Problem 5.12

Given $f(x) = -2x^6 - 1.6x^4 + 12x + 1$, use bisection to determine the maximum of this function. Employ initial guesses of $x_l = 0$ and $x_u = 1$, and perform iterations until the approximate relative error falls below 5%.

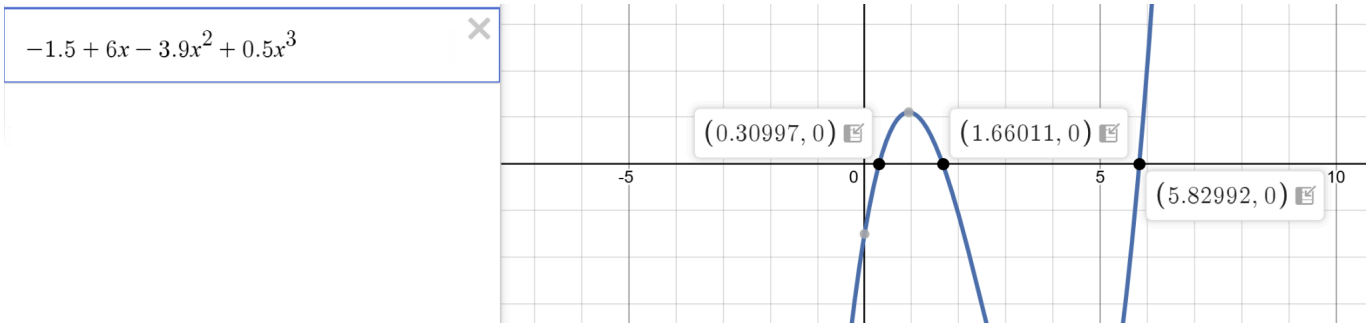
The calculated x value for the max of the given function by my bisection equation is 0.90625

```rust
fn main() {
    let res: f64 = bisection(xl: 0.0, xu: 1.0, es: 5.0, imax: 1000, xr: 0.6);
    println!(
        "The approximate coords for the maximum of this function are: ({res}, {:.5})",
        polynomial(res)
    );
    println!(
        "The true x value for the maximum of this function are: (0.90449, {:.5})",
        polynomial(0.90449)
    );
}

fn bisection(xl: f64, xu: f64, es: f64, imax: usize, xr: f64) -> f64 {
    let mut iter: usize = 0;
    let mut ea: f64 = 100.0;
    let mut xu: f64 = xu;
    let mut xl: f64 = xl;
    let mut xr: f64 = xr;
    loop {
        let xr_old: f64 = xr;
        xr = (xl + xu) / 2 as f64;
        if xr != 0 as f64 {
            ea = {
                let absv: f64 = xr - xr_old;
                (absv.abs() / xr) * 100 as f64
            };
        }
        let test: f64 = first_dx(xl) * first_dx(xr);
        if test < 0 as f64 {
            xu = xr;
        } else if test > 0 as f64 {
            xl = xr;
        } else {
            ea = 0.0;
        }
        if ea < es || iter >= imax {
            break;
        }
        iter += 1;
    }
    xr
} fn bisection

fn polynomial(x: f64) -> f64 {
    -2.0 * x.powi(6) - 1.6 * x.powi(4) + 12.0 * x + 1.0
}

fn first_dx(x: f64) -> f64 {
    -12.0 * x.powi(5) - 6.4 * x.powi(3) + 12.0
}
```

```
The approximate coords for the maximum of this function are: (0.90625, 9.68783)
The true x value for the maximum of this function are: (0.90449, 9.68792)
```

Problem 6.4

Determine the real roots of $f(x) = -1.5 + 6x - 3.9x^2 + 0.5x^3$ (a) graphically and



(b) using the Newton-Raphson method to within εs = 0.01%.

```rust
fn main() {
    let guesses: [f64; 3] = [0.0, 1.0, 2.0];

    for guess: f64 in guesses {
        let res: Option<f64> = newton_raphson_method(x0: guess, itermax: 100, es: 0.01);
        match res {
            Some(x: f64) => println!(
                "Starting at {guess}, root found: ({x:.5}, {:.5})",
                og_func(x).abs()
            ),
            None => println!("Starting at {guess}, no root was found within the given iterations"),
        }
    }
}

fn newton_raphson_method(x0: f64, itermax: usize, es: f64) -> Option<f64> {
    let mut iter: usize = 0;
    let mut xr: f64 = x0;
    let mut ea: f64 = 100 as f64;
    loop {
        let xr_old: f64 = xr;
        xr = xr_old - (og_func(xr) / first_dx(xr));
        let poss_root: f64 = og_func(xr);
        iter += 1;
        if xr != 0 as f64 {
            ea = ((xr - xr_old).abs() / xr) * 100.0;
        }
        if (ea < es || iter >= itermax) && poss_root.abs() < 1e-9 {
            break;
        }
    }
    if iter >= itermax {
        return None;
    }
    Some(xr)
}

fn og_func(x: f64) -> f64 {
    -1.5 + 6.0 * x - 3.9 * x.powi(2) + 0.5 * x.powi(3)
}

fn first_dx(x: f64) -> f64 {
    6.0 - 7.8 * x + 1.5 * x.powi(2)
}
```

```
Starting at 0, root found: (0.30997, 0.00000)
Starting at 1, root found: (5.82992, 0.00000)
Starting at 2, root found: (1.66011, 0.00000)
```