PROBLEM SET #1

APC 523/MAE 507/AST 523 : Numerical Algorithms for Scientific Computing

Vivek Kumar

March 13, 2019

1 Error in (symmetric) rounding vs chopping

Assertion: When mapping a real number x to a nearby machine number in $\mathbb{R}(p,q)$, the upper bound in the relative error for symmetric rounding is:

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

Proof:
Consider the number x to be represented as:

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

If the number is to be rounded to p terms, two cases arise:

CASE I. The $(p+1)^{\text{th}}$ is 0.
In this scenario the difference between the true value and the rounded value is given by:

$$x-\mathrm{rd}(x) = \pm\left(\sum_{l=p+2}^{\infty} b_{-l}2^{-l}\right)2^{e}$$

The maximum relative error can then be computed as:

$$\begin{aligned}\max\left|\frac{x-\mathrm{rd}(x)}{x}\right| &= \frac{\max|x-\mathrm{rd}(x)|}{\min|x|}\\ &= \frac{2^{-p-1}2^{e}}{2^{-1}2^{e}}\\ &= 2^{-p}\end{aligned}$$

which is what we set to prove.

CASE II. The $(p+1)^{\text{th}}$ is 1.
In this scenario the maximum difference between the true and the rounded value is obtained as:

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

This is the case we all the leading terms from $(p+2)$ are 1. Hence the maximum relative error can be computed as before:

$$\begin{aligned}\max\left|\frac{x-\mathrm{rd}(x)}{x}\right| &= \frac{\max|x-\mathrm{rd}(x)|}{\min|x|}\\ &= \frac{\left(2^{-p}-2^{-p-1}\right)2^{e}}{2^{-1}2^{e}}\\ &= 2^{-p}\end{aligned}$$

which is what we set to prove

Both the cases show that the maximum symmetric rounding off error is 2^{-p}

2 An accurate implementation of e^x

(a) Work out the terms of the infinite series upto $n = 30$ by rounding upto 5-significant figures

n	numerator	denominator	n^{th} term
0	1.0000	1.0000	1.0000
1	5.50000E+00	1.00000E+00	5.50000E+00
2	3.02500E+01	2.00000E+00	1.51250E+01
3	1.66380E+02	6.00000E+00	2.77300E+01
4	9.15090E+02	2.40000E+01	3.81290E+01
5	5.03300E+03	1.20000E+02	4.19420E+01
6	2.76820E+04	7.20000E+02	3.84470E+01
7	1.52250E+05	5.04000E+03	3.02080E+01
8	8.37380E+05	4.03200E+04	2.07680E+01
9	4.60560E+06	3.62880E+05	1.26920E+01
10	2.53310E+07	3.62880E+06	6.98050E+00
11	1.39320E+08	3.99170E+07	3.49020E+00
12	7.66260E+08	4.79000E+08	1.59970E+00
13	4.21440E+09	6.22700E+09	6.76790E-01
14	2.31790E+10	8.71780E+10	2.65880E-01
15	1.27480E+11	1.30770E+12	9.74840E-02
16	7.01140E+11	2.09230E+13	3.35100E-02
17	3.85630E+12	3.55690E+14	1.08420E-02
18	2.12100E+13	6.40240E+15	3.31280E-03
19	1.16660E+14	1.21650E+17	9.58980E-04
20	6.41630E+14	2.43300E+18	2.63720E-04
21	3.52900E+15	5.10930E+19	6.90700E-05
22	1.94100E+16	1.12400E+21	1.72690E-05
23	1.06760E+17	2.58520E+22	4.12970E-06
24	5.87180E+17	6.20450E+23	9.46380E-07
25	3.22950E+18	1.55110E+25	2.08210E-07
26	1.77620E+19	4.03290E+26	4.40430E-08
27	9.76910E+19	1.08890E+28	8.97150E-09
28	5.37300E+20	3.04890E+29	1.76230E-09
29	2.95510E+21	8.84180E+30	3.34220E-10
30	1.62530E+22	2.65250E+32	6.12740E-11

(b) Compute the $e^{5.5}$ using partial sums for left to right

n	value
0	1.0000
1	6.50000E+00
2	2.16250E+01
3	4.93550E+01
4	8.74840E+01
5	1.29430E+02
6	1.67880E+02
7	1.98090E+02
8	2.18860E+02
9	2.31550E+02
10	2.38530E+02
11	2.42020E+02
12	2.43620E+02
13	2.44300E+02
14	2.44570E+02
15	2.44670E+02
16	2.44700E+02
17	2.44710E+02
18	2.44710E+02
19	2.44710E+02
20	2.44710E+02
21	2.44710E+02
22	2.44710E+02
23	2.44710E+02
24	2.44710E+02
25	2.44710E+02
26	2.44710E+02
27	2.44710E+02
28	2.44710E+02
29	2.44710E+02
30	2.44710E+02

- For $k = 17$, $e^{5.5}$ converges to 5-significant figures
- The value of $e^{5.5}$ computed using built-in function is 244.691932
- The relative error is 0.000074

(c) For summing right to left

n	value
0	3.34220E-10
1	2.09650E-09
2	1.10680E-08
3	5.51110E-08
4	2.63320E-07
5	1.20970E-06
6	5.33940E-06
7	2.26080E-05
8	9.16780E-05
9	3.55400E-04
10	1.31440E-03
11	4.62720E-03
12	1.54690E-02
13	4.89790E-02
14	1.46460E-01
15	4.12340E-01
16	1.08910E+00
17	2.68880E+00
18	6.17900E+00
19	1.31600E+01
20	2.58520E+01
21	4.66200E+01
22	7.68280E+01
23	1.15280E+02
24	1.57220E+02
25	1.95350E+02
26	2.23080E+02
27	2.38200E+02
28	2.43700E+02
29	2.44700E+02
30	2.44700E+02

- For $k = 29$, $e^{5.5}$ converges to 5-significant figures
- The value of $e^{5.5}$ computed using built-in function is 244.691932
- The relative error is 0.000033

(d) Check

3 Error propagation in exponentiation

(a) Derive the upper bound on relative error resulting from machine arithmetic for the two different algorithms

(i) Multiplying x's one we get:

$$\mathrm{fl}\,(x \times x) = x^2(1+\epsilon)$$

By induction, the relative error in computing x^n by repeated multiplication is $(1+\epsilon)^{n-1}$. If we neglect any terms of type $\mathcal{O}\left(\mathrm{eps}^2\right)$ and higher the error can be written as:

$$\text{Relative Error} = (n-1)\,\epsilon$$

(ii) Using exponentiation:

$$\begin{aligned}
\mathrm{fl}\left(e^{\mathrm{fl}(n(\mathrm{fl}(\ln x)))}\right) &= \mathrm{fl}\left(e^{\mathrm{fl}(n(\ln x(1+\epsilon_l)))}\right) \\
&= \mathrm{fl}\left(e^{n\ln x(1+\epsilon_l+\epsilon_n)}\right) \\
&= e^{n\ln x(1+\epsilon_l+\epsilon_n)}\left(1+\epsilon_{\mathrm{exp}}\right) \\
&= x^{n(1+\epsilon_l+\epsilon_n)}\left(1+\epsilon_{\mathrm{exp}}\right) \\
&= x^n\left(x^{n(\epsilon_l+\epsilon_n)}\left(1+\epsilon_{\mathrm{exp}}\right)\right)
\end{aligned}$$

(b) Suppose x is positive and a is nonzero. Determine the propagated relative error ϵ in x^a when:

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

(ii) a is an exact machine number but x is subject to a relative error ϵ_x

NOTE: Since a is an arbitrary positive number we only focus on the exponentiation method

4 Conditioning

Consider the function

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

(a) The condition (cond f)(x) of the function in terms of x is given as:

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

The function $\frac{x}{e^x-1}$ is a monotonically decreasing function between $[0, 1]$ with values being 1 at 0 and 0.58 at 1. Hence the problem is well conditioned on this interval

(b)

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

(a) The code stopped at $n = 10^{17}$

(b) The final converged value is 1.0

(c) A table of the intermediate values computed for $0 \leq n \leq n_{\text{stop}}$

n	value
1.0E+00	2.0000000000000
1.0E+01	2.5937424601000
1.0E+02	2.7048138294215
1.0E+03	2.7169239322356
1.0E+04	2.7181459268249
1.0E+05	2.7182682371923
1.0E+06	2.7182804690958
1.0E+07	2.7182816941321
1.0E+08	2.7182817983474
1.0E+09	2.7182820520116
1.0E+10	2.7182820532348
1.0E+11	2.7182820533571
1.0E+12	2.7185234960372
1.0E+13	2.7161100340869
1.0E+14	2.7161100340870
1.0E+15	3.0350352065493
1.0E+16	1.0000000000000
1.0E+17	1.0000000000000

The reason why it converges to that value is that the term $\frac{1}{n}$ gets ignored when n reaches 10^{16} as the machine epsilon for floating point in python3 is $\approx 10^{-17}$ (This can be tested by computing $0.1 + 0.1 + 0.1 - 0.3$).

6 Fun with square roots

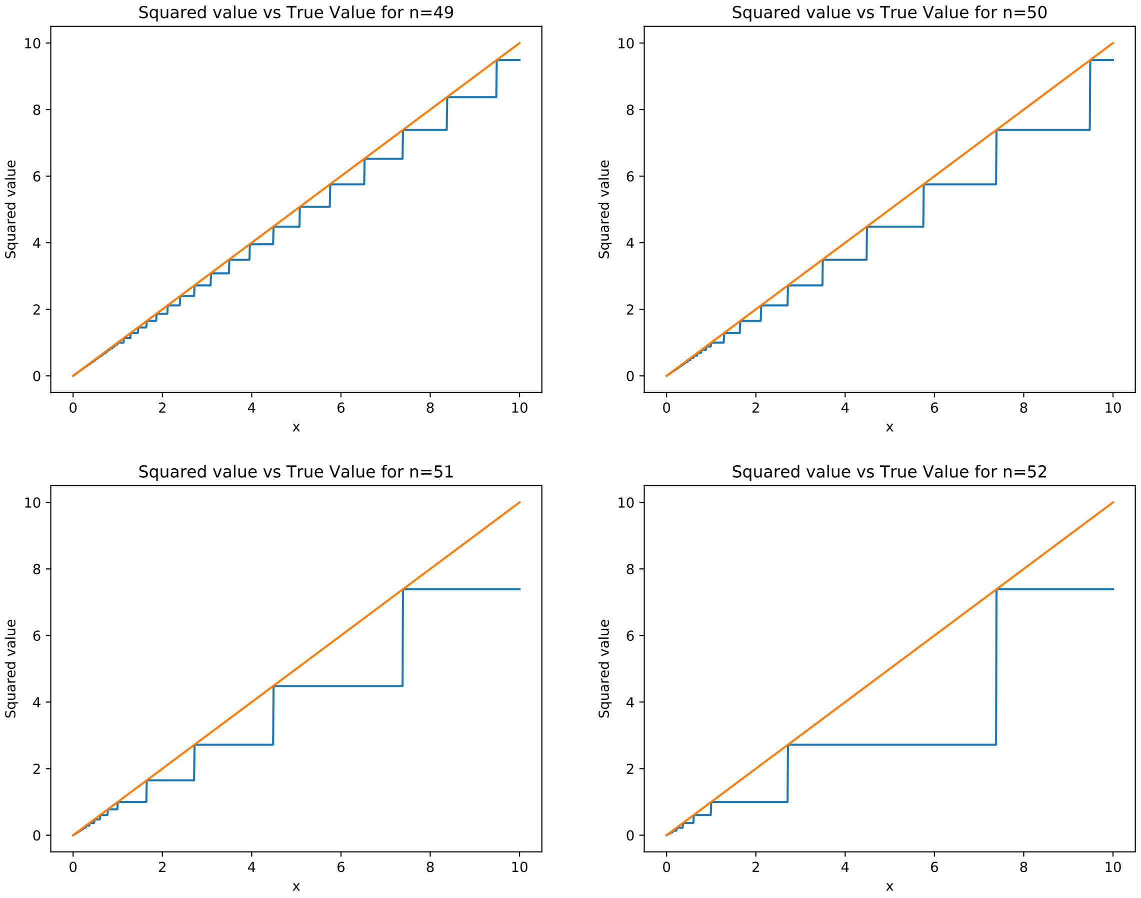

Figure 1: Plots for different values of n

7 The issue with polynomial roots

(a) The coefficients of the polynomial are:

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

(b) Yes, The newton raphson method converges to 20.00003189 when the initial guess of 21 is provided.
Using the inbuilt function the largest root obtained is 20.00054209, which is pretty much the same.Further various complex roots are calculated using the inbuilt function

(c) Changing the coefficient a_{20} from $1 \rightarrow 1 + \delta$

δ	Largest root using NR	Largest root using inbuilt function
10^{-8}	9.5854	$20.648 + 1.1869j$
10^{-6}	7.7527	$23.149 + 2.7410j$
10^{-4}	5.9693	$28.400 + 6.5104j$
10^{-2}	5.4696	$38.478 + 20.834j$

(d) Changing the value of coefficient a_{19} from $-210 \rightarrow -210 - 2^{-23}$

(e) Consider a monic degree-n polynomial

8 Recurrence in reverse

(a) The reverse recurrence relation is given by:

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

Computing for a few terms down the chain we obtain:

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

One can denote the pattern as:

$$y_{n-p} = (-1)^p \frac{y_n}{n\left(n-1\right)\left(n-2\right)\ldots\left(n-p+1\right)} + e\left[\frac{1}{n-(p-1)} - \frac{1}{\left(n-(p-1)\right)\left(n-(p-2)\right)} + \frac{1}{\left(n-(p-1)\right)\left(n-(p-2)\right)\left(n-(p-3)\right)} + \ldots\right]$$

To obtain the value of y_k in terms of y_N we replace $n-p$ with k and simplify:

$$\begin{aligned} y_k &= (-1)^{n-k} \frac{y_n}{n\left(n-1\right)\left(n-2\right)\ldots\left(k+1\right)} + \text{exponent terms} \\ &= (-1)^{n-k}\frac{y_n k!}{n!} + \text{exponent terms} \end{aligned}$$

The condition number is given as:

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

Since the k is less than N, y_k is greater than y_N, the upper bound on the condition number, $\left(\text{cond } g_k\right)\left(y_k\right)$, obtained as:

$$\left(\text{cond } g_k\right)\left(y_k\right) \le \frac{k!}{N!}$$

as $\frac{y_N}{y_k} \le 1$

(b) We know the condition number represents:

$$\begin{aligned} \varepsilon_y &= \left(\text{cond } g_k\right)\varepsilon_x \\ \frac{\Delta y_k}{y_k} &\le \frac{k!}{N!} \quad \le \varepsilon \\ N! &\ge \frac{k!}{\varepsilon} \end{aligned}$$

Here, we have assumed $\varepsilon_x = 1$ and ε is a predefined target error in y_k.

(c) For `python3` the machine epsilon for float is $1.0e^{-15}$(Obtained using numpy.finfo(float)). Using this machine epsilon the value of N obtained is 31. [Check code]

(d) The computed value of y_{20} is 0.123803830762570 and the value of y_{20} directly by integration is 0.123803830762570. [Check code]