The Written section is supported by the Commentary section, which in turn, is supported by the MATLAB Code section.

HW #3

Math 375 -02/15/20

Michael John Tanguay, Mathew Gervasi

Commentary

1a.) The current arrangement is the best arrangement (for those listed) in the prevention of the lost precision for values abs(x) << 1 and values of abs(x) < 1; There are some fluctuations most likely due to rounding and truncation error. The MATLAB attached suggests that the value closest to zero with the most significant digits obtained by the true value (use of Taylor approximations) subtracted by that of the calculated arrangement yields the least precision lost. There is a plethora of ways to manipulate and therefore could exist an arrangement higher in precision.

1b.) Arrangement a.) is the best arrangement (for those listed) in the prevention of the lost precision for values, x to exist in [1,100]. The MATLAB attached suggests that the value closest to zero with the most significant digits obtained by the true value (use of rationalization) subtracted by that of the calculated arrangement yields the least precision lost. There is a plethora of ways to manipulate and therefore could exist an arrangement higher in precision.

2b)

TABLE 1.1 – Derivative value of -0.415224913494810

h-value	***Derivative approx***	***Abs Error***	***P_k***
0.5	-0.967741935483871	0.552517021989061	0
0.1	-0.438340151957919	0.023115238463109	1.972112216109422
0.05	-0.420871730571947	0.005646817077138	2.033334443327634
0.025	-0.416628466682471	0.001403553187662	2.008354161617389
0.0125	-0.415575293954080	0.000350380459270	2.002089512684769

The value of P_k approaches 2. Based on the theory discussed in class, the error should decrease by a fourth for h to be cut in half. In the second and third row 0.0231152/4 = 0.00577881. As one can see, it is not exactly a fourth, However, as h approaches 0, the factor of a fourth becomes more exact.

2c.)

TABLE 1.1 – Derivative value of -0.415224913494810

h-value	***Derivative approx***	***Abs Error***	***P_k***
0.5	-0.261872890782602	0.153352022712208	0
0.1	-0.415048923443290	0.000175990051520	4.206477205075599
0.05	-0.415214045385979	0.000010868108830	4.017321054928710
0.025	-0.415224236377949	0.000000677116861	4.004552263699980
0.0125	0	0	0

The value of P_k approaches 4. Based on the theory discussed in class, the error should decrease by a 16^{th} for h to be cut in half. In the second and third row 0.000175990051520/0.000010868108830 = 16.193254435785772. As one can see, it is not exactly a 16^{th}, However, as h approaches 0, the factor of a 16^{th} becomes more exact.

MATLAB Code

1.)

Functions

```
function y = ApproxExp(x,n);
% Output parameter:  y (nth order Taylor approximation of e^x)
% Input parameters:  x (scalar)
%                    n (integer)
sum = 1;
for k=1:n
    sum = sum + x.^k./factorial(k);
end
y = sum;
```

Input

```
%Cancellation
clear, clc, close all
format long
x = [0.1, 0.01, 0.001, 0.0001, 0.00001, 0.000001];
e = ApproxExp(x,8); True  = (e - 1)./x - 1;
y1 = (exp(x)- 1)./x - 1; prec1 = abs(True - y1)'
y2 = (exp(x)- 1 - x)./x; prec2 = abs(True - y2)'
y3 = (exp(x)./x - 1./x - 1); prec3 = abs(True - y3)'
y4 = (exp(x)./x + (-1-x)./x); prec4 = abs(True - y4)'
```

Output:

```
prec1 =
   1.0e-10 *
   0.000310862446895
   0.000222044604925
                   0
                   0
   0.222044604925031
                   0
prec2 =
   1.0e-10 *
   0.000310307335383
   0.000222252771742
   0.000000208166817
   0.000000479217360
   0.222045422955570
   0.000000452518882
prec3 =
   1.0e-10 *

   0.000310862446895
   0.000275335310107
   0.000639488462184
   0.003907985046681
   0.202007299776596
   0.416235934608267
prec4 =
   1.0e-10 *
   0.000310862446895
   0.000275335310107
   0.000497379915032
   0.003907985046681
   0.347526452060265
   0.416235934608267
```

2.)

Input

```
%%
clear, clc, close all
format long
x = linspace(1,100,5);
True = 1./((1+x.^8).^(1/2) + x.^4);
y1 = (1+x.^8).^(.5) - x.^4; prec1 = abs(True - y1)'
y2 = (((1+x.^8).^(.5) - x.^4).^2)./((1+x.^8).^(.5) - x.^4); prec2 = abs(True - y2)'
y2 = (((1+x.^8).^(.5) - x.^4).^4)./((1+x.^8).^(.5) - x.^4).^3; prec2 = abs(True - y2)'
y3 =  (1+x.^8).^(.5) + x.^4 - 2.*x.^4;  prec3 = abs(True - y3)'
y3 =  (1+x.^8).^(.5) + 2.*x.^4 - 3.*x.^4;  prec3 = abs(True - y3)'
```

Output

```
prec1
   1.0e-08 *
   0.000000005551115
   0.000214240328460
   0.050997644015562
   0.069235254576864
   0.500000000000000
```

```
prec2 =
   1.0e-09 *
   0.000000055511151
   0.002142403284603
   0.509976440155617
   0.692352545768643
                 NaN
prec2 =
   1.0e-09 *
                   0
   0.002142403284815
   0.509976440155617
   0.692352545768643
                 NaN
```

```
prec3 =
   1.0e-08 *
   0.000000016653345
   0.000214240328460
   0.050997644015562
   0.069235254576864
   0.500000000000000
prec3 =
   1.0e-08 *
   0.000000016653345
   0.011855772511154
   0.237262158938657
   0.069235254576864
   0.500000000000000
```

2b.)

<u>Function</u>

```
function [] = diff_central()
f = @(x) inv(1+30.*x.^2);
h=[0.5, 0.1, 0.05, 0.025, 0.0125]; x = 0.5;
f_prime_true = -((1+30.*x.^2).^-2) *60.*x
for i = 1:length(h)
f_prime_apprx_1(i) = (f(x+h(i))-f(x-h(i)))/
(2*h(i));
err_1(i) = abs(f_prime_true -
f_prime_apprx_1(i));
end
for i = 1:length(h)-1
p(i) = log(err_1(i)/err_1(i-1))/log(h(i)/h(i-1));
end
f_prime_apprx_1'
err_1'
p'
```

2c.)

<u>Function</u>

```
function [] = diff_richard()
% Run with   >> diff_richard()
%
% Approximate derivative of sin(pi x) with 4th
order Richardsons extrap
% Returns a log ratio consistent with a linear
convg. rate
f = @(x) inv(1+30.*x.^2);
h= [0.5, 0.1, 0.05, 0.025, 0.0125]; x = 0.5;
f_prime_true = -((1+30.*x.^2).^-2) *60.*x
for i = 1:length(h)
phi_h(i) = (f(x+h(i))-f(x-h(i)))./ (2.*h(i));
end
for i = 1:length(h) -1
f_prime_apprx(i) = phi_h(i+1)
+(1/3)*(phi_h(i+1) - phi_h(i));
end
for i = 1:length(f_prime_apprx)
err_1(i) = abs(f_prime_true - f_prime_apprx(i));
end
for i = length(err_1):-1:2
   p(i) = log(err_1(i)/err_1(i-1))/log(h(i)/h(i-1));
end
f_prime_apprx'
err_1'
p'
end
```