Project 1

Pranav Bhogal

In [ ]:
```
using Printf
using Plots
using LaTeXStrings
```

Question 1

In [ ]:
```
function bisection(a::Real, b::Real, f::Function; abs_tol = 1e-5, max_iters = 100)
    # Check that the conditions of the algorithm are satisfied and we have not alre
    @assert f(a)*f(b) ≤ 0 "Not in an interval with a sign change!"
    @assert (f(a) != 0) & (f(b) != 0) "One or both of the endpoints are zeros!"

    converged = false
    n = 0

    while !converged
        n += 1

        # Main step for algorithm
        p = (a + b)/2
        if f(a)*f(p) < 0
             b = p
        elseif f(b)*f(p) < 0
            a = p
        else
            println("Found exact zero at midpoint of iteration $(n)")
            return p
        end

        # Status updates
        println("n: $(n), a: $(a), b: $(b), abserr: $(b-a)")

        # Check convergence
        if b - a < abs_tol
            converged = true
            return p
        end

        # Return if algorithm does not converge
        if n == max_iters
            println("Did not converge in $(max_iters) iterations! Returning last va
            return p
        end
    end
end
```

```
bisection (generic function with 1 method)
```

part a

In [ ]:
```
a = 2
b = 3
@show n_max= ceil(Int, log2(10^8 * (b-a)));
```

```
invalid redefinition of constant a



Stacktrace:

 [1] top-level scope

   @ c:\Users\prana\OneDrive\Documents\GitHub\Math_300_2022\homework\project1.ipyn
b:1
```

part b

In [ ]:
```
f(x) = x^3 - 25
bisection(2, 3, f, abs_tol = 1e-8)
```

```
n: 1, a: 2.5, b: 3, abserr: 0.5
n: 2, a: 2.75, b: 3, abserr: 0.25
n: 3, a: 2.875, b: 3, abserr: 0.125
n: 4, a: 2.875, b: 2.9375, abserr: 0.0625
n: 5, a: 2.90625, b: 2.9375, abserr: 0.03125
n: 6, a: 2.921875, b: 2.9375, abserr: 0.015625
n: 7, a: 2.921875, b: 2.9296875, abserr: 0.0078125
n: 8, a: 2.921875, b: 2.92578125, abserr: 0.00390625
n: 9, a: 2.923828125, b: 2.92578125, abserr: 0.001953125
n: 10, a: 2.923828125, b: 2.9248046875, abserr: 0.0009765625
n: 11, a: 2.923828125, b: 2.92431640625, abserr: 0.00048828125
n: 12, a: 2.923828125, b: 2.924072265625, abserr: 0.000244140625
n: 13, a: 2.9239501953125, b: 2.924072265625, abserr: 0.0001220703125
n: 14, a: 2.92401123046875, b: 2.924072265625, abserr: 6.103515625e-5
n: 15, a: 2.92401123046875, b: 2.924041748046875, abserr: 3.0517578125e-5
n: 16, a: 2.92401123046875, b: 2.9240264892578125, abserr: 1.52587890625e-5
n: 17, a: 2.92401123046875, b: 2.9240188598632812, abserr: 7.62939453125e-6
n: 18, a: 2.9240150451660156, b: 2.9240188598632812, abserr: 3.814697265625e-6
n: 19, a: 2.9240169525146484, b: 2.9240188598632812, abserr: 1.9073486328125e-6
n: 20, a: 2.9240169525146484, b: 2.924017906188965, abserr: 9.5367431640625e-7
n: 21, a: 2.9240174293518066, b: 2.924017906188965, abserr: 4.76837158203125e-7
n: 22, a: 2.9240176677703857, b: 2.924017906188965, abserr: 2.384185791015625e-7
n: 23, a: 2.9240176677703857, b: 2.9240177869796753, abserr: 1.1920928955078125e-7
n: 24, a: 2.9240177273750305, b: 2.9240177869796753, abserr: 5.960464477539063e-8
n: 25, a: 2.9240177273750305, b: 2.924017757177353, abserr: 2.9802322387695312e-8
n: 26, a: 2.9240177273750305, b: 2.9240177422761917, abserr: 1.4901161193847656e-8
n: 27, a: 2.924017734825611, b: 2.9240177422761917, abserr: 7.450580596923828e-9
2.924017734825611
```

In [ ]:
```
function newton(f, df, p0, n_max, rel_tol; verbose = true)

    converged = false;
    p = p0;
    p_old = p0;

    for i in 1:n_max

        p = p_old - f(p_old)/df(p_old);

        if verbose
            println("n: $(i), p: $(p), f(p): $(f(p)), abserr: $(p - p_old)")
        end


        if (i>1)
            if abs(p-p_old)/abs(p)< rel_tol
                converged = true;
                break
            end
        end

        p_old = p;

    end

    if !converged
        @printf("ERROR: Did not converge after %d iterations\n", n_max);
    end

    return p

end
```

```
newton (generic function with 1 method)
```

part c

In [ ]:
```
f(x) = x^3 - 25;
df = x->3*x^2;
p0 = 3;
rel_tol = 1e-8;
n_max = 100;

p = newton(f, df, p0, n_max, rel_tol);
```

```
n: 1, p: 2.925925925925926, f(p): 0.04897627394198523, abserr: -0.07407407407407396
n: 2, p: 2.924018982396379, f(p): 3.1912871790495956e-5, abserr: -0.001906943529546
9006
n: 3, p: 2.9240177382133954, f(p): 1.3578471680375515e-11, abserr: -1.2441829837506
19e-6
n: 4, p: 2.924017738212866, f(p): -3.552713678800501e-15, abserr: -5.29354338141274
6e-13
```

part d

The number of significant digits increases relatively slowly and in a linear fashion for the bisection method. In newton's method the number of significant digits was large right from the begining.

Question 2

part a

In [ ]:
```
f(x) = 230*x^4 + 18*x^3 + 9*x^2 - 221*x - 9;
df = x->920*x^3 + 54*x^2 + 18*x - 221
rel_tol = 1e-6;
n_max = 100;
p0 = 0;

p = newton(f, df, p0, n_max, rel_tol);
```

```
n: 1, p: -0.04072398190045249, f(p): 0.01434289269106337, abserr: -0.04072398190045249
n: 2, p: -0.0406592884873237, f(p): 3.803668491286771e-8, abserr: 6.469341312879268e-5
n: 3, p: -0.04065928831575886, f(p): 0.0, abserr: 1.715648387246027e-10
```

In [ ]:
```
p0 = 1;
p = newton(f, df, p0, n_max, rel_tol);
```

```
n: 1, p: 0.9649805447470817, f(p): 1.729702780624109, abserr: -0.035019455252918275
n: 2, p: 0.96241172497926, f(p): 0.008867662019611089, abserr: -0.0025688197678217506
n: 3, p: 0.9623984191063186, f(p): 2.3709401375526795e-7, abserr: -1.3305872941349328e-5
n: 4, p: 0.9623984187505414, f(p): 0.0, abserr: -3.557771854900693e-10
```

In [ ]:
```
function secant(f, p0, p1, n_max, rel_tol; verbose = true)

    converged = false;

    p = p0;
    for i in 1:n_max

        p = p1 - f(p1) * (p1-p0)/(f(p1)-f(p0));

        if verbose
            @printf(" %d: p = %.12g, f(p) = %g\n", i, p, f(p));
        end


        if (i>1)
            if abs(p-p1)/abs(p1)< rel_tol
                converged = true;
                break
            end
        end
        p0 = p1;
        p1 = p;

    end

    if !converged
        @printf("ERROR: Did not converge after %d iterations\n", n_max);
    end

    return p

end
```

```
secant (generic function with 1 method)
```

part b

In [ ]:
```
f(x) = 230*x^4 + 18*x^3 + 9*x^2 - 221*x - 9;
p0 = 1;
p1 = 2;
rel_tol = 1e-6;
n_max = 100;

p = secant(f, p0, p1, n_max, rel_tol);
```

```
 1: p = 0.99201655825, f(p) = 20.9363
 2: p = 0.98578780779, f(p) = 16.3312
 3: p = 0.963698523111, f(p) = 0.86867
 4: p = 0.962457566912, f(p) = 0.0394217
 5: p = 0.962398572997, f(p) = 0.000102792
 6: p = 0.962398418769, f(p) = 1.22166e-08
```

In [ ]:
```
f(x) = 230*x^4 + 18*x^3 + 9*x^2 - 221*x - 9;
p0 = -1;
p1 = 0;
rel_tol = 1e-6;
n_max = 100;

p = secant(f, p0, p1, n_max, rel_tol);
```

```
 1: p = -0.0203619909502, f(p) = -4.49638
 2: p = -0.0406912564352, f(p) = 0.00708748
 3: p = -0.0406592625777, f(p) = -5.70624e-06
 4: p = -0.0406592883157, f(p) = -7.47846e-12
```

part c

In [ ]:
```
function muller(f, p0, p1, p2, n_max, rel_tol; verbose = true)

    converged = false;
    p = p2;

    for i in 1:n_max

        # solve for the constants a, b, and c
        c = f(p2);
        A = [(p0-p2)^2 p0-p2; (p1-p2)^2 p1-p2 ];
        x = A\[f(p0)-c; f(p1)-c];
        a = x[1];
        b = x[2];

        # take the root with larger denominator
        if abs(b + sqrt(b^2-4*a*c))> abs(b - sqrt(b^2-4*a*c))
            p = p2 - 2*c/(b + sqrt(b^2-4*a*c));
        else
            p = p2 - 2*c/(b - sqrt(b^2-4*a*c));
        end

        if verbose
            @printf(" %d: p = %.15g + i %.15g, |f(p)| = %g\n", i,
                real(p), imag(p), abs(f(p)));
        end


        if (i>1)
            if abs(p-p2)/abs(p)< rel_tol
                converged = true;
                break
            end
        end

        # update entries
        p0 = p1;
        p1 = p2;
        p2 = p;

    end

    if !converged
        @printf("ERROR: Did not converge after %d iterations\n", n_max);
    end

    return p

end
```

```
muller (generic function with 1 method)
```

In [ ]:
```
f(x) = 230*x^4 + 18*x^3 + 9*x^2 - 221*x - 9;

p0 = -2
p1 = -1;
p2 = 0;

rel_tol = 1e-6;
n_max = 100;

p = muller(f, p0, p1, p2, n_max, rel_tol);
```

```
 1: p = 0.00792668512248133 + i 0, |f(p)| = 10.7512
 2: p = -0.0389266709197757 + i 0, |f(p)| = 0.384102
 3: p = -0.0406592648750403 + i 0, |f(p)| = 5.19691e-06
 4: p = -0.040659288315828 6 + i 0, |f(p)| = 1.54579e-11
```

In [ ]:
```
f(x) = 230*x^4 + 18*x^3 + 9*x^2 - 221*x - 9;

p0 = 0
p1 = 1;
p2 = 2;

rel_tol = 1e-6;
n_max = 100;

p = muller(f, p0, p1, p2, n_max, rel_tol);
```

```
 1: p = 0.983949082349461 + i 0, |f(p)| = 14.9926
 2: p = 0.960752804805569 + i 0, |f(p)| = 1.09303
 3: p = 0.962464045474489 + i 0, |f(p)| = 0.0437402
 4: p = 0.962398421921513 + i 0, |f(p)| = 2.11317e-06
 5: p = 0.962398418750542 + i 0, |f(p)| = 3.69482e-13
```

In [ ]:
```
f(x) = 230*x^4 + 18*x^3 + 9*x^2 - 221*x - 9;
p0 = -0.9 - 1*im;
p1 = -0.3 - 1*im;
p2 = -1 - 1*im;

rel_tol = 1e-6;
n_max = 100;

p = muller(f, p0, p1, p2, n_max, rel_tol);
```

```
 1: p = -0.59384647620523 + i -0.652944336667262, |f(p)| = 117.705
 2: p = -0.464901117476971 + i -0.889624177080721, |f(p)| = 28.3156
 3: p = -0.49462663647666 + i -0.869677858813927, |f(p)| = 4.31773
 4: p = -0.500016793115442 + i -0.865941303814642, |f(p)| = 0.0569143
 5: p = -0.500000001222932 + i -0.866025371432536, |f(p)| = 2.14883e-05
 6: p = -0.50000000000002 + i -0.866025403784454, |f(p)| = 1.64487e-11
```

In [ ]:
```
f(x) = 230*x^4 + 18*x^3 + 9*x^2 - 221*x - 9;
p0 = 0+ 1*im;
p1 = -1 + 1*im;
p2 = -2 + 1*im;

rel_tol = 1e-6;
n_max = 100;

p = muller(f, p0, p1, p2, n_max, rel_tol);
```

```
1: p = -0.921047663063056 + i 0.718969415274924, |f(p)| = 385.488
2: p = -0.68680289823207 + i 0.937267253041728, |f(p)| = 180.024
3: p = -0.58241790691688 + i 0.913648461742274, |f(p)| = 74.3579
4: p = -0.503307061814207 + i 0.876576668546098, |f(p)| = 7.50196
5: p = -0.499689498558413 + i 0.86603795247068, |f(p)| = 0.2062
6: p = -0.500000426347521 + i 0.866025230739865, |f(p)| = 0.000305401
7: p = -0.500000000000619 + i 0.866025403782352, |f(p)| = 1.4444e-09
```

Question 3

part a

In [ ]:

```
function a(p, tolerance, maxn)

    # print out the absolute error of the approximation
    println("Error: ", abs(p - (7^(1/5))))
    # if the absolute error is within the tolerance,
    # print number of iterations and the approximation
    if abs(p - (7^(1/5))) == 0
        println("Iterations: ", 100 - maxn)
        println("Actual Value: ", 7^(1/5))
        println("Approximation: ", p)
    elseif maxn == 0
        println("Function diverges")
    # recalculate new approximation
    else
        maxn = maxn - 1
        new_p = p * ((1+((7-(p^5))/(p^2)))^3)
        c(new_p, tolerance, maxn)
    end
end

a(1, 10^-8, 100)
```

```
Error: 0.4757731615945522
Error: 341.52422683840547
Error: 2.253933861495501e25
Error: 3.383854504272191e253
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
```

```
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Function diverges
```

part b

In [ ]:
```
function b(p, tolerance, maxn)

    # print out the absolute error of the approximation
    println("Error: ", abs(p - (7^(1/5))))
    # if the absolute error is within the tolerance,
    # print number of iterations and the approximation
    if abs(p - (7^(1/5))) == 0
        println("Iterations: ", 100 - maxn)
        println("Actual Value: ", 7^(1/5))
        println("Approximation: ", p)
    elseif maxn == 0
        println("Function diverges")
    # recalculate new approximation
    else
        maxn = maxn - 1
        new_p = p - (((p^5)-7)/(p^2))
        c(new_p, tolerance, maxn)
    end
end

b(1, 10^-8, 100)
```

```
Error: 0.4757731615945522
Error: 5.524226838405448
Error: 2.796610720267935e8
Error: 2.9263088323271037e84
Error: Inf
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
```

```
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Error: NaN
Function diverges
```

part c

In [ ]:
```
function c(p, tolerance, maxn)

    # print out the absolute error of the approximation
    println("Error: ", abs(p - (7^(1/5))))
    # if the absolute error is within the tolerance,
    # print number of iterations and the approximation
    if abs(p - (7^(1/5))) == 0
        println("Iterations: ", 100 - maxn)
        println("Actual Value: ", 7^(1/5))
        println("Approximation: ", p)
    elseif maxn == 0
        println("Function diverges")
    # increment number of iterations
    # decrement max loops
    # recalculate new approximation
    else
        maxn = maxn - 1
        new_p = p - (((p^5)-7)/(5*(p^4)))
        c(new_p, tolerance, maxn)
    end
end


# test function with provided initial p = 1,
# iterations = 0,
# a very tight tolerance (as close to zero as possible),
# and a sample number of loops (enough to see
# convergence or divergence)
c(1, 10^-8, 100)
```

```
Error: 0.4757731615945522
Error: 0.724226838405448
Error: 0.3439905157498917
Error: 0.10770166830160921
Error: 0.0136878125508757
Error: 0.0002492745373088301
Error: 8.418205599269868e-8
Error: 9.547918011776346e-15
Error: 2.220446049250313e-16
Error: 0.0
Iterations: 9
Actual Value: 1.4757731615945522
Approximation: 1.4757731615945522
```

part d

In [ ]:
```
function d(p, tolerance, maxn)

    # print out the absolute error of the approximation
    println("Error: ", abs(p - (7^(1/5))))
    # if the absolute error is within the tolerance,
    # print number of iterations and the approximation
    if abs(p - (7^(1/5))) == 0
        println("Iterations: ", 100 - maxn)
        println("Actual Value: ", 7^(1/5))
        println("Approximation: ", p)
    elseif maxn == 0
        println("Function diverges")
    # recalculate new approximation
    else
        maxn = maxn - 1
        new_p = p - (((p^5)-7)/12)
        c(new_p, tolerance, maxn)
    end
end

d(1, 10^-8, 100)
```

```
Error: 0.4757731615945522
Error: 0.024226838405447815
Error: 0.0007700482819910093
Error: 8.027739175631154e-7
Error: 8.733014311701481e-13
Error: 0.0
Iterations: 95
Actual Value: 1.4757731615945522
Approximation: 1.4757731615945522
```

fastest to slowest convergence

c -> d -> b -> a

we can also observe that b and c will never converge.