Math 551 **Introduction to Scientific Computing** **Summer I 2019**

Solutions Homework Set 2

1. Find $P_2(x)$ for $f(x) = e^x \cos x$ expanded about $x_0 = 0$. Then find a bound on the error $|f(x) - P_2(x)|$ in using P_2 to approximate f on $[0, 1]$.

   ANS: First note that $f'(x) = e^x(\cos x - \sin x)$, $f''(x) = -2e^x \sin x$, and $f'''(x) = -2e^x(\cos x + \sin x)$.

   We have for $x_0 = 0$:

$$\begin{aligned} P_2(x) &= f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 \\ &= 1 + 1(x - 0) + \frac{0}{2}(x - 0)^2 \\ &= \mathbf{1 + x} \end{aligned}$$

   To bound the error in using $P_2(x)$ to approximate $f(x)$ over the interval $[0, 1]$, from Taylor's theorem we have

$$\begin{aligned} |f(x) - P_2(x)| &= \left| \frac{f^{(3)}(\xi_x)}{3!}(x - 0)^3 \right| \quad \text{for } \xi_x \in (0, 1) \\ &= \left| \frac{-2e^{\xi_x}(\cos(\xi_x) + \sin(\xi_x))}{3!}(x - 0)^3 \right| \quad \text{for } \xi_x \in (0, 1) \\ &\leq \frac{\left|2e^{\xi_x}(\cos(\xi_x) + \sin(\xi_x))\right|}{3!} \quad \text{for } \xi_x \in (0, 1) \end{aligned}$$

   since x^3 is maximized at $x = 1$. We are left to find an upper bound for the expression in the numerator, i.e., we have to find the maximum absolute value of $g(x) = 2e^x(\cos x + \sin x)$ over the interval $[0, 1]$. From Calculus we know this can only occur at the endpoints of the interval, $x = 0, 1$, or in $(0, 1)$ where $g'(x) = 4e^x \cos x = 0$. But $4e^x > 0$ and there is no point in $[0, 1]$ where $\cos x = 0$ (check this, if need be, by drawing a picture!).

   So checking $x = 0, 1$ (the endpoints of the interval) we have

$$|g(0)| = |2e^0(\cos(0) + \sin(0)| = 2,$$

   and

$$|g(1)| = |2e^1(\cos(1) + \sin(1)| \approx 7.512098454189456.$$

   Using the larger of these one can conclude that

$$\max_{x \in [0,1]} |f(x) - P_2(x)| \leq \frac{7.512098454189456}{6} \approx 1.252016409031576.$$

2. The floating point representation of a number is $x = \pm(0.a_1a_2\ldots a_n)_\beta \times \beta^e$, where $a_1 \neq 0$, $-M \leq e \leq M$. Suppose $\beta = 2$, $n = 8$, and $M = 4$.

   (a) Find the smallest positive (x_{min}) and largest positive (x_{max}) floating point numbers that can be represented. Give the answers in decimal form (base 10).

   (b) Find the floating point number in this system that is closest to π.

   ANS: For (a) we have

$$x_{min} = +(0.10000000)_2 \times 2^{-4} = (2^{-1} \times 2^{-4})_{10} = (2^{-5})_{10} = \frac{1}{32} = 0.03125\,,$$

   and

$$x_{max} = +(0.11111111)_2 \times 2^4 = (\frac{1}{2} + \frac{1}{4} + \cdots + \frac{1}{256})_{10} \times 2^4 = \frac{255}{256} \times 2^4 = \frac{255}{16} = 15.9375\,.$$

   For (b), the closest number to π in this system, x_π is

$$x_\pi = +(0.11001001)_2 \times 2^2 = (\frac{1}{2} + \frac{1}{4} + \frac{1}{32} + \frac{1}{256})_{10} \times 2^2 = 3.140625\,.$$

3. Find the two roots of $x^2 - 50x + 1 = 0$, performing all calculations in 5 decimal digit arithmetic (i.e. $\beta = 10$, $t = 5$). Thus, round the answer of each arithmetic operation to 5 significant digits.

noindent**ANS:** First apply the quadratic formula, rounding all computations to 5 significant digits,

$$x_{1,2} = \frac{50 \pm \sqrt{2500 - 4}}{2} = \frac{50 \pm 49.960}{2} = 49.980, 0.020000\,,$$

and we see that the root $x_2 = 0.02000$ has only **1** significant digit of accuracy. This was due to the subtraction of two nearly equal numbers. Let's reformulate the formula for this root as

$$\frac{-b - \sqrt{b^2 - 4ac}}{2a} = \frac{-b - \sqrt{b^2 - 4ac}}{2a} * \frac{-b - \sqrt{b^2 - 4ac}}{-b - \sqrt{b^2 - 4ac}} = \frac{2c}{-b + \sqrt{b^2 - 4ac}}\,.$$

Using this formula for x_2, again rounding all computaions to 5 significant digits, gives

$$x_2 = \frac{2}{50 + \sqrt{2496}} = \frac{2}{50 + 49.960} = \frac{2}{99.960} = 0.020008\,,$$

accurate to **5** digits! Note using MATLAB we have

```
>> roots([1 -50 1])

ans =

          49.9799919935936
         0.0200080064064072
```

4. Recall that the *machine epsilon* of a computer is the smallest positive floating point number *eps* such that $fl(1+eps) > 1$. We can determine *eps* on a given machine, for a given floating point precision, by evaluating the expression

$$(1 + x) - 1 \qquad (*)$$

for decreasing values of x. The smallest representable positive x for which $(*)$ is nonzero is *eps*. On a binary machine it is enough to consider the sequence $x_n = 2^{-n}$ for $n = 1, 2, \ldots$ (Why?).

Write a MATLAB code to determine *eps* on the machine you are using, and compare it with the value of *eps* in MATLAB (type 'eps' in MATLAB to see this value). What is the relationship between the two. (Note: you may find it useful to first issue the MATLAB command '*format long e*' so that you are sure of when an expression computes identically to 0). Include a copy of your code.

ANS: Here is the code:

```
function [y,n]=myeps
%
% MYEPS determines the machine epsilon.  It returns the
%       value and the corresponding power of 2.
%
n=0; x=1;
while ((1+x) > 1)
  n=n-1;   x=x/2;
end
y=2*x;  % while loop exited with x=eps/2, so recover proper
n=n+1;  % value, and set corresponding power of 2
```

Executing this function gives

```
>> [y,n]=myeps

y =

      2.22044604925031e-16


n =

   -52
```

showing the $y = 2^{-52}$. Further, we see

```
>> eps-y

ans =

     0
```

and we see that the ϵ_M we found (returned as $\mathbf{y}$) is equal to MATLAB's *eps*.

5. Consider evaluating the integrals

$$y_n = \int_0^1 \frac{x^n}{x+10}\, dx$$

for $n = 1, 2, \ldots, 30$.

(a) Show analytically that $y_n + 10y_{n-1} = 1/n$.

(b) Show that $y_0 = \log 11 - \log 10$ and then use it with the recursion

$$y_n = \frac{1}{n} - 10y_{n-1}$$

to numerically generate y_1 through y_{30}

(c) Show for $n \geq 0$ that $0 \leq y_n \leq 1$, and discuss the results in (b) in light of this.

ANS: For (a) we have

$$y_n + 10y_{n-1} = \int_0^1 \frac{x^n}{x+10}dx + 10\int_0^1 \frac{x^{n-1}}{x+10}dx = \int_0^1 x^{n-1}dx = \frac{1}{n}\,,$$

and (b)

$$y_0 = \int_0^1 \frac{x^0}{x+10}dx = \int_0^1 \frac{1}{x+10}dx = \ln 11 - \ln 10\,.$$

Below is the MATLAB code and $y_n, n = 0, \ldots, 30$.

```
y = log(11)-log(10);
disp(' ')
disp('n   y_n ')
disp('----------');
for n = 1:30
    y = 1/n - 10*y;
    disp([num2str(n),' ',num2str(y)])
end
```

```
n   y_n
----------
1 0.046898
2 0.031018
3 0.023154
4 0.018465
5 0.015353
6 0.013138
7 0.011481
8 0.010194
9 0.0091673
10 0.008327
11 0.0076386
12 0.0069473
13 0.0074503
```

```
14 -0.0030745
15 0.097411
16 -0.91161
17 9.175
18 -91.694
19 916.9927
20 -9169.8773
21 91698.8211
22 -916988.1656
23 9169881.6991
24 -91698816.9496
25 916988169.5363
26 -9169881695.325
27 91698816953.2869
28 -916988169532.8334
29 9169881695328.369
30 -91698816953283.66
```

As for (c), note that $0 \leq x^n/(x+10) \leq 1$, hence y_n is bounded by $(1-0)*1 = 1$. However the iterates start to diverge/blow_up around the 16th iterate and begin to grow rapidly. Why? Note that in the recursion formula $y_n = \frac{1}{n} - 10y_{n-1}$ the previous iterate y_{n-1} is multiplied by 10, so any error in that iterate is **amplified** by an order of magnitude. The true limit is 0.