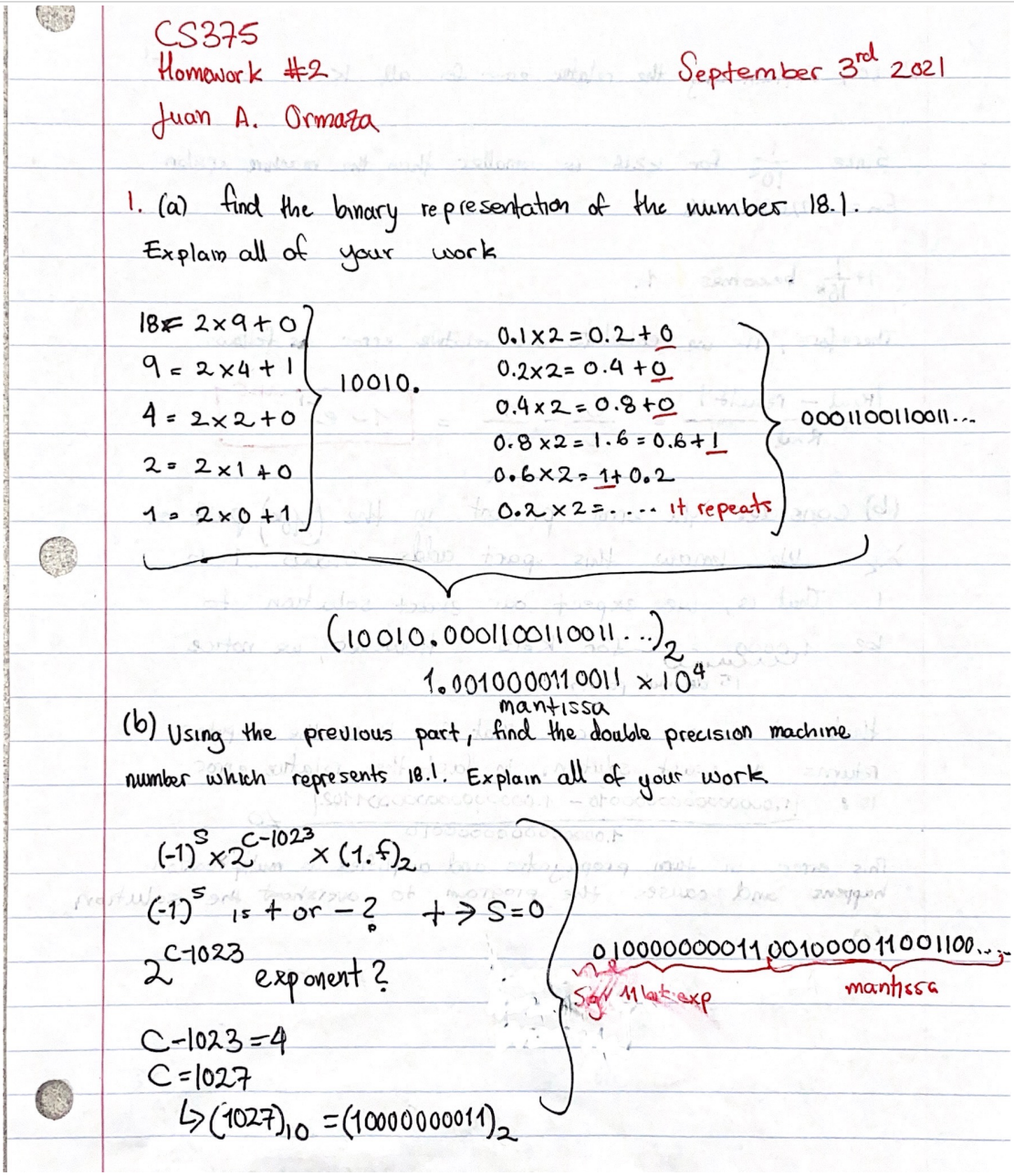
Contents

- HomeWork 2
- problem 2.b
- problem 3.a
- problem 3.b
- problem 3.c
- problem 4.a
- problem 4.b
- problem 4.c
- problem 4.d

HomeWork 2

Juan Alejandro Ormaza Sep 7 2020 CS375

```
clear all; clc;
format long e
```



2. (a) Explain why the relative error for all $k \geq 16$ is $1 - e^{-1}$

Since $\frac{1}{10^k}$ for $k \geq 16$ is smaller than the machine epsilon

$\varepsilon_m = 2.2204 e-16$

$1 + \frac{1}{10^k}$ becomes 1.

Therefore, if we calculate our relative error as follows...

$$\frac{|\text{Real} - \text{result}|}{\text{Real}} = \frac{e}{e} - \frac{1}{e} = \boxed{1 - e^{-1}} \quad \text{ans}$$

(b) Consider the error present in the $\left(\frac{1}{10^k}\right)$ part of X_k We know this part adds 0.0000....1 to 1. That is, we expect our exact solution to be $1.\underbrace{0000\ldots1}_{15 \text{ decimal points}}$ for $k=15$. However, we notice that it is not the case that for $k=15$ the computer returns an exact solution. In fact the relative error is: $\frac{|1.000000000000010 - 1.000000000000001102|}{1.000000000000010} \neq 0$

This error in turn propagates and amplifies as multiplication happens and causes the program to overshoot the solution.

problem 2.b

```
x=12:1:15;
x_k=num2str(1+(1./10.^(x)),25);
fprintf(x_k); % this line prints the x_k values for k=12 to k=15
```

```
1.0000000000001000088900582      1.0000000000000099920072216      1.0000000000000009992007222      1.0000000000000001110223025
```

3. (a) if $f(x)=\sin(x)$ let the taylor series expansion be

$$\sin(x)=x-\frac{x^3}{3!}+\frac{x^5}{5!}-\frac{x^7}{7!}+\dots$$

to make sure our error is less than 2×10^{-8} for $x\in[0,\pi/2]$ we use the taylor approximation error of $E_n(x)=\frac{x-c}{(n+1)!}f^{n+1}(\xi)$

The expansion for our function around $c=0$ (otherwise known as Mclaurin series) is:

$$E_n(x)=x-\frac{x^3}{3!}+\frac{x^5}{5!}-\frac{x^7}{7!}\dots=\sum_{n=0}^{\infty}(-1)^n\frac{x^{2n+1}}{(2n+1)!}$$

Since $x=1\in[0,\pi/2]$ $\rightarrow 1-\frac{1}{3!}+\frac{1}{5!}-\frac{1}{7!}+\dots$

we would like to find the error at $x=1$ so that it is smaller than 2×10^{-8}

$$e_{x=1}=\left|(-1)^n\frac{(1)^{2n+1}}{(2n+1)!}\right|<2\times10^{-8}\left\{\begin{array}{l}\text{alternating series theorem}\\ \frac{1}{(2n+1)!}<2\times10^{-8}\end{array}\right\}\log(2n+1)!>-\log2+8 \quad (1)$$

We now use matlab in order to iterate n for different values until the condition listed above is not satisfied at which point we can conclude that we found an n for which our error is acceptable.

We found $n=5$

(b) & (c) matlab

problem 3.a

```
sinErrorExpansion=@(n) 1/(factorial(2*n+1));
n=1;

while sinErrorExpansion(n)>2e-8
    n=n+1;
end

n=n-1; %we take one from n because we know that at the last loop of the while
% loop the condition for n was not met.
n
```

```
n = 5
```

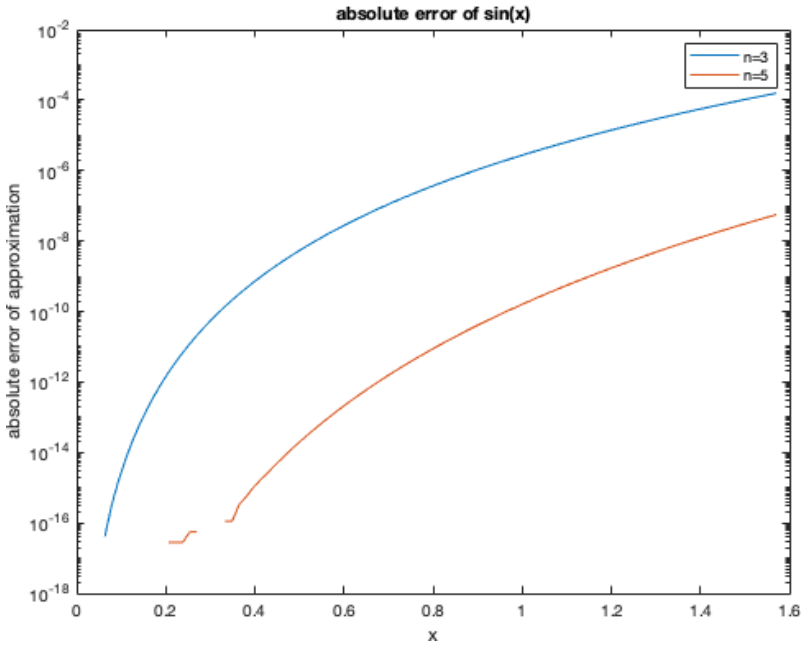
problem 3.b

```
test=my_sin(1,5) % this is a test to find if my_sin works
test2=sin(1) % and it appears it did, since the answer resembles
% the built-in sin(x) function
```

```
test =

      8.414709846480680e-01


test2 =
      8.414709848078965e-01
```

problem 3.c

```
x=linspace(0,pi/2,100); % creates the x array for the domain of the function.
error1 = abs(sin(x)-my_sin(x,3));
error2 = abs(sin(x)-my_sin(x,5));

figure();
semilogy(x,error1,x,error2);
xlabel('x');
ylabel('absolute error of approximation');
title('absolute error of sin(x)');
legend('n=3','n=5')

% from the figure we see that the error is smaller than 2e-8 (the bound)
% for n=5 (my result from 3.a). For n=3, the situation is different as we
% find the error is larger and actually exceeds 2e-8 after x~=0.5.
```



4. Consider $f(x) = ax^2 + bx + c$

Suppose we are trying to solve this equation for $a = 0.5$, $b = 1000$, $c = 5 \times 10^{-7}$

the quadratic formula says that one of the roots is.

$$x = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

4.a

notice that $2a = 1$ with $a = 0.5$

$\therefore$ we can write x as follows:

$$x = -b + \sqrt{b^2 - 4ac}$$

Since a and c are small notice that

$b \approx \sqrt{b^2 - 4ac}$

Therefore, for our loss of precision theorem we take $x = 1000 = b$ and $y = \sqrt{b^2 - 4ac}$

$\therefore$

$$10^{-13} < 1 - \frac{y}{x} < 10^{-12}$$

$q = 12$ $p = 13$ } which satisfies $q < p$

4.d

$$x = \frac{(-b + \sqrt{b^2 - 4ac})}{2a} \cdot \frac{+b \cdot + \sqrt{b^2 - 4ac}}{b + \sqrt{b^2 - 4ac}}$$

$$x = \frac{-b^2 + b^2 - 4ac}{2a(b + \sqrt{b^2 - 4a})} = \frac{-4ac}{2a(b + \sqrt{b^2 - 4a})}$$

$$\boxed{x = \frac{-2c}{b + \sqrt{b^2 - 4a}}} \rightarrow \text{Now to matlab.}$$

problem 4.a

```
right_side=@(a,b,c) sqrt(b^2 - 4*a*c);

y=right_side(0.5,1000,5e-7);
x=1000; %% this is the same as b

%%%from the lost of precision theorem we get.
lostOfPrecision=1-y/x
```

```
lostOfPrecision =

     5.000444502911705e-13
```

problem 4.b

```
%%%here we compute the "true" roots using the matlab built-in functions.
poli = [0.5 1000 5e-7];
realRoots = roots(poli)
```

```
realRoots =

     -1.999999999999500e+03
     -5.000000000001249e-10
```

problem 4.c

```
rootFunction=@(a,b,c) -b + sqrt(b^2 - 4*a*c);
myResult = rootFunction(0.5,1000,5e-7)
%%%the result of using equation is innacurate because of loss of precision.
%%%However, this is to be expected as we lost 12 to 13 bits of precision
%%%according to part a of this problem.
```

```
myResult =

     -4.999947122996673e-10
```

problem 4.d

```
newRootFunction=@(a,b,c) -2*c/(b + sqrt(b^2 - 4*a*c));
newRootFunction(0.5,1000,5e-7)

%%%by modifying the formulation for our solution we get to a more
% "accurate" solution that more closely resembles the result given by the
% roots function. In fact, it is the same result up to the 16 decimal
% point.
```

```
ans =

     -5.000000000001249e-10
```

Published with MATLAB® R2021a