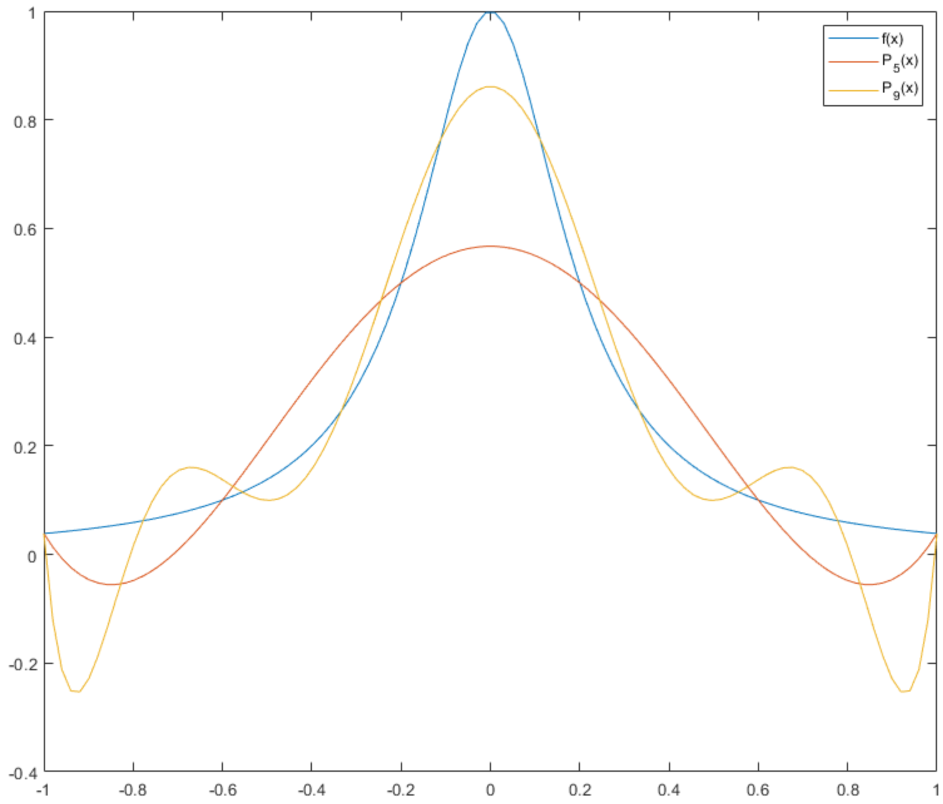
In numerical analysis we are often tasked to derive well-behaved and efficiently computed approximations to complex mathematical functions often times in the form of a polynomial. This idea of polynomial interpolation, at it's most basic form we encounter an interesting problem known as Runge's Phenomenon. The following is called the Runge function,

$$f(x) = \frac{1}{1 + 25x^2}.$$

Now consider the following degree n polynomial interpolants, $P_n(x)$ on the interval $[-1, 1]$ with $n + 1$ equally spaced points.

Figure 1: Demonstration of Runge's Phenomenon



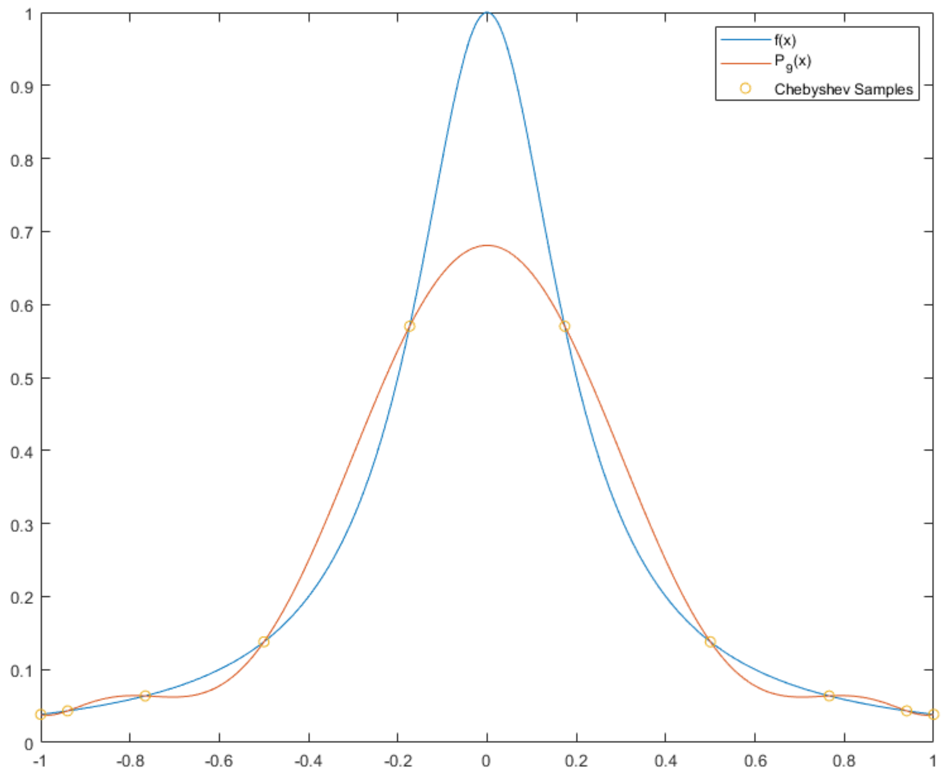
We can see that as we interpolate our function $f(x)$ with larger and larger polynomials the oscillation at the end of the interval increases, and at the extremes it is a well known result that the interpolation error diverges as the degree of the polynomial increases. Luckily there are many ways to mitigate this problem, and the method of interest as it pertains to this project is using Chebyshev interpolation points or (Gauss-Lobatto points).

A set of $n + 1$ Chebyshev points can be described with the following function,

$$x_i = cos(\frac{\pi i}{n}), i = 0, 1, \ldots, n.$$

Plotting these points, we can see that they are more densely distributed towards the edges of the interval, and we can also see that interpolating a polynomial on these points diminished the error we saw previously with equally spaced points.

Figure 2: Demonstration of Chebyshev Sampling on Polynomial Interpolation



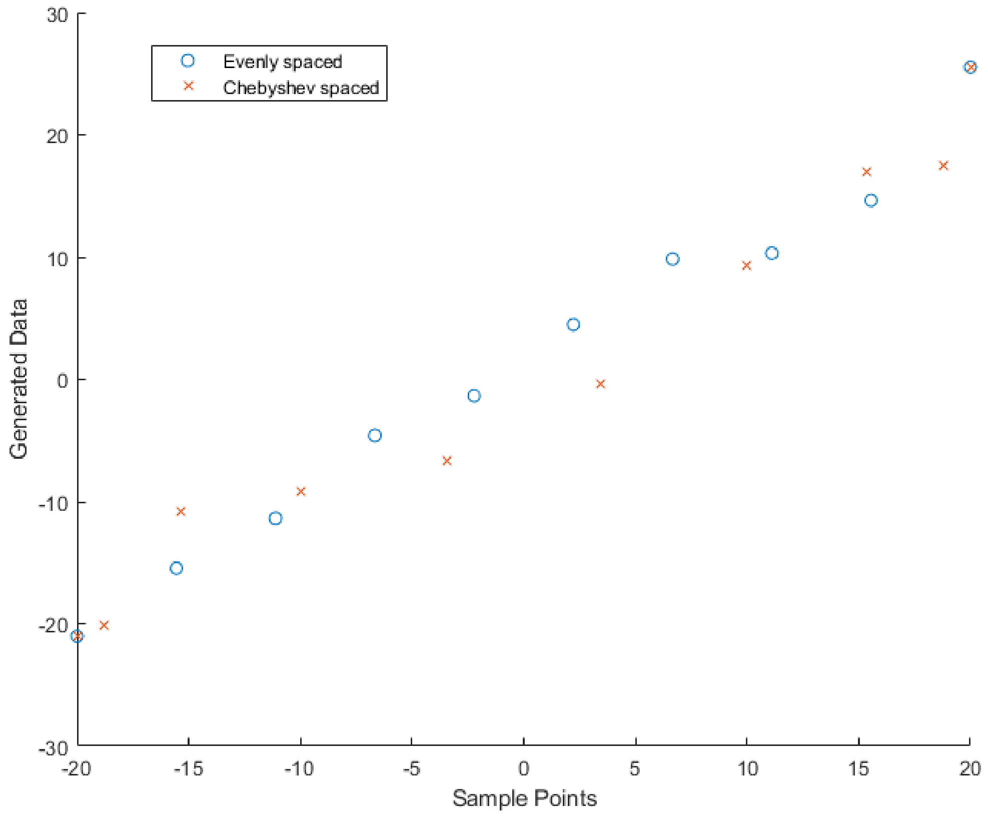
For this project we will be exploring how these points might effect the statistical accuracy of a simple linear regression, comparing them to sampling data taken at evenly spaced points. To begin comparing these two sampling methods we must first simulate a data set that actually contains samples at the desired sample points. Consider the following MATLAB code,

Function:

```
function [y_1, y_2] = Generate_Sample_Data(f,a,b,s)
% This function takes a sampling function, an interval [a, b]
% and a number of nodes s. This function returns the data
% values, y for a given x, such that y = f(x) +- norm(0,3).

    % Generates s, evenly spaced points on [a,b]
    x_1 = linspace(a,b,s);
    % Generates s, chebyshev spaced points on [a,b]
    x_2 = chebypoint(a,b,s-1);

    % Set Theory logic to ensure each sample point generates data once
    x_1_isolated = setdiff(x_1, x_2);
    x_2_isolated = setdiff(x_2, x_1);
    x_intersect = intersect(x_1, x_2);
    y_intersect = gen_data_function(x_intersect,3,f);
    y_1_isolated = [gen_data_function(x_1_isolated,3,f)];
    y_2_isolated = [gen_data_function(x_2_isolated,3,f)];

    % Return each data value separatly and in asending order
    y_1 = sort([y_1_isolated, y_intersect]);
    y_2 = sort([y_2_isolated, y_intersect]);

end
```

When we pass, $f(x) = x$, $a = -20$, $b = -20$, and $n = 10$ this MATLAB code generates a data set similar to the following,

Figure 3: Example of Generated DataSet

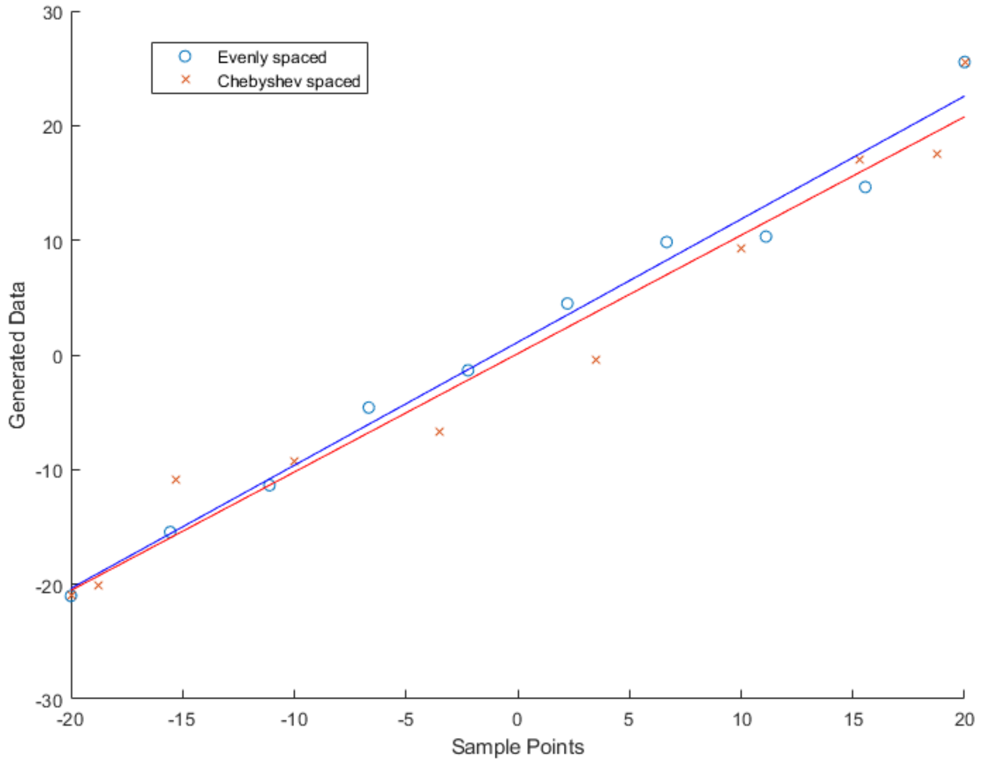


From here we can simply fit a simple linear regression to both sample sets and start our analysis of the key differences. The following MATLAB code will fit a linear model and returns a struct with all the necessary information, similarly to the *lm* function in r.

Console:

```
reg_1 = fitlm(x_1, y_1);
reg_2 = fitlm(x_2, y_2);
```

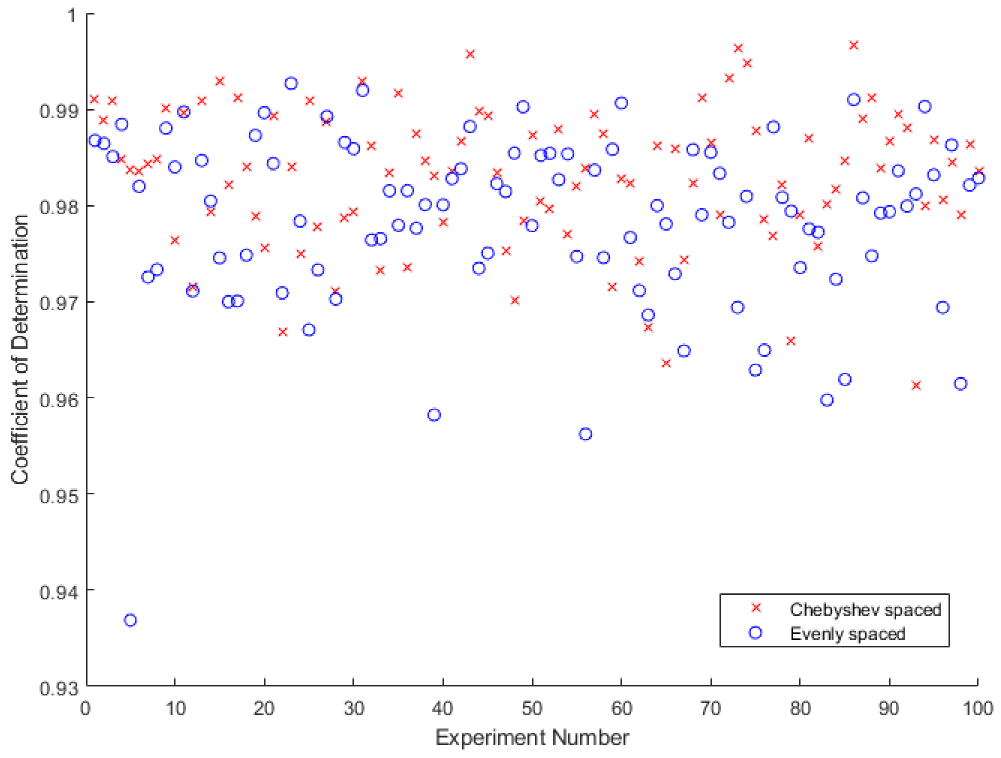
Figure 4: Example of Generated DataSet



For the analysis we conducted the same experiment 100 times with 15 sample points for each sample method, on an interval from $[-20, 20]$. To see if on average there is a meaningful difference between the sampling procedure, we conducted paired sample t-test on goodness of fit metrics like the correlation coefficients, standard errors, and the Root Mean Square Errors. Before we get to the results let's first discuss the assumptions of a paired t-test. To conduct a paired t-test the data must be continuous, the data, must also be approximatly normally distributed, and finally the sample of pairs must be a simple random sample. In the case of our data all these conditions apply, and the only rule that is slightly bent is that the data follows a normal distribution, however with a sample size of 100 the central limit theorem kicks in and forces the sample mean of our goodness of fit variables close to normal.

Looking at our findings the paired t-test showed that there was a significant statistical difference in the mean value of the Coefficient of determination. The following is a plot of each r^2 value for each sampling method along with the results of the paired t-test.

Figure 5: Plot of r^2 Data



Console:

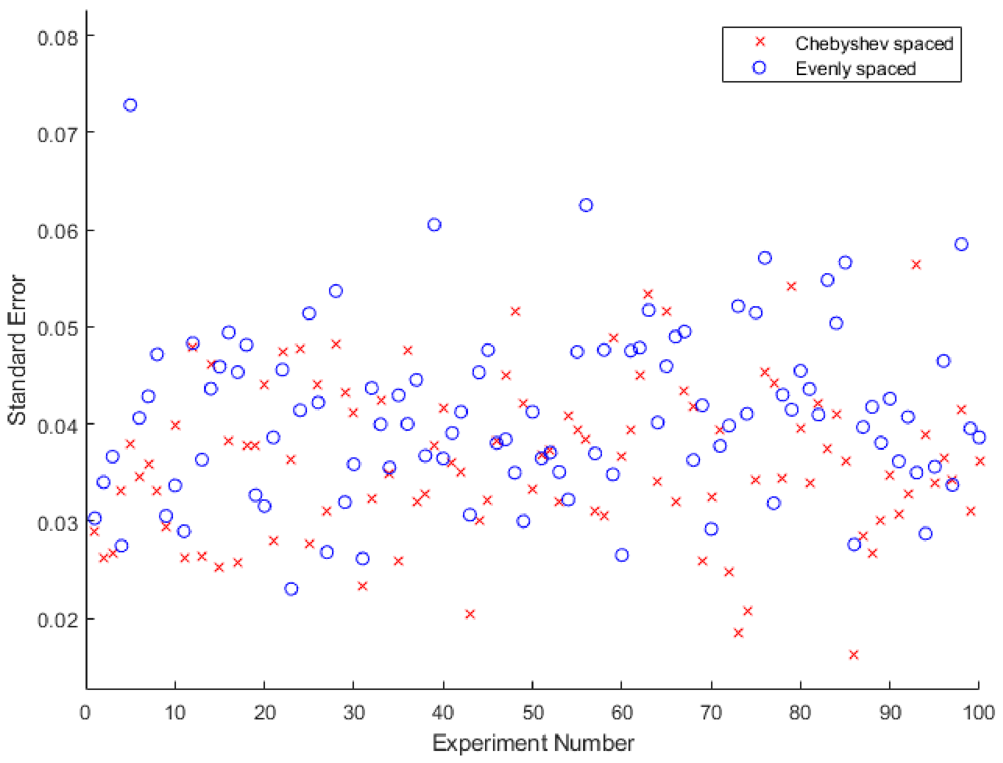
```
h =
        1 (Rejection of Null)

p =
        1.502225970497423e-04

ci =
    -0.006540849291653    -0.002161614356217

stats =
       tstat:  -3.943057667663062
```

```
df: 99
sd: 0.011035171662894
```

From the plot of our data there is a somewhat larger concentration of Chebyshev pints towards the top of the scatter plot and some outliers for the evenly spaced points towards the bottom. With a 95% confidence interval that the mean of the difference between r^2 values is between $(-0.00654, -0.00216)$ its hard to justify that there is any meaningful difference between the two sampling methods with respect to r^2.

Looking at our data for standard error, the paired t-test also showed that there was a significant statistical difference in the mean value of the Standard Error. The following is a plot of each standard error value for each sampling method along with the results of the paired t-test.

Figure 6: Plot of $StandardError$ Data



Console:

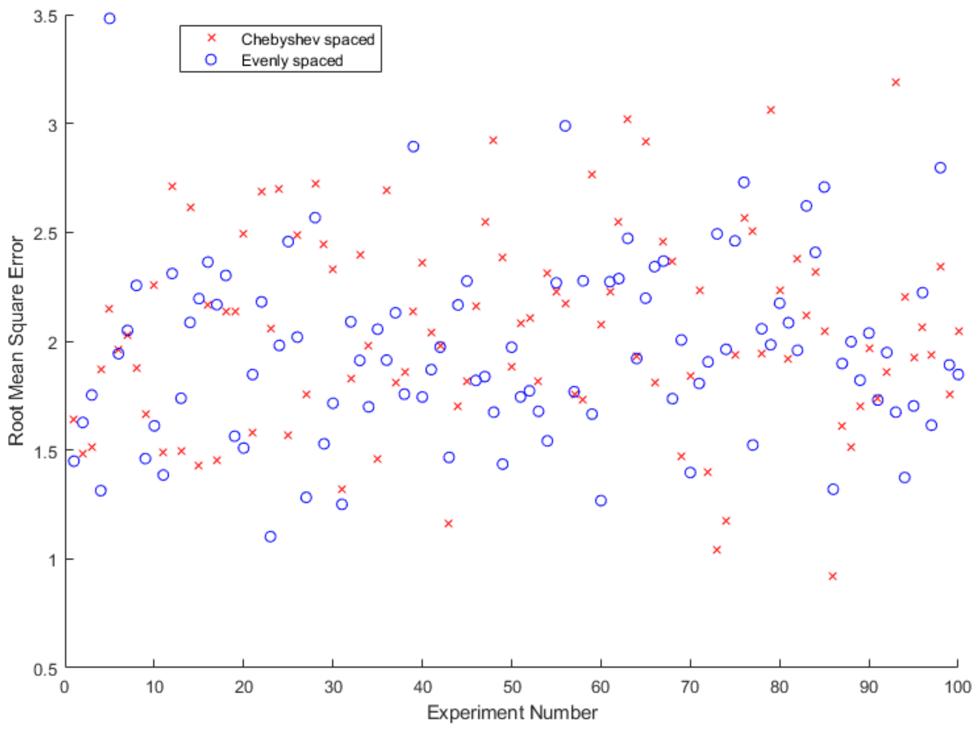
```
h =
    1 (Rejection of Null)
```

```
p =
      2.337795833246397e-05

ci =
   0.002662136466848     0.006963180529760

stats =
     tstat:  4.440483954524253
        df:  99
        sd:  0.010838139598277
```

We an see from the plot of our data that there are around seven experiment where the evenly space sampling gave the linear model a somewhat larger standard error. At between experiments 70 to 100 there is a small cluster of chebyshev spaced linear models that achieved lower than expected standard error. Again however our paired t-test showed that on average the difference between the two standard errors, with a 95% confidence interval is between $(0.00266, 0.00696)$. Given that the difference is so small, I could see it being hard to justify a meaningful difference between the two sample methods with respect to standard error.

Looking at our data for Root Mean Squared Error the paired t-test showed there was no significant statistical difference in the mean value of the Root Mean Square Error. The following is a plot of each Root Mean Square Error value for each sampling method along with the results of the paired t-test.

Figure 7: Plot of *RootMeanSquareError* Data



Console:

```
h =
     0 (Failed to reject the Null)

p =
   0.127516193682472

ci =
  -0.198505099997342    0.025219941225287

stats =
     tstat: -1.536865733110788
        df: 99
        sd: 0.563761540903470
```

To conclude I think our finding show that in regards to some metrics it can be marginally beneficial to use a chebyshev sampling method. I took on the idea for this project after our discussion of linear models. Interested in how chebyshev sampling seems to effect interpolation I figured it might be interesting to see how it would affect regression. Surprisingly from the results of this project it does seem to have a difference, albeit a miniscule one on the accuracy of our regressions.

The rest of this document contains all the relevant code used throughout this project.
Function:

```
function [x] = chebypoint(a,b,n)
% This function takes an interval [a,b],
% and the number of desired chebyshev points
% and returns the chebyshev points normalized
% on the interval [a,b].
x = [];

    for i = 0:n
        x = [x, cos(pi+((pi*i)/n))];
    end
x = ((b - a)/2).*x + (a + (b - a)/2);
end
```

Function:

```
function [Y] = gen_data_function(x,a,f)

%evaluate x on function f
y = f(x);

%generate data set with bounded b rand adjustment,
Y = a.*randn(1, length(x)) + y;

%produce data by changing with rand adjust
```

Function:

```
function [r2_1 , r2_2 , reg_even ,reg_cheby , std_error_1 ,
          std_error_2 , RMSE_1, RMSE_2] = GenerateSampleData(f ,a ,b ,n , s)


 % This function takes a sampling function , an interval [a, b]
 % and a number of nodes s, and the number of experiments n.
 % This function returns the data
 % values , y for a given x, such that y = f(x) +- norm(0 ,3).
 % It also returns all the linear model structs in a cell
 % for each type of sampling , reg_even , reg_cheby . And it
 % returns all the relevant analysis data like r^2, RMSE,
 % and standard error .



   r2_1 = [];
   r2_2 = [];

   reg_even = cell(1 ,n);
   reg_cheby = cell(1 ,n);

   std_error_1 = [];
   std_error_2 = [];

   RMSE_1 = [];
   RMSE_2 = [];

for i = 1:n
    x_1 = linspace(a ,b ,s );
    x_2 = chebypoint(a ,b ,s -1);
    x_1_isolated = setdiff(x_1 , x_2 );
    x_2_isolated = setdiff(x_2 , x_1 );
    x_intersect = intersect(x_1 , x_2 );
    y_intersect = gen_data( x_intersect ,3 , f );

    y_1_isolated = [gen_data( x_1_isolated ,3 , f )];
    y_2_isolated = [gen_data( x_2_isolated ,3 , f )];
    y_1 = sort([ y_1_isolated , y_intersect ]);
    y_2 = sort([ y_2_isolated , y_intersect ]);

    reg_1 = fitlm(x_1 , y_1 );
    reg_2 = fitlm(x_2 , y_2 );
    reg_even{i} = reg_1;
    reg_cheby{i} = reg_2;

    RMSE_1 = [RMSE_1, reg_even{1,i}.RMSE];
    RMSE_2 = [RMSE_2, reg_cheby{1,i}.RMSE];

    std_error_1 = [std_error_1 , reg_even{1,i}.Coefficients.SE(2)];
    std_error_2 = [std_error_2 , reg_cheby{1,i}.Coefficients.SE(2)];

    r2_1 = [r2_1 , reg_1.Rsquared.Ordinary];
    r2_2 = [r2_2 , reg_2.Rsquared.Ordinary];

    end


end
```