Chapter 1

1.1 Non-linear Regression

1.1.1 Polynomial Regression

Brief Introduction

The aim of Polynomial regression is to find the best polynomial fit to a given set of points. The best fit depends on order of the polynomial. Higher order polynomials are more susceptible to noise and oscillate violently. A lower order polynomial gives better generalization, but may often fail to capture diversity in the curve.A good estimate usually tries to find a balance between the two.

The polynomial regression model

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \ldots + \beta_m x_i^m + \epsilon_i \qquad (1, 2, \ldots, n)$$

can be expressed in matrix form as

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_1 & x_1^2 & \ldots & x_1^m \\ 1 & x_2 & x_2^2 & \ldots & x_2^m \\ 1 & x_3 & x_3^2 & \ldots & x_3^m \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \ldots & x_n^m \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_m \end{bmatrix} + \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \vdots \\ \varepsilon_n \end{bmatrix},$$

which when using pure matrix notation is written as

$$\vec{y} = \mathbf{X}\vec{\beta} + \vec{\varepsilon}.$$

The polynomial regression can be expressed in terms of multiple linear regression with parameters $[1, x, x^2, \ldots, x^n]$, and hence, can be solved using least square optimization algorithms with

$$\widehat{\vec{\beta}} = (\mathbf{X}^\mathsf{T}\mathbf{X})^{-1}\,\mathbf{X}^\mathsf{T}\vec{y},$$

.

Polynomial used for fit

For contactor tolerance curve, a 4th order polynomial of type

$$y = a1 + a2 * x + a_3 * x^2 + a_4 * x^3 + a_5 * x^4$$

. You can see the figure resulting from fit. The contactor curve fitting was done between y and x, with y as the independent variable. In other words, we fit a curve $x = f(y)$ instead of $y = f(x)$.

1.1.2 Exponential regression

An alternate procedure,particularly for some contactor curves, is to use exponential regression, by fitting

$$y(x) = \beta_0 e^{\beta_1 x}$$

.We then try to estimate the parameters β_0 and β_1 subject to given vector points Y^T and X^T by minimizing the objective function

$$\phi = \frac{1}{2}\sum_{i=1}^{n}(\beta_0 e^{\beta_1 x_i} - y_i)^2.$$

subject to the given vector points

$$Y^T = (y_1, y_2, \ldots, y_n)$$
$$X^T = (x_1, x_2, \ldots, x_n)$$

Exponential regression is useful for contactor curves with phase shifts of $(0^0, \ldots, 20^0)$.

1.1.3 Step Regression

A step regression aims to fit a modified step function to the given data points. The modified step function can be formulated as

$$y = \begin{cases} 0, & x <= x_{base} \\ y_{base}, & x > x_{base} \end{cases}$$

where x_{base} and y_{base} are the threshold values for the independent and dependent variables x and y respectively. A least-squares fit usually gives a very poor estimate of the function, as it contains a large number of local minima. To fix this , usually it is necessary to initialize the threshold values to near optimal values. Consequently, it is used only with rectangular sag curves.

1.2 Levenberg-Marquardt Algorithm

1.2.1 The problem statement

The primary application of the LevenbergMarquardt algorithm is in the least-squares curve fitting problem; given a set of m points (x_i, y_i), determine the parameters β to the model curve $f(x, \beta)$, so that the sum of squares of deviations is minimized.

$$\hat{\beta} = \underset{\beta}{\operatorname{argmin}}\, S(\beta) \equiv \operatorname{argmin}_\beta \sum_{i=1}^{m}[y_i - f(x_i, \boldsymbol{\beta})]^2.$$

1.2.2 Solution

Like other numeric minimization algorithms, the LevenbergMarquardt algorithm is an iterative procedure. To start a minimization, the user has to provide an initial guess for the parameter vector, β. In cases with only one minimum, an uninformed standard guess like $\beta^T = (1, 1, \ldots, 1)$ will work fine; in cases with multiple minima, the algorithm converges to the global minimum only if the initial guess is already somewhat close to the final solution.

In each iteration step, the parameter vector β is replaced by a new estimate $\beta + \delta$. To determine δ, the function $f(x_i, \beta + \delta)$

$$f(x_i, \boldsymbol{\beta} + \boldsymbol{\delta})$$

is approximated by its linearization:

$$f(x_i, \boldsymbol{\beta} + \boldsymbol{\delta}) \approx f(x_i, \boldsymbol{\beta}) + J_i\boldsymbol{\delta},$$

where

$$J_i = \frac{\partial f(x_i, \boldsymbol{\beta})}{\partial \boldsymbol{\beta}}$$

.

is the gradient (row-vector in this case) of f with respect to β. The sum $S(\beta)$ has it's minimum at zero gradient with respect to β. The first order approximation of $f(x_i, \beta + \delta)$ is given by:

$$S(\boldsymbol{\beta} + \boldsymbol{\delta}) \approx \sum_{i=1}^{m} \left(y_i - f(x_i, \boldsymbol{\beta}) - J_i\boldsymbol{\delta}\right)^2,$$

Or in vector notation,

$$S(\boldsymbol{\beta} + \boldsymbol{\delta}) \approx [\mathbf{y} - \mathbf{f}(\boldsymbol{\beta})]^T[\mathbf{y} - \mathbf{f}(\boldsymbol{\beta})] - 2[\mathbf{y} - \mathbf{f}(\boldsymbol{\beta})]^T\mathbf{J}\boldsymbol{\delta} + \boldsymbol{\delta}^T\mathbf{J}^T\mathbf{J}\boldsymbol{\delta}.$$

Taking the derivative of $S(\beta+\delta)$ with respect to δ and setting the result to zero yields

$$(\mathbf{J}^T\mathbf{J})\boldsymbol{\delta} = \mathbf{J}^T[\mathbf{y} - f(\boldsymbol{\beta})],$$

Or as a damped version,

$$(\mathbf{J}^T\mathbf{J} + \lambda\mathbf{I})\boldsymbol{\delta} = \mathbf{J}^T[\mathbf{y} - \mathbf{f}(\boldsymbol{\beta})],$$

Marquardt proposed an alternative modification to Levenberg's method to avoid slow convergence with large damping factor λ as,

$$[\mathbf{J}^T\mathbf{J} + \lambda\,\mathrm{diag}(\mathbf{J}^T\mathbf{J})]\boldsymbol{\delta} = \mathbf{J}^T[\mathbf{y} - \mathbf{f}(\boldsymbol{\beta})].$$

The value of damping factor λ is decreased or increased per iteration as follows:

1. Choose initial parameters λ_i and reduction parameter ν. The are usually drawn randomly from normal distribution.

2. Perform a step $S(\beta)$ with λ_i and check if $S(\beta)$ has reduced.

3. Now decrease λ_i as $\frac{\lambda_i}{\nu}$, and evaluate $S(\beta)$.

4. If a reduction is observed in $S(\beta)$, use the new value of $\lambda = \frac{\lambda_i}{\nu}$.

5. If step(3) doesn't produce good enough results, then multiply λ_i with ν and evaluate $S(\beta)$ again. If a reduction is observed, update λ as $lambda = \lambda_i * \nu$

6. Perform steps $(2 \dots 5)$ until minima is reached.

For more information refer Wikipedia page at Levenberg-Marquardt Algorithm

1.3 Trip calculations

1.3.1 Rectangular Sag curves

For Rectangular sag curves, two methods has been formulated.

1. In this method, the points are assigned uniform probability through tanh operator. The sag points (x_{sag}, y_{sag}) are compared, and origin shifted with respect to $(x_{tolerance}, y_{tolerance})$. The probability is then assigned as

   $$probability = \tanh((y_{sag} - y_{min}) * (y_{max} - y_{sag})) * \tanh((x_{sag} - x_{min}) * (x_{max} - x_{sag}))$$

   But it gives false positives for region

   $$x_{min} \leq x_{sag} \leq x_{max}$$

   . To mitigate this 2^{nd} procedure is proposed.

2. This method works by dividing the tolerance curves into multiple areas, as shown in 1.1. .

Figure 1.1: Rectangular Tolerance curve showing different areas of possible trip

In region I and V, the equipment trip is certain, with probability of 0 and 1 for trip respectively.

Figure 1.2: A typical sag point on region III

Region	Trip Status
I	Won't trip
II	May trip depending on voltage sag severity
III	May trip, depending on both voltage severity and duration
IV	May trip depending upon duration
V	Will trip

Table 1.1: Summary of trip by region

Regions II, IV and III are uncertain. **Our first contribution** is a method for determining the probability of trip in the regions. In region II, the probability depends on voltage sag only, whereas in region IV it depends on duration entirely. In region III, it depends on both. It is summarized in table 1.1

A uniform exponential probability distribution is assigned to them, as per equations below:

$$y = \begin{cases} 0, x_{sag} \leq x_{min}, y_{sag} \geq y_{min} \\ e^{\ln(2)*(x_{sag}-x_{min})/(x_{max}-x_{min}))-1}, x_{min} \leq x_{sag} \leq x_{max}, y_{sag} \leq y_{min} \\ e^{\ln(2)*(y_{sag}-y_{max})/(y_{min}-y_{max}))-1}, x_{sag} \geq x_{max}, y_{sag} \geq y_{max} \\ 1, x_{max} \leq x_{sag}, y_{sag} \leq y_{max} \end{cases}$$

Else

$$y = e^{\ln(2)*(x_{sag}-x_{min})/(x_{max}-x_{min}))-1} * e^{\ln(2)*(y_{sag}-y_{max})/(y_{min}-y_{max}))-1}$$

If ,

$$x_{min} \leq x_{sag} \leq x_{max}, y_{min} \geq y_{sag} \geq y_{max}$$

This region-wise division gives a better estimate of trip probability.

1.3.2 Contactor sag curves

For contactor Tolerance Curves, the probability is calculated by two methods as follows:

Method 1: The probability is calculated by taking Pseudo inverse. It is therefore, ideal for 90_o point on curve only.

Algorithm:

1. Fit polynomial curve $y = f(x)$.
2. get a sag point (x_{sag}, y_{sag})
3. Determine corresponding values (x_f, y_f) from fitted curve as

$$y_f = f(x_{sag})$$

$$x_f = (b^T * b)^{-1} * b^T * y$$

. Here, we have taken the Moore-Penrose Pseudo inverse to determine x_f.

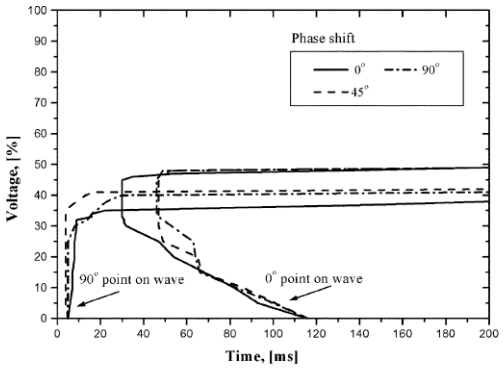

Figure 1.3: Typical Contactor curves

4. Calculate probability as

$$y = \begin{cases} 0, & x_{sag} \leq x_f, y_{sag} \geq y_f \\ \tanh(\eta * (x_f - x_{sag}) * (y_f - y_{sag})), & otherwise \end{cases}$$

   Where η is a parameter used to control probability steepness.

Method 2: The probability is calculated by estimating parameters with numerical methods or simulated annealing. The algorithm works moderately well with 0_o point on curve, and well with 90_o point on curve.

Algorithm:

1. Fit polynomial curve $y = f(x)$.
2. get a sag point (x_{sag}, y_{sag})
3. Determine corresponding values (x_f, y_f) from fitted curve as

$$y_f = f(x_{sag})$$

   x_f can only be calculated by minimizing least squares error

$$\sum_{i=0}^{n} (f(y) - y_{sag})^2$$

   by numerical methods, or simulated annealing, to find x_f, which will minimize the given error function. We used simulated annealing.

4. Calculate probability as

$$y = \begin{cases} 0, & x_{sag} \leq x_f, y_{sag} \geq y_f \\ \tanh(\eta * (x_f - x_{sag}) * (y_f - y_{sag})), & otherwise \end{cases}$$

   Where η is a parameter used to control probability steepness.

Caveats: The process is not error free, especially for 0_o point on curve. We are working on a cubic spline interpolation method, which gives better results.

1.3.3 Process Trip

Let us assume two machines with probability of trip p_1 and p_2 are connected in

1. Series: Process probability is

$$y = \max(p_1, p_2)$$

2. Parallel: Process probability is

$$y = \min(p_1, p_2)$$

For multiple series parallel machines, stack up the probabilities in cascade as per above rules. The individual probabilities can be easily determined as well.