Damian Franco

CS 375 – 001

Homework #2:

1) The floating-point representation of 100.2 is…

   $100 - (1100100)_2$
   $0.2 - (\overline{0011})_2$
   $100.2 = (1100100.\overline{0011})_2$

   $V = (-1)^2(1.b_1\ldots.b_{52}) * 2^{F-1023}$

   True exponent = 6
   F = 1029
   $= (1027)_{10} + (2)_{10}$
   $= (10000000011)_2 + (10)_2$
   $= (10000000101)_2$

   fl(100.2) = (1.001 0001 1001 1001 1001 1001 1001
   1001 1001 1001 1001 1001 1001 1) * 2^6

   Hex format: 0x40590ccccccccccd

   $\text{decRep} = (1.0010\,\overline{0011})_2 * 2^{-53} * 2^6$

   $= (1.0010\,\overline{0011})_2 * 2^{-47}$

   $= (0.\overline{0011})_2 * 2^{-43}$

   $= (0.2)_{10} * 2^{-43}$

   - $2^{-52} * 2^6 = 2^{-46}$

   $fl(100.2) = 100.2 - 0.2 * 2^{-43} + 2^{-46}$

   $= 100.2 + 0.1 * 2^{-46}$

   - $\varepsilon_{mach} = 2^{-52}$

   $|fl(100.2) - 100.2| / |100.2| < \varepsilon_{mach}$

   $|100.2 + 0.1 * 2^{-46} - 100.2| / 100.2 < \varepsilon_{mach}$

$|0.1 * 2^{-46}| / 100.2 < \varepsilon_{mach}$

$11 / 10020 * 2^{-46} < 2^{-52}$

$1.56 * 10^{-17} < 2.22 * 10^{-17}$ ✔

Finding the floating-point representation is very important because that can give us some insight on how much of and error or accurate our computing mathematical operations could be. The first thing when converting a decimal value to do is to convert both that whole number and decimal number to binary. Once we have them in binary, the next step is to put those two binary strings together and plug it in the V equation to find the floating point. From there the floating point should be relatively easy to find. When calculation whether fl(100.2) is less than the machine epsilon, there must be some steps taken there to get the true value of fl(100.2). Once the value of fl(100.2) is found, then when compared to 2^-52, the value for the floating-point representation is smaller than the value of the machine epsilon.

2) Compute the roots of $x^2 + 201x - 10^{-12} = 0$

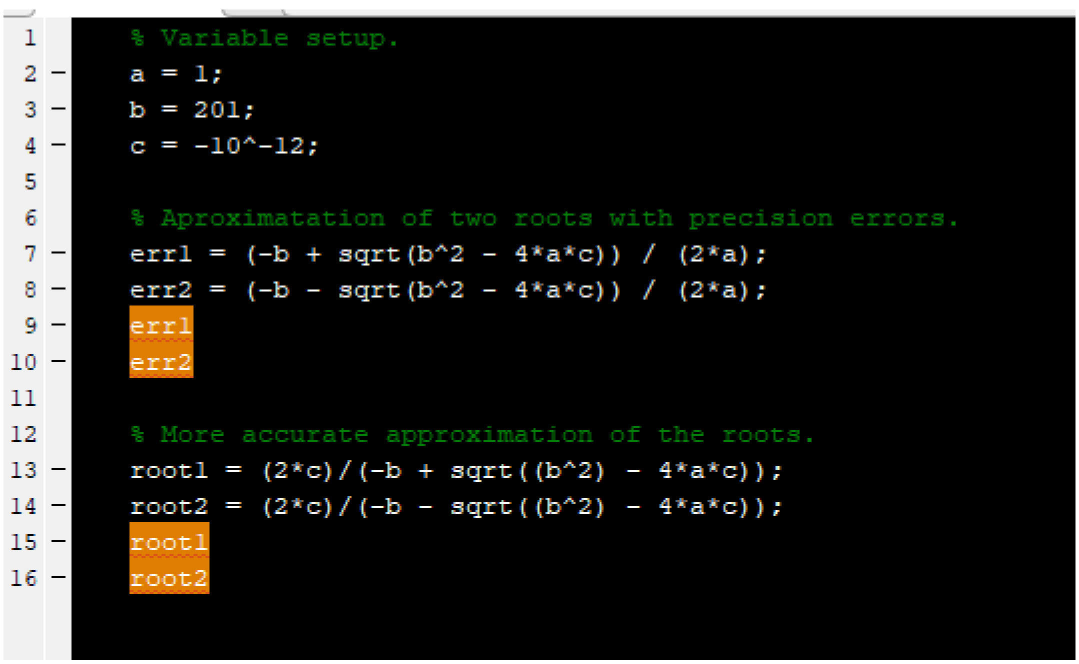
The most accurate way to compute the two roots of the equation above is to use the "Citardauq Formula" which is a form of the quadratic formula that will not suffer serious loss of precision. Floating point precision will give slightly wrong results with the quadratic formula.

Source Code (*HW2_2.m*):

```
% Variable setup.
a = 1;
b = 201;
c = -10^-12;

% Aproximatation of two roots with precision errors.
err1 = (-b + sqrt(b^2 - 4*a*c)) / (2*a);
err2 = (-b - sqrt(b^2 - 4*a*c)) / (2*a);
err1
err2

% More accurate approximation of the roots.
root1 = (2*c)/(-b + sqrt((b^2) - 4*a*c));
root2 = (2*c)/(-b - sqrt((b^2) - 4*a*c));
root1
root2
```



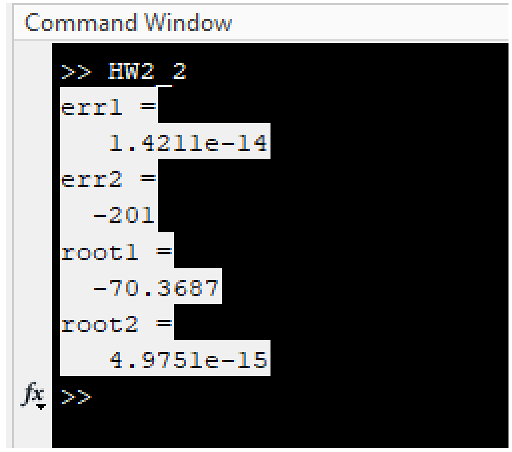
```
% Variable setup.
a = 1;
b = 201;
c = -10^-12;

% Aproximatation of two roots with precision errors.
err1 = (-b + sqrt(b^2 - 4*a*c)) / (2*a);
err2 = (-b - sqrt(b^2 - 4*a*c)) / (2*a);
err1
err2

% More accurate approximation of the roots.
root1 = (2*c)/(-b + sqrt((b^2) - 4*a*c));
root2 = (2*c)/(-b - sqrt((b^2) - 4*a*c));
root1
root2
```

Output from program:

```
% Roots with errors.
err1 =
  1.4211e-14
err2 =
  -201
% Roots without errors.
root1 =
  -70.3687
root2 =
  4.9751e-15
```



3) For problem #3, the main point for this problem is to find out precision of certain equations compared to other. We were tasked with creating a function that takes in a scale for the zeroth order cylindrical Bessel function. The function should output a vector of values that can be related to the accuracy of that function call, then we had to compare them to MATLAB built in Bessel function. In my finding, I noticed that my numbers are all over the place and frankly does not look too correct so I am hoping for some partial credit on this one. But regardless, I definitely took notice on how some function and equations have a huge impact on the precision and accuracy which could be good or bad impact.

3a)

Source code (*ApproxBesselJ0*.m)

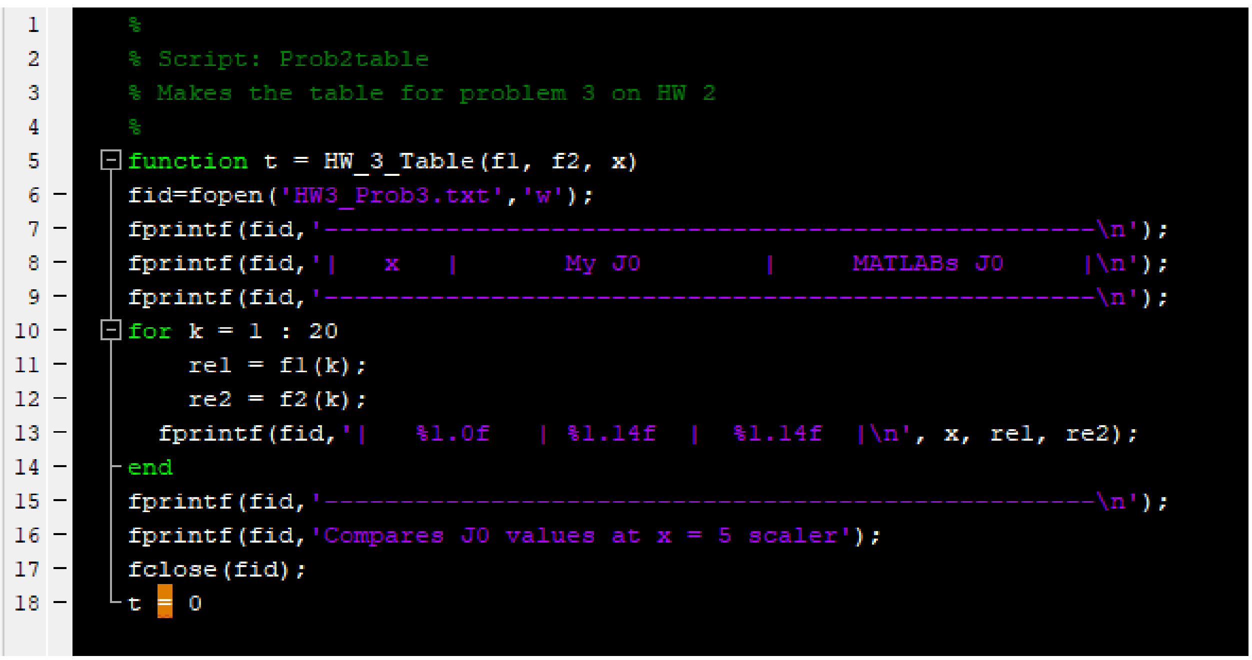
```
%
% Function: ApproxBesselJ0(x)
% This function will approximate the based of the cylindrical
% Bessel function Maclaurin series.
%
function J0 = ApproxBesselJ0(x)
out = [];
N = 1e-8;
term = 0;
J0 = term;
k = 0
while(20 > k)
    term = (-1)^k * (1/factorial(k))^2 * (x/2)^(2*k);
    J0 = term + J0;
    out = [out, J0];
    k = k + 1;
end
J0 = out
```

```
%
% Function: ApproxBesselJ0(x)
% This function will approximate the based of the cylindrical
% Bessel function Maclaurin series.
%
function J0 = ApproxBesselJ0(x)
out = [];
N = 1e-8;
term = 0;
J0 = term;
k = 0
while(20 > k)

    term = (-1)^k * (1/factorial(k))^2 * (x/2)^(2*k);
    J0 = term + J0;
    out = [out, J0];
    k = k + 1;
end
J0 = out
```

3b) Source code (HW_3_Table.m)

```
%
% Script: Prob2table
% Makes the table for problem 3 on HW 2
%
function t = HW_3_Table(f1, f2, x)
fid=fopen('HW3_Prob3.txt','w');
fprintf(fid,'---------------------------------------------------\n');
fprintf(fid,'|   x   |       My J0         |     MATLABs J0     |\n');
fprintf(fid,'---------------------------------------------------\n');
for k = 1 : 20
    re1 = f1(k);
    re2 = f2(k);
  fprintf(fid,'|   %1.0f    | %1.14f  |  %1.14f  |\n', x, re1, re2);
end
fprintf(fid,'---------------------------------------------------\n');
fprintf(fid,'Compares J0 values at x = 5 scaler');
fclose(fid);
t=0
```



```
%
% Script: Prob2table
% Makes the table for problem 3 on HW 2
%
function t = HW_3_Table(f1, f2, x)
fid=fopen('HW3_Prob3.txt','w');
fprintf(fid,'---------------------------------------------------\n');
fprintf(fid,'|   x   |       My J0         |     MATLABs J0     |\n');
fprintf(fid,'---------------------------------------------------\n');
for k = 1 : 20
    re1 = f1(k);
    re2 = f2(k);
  fprintf(fid,'|   %1.0f    | %1.14f  |  %1.14f  |\n', x, re1, re2);
end
fprintf(fid,'---------------------------------------------------\n');
fprintf(fid,'Compares J0 values at x = 5 scaler');
fclose(fid);
t = 0
```

Text (table) files:

- Scaler Multiplier = 5 (HW3_Prob3_5.txt)

HW3_Prob3_5.txt - Notepad

File Edit Format View Help

x	My J0	MATLABs J0
5	1.00000000000000	1.00000000000000
5	-5.25000000000000	0.00000000000000
5	4.51562500000000	0.00000000000000
5	-2.26605902777778	0.00000000000000
5	0.38303629557292	0.00000000000000
5	-0.27923753526476	0.99750156206604
5	-0.16425943963322	0.00000000000000
5	-0.17892501305561	0.00000000000000
5	-0.17749282815108	0.00000000000000
5	-0.17760333624556	0.00000000000000
5	-0.17759642948966	0.99002497223958
5	-0.17759678624358	0.00000000000000
5	-0.17759677075947	0.00000000000000
5	-0.17759677133210	0.00000000000000
5	-0.17759677131384	0.00000000000000
5	-0.17759677131435	0.97762624653830
5	-0.17759677131434	0.00000000000000
5	-0.17759677131434	0.00000000000000
5	-0.17759677131434	0.00000000000000
5	-0.17759677131434	0.00000000000000

Compares J0 values at x = 5 scaler.

- Scaler Multiplier = 20 (HW3_Prob3_20)

HW3_Prob3_20.txt - Notepad

File Edit Format View Help

x	My J0	MATLABs J0
20	1.00000000000000	1.00000000000000
20	-99.00000000000000	0.00000000000000
20	2401.00000000000000	0.00000000000000
20	-25376.77777777777737	0.00000000000000
20	148234.33333333331393	0.00000000000000
20	-546210.11111111124046	0.99750156206604
20	1382802.23456790111959	0.00000000000000
20	-2553957.65457294043154	0.00000000000000
20	3597229.67220962513238	0.00000000000000
20	-3996828.75591699872166	0.00000000000000
20	3597229.67220962326974	0.99002497223958
20	-2678851.67334957048297	0.00000000000000
20	1679538.14995542541146	0.00000000000000
20	-899390.73957415716723	0.00000000000000
20	416389.30610420135781	0.00000000000000
20	-168401.82530840253457	0.97762624653830
20	60032.21039964584634	0.00000000000000
20	-19010.70853047468700	0.00000000000000
20	5385.25410227856264	0.00000000000000
20	-1372.63028352566289	0.00000000000000

Compares J0 values at x = 20 scaler.

- Scaler Multiplier = 40 (HW3_Prob3_40)

HW3_Prob3_40.txt - Notepad

File Edit Format View Help

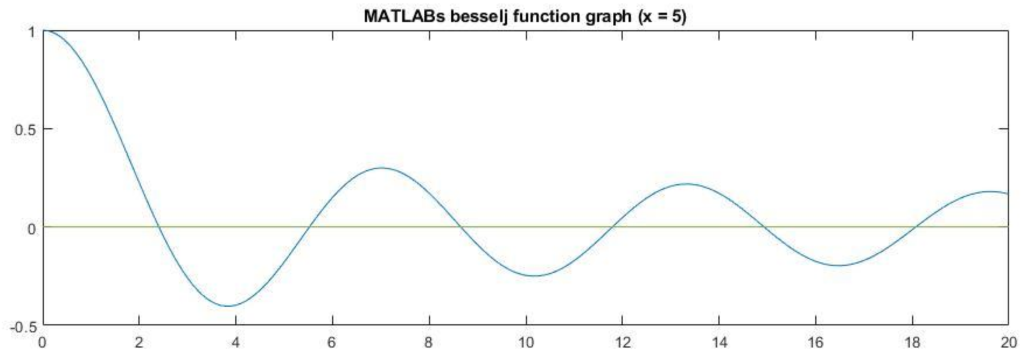
x	My J0	MATLABs J0
40	1.00000000000000	1.00000000000000
40	-399.00000000000000	0.00000000000000
40	39601.00000000000000	0.00000000000000
40	-1738176.77777777775191	0.00000000000000
40	42706267.66666666418314	0.00000000000000
40	-668404843.44444453716278	0.99750156206604
40	7232829724.45678997039795	0.00000000000000
40	-57267044299.22676086425781	0.00000000000000
40	345857168348.79547119140625	0.00000000000000
40	-1644879684234.030273437500	0.00000000000000
40	6318067726097.2705078125000	0.99002497223958
40	-20005725365907.03906250000	0.00000000000000
40	53115922111882.710937500000	0.00000000000000
40	-119953066001229.0625000000	0.00000000000000
40	233248950556141.93750000000	0.00000000000000
40	-394665745545851.06250000	0.97762624653830
40	586450967113513.0000000000	0.00000000000000
40	-771496040027475.000000000	0.00000000000000
40	904981746566337.2500000000	0.00000000000000
40	-952611368773067.250000000	0.00000000000000

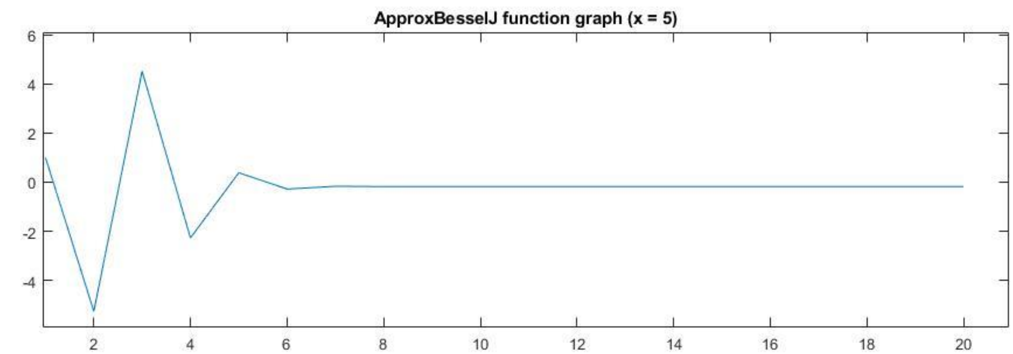
Compares J0 values at x = 40 scaler.

3c) We can see that the loss of precision is very prevalent when using some function and equations in numerical computing in part (b). The values were vastly different and we can see that the effective ness of some equations and functions is much better than others.

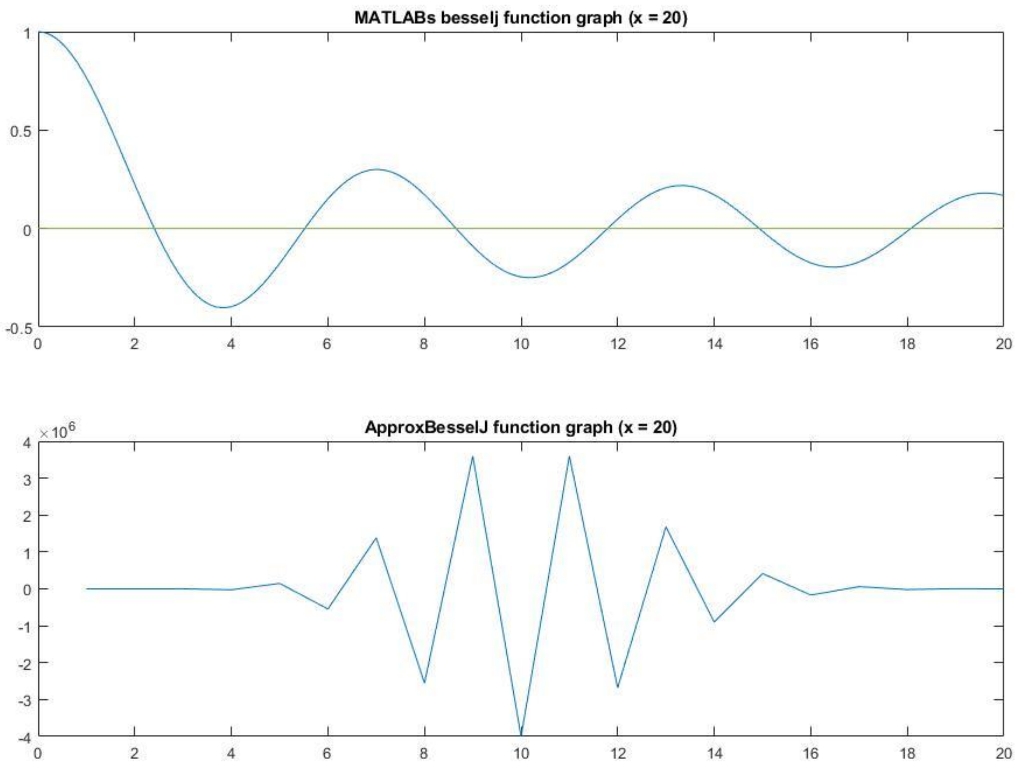
Graphs of findings:

- For scaler = 5 (*Prob3C_5*.m)





- For scaler = 5 (*Prob3C_20*.m)



- For scaler = 5 (*Prob3C_5*.m)