Estimation of software project effort using nonlinear regression models

by

Lianfen Qian, Florida Atlantic University, Boca Raton, FL 33431
Qingchuan Yao, Motorola Global Software Group-United States, Plantation, FL 33322

ABSTRACT – Accurate estimation of software project effort plays an important role in software project management and is one of the most difficult empirical modeling tasks in software engineering. This paper introduces a piecewise linear regression model with a change point [9, 10] which includes the well-known COCOMO model as a special case. Our model is used to estimate software project effort using a NASA software project dataset [2] and a small software project effort dataset from Global Software Group-Florida (GSG-FL) of a local company.

For comparison, we also study three other models: Simple linear regression, COCOMO model [4] and the model based on radial basis functions [14]. Among these four models, we conclude from the empirical results that a piecewise linear model is the best one in modeling the NASA software project effort and GSG-FL dataset.

The empirical results show that one has to consider different models due to a structure change in effort for projects from medium then to large size. We also found that the coding methodology is not a significant factor in predicting the effort, after accounting project size and Development Lines (DL), but it is a significant factor, if only accounting project size, not DL.

KEY WORDS - Piecewise linear regression, change-point, software effort estimation, M-estimation, COCOMO, software project management, NASA, GSG-FL.

1. INTRODUCTION AND MOTIVATION

A regression model with a regression function that is piecewise linear over different domains of the design variable is called a piecewise linear regression model. There are two types of such models, called restricted and unrestricted. In the restricted case, the regression function is continuous at the change points but not differentiable, while in the unrestricted case it is discontinuous at the change points. Examples of important applications of these models in various scientific fields are discussed by numerous researchers. Anderson and Nelson [1] used a special type of the restricted case of the two-phase regression model [15], called the linear-plateau model, to predict crop yield based on the amount of nitrogen in the soil. Eubank [6] gave examples of a wide variety of applications where the regression function is difficult or impossible to specify, but can be approximated by simpler segmented models. Some other important examples are listed in a recent paper of Müller, and Stadtmüller [12] and references therein.

Though the piecewise linear regression models have been widely researched and used as mentioned above. They have not received much attention linked to software engineering community. In this paper, we present this link to one of the real applications on software engineering modeling: that is, to model the well-known NASA software project data set from [2]. This data set contains the development effort (Y), and development lines (DL) of code and methodology (ME). DL is in KLOC and Y is in man-months. The dataset for the 18 projects is reproduced in Table 1. Here, ME is a composite measure of methodologies employed in this NASA software environment. The main reason to choose this dataset is for comparing our model with the recent developed Radial Basis Function (RBF) model in [14] using the same dataset. We show that our model is an excellent candidate in modeling the software project effort.

Table 1. NASA Software Project Data [2]

<table>
<thead>
<tr>
<th>Project No.</th>
<th>Independent Variable</th>
<th>Dependent Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DL</td>
<td>ME</td>
</tr>
<tr>
<td>1</td>
<td>90.2</td>
<td>30</td>
</tr>
<tr>
<td>2</td>
<td>46.2</td>
<td>20</td>
</tr>
<tr>
<td>3</td>
<td>46.5</td>
<td>19</td>
</tr>
<tr>
<td>4</td>
<td>54.5</td>
<td>20</td>
</tr>
<tr>
<td>5</td>
<td>31.1</td>
<td>35</td>
</tr>
<tr>
<td>6</td>
<td>67.5</td>
<td>29</td>
</tr>
<tr>
<td>7</td>
<td>12.8</td>
<td>26</td>
</tr>
<tr>
<td>8</td>
<td>10.5</td>
<td>34</td>
</tr>
<tr>
<td>9</td>
<td>21.5</td>
<td>31</td>
</tr>
<tr>
<td>10</td>
<td>3.1</td>
<td>26</td>
</tr>
<tr>
<td>11</td>
<td>4.2</td>
<td>19</td>
</tr>
<tr>
<td>12</td>
<td>7.8</td>
<td>31</td>
</tr>
<tr>
<td>13</td>
<td>2.1</td>
<td>28</td>
</tr>
<tr>
<td>14</td>
<td>5.0</td>
<td>29</td>
</tr>
<tr>
<td>15</td>
<td>78.6</td>
<td>35</td>
</tr>
<tr>
<td>16</td>
<td>9.7</td>
<td>27</td>
</tr>
<tr>
<td>17</td>
<td>12.5</td>
<td>27</td>
</tr>
<tr>
<td>18</td>
<td>100.8</td>
<td>34</td>
</tr>
</tbody>
</table>
Numerous research articles have been focusing on the software cost estimation. A detail empirical evaluation of several models (SLIM, COCOMO, Function Points, and ESTIMACS) was presented in [8] for a dataset of 15 large complete business data processing projects, collected by a national computer consulting and services firm specializing in the design and development of data processing software. They conclude that “all of the models tested failed to sufficiently reflect the underlying factors affecting productivity. Further research will be required to develop understanding in this area”. Shepperd and Schofield [13] provide a recent survey of such models including models based on simple linear regression. They also point out that accurate project effort prediction is an important goal for the software engineering community.

As [13] points out, an important aspect of any software development project is to know how much it will cost. In most cases the major cost factor is labor. For this reason, estimation of software project effort is one of the most important empirical modeling tasks in software engineering, as indicated by the large number of models developed over the past twenty years.

Recently, Shin and Goel [14] studied the modeling of the software project effort using radial basis functions (RBF). They used the well-known NASA software project dataset to validate the model. The method proposed in [14] is complicated and contains many parameters to be estimated. It uses the sum of a weighted $m$ basis functions to estimate the regression function. They use a set of $m$ basis functions each having two unknown parameters. Thus, the model contains $2m + m = 3m$ parameters to be estimated for a fixed $m$. If the model contains a constant term, then, they need to estimate $3m + 1$ parameters. The number $m$ itself needs to be estimated also.

A simple scatterplot (the dots in Figure 1) of the NASA software dataset shows that the data appear to follow a two-phase linear pattern for the software project effort against the lines of code (DL). The regressor DL does not change significantly at any point, but the dependent variable Effort shows a structure change at some value of the regressor. Based on this observation, it is appropriate to analyze the data using two-phase linear regression models with an unknown change point. In this paper, we pursue this approach using two estimation methods, Least Squared Estimator (LSE) and Least Absolute Deviation Estimator (LADE), for the model parameters. For comparison, we also fit the data set by the simple linear regression model using the least squared estimation method. All three fitted curves are drawn in Figure 1. See Appendix A for the definitions of least squared and least absolute deviation estimations.

The paper is organized as follows. Section 2 describes the general piecewise linear regression model and a computational scheme for $M$-estimators. Section 3 reports a simulation study to verify the finite sample properties. Section 4 reports the empirical results from the multiple-phase linear regression modeling for the well-known NASA software project effort dataset. These empirical results then are compared with the empirical results from the simple linear regression, the COCOMO model and the model using radial basis functions. Section 5 reports the effectiveness of using the two-phase regression in modeling a GSG-FL software project effort dataset from a local company. We conclude the findings in Section 6. Section 7 gives some of the future research directions.

One of the motivations for writing this paper is to introduce the piecewise linear regression models into the software engineering community. Although in this paper we apply the models only to the estimation of the software project effort, the models can easily be applied to other modeling fields in software engineering such as predicting software faults and the process control for software inspection.

2. PIECEWISE LINEAR REGRESSION

In a simple linear regression model (for example, the dotted dash line in Figure 1), one uses a single straight line to model a given dataset. In a general piecewise linear regression model, one uses several pieces of straight line segments with unknown change-points (break points or thresholds) to model some datasets. For example, the solid (or dotted) line segments in Figure 1 is a two-piece linear regression model in...
modeling the well known NASA software project effort dataset. In general, piecewise linear regression models may have many pieces of linear segments with many unknown change-points. In estimating model parameters including the change-point estimation, we use two estimation methods: least squared estimator (LSE) and least absolute deviation estimator (LAD). Detailed mathematical representation and the M-estimation methods are given in Appendix A.

2.1 SPECIAL CASE 1: COCOMO MODELS

Let KDSI be the number of thousands of delivered source instructions, and MM be the number of man-months required to develop the most common type of software product. Denote \( X = \ln(\text{KDSI}) \), and \( Y = \ln(\text{MM}) \). Then the basic COCOMO model for five modes of software projects is a special case of the model (A.1 in Appendix A) for \( p=1 \), \( q=5 \). The five thresholds or change points classify the software projects by the size: Small (2KDSI), Intermediate (8KDSI), Medium (32 KDSI), Large (128 KDSI) and Very Large (512 KDSI). The basic COCOMO model assumes that the sorting is fixed for all types of software projects, regardless of the companies where the software products are produced. Those thresholds or change points are assumed known before modeling and fixed. But in real life, those change points are not known, and depend on the producers of the software products, thus they need to be estimated also.

Since software effort may depend on other variables, the basic COCOMO model is not enough to capture the variability in effort by using DSI only. Thus, the intermediate COCOMO model introduces additional 15 predictors to capture variability in effort not explained by DSI. In the model (A.1), let \( p=16 \), \( q=5 \), \( X_i = \ln(\text{KDSI}) \), \( \{X_i\} \), \( i=2, \ldots, 16 \), are the additional 15 predictor variables, and \( Y = \ln(\text{MM}) \), then the model (A.1) reduces to the intermediate COCOMO model [4].

The Special Case 2: Two-Phase Model is included in Appendix A.

2.2 FLOW CHART OF THE ALGORITHM

A flow chart of the proposed model estimation algorithm is created in Chart 1 below. The proposed two-stage technique for grip search and minimizing the M-function to obtain the estimators including the change-points is described in Section 3.

3. SIMULATION STUDY

In this section, we conduct Monte Carlo simulation to study finite sample properties of these estimators. We first discuss how to implement the estimation procedures, then illustrate the performance of these estimators using different assessing criteria. For example, we use \( \rho(x) = x^2 \) and \( \rho(x) = |x| \) for comparison of different estimation methods. The corresponding estimators for these two dispersion functions are the Least Squared (LS) and Least Absolute Deviation (LAD) estimators, respectively. The two estimation methods will be used in Section 4 to model the well-known NASA software project effort dataset and in Section 5 to model a GSG-FL software project effort dataset from a local company.

For the LS and LAD estimators, we simulate samples for various error densities from the following simple model:

\[
Y_i = (a_0+X_i)I(X_i \leq r) + (b_0+b_i)X_iI(X_i > r) + \epsilon_i, \\
i = 1, \ldots, n, \tag{3.5}
\]

where \( \{X_i\} \) is a random sample from the standard normal distribution and \( \{\epsilon_i\} \) is from various error densities.

From model (3.5), the regression function in the objective function for minimization is a linear function in coefficient parameters and nonlinear in change-point parameter. Thus, we use a two-stage technique to obtain the estimators. We describe the scheme for LAD estimator first.

Step 1: For any given change point \( s \), we compute the LAD estimators for coefficient parameters. Our program on the LAD estimation is based on the Barrodale-Roberts (1973) algorithm, a specialized linear programming algorithm.

Step 2: We substitute the LAD estimators from step one for the coefficient parameters back to the objective function to get a profile objective function. Then we seek the minimum of the profile objective function in \( s \) over the order statistics of the sample of \( X \). The sample order statistic that gives rise to the minimum of the profile objective function in \( s \) is the estimate of the change point, and the associated LAD estimates of other parameters are estimates of the coefficient parameters.

The estimates using the LS estimation are obtained by the same way and are used for comparison.

The samples were generated from model (3.5) with \( \theta = (a_0, a_i, b_0, b_i, r) = (0.5, -1, -0.7, 1, 0) \), and for two different error distributions: \( N(0,1) \) and Double exponential Dexp(0,1). The sample sizes used are 100, 200 and 500. For each sample size, the results are based on 500 replications. The function \( \rho = -\log g \), where \( g \) is the density function of the error \( \epsilon \). When \( g \) is normal and Dexp(0,1), the estimations correspond to LS and
Table 2 shows the empirical coverages of the estimates of the true parameters within two standard deviations from the sample means for n=100, 200 and 500, respectively. One observes that the estimates have very high percent coverage within two standard deviations from the simulated sample means. For the change-point estimator, it is seen that as the sample size increases, the coverage increases dramatically with the coverage range from 91.4% to 98.6%, while the range of the coverage for the coefficient parameters is from 93.6% to 97.6%. This also confirms the \( n \)-consistency of the change-point estimator.

4. MODELING THE NASA DATASET

4.1 MODELING EFFORT BASED ON DL

In this section, we model the NASA software data set using a two-phase linear regression model. The data set contains Effort and DL for 18 software projects. Table 3 lists the parameter estimators for the two-phase linear regression for the software effort data. Through a grid search, we found \( \hat{\beta}_n = 31.1 \) and \((\hat{a}_0, \hat{a}_1, \hat{b}_0, \hat{b}_1)\) as reported in the Table 3. Out of the 18 projects, there are 11 projects having the effort less than or equal to 31.1 and 7 larger than 31.1. This indicates that
there is a structure change in Effort when DL exceeds 31.1 thousand lines. It is also useful to look at the estimated standard deviation for each piece given by

\[ s_1 = \left( \frac{1}{n} \sum_{i=1}^{n} (Y_i - \hat{a}_0 - \hat{a}_1 X_i)^2 I(X_i \leq 31.1) \right)^{1/2}, \]

\[ s_2 = \left( \frac{1}{n} \sum_{i=1}^{n} (Y_i - \hat{b}_0 - \hat{b}_1 X_i)^2 I(X_i > 31.1) \right)^{1/2}. \]

Notice that from Table 3, for the LAD and the LS estimators, both \( s_1 \) and \( s_2 \) are smaller using two-phase regression than the single square-root mean square error 10.13 using simple linear regression. The overall roots of mean square errors for two-phase regression using LAD and LS are 5.67 and 5.37, respectively. One observes further that the second piece has larger variability in errors.

Table 3. The parameter estimates and estimated standard deviation for each piece

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Change-point</th>
<th>( a_0 )</th>
<th>( a_1 )</th>
<th>( b_0 )</th>
<th>( b_1 )</th>
<th>( s_1 )</th>
<th>( s_2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAD</td>
<td>31.1</td>
<td>3.38</td>
<td>1.17</td>
<td>46.66</td>
<td>.77</td>
<td>2.92</td>
<td>8.32</td>
</tr>
<tr>
<td>LS</td>
<td>31.1</td>
<td>2.52</td>
<td>1.21</td>
<td>46.99</td>
<td>.80</td>
<td>2.87</td>
<td>7.83</td>
</tr>
</tbody>
</table>

From the big difference between the estimated \( a_0 \) and \( b_0 \) values in Table 3, one observes that much more effort for Medium, Large or Very Large software projects is needed than those for Small and Intermediate projects at NASA. This more effort may include more preparation, understanding, designing, communicating and managing for the “large” project than for the “small” project.

One also observes that the rate of change (\( a_1 \) and \( b_1 \)) in Effort vs DL is relatively smaller for the Medium, Large or Very Large software projects than the one for Small and Intermediate projects at NASA. It means that the productivity (the rate of change in DL vs Effort) is relatively larger for the Medium, Large or Very Large software projects than the one for Small and Intermediate projects at NASA.

4.2 MODEL ASSESSMENT

One of the criteria for assessment of the performance of model fitting used in the software engineering literature is the mean magnitude of relative error defined as

\[ \overline{MRE} = \frac{1}{n} \sum_{i=1}^{n} \left| \frac{Y_i - \hat{Y}_i}{\hat{Y}_i} \right| = \frac{1}{n} \sum_{i=1}^{n} MRE_i, \]

where \( \hat{Y}_i \) is the fitted value of \( Y_i \). The implicit assumption in this summary measure is that the seriousness of the absolute error is proportional to the size of the observations. A companion summary measure related to \( MRE \) is the prediction at level \( \hat{p} \), \( PRED(\hat{p}) = \frac{\hat{Y}_i}{\hat{p}} \), where \( k \) is the number of observations whose \( MRE \) is less than or equal to \( \hat{p} \), and \( n \) is the sample size. These two measures of goodness-of-fit are recommended by Conte, Dunsmore and Shen [5] in software engineering modeling. They also recommended an upper limit of 25% for \( MRE \) and a lower limit of 75% for \( PRED(\hat{p}) \) as the values to be acceptable for effort estimation models.

We use \( MRE \) and \( PRED(\hat{p}) \) as our model selection criteria. To see the stability of the models, the leave-one-out cross-validation (LOOCV) method [14] is applied. Table 4 reports the two measures \( MRE \) and \( PRED(\hat{p}) \) for both training and generalization data using all four models: Two-phase, Linear, COCOMO and SG (RBF). Comparing the results, one concludes that Two-phase is the best choice in modeling the NASA software project effort. The SG column in Table 4 is cited from Table 6 in Shin and Goel [14] under their best choices of \( \delta = 1\% \) or \( \delta = 2\% \) and \( \sigma = .55 \) and \( m = 3 \). Furthermore, the two-phase linear regression using LAD estimation method is the most robust model.

Table 4. Model comparisons

<table>
<thead>
<tr>
<th>Model</th>
<th>Method</th>
<th>Two-phase</th>
<th>Linear</th>
<th>COCOMO</th>
<th>SG</th>
</tr>
</thead>
<tbody>
<tr>
<td>MRE</td>
<td>T</td>
<td>.128</td>
<td>.126</td>
<td>.211</td>
<td>.145</td>
</tr>
<tr>
<td></td>
<td>G</td>
<td>.153</td>
<td>.151</td>
<td>.233</td>
<td>.216</td>
</tr>
<tr>
<td>P(25)</td>
<td>T</td>
<td>88.9%</td>
<td>83.3%</td>
<td>77.8%</td>
<td>83.3%</td>
</tr>
<tr>
<td></td>
<td>G</td>
<td>88.9%</td>
<td>83.3%</td>
<td>72.2%</td>
<td>77.8%</td>
</tr>
</tbody>
</table>

*T=Training, G=Generalization, P(25)=PRED(25)

Based on \( MRE \) and \( PRED(\hat{p}) \), one observes that the two-phase regression is much better than the simple linear regression model. The mean magnitude of relative error is reduced by about half using the two-phase regression model. Within two-phase regression, the two estimators using the LAD and the LS estimators are comparable and the LAD estimator is little better in prediction than the LS estimator. The relative efficiencies (relative efficiency is the ratio of mean square errors) of LS and LAD estimators using two-phase regression with respect to simple regression are 3.56 and 3.19, respectively. The two phases classify the sizes of the software projects. For the NASA software team, the change-point is 31.1 thousand lines. Thus, NASA projects
with less than or equal 31.1 KLOC are considered as “small” projects, while projects with larger than 31.1 thousand KLOC are considered as “large” projects. “Small” and “large” projects require significantly different effort per KLOC.

4.3 MODELING EFFORT ON DL AND ME

Now, we inspect the relationship between Effort and (DL, ME). Figure 2 shows the relationship between Effort and ME separated for small projects (DL ≤ 31.1) and large projects (DL > 31.1). Denote Ind = I(DL ≤ 31.1).

One observes that there is a weak linear relationship between Effort and ME for both modes of projects (small or large).

Thus, we use the two-phase regression model including ME and the indicator of the size of the projects Ind = I(DL ≤ 31.1) to get the regression equation as follows:

\[
\text{Effort} = (-14.3 + 1.05 \text{ ME}) I(DL \leq 31.1) \\
+ (49.7 + 1.97 \text{ ME}) I(DL > 31.1).
\] (4.1)

The adjusted coefficient of determination of (4.1) is 92.9% and the root mean squared error is 12.21. The standard deviations of the coefficient estimators of ME, I(DL ≤ 31.1) and ME*I(DL ≤ 31.1) are .72, 32.12 and 1.14, respectively. The corresponding p-values are .016, .066 and .434, hence for separated size of software projects, ME is a significant predictor if not including the DL predictor.

<table>
<thead>
<tr>
<th>Predictor</th>
<th>Estimate</th>
<th>SEE</th>
<th>P-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constant</td>
<td>49.72</td>
<td>19.79</td>
<td>.025</td>
</tr>
<tr>
<td>DL</td>
<td>1.18</td>
<td>0.24</td>
<td>.000</td>
</tr>
<tr>
<td>ME</td>
<td>-1.31</td>
<td>0.74</td>
<td>.103</td>
</tr>
<tr>
<td>Ind</td>
<td>-38.96</td>
<td>32.12</td>
<td>.066</td>
</tr>
<tr>
<td>ME*Ind</td>
<td>0.21</td>
<td>0.35</td>
<td>.564</td>
</tr>
<tr>
<td>ME*Ind</td>
<td>0.74</td>
<td>0.90</td>
<td>.428</td>
</tr>
</tbody>
</table>

5. MODELING GSG-FL DATASET

In this section, we model the current available software data set from local company GSG-FL projects. The data set is included in Table 7 below:

<table>
<thead>
<tr>
<th>Project No.</th>
<th>KLOC</th>
<th>Staff Weeks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10.634</td>
<td>15.4</td>
</tr>
<tr>
<td>2</td>
<td>3.610</td>
<td>19.6</td>
</tr>
<tr>
<td>3</td>
<td>1.087</td>
<td>11.6</td>
</tr>
</tbody>
</table>
To illustrate the powerfulness of the two-phase modeling, we model this dataset for the natural log-transformed effort in weeks against the natural log-transformed KLOC. Also, we compare the two-phase regression model with the basic COCOMO model. Table 8 reports the estimated coefficient estimates and the estimated change-point which is corresponding to the size 1.726 KLOC of the software project. Notice that the natural logarithm $\ln(1.726) = 0.546$.

Table 8. GSG-FL dataset parameter estimates

<table>
<thead>
<tr>
<th>Model</th>
<th>Method</th>
<th>Change-point</th>
<th>$a_0$</th>
<th>$a_1$</th>
<th>$b_0$</th>
<th>$b_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-Phase</td>
<td>LAD</td>
<td>.546</td>
<td>2.34</td>
<td>.65</td>
<td>2.15</td>
<td>.25</td>
</tr>
<tr>
<td></td>
<td>LS</td>
<td>.546</td>
<td>2.34</td>
<td>.65</td>
<td>1.96</td>
<td>.38</td>
</tr>
<tr>
<td></td>
<td>COCOMO</td>
<td>2.09</td>
<td>.35</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 3 shows the graphic visualization of the fitted models using the two-phase regression models with LAD and LS estimations and the COCOMO model for the GSG-FL software project effort dataset.

To assess the goodness-of-fit of the models, we still use $MRE$ and $PRED(.25)$ criteria. Table 9 reports the results.

Table 9. Model assessment and comparisons for GSG-FL dataset

<table>
<thead>
<tr>
<th>Model</th>
<th>Two-phase</th>
<th>COCOMO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>LAD</td>
<td>LS</td>
</tr>
<tr>
<td>$MRE$</td>
<td>.149</td>
<td>.164</td>
</tr>
<tr>
<td>$PRED(.25)$</td>
<td>75%</td>
<td>75%</td>
</tr>
</tbody>
</table>

As we can see from Table 9, the two-phase regression is very promising even for a small dataset. The empirical results show that the two-phase model is much more accurate and effective than the COCOMO model.

6. CONCLUSION

In this paper, we have proposed multiple phase linear regression models for modeling software effort estimation using the DL and other predictors. Through modeling the well-known NASA software project dataset, it shows that separation of the sizes of the software projects is necessary in modeling the effort. The separation criterion (change-point) is
also estimated through the model. The estimation methods used in this paper are least squared and least absolute deviation estimations. The latter one is preferred since it is robust to outliers.

It also found that the methodology used in coding is not a significant factor in predicting the effort when the separation and DL are included already, but it is significant factor if DL is not included even though the separation is done. This implies that the DL and the ME have multicolinearity.

It should be pointed out that the well-known basic and intermediate COCOMO models are the special case of our model. For the NASA software project dataset, we compared four models: Two-phase linear regression, simple linear regression, COCOMO models and models based on RBF [14]. For the GSG-FL software project data set, we compared our model with the COCOMO model. The conclusion is that our piecewise linear regression model is the best in modeling the well-known NASA software project dataset and in modeling the small GSG-FL software project dataset which has only 8 pairs of data.

Another important feature is that the multiple-phase linear regression model is simple as soon as the change-point is estimated. It captures high variability in the effort and with small root mean square error.

7. FUTURE WORK

Some of the future research directions may include the following:

1. Apply the general piecewise linear regression models to other software engineering fields such as predicting software errors and defects, and the process control for software inspection.
2. The natural extension of the piecewise linear regression is the piecewise non-linear regression model defined as below:

\[
f(x, \theta_1, \ldots, \theta_{q+1}) = \begin{cases} 
  f_1(x, \theta_1), & \text{if } x \in \mathbb{R}_1 \\
  \ldots \\
  f_{q+1}(x, \theta_{q+1}), & \text{if } x \in \mathbb{R}_{q+1}
\end{cases}
\]

where \( f_i, i = 1, \ldots, q + 1 \) are nonlinear functions in \( \theta_i, i = 1, \ldots, q + 1 \). This model allows \( q \) change points and each piece is a nonlinear function. Thus, the piecewise linear and COCOMO models are the special cases of the above model.

8. ACKNOWLEDGEMENTS

The authors would sincerely like to thank the second author’s manager, Charles Schultz, for carefully reading the first draft and giving his valuable comments that improve the presentation of the paper. Also, the authors would sincerely like to thank the anonymous reviewers for the valuable comments. The authors would also like to thank Valquiria Cordeiro for providing the GSG-FL dataset.

9. REFERENCES


A.2 SPECIAL CASE 2: TWO-PHASE MODEL

For the simplicity of the presentation, we consider the case for \( q = 1 \), which is reduced to the so called two-phase linear regression model. Denote the single change point by \( r \). Let

\[
f(x, \vartheta) = (\alpha_0 + \alpha_1 x)(x \leq s) + (\beta_0 + \beta_1 x)(x > s),
\]

where \( x \in \mathbb{R}, \vartheta = (\alpha_0, \alpha_1, \beta_0, \beta_1, s) \in \mathbb{R}^5 \), be the two-phase linear regression function and

\[
\vartheta = (a_0, a_1, b_0, b_1, r) \in \mathbb{R}^5.
\]

Then, the two-phase linear regression model is

\[
Y_i - f(X_i, \vartheta) = \varepsilon_i, \quad i = 1, \ldots, n, \tag{A.2}
\]

where \( \varepsilon_1, \ldots, \varepsilon_n \) are independent identically distributed (i.i.d.) random variables, and \( (X_i, Y_i), i = 1, \ldots, n \), are the observations. The jump size at the true jump-point \( r \) in the regression function \( f \) is given by \( J = b_0 - a_0 + r (b_1 - a_1) \). We shall make the usual identifiability assumption that the two line segments are different and that \( J \) is fixed and non-zero, i.e.,

\[
J \neq 0. \tag{A.3}
\]

It is convenient to write \( f(x, \vartheta) = f_j(x, \vartheta_j) \), for \( \vartheta = (\vartheta_j, s) \) with \( \vartheta_j \in \mathbb{R}^q, \quad s \in \mathbb{R} \), and refer to \( \vartheta_j \) and \( s \) as the coefficient and the change point parameters, respectively. Let

\[
\dot{f}_j(x) = \frac{\partial f_j(x, \vartheta_j)}{\partial \vartheta_j} = (I(x \leq s), x I(x \leq s), I(x > s), x I(x > s))',
\]

be the vector of partial derivatives of \( f_j(x, \vartheta_j) \) with respect to \( \vartheta_j \). Observe that \( f(x, \vartheta) = \vartheta_j' \dot{f}_j(x) \).

A.3 M-ESTIMATORS

To define M-estimators of \( \vartheta \), let \( \bar{\mathbb{R}} = \mathbb{R} \cup \{-\infty, \infty\} \). The set \( \bar{\mathbb{R}} \) is compact under the metric \( d(x, y) = |\arctan x - \arctan y| \), \( x, y \in \mathbb{R} \). Throughout we assume that \( \vartheta \) is an interior point of the parameter space \( \Theta = K \times \bar{\mathbb{R}} \) for a known compact set \( K \) in \( \mathbb{R}^q \). A typical point in \( \Theta \) will be denoted by \( \vartheta = (\vartheta_j, \vartheta_j', s) = (\vartheta_j, s)' \). Define the M-process corresponding to a function \( \rho : \mathbb{R} \to [0, \infty) \) by

\[
M_n(\vartheta) = \sum_{i=1}^n \rho(Y_i - \vartheta_j' \dot{f}_j(X_i)), \quad \vartheta_j \in \mathbb{R}^q, s \in \mathbb{R},
\]
A measurable map \( \hat{\theta}_n = \hat{\theta}_n ((X_1,Y_1), \ldots, (X_n,Y_n)) : \mathbb{R}^{2n} \to \Theta \), is an \( M \)-estimator, if

\[
M_n (\hat{\theta}_n) = \inf_{\theta \in \Theta} M_n (\theta), \quad a.s.
\]

Often we shall write \( M_n (\theta_1,s) \) for \( M_n (\theta) \).

Note that the function \( M_n (\theta_1,s) \) is not continuous in \( s \), but because of (2.3), for each \( \theta_1 \), \( M_n (\theta_1,s) \) as a function of \( s \) is constant on the intervals \( (X_{i-1}, X_i), 1 \leq i \leq n+1 \), where \( \{X_0, 1 \leq i \leq n\} \) are the ordered design variables with \( X_0 = \infty, X_{n+1} = \infty \). Thus, to compute the \( M \)-estimator, we proceed as follows: First, for each fixed \( s \), obtain the minimizer \( \hat{\theta}_1(s) \) of \( M_n (\theta_1,s) \) with respect to \( \theta_1 \) over \( K \). Notice that \( \hat{\theta}_1(s) \) is constant in \( s \) over any interval of two consecutive ordered \( X_i \)'s and that the profile \( M \)-process \( M_n (\hat{\theta}_1(s), s) \) has only finite number of possible values with the jump points located at \( X_0 \)'s. At the second stage, compute minimizer \( \hat{\hat{\theta}}_n \) of \( M_n (\hat{\theta}_1(s), s) \) with respect to \( s \) over \( \{X_0, 1 \leq i \leq n\} \). To make it unique we take this minimizer to be the left end point of the interval over which it is obtained. The associated \( \hat{\theta}_1(s) = \hat{\hat{\theta}}_1 \) becomes the \( M \)-estimator of \( \theta_1 \). Then the estimator \( \hat{\theta}_n = \left( \hat{\hat{\theta}}_1 \right) \) is the \( M \)-estimator of the underlying parameter \( \theta \).

This method of computation is used in the simulation and in the real application presented in sections 3 and 4 below.

When \( \rho(x) = x^2 \) and \( \rho(x) = |x| \), the estimator \( \hat{\rho}_n \) is called a least squared estimator and least absolute deviation estimator of \( \rho \), respectively.

### A.4 CURRENT RESEARCH STATUS

Many results about various inference procedures pertaining to the change point have been developed when the regressor is non-random or random design. Maronna and Yohai [11] studies the piecewise linear regression model with a random regressor variable. It was noted that the random regressor model is appropriate when the dependent variable may undergo a systematic change at some unknown points, while the regressor variable does not change significantly and affects the dependent variable through the correlation between the independent and dependent variables. Koul and Qian [9], Koul, Qin and Surgailis [10] established the consistency and the limiting distribution for two methods (MLE and M-estimation) of parameter estimations in two-piece random design regression models with the fixed jump size and for a class of error densities that excludes the double exponential and such non-smooth densities. It has been proved in [9] and [10] that the change point estimator is \( n \)-consistent (the rate of convergence is \( n \), the sample size) and the asymptotic distribution of the standardized change point estimator is related to a compound Poisson process. A process \( X(t) \) is called compound Poisson if it is a sum of \( N \) mutually independent variables with common distribution, where \( N \) is a Poisson random variable with mean \( \lambda t \), i.e.,

\[
P(N=n) = e^{-\lambda t} \frac{((\lambda t))^n}{n!}.
\]

Please see [7] for details.

### 11. APPENDIX B

#### MULTIPLE LINEAR REGRESSION

**Overview**

Multiple linear regression is used to account for (predict) the variance in a dependent variable, based on linear combinations of independent variable, dichotomous, or dummy variables. (Dummy variables are values of a categorical variable treated as if they were separate variables, usually coded for 1 if present in a case or 0 if absent, such as Ind=1 or Ind=0 for a small or large software project, respectively.)

The multiple linear regression equation takes the form

\[
y = b_1 x_1 + b_2 x_2 + \ldots + b_p x_p + c.
\]

The \( b \)'s are the regression coefficients, representing the amount the dependent variable changes when the independent variable changes 1 unit holding all other variables constant. The \( c \) is the constant or intercept, where the regression line intercepts the y axis, representing the amount the dependent variable will be when all the independent variables are 0. \( y \) is called the estimated or fitted dependent. Equations such as that above, with no interaction effects (see below), are called main effects models.

Associated with multiple linear regression is \( R^2 \), multiple correlation, which is the percent of variance in the dependent variable explained collectively by all of the independent variables. The adjusted \( R^2_{adj} \) is computed in taking the number of independent variables in the model into consideration. Suppose that \( SST \) is the total variability in \( y \), \( SSE \) is the variability left unexplained by \( x_1, \ldots, x_p \), or due to the error. Then

\[
R^2_{adj} = 1 - \frac{p \cdot SSE}{(n - p - 1)SST}.
\]
Multiple linear regression with a dummy variable yields two different models for each value of the dummy variable. For example, when \( \text{Ind}=1 \), the fitted equation is the model good for small software projects, and \( \text{Ind}=0 \) is good for large software projects.

**Key Terms and Concepts**

**Predicted values**, also called **fitted values**, are the values of each case based on using the regression equation for all cases in the analysis.

**Residuals** are the difference between the observed values and those predicted by the regression equation.

**Dummy variables** are a way of adding the values of a nominal or ordinal variable to a regression equation. Each value of the categorical independent except one is entered as a dichotomy (For example: \( \text{Ind} = 1 \) if software project is small with \( DL \leq 31.1 \), otherwise 0); One class must be left out to prevent perfect multicollinearity in the model. Thus for two modes of software project, we only need to introduce one dummy variable, \( \text{Ind} \).

**Interaction effects** are sometimes called **moderator effects** because the interacting third variable which changes the relation between two original variables is a moderator variable which moderates the original relationship. For instance, the relation between Effort and DL may be moderated depending on the modes of software projects.

**Interaction terms** may be added to the model to incorporate the joint effect of two variables (ex., DL and Ind) on a dependent variable (ex., Effort) over and above their separate effects. One adds interaction terms to the model as crossproducts of the standardized independents and/or dummy independents, typically placing them after the simple "main effects" independent variables. Crossproduct interaction terms may be highly correlated (multicollinear - see below) with the corresponding simple independent variables in the regression equation, creating problems with assessing the relative importance of main effects and interaction effects.

**Note**: Because of possible multicollinearity, it may well be desirable to use **centered variables** (where one has subtracted the mean from each datum) -- a transformation which often reduces multicollinearity.

The significance of an interaction effect is the same as for any other variable, except in the case of a set of dummy variables representing a single ordinal variable. When an ordinal variable has been entered as a set of dummy variables, the interaction of another variable with the ordinal variable will involve multiple interaction terms. In this case the F-test of the significance of the interaction of the two variables is the significance of the change of R-square of the equation with the interaction terms and the equation without the set of terms associated with the ordinal variable.

The **regression coefficient**, \( b \), is the average amount the dependent increases when the independent increases one unit and other independents are held constant. Put another way, the \( b \) coefficient is the slope of the regression line: the larger the \( b \), the steeper the slope, the more the dependent changes for each unit change in the independent. The \( b \) coefficient is the unstandardized simple regression coefficient for the case of one independent. When there are two or more independents, the \( b \) coefficient is a **partial regression coefficient**, though it is common simply to call it a "regression coefficient" also.

**Interpreting \( b \) for dummy variables**. For \( b \) coefficients for dummy variables which have been binary coded (the usual \( 1=\text{present, } 0=\text{not present method discussed above} \)), \( b \) is relative to the **reference category** (the category left out). Thus for the set of dummy variables for "project modes," assuming "Large" is the reference category and Effort is the dependent, a \( b \) of \(-64.01 \) in Table 5 for the dummy "Small" means that the expected effort for the small project is 64.01 less than the average of "Large" projects.

**Dynamic inference** is drawing the interpretation that the dependent changes \( b \) units because the independent changes one unit. That is, one assumes that there is a change process (a dynamic) which directly relates unit changes in \( x \) to \( b \) changes in \( y \). This assumption implies two further assumptions which may or may not be true: (1) \( b \) is stable for all subsamples or the population (cross-unit invariance) and thus is not an artificial average which is often unrepresentative of particular groups; and (2) \( b \) is stable across time when later re-samples of the population are taken (cross-time invariance).

**Standard Error of Estimate (SEE), confidence intervals, and prediction intervals**. In regression the confidence refers to more than one thing. Note the confidence and prediction intervals will improve (narrow) if sample size is increased, or the confidence level is decreased (For example, from 95% to 90%).

- **The confidence interval of the regression coefficient**. Based on t-tests, the confidence interval is the plus/minus range around the observed sample regression coefficient, within which we can be, say, 95% confident the real regression coefficient for the population regression lies. Confidence limits are relevant only to random sample datasets. If the confidence interval includes 0, then there is no significant linear relationship between \( x \) and \( y \). We
then do not reject the null hypothesis that $x$ is independent of $y$.

- **The confidence interval of $y$.** For the 95% confidence level, the confidence interval of the mean function of $y$ at $x$ is the estimated value plus/minus $1.96$ times the standard error (SEE) of the estimate. SEE is $\text{SQRT}(\text{RSS}/\text{df})$, where RSS is the sum of squares of residuals and df is degrees of freedom $= (n - p - 1)$, where $n$ is sample size and $p$ is the number of independent variables. Some 95 times out of a hundred, the true mean of $y$ will be within the confidence limits around the observed mean of $n$ sampled cases. Note the confidence interval of $y$ deals with the mean, not an individual case of $y$.

- **The prediction interval of $y$.** For the 95% confidence level, the prediction interval of an individual $y$ is the estimated value plus or minus $1.96$ times squared root of $(\text{SEE} + S_{y}^2)$, where $S_{y}^2$ is the standard error of the mean prediction. Thus some 95 times out of a hundred, a case with the given values on the independent variables would lie within the computed prediction limits. The prediction interval will be wider (less certain) than the confidence interval, since it deals with an interval estimate of individual cases, not means. Note a number of textbooks do not distinguish between confidence and prediction intervals and confound this difference.

- **F test:** The F test is used to test the significance of $R$, which is the same as testing the significance of $R^2$, which is the same as testing the significance of the regression model as a whole. If prob($F$) $< .05$, then the model is considered significantly better than would be expected by chance and we reject the null hypothesis of no linear relationship of $y$ to the independents. $F$ is a function of $R^2$, the number of independents, and the number of cases. $F$ is computed with $k$ and $(n - p - 1)$ degrees of freedom, where $p$ = number of terms in the equation not counting the constant. That is

$$F = \frac{R^2 / p}{(1 - R^2)/(n - p - 1)}.$$  

- **Multicolinearity** is the intercorrelation of the independent variables. $R^2$'s near 1 violate the assumption of no perfect colinearity, while high $R^2$'s increase the standard error of the beta coefficients and make assessment of the unique role of each independent difficult or impossible. While simple correlations tell something about multicolinearity, the preferred method of assessing multicolinearity is to regress each independent on all the other independent variables in the equation. Inspection of the correlation matrix reveals only bivariate multicolinearity. For bivariate correlations that are greater than .90, we say there are strong bivariate multicolinearity between the corresponding independent variables. To assess multivariate multicolinearity, one uses tolerance or variance inflation factor. The variance inflation factor (VIF) is a measure of how much the variance of an estimated regression coefficient increases if your predictors are correlated (multicolinear). The length of the confidence intervals for your parameter estimates will be increased by the square root of the respective VIFs as compared to the case of uncorrelated predictors. If $X_1, X_2, ..., X_p$ are the $p$ predictors, the VIF for predictor $j$ is $1/(1 - R^2_j)$, where $R^2_j$ is the $R^2$ from regressing $X_j$ on the remaining $p - 1$ predictors. If the correlations of $X_j$ with the other predictors are zero, the VIF will be 1. The VIF increases as $X_j$ becomes more highly correlated with the remaining predictors. Montgomery and Peck [17] suggest that if VIF is $> 5-10$, then the regression coefficients are poorly estimated. Options to break up the multicolinearity include collecting additional data or using different predictors. For additional information, see [16] and [17].