M/CS 375 Project 2

Isaac Boaz, Cole Yamamura

March 5, 2023

Problem 1

(a) $f(x) = 2x^3 - 6x - 1$ can be rewritten as:

$$
\begin{aligned}
f_1(x) &= \sqrt[3]{3x + \frac{1}{2}} \\
f_2(x) &= \frac{1}{2x^2 - 6} \\
f_3(x) &= \sqrt[3]{3x + \frac{1}{2}} = f_1
\end{aligned}
$$

(b) Taking the derivatives of these $f(x)$'s to show they all converge:

$$
\begin{aligned}
f_1'(x) &= \sqrt[-2/3]{\frac{1}{2} + 3x} \\
f_1'(r_1) &= -0.18549\ldots - 0.3212\ldots i \\
|f_1'(r_1)| &\approx 0.37099 \\
|f_1'(r_1)| &< 1 \checkmark
\end{aligned}
$$

$$
\begin{aligned}
f_2'(x) &= -\frac{x}{(x^2 - 3)^2} \\
f_2'(r_2) &= 0.0190528213703384 \\
|f_2'(r_2)| &= 0.0190528213703384 \\
|f_2'(r_2)| &< 1 \checkmark
\end{aligned}
$$

$$
\begin{aligned}
f_3'(x) &= \sqrt[-2/3]{\frac{1}{2} + 3x} = f_1' \\
f_3'(r_3) &= 0.06924771700019839 \\
|f_3'(r_3)| &= 0.06924771700019839 \\
|f_3'(r_3)| &< 1 \checkmark
\end{aligned}
$$

Problem 1, Programming

The program for part 1 is detailed below.

```
function x = iteratefixed(fun, x, actualRoot, TOL)
    fprintf(" i \t xi \t \t \t g(xi) \t \t error \t \t \t error/lastError\n");
    err = 0;
    step = 0;
    while abs(x - actualRoot) > TOL
        px = x;
        gx = fun(x);
        lastError = err;
        err = abs(x - actualRoot);
        fprintf("%2d %13.9f %13.9f %13.9f", step, px, gx, err);
        if lastError > 0 && err > 0
            fprintf("%13.9f", err / lastError);
        end
            fprintf("\n");
        step = step + 1;
        x = gx;
    end
end
```

i	xi	g(xi)	error	error/lastError
0	−2.000000000	−1.765174168	0.358216473	
1	−1.765174168	−1.686340658	0.123390640	0.344458308
2	−1.686340658	−1.658150305	0.044557131	0.361106244
3	−1.658150305	−1.647833203	0.016366778	0.367321176
4	−1.647833203	−1.644024866	0.006049676	0.369631435
5	−1.644024866	−1.642614632	0.002241339	0.370489116
6	−1.642614632	−1.642091805	0.000831105	0.370807391
7	−1.642091805	−1.641897889	0.000308278	0.370925480
8	−1.641897889	−1.641825954	0.000114362	0.370969292
9	−1.641825954	−1.641799267	0.000042427	0.370985546
10	−1.641799267	−1.641789367	0.000015740	0.370991576
11	−1.641789367	−1.641785694	0.000005839	0.370993813
12	−1.641785694	−1.641784331	0.000002166	0.370994643
13	−1.641784331	−1.641783826	0.000000804	0.370994951
14	−1.641783826	−1.641783638	0.000000298	0.370995065
15	−1.641783638	−1.641783568	0.000000111	0.370995108
16	−1.641783568	−1.641783543	0.000000041	0.370995122
17	−1.641783543	−1.641783533	0.000000015	0.370995124

ans =

−1.6418

i	xi	g(xi)	error	error/lastError
0	−1.000000000	−0.250000000	0.831745598	
1	−0.250000000	−0.170212766	0.081745598	0.098281973
2	−0.170212766	−0.168291940	0.001958364	0.023956815
3	−0.168291940	−0.168255117	0.000037538	0.019167986
4	−0.168255117	−0.168254415	0.000000715	0.019055470
5	−0.168254415	−0.168254402	0.000000014	0.019076087

ans =

−0.1683

i	xi	g(xi)	error	error/lastError
0	3.000000000	2.117911792	1.189962071	
1	2.117911792	1.899513761	0.307873863	0.258725778
2	1.899513761	1.836946464	0.089475832	0.290624968
3	1.836946464	1.818214183	0.026908534	0.300735226
4	1.818214183	1.812530120	0.008176254	0.303853569
5	1.812530120	1.810798297	0.002492190	0.304808320
6	1.810798297	1.810269985	0.000760367	0.305100009
7	1.810269985	1.810108756	0.000232056	0.305189064
8	1.810108756	1.810059547	0.000070827	0.305216244
9	1.810059547	1.810044528	0.000021618	0.305224523
10	1.810044528	1.810039943	0.000006598	0.305226995
11	1.810039943	1.810038544	0.000002014	0.305227568
12	1.810038544	1.810038117	0.000000615	0.305227149
13	1.810038117	1.810037987	0.000000188	0.305225076
14	1.810037987	1.810037947	0.000000057	0.305218069
15	1.810037947	1.810037935	0.000000017	0.305195048

ans =

1.8100

Problem 2

(a) To find the rate of convergence using Netwon's Method, first find the multiplicity of $f(x)$ at $r = 0$

$$\begin{aligned}
f(x) &= e^{sin^3 x} + x^6 - 2x^4 - x^3 - 1 \\
f'(x) &= (6x^3 - 8x - 3)x^2 + 3e^{sin^3 x} sin^2(x)cos(x) \\
f'(0) &= 0 \\
f''(x) &= 6x(5x^3 - 4x - 1) - 3e^{sin^3(x)} sin^3(x) + 3e^{sin^3(x)}(3sin^3(x) + 2)sin(x)cos^2(x) \\
f''(0) &= 0 \\
f^{(3)}(x) &= 3\big(40x^3 - 16x + e^{sin^3(x)}(9sin^6(x) + 18sin^3(x) + 2)cos^3(x) - e^{sin^3(x)} sin^2(x)(9sin^3(x) + 7)cos(x) - 2\big) \\
f^{(3)}(0) &= 0 \\
f^{(4)}(x) &= \cdots \\
f^{(4)}(0) &= -48 \rightarrow m = 4
\end{aligned}$$

Since $m = 4$, we know that Newton's method would have linear convergence (more precisely, $e_i \approx \frac{3}{4} e_{i-1}$)

(b) f has another root in $[1, 2]$ because $f(1) < 0$ and $f(2) > 0$.

(c) Plotting f and looking between $[1, 2]$, we see that the slope grows exponentially, indicating it has a low multiplicity (ie likely it is a simple root).

Problem 2 Programming

The program for part 2 is detailed below.

```
function x = iteratenewtons(fun, dfun, x, actualRoot, TOL)
fprintf(" i \t xi \t \t error \t \t \t error/lastError^2\n");
    err = abs(x - actualRoot);
    step = 1;
    lastError = 0;
    fprintf("%2d %13.9f %13.9f\n", 0, x, err);
    while err > TOL
        px = x;
        x = px - fun(px) / dfun(px);
        lastError = abs(px - actualRoot);
        err = abs(x - actualRoot);
        fprintf("%2d %13.9f %13.9f ", step, x, err);
        if lastError > 0 && err > 0
            fprintf("%13.9f", err / (lastError ^2));
        end
        fprintf("\n");
        step = step + 1;
    end
end
```

```
i          xi              error           error/lastError^2
0     2.000000000     0.469866492
1     1.785128336     0.254994828     1.155001163
2     1.638833023     0.108699515     1.671725109
3     1.557786334     0.027652826     2.340368801
4     1.532097594     0.001964086     2.568511006
5     1.530105426     0.000028082     7.279586801
6     1.530134118     0.000000610   773.296975370
7     1.530133495     0.000000013   35433.350581781
8     1.530133508     0.000000000   1639978.902695282

ans =

    1.5301
```

Problem 3

(a)
1. Inner for loop is run $(m + 100 - 50) + 1 = m + 51$ times
2. Outer for loop changes m, so we simply sum $\sum_{m=1}^{n}$

$$\sum_{m=1}^{N} m + 51 = \frac{N(N + 103)}{2}$$

(b)
1. Inner for loop is run $(m - 10) + 1 = m - 9$ times
2. Second loop modifies m, so sum $\sum_{m=11}^{100}$
3. Outmost loop simply runs N times.

$$N \cdot \sum_{m=11}^{100} m - 9 = 4185N$$

(c)
1. Inner for loop is run $(m - 1) + 1 = m$ times
2. Second loop doesn't modify m, so multiply inner by $(m + 1 - 1) + 1 = m + 1$
3. Outmost loop modifies m, so sum $\sum_{m=1}^{N}$

$$\sum_{m=1}^{N} (m + 1)m = \frac{N(N + 1)(N + 2)}{3}$$