, 17 become 16.73074488+2.8126249j, 16.73074488-2.8126249j

(e)

(i) To find (cond $\Omega_k)(\vec{a})$, let's impose a perturbation on a_l, such that $a_l(1+\epsilon_l)$ results in a new root $\Omega_k(1+\epsilon_k)$

By definition, $a_0 + a_1\Omega_k + \cdots a_n\Omega_k^n = 0$. In addition, after perturbation, we have

$$\begin{aligned}
0 &= a_0 + a_1\Omega_k(1+\epsilon_k) + \cdots + a_l(1+\epsilon_l)\Omega_k^l(1+\epsilon_k)^l + \cdots + a_n\Omega_k^n(1+\epsilon_k)^n \\
&= a_1\Omega_k(\epsilon_k) + a_2\Omega_k^2(2\epsilon_k) + \cdots + a_l\Omega_k^l(l\epsilon_k + \epsilon_l) + \cdots a_n\Omega_k^n(n\epsilon_k) \\
&= p'(\Omega_k)\Omega_k\epsilon_k + a_l\Omega_k^l\epsilon_l \\
\Rightarrow \Gamma_{kl} &= |\frac{\epsilon_k}{\epsilon_l}| = |\frac{a_l\Omega_k^l}{p'(\Omega_k)\Omega_k}|
\end{aligned}$$

Therefore,

$$(\text{cond } \Omega_k)(\vec{a}) = \sum_{l=0}^{n} |\frac{a_l\Omega_k^l}{p'(\Omega_k)\Omega_k}|$$

(ii)

Ω_k	(cond $\Omega_k)(\vec{a})$
14	251350894804.7394
16	104194884779.65079
17	71181306412.37926
20	35518935656.3619

(iii)

There is no smart algorithm because the problem is by nature ill-conditioned.

PROBLEM 8

(a)

Use the recurrence relation $y_{n-1} = \frac{e-y_n}{n}$

$y_{k-1} = \frac{e-y_k(1+\epsilon_k)}{k} = \frac{e-y_k}{k}(1 + \frac{y_k}{e-y_k}\epsilon_k)$

$\Rightarrow \epsilon_{k-1} = \frac{y_k}{e-y_k}\epsilon_k$

Estimation of Upper Bound:

$y_k \le \int_0^1 ex^k dx = \frac{e}{k+1} \Rightarrow \epsilon_{k-1} \le \frac{\frac{e}{k+1}}{e-\frac{e}{k+1}} = \frac{1}{k}\epsilon_k$

$\Rightarrow \epsilon_k \le (\frac{1}{k})^{N-k}\epsilon_N$

$\Rightarrow (\text{cond } g_k)(y_N) \le (\frac{1}{k})^{N-k}$

(b)

$\epsilon_N = 1 \Rightarrow \epsilon_k \le (\frac{1}{k})^{N-k}\epsilon_N = (\frac{1}{k})^{N-k}$

$\Rightarrow N \ge k + \log_k(1/\epsilon)$

(c)

$\epsilon = EPA = 2.2(10^{-16}) \Rightarrow N = 20 - \log_{20}(2.2) + 16\log_{20}(10) = 32.03 \approx 33$

(d)

From Wolfram : $y_{20} = 0.12380$

$N = 33 \Rightarrow y_{20} = 0.12380$, so the result can be verified.