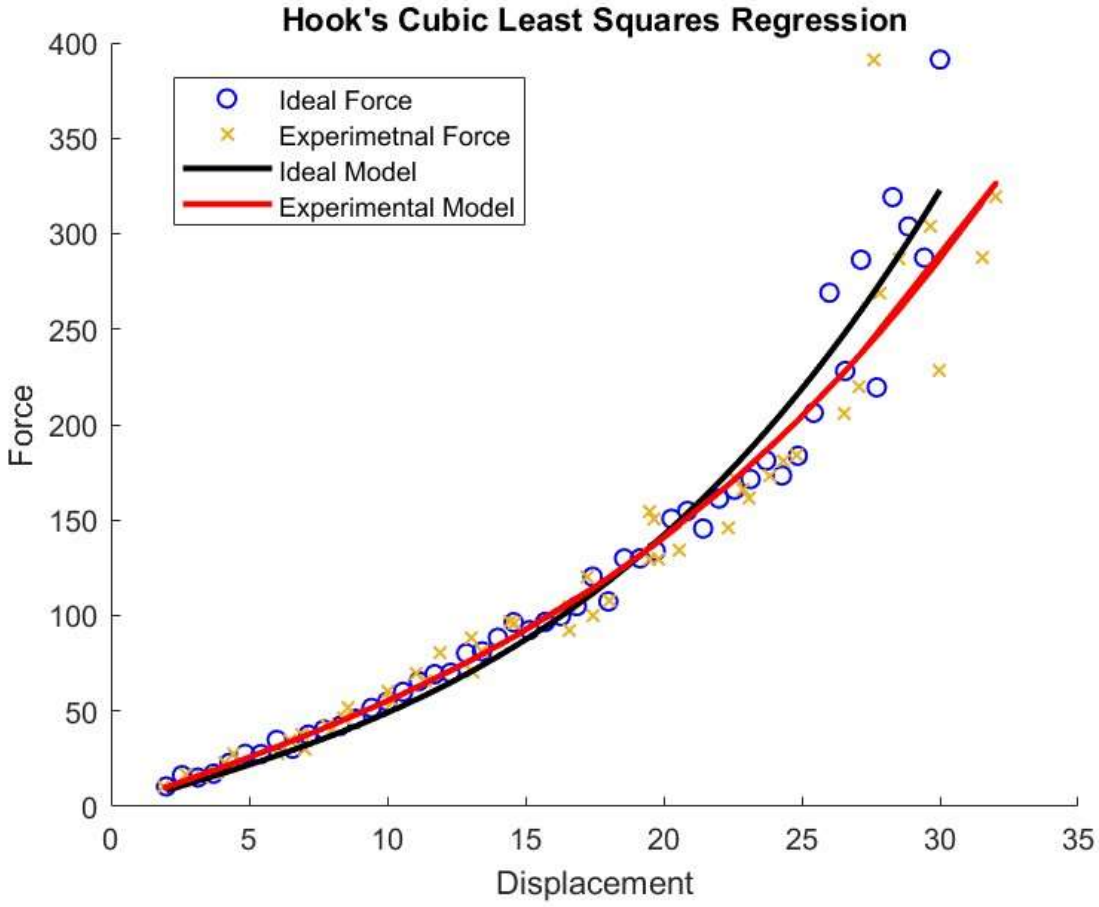
```
classdef Set_5
    methods (Static = true)

        function [gx, ex, f] = process_data(file1)


            opts1 = detectImportOptions(file1);

            data1 = readmatrix(file1, opts1);

            gx = data1(1:50, 1);

            ex = data1(1:50, 2);

            f = data1(1:50, 3);



        end

        function [gx, f] = process_RHdata(file1)


            opts1 = detectImportOptions(file1);

            data1 = readmatrix(file1, opts1);

            gx = data1(1:13, 1);


            f = data1(1:13, 2);



        end

        %% Output
            % for ideal data:
                % k1 = 4.17906362865596
                % k3 = 0.00731462542535269
            % for experimental data:
                % k1 = 5.021387215396081
                % k3 = 0.005043648279854
        function [K]  = cubic_spring(x, F)

            x3 = x .^3;

            A = [x x3];

            K = (A' * A) \ (A' * F);
        end
```



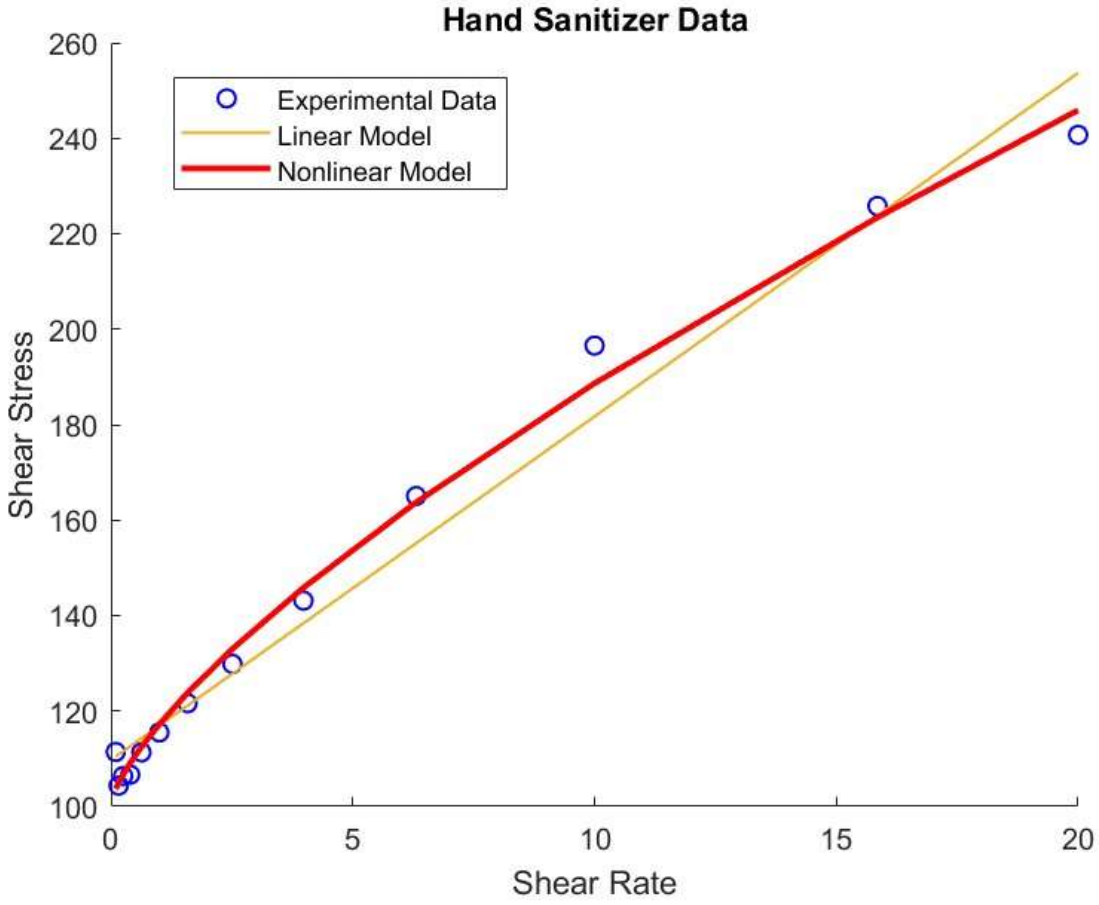
function spring_plot(x, ex, F, K, Ke)

```
Ax = [x x .^3];
Aex = [ex ex .^3];

Fx = Ax * K;
Fex = Aex * Ke;


f1 = figure(1);
hold on
title("Hook's Cubic Least Squares Regression");
xlabel("Displacement");
ylabel("Force");

plot(x, F, 'ob', 'LineWidth', 1);
plot(ex, F, 'x', 'Color', [0.9290 0.6940 0.1250], 'LineWidth', 1);
plot(x, Fx, '-k', 'LineWidth', 2);
plot(ex, Fex, '-r', 'LineWidth', 2);
```

```
            legend("Ideal Force", "Experimetnal Force", "Ideal Model", "Experimental
Model");
            hold off



        end

        %% Output
            % for ideal data:
                % k1 = 4.17906362865596
                % k3 = 0.00731462542535269
                % F50 = 1.123281359601883e+03
            % for experimental data:
                % k1 = 5.021387215396081
                % k3 = 0.005043648279854
                % F50 = 8.815253957515517e+02
        function  [k1, k3, k1e, k3e, F50, F50e] = spring_compare()

            file = "../data/x-x_w_error-y.csv";


            [x, ex, F] = Set_5.process_data(file);

            K = Set_5.cubic_spring(x, F);

            Ke = Set_5.cubic_spring(ex, F);

            fifty = [50 50^3];

            k1 = K(1);
            k3 = K(2);
            k1e = Ke(1);
            k3e = Ke(2);

            F50 = fifty * K;

            F50e = fifty * Ke;

            Set_5.spring_plot(x, ex, F, K, Ke);


        end


        %% Output:
            % Tau = 1.097396637644277e+02
            % nu = 7.197512145916267
        function A=bingham(x,y)

            Z = [ones(length(x), 1) x];

            A = (Z' * Z) \ (Z' * y);
        end
```

```
        %% Input:
            % Seed is determined from the result of the linear least squares
regression
                % Tau = 1.097396637644277e+02
                % K = 7.197512145916267
                % n = 1

        %% Output:
                % Tau = 1.004899149000514e+02
                % K = 16.730645749178155
                % n = 0.721791776180352
        function B = HB(x,y,T,K,n)

            err = 100;

            accept = .02;


            while (err > accept )

                resid = y - T - K * (x .^n);

                dtau = ones(length(x), 1);

                dk = x .^ n;

                dn = K * (log(x) .* (x .^ n));


                J = [dtau dk dn];

                B = (J' * J ) \ (J' * resid);


                err = (abs(B(1) / T) + abs(B(2) / K) + abs(B(3) / n));


                T = T + B(1);
                K = K + B(2);
                n = n + B(3);




            end


            B(1) = T;
            B(2) = K;
            B(3) = n;
```

```
end
```




```
function []=model_compare()
            filerh = "../data/rheo_data.csv";

            [x, y] = Set_5.process_RHdata(filerh);

            F = Set_5.bingham(x, y);

            B = Set_5.HB(x, y, F(1), F(2), 1);

            Bp = [B(1); B(2)];




            lin = [ones(length(x), 1)  x];
            nlin= [ones(length(x), 1)  x.^B(3)];

            linearFit = lin * F;
            nonlinearFit = nlin * Bp;


            f2 = figure(2);
```

```
            hold on
            title("Hand Sanitizer Data");
            xlabel("Shear Rate");
            ylabel("Shear Stress");

            plot(x, y, 'ob', 'LineWidth', 1);
            plot(x, linearFit, '-', 'Color', [0.9290 0.6940 0.1250], 'LineWidth', 1);
            plot(x, nonlinearFit, '-r', 'LineWidth', 2);
            legend("Experimental Data", "Linear Model", "Nonlinear Model");
            hold off









        end



    end

end
```