Hand in 2

To use the equations in Newtons method all of the equations were rearranged to be equal to 0. Thereafter, the equations and constants were written into a function like this.

```
VecDoub vecfunc(VecDoub_I x)
{
    // n and d are held constant
    double d = 30;
    // std::cout << "n_var" << n_var << std::endl;
    //  x consists of {L_0, L, p, x, theta, phi, a, H}
    double L_0 = x[0];
    double L = x[1];
    double p = x[2];
    double x_ = x[3];
    double theta = x[4];
    double phi = x[5];
    double a = x[6];
    double H = x[7];

    // material constants
    double v = 120;
    double k = 2.5;
    double w = 4.0;
    double alpha = 2 * 10e-7;

    VecDoub result(8);
    result[0] = a * (cosh(x_ / a) - 1) - p;
    result[1] = 2 * a * sinh(x_ / a) - L;
    result[2] = 2 * x_ + 2 * k * cos(theta) - d;
    result[3] = p + k * sin(theta) - n_var;
    result[4] = sinh(x_ / a) - tan(phi);
    result[5] = (1 + v / (w * L_0)) * tan(phi) - tan(theta);
    result[6] = L_0 * (1 + alpha * H) - L;
    result[7] = (w * L_0) / (2 * sin(phi)) - H;

    return result;
}
```

Then the main() function was set up like this, where I loop through the different n values, and find the solutions for each.

```
int main()
{
    // matrix for storing L_0 and H for each n
```

```
    MatDoub L_0_H(6, 2);
    // initial guess;
    // x consists of {L_0, L, p, x, theta, phi, a, H}
    VecDoub_IO x_guess(8);
    x_guess[0] = 24;
    x_guess[1] = 24;
    x_guess[2] = 1;
    x_guess[3] = 12;
    x_guess[4] = 0.1;
    x_guess[5] = 0.1;
    x_guess[6] = 100;
    x_guess[7] = 100;

    std::vector<double> n_vec = {0.1, 0.2, 0.5, 1.0, 2.0, 5.0};

    bool check = true;
    /* ------------------------------- n = 0.1 ----------------------
---------- */
    for (size_t i = 0; i < n_vec.size(); i++)
    {
        n_var = n_vec[i];
        cout << GREEN << "n = " << n_vec[i] << RESET << endl;
        VecDoub_IO x = x_guess;
        newto(x, check, vecfunc);

        util::print(x, "solution_x");
        L_0_H[i][0] = x[0];
        L_0_H[i][1] = x[7];
    }

    // loop through the matrix and print the values with corresponding
n
    cout << YELLOW << setw(5) << "n" << setw(15) << "L_0" << setw(15)
<< "H" << RESET << endl;
    for (size_t i = 0; i < L_0_H.nrows(); i++)
    {
        cout << setw(5) << n_vec[i] << setw(15) << L_0_H[i][0] <<
setw(15) << L_0_H[i][1] << endl;
    }

    return 0;
}
```

The defines for the colors can be seen here

```
// ANSI color codes
```

```
#define RESET "\033[0m"
#define RED "\033[31m"
#define GREEN "\033[32m"
#define YELLOW "\033[33m"
#define BLUE "\033[34m"
#define MAGENTA "\033[35m"
#define CYAN "\033[36m"
#define WHITE "\033[37m"
```

This gives the following values for L_0 and H

n	L_0	H
0.1	24.7133	5824.64
0.2	24.8593	2924.83
0.5	24.9708	1173.09
1	25.0875	587.314
2	25.4431	294.612
5	27.5172	124.517