Predictive Quality Control

C.V. Radhakrishnan[a,1,**], K. Bazargan[a,b,2]

[a] *River Valley Technologies, SJP Building, Cotton Hills, Trivandrum, Kerala, India 695014*
[b] *River Valleysfgs Technologies, 9, Browns Court, Kennford, Exeter, United Kingdom*

Abstract

Abstract, should normally be not longer than 200 words. All work and no play makes jack a dull boy.

Keywords: quadruple exiton, polariton, WGM

*Corresponding author
**Principal corresponding author
Email address: `cvr@river-valley.com` (C.V. Radhakrishnan)
URL: `http://www.elsevier.com` (K. Bazargan)
[1] This is the specimen author footnote.
[2] Another author footnote, but a little more longer.

PACS: 71.35.-y, 71.35.Lk, 71.36.+c
2008 MSC: 23-557

1. Introduction

1.1. How is quality measured in present time?

Present day quality measurements are done using various metrological tools which quantify various shapes and/or form features of the part. These features are given their individual limits by the designer which help a metrologist in determining whether a part that has just been manufactured is fit to be deemed a good part or not. However, most of these tools are completely external equipments which need to be explicitly used over the part(s) to see the results. This usually calls for extra man-power and reduced throughput. So what manufacturers end up doing is that they keep an inspection frequency for non critical parts where quality checks on these parts are made at regular intervals until one sees an anomaly. How this anomaly is responded to again depends on what feature we are measuring.

The main problem with such a way of measurement is that :-

1. We have to make a compromise between the throughput and inspection interval.
2. As the quality measurement is isolated from part manufacturing, you can't detect a bad part unless it has already been manufactured.

1.2. How do we propose to measure it?

The main gist of the ideology we have applied is that, when a part is being made, the information of each cut made by the machine is reflected in some real-time aspect of the machine. It could be noise, power, vibration, loads, etc. So what we do is, we record these variables (in our case, power) real-time using appropriate tools and find a correlation between these readings and the quality of the part manufactured. This correlation can be tightened depending on the kind of accuracy one is looking for. It also has an advantage over the conventional approach in certain key aspects such as:-

1. The computational time is significantly less once a correlation model has been made. Also, buying computational time is cheaper than wasting producing time.
2. The feedback we get is near to real time and we could infact get early warnings as the part is being made.
3. The kind of investment needed for this is much lesser compared to buying precise instruments which, other than being pretty expensive, usually have only a single functionality.

2. Methodology

2.1. Getting Quality data

As we need to correlate between power readings during a producing cycle and quality, we are essentially trying to create a model or create a procedure to generate a model which would take the power readings as input and give the quality as output. Thus, we needed some input quality data for a training set.

For this, we monitored the Ra values of a part, which underwent turning operation using a new tool insert, at regular intervals (once in 25 parts for first 500 cycles and once in every 10 parts from there on) for 1030 parts. These measurements were made at two different faces and three different locations in each face. We observed a downward trend in the Ra values for the first 300 parts after which they were more or less constant without any particular trend (save for an anomalous jump at partt 500) till part 1000 after which they again rose up and went outside the specification limit (see Fig 1).

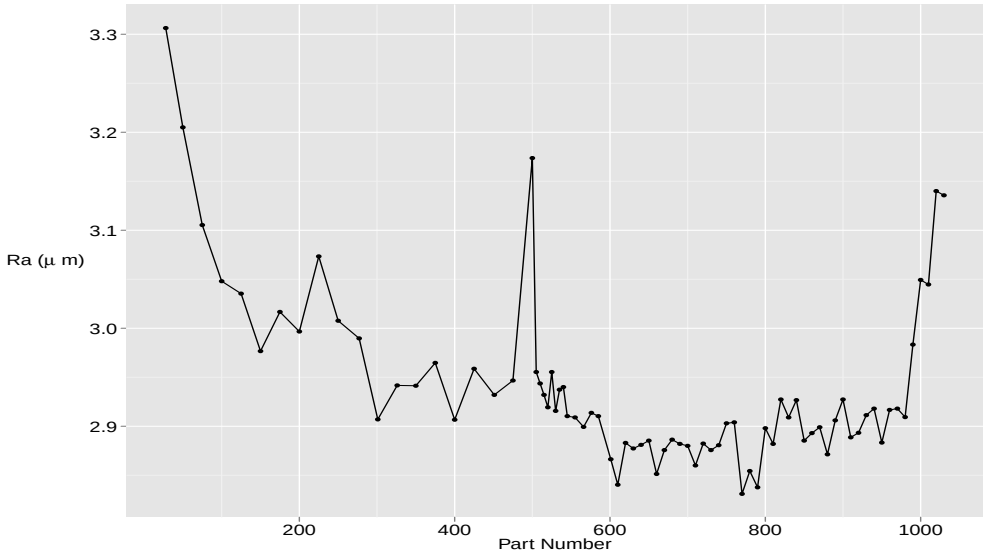

Figure 1: Quality trend

So we decided to use the trend evident in the first 300 parts for our model creation and verification. We created a 2 degree polynomial fit for the first three hundred parts using 12 readings as our control points. The equation turns out to be

$$y \approx 3.35664 - 0.00324170.x + 0.000006684611.x^2 \tag{1}$$

Where, y is the Ra value and x is the part number. We see a maximum error of $0.0833\mu m$ in the fit. For figure, see Fig.2 for the curve and Fig.3 for the residuals' curve.

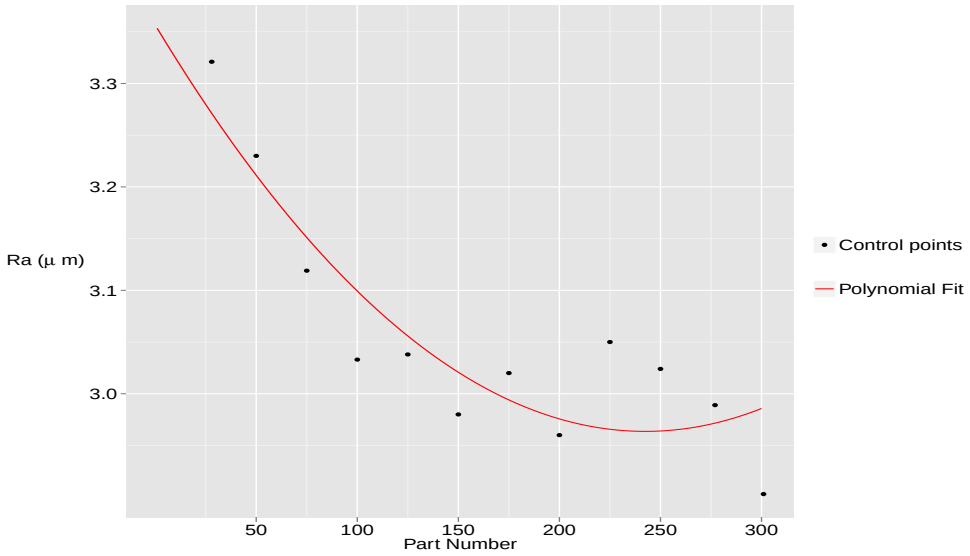

Figure 2: Polynomial fit of Ra values

Now this sample of 300 points was used as our quality data sample which would be used for creating the model and verifying the same.

2.2. Detect Cycles

For input of the model we decided to use power readings. We decided to go with power readings as one would expect a direct correlation between energy consumed and material removed. Also, it depends very little on parts of the machine which are not responsible for cutting (eg. housing) and thus we can make our system more flexible for varioua machines. Also, power readings can be easily obtained using conventional power meters.

As we plan to make an autonomous correlation model, we need to be able to correctly identify the start and end times of a production cycle. In relatively newer machines, this data can be obtained via the controller. However, the number of machines which are not capable of that are quite large. So we need to come up with a method to let the machine identify a cycle and record its start and end time. We use a simple pattern matching approach here. We are going to analyze the power readings we get as a stream of data and in it look for recognisable patterns which would indicate whether and where a cycle occurred. So to start, we need a known producing cycle which we can use as a

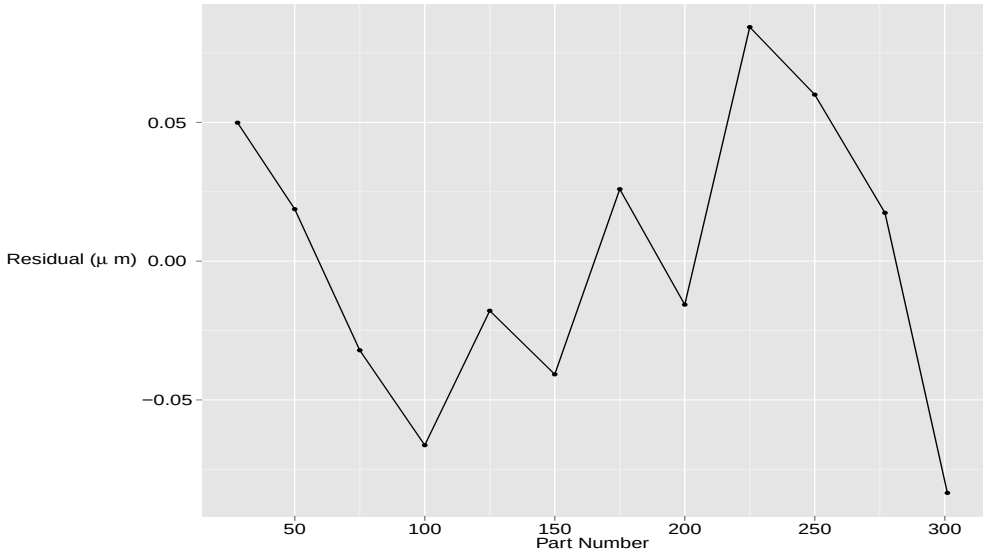

Figure 3: Residuals of polynomial fit for Ra values

template for comparisons. Once we decide this template we use an algorithm called **Dynamic Time Warping** (DTW).

2.2.1. Dynamic Time Warping

Dynamic time warping is a method to compute the similarity between two sequences. To compare these two sequences, one can *warp* (stretch/compress) the sequences non-linearly. Originally used as a tool to detect speech patterns in *Autonomous speech recognition,* DTW is now used in many applications which need to compare one sequence to another. One can limit the amount of warping and thus ensure that realistic results are obtained.

As we can see in Fig 4, even if we have a time shift and small variations in data, the DTW algorithm maps significant features appropriately. This robustness is a result of the warping along time direction allowed. One can also limit the amount of warping permissible so that we can ensure that only physically possible cases are deemed similar. The distance between two sequences being compared is found has sum of distances (euclidean, manhattan, etc) between these mapped points.

2.3. Extracting input from power consumption pattern

Now we have the start and end times for 300 producing cycles and we can use this data to get power consumption patterns for each individual part and use

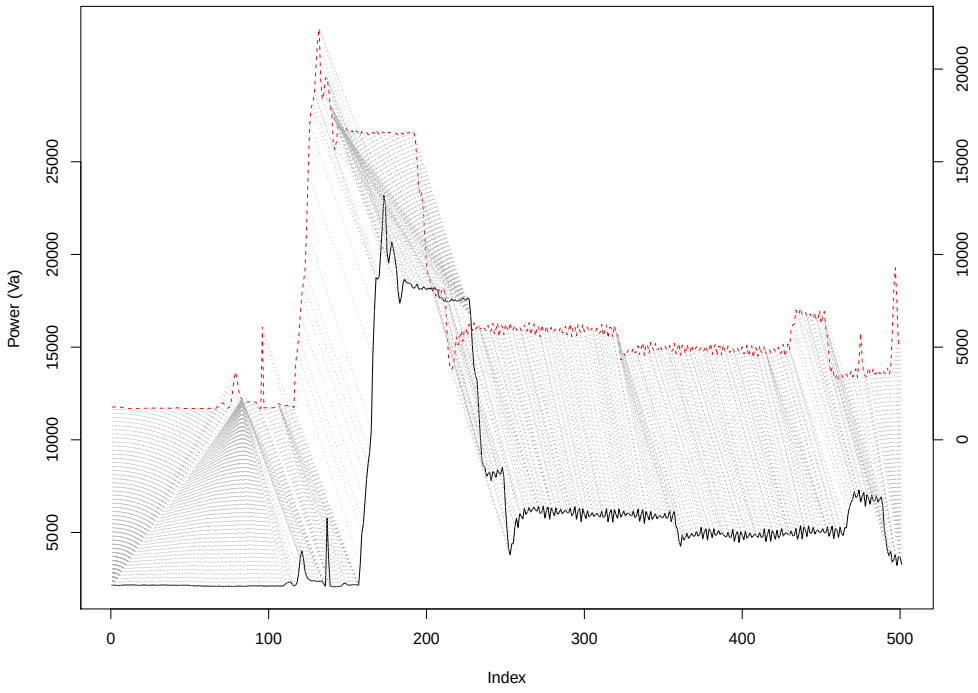

Figure 4: Mapping of two time streams using DTW

it as necessary. However, we can't use these patterns directly as inputs for any machine llearning algorithm as it has many inherent problems. The main reason being that small time offsets in the time series data could throw off the result by a large amount. So we basically needed to process our power pattern to extract more stable information from it. However in the process of making it stable, we also have to make sure that we don't lose the individuality of the pattern as we depend on this individual character for our quality reading. We resolved this using two methods:-

1. Scalar extractions from pattern.
2. Cycle splitting.

2.3.1. Scalar extractions from splitting

Here we extract from our cycle a set of scalars which are stable and do not get affected due to small changes like minor time scaling, peaks,etc. These scalars could be mean, standard deviation, DTW distance, median, mode, amplitude, etc. However, as pointed out earlier, using these scalars alone would render many cycles to be same as we are wasting a lot of information. So we use, not one, but a combination of these scalars as now we get more options of variability thus giving us our desired result of stable and unique inputs.

2.3.2. Cycle splitting

This is a much more important process and increases the effectiveness of scalar extraction. What we essentially do cut the cycle into a number of parts using different methods (which will be discussed later) and apply the scalar extraction for each segment. So now we increase our number of inputs and also increase the individuality of each input in a stable way.

Much more importantly, when we later analyze our results, we can actually pinpoint which segments of the cycle are primarily responsible for the quality parameter being measured (in our case, Ra). This kind of information can prove to be really useful as now we can know what part of the cycle is primarily responsible for which quality parameter. This can in turn help us detecting tool damages when we observe a deviation in certain quality parameter. Splitting the cycle gives us one more important advantage. Now we can calculate the quality metric of a cycle by including one split at a time, into our input set, sequentially. This can be really helpful in predicting parameter qualities before the machining is over and thus can help us in taking corrective action.

We primarily employed four different methods to split cycles. They were:-

1. **Uniform splitting**: As the name suggests, we split the cycle into equally sized sections. The number of parts would depend on how much accuracy we want and how much computation cost we can afford (Fig 5).
2. **Forward feedback splitting**: This is a process in which we make a cut by splitting a section into two equal halves and decide whether we keep that cut by checking if the amount of improvement we get is significant

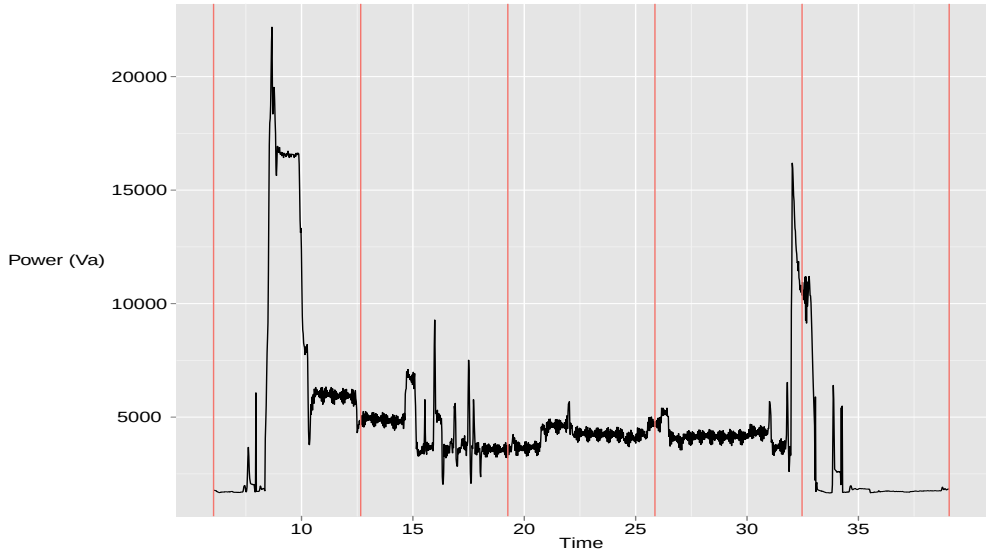

Figure 5: Uniform splitting (5 sections)

enough. It is slightly biased to create more cuts in the beginning of the cycle as we move on to the former section after a cut has been made and move to the next section only if no more cuts can be made in the former one (Fig 6).

3. **Reverse feedback splitting**: This is almost like the Forward Feedback Splitting but instead of biasing cuts to the start, this one biases cuts to the end (7).
4. **Phase dependent splits**: These are cuts which signify a change in process in the machining. Theoretically, these should be the most optimum cuts as they point towards specific process. However, finding them out autonomously can be tricky if wanted for all situations thus these cuts have to be entered manually (Fig 8).

Now, after splitting the cycle appropriately and using scalar extraction for each split, we have a bunch of variables for a cycle which we will later use to get a wider set of data.

2.4. Model Selection

There are a plethora of machine learning algorithms which we could use to establish a relation between our input and output. We decided to try the very simple models and thus decided to use *Polynomial Regression.* However, as our number of variables were bound to be high, we decided to keep the model complexity to a minimum and thus decided to go forward with *Regularised Polynomial Regression.*

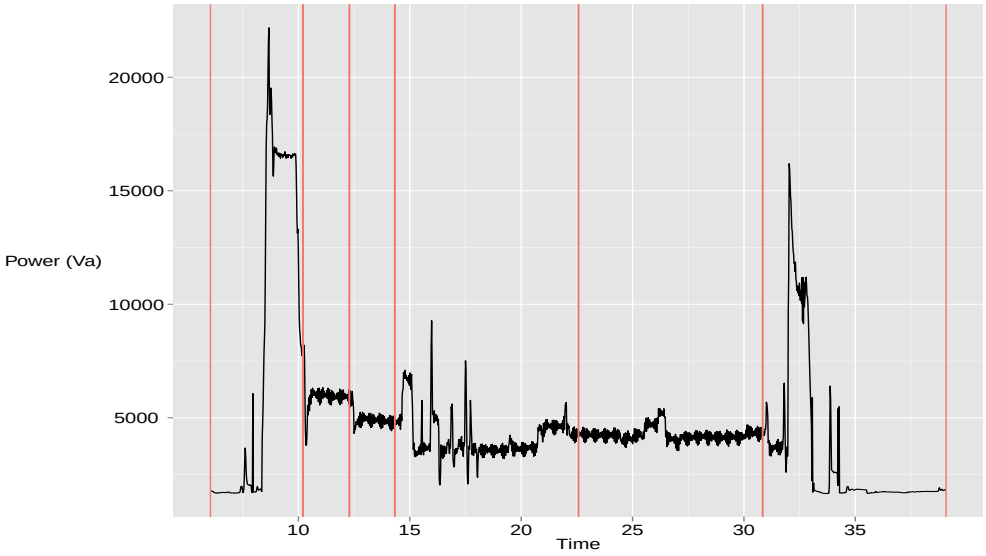

Figure 6: Forward feedback splitting

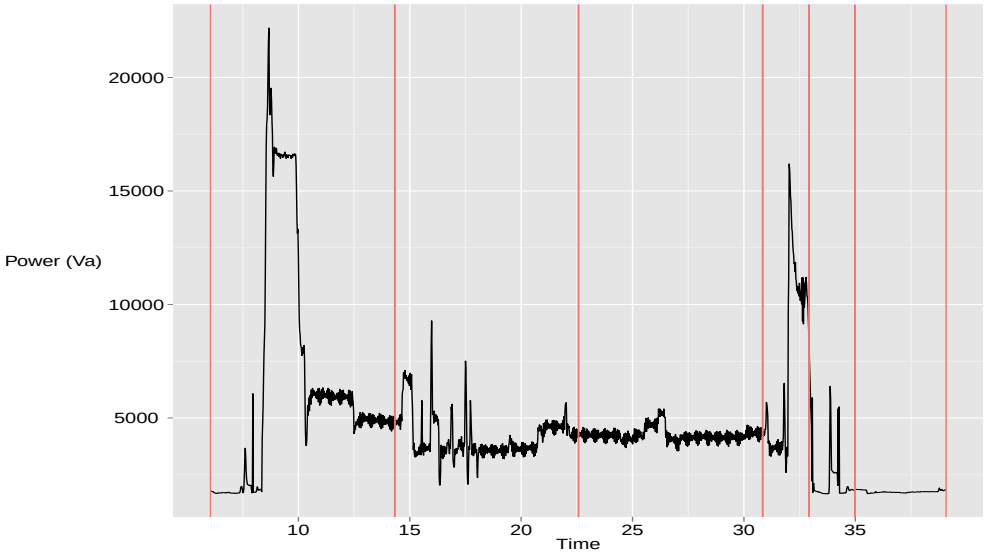

Figure 7: Reverse Feedback splitting

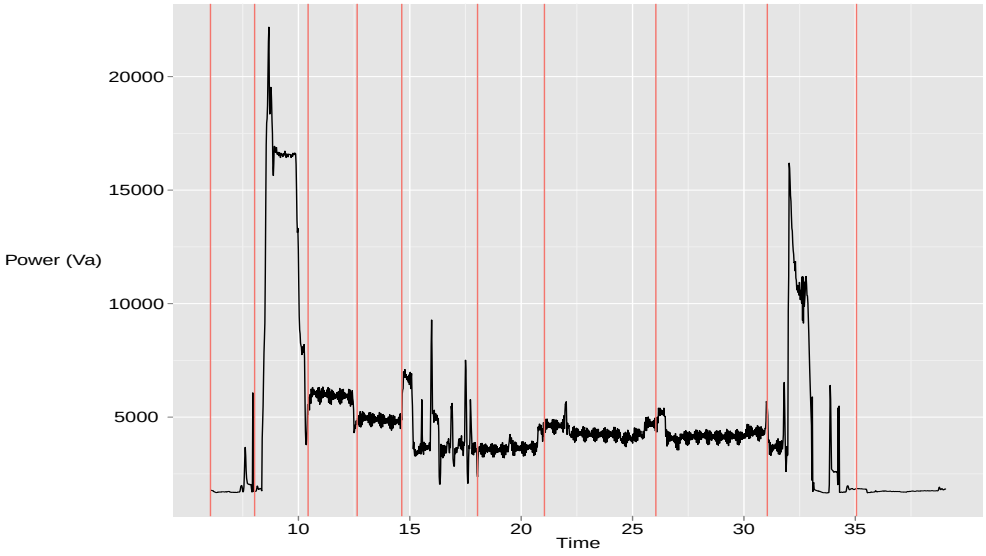

Figure 8: Phase dependent split

2.4.1. Regularised Regression

Say we try to create a linear relation between input variables$x_1, x_2, x_3...., x_n$and output variable y.As it is linear, we can assume our equation to be

$$y = a_{0+}a_1x_1 + a_2x_2 + a_3x_3..... + a_nx_n \tag{2}$$

Where $a_1, a_2, a_3.....a_n$ are the coefficients which we have to find.
Let us define input variable $\mathbf{X}$ and $\mathbf{A}$ as

$$\mathbf{X_i} = (x_1^i, x_2^i, x_3^i, ..., x_n^i) \tag{3}$$

$$\mathbf{A} = (a_1, a_2, a_3, ..., a_n) \tag{4}$$

where x_j^i is the i^{th} instance of the j^{th} variable. Thus, we get

$$y_i = a_{0+}a_1x_1^i + a_2x_2^i + a_3x_3^i..... + a_nx_n^i \tag{5}$$

Now, say we have the p values of $\mathbf{X}_i$and its corresponding output which we denote by $\mathbf{Y}_i$. So to find the best approximation of $\mathbf{A}$, we minimize the difference between $\mathbf{Y}_i$and y_i. Let our final coefficients' set be $\mathbf{A}_{LS}$. Using least square method, we get

$$\mathbf{A_{LS}} = \arg\min_{\mathbf{A}}(\sum_{i=1}^{p}(y_i - \mathbf{Y_i})^2) \tag{6}$$

which can be re-written as

$$\mathbf{A_{LS}} = \arg\min(\sum_{i=1}^{p}(\mathbf{A}^T\mathbf{X_i} - \mathbf{Y_i})^2) \tag{7}$$

Thus as a result of linear regression, we finally get

$$y = \mathbf{A_{LS}^T X} \tag{8}$$

when x_j are polynomial forms of each other, this is called polynomial regression. However, when the number of variables are high, i:e, when n is large, we tend to get highly complex models with high values of a_i which tend to overfit y as a function of $\mathbf{X}$. To remedy this, we include a regularization parameter λ, such that, $\lambda > 0$ and

$$\mathbf{A_{LS}} = \arg\min(\sum_{i=1}^{p}(\mathbf{A}^T\mathbf{X_i} - \mathbf{Y_i})^2 + \lambda\sum_{j=1}^{n}a_j^2) \tag{9}$$

Thus, now we can control, via λ, how complex a system we want. Too high a λ gives us an overfit solution while too low a λ gives us an underfit one.

2.5. Model Creation

To implement *Regularised Polynomial Regression,* we first need to use our present variable set (which we got after extracting scalars from split sections of a cycle) to form polynomial variables. Firstly we divide our 300 cycles into two sets, a *training set* and a *testing set.* each cycle is assigned to one of these sets randomly and the ratio between the number of elements in the two sets can be decided by us. we chose a 1:1 ratio. Now, in the training set, for each variable (scalar from a split section), we have 150 other instances of the same corresponding to 150 cycles. This gives us a vector of that corresponding variable.This way we have a vector corresponding to all the variables. These vectors will be used to create the polynomial variables which will eventually be used as inputs into the *Regularised Linear Regression.* This we do in two ways:-

1. **Orthogonal polynomials-** We create new vectors which are orthonormal to each of the vector of variables we have. The degree of these orthogonal variables were varied to see which would yield reasonable results.
2. **Mixed polynomials-** Here we create new vectors using the old variable vectors (denoted as $x_1, x_2, x_3 ...., x_n$) which can be shown asllll

$$x_{new}(i_1, i_2, i_3, ...i_n) = \prod_{j=1}^{n} x_j^{i_j}; \tag{10}$$

where

$$i_1, i_2, i_3....i_n = 0, 1, 2, 3, 4.....; \tag{11}$$

and

$$\sum_{j=1}^{n} i_j <= p; \tag{12}$$

Where p is the maximum degree which we vary to get optimum results. As expected, we get more number of variables using this method instead of the orthogonal polynomials one. However, it helps us know if there are any meaningful interactions within the variables.

Now that we have our input vectors' set created, we just use *Regularised Linear Regression* to get a list of possible relations between our input quality. In the next section we discuss how we select the right parameters from this given list.

2.6. Parameter comparison and selection

The output of the *regularised polynomial regression* is a list of solutions. Each solution contains a collection of coefficients which we use for a weighted summation of our newly calculated variable vectors. To compare between these various solutions, we use them and predict quality parameter values for our test set and then we compare these predicted values against our polynomial fit values (from 2.1) and find out the *root mean square error(RMSE),* where we assume that the polynomial fit values are the true values. We use this *RMSE* as a metric to compare between the solutions and pick the solution with minimum *RMSE.*

3. Results and Conclusion

In this section we will show the results we obtained for various cases by changing the degree of our polynomial variable, the method of polynomial creation (see 2.5) and mode of splitting (see 2.3.2) .

3.1. Uniform Splitting

Like explained in 2.3.2, we split each detected cycle into equal sized sections before running our algorithms. Here we use a table to denote the performance characteristics of a table which would help us take a decision about how good/bad a method is. A table denotes to a method which splits a cycle into a fixed number of equal-sized segments and uses a particular method for polynomial variable generation. The first column shows the maximum degree used for polynomial variable formation. The second number shows the degree of freedom of the model which in our case is the number of variables with non-zero coefficients. The third and fourth columns show the *root mean square error (RMSE)* of the best solution available for that degree (see 2.6) for the test set and training set respectively.Finally, the last two columns denote the maximum error for of the best solution for both the test set and the training set respectively.

3.1.1. One section

As the name suggests, here the entire cycle is a single section. Thus the cycle without splits would look like Fig 9.

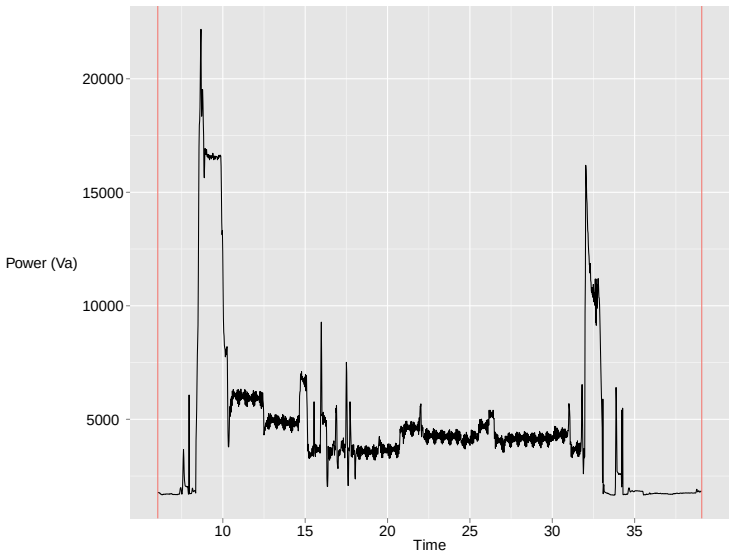

Figure 9: Uniform split single section,

The results have been summarised in Table 1 for orthonormal variables and in Table 2 for mixed polynomial variable.

Degree	Degree of Freedom	RMSE for test set (μm)	RMSE for training set(μm)	Maximum error for test set(μm)	Maximum error for training set(μm)
2	8	0.0990	0.0986	0.3566	0.3472
3	12	0.0986	0.0983	0.3467	0.3430
4	15	0.0997	0.0953	0.3002	0.3392
5	19	0.0957	0.0929	0.2809	0.3437

Table 1: Single section uniform splitting, orthonormal polynomial variables

3.1.2. Two sections

Here, the cycle is split into two sections. Thus the cycle without splits would look like Fig 10.

The results have been summarised in Table 3 for orthonormal variables and in Table 4 for mixed polynomial variable.

Degree	Degree of Freedom	RMSE for test set (μm)	RMSE for training set(μm)	Maximum error for test set(μm)	Maximum error for training set(μm)
2	8	0.0975	0.1007	0.3416	0.3480
3	6	0.0976	0.1005	0.3443	0.3472
4	12	0.0976	0.1006	0.3441	0.3466
5	13	0.0976	0.1004	0.3390	0.3464

Table 2: Single section uniform split, mixed polynomial variables.

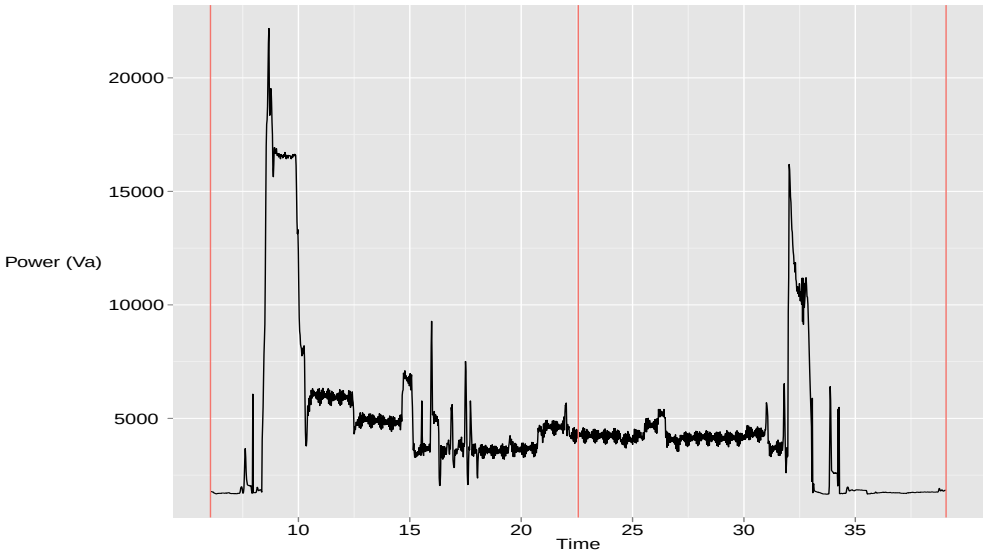

Figure 10: Uiform split- Two sections

Degree	Degree of Freedom	RMSE for test set (μm)	RMSE for training set(μm)	Maximum error for test set(μm)	Maximum error for training set(μm)
2	8	0.0668	0.0586	0.4173	0.1650
3	12	0.0645	0.0564	0.4235	0.1685
4	15	0.0674	0.0564	0.4358	0.1777
5	18	0.0685	0.0551	0.4072	0.1686

Table 3: Uniform splitting-two sections, orthonormal variables

Degree	Degree of Freedom	RMSE for test set (μm)	RMSE for training set(μm)	Maximum error for test set(μm)	Maximum error for training set(μm)
2	9	0.0696	0.0581	0.4444	0.1523
3	9	0.0689	0.0580	0.4405	0.1534
4	13	0.0683	0.0576	0.4382	0.1550
5	13	0.0677	0.0579	0.4287	0.1572

Table 4: Uniform splitting-two sections, mixed polynomial variables

3.1.3. Three sections

Here, the cycle is split into three sections. Thus the cycle without splits would look like Fig 11 and the results have been summarised in Table 5 for orthonormal polynomial variable and Table 6 for mixed polynomial variable.

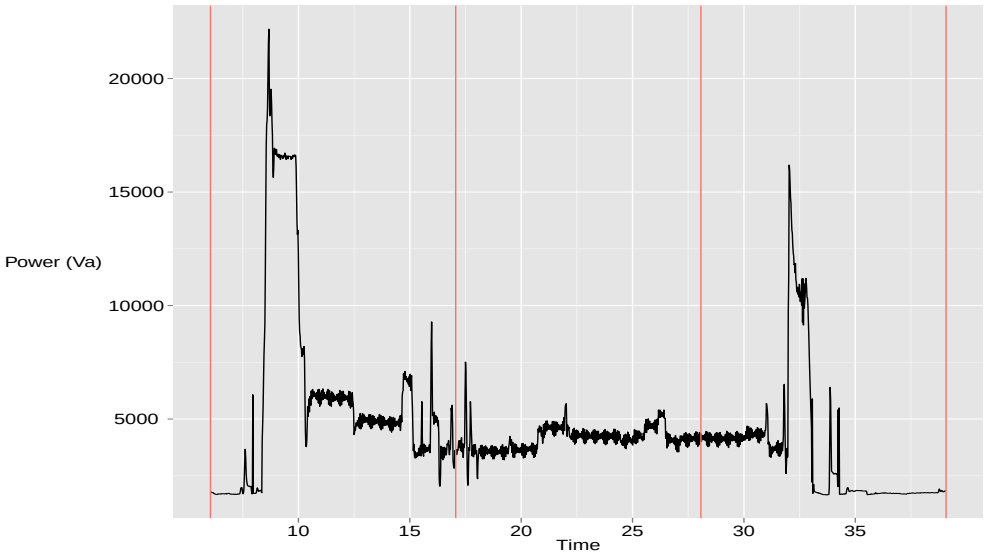

Figure 11: Uniform split, Three sections

3.1.4. Five sections

Here, the cycle is split into five sections. Thus the cycle without splits would look like Fig 12 and the results have been summarised in Table 7 for orthonormal variables and Table 8 for mixed polynomial variable.

Degree	Degree of Freedom	RMSE for test set (μm)	RMSE for training set(μm)	Maximum error for test set(μm)	Maximum error for training set(μm)
2	12	0.0448	0.0413	0.1421	0.1679
3	13	0.0480	0.0405	0.1483	0.1436
4	13	0.0495	0.0416	0.1407	0.1336
5	20	0.0522	0.0354	0.2816	0.1209

Table 5: Uniform splitting-three sections, orthonormal variables

Degree	Degree of Freedom	RMSE for test set (μm)	RMSE for training set(μm)	Maximum error for test set(μm)	Maximum error for training set(μm)
2	14	0.0431	0.0426	0.1399	0.2375
3	9	0.0427	0.0454	0.1423	0.2728
4	7	0.0427	0.0477	0.1417	0.2746
5	9	0.0420	0.0466	0.1303	0.2706

Table 6: Uniform splitting-three sections, orthonormal variables

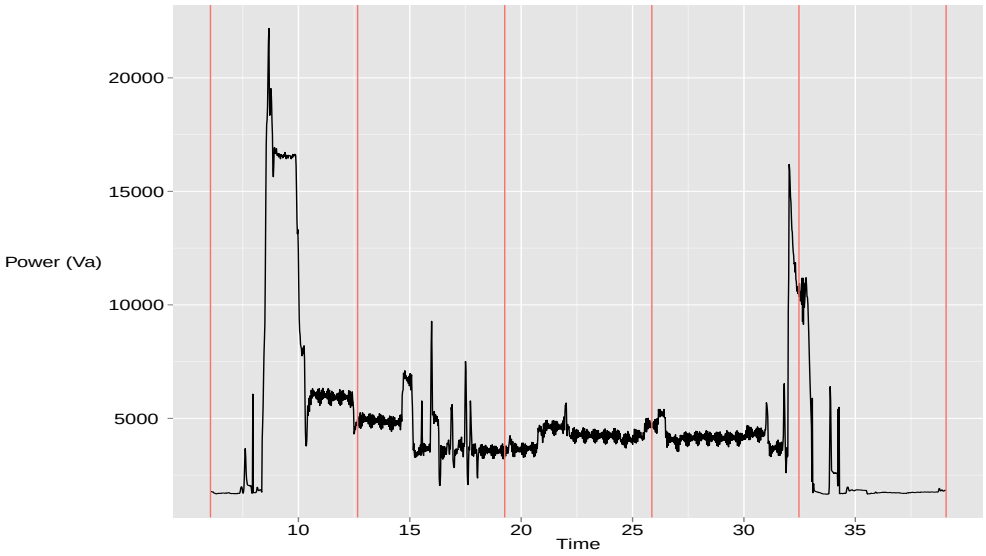

Figure 12: Uniform splitting-Five sections

Degree	Degree of Freedom	RMSE for test set (μm)	RMSE for training set(μm)	Maximum error for test set(μm)	Maximum error for training set(μm)
2	20	0.0561	0.0285	0.4086	0.0835
3	10	0.0647	0.0357	0.6190	0.1278
4	10	0.0647	0.0357	0.6190	0.1278
5	2	0.1156	0.0341	0.8198	0.2196

Table 7: Uniform splits, Five sections-Orthonormal variable polynomial

Degree	Degree of Freedom	RMSE for test set (μm)	RMSE for training set(μm)	Maximum error for test set(μm)	Maximum error for training set(μm)
2	11	0.0850	0.0315	0.8171	0.0984
3	11	0.0881	0.0315	0.8564	0.0986
4	14	0.0931	0.0316	0.9168	0.1009
5	5	0.0977	0.0383	0.8786	0.1595

Table 8: Uniform splits,Five sections-mixed polynomial variable

3.1.5. Six sections

Here, the cycle is split into six sections. Thus the cycle without splits would look like Fig 13 and the results have been summarised in Table 9 for orthonormal variables and Table 10 for mixed polynomial variable.

Degree	Degree of Freedom	RMSE for test set (μm)	RMSE for training set(μm)	Maximum error for test set(μm)	Maximum error for training set(μm)
2	23	0.0367	0.0226	0.2078	0.0797
3	34	0.0409	0.0207	0.2424	0.0728
4	44	0.0397	0.0200	0.2355	0.0750
5	28	0.0502	0.0242	0.3099	0.0731

Table 9: Uniform splits,Six sections-Orthonormal variables

3.1.6. Discussion

Here we are primarily concerned with the RMSE value of each solution. As we can see from the tables:-

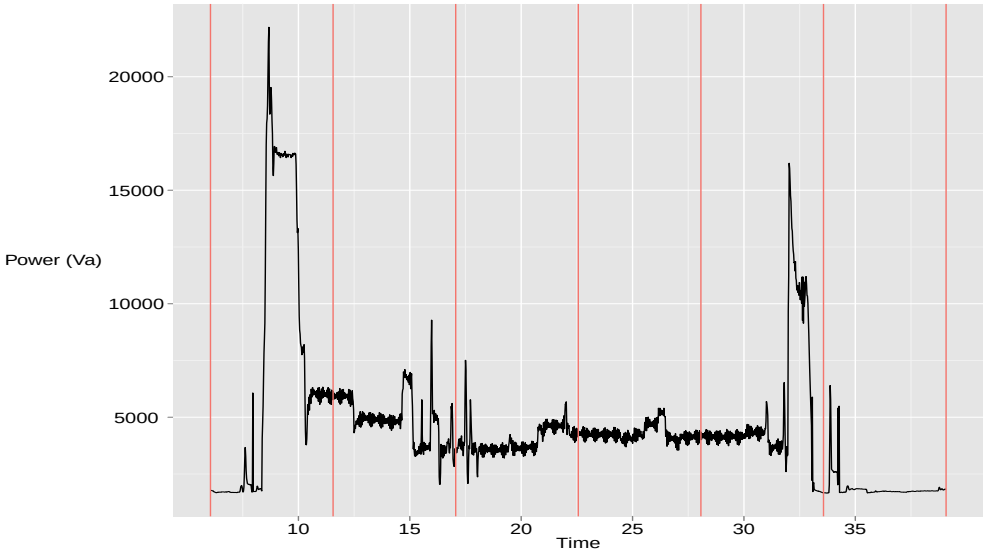

Figure 13: Uniform splitting-Six sections

Degree	Degree of Freedom	RMSE for test set (μm)	RMSE for training set(μm)	Maximum error for test set(μm)	Maximum error for training set(μm)
2	75	0.0382	0.0215	0.1901	0.0855
3	11	0.0395	0.0279	0.1959	0.0914
4	13	0.0383	0.0265	0.1980	0.0902
5	17	0.0373	0.0256	0.1973	0.0960

Table 10: Uniform splits,Six sections - mixed polynomial variable

1. Usually, the orthonormal and mixed polynomials offer outputs usually have more or less similar values of RMSE. However, the time taken to implement the 'mixed polynomial variable' method increases much more rapidly than the 'orthonormal variables method' as we increase the degree and number of splits.
2. For higher degree orthonormal variables' method, we may get high RMSE due to usually just ONE extremely high error value which indicates .
3. For most of the cases, the RMSE remains more or less the same as we change the degree keeping other factors constant.
4. Usually, as we increase the number of splits, the RMSE value keeps on dropping until we reach five sections where we see the RMSE shoot up after which the RMSE goes back down. Although this may seem to be an anomaly in the begining, there is a perfectly reasonable explanation for which becomes evident when we have a look at the way the splits are made. In all the splits other than 5 spliot

AppendixA. Appendix name

Appendix, only when needed.

———————

You can use either BibTEX:

———————————

Or plain bibliography:

[1] Frank Mittelbach and Michel Goossens: *The LaTeX Companion Second Edition*. Addison-Wesley, 2004.

[2] Scott Pakin. The comprehensive LaTeX symbol list, 2005.