4.

@x0 = -2.5

```
x0 = -2.5;
error = 1e-6;
iterations = 50;
xvalsh = zeros(iterations, 1);
xvalsn = zeros(iterations, 1);
xhalley = x0;
xnewton = x0;
for i = 1:iterations
    fh = xhalley.^3 - 3*xhalley + 2;
    fph = 3*xhalley.^2 - 3;
    fpph = 6*xhalley;
    xnexth = xhalley - (2*fh*fph)/(2*fph.^2 - fh*fpph);
    xvalsh(i) = xnexth;
    if abs(xnexth - xhalley) < error
        break;
    end
    xhalley = xnexth;
end
disp('Halleys Method:')
```

```
Halleys Method:
```

```
fprintf("%.8f\n",xvalsh)
```

```
-2.02272727
-2.00000380
-2.00000000
-2.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
```

```
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
```

```
for i = 1:iterations
    fn = xnewton.^3 - 3*xnewton + 2;
    fpn = 3*xnewton.^2 - 3;
    xnextn = xnewton - fn/fpn;
    xvalsn(i) = xnextn;
    if abs(xnextn - xnewton) < error
        break;
    end
    xnewton = xnextn;
end
disp('Newtons Method:')
```

```
Newtons Method:
```

```
fprintf("%.8f\n",xvalsn)
```

```
-2.11111111
-2.00740741
-2.00003631
-2.00000000
-2.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
```

```
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
```

@x0 = 0

```
x0 = 0;
error = 1e-6;
iterations = 50;
xvalsh = zeros(iterations, 1);
xvalsn = zeros(iterations, 1);
xhalley = x0;
xnewton = x0;
for i = 1:iterations
    fh = xhalley.^3 - 3*xhalley + 2;
    fph = 3*xhalley.^2 - 3;
    fpph = 6*xhalley;
    xnexth = xhalley - (2*fh*fph)/(2*fph.^2 - fh*fpph);
    xvalsh(i) = xnexth;
    if abs(xnexth - xhalley) < error
        break;
    end
    xhalley = xnexth;
end
disp('Halleys Method:')
```

```
Halleys Method:
```

```
fprintf("%.8f\n",xvalsh)
```

```
0.66666667
0.89265537
0.96463841
0.98825892
0.99609140
0.99869770
0.99956596
0.99985533
0.99995178
0.99998393
0.99999464
0.99999821
0.99999940
0.99999980
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
```

```
for i = 1:iterations
    fn = xnewton.^3 - 3*xnewton + 2;
    fpn = 3*xnewton.^2 - 3;
    xnextn = xnewton - fn/fpn;
    xvalsn(i) = xnextn;
    if abs(xnextn - xnewton) < error
        break;
    end
    xnewton = xnextn;
end
```

```
disp('Newtons Method:')
```

```
Newtons Method:
```

```
fprintf("%.8f\n",xvalsn)
```

```
0.66666667
0.84444444
0.92440875
0.96269925
0.98146777
0.99076277
0.99538853
0.99769604
0.99884846
0.99942434
0.99971220
0.99985611
0.99992805
0.99996403
0.99998201
0.99999101
0.99999550
0.99999775
0.99999888
0.99999944
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
```

Yes, convergence for both methods is much slower when x0 = 0. When we look at a graph of x^3-x^2-x-1, we see that there is a critical point at (-0.333, -0.815). Since our inital guess is x0 = 0, the deriviative of the function at this point is much smaller and closer to 0 compared to when x0 = -2.5. Since both methods deal with the derivative(s) of the function in the denominator, this could cause some issues when trying to converge

to a value. This is especially true for Halley's Method since this method involves the second derivative in the denominator, which would be even smaller than the first derivative at x = 0, which is already small.

5.

Secant Method Implementation

```
% x0 = guess 1;
% x1 = guess 2;
% error = 1e-6;
% iterations = 50;
% xvalss = zeros(iterations, 1);
% xsecant1 = x0;
% xsecant2 = x1
% for i = 1:iterations
%     fs1 = function substituting x with xsecant1
%     fs2 = function subsituting x with xsecant2
%     xnexts = xsecant2 - fs2*(xsecant2 - xsecant1)/(fs2 - fs1);
%     xvalss(i) = xnexts;
%     if abs(xnexts - xsecant2) < error
%         break;
%     end
%     xsecant1 = xsecant2;
%     xsecant2 = xnexts;
% end
```

6.

a. f(x) = x.^3 - x.^2 - x - 1, x0 = 1.0, x1 = 2.0

```
x0 = 1.0;
x1 = 2.0;
error = 1e-6;
iterations = 10;
xvalss = zeros(iterations, 1);
xsecant1 = x0;
xsecant2 = x1;
for i = 1:iterations
     fs1 = xsecant1.^3 - xsecant1.^2 - xsecant1 - 1;
     fs2 = xsecant2.^3 - xsecant2.^2 - xsecant2 - 1;
     xnexts = xsecant2 - fs2*(xsecant2 - xsecant1)/(fs2 - fs1);
     xvalss(i) = xnexts;
     if abs(xnexts - xsecant2) < error
         break;
     end
     xsecant1 = xsecant2;
     xsecant2 = xnexts;
end
fprintf("%.8f\n",xvalss)
```

```
1.66666667
1.81632653
1.84299391
1.83921563
1.83928654
1.83928676
0.00000000
0.00000000
0.00000000
0.00000000
```

b. f(x) = e^-x - sin(x), x0 = -1.0, x1 = 0.0

```
x0 = -1.0;
x1 = 0.0;
error = 1e-6;
iterations = 10;
xvalss = zeros(iterations, 1);
xsecant1 = x0;
xsecant2 = x1;
for i = 1:iterations
     fs1 = exp(-xsecant1) - sin(xsecant1);
     fs2 = exp(-xsecant2) - sin(xsecant2);
     xnexts = xsecant2 - fs2*(xsecant2 - xsecant1)/(fs2 - fs1);
     xvalss(i) = xnexts;
     if abs(xnexts - xsecant2) < error
         break;
     end
     xsecant1 = xsecant2;
     xsecant2 = xnexts;
end
fprintf("%.8f\n",xvalss)
```

```
0.39066272
0.55476658
0.58612849
0.58850081
0.58853271
0.58853274
0.00000000
0.00000000
0.00000000
0.00000000
```

c. f(x) = 1 + .3cos(x) - x, x0 = 3.0, x1 = 4.0

```
x0 = 3.0;
x1 = 4.0;
error = 1e-6;
iterations = 50;
xvalss = zeros(iterations, 1);
```

```
xsecant1 = x0;
xsecant2 = x1;
for i = 1:iterations
     fs1 = 1 + 0.3*cos(xsecant1) - xsecant1;
     fs2 = 1 + 0.3*cos(xsecant2) - xsecant2;
     xnexts = xsecant2 - fs2*(xsecant2 - xsecant1)/(fs2 - fs1);
     xvalss(i) = xnexts;
     if abs(xnexts - xsecant2) < error
         break;
     end
     xsecant1 = xsecant2;
     xsecant2 = xnexts;
end
fprintf("%.8f\n",xvalss)
```

```
0.44521226
1.17492314
1.12606672
1.12841974
1.12842509
1.12842509
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
```

```
0.00000000
0.00000000
0.00000000
0.00000000
0.00000000
```

7.

```
x0 = 0;
error = 1e-8;
iterations = 100;
for i = 1:iterations
    xnext = exp(-x0);
    fprintf("%.8f\n",xnext)
    if abs(xnext - x0) < error
        break;
    end

    x0 = xnext;
end
```

```
1.00000000
0.36787944
0.69220063
0.50047350
0.60624354
0.54539579
0.57961234
0.56011546
0.57114312
0.56487935
0.56842873
0.56641473
0.56755664
0.56690891
0.56727623
0.56706790
0.56718605
0.56711904
0.56715704
0.56713549
0.56714771
0.56714078
0.56714471
0.56714248
0.56714375
0.56714303
0.56714344
0.56714321
0.56714334
0.56714326
0.56714331
0.56714328
0.56714330
0.56714329
```

8.

```
bits = 8;
qpfpn = 0:(2^bits - 1);
int = qpfpn./2^4;
frac = qpfpn - int*2^4;
base10 = int + frac/2^4;
disp(base10(base10 >= 0));
```

```
         0    0.0625    0.1250    0.1875    0.2500    0.3125    0.3750    0.4375    0.5000    0.5625    0.
```

I'm not sure how to display this vertically, but the values go from 0 to 15.9375. It will be easier to see if you look at my .mlx file instead of PDF because I think the PDF cuts it off on the screen at a certain point.