553.481/681 Numerical Analysis – Homework 1 Solutions

Problem 1. (a) Any binary number $(\cdots b_2 b_1 b_0 . b_{-1} b_{-2} \cdots)_2$ can be written as the decimal $\sum_i b_i 2^i = \sum_i (b_{4i+3} \cdot 2^3 + b_{4i+2} \cdot 2^2 + b_{4i+1} \cdot 2 + b_{4i}) 2^{4i}$. Since $\forall i : b_i \in \{0, 1\}$, we must have that $b_{4i+3} \cdot 2^3 + b_{4i+2} \cdot 2^2 + b_{4i+1} \cdot 2 + b_{4i} \in \{0, \ldots, 15\}$, i.e. is represented by one hexadecimal symbol, and noting that $2^{4i} = 16^i$, we're done.

(b) Coded in `MATLAB`.

```
a=10;b=11;c=12;d=13;e=14;f=15;

h=[1 f b c 4 7 a 3];
n=length(h);

bb=[];
for ii=1:n
        z=h(ii);
        z0=mod(z,2);
        z=(z-z0)/2;
        z1=mod(z,2);
        z=(z-z1)/2;
        z2=mod(z,2);
        z3=(z-z2)/2;
        bb=[bb, [z3 z2 z1 z0]];
end

bb=num2str(bb);
bb= bb(~isspace(bb))
```

(c) With `h='ef8224d4'`, we obtain the binary representation `11101111100000100010010011010100`.

Problem 2. (a) We can convert $(c00)_{16} = 12 * 16^2 + 0 * 16^1 + 0 * 16^0 = 3072$. Since $3072 > 2^{11} = 2048$, the sign is $\sigma = -1$ and we subtract the bias plus $2^{11} = 2048$ from this characteristic to find the exponent

$$E = 3072 - (2^{r-1} - 1 + 2^{11}) = 3072 - (2^1 0 - 1 + 2048) = 3072 - 3071 = 1$$

(b) We can convert

$$\begin{aligned} F = (0.ae10bbc76824b)_{16} &= (a, e, 1, \ldots, 2, 4, b) \cdot (16^{-1}, 16^{-2}, 16^{-3}, \ldots, 16^{-11}, 16^{-12} 16^{-13}) \\ &= 0.679942833121589 \end{aligned}$$

(c) Per IEEE format we know the number is

$$\begin{aligned} \sigma[1 + F] \cdot 2^E &= -1[1 + 0.679942833121589] \cdot 2^1 \\ &= -3.359885666243178 \end{aligned}$$

Note that this is a double-precision approximation to the negative of the reciprocal Fibonacci constant:

`https://en.wikipedia.org/wiki/Reciprocal_Fibonacci_constant`

Problem 3. (a) Note $(1.0\ldots0)_\beta \cdot \beta^L = \beta^L$.

(b) Let $\xi := \beta - 1$ and write

$$\begin{aligned}(\xi.\xi\cdots\xi)_\beta \cdot \beta^U &= \xi \sum_{i=0}^{p} \beta^{-k} \cdot \beta^U \\ &= (\beta-1)\sum_{i=0}^{p} \beta^{-k} \cdot \beta^U \\ &= (\beta-1)\frac{1/\beta^{p+1}-1}{1/\beta-1} \cdot \beta^U \\ &= (\beta-1)\frac{\beta^{-p}-\beta}{1-\beta} \cdot \beta^U \\ &= (\beta-\beta^{-p}) \cdot \beta^U.\end{aligned}$$

(c) Write $(0.0\cdots01)_\beta \cdot \beta^L = \beta^{-p} \cdot \beta^L = \beta^{L-p}$.

Problem 4. (a) Note that $2\cos x = e^{ix} + e^{-ix}$. Therefore

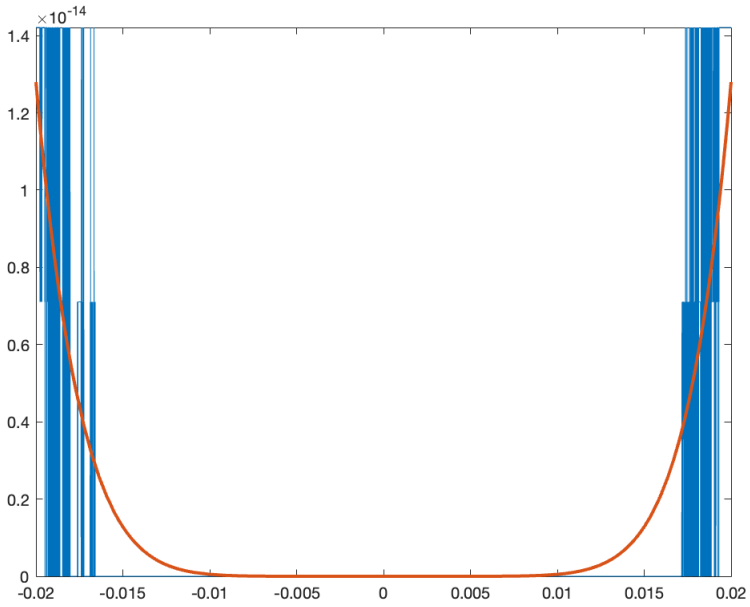
$$\begin{aligned}[2(1-\cos(x)]^4 &= [2-(e^{ix}+e^{-ix})]^4 \\ &= [4-4(e^{ix}+e^{-ix})+(e^{ix}+e^{-ix})^2]^2 \\ &= 16-4\cdot 8(e^{ix}+e^{-ix})+6\cdot 4(e^{ix}+e^{-ix})^2-4\cdot 2(e^{ix}+e^{-ix})^3+(e^{ix}+e^{-ix})^4 \\ &= 16-64\cos(x)+24\cdot(2+e^{2ix}+e^{-2ix})-8(e^{3ix}+3e^{ix}+3e^{-ix}+e^{-3ix}) \\ &\qquad +(e^{4ix}+4e^{2ix}+6+4e^{-2ix}+e^{-4ix}) \\ &= 16-64\cos(x)+24\cdot 2[1+\cos(2x)]-16(3\cos(x)+\cos(3x))+2(3+4\cos(2x)+\cos(4x))\end{aligned}$$

Finally,

$$\begin{aligned}8(1-\cos(x))^4 &= 8-32\cos(x)+24[1+\cos(2x)]-8(3\cos(x)+\cos(3x))+3+4\cos(2x)+\cos(4x) \\ 8(1-\cos(x))^4 &= 35-32\cos(x)+24\cos(2x)-24\cos(x)-8\cos(3x)+4\cos(2x)+\cos(4x) \\ 8(1-\cos(x))^4 &= \cos(4x)-8\cos(3x)+28\cos(2x)-56\cos(x)+35\end{aligned}$$

(b) Code in `MATLAB`. The graph is below where $f(x)$ is in blue.

```
 f=@(x) cos(4*x)-8*cos(3*x)+28*cos(2*x)-56*cos(x)+35;
fplot(f,[-0.02,.02])
xx=-0.02:.0001:.02;
xm1=min(f(xx))
g=@(x) 8*(1-cos(x)).^4;
hold on
fplot(g,[-0.02,.02],'LineWidth',2)
xm2=min(g(xx))
```



No, the result does not look like the plot of a smooth polynomial.

(c) From the identity in part a) it is clear that $x = 0$ is a minimum, since $8(1 - \cos x)^4$ is a nonnegative function that is zero at $x = 0$. However, when using the Matlab command, the minimum it computes for $f(x)$ is -7.1054×10^{-15}. A negative minimum value is not mathematically possible, since $f(x) \geq 0$! This error is due to round-off, which we expect from the very "rough" plot of the function in part (b).

(d) In the plot from part (b), $g(x)$ is the red curve. This plot is much more accurate. The only propagated errors in this formulation are from multiplication, which is very well-conditioned. By contrast, $f(x)$ sums up 5 terms with alternating signs. Propagated round-off errors from addition are very substantial because of *loss of significance*. Note that at $x = 0$ there should be exact cancellation $f(0) = 1 - 8 + 28 - 56 + 35 = 64 - 64 = 0$ and near-cancellation for small values of x near $x = 0$.

(e) Using the code at the beginning of the problem, the computed minimum for $g(x)$ is 0, so the problems are eliminated. This is because $g(x)$ uses only multiplication, which exactly preserves positivity of products of positive numbers and which is also very well-conditioned.

Problem 5. (a) If you multiple the second row by α and add it to the first row, then you get

$$(1-\alpha^2)x + 0 = 1$$
$$x = \frac{1}{1-\alpha^2}$$

Therefore using the second row,

$$y = \frac{\alpha x}{2} = \frac{\alpha}{2(1-\alpha^2)}$$

Therefore

$$K_x(\alpha) := \frac{\alpha x'(\alpha)}{x(\alpha)} = \frac{2\alpha^2/(1-\alpha^2)^2}{1/(1-\alpha^2)} = \frac{2\alpha^2}{1-\alpha^2},$$

and

$$K_y(\alpha) := \frac{\alpha y'(\alpha)}{y(\alpha)} = \frac{2\alpha(\alpha^2+1)/[2(1-\alpha^2)]^2}{\alpha/2(1-\alpha^2)} = \frac{\alpha^2+1}{1-\alpha^2}.$$

By inspection it's evident that as $\alpha \to 1$, $K_x(\alpha), K_y(\alpha) \to \pm\infty$.

(b) If you subtract the second row of the system from the first, you get

$$(1+\alpha)x - (1+\alpha)2y = 1$$
$$(1+\alpha)(x-2y) = 1$$

Therefore since $z = x - 2y$, then

$$z = \frac{1}{1+\alpha}$$

Therefore

$$K_z(\alpha) = \frac{\alpha z'(\alpha)}{z(\alpha)} = \frac{-\alpha/(1+\alpha)^2}{1/(1+\alpha)} = -\frac{\alpha}{1+\alpha},$$

and as $\alpha \to 1$, $K(z) \to -1/2$.

(c) In `MATLAB`:

```
al=1-1e-8;
A=[1 -2*al; -al 2];
b=[1; 0];
x=A\b;
z1=x(1)-2*x(2)

z2=1/(1+al)

relerr=z1/z2-1

Kx=2*al^2/(1-al^2)
errest=Kx*eps
```

The method by addition estimates $z \approx 0.5$ and the method by direct computation estimates $z \approx 0.5000000025$. The relative difference is $-4.999999969612645 \cdot 10^{-9}$, much bigger than double-precision error. The direct computation should be more accurate because $\alpha = 0.99999999$ is very close to 1, where $K_x(\alpha), K_y(\alpha)$ become big. For example, at $\alpha = 0.99999999$ we have $K_x(\alpha) = 9.999999805263558 \times 10^7$. Based on that, we would estimate the error to be $2.220446006010137 \times 10^{-8}$. Thus, inaccuracy in approximations to x and y carry over to inaccuracy in the approximate value of z.

Problem 6. (a) Note that

$$\frac{2n+1}{n^2(n+1)^2} = \frac{1}{n} - \frac{1}{(n+1)^2}$$

Therefore we can get the formula for S_k:

$$S_k = \sum_{n=1}^{k} \frac{2n+1}{n^2(n+1)^2} = \sum_{n=1}^{k} \left[\frac{1}{n} - \frac{1}{(n+1)^2}\right] = 1 - \frac{1}{(k+1)^2}$$

Note then that

$$S = \lim_{k\to\infty} S_k = \lim_{k\to\infty} (1 - \frac{1}{(k+1)^2}) = 1$$

We also can find a formula for the remainder:

$$R_k = 1 - S_k = 1 - (1 - \frac{1}{(k+1)^2}) = \frac{1}{(k+1)^2}$$

(b) We find that the algorithm terminates when n reaches 330281, with $S_{330281} = 0.999999999992635$ and relative error $7.365108523060826 \cdot 10^{-12}$. The exact remainder R_{330281} is $9.167062418193335 \cdot 10^{-12}$, which is the same order of magnitude as the relative error.

(c) Double precision was not obtained in (b). This is because the sum was terminated for $n = 330281$, when the remainder was over 10,000 times larger than 10^{-16} ($\frac{7.3651 \cdot 10^{-12}}{10^{-16}} = 73651$).
This happened because the next term $\frac{2n+1}{n^2(n+1)^2}$ in the sum with $n = 330282$ is only $5.551076 \cdot 10^{-17}$, which is smaller than the unit round in Matlab. Hence, adding this tiny number to $S_{330281} \doteq 1$ did not lead to an improved value of S_{330282} so that the while-loop terminated. To avoid this problem, one should always sum numbers in floating-point arithmetic from small terms to large ones, not from large to small. To decide how large an n must be considered, we use the remainder formula to find an n so that $R_n = 1/(n+1)^2 \doteq eps$ and thus find that $n = 67108864 \doteq 1/\sqrt{eps}$. Below is a MATLAB script that implements an improved algorithm:

```
NN=round(1/sqrt(eps))
nn=NN;
RR=1/(nn+1)^2
dS=(2*nn+1)/nn^2/(nn+1)^2;
SS=dS;
for nn=NN-1:-1:1
    dS=(2*nn+1)/nn^2/(nn+1)^2;
    SS=SS+dS;
end
SS=SS
relerr=abs(SS-1)
```

Running this script yields $S_{67108864} = 1.0000000000000000$ with a relative error $2.220445983075866 \cdot 10^{-16}$, which is now accurate to double precision.