Methods in Functional Data Analysis: Modeling Response Curves as Functions of Experimental Factors in Designed Experiments

Jack DeGroot

August 23, 2023

Abstract

This paper introduces different methods of analyzing designed experiments that have comparatively small datasets, with a focused aim of modeling functional response curves directly as functions of the experimental factors. In contrast to conventional experimenters who typically rely on scalar characteristics for modeling, our research advocates for the modeling of the entire functional response curve, a strategy that has proven to be informative and valuable in industry settings. This unique approach distinguishes us from typical functional data analyses as we aim to model the curves as functions of experimental factors. By modeling our response curves as functions of experimental factors we are able to observe how changes in experimental factors influence the shape and behavior of the curves, providing insights into the dynamics of the underlying system. This allows for the optimization of these factors for desired responses, thus enhancing prediction accuracy. In this Master's Project we investigate JMP's Fit Curve method, functional principal components (FPC), and a Bayesian hierarchical two-stage mixed-effects model for this project of modeling functional response curves as functions of experimental factors.

1 Introduction

In this paper I will first provide the descriptions of the 3 data sets that we are using. We will be using data comprised of Viscosity vs. Shear Rate, Fly Ash vs. Temperature, and a Design of Experiment Data set consisting of 10 experimental factors vs. Y Response. I then will provide the three methods that will be implemented into our data to provide us with the most accurate prediction. These three methods are JMP's Fit Curve, functional principal components (FPC), and a Bayesian hierarchical two-stage mixed-effects. After successfully implementing the methodology described I will present the results of each methodology used on each of the three data sets. This will provide us with the 3 examples for us to compare the accuracy of actual results by the predicted results from our methods. I then will go over some of the challenges I faced when researching my project. Finally, this Research paper concludes with our conclusion and addressing possible future work in this realm of functional data analysis in mixture experiments.

2 Data

2.1 Viscosity Data

Our first data set involves Viscosity over varying shear rates within the chemical industry. This example has real world applicability as we observe that viscosity over varying shear rates is a critical-to-customer attribute because it directly affects the consistency or "flow" of products such as shampoo, body wash, or cleansers. Using a single viscosity value at a fixed shear rate would fail to capture the differences in rheological properties among different chemical formulations. Thus, optimizing the formulation to target a competing product's rheological profile would be beneficially to chemical company.

2.2 Fly-Ash Data

Our second data set is Fly ash data which is a common effluent of coal-fired power plants and is increasingly used as an admixture in concrete formulations. Although it takes 28 days for concrete to reach its full strength, the reaction temperature curves in the first 24 hours are indicative of long term strength. Using a mixture amount experiment with three admixture components and total amount of addition as a process variable, we analyze the 24-hour curing temperature data from each of 15 Runs in the experiment. Each Run consists of 1437 rows of data to resulting in our functional response.

2.3 DoE 10 Factors Data

Our final data set is a definitive screening design data that consists of 10 factors with time ranging from zero to three hundred and thirty five hours. There are 24 Runs in our data with each running containing 15 rows of data resulting in our functional response over time. This is based upon real data but the source is propriety so we cannot disclose what the factors are or what the response is.

3 Background

3.1 Self-Validating Ensemble Model (SVEM)

For both our Fit Curve method and FPC method for this paper we need to utilize Self-Validating Ensemble Model (SVEM) in order to produce our predicted values. Hence we now give a brief background in SVEM so that the reader can have an understand of its concept and how it is utilized within our project. SVEM blends concepts from bootstrapping with those of ensemble modeling to fit validated predictive models to data from designed experiments. SVEM has direct implementation of design of experiments, but it is also a general algorithm that can be used to fit many different types of data and predictive modeling strategies. When implementing the SVEM algorithm we use a different bootstrapping procedure with a modified weighting scheme called fractionally weighted bootstrapping (FWB) (Xu et al., 2020). FWB does not sample with replacement. Instead every observation is included in every re-sample. However the fractional weights assigned to each observation are different in every re-sample. Because FWB does not exclude observations in re-samples it is ideal for data affiliated with experimental design where preserving the full structure of the design in the modeling process is imperative. SVEM uses FWB to assign observations to either a training or a "self-validation" sample. Overall, SVEM uses a bootstrapping procedure to construct, for each bootstrap replicate, training and validation sets for fitting and tuning a model. SVEM uses every observation as a member of both sets through a strategic weighted resampling scheme that resembles the fractionally weighted bootstrap in Xu et al. (2020). Now, the fractional weights in SVEM are random draws from an exponential distribution with mean 1. The weighting scheme first generates a set of N random uniform $(0, 1)$ weights and then the inverse probability transform with the exponential distribution is used to generate the fractional weights. [**lemkus2021**] Below, Equation 1 illustrates the computation of the weights, where $U[0, 1]$ represents a uniform distribution on the interval $(0, 1)$, and F is the cumulative distribution function for an exponential distribution with mean 1.

- Generate $u_i \sim U[0, 1]$ for $i = 1...N$
- Training weights: $w_{T,i} = F^{-1}(u_i) = -\log(u_i)$ for $i = 1...N$
- Self-Validation weights: $w_{V,i} = F^{-1}(1 - u_i) = -\log(1 - u_i)$ for $i = 1...N$

$$\begin{aligned} w_{T,i} &= F^{-1}(u_i) = -\log(u_i), \quad i = 1...N \\ w_{V,i} &= F^{-1}(1 - u_i) = -\log(1 - u_i), \quad i = 1...N \end{aligned} \tag{1}$$

The two sets of weights are assigned in pairs to the training and self-validation partitions. This relationship is such that any given $w_{T,i}$ assigned randomly to a case in the training data will have a corresponding $w_{V,i}$ assigned to the same case in the self-validation data. Because of the way the weights are constructed, large values for any given $w_{T,i}$ will have small values for its corresponding $w_{V,i}$ and vice versa. This relationship has a typical inverse relationship between the training and self-validation weights that supports the self-validation approach. [**lemkus2021**]

4 Methods

4.1 Fit Curve Method

As defined by JMP the Fit Curve platform enables users to fit models that are nonlinear in the parameters. This is especially important to us because in all of our datasets we are working with nonlinear equations. The following are examples of equations for linear and nonlinear functions.

Linear function:

$$y = b_0 + b_1 e^x \tag{2}$$

Nonlinear function:

$$y = b_0 + b_1 e^{b_2 x} \tag{3}$$

The Fit Curve platform provides predefined models, such as polynomial, logistic, probit, Gompertz, exponential, peak, pharmacokinetic, rate, and dissolution models. Specifying a grouping variable lets you estimate separate model parameters for each level of the grouping variable. The fitted models and estimated parameters can be compared across the levels of the grouping variable. There is also a set of options to compare dissolution profiles, with options for both parametric and nonparametric techniques. For our project we are concerned with using a Four Parameter Bi-Exponential model for our Viscosity example, a Peak Gaussian model for our Fly-Ash example, and finally a Weibull Growth Model. We will explain the implementation of these models within the Fit Curve platform later in this paper.
Nonetheless, Fit Curve is a user friendly method in JMP also enables users to build their desired model to create the prediction formula or save a parameter table. For our case, we save the parameter table of our desired model and then utilize SVEM to create our predicted values.

4.2 Functional Principal components Model (FPC)

The FPC technique has been used frequently for analyzing functional data such as magnetic resonance imaging (MRI) data, weather data, and stock exchange trends. Significant research has been undertaken using FPC, and the method helps to analyze continuous functions and enables the extraction of key features that epitomize dominant aspects of the original dataset.
The FPC technique is an approach used to reduce the dimensionality of large datasets while helping to enhance interpretability, yet loses minimum information from the data. The FPC can be thought of as an extended version of the traditional multivariate principal component (PCA) with infinite-dimensional vectors, so basically, the function. To apply the FPC, the data has to be given in the form of a smooth function. Conceptually for reference, we take a look at the voltage data are discrete measurements over time from "Prediction of Lithium-Ion Battery Capacity by Functional Principal Component Analysis of Monitoring Data" written by MD Shoriat Ullah and Kangwon Seo. I felt that Ullah and Seo did a fantastic job explaining conceptually how FPC is utilized in their example. We see that the first attempt to represent v_{ij} as a function $v_i(t)$ using a B-spline basis expansion with an appropriate number of knots. The FPC aims to find weight functions $\phi(t)$ that mostly explain the function-to-function variability. The first functional principal component $\phi_1(t)$ is chosen to maximize the mean square $\sum_{i=1}^{N} \frac{\xi_{i1}^2}{N}$ where:

$$\xi_{i1} = \int \phi_1(t) v_i(t) dt \tag{4}$$

with the constraint of the unit squared norm $\int \phi_1(t)^2 dt = \|\phi_1\|^2 = 1$. The second and subsequent FPCs can also be found by solving the same optimization problem with the additional orthogonality constraints of $\int \phi_k(t)\phi_m(t)dt = 0$, for $k < m$. Let us define the covariance function $G(s,t) = \frac{1}{N-1} \sum_{i=1}^{N} v_i(s) v_i(t)$; then, it can be shown that the above optimization problem is reduced to the following eigen-equation:

$$\int G(s,t)\phi(t)dt = \lambda\phi(s) \tag{5}$$

where $\phi()$ is an eigenfunction and λ is an eigenvalue. This continuous functional eigen-analysis problem can be solved by an approximately equivalent matrix eigen-analysis task. It can also be shown that each curve of $v_i(t)$, for $i = 1, \ldots, N$ is approximated by the expansion in terms of a small number of orthonormal basis functions ϕ's by the following form:

$$v_i(t) \approx \mu(t) + \sum_{k=1}^{K} \xi_{ik}\phi_k(t) \tag{6}$$

where $\mu(t) = \frac{1}{N}\sum_{i=1}^{N} v_i(t)$ is the mean function, $\xi_{ik} = \int \phi_k(t)v_i(t)dt$ is the k-th FPC score, and $\phi_k()$ is the k-th eigenfunction. JMP's utilization of FPC is user friendly and allows us to easily model the Functional Principal Components as functions of the experimental factors. [**ullah2022**]

4.3 Bayesian Hierarchical Two-Stage Mixed-Effects

Our last model was based upon the research paper "Bayesian Modeling and Optimization of Functional Responses Affected by Noise Factors" written by Enrique Del Castillo, Bianca M. Colosimo, and Hussam Alshraideh. The goal of their paper was to present a Bayesian predictive modeling approach for functional response systems that optimizes the shape, or profile, of the functional response. An overall summary of the approach presented in this paper is as follows: Suppose a parametric model describes adequately each of the observed profile functions $y_i = (Y_{i1}, Y_{i2}, \ldots, Y_{iJ})^0$, $i = 1, \ldots, N$. Denote the posterior predictive density for a new profile as $p(y|x_c, \text{data})$, where data is short for all past observed profiles and experimental conditions. If after running the experiment $p(y|x_c, \text{data})$ is available (either numerically or in closed form), it is possible to perform an optimization where the posterior probability of achieving a function falling within a given specification region $A(s)$ is maximized, that is, we would solve:

$$\max p(x_c) = P(y \in A(s)|x_c, \text{data}) \quad = \int_A p(y|x_c, \text{data}), dy$$

subject to: $x_c \in R_c$, where R_c is a feasible region for the controllable factors.

This is an approach first suggested by J.J. Peterson in "A Posterior Predictive Approach to Multiple Response Surface Optimization". Peterson (2004), who proposed it for the multiple-response optimization of linear regression models. The main advantage of this method is that it accounts for the uncertainty in the parameters of the model (via the predictive density, which integrates over the posterior of the model parameters) and considers any correlation present between the responses . It also provides an easy-to-explain probability metric about the "capability" of achieving the specifications A(s).

If noise factors are present in the system and affect the responses (RPD case), a Bayesian approach that provides solutions that are robust with respect to both noise factor variability and regression coefficient uncertainty was proposed by Miro in "A Bayesian Approach for Multiple Response Surface Optimization in the Presence of Noise Variables" (2004) as an extension to Peterson's method. It consists in solving:

$$\max p(x_c)\text{RPD} = \int_{\text{all } x_n} P(y \in A(s)|x, \text{data}) \cdot p(x_n)\, dx_n$$

subject to: $x_c \in R_c$. That is, after obtaining $p(x)$ for fixed $x = (x_c, x_n)$, a second integration is performed over the (assumed-known) density of the noise factors $p(x_n)$ to obtain the probability of conformance for the RPD problem, $p(x_c)$RPD.

Specifically we investigate example three and four which originate from Govaerts, B. and Noel, J. 2005 Paper "Analyzing the Results of a Designed Experiment when the Response Is a Curve: Methodology and Application in Metal Injection Moulding". Using Govaerts and Noels Data, Castillo, Colosimo, and Alshraideh implement their Bayesian Hierarchical Two Stage model by first fitting a linear model based on splines to their data. They use:

$$Y(s) = \beta_0 + \beta_1 \log(s) + \beta_2(\log(s) - 1.4955)^2 + \beta_3(\log(s) - 1.7374) + \beta_4(\log(s) - 1.7374)^2, \quad (7)$$

where $(x)_+ = \max(0, x)$ denotes the positive part (the locations of the knots were obtained by fitting this model to the trimmed average of $Y(s_j)$ at each location s_j using nonlinear least squares). On a second stage, each β_i was modeled as a quadratic polynomial in the two factors. From this, Castillo, Colosimo, and Alshraideh enter stage two of their method and utilize Gibbs Sampling from Markov chain Monte Carlo Method (MCMC). Utilizing methods from Lange, N.; Carlin, B. P.; and Gelfand, A. E. (1992) from their paper "Hierarchical Bayes Models for the Progression of HIV Infection Using Longitudinal CD4 T-Cell Numbers" their methodology gives the full conditionals for the parameters $(\theta, \{\omega_i^*\}, \{\sigma_i^2\}, \varphi^*\omega)$ in the linear mixed model, where the σ_i^2's model different variances among the observed profiles [**delcastillo2012**]. Provided are the full conditionals of the other parameters that Castillo, Colosimo, and Alshraideh used in their example.

$$\begin{aligned}
\theta &\sim N_{pq}(0, B_0), \quad \text{with } B_0 = 10000I \quad \text{(noninformative for } \theta\text{)}. \\
\sigma^2 &\sim \text{IG}(1, 2) \quad \text{(inverse-gamma distribution)}. \\
\varphi^*\omega &\sim \text{Wishart}\left((\Gamma_0 R_0)^{-1}, \Gamma_0\right), \quad \text{where } E(\varphi^*\omega) \approx R_0.
\end{aligned}$$

where the different values of Γ_0, σ^2, and R_0 parameters used in this paper are given in each of the corresponding examples.

For the Gibbs Sampling they let Y be an $N \times J$ matrix containing all the observed functional observations at each of the N experimental runs. The Gibbs sampling scheme is as follows:

1. Sample β from $\beta|Y, \sigma^2, \Sigma_w$:
   $\left(\sum_{i=1}^N X_i^T V_i^{-1} X_i + B_1^T\right)^{-1} \left(\sum_{i=1}^N X_i^T V_i^{-1} y_i + B_1^T 0\right)$,
   $\left(\sum_{i=1}^N X_i^T V_i^{-1} X_i + B_1^T\right)^{-1}$,
   where $V_i = S_i^* \Sigma_w (S_i^*)^T + \sigma^2 I$, $X_i = x(m)_i^T \otimes S$, and S_i^* equals the first p^2 columns of X_i.
2. Sample the random effects w_i^*, $i = 1, \ldots, N$, from
   $\{w_i^*\}|Y, \beta, \sigma^2, \Sigma_w$:
   $\left(S_i^{*T} S_i^{*T}/\sigma^2 + \Sigma_w^{-1}\right)^{-1} S_i^{*T} R(w)_i/\sigma^2$,
   $\left(S_i^{*T} S_i^{*T}/\sigma^2 + \Sigma_w^{-1}\right)^{-1}$,
   where $R(w)_i = y_i - X_i\beta$. Note how the mean is the empirical Bayes estimate of w_i^*.
3. Sample Σ_w^{-1} from $\Sigma_w^{-1}|\{w_i^*\}$:
   $\text{Wishart}\left(\left(\sum_{i=1}^N w_i^* w_i^{*T} + \nu_0 R^T\right)^{-1}, N + \nu_0\right)$.

4. Sample σ^2 from $\sigma^2|Y, \beta, \{w_i^*\}$:
   Gamma $\left(1 + JN/2, \left(\sigma^2/2 + \frac{1}{2}R(\sigma^2)^T R(\sigma^2)\right)^{-1}\right)$,
   where $R(\sigma^2) = Y - X\beta - \text{vec}(S_i^* w_1^*, \ldots, S_i^* w_N^*)$ (a $NJ \times 1$ vector).

The MCMC sampling of the mixed-effects model parameters needs not be conducted within the optimization routine necessary to solve Equation (2); otherwise, this would imply a tremendous computational burden. The reason for this is given by the model written as in Equation (6). Given realizations of the posterior of $\Theta = (\beta, \{w_i^*\}, \Sigma_w, \sigma^2)$, $p(\Theta|\text{data})$, we can simulate draws of the posterior predictive density by composition [**govaerts2005**] as follows:

$$p(y|\text{data}, x) = \int p(y, \Theta|\text{data}, x)d\Theta = \int p(y|\text{data}, \Theta, x)p(\Theta|\text{data})d\Theta. \tag{8}$$

Thus, we conduct a simulation of the Gibbs sampling chain until convergence, approximating in this way $p(\Theta|\text{data})$ and simply sample from it. We then substitute the sampled values into the marginal likelihood $p(y|\text{data}, \Theta, x)$ whenever a $y|\text{data}, x$ vector is needed in the optimization routine. Hence, the MCMC computations are only run once, before performing the optimization. This represents a common random-numbers strategy used in the optimization routine for the posterior parameters, similar to that recommended for the noise factors. The convergence of the MCMC iterations was verified by time plots and computation of the autocorrelation function of all parameters [**delcastillo2012**].

In summary, what we are concerned with is the the predicted responses. These are obtained from the mean of the posterior density of yi at each setting xi. After email communication with Dr. Enrique Castillo, he provided me with the MATLAB code for me to run my project on but also included the examples in his group's research paper to properly see how the code works. My findings of Dr. Enrique Castillo and his group's code with his examples and my own project are found in the Results section of this Research Paper.

5 Implementation

5.1 Fit Curve

We apply our Fit Curve approach by using JMP. In JMP under the analyze tab and within specialized modeling we see Fit Curve. For all three examples we apply the associated inputs which for all three is time. The Y outputs which is Viscosity, TempC, and Y. The supplementary z elements which are the experimental settings: A, B, C, D, and E, Limestone, Gypsum, Fly Ash, and Amount, and then finally X1, X2, X3, X4, X5, X6, X7, X8, X9, and X10. We finally put group everything by Run. Once implemented we have our curve and we apply the best models to our data. We test different models for each of our three data and find the best models.

First for our Viscosity data, we find that a 4 Parameter Bi-exponential Model fits our data best and provides us with the best possible prediction for Y. Below you can see our fitted curve from the 4 Parameter Bi-exponential. Specifically at the bottom of the graph, we can see that for each run each curve fits well for the data. This is statistically shown through our R Square value which is around 0.95.

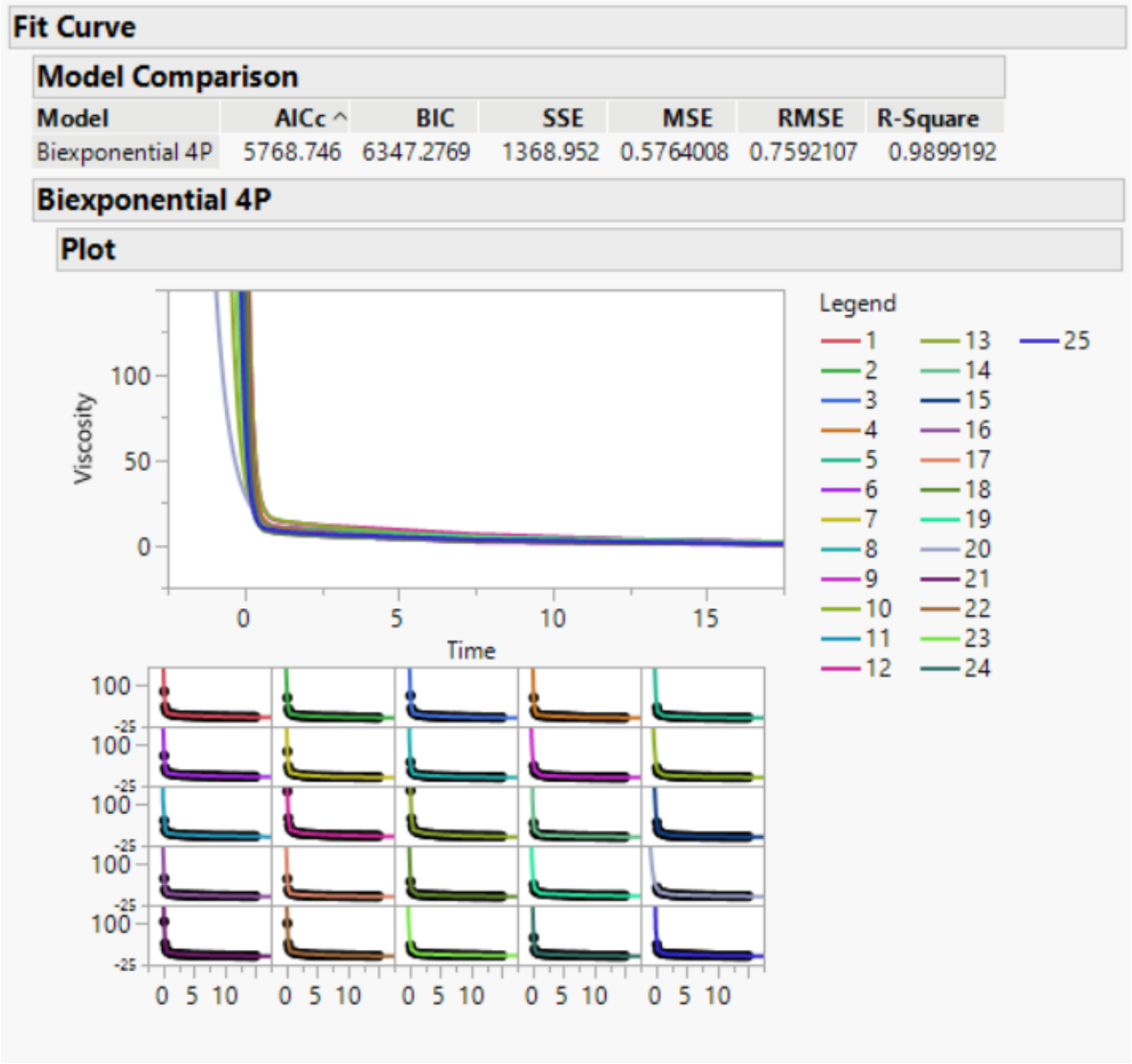
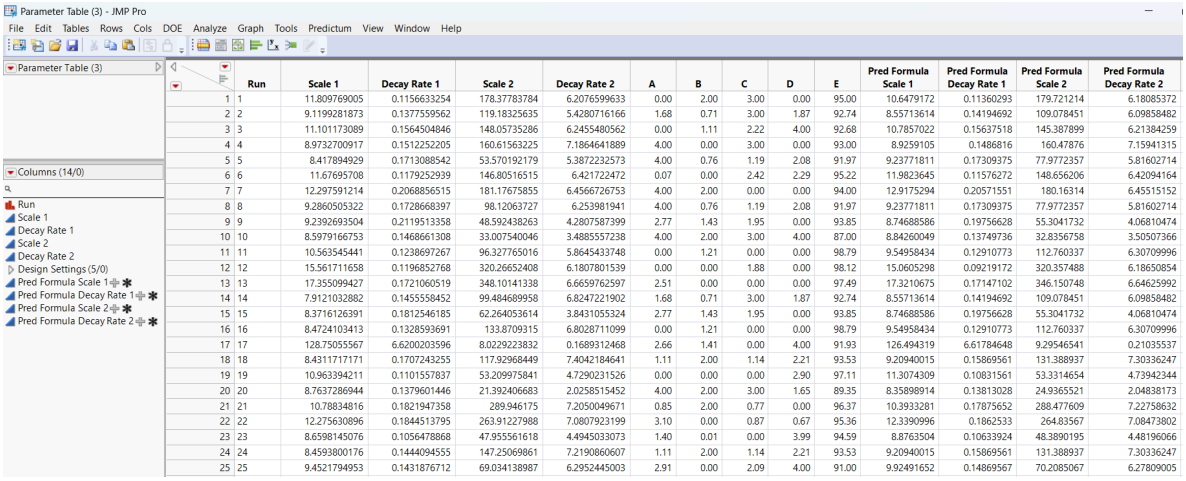


Figure 1: Viscosity 4 Parameter Bi-exponential Fit

We then save a parameter table from our 4 Parameter Bi-exponential Model where we receive estimates for our models formula parameters: Scale 1, Decay Rate 1, Scale 2 and Decay Rate 2. We then utilize the SVEM specifically a Neural Network with 50 bootstrapped Models and 5 TanH first layer Nodes within the Predictum Tab in JMP to create a Prediction Formula for each formula Parameter. Below you can see our parameter table for Viscosity which includes both our model estimates for our Peak Gaussian Parameters and our Fit Curve prediction formula for our 4 Parameter Bi-exponential formula Parameters.



	Run	Scale 1	Decay Rate 1	Scale 2	Decay Rate 2	A	B	C	D	E	Pred Formula Scale 1	Pred Formula Decay Rate 1	Pred Formula Scale 2	Pred Formula Decay Rate 2
1	1	11.809769005	0.1156633254	178.37783784	6.2076599633	0.00	2.00	3.00	0.00	95.00	10.6479172	0.11360293	179.721214	6.18085372
2	2	9.1199281873	0.1377559562	119.18325635	5.4280716166	1.68	0.71	3.00	1.87	92.74	8.55713614	0.14194692	109.078451	6.09858482
3	3	11.101173089	0.1564504846	148.05735286	6.2455480562	0.00	1.11	2.22	4.00	92.68	10.7857022	0.15637518	145.387899	6.21384259
4	4	8.9732700917	0.1512252205	160.61563225	7.1864641889	4.00	0.00	3.00	0.00	93.00	8.9259105	0.1486816	160.47876	7.15941315
5	5	8.417894929	0.1713088542	53.570192179	5.3872232573	4.00	0.76	1.19	2.08	91.97	9.23771811	0.17309375	77.9772357	5.81602714
6	6	11.67695708	0.1179252939	146.80516515	6.421722472	0.07	0.00	2.42	2.29	95.22	11.9823645	0.11576272	148.656206	6.42094164
7	7	12.297591214	0.2068856515	181.17675855	6.4566726753	4.00	2.00	0.00	0.00	94.00	12.9175294	0.20571551	180.16314	6.45515152
8	8	9.2860505322	0.1728668397	98.12063727	6.253981941	4.00	0.76	1.19	2.08	91.97	9.23771811	0.17309375	77.9772357	5.81602714
9	9	9.2392693504	0.2119513358	48.592438263	4.2807587399	2.77	1.43	1.95	0.00	93.85	8.74688586	0.19756628	55.3041732	4.06810474
10	10	8.5979166753	0.1468661308	33.007540046	3.4885557238	4.00	2.00	3.00	4.00	87.00	8.84260049	0.13749736	32.8356758	3.50507366
11	11	10.563545441	0.1238697267	96.327765016	5.8645433748	0.00	1.21	0.00	0.00	98.79	9.54958434	0.12910773	112.760337	6.30709996
12	12	15.561711658	0.1196852768	320.26652408	6.1807801539	0.00	0.00	1.88	0.00	98.12	15.0605298	0.09219172	320.357488	6.18650854
13	13	17.355099427	0.1721060519	348.10141338	6.6659762597	2.51	0.00	0.00	0.00	97.49	17.3210675	0.17147102	346.150748	6.64625992
14	14	7.9121032882	0.1455558452	99.484689958	6.8247221902	1.68	0.71	3.00	1.87	92.74	8.55713614	0.14194692	109.078451	6.09858482
15	15	8.3716126391	0.1812546185	62.264053614	3.8431055324	2.77	1.43	1.95	0.00	93.85	8.74688586	0.19756628	55.3041732	4.06810474
16	16	8.4724103413	0.1328593691	133.8709315	6.8028711099	0.00	1.21	0.00	0.00	98.79	9.54958434	0.12910773	112.760337	6.30709996
17	17	128.75055567	6.6200203596	8.0229223832	0.1689312468	2.66	1.41	0.00	4.00	91.93	126.494319	6.61784648	9.29546541	0.21035537
18	18	8.4311717171	0.1707243255	117.92968449	7.4042184641	1.11	2.00	1.14	2.21	93.53	9.20940015	0.15869561	131.388937	7.30336247
19	19	10.963394211	0.1101557837	53.209975841	4.7290231526	0.00	0.00	0.00	2.90	97.11	11.3074309	0.10831561	53.3314654	4.73942344
20	20	8.7637286944	0.1379601446	21.392406683	2.0258515452	4.00	2.00	3.00	1.65	89.35	8.35898914	0.13813028	24.9365521	2.04838173
21	21	10.78834816	0.1821947358	289.946175	7.2050049671	0.85	2.00	0.77	0.00	96.37	10.3933281	0.17875652	288.477609	7.22758632
22	22	12.275630896	0.1844513795	263.91227988	7.0807923199	3.10	0.00	0.87	0.67	95.36	12.3390996	0.1862533	264.83567	7.08473802
23	23	8.6598145076	0.1056478868	47.955561618	4.4945033073	1.40	0.01	0.00	3.99	94.59	8.8763504	0.10633924	48.3890195	4.48196066
24	24	8.4593800176	0.1444094555	147.25069861	7.2190860607	1.11	2.00	1.14	2.21	93.53	9.20940015	0.15869561	131.388937	7.30336247
25	25	9.4521794953	0.1431876712	69.034138987	6.2952445003	2.91	0.00	2.09	4.00	91.00	9.92491652	0.14869567	70.2085067	6.27809005

Figure 2: Viscosity Parameter Table

We then copy and paste our Scale 1, Decay Rate 1, Scale 2 and Decay Rate 2 prediction formulas to our original data set and implement them into a new column that has the 4 Parameter Bi-exponential Model Formula which is defined as:

$$\begin{aligned}&\text{Pred Formula Scale 1} \times \exp(-\text{Pred Formula Decay Rate 1} \times \text{Time})\\ &+ \text{Pred Formula Scale 2} \times \exp(-\text{Pred Formula Decay Rate 2} \times \text{Time})\end{aligned}$$

Once JMP computes the column we finally have our Fit Curve Viscosity Prediction. We then use graph builder to compare our Fit Curve approach Viscosity prediction with the actual Viscosity and the other two Viscosity prediction Methods.

For Fly Ash we find that a Peak Gaussian Model fits our data best and provides us with the best possible prediction for Y. Below you can see our fitted curve from the Peak Gaussian Model. Specifically at the bottom of the graph, we can see that for each run each curve fits well for the data. This is statistically shown through our R Square value which is around 0.95.

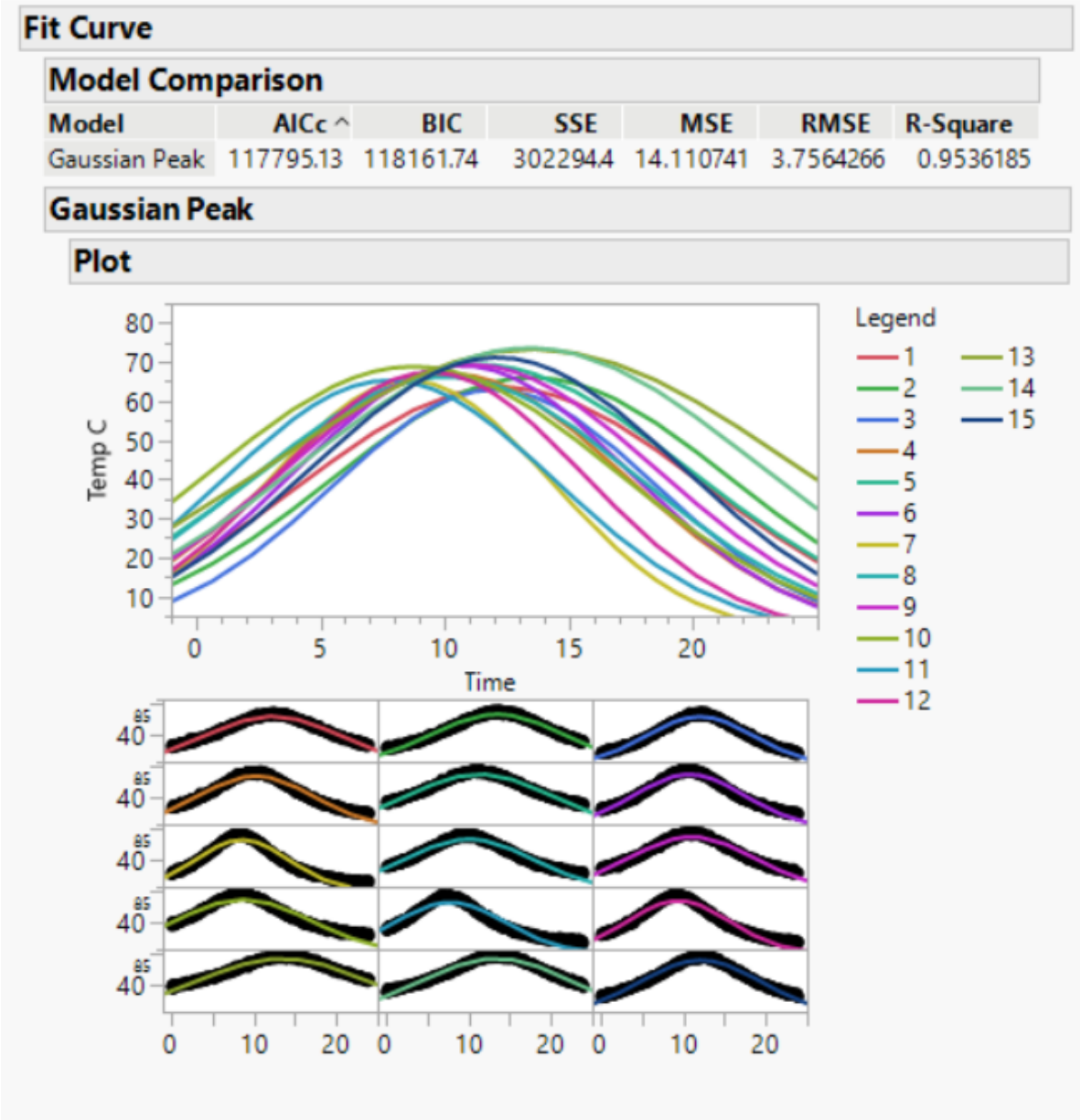
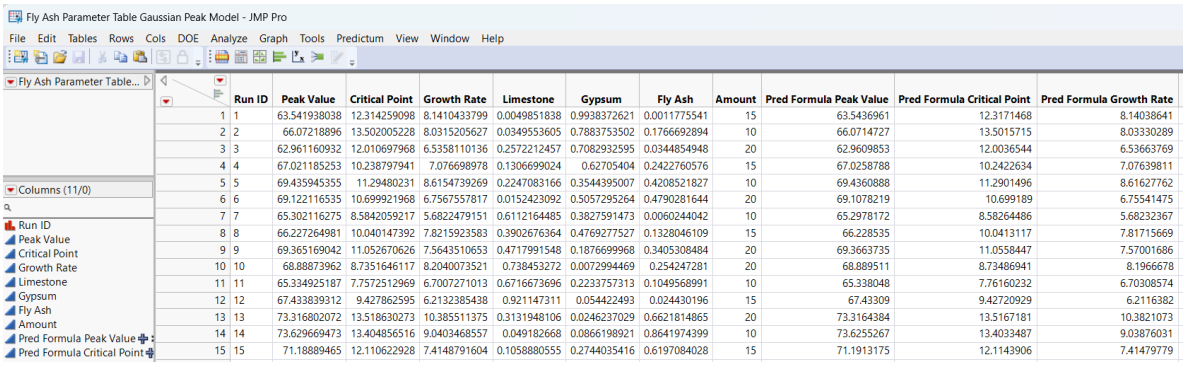


Figure 3: Fly Ash Peak Gaussian Fit

We then save a parameter table from our Peak Gaussian Model where we receive estimates for our models formula parameters: Peak Value, Critical Value, and Growth Rate. We then utilize the SVEM specifically a Neural Network with 50 bootstrapped Models and 5 TanH first layer Nodes within the Predictum Tab in JMP to create a Prediction Formula for each formula Parameter. Below you can see our parameter table for DoE 10 which includes both our model estimates for our Peak Gaussian Parameters and our Fit Curve prediction formula for our Peak Gaussian formula Parameters.



	Run ID	Peak Value	Critical Point	Growth Rate	Limestone	Gypsum	Fly Ash	Amount	Pred Formula Peak Value	Pred Formula Critical Point	Pred Formula Growth Rate
1	1	63.541938038	12.314259098	8.1410433799	0.0049851838	0.9938372621	0.0011775541	15	63.5436961	12.3171468	8.14038641
2	2	66.07218896	13.502005228	8.0315205627	0.0349553605	0.7883753502	0.1766692894	10	66.0714727	13.5015715	8.03330289
3	3	62.961160932	12.010697968	6.5358110136	0.2572212457	0.7082932595	0.0344854948	20	62.9609853	12.0036544	6.53663769
4	4	67.021185253	10.238797941	7.076698978	0.1306699024	0.62705404	0.2422760576	15	67.0258788	10.2422634	7.07639811
5	5	69.435945355	11.29480231	8.6154739269	0.2247083166	0.3544395007	0.4208521827	10	69.4360888	11.2901496	8.61627762
6	6	69.122116535	10.699921968	6.7567557817	0.0152423092	0.5057295264	0.4790281644	20	69.1078219	10.699189	6.75541475
7	7	65.302116275	8.5842059217	5.6822479151	0.6112164485	0.3827591473	0.0060244042	10	65.2978172	8.58264486	5.68232367
8	8	66.227264981	10.040147392	7.8215923583	0.3902676364	0.4769277527	0.1328046109	15	66.228535	10.0413117	7.81715669
9	9	69.365169042	11.052670626	7.5643510653	0.4717991548	0.1876699968	0.3405308484	20	69.3663735	11.0558447	7.57001686
10	10	68.88873962	8.7351646117	8.2040073521	0.738453272	0.0072994469	0.254247281	20	68.889511	8.73486941	8.1966678
11	11	65.334925187	7.7572512969	6.7007271013	0.6716673696	0.2233757313	0.1049568991	10	65.338048	7.76160232	6.70308574
12	12	67.433839312	9.427862595	6.2132385438	0.921147311	0.054422493	0.024430196	15	67.43309	9.42720929	6.2116382
13	13	73.316802072	13.518630273	10.385511375	0.3131948106	0.0246237029	0.6621814865	20	73.3164384	13.5167181	10.3821073
14	14	73.629669473	13.404856516	9.0403468557	0.049182668	0.0866198921	0.8641974399	10	73.6255267	13.4033487	9.03876031
15	15	71.18889465	12.110622928	7.4148791604	0.1058880555	0.2744035416	0.6197084028	15	71.1913175	12.1143906	7.41479779

Figure 4: Fly Ash Peak Gaussian Parameter Table

We then copy and paste our Peak Value, Critical Value, and Growth Rate prediction formulas to our orignal data set and implement them into a new column that has the Peak Gaussian Formula which is defined as:

$$\text{Pred Peak Value} \times \exp\left(-\frac{1}{2}\left(\frac{(\text{Time} - \text{Pred Critical Value})}{\text{Pred Growth Rate}}\right)^2\right)$$

Once JMP computes the column we finally have our Fit Curve TempC Prediction. We then use graph builder to compare our Fit Curve approach TempC prediction with the actual TempC and the other two TempC prediction Methods.

For DoE 10 factor data we find that a Weibull Growth Model fits our data best and provides us with the best possible prediction for Y. Below you can see our fitted curve from the Weibull Growth model. Specifically at the bottom of the graph, we can see that for each run each curve fits well for the data. This is statistically shown through our R Square value which is around 0.97.

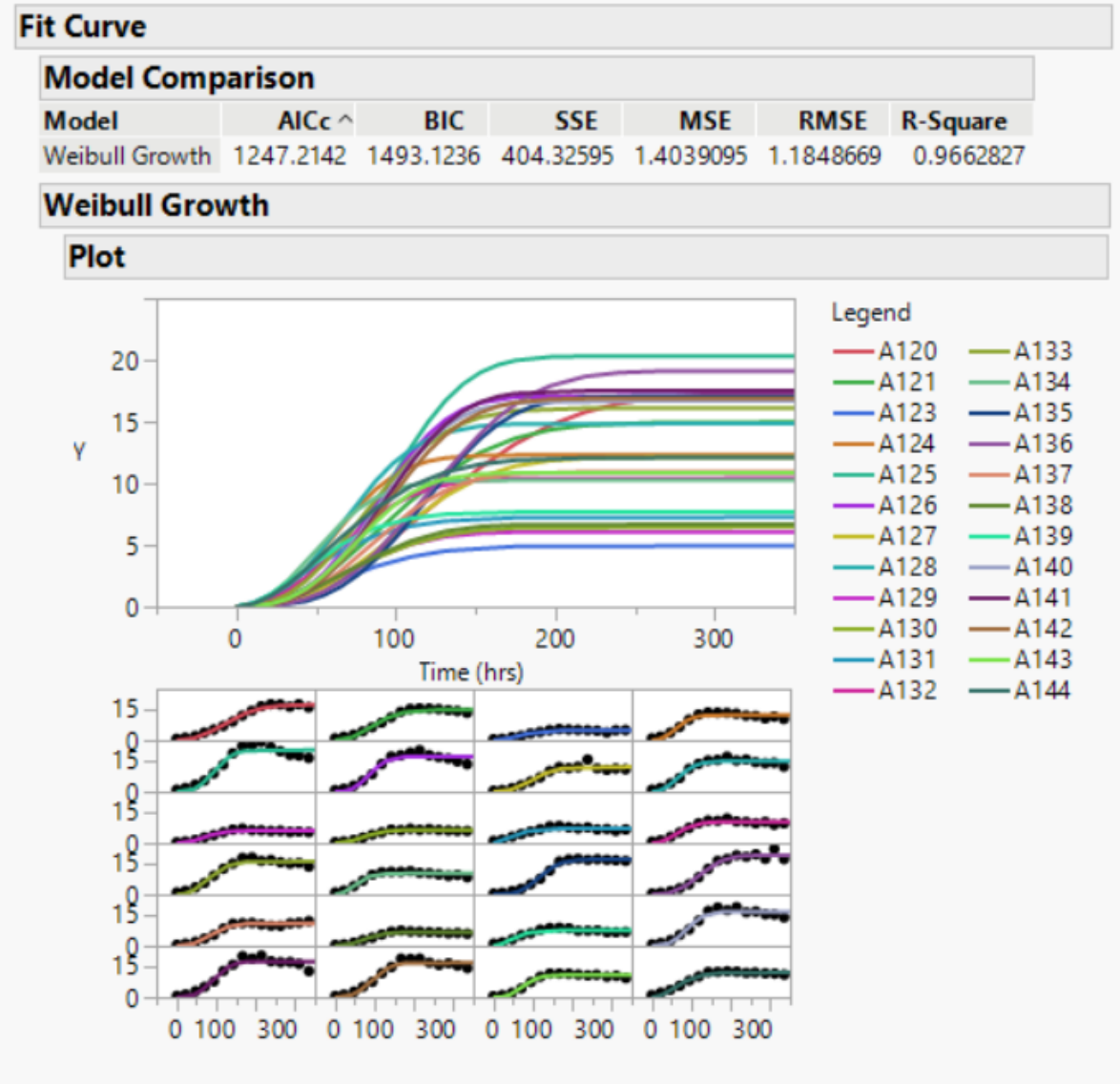
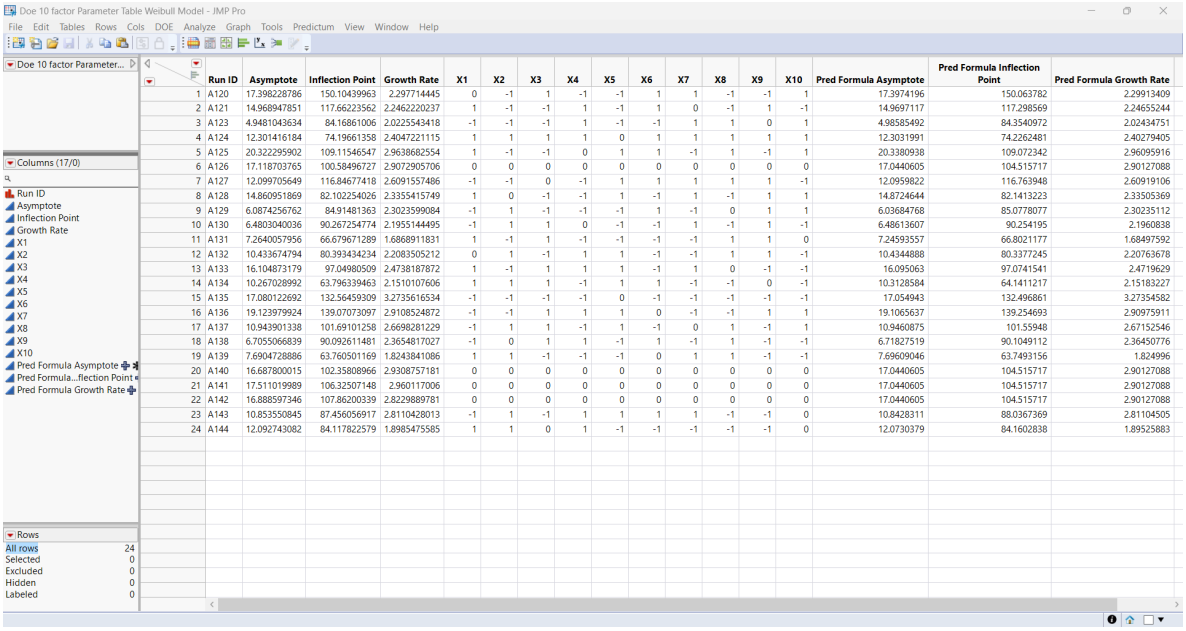


Figure 5: DoE 10 Weibull Growth Fit

We then save a parameter table from our Weibull Model where we receive estimates for our models formula parameters: Asymptote, Inflection Point, and Growth Rate. We then utilize the SVEM specifically a Neural Network 50 bootstrapped Models and 5 TanH first layer Nodes within the Predictum Tab in JMP to create a Prediction Formula for each formula Parameter. Below you can see our parameter table for DoE 10 which includes both our model estimates for our Weibull Growth Parameters and our Fit Curve prediction formula for our Weibull Growth formula Parameters.



	Run ID	Asymptote	Inflection Point	Growth Rate	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	Pred Formula Asymptote	Pred Formula Inflection Point	Pred Formula Growth Rate
1	A120	17.398228786	150.10439963	2.297714445	0	-1	1	-1	-1	1	1	-1	-1	1	17.3974196	150.063782	2.29913409
2	A121	14.968947851	117.66223562	2.2462220237	1	-1	-1	1	-1	1	0	-1	1	-1	14.9697117	117.298569	2.24655244
3	A123	4.9481043634	84.16861006	2.0225543418	-1	-1	-1	1	-1	-1	1	1	0	1	4.98585492	84.3540972	2.02434751
4	A124	12.301416184	74.19661358	2.4047221115	1	1	1	1	0	1	1	1	1	1	12.3031991	74.2262481	2.40279405
5	A125	20.322295902	109.11546547	2.9638682554	1	-1	-1	0	1	1	-1	1	-1	1	20.3380938	109.072342	2.96095916
6	A126	17.118703765	100.58496727	2.9072905706	0	0	0	0	0	0	0	0	0	0	17.0440605	104.515717	2.90127088
7	A127	12.099705649	116.84677418	2.6091557486	-1	-1	0	-1	1	1	1	1	1	-1	12.0959822	116.763948	2.60919106
8	A128	14.860951869	82.102254026	2.3355415749	1	0	-1	-1	1	-1	1	-1	1	1	14.8724644	82.1413223	2.33505369
9	A129	6.0874256762	84.91481363	2.3023599084	-1	1	-1	-1	-1	1	-1	0	1	1	6.03684768	85.0778077	2.30235112
10	A130	6.4803040036	90.267254774	2.1955144495	-1	1	1	0	-1	-1	1	-1	1	-1	6.48613607	90.254195	2.1960838
11	A131	7.2640057956	66.679671289	1.6868911831	1	-1	1	-1	-1	-1	-1	1	1	0	7.24593557	66.8021177	1.68497592
12	A132	10.433674794	80.393434234	2.2083505212	0	1	-1	1	1	-1	-1	1	1	-1	10.4344888	80.3377245	2.20763678
13	A133	16.104873179	97.04980509	2.4738187872	1	-1	1	1	1	-1	1	0	-1	-1	16.095063	97.0741541	2.4719629
14	A134	10.267028992	63.796339463	2.1510107606	1	1	1	-1	1	1	-1	-1	0	-1	10.3128584	64.1411217	2.15183227
15	A135	17.080122692	132.56459309	3.2735616534	-1	-1	-1	-1	0	-1	-1	-1	-1	-1	17.054943	132.496861	3.27354582
16	A136	19.123979924	139.07073097	2.9108524872	-1	-1	1	1	1	0	-1	-1	1	1	19.1065637	139.254693	2.90975911
17	A137	10.943901338	101.69101258	2.6698281229	-1	1	1	-1	1	-1	0	1	-1	1	10.9460875	101.55948	2.67152546
18	A138	6.7055066839	90.092611481	2.3654817027	-1	0	1	1	-1	1	-1	1	-1	-1	6.71827519	90.1049112	2.36450776
19	A139	7.6904728886	63.760501169	1.8243841086	1	1	-1	-1	-1	0	1	1	-1	-1	7.69609046	63.7493156	1.824996
20	A140	16.687800015	102.35808966	2.9308757181	0	0	0	0	0	0	0	0	0	0	17.0440605	104.515717	2.90127088
21	A141	17.511019989	106.32507148	2.960117006	0	0	0	0	0	0	0	0	0	0	17.0440605	104.515717	2.90127088
22	A142	16.888597346	107.86200339	2.8229889781	0	0	0	0	0	0	0	0	0	0	17.0440605	104.515717	2.90127088
23	A143	10.853550845	87.456056917	2.8110428013	-1	1	-1	1	1	1	1	-1	-1	0	10.8428311	88.0367369	2.81104505
24	A144	12.092743082	84.117822579	1.8985475585	1	1	0	1	-1	-1	-1	-1	-1	0	12.0730379	84.1602838	1.89525883

Figure 6: DoE 10 Weibull Growth Parameter Table

We then copy and paste our Asymptote, Inflection Point, and Growth Rate prediction formulas to our orignal data set and implement them into a new column that has the Weibull Growth Formula which is defined as:

$$\text{Pred Asymptote} \times \left(1 - \exp\left(-\left(\frac{\text{"Time (hrs)"}}{\text{Pred Inflection Point}}\right)^{\text{Pred Growth Rate}}\right)\right)$$

Once JMP computes the column we finally have our Fit Curve Y Prediction. We then use graph builder to compare our Fit Curve approach Y prediction with the actual Y and the other two Y prediction Methods.

5.2 FPC

We implement our functional principal components by using JMP. In JMP we use the Functional data Explorer within JMP's analyze tab. This open a window we we set our input as time our output as Viscosity, TempC, or Y depending on which example we are working on. Then we take our examples experimental setting as our z supplementary inputs and set our ID to Run. After completion a new window appears and we then set our model controls for B Splines on our data. In all of our three examples we decide to have 3 degree spline with the number of knots ranging initially ranging from zero to twenty. When this is finalized and computed we then decide when is the best case for our examples. For our Viscosity Data, we determined that 2 eigenfunction FPC were optimal in our design with degree 3 with 20 knots. Below you can see JMP output for our model. The charts provided, helped determine our best possible model for our FPC estimates. We observe that almost 99 percent of data is represented and that we have the lowest BIC value under two eigenfunctions.

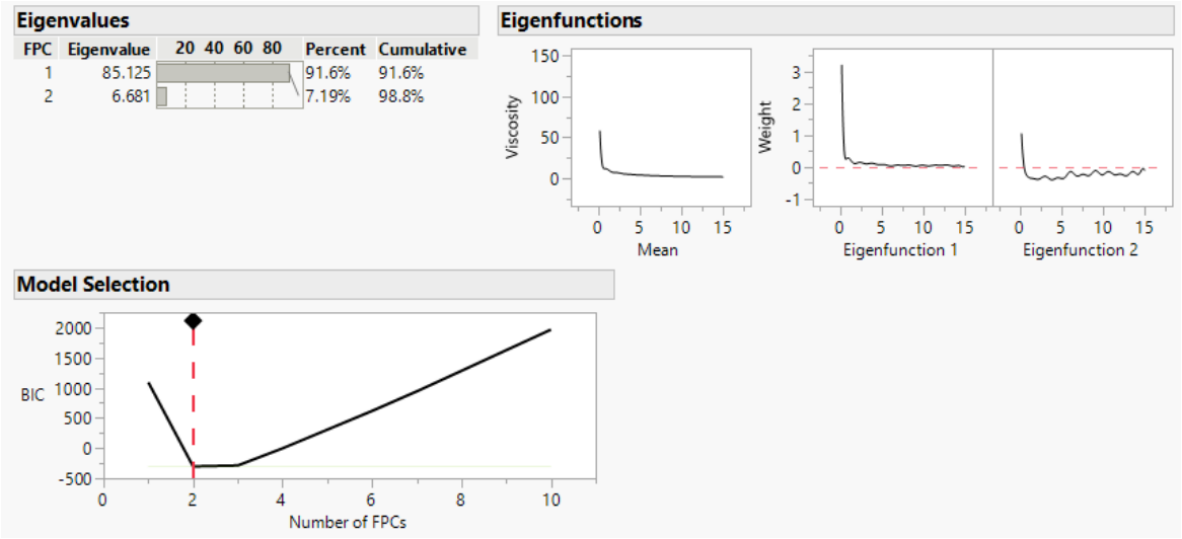


Figure 7: JMP output from Viscosity FPC

After we determine that two eigenfunctions are appropriate we then save our eigenfunction estimates and create a parameter table which is shown below. Our eigenfunctions are now saved under Viscosity FPC 1 and Viscosity FPC 2. The next step that we take is to create a prediction formula for our Viscosity FPC 1 and Viscosity FPC 2 with our experimental factors. Hence we go under JMP's Predictum tab and use the SVEM specifically a neural network 50 bootstrapped Models and 3 TanH first layer Nodes. We set our response as our Viscosity FPC 1 and our inputs as A, B, C, D, and E and have JMP compute our findings. Once complete, we save our prediction formula and repeat this process for our Viscosity FPC 2.

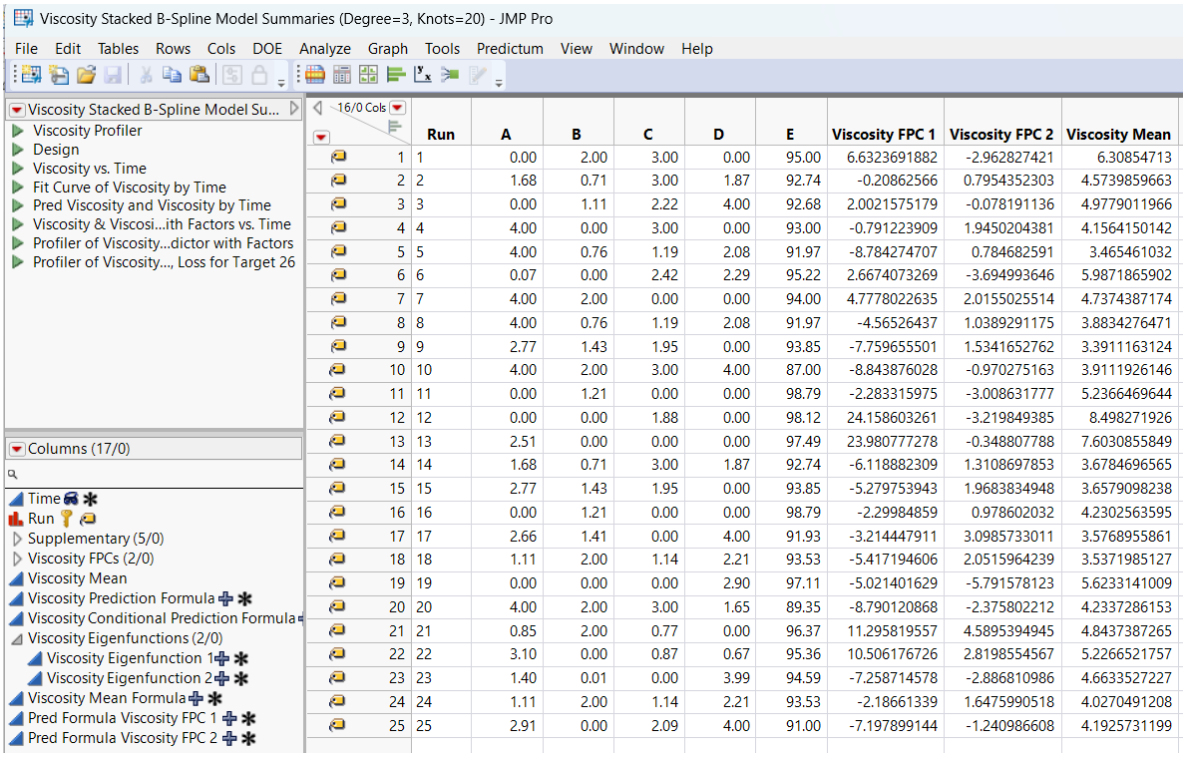


Figure 8: Viscosity FPC Estimates

Viscosity Eigenfunction 1	Viscosity Eigenfunction 2	Viscosity Mean Formula	Pred Formula Viscosity FPC 1	Pred Formula Viscosity FPC 2
0.0547836977	-0.237955323	3.1223998889	6.69774816	-2.8006636
0.0547836977	-0.237955323	3.1223998889	-3.1012768	0.99932353
0.0547836977	-0.237955323	3.1223998889	1.6586683	-0.1062224
0.0547836977	-0.237955323	3.1223998889	-0.7339863	1.95936683
0.0547836977	-0.237955323	3.1223998889	-6.5594034	0.94690461
0.0547836977	-0.237955323	3.1223998889	2.60225408	-3.6374757
0.0547836977	-0.237955323	3.1223998889	4.8001935	2.06474642
0.0547836977	-0.237955323	3.1223998889	-6.5594034	0.94690461
0.0547836977	-0.237955323	3.1223998889	-6.5353812	1.72711272
0.0547836977	-0.237955323	3.1223998889	-8.7382083	-1.0236173
0.0547836977	-0.237955323	3.1223998889	-2.1837683	-1.2355245
0.0547836977	-0.237955323	3.1223998889	24.0465461	-3.231248
0.0547836977	-0.237955323	3.1223998889	23.8708559	-0.3400818
0.0547836977	-0.237955323	3.1223998889	-3.1012768	0.99932353
0.0547836977	-0.237955323	3.1223998889	-6.5353812	1.72711272
0.0547836977	-0.237955323	3.1223998889	-2.1837683	-1.2355245
0.0547836977	-0.237955323	3.1223998889	-3.36745	3.05625301
0.0547836977	-0.237955323	3.1223998889	-3.6695606	1.85253041
0.0547836977	-0.237955323	3.1223998889	-5.0183788	-5.728365
0.0547836977	-0.237955323	3.1223998889	-8.8894183	-2.2105708
0.0547836977	-0.237955323	3.1223998889	11.2300234	4.48508968
0.0547836977	-0.237955323	3.1223998889	10.6443447	2.7823713
0.0547836977	-0.237955323	3.1223998889	-7.0656385	-2.8951817
0.0547836977	-0.237955323	3.1223998889	-3.6695606	1.85253041
0.0547836977	-0.237955323	3.1223998889	-7.0471687	-1.1806496

Figure 9: Viscosity FPC Estimates and FPC Prediction Formulas from SVEM

Once this is complete we copy and paste our Viscosity FPC 1 and Visocsity FPC 2 Prediction Formula's (with the necessary columns) to our original data base so we can create our prediction. Specifcally we implement a column with the following formula:

$$\begin{aligned}&\text{Viscosity Eigenfunction 1} \times \text{Viscosity Eigenfunction 1}\\ &+ \text{Viscosity Eigenfunction 2} \times \text{Viscosity Eigenfunction 2}\\ &+ \text{Viscosity Mean Formula}\end{aligned}$$

This finally creates our final FPC prediction for all 2500 data points in our Viscosity vs Shear Rate data. We then compare and analyze the FPC in graph builder and compare against the other two methods.

For our Fly Ash Case we found that degree 3 with 7 splines was the best fit. Below you can see JMP output for our model. Once again, the charts helped determine our best possible model for our FPC estimates. We observe that almost 99 percent of data is represented and that we have the lowest BIC value under five eigenfunctions.

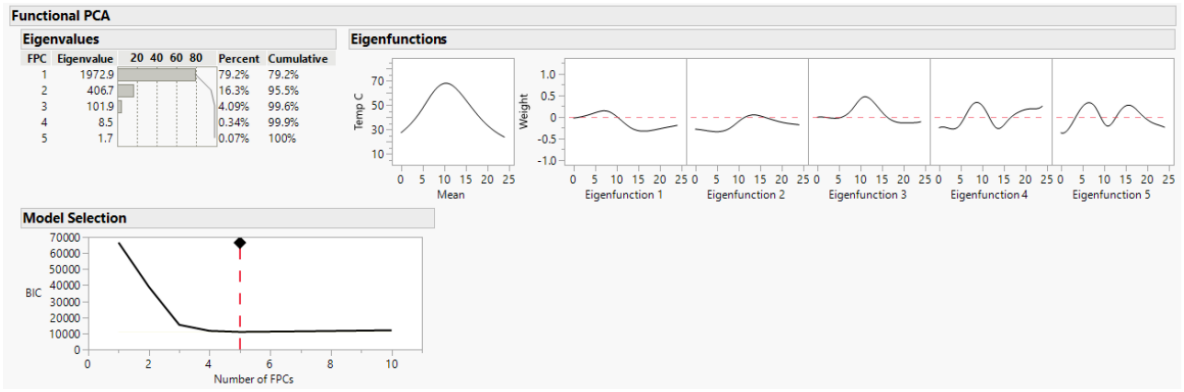


Figure 10: JMP output from Fly-Ash FPC

After determing that five eigenfunctions are appropriate we then save our eigenfunction estimates and create a parameter table which is shown below. Our eigenfunctions are now saved under TempC FPC 1, TempC FPC 2, TempC FPC 3, TempC FPC 4, and TempC FPC 5. The next step that we take is to create a prediction formula for our all five of our TempC FPC's 2 with our experimental factors. Hence we go under JMP's Predictum tab and use the SVEM specifically a neural network 50 bootstrapped Models and 3 TanH first layer Nodes. We set our response as the specified TempC FPC and our inputs as Limestone, Gypsum, Fly Ash, and Amount and have JMP compute our findings.

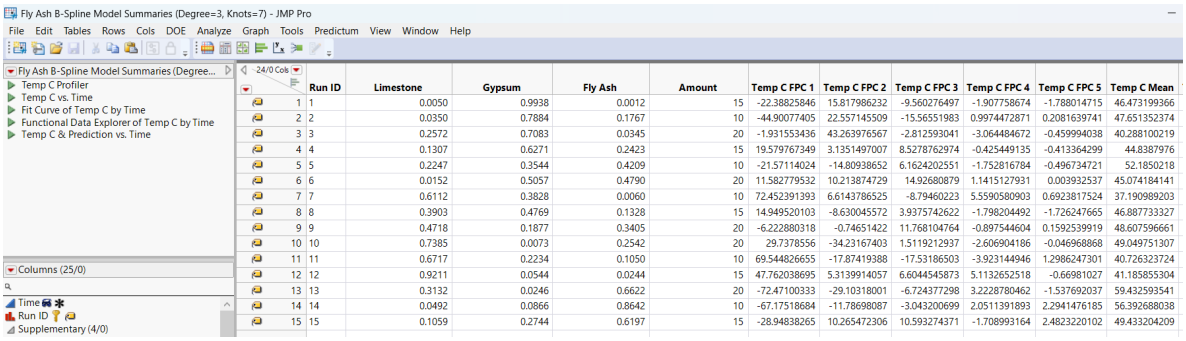


Figure 11: Fly-Ash FPC Estimates

Pred Formula Temp C FPC 1	Pred Formula Temp C FPC 2	Pred Formula Temp C FPC 3	Pred Formula Temp C FPC 4	Pred Formula Temp C FPC 5
-22.186634	16.5839457	-9.570796	-1.8828531	-1.786462
-44.77189	21.7908137	-15.474099	0.9403604	0.20872898
-1.8541081	42.3901738	-2.6197435	-3.0134207	-0.4598926
18.7288822	3.82286864	8.53802659	-0.4877612	-0.413962
-21.59714	-14.538186	5.97906495	-1.7203598	-0.4975868
11.577213	10.9051224	14.9652067	1.09759731	0.00424066
72.7292167	5.74675651	-9.0354808	5.42889034	0.69120014
15.2322285	-7.3076886	3.80083403	-1.8104989	-1.7236592
-6.4490191	-2.6971724	11.497765	-0.9492288	0.15621134
30.0271175	-33.175217	1.68179303	-2.5868259	-0.0458549
69.2559525	-17.114204	-17.291695	-3.6279003	1.29700523
47.8117967	4.86148697	6.10682396	5.02076303	-0.6715553
-72.29975	-28.631457	-6.4846751	3.18755288	-1.5375228
-67.149812	-12.194748	-2.8046074	2.01989947	2.2938799
-28.931266	9.53576914	10.6331463	-1.578259	2.48564261

Figure 12: Fly-Ash FPC Prediction Formulas from SVEM

We save our prediction formulas using SVEM and then copy and paste them over to our original

data set. Like in our Viscosity example we use the same formula mentioned before to compute our final FPC prediction for all 21,468 data points in our Fly-Ash data. We then compare and analyze the FPC in graph builder and compare against the other two methods.

Finally for our DoE 10 Factor Case we found that degree 3 with 3 splines was the best fit. Below you can see JMP output for our model. Once again, the charts helped determine our best possible model for our FPC estimates. We observe that almost 99 percent of data is represented and that we have the lowest BIC value under five eigenfunctions.

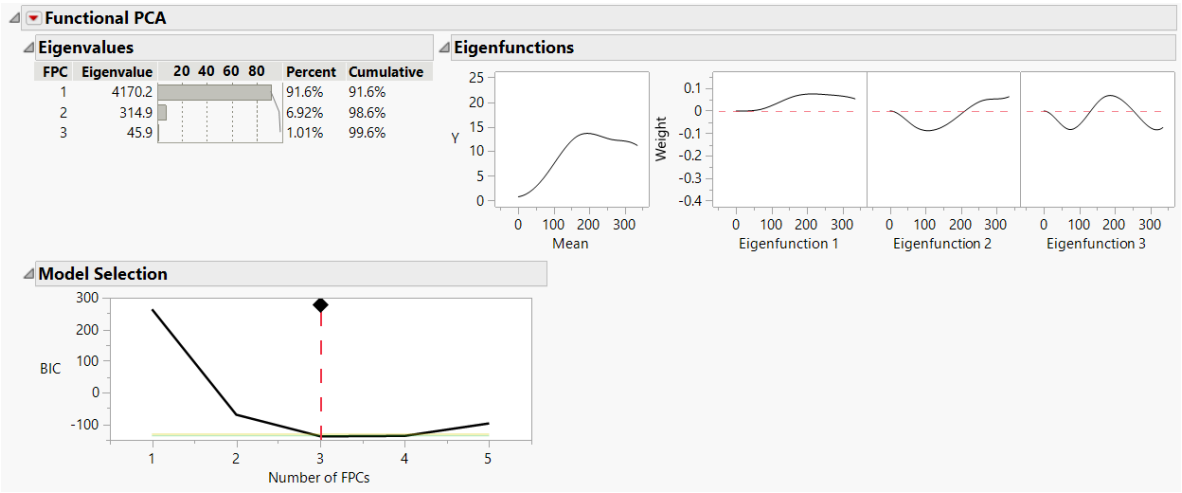


Figure 13: JMP output from DoE 10 FPC

After determing that three eigenfunctions are appropriate we then save our eigenfunction estimates and create a parameter table which is shown below. Our eigenfunctions are now saved under Y FPC 1, Y FPC 2, and Y FPC 3. The next step that we take is to create a prediction formula for our all three of our TempC FPC's 2 with our experimental factors. Hence we go under JMP's Predictum tab and use the SVEM Method specifically a neural network 50 bootstrapped Models and 3 TanH first layer Nodes. We set our response as the specified TempC FPC and our inputs as X1, X2, X3, X4, X5, X6, X7, X8, X9, and X10 have JMP compute our findings.

Y Eigenfunction 1	Y Eigenfunction 2	Y Eigenfunction 3	Y Mean Formula	Pred Formula Y FPC 1	Pred Formula Y FPC 2	Pred Formula Y FPC 3
0.0672599647	-0.052259778	0.0603096669	13.136642808	36.1537799	39.8552951	-11.770518
0.0672599647	-0.052259778	0.0603096669	13.136642808	25.4278128	13.716871	-2.3313991
0.0672599647	-0.052259778	0.0603096669	13.136642808	-109.38355	9.51974565	6.85828497
0.0672599647	-0.052259778	0.0603096669	13.136642808	1.39566456	-25.49411	-8.1302788
0.0672599647	-0.052259778	0.0603096669	13.136642808	113.491317	-10.770133	13.0658978
0.0672599647	-0.052259778	0.0603096669	13.136642808	66.3554282	-7.8320084	5.76091881
0.0672599647	-0.052259778	0.0603096669	13.136642808	-14.062828	14.4939738	6.71863766
0.0672599647	-0.052259778	0.0603096669	13.136642808	40.2570277	-23.346495	-9.036965
0.0672599647	-0.052259778	0.0603096669	13.136642808	-93.052698	5.09210218	6.15261479
0.0672599647	-0.052259778	0.0603096669	13.136642808	-90.134081	8.50445835	4.6892704
0.0672599647	-0.052259778	0.0603096669	13.136642808	-73.756652	-5.5885741	0.18971231
0.0672599647	-0.052259778	0.0603096669	13.136642808	-29.40032	-10.870457	-2.9379029
0.0672599647	-0.052259778	0.0603096669	13.136642808	54.1506361	-11.957765	-2.0955193
0.0672599647	-0.052259778	0.0603096669	13.136642808	-26.272251	-22.678643	-9.8017183
0.0672599647	-0.052259778	0.0603096669	13.136642808	47.901057	29.2855394	2.4167215
0.0672599647	-0.052259778	0.0603096669	13.136642808	71.7888993	37.8931626	-7.5916218
0.0672599647	-0.052259778	0.0603096669	13.136642808	-28.235429	5.64797864	-3.6917341
0.0672599647	-0.052259778	0.0603096669	13.136642808	-84.813415	6.23025448	5.90712013
0.0672599647	-0.052259778	0.0603096669	13.136642808	-70.161036	-10.885016	-1.8235276
0.0672599647	-0.052259778	0.0603096669	13.136642808	66.3554282	-7.8320084	5.76091881
0.0672599647	-0.052259778	0.0603096669	13.136642808	66.3554282	-7.8320084	5.76091881
0.0672599647	-0.052259778	0.0603096669	13.136642808	66.3554282	-7.8320084	5.76091881
0.0672599647	-0.052259778	0.0603096669	13.136642808	-22.045457	-7.3418117	-1.8341171
0.0672599647	-0.052259778	0.0603096669	13.136642808	-6.5316858	-9.0304848	-8.2026639

Figure 14: DoE 10 FPC Estimates and FPC Prediction Formulas from SVEM

We save our prediction formulas using SVEM and then copy and paste them over to our original data set. Like in our Viscosity example we use the same formula mentioned before to compute our final FPC prediction for all 360 data points in our DoE 10 Factor data. We then compare and analyze the FPC in graph builder and compare against the other two methods.

5.3 Bayesian Hierarchical Two-Stage Mixed-Effects

Analyzing Enrique Del Castillo, Bianca M. Colosimo, and Hussam Alshraideh MATLAB code, we observe that the code is outdated. The MATLAB code named "BayesianOptimizationPrograms" that I was sent from Dr.Castillo was created in 2008 with the last modification of the MATLAB code being July of 2009. This MATLAB code is 14 years old and contained MATLAB functions within MATLAB functions to perform the associated Bayesian Hierarchical Two-Stage Mixed-Effects within their papers Examples. What I discovered after succesffuly running thier code for Example 3: Metal Injections (which is an example that closely resembles what we are trying to perform in this paper) is that some of the MATLAB function files were unnecssary for us to utlilize, specifically the analysis parts. In this paper we are focused on prediction, hence the MATLAB functions we are concerned with is the MCMC Gibbs sampling and Bayesian optimization analysis which was performed in the MCMCMixed Effects MATLAB File. This function applies a Markov Chain Monte Carlo (MCMC) approach for estimating the parameters of a mixed effects model. Specifically, the function starts with defining priors for the coefficients (beta), the inverse of the covariance matrix (DInv), and the residual variance (sigma2). It then generates initial samples from these priors to start the MCMC simulation. These priors are mentioned in the previous section and in Appendix A of Castillo's research paper. Now, in each iteration of the MCMC, the function updates the parameters by sampling from their full conditional distributions. [**delcastillo2012**] The Chib and Carlin algorithms are applied for sampling beta. Also, a loop attempts to iterate between bi and D updates to reduce autocorrelation in the Ds. Finally, the function stores the parameter values from each iteration of the MCMC in the pars cell array. The pars array is then returned by the function and can be used to analyze the posterior distributions of the parameters. I then transform the pars matrix into the smaller matrix of our desired parameters and then calculate their mean posterior parameter for each run. This is how we obtain our estimates for our parameters relating back to our experimental designs. All in all, there are a few problems with our MATLAB code that we have determined after successfully running it. First, the

MATLAB functions is coded so that the model assumes a linear relationship between the dependent variable and independent variables with both fixed and random effects. This function allows different covariance matrices (S and Ssecond) for the two terms in the Laird and Ware mixed effects model. In all of our examples we do not utilize a linear model so there occurs a problem. Second, as we are aware, the past 14 years has brought upon massive technological advancements in coding. The code is outdated and not optimized with the latest packages and up to date technology. However, I was successfully in running and analyzing similar results as Enrique Del Castillo, Bianca M. Colosimo, and Hussam Alshraideh did in there Example 3: Metal Injections within their paper, the application to our design of experiment project was not a great success. As you will see in the next section "Results" I have provided the MATLAB results from our Viscosity data example and have shown that the what our code produces is no where near where our actual Viscosity measurements should be. Further analyzing of our results by transferring them into a JMP Data table, I produced a actual by predicted plot that had no similarity. From this, I decided that the best way to move forward with our Bayesian Hierarchical Two-Stage Mixed-Effects model was to translate the associated MATLAB code over to R. R is my preferred language and I was able to apply the necessary and most up to date packages and functions to properly run our Viscosity vs. Shear Rate example. To transfer the MATLAB code into R I made to use the brms package offered in STAN. The brms package is used to fit Bayesian generalized (non-)linear multivariate multilevel models using 'Stan' for full Bayesian inference. A wide range of distributions and link functions are supported, allowing users to fit – among others – linear, robust linear, count data, survival, response times, ordinal, zero-inflated, hurdle, and even self-defined mixture models all in a multilevel context. Further modeling options include both theory-driven and data-driven non-linear terms, auto-correlation structures, censoring and truncation, meta-analytic standard errors, and quite a few more. In addition, all parameters of the response distribution can be predicted in order to perform distributional regression. Prior specifications are flexible and explicitly encourage users to apply prior distributions that actually reflect their prior knowledge. Models can easily be evaluated and compared using several methods assessing posterior or prior predictions. Hence, the brms package is exactly what we are looking for to perform this Bayesian Hierarchical Two-Stage Mixed-Effects Model. Within our brms package, I used function bf() which applied interactions (including three-way and four-way interactions) between the predictors Scale1, DecayRate1, Scale2, and DecayRate2. The (0 + Scale1 + DecayRate1 + Scale2 + DecayRate2 — Run) part of the code suggests a multilevel model where the intercept and slopes for the predictors vary by Run. This is implements our two stage mixed effects model that we desire as well as ensure our model is predicting out estimates for each specific run. I then use the brm() function offered in the brms package to run our Bayesian generalized (non-)linear multivariate multilevel model. After evaluation we decided to use Scheffé's polynomial in our model to understand the relationship between our input parameters and response while excluding an intercept. Due to computational contestants from my Laptop/Computer Model we ran this model on 4000 iterations with a maxtreedepth of 17. Once our model converged and finalized computation we utilized the posteriorsamples() function I then extract the posterior distribution of our desired parameters for each run, take the mean of them and I am left with our mean posterior parameter estimates for each run. These estimates are then transferred over to JMP for us to compare with our two other methods.

6 Results

6.1 Example 1: Viscosity Data

6.1.1 Fit Curve

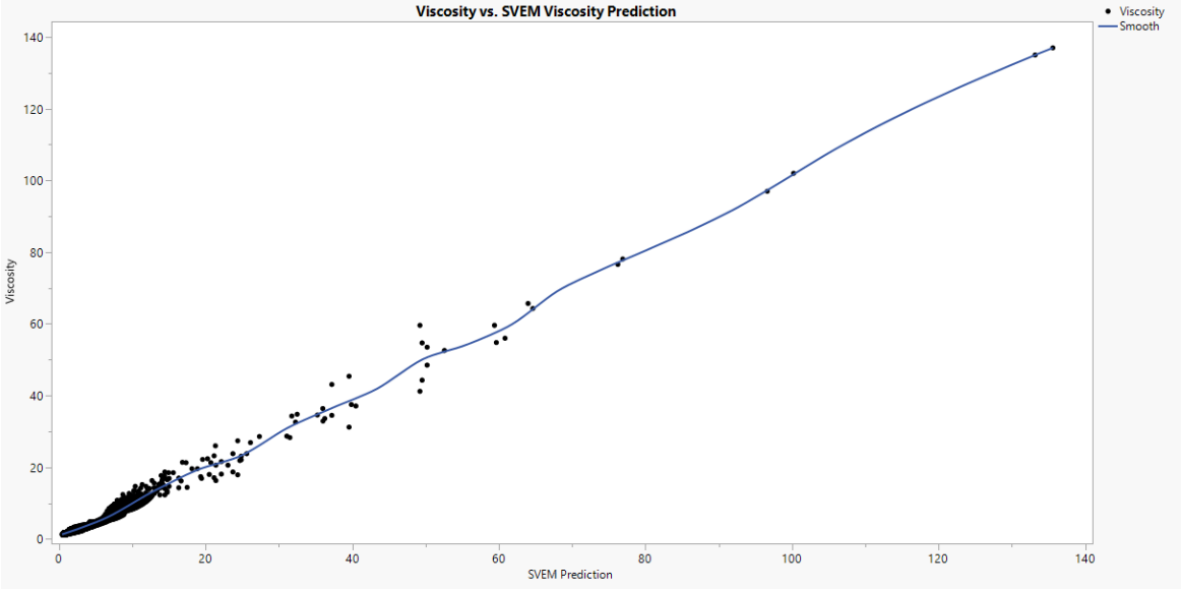


Figure 15: Actual By Predicted Plot for the Fit Curve Method for our Viscosity Example

As mentioned in the prior section we observe the actual by predicted plot created from graph builder in JMP for our Viscosity data using the Fit Curve approach and SVEM. We used a 4 parameter Bi-exponential model for fitting our data and to produce our predictions for Viscosity. Observing our predicted by actual plot we can see that the fit is well and that there is a clear linear trend, exhibiting that our predictions are accurate to the actual Viscosity data.

6.1.2 FPC

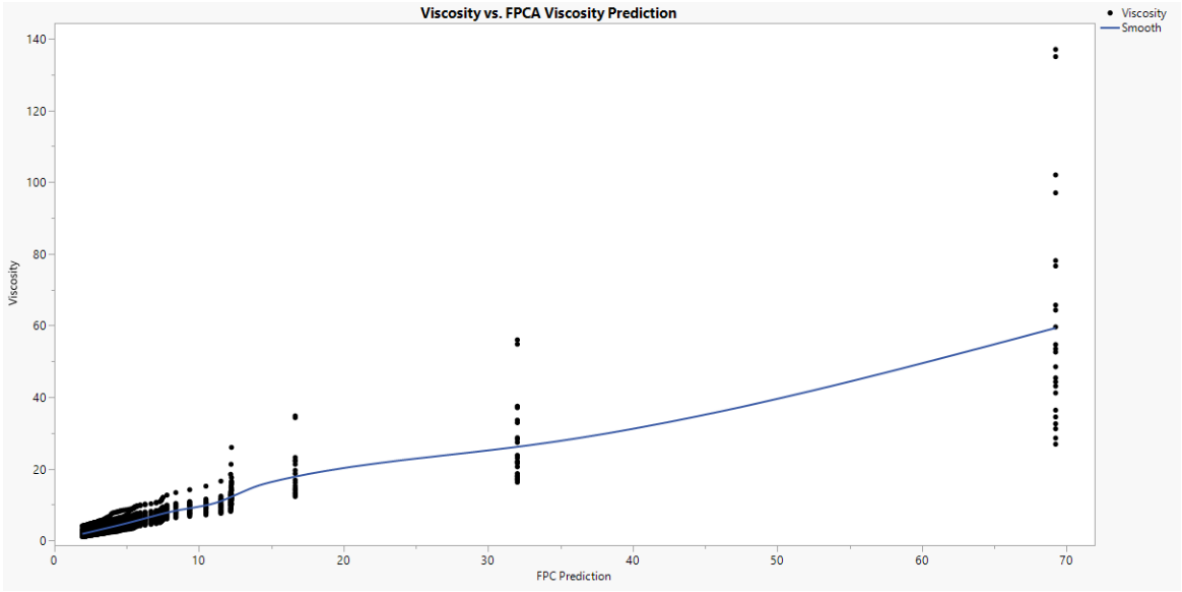


Figure 16: Actual By Predicted Plot for the FPC Method for our Viscosity Example

We now observe the actual by predicted plot created from graph builder in JMP for our Viscosity data using FPC. Using FPC and SVEM to produce our predictions for Viscosity, we observe our predicted by actual plot above. We can see that the fit is not well and that there is a clear linear trend, exhibiting that our predictions are accurate to the actual Viscosity data.

6.1.3 Bayesian Hierarchical Two-Stage Mixed-Effects: MATLAB Code

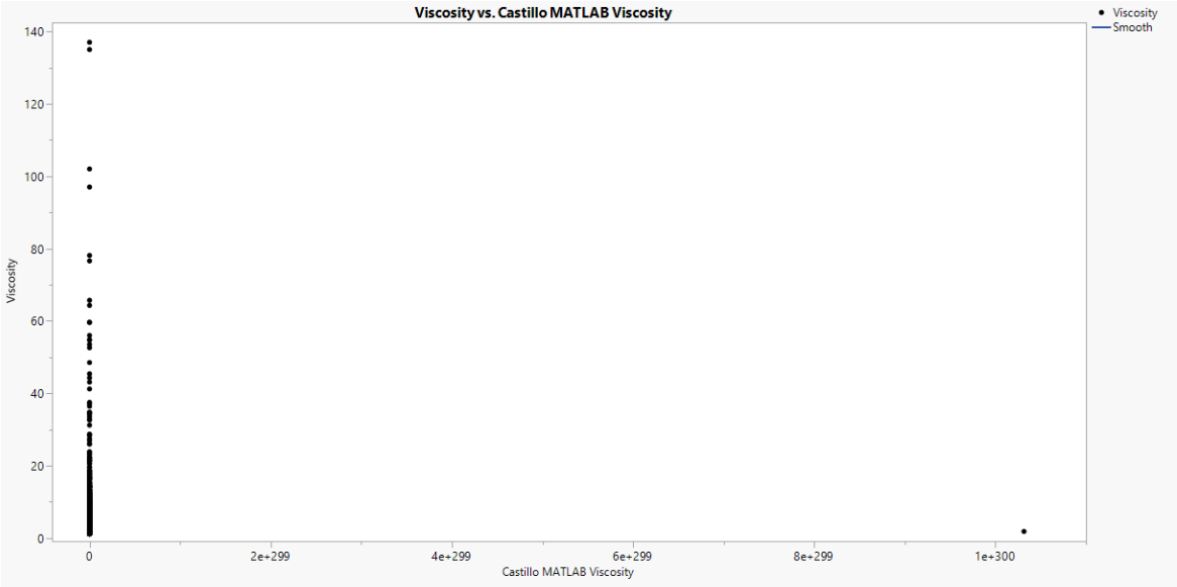


Figure 17: Actual By Predicted Plot for the our Bayesian Hierarchical Two-Stage Mixed-Effects: MATLAB Code using MATLAB for our Viscosity Example

I include my MATLAB code results into this portion of the article because of the extraneous work that was involved in producing the Viscosity predicted values. While the results that we observe show that the method is poor at prediction, the work that I put into running the MATLAB code for Bayesian Hierarchical Two-Stage Mixed-Effects is worthwhile to share. First, We observe that the actual by predicted plot from the MATLAB code for our Bayesian Hierarchical Two-Stage Mixed-Effects. The results that we produce are visually confusing. The results do not have any relevance to our actual Viscosity values and it informs us that our method of Bayesian Hierarchical Two-Stage Mixed-Effects is inefficient at predicting viscosity values. I further investigate this MATLAB code for Bayesian Hierarchical Two-Stage Mixed-Effects below.

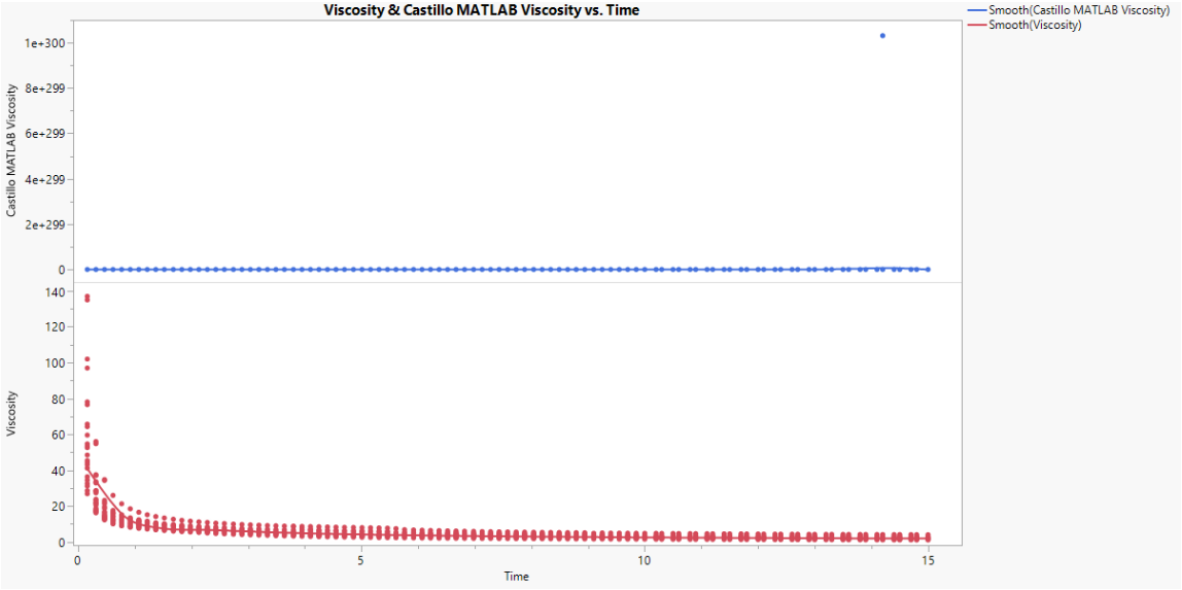


Figure 18: Comparison of Bayesian Hierarchical Two-Stage Mixed-Effects using MATLAB vs. Viscosity

We once again observe the MATLAB code for Bayesian Hierarchical Two-Stage Mixed-Effects and have its results directly compared with the actual viscosity values by having a plot by plot comparison. We observe that the viscosity values produce an exponentially decreasing curve while the predicted viscosity values produce a linear line that is constantly near zero.

6.1.4 Bayesian Hierarchical Two-Stage Mixed-Effects: R Code

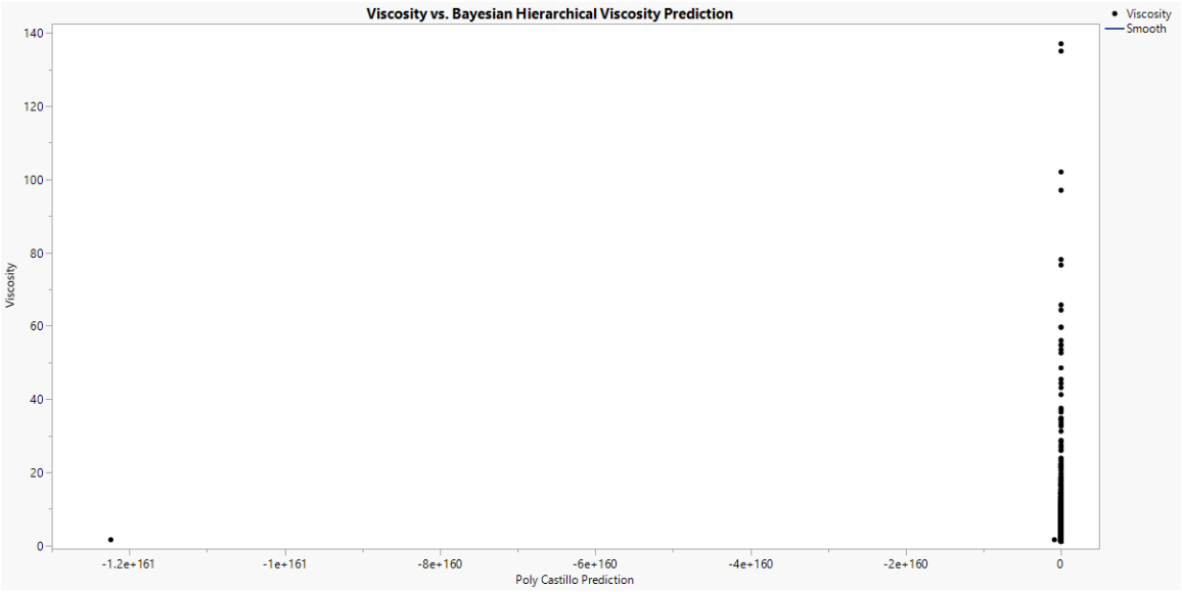


Figure 19: Actual By Predicted Plot for the Bayesian Hierarchical Method for our Viscosity Example

As previously described we transformed our MATLAB code into R in order to improve our Bayesian Hierarchical Two-Stage Mixed-Effects method. Once we obtained prediction results we modeled them against actual viscosity values and once again found similiar results to the MATLAB code actual by predicted plot. We see that visually our actual by predicted plot is a straight vertical linear line which indicates that our Bayesian Hierarchical Two-Stage Mixed-Effects method is very poor at prediction.

6.1.5 Method Estimates

Viscosity	Fit Curve Prediction	FPC Prediction	Bayes Hierarchical Prediction	Fit Curve Difference	FPC Difference	Bayes Difference	Fit Curve AVG Diff	FPC AVG Diff	Bayes AVG Diff
78.1	76.89427869	69.22803161	0.541896957	1.20572131	8.87196839	77.55810304	0.012925112	-0.2119564	4.9869E+157
59.6	49.22557585	69.22803161	0.288458804	10.37442415	-9.62803161	59.3115412			
65.7	63.97984407	69.22803161	0.027420118	1.72015593	-3.52803161	65.67257988			
59.6	59.39299388	69.22803161	0.345530894	0.20700612	-9.62803161	59.25446911			
31.2	39.55436736	69.22803161	0.094924029	-8.35436736	-38.02803161	31.10507597			
64.3	64.63262114	69.22803161	0.040752489	-0.33262114	-4.92803161	64.25924751			
76.6	76.22211526	69.22803161	0.031609765	0.37788474	7.37196839	76.56839024			
45.4	39.55436736	69.22803161	-19.51297563	5.84563264	-23.82803161	64.91297563			
34.5	37.20136497	69.22803161	-0.443893544	-2.70136497	-34.72803161	34.94389354			
28.6	27.32424489	69.22803161	-0.175390515	1.27575511	-40.62803161	28.77539052			
48.5	50.19964946	69.22803161	0.047330163	-1.69964946	-20.72803161	48.45266984			
135	133.1606302	69.22803161	0.069670561	1.8393698	65.77196839	134.9303294			
137	135.5764018	69.22803161	-0.102476073	1.4235982	67.77196839	137.1024761			
41.2	49.22557585	69.22803161	0.002440019	-8.02557585	-28.02803161	41.19755998			
43.1	37.20136497	69.22803161	-0.085166595	5.89863503	-26.12803161	43.1851666			
53.5	50.19964946	69.22803161	0.207553185	3.30035054	-15.72803161	53.29244682			
52.6	52.57140711	69.22803161	0.030895609	0.02859289	-16.62803161	52.56910439			
44.3	49.51835389	69.22803161	0.10600015	-5.21835389	-24.92803161	44.19399985			
36.4	35.97740312	69.22803161	-0.76339148	0.42259688	-32.82803161	37.16339148			
26.9	26.1063996	69.22803161	0.079887992	0.7936004	-42.32803161	26.82011201			
102	100.2035109	69.22803161	0.004605224	1.7964891	32.77196839	101.9953948			
97	96.61963834	69.22803161	-0.204152368	0.38036166	27.77196839	97.20415237			
32.6	32.2415862	69.22803161	-0.001586551	0.3584138	-36.62803161	32.60158655			
54.7	49.51835389	69.22803161	-1.094220447	5.18164611	-14.52803161	55.79422045			
34.6	35.24166006	69.22803161	-1.03100053	-0.64166006	-34.62803161	35.63100053			

Figure 20: Method Estimates Comparison for Viscosity Example

Above we are presented a sample of our predicted estimates from each Method used in our Viscosity Example. As we can see the Fit Curve method provides us with predicted values that are very similar to those of the actual values for Viscosity. This is shown through direct comparison as well as calculating the average difference, which we find to be 0.01 difference. Thus providing the evidence that for our Viscosity Example Fit Curve produces the most accurate predicted values. In addition, we observe that our Bayesian Hierarchical Two-Stage Mixed Effects model in R provide us predicted values that are no where near the actual Viscosity values, while our FPC fairly accurate predicted values for Viscosity. FPC however looks to be only calculating and overall predicted value for all 25 runs instead of calculating a predicted value for each specific run, hence why it is less accurate than the Fit Curve Method. We now move on to compare plots of the three methods used.

6.1.6 Summary

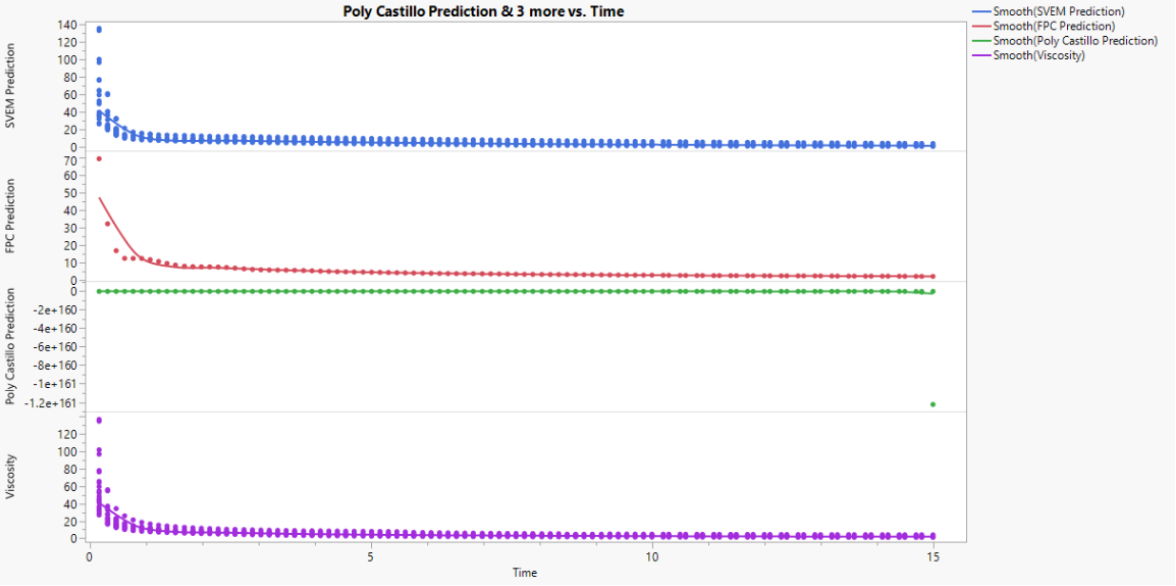


Figure 21: JMP Graph Builder Comparison of Viscosity Example

Above we have our 3 methods: Fit Curve, FPC, and Bayesian Hierarchical Two-Stage Mixed-Effects Model compared alongside our Actual Viscosity Data. We can see that overall shape for Fit Curve and FPC is very similiar to our Actual Viscosity values. We also observe that the scales at which these predicted Viscosity values are at for Fit Curve and FPC are similiar to the actual values for Viscosity. Now when we observe our Bayesian Hierarchical Two-Stage Mixed-Effects Model we see that not only the shape of our predicted values are off but also the scale that is predicted is no where near where the actual Viscosity values. From this, we can conclude that Fit Curve and FPC are much more effective methods at prediction compared to the Bayesian Hierarchical Two-Stage Mixed-Effects Model.

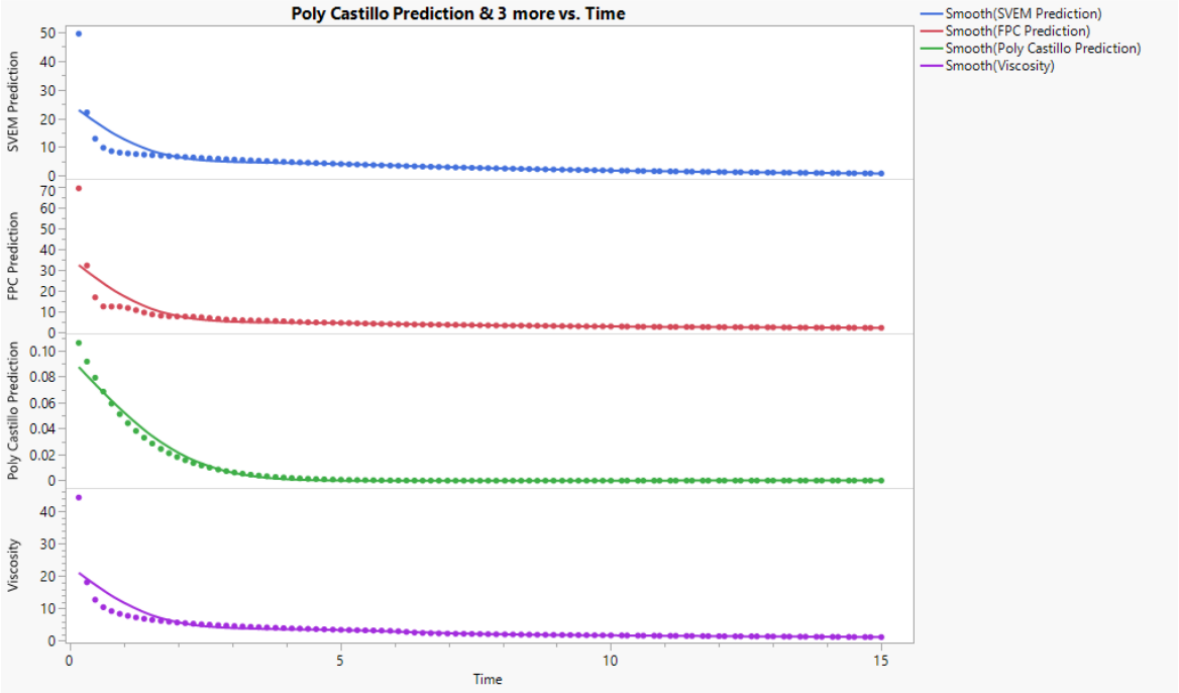


Figure 22: JMP Graph Builder Comparison of Viscosity Example Run 18

Above we have a specific Run picked out from our Viscosity data. I chose to pick Run 18 out of our 25 runs because it was the best fitting curve for all of our methods. However, when you observe the scale for our Bayesian Hierarchical Two-Stage Mixed-Effects Model we once again see that the scale is much smaller compared to the actual Viscosity values. This once again shows that our Bayesian Hierarchical Two-Stage Mixed-Effects Model is not effective at prediction, but I do also believe that because there is an exponentially decaying curve displayed through out Bayesian Hierarchical Two-Stage Mixed-Effects Model Viscosity prediction that there is opportunity for further research to explore deeper in order to optimize and create Bayesian Hierarchical Two-Stage Mixed-Effects Models for much improved prediction.

6.2 Example 2: Fly-Ash Data

6.2.1 Fit Curve

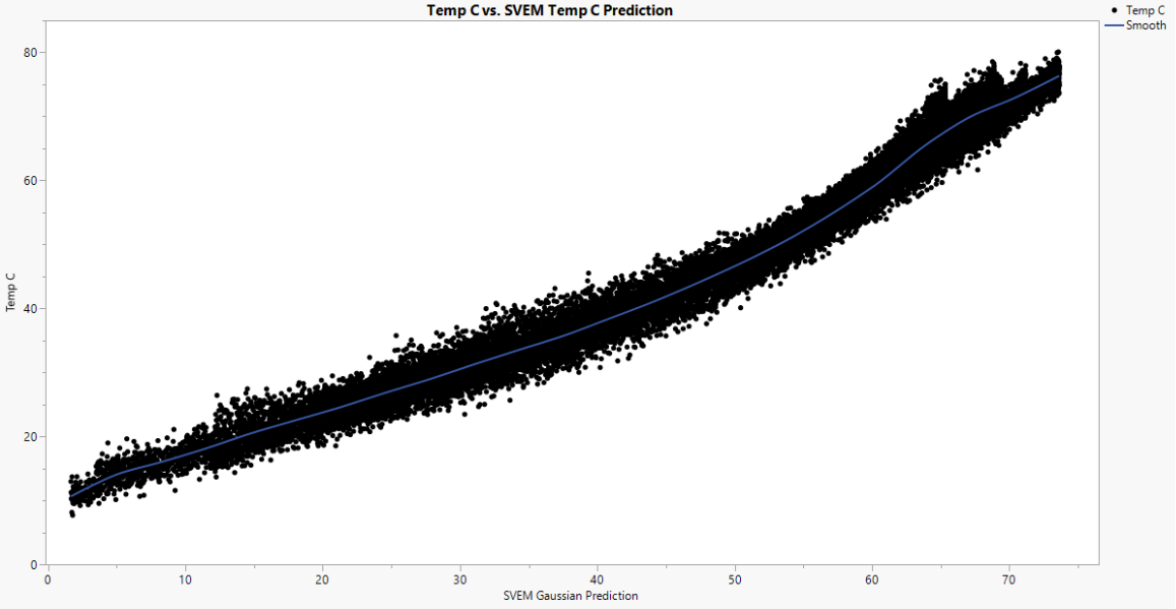


Figure 23: Actual By Predicted Plot for the Fit Curve Method for our Fly-Ash Example

Observing our Fly-Ash Data with the Fit cruve produced actual by predicted plot we see that our Fit Curve method performs well from the linear trend that is displayed. This result was expected due to the success that occurred in our first Viscosity example.

6.2.2 FPC

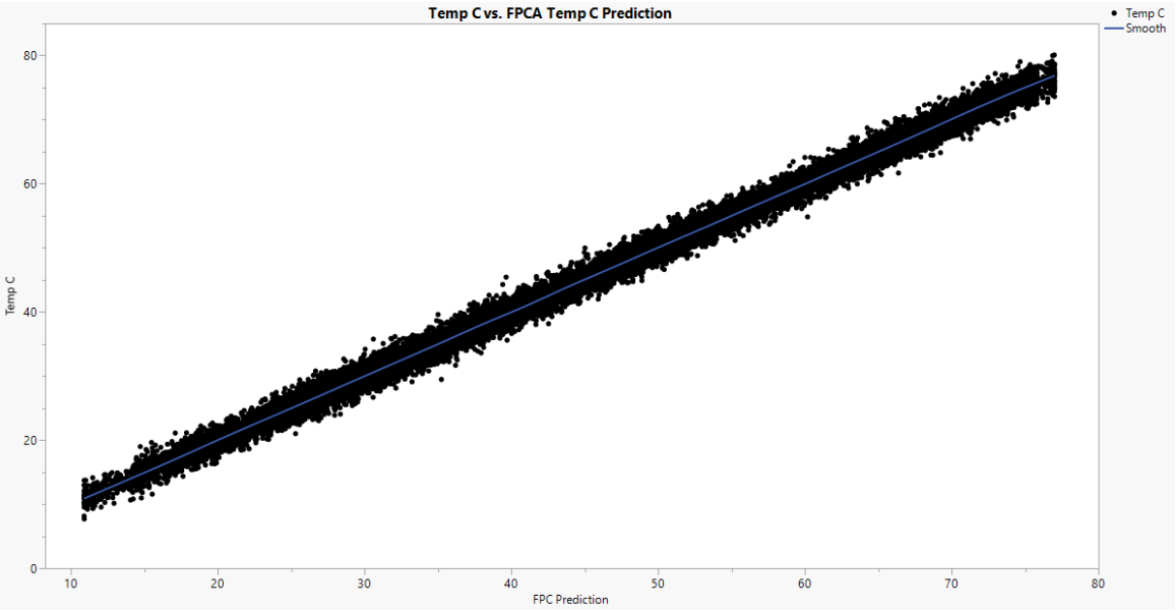


Figure 24: Actual By Predicted Plot for the FPC Method for our Fly-Ash Example

Looking at our actual by predicted plot produced from FPC, we see that we actually have a improved linear and straight trend when compared to the previous Fit Curve method. This was not expected

due to the first viscosity example results where we saw that Fit Curve produced a straighter and more linear line compared to FPC. Now with Fly-Ash data, this result is now switched and it looks like FPC produces better predicted values compared to the Fit Curve method.

6.2.3 Bayesian Hierarchical Two-Stage Mixed-Effects: R Code

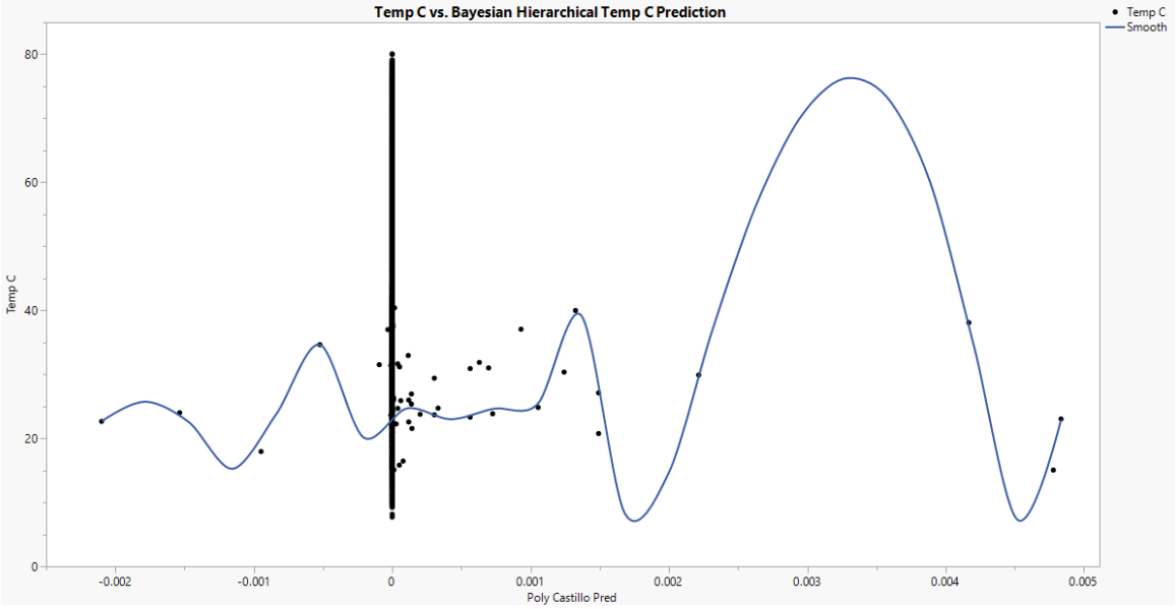


Figure 25: Actual By Predicted Plot for the Bayesian Hierarchical Method for our Fly-Ash Example

Finally, we observe our Bayesian Hierarchical Two-Stage Mixed-Effects R Code where we have an actual by predicted plot that has a horrendous two curve trend. This result is concerning because there are two trends occurring which indicates that there is no similiarty between the predicted values vs. the actual values. This further shows that the Bayesian Hierarchical Two-Stage Mixed-Effects method is very poor at prediction.

6.2.4 Method Estimates

Temp C	FPC Prediction	Fit Curve Gaussian Prediction	Bayes H Pred	Fit Curve Diff	FPC Diff	Bayes Diff	Fit Curve AVG Diff	FPC AVG Diff	Bayes AVG Diff
26.06025748	24.69849559	20.36233878	1.02591E-05	5.6979187	1.36176189	26.06024722	0.343778951	0.00068568	47.06262094
23.85572424	24.73533153	20.42276942	3.67892E-11	3.43295482	-0.87960729	23.85572424			
24.98648724	24.77452623	20.48708677	0	4.49940047	0.21196101	24.98648724			
25.25863837	24.81146884	20.54772391	0	4.71091446	0.44716953	25.25863837			
22.55245157	24.85077749	20.61226028	0.000119586	1.94019129	-2.29832592	22.55233198			
27.76963908	24.88782792	20.67310318	0	7.0965359	2.88181116	27.76963908			
26.33225613	24.92957265	20.74167036	0	5.59058577	1.40268348	26.33225613			
25.81461031	24.96673539	20.80272447	2.42547E-24	5.01188584	0.84787492	25.81461031			
26.9271262	25.00627919	20.86770306	0.000138764	6.05942314	1.92084701	26.92698744			
26.91590129	25.04355211	20.9289614	-5.70202E-07	5.98693989	1.87234918	26.91590186			
24.0898282	25.08321359	20.99415656	-1.0058E-135	3.09567164	-0.99338539	24.0898282			
25.99652141	25.12059786	21.05561834	1.15319E-98	4.94090307	0.87592355	25.99652141			
25.17932021	25.1627202	21.12488039	9.95501E-10	4.05443982	0.01660001	25.17932021			
24.69576352	25.2002204	21.18655095	5.50592E-10	3.50921257	-0.50445688	24.69576352			
24.6145931	25.24012462	21.25218323	-5.16046E-81	3.36240987	-0.62553152	24.6145931			
24.09040938	25.27773853	21.31405559	3.67322E-52	2.77635379	-1.18732915	24.09040938			
26.46130887	25.31776418	21.37990186	5.7222E-272	5.08140701	1.14354469	26.46130887			
27.25819228	25.35549296	21.4419752	0	5.81621708	1.90269932	27.25819228			
26.15634548	25.39800491	21.51192383	0	4.64442165	0.75834057	26.15634548			
24.57208186	25.43585325	21.57420337	1.84691E-20	2.99787849	-0.86377139	24.57208186			
25.50065392	25.47612923	21.64048136	0	3.86017256	0.02452469	25.50065392			
24.79933759	25.5140948	21.70296016	0	3.09637743	-0.71475721	24.79933759			
25.63436773	25.55449595	21.7694494	1.4203E-152	3.86491833	0.07987178	25.63436773			
28.24888208	25.59257991	21.83212659	1.2177E-16	6.41675549	2.65630217	28.24888208			
23.64410209	25.63549348	21.90275302	-1.24968E-24	1.74134907	-1.99139139	23.64410209			
24.98780735	25.67370062	21.96563369	0	3.02217366	-0.68589327	24.98780735			

Figure 26: Method Estimates Comparison for FlyAsh Example

Above we obserev our chart for comparison of actual values and predicted values from each of our three methods. We observe that our FPC method produces that most accurate predicted values for TempC from FPC providing the smallest value of average difference between its predicted TempC values and actual TempC values. Our Fit Curve Method also provides accurate predicted values compared to TempC actual values, however our Bayesian Hierarchical Two Stage Mixed-Effects Model provides predicted values that are once again no where near the actual values of our response. This is demonstrated through direct comparison of our actual and predicted values for Bayesian Hierarchical Two Stage Mixed-Effects Model and the very high value we recieve when computing the average difference between predicted and actual values. We now move on to compare plots of the three methods used.

6.2.5 Summary

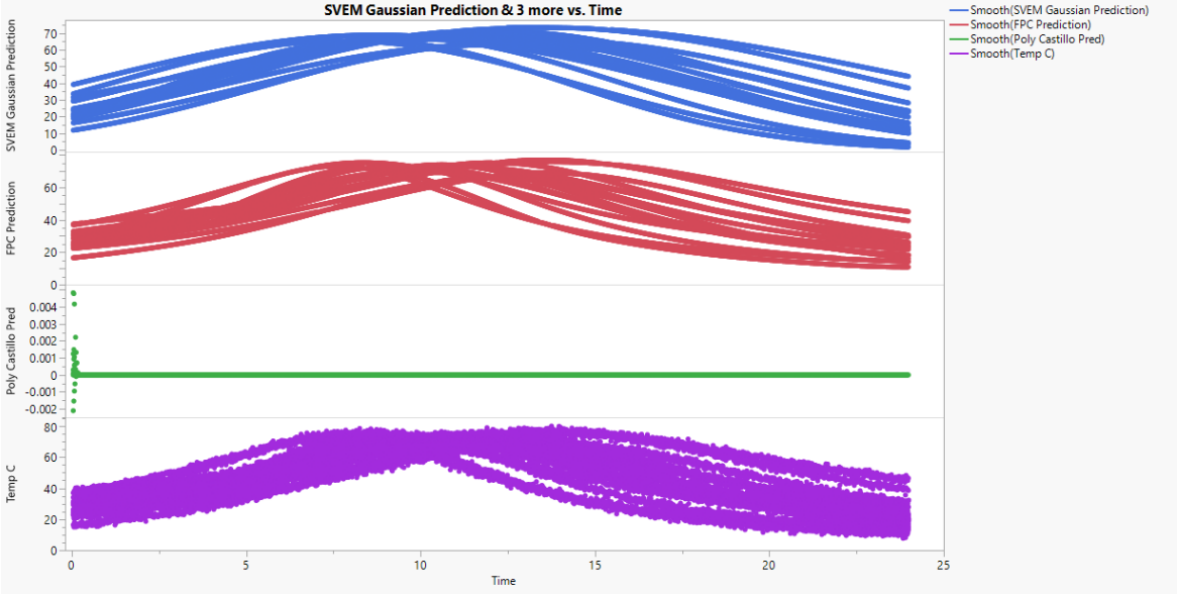


Figure 27: JMP Graph Builder Comparison of Fly Ash Example

Above we have our 3 methods: Fit Curve, FPC, and Bayesian Hierarchical Two-Stage Mixed-Effects Model compared alongside our Actual TempC Data. We can see that overall shape for Fit Curve and FPC is very similiar to our Actual TempC values. We also observe that the scales at which these predicted Viscosity values are at for Fit Curve and FPC are similar to the actual values for TempC. Now when we observe our Bayesian Hierarchical Two-Stage Mixed-Effects Model we see that not only the shape of our predicted values are off but also the scale that is predicted is no where near where the actual Viscosity values. From this, we can conclude that Fit Curve and FPC are much more effective methods at prediction compared to the Bayesian Hierarchical Two-Stage Mixed-Effects Model. This is a very similiar conclusion that we determined in example one. As mentioned before, the small difference in this TempC example compared to our Viscosity example is the comparison with the actual by predicted plots. Comparing our Fit Curve and FPC actual by predicted plot we can see that our FPC actual by predicted plot has a much more linear trend than Fit Curve's. This distinction in linear fit can make us conclude that for our Fly-Ash data FPC is better at predicting than Fit Curves.

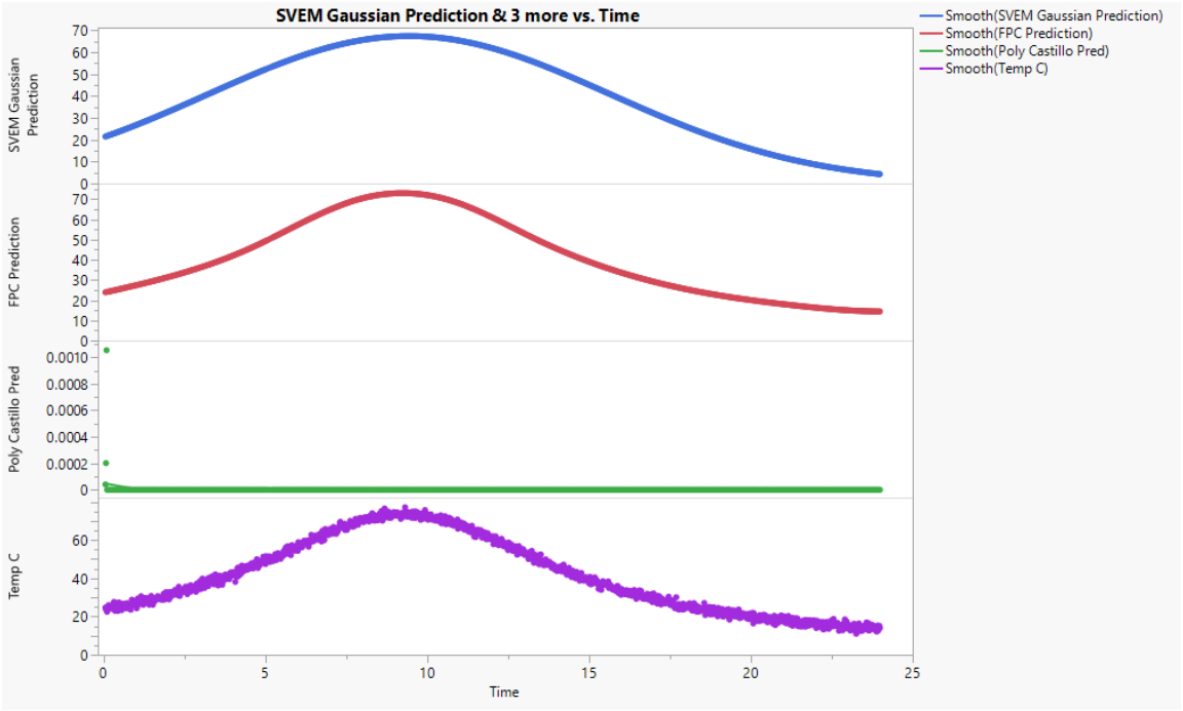


Figure 28: JMP Graph Builder Comparison of Fly Ash Example Run 12

Similar to what I did in our Viscosity data, I looked at each Run's curve to see how each method performed prediction wise. Unlike our Viscosity data, I have found that with our Fly-Ash data there is no Run where the Bayesian Hierarchical Two-Stage Mixed-Effects Model produced any prediction curve that had any similar shape to our actual TempC values. This further provides evidence that the Bayesian Hierarchical Two-Stage Mixed-Effects Model is to suitable for prediction or for any further research in investigating how one can improve and optimize design within the model for prediction accuracy.

6.3 Example 3: DoE 10 Factors Data

6.3.1 Fit Curve

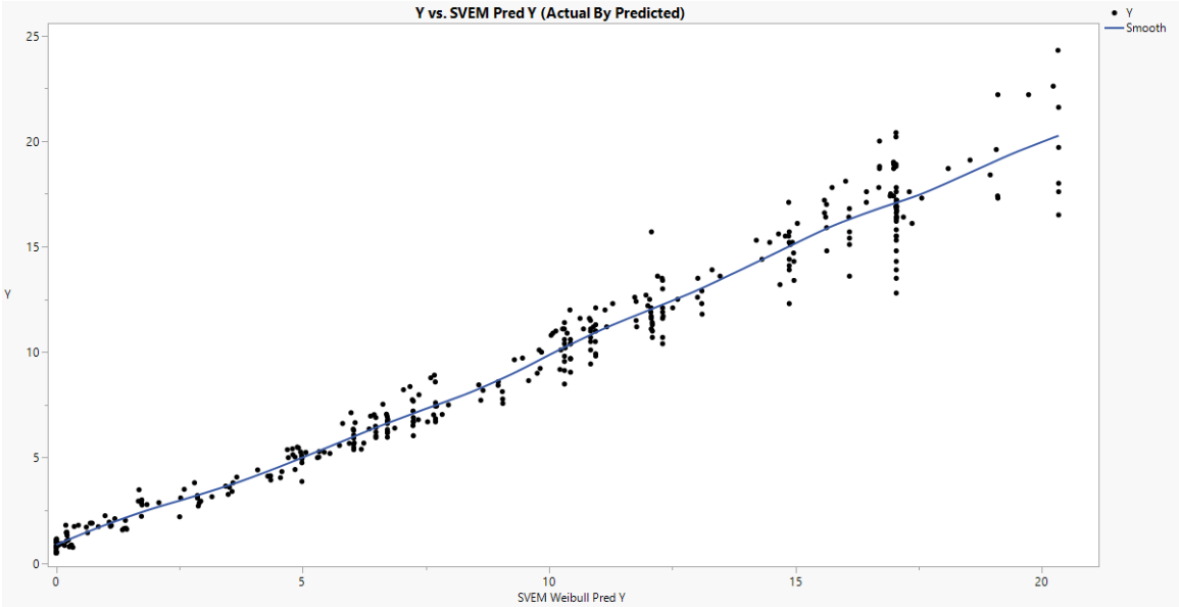


Figure 29: Actual By Predicted Plot for the Fit Curve Method for our DoE 10 Factor Example

We observe our third example's Fit Curve actual by predicted plot where we can see that there is once again a very good looking plot that has a linear straight trend. We can observe that toward the tail end as prediction values get higher that there is increased variation. Overall, we observe that the Fit Curve method is very good at prediction.

6.3.2 FPC

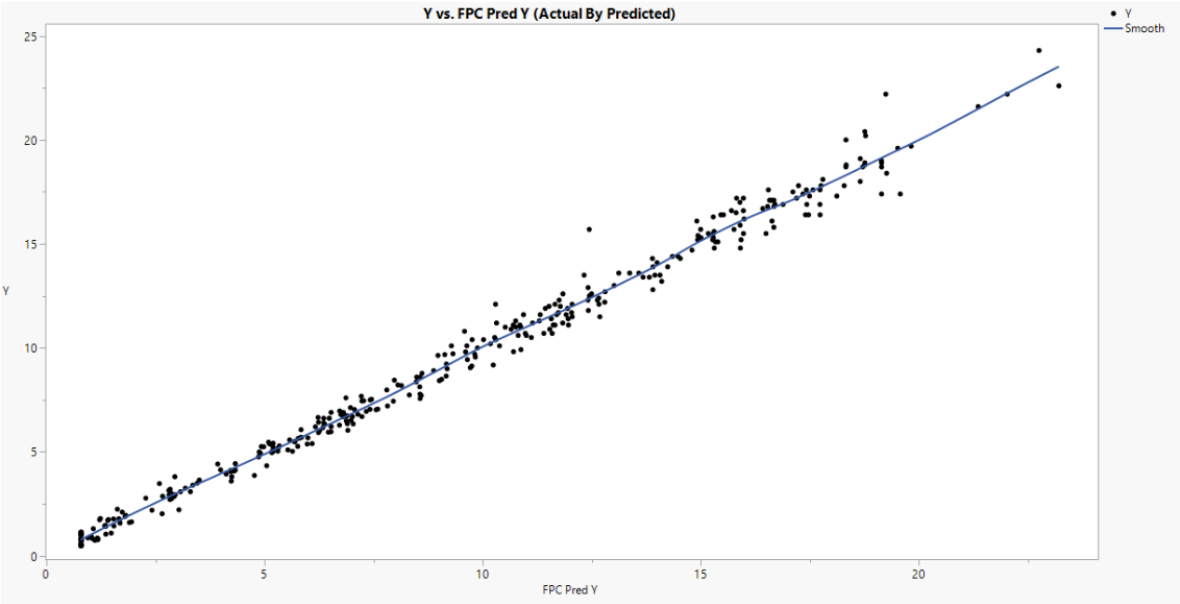


Figure 30: Actual By Predicted Plot for the FPC Method for our DoE 10 Factor Example.

Observing our third example's FPC actual by predicted plot we can see a very similiar result that was produced in our second example. We can see a good looking actual by predicted plot that has less variation compared to the Fit Curve method and a trend that is a little more straight. From our actual by predicted plot in this third example as well in our second example there can be assumptions made that the FPC method performs better at prediction compared to Fit Curve.

6.3.3 Bayesian Hierarchical Two-Stage Mixed-Effects: R Code

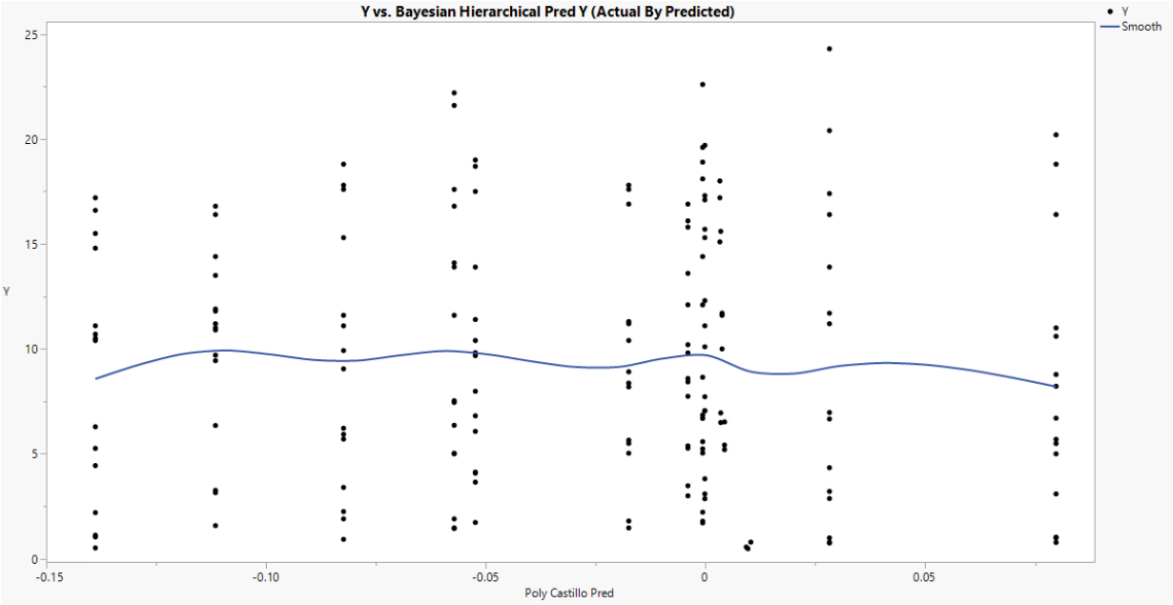


Figure 31: Actual By Predicted Plot for the Bayesian Hierarchical Method for our DoE 10 Factor Example

Lastly, in our third example we observe the Bayesian Hierarchical Two-Stage Mixed-Effects method which produces a plot that has data points all scattered thorughout the plot. Like we saw before, there is not a trend to follow in any of the Bayesian Hierarchical Two-Stage Mixed-Effects method. This further showcases that the Bayesian Hierarchical Two-Stage Mixed-Effects is very poor at prediction.

6.3.4 Method Estimates

Y	FPC Pred Y	Fit Curve Weibull Pred Y	Bayes H. Pred	Fit Curve Diff	FPC Diff	Bayes H. Diff	Fit Curve AVG Diff	FPC AVG Diff	Bayes AVG Diff
0.793	0.790538985	7.59965E-05	0.010291123	0.792924004	0.002461015	0.782708877	0.142926243	-0.0150867	9.419034773
1.31	1.075702348	0.220675436		1.089324564	0.234297652				
2.11	1.738328394	1.189033149		0.920966851	0.371671606				
3.81	2.945000169	2.805855257		1.004144743	0.864999831				
5.25	4.980278613	5.064744429		0.185255571	0.269721387				
7.03	7.558838087	7.657622141		-0.627622141	-0.528838087				
9.18	10.24314337	10.22087656		-1.04087656	-1.06314337				
12.1	12.66630747	12.50846603		-0.40846603	-0.56630747				
14.4	14.47655255	14.31865657	-0.111510744	0.08134343	-0.07655255	14.51151074			
16.6	15.701802	15.58752407	-0.138839085	1.01247593	0.898198	16.73883909			
17.6	16.54980418	16.4393108	-0.082368574	1.1606892	1.05019582	17.68236857			
17.5	17.11109483	16.92959232	-0.052385944	0.57040768	0.38890517	17.55238594			
16.4	17.47078908	17.19001965	0.079742096	-0.79001965	-1.07078908	16.3202579			
17.6	17.41890659	17.30843706	-0.017484355	0.29156294	0.18109341	17.61748436			
16.1	16.63180947	17.36535205	-0.004016979	-1.26535205	-0.53180947	16.10401698			
1.01	0.791141018	0.000176098		1.009823902	0.218858982				
1.74	1.219758413	0.363931905		1.376068095	0.520241587				
2.78	2.27924014	1.839567174		0.940432826	0.50075986				
4.42	3.927367151	4.086349848		0.333650152	0.492632849				
6.4	6.355326393	6.867633759		-0.467633759	0.044673607				
8.65	9.164571193	9.583505899	-0.000664493	-0.933505899	-0.514571193	8.650664493			
11.2	11.83462026	11.7769674	0.028191064	-0.5769674	-0.63462026	11.17180894			
13.9	13.91184063	13.30617641	-0.057197954	0.59382359	-0.01184063	13.95719795			
15.3	15.00165684	14.2059063	-0.000172266	1.0940937	0.29834316	15.30017227			
15.6	15.3063896	14.655703	0.003444766	0.944297	0.2936104	15.59655523			
15.5	15.17133767	14.86078738		0.63921262	0.32866233				

Figure 32: Method Estimates for DoE 10 Factor Example

Observing our table above we can see that our DoE 10 Factor Example provided to be the most difficult example for our three methods to compute predicted values for. However, after observing our table we see that our FPC method produces the most accurate predicted values out of our three methods. This is shown by FPC having the smallest average difference values between actual by predicted values. Once again, we see that our Fit Curve Method also provided accurate predicted values for our DoE 10 factor example while our Bayesian Hierarchical Two-Stage Mixed-Effects provided predicted values that were no where near our actual values for Y. We now move on to compare plots of the three methods used.

6.3.5 Summary

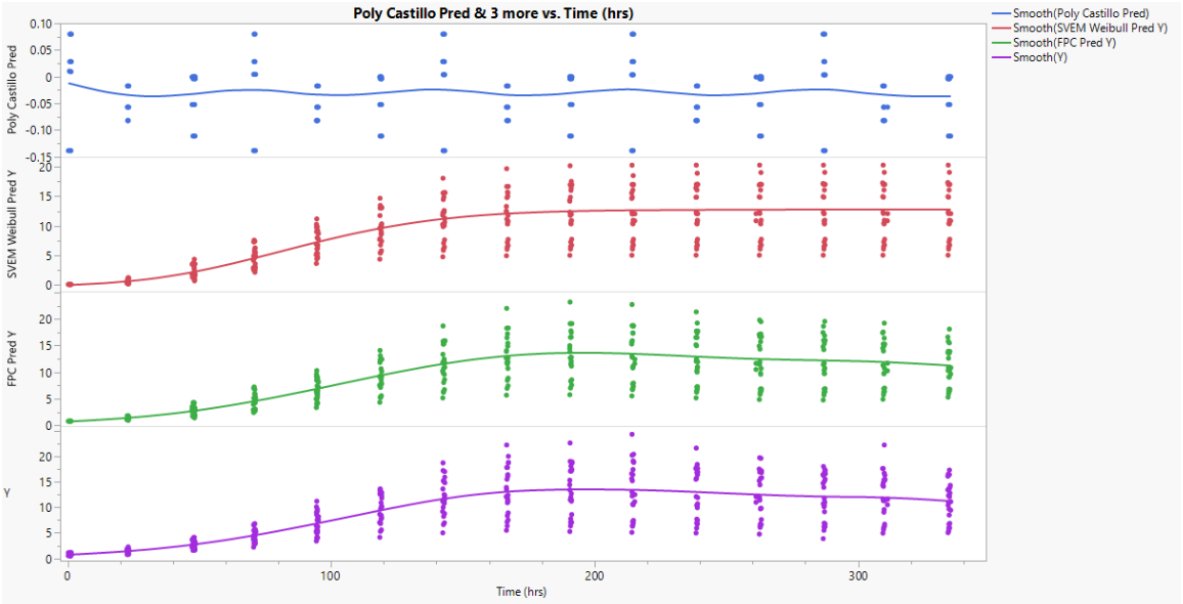


Figure 33: JMP Graph Builder Comparison of DoE 10 Example

Above we have our 3 methods: Fit Curve, FPC, and Bayesian Hierarchical Two-Stage Mixed-Effects Model compared alongside our Actual Y DoE Data. We can see that overall shape for Fit Curve and FPC is very similiar to our Actual Viscosity values. We also observe that the scales at which these predicted Viscosity values are at for Fit Curve and FPC are similiar to the actual values for Y. Now when we observe our Bayesian Hierarchical Two-Stage Mixed-Effects Model we see that not only the shape of our predicted values are off but also the scale that is predicted is no where near where the actual Viscosity values. From this, we can conclude that Fit Curve and FPC are much more effective methods at prediction compared to the Bayesian Hierarchical Two-Stage Mixed-Effects Model. If we observe closesly we can see that the FPC method actually provides us the inisghts to conclude that the FPC is better at performing prediction than Fit Curve. This si displayed through the end tail of FPC having less variation and for following the actual Y values which we can see from the little dip the linear trend takes inn the third to the bottom graph.

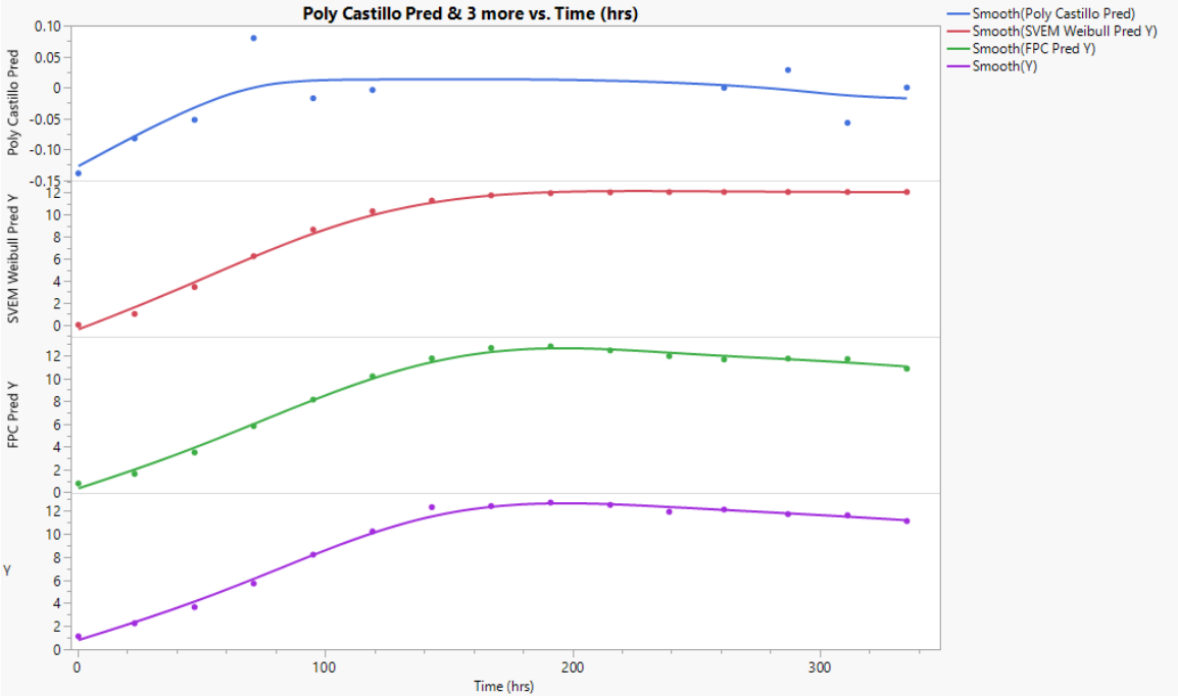


Figure 34: JMP Graph Builder Comparison of DoE 10 Example Run 24

Similar to what I did in our Viscosity and Fly-Ash data, I looked at each Run's curve to see how each method performed prediction wise. Unlike our Viscosity data, I have found that with our Fly-Ash data there is no Run where the Bayesian Hierarchical Two-Stage Mixed-Effects Model produced any prediction curve that had any similar shape to our actual TempC values. The one run that I found that was notable to mention was run 24's plots. As you can see all four plots have a similiar trend in shape and curvature. However upon further pevaluation we can see evidence that the Bayesian Hierarchical Two-Stage Mixed-Effects Model is to suitable for prediction. From our three examples all concluding that this method is very poor at prediction, we also consider that for any further research in investigating how one can improve and optimize design within the model for prediction accuracy it shall not be explored.

7 Conclusions and Further Work

Through this project, it was clear that the JMP's Fit Curve Method and Functional Principal Components proved to be very effective methods in modeling functional data responses as a curves based upon experimental factors. In our first example with the Viscosity data set we saw that the Fit Curve method outperformed FPC, while in our second example with Fly-Ash data and our third example with DoE factor data FPC outperformed JMP's Fit Curve method on accuracy of predicted values against actual values. Now, when modeling our Bayesian Hierarchical Two-Stage Mixed-Effects method we can see that in all three examples the prediction produced had very poor accuracy. This was shown in not only the graphs but was very obvious when observing the predicted values against actual values in our charts in the method estimates sub-sections of our paper. Overall, we can conclude that in our project that JMP's Fit Curve method and FPC provided us the greatest accuracy in predicting values for our responses in each of our three data sets. We can consider that FPC proves to be more accurate in predicting response values then JMP's Fit Curve Method but further research would need to be done in order to conclude this assumption. Finally, we can conclude that the Bayesian Hierarchical Two-Stage Mixed-Effects method should not be implemented as a method for modeling functional data responses as a curves based upon experimental factors. Bayesian Hierarchical Two-Stage Mixed-Effects method provides very poor accuracy and consideration of further research on this method should very cautious due to the inaccuracy of the methods predicted values. When considering further research, there are two

possible avenues someone can explore. The first is with simulation, specifically simulating our functional responses as curves based upon experimental factors. During the beginning of this Master Project we investigated methods of simulation for functional data which we were successful in finding multiple articles. We were also successful in findings articles about modeling functional responses as curves based upon experimental factors. However, when combinging the two subject into one, there was very little research on the simulation of functional responses as curves based upon experimental factors. The biggest problem that we found was not the simulation of responses for our functional data, but having to relate those simulations back to our experimental factors in addition to being able to simulate all of those different combination of experimental factors. The other possible avenue for future research is with modeling response surfaces and not just response curves. So, instead of having a 2 dimensional curve, as we have shown numerous times throughout this paper, modeling response surfaces could create a 3 dimensional image for us to analyze and interpret. Currently there is no research on how to model response surfaces for functional data, so this could potentially be a possible dissertation for an ambitious individual.

8 Appendix

8.1 Appendix A: MATLAB Code

```
% Master Project

designmatrix = readtable("C:\Users\degro\OneDrive\Documents\MasterProject\
    MATLAB\DesignMatrix.txt");
designmatrix.Properties.VariableNames(1) = "A";
designmatrix.Properties.VariableNames(2) = "B";
designmatrix.Properties.VariableNames(3) = "C";
designmatrix.Properties.VariableNames(4) = "D";
designmatrix.Properties.VariableNames(5) = "E";

S = designmatrix

%D = fixedmatrix

%response = readtable("C:\Users\Gary\Documents\Jack Master Project\
    OutputFromBiExp.txt");
%response.Properties.VariableNames(1) = "Viscosity";

%Y = response

model = [
    0 0 0 0 0  % The intercept (no interaction)
    1 0 0 0 0  % Main effects we need first of second group for time
    0 1 0 0 0  % we need second of third group for scale 1
    0 0 1 0 0  % we need third of fourth group for decay 1
    0 0 0 1 0  % we need fourth of fifth group of scale 2
    0 0 0 0 1  % we need fifth of sixth group of decay 2
   ];

Regressormatrix = readtable("C:\Users\degro\OneDrive\Documents\MasterProject\
    MATLAB\RegressorMatrix.txt");
Regressormatrix.Properties.VariableNames(1) = "Time";
Regressormatrix.Properties.VariableNames(2) = "Scale 1";
Regressormatrix.Properties.VariableNames(3) = "Decay 1";
Regressormatrix.Properties.VariableNames(4) = "Scale 2";
Regressormatrix.Properties.VariableNames(5) = "Decay 2";

D = Regressormatrix
```

```
Responsematrix = readtable("C:\Users\degro\OneDrive\Documents\MasterProject\
    MATLAB\ResponseMatrix.txt");
Responsematrix.Properties.VariableNames(1) = "Viscosity";

Y = Responsematrix

r=0;

D = table2array(D); % Convert table to array
Y = table2array(Y); % Convert table to array
S = table2array(S); % Convert table to array


%Bounds for higher asymptote
L=[0 7.912103 0.1056479 21.39241 2.025852]';
U=[0.02 17.355099 0.2119513 348.10141 7.404218]';


   AnalysisType='mixedEffects';
%%Optimal S that achieves the estimated covariance%%%%%%%%%%%%%%%%%%%%%
   switch AnalysisType
         case 'mixedEffects'
         [Snew]=scaleOpt(S,Y,D,model);
          Ssecond=Snew;
           otherwise
    % If no attempt at modeling covariance, simply leave Ssecond=S
      Ssecond=S;
  end;




stdNoise=1;
nSim=500;
nInitialPoints=3;
nSimMCMC=2500;

[N,k]=size(D);
        X=x2fx(D,model);
           BigX = [];
           Ystacked = [];

for i=1:N
    kronResult = kron(X(i,:), S);
    BigX = [BigX; kronResult];
    Ystacked = [Ystacked; repmat(Y(i,:), size(kronResult, 1), 1)];
end

        [N,J]=size(Y);
        [N,q]=size(X);
        [J,p]=size(S);
        DiffLow=1;
        DiffHigh=2;

rng('default');  % Reset the generator to its default settings
rng(12);

[pars]=MCMCMixedEffects(Y,X,S,Ssecond,Ystacked,BigX,nSimMCMC,J,N,p,q);

selected_pars = pars(1:2500, 1);

selected_pars_matrix = cell2mat(selected_pars);
```

```
selected_cols = [12, 18, 24, 30];
selected_values = selected_pars_matrix(:, selected_cols);

selected_pars_table = array2table(selected_values, 'VariableNames', {'Scale1',
    'Decay1', 'Scale2', 'Decay2'});

mean_values = mean(selected_values);

writetable(selected_pars_table, 'MATLABCastilloEstimates.csv');
```

8.2 Appendix B: R Code

```
# Load the required libraries
library(readxl)
library(dplyr)
library(brms)
library(rstan)
library(rstudioapi)

library(readxl)
Viscosity_Stacked_with_Pred_Formula <- read_excel("C:/Users/Gary/Downloads/
    Viscosity Stacked with Pred Formula (2).xlsx")

Viscosity_Table <- read_excel("C:/Users/Gary/Documents/Jack Master Project/
    ParameterTableJack.xlsx")
Viscosity_Table$ViscosityPred <- Viscosity_Stacked_with_Pred_Formula$`
    Viscosity Predictor with Factors`[1:25]

names(Viscosity_Table) <- gsub(" ", "_", names(Viscosity_Table))

poly_formula <- bf(
  ViscosityPred ~ 1 + Scale_1 + Decay_Rate_1 + Scale_2 + Decay_Rate_2 +
    Scale_1:Decay_Rate_1 + Scale_1:Scale_2 + Scale_1:Decay_Rate_2 +
    Decay_Rate_1:Scale_2 + Decay_Rate_1:Decay_Rate_2 +
    Scale_2:Decay_Rate_2 +
    Scale_1:Decay_Rate_1:Scale_2 + Scale_1:Decay_Rate_1:Decay_Rate_2 +
    Scale_1:Scale_2:Decay_Rate_2 +
    Decay_Rate_1:Scale_2:Decay_Rate_2 +
    Scale_1:Decay_Rate_1:Scale_2:Decay_Rate_2 +
    (1 + Scale_1 + Decay_Rate_1 + Scale_2 + Decay_Rate_2 | Run),
  nl = FALSE
)

# Specify priors
prior1 <- prior(normal(0, 100), class = "Intercept")
prior2 <- prior(normal(0, 100), class = "b")
prior3 <- prior(normal(0, 100), class = "sd", group = "Run")

# Run the model
fit_brms <- brm(
  formula = poly_formula,
  data = Viscosity_Table,
  family = gaussian(),
  prior = c(prior1, prior2, prior3),
  control = list(adapt_delta = 0.9999, max_treedepth = 15),
  cores = 4,
  iter = 2500
)
```

```
# Get summary of model
(summary_brms <- summary(fit_brms))

# Display pairs plot
pairs(fit_brms)

# Get summary of model
(summary_brms <- summary(fit_brms))

# For the fixed effects
(fixed_effects <- fixef(fit_brms))

marginal_effects(fit_brms)

# For the random effects
(random_effects <- ranef(fit_brms))


predicted_responses <- fitted(fit_brms)
predicted_responses


# Extract posterior samples
post_samples <- posterior_samples(fit_brms)


# View the posterior samples
print(head(post_samples))

post_samples$`r_Run__Scale1[1,Intercept]`

posterior_means <- colMeans(post_samples)
print(posterior_means)



# Load necessary library
library(ggplot2)


# Define a sequence of time points at which you want to generate predictions
time_values <- seq(from = min(Viscosity_Stacked_with_Pred_Formula$Time),
                   to = max(Viscosity_Stacked_with_Pred_Formula$Time),
                   length.out = 100)

# Generate predictions
predictions <- posterior_means["b_Scale1_Intercept"] * exp(-posterior_means["
    b_Decay1_Intercept"] * time_values) +
  posterior_means["b_Scale2_Intercept"] * exp(-posterior_means["
      b_Decay2_Intercept"] * time_values)

# Plot the predictions
plot(time_values, predictions, type = "l",
     xlab = "Time", ylab = "Predicted Viscosity",
     main = "Predicted Viscosity as a function of Time")

# Get summary of model
(summary_brms <- summary(fit_brms))

# For the fixed effects
(fixed_effects <- fixef(fit_brms))
```

```
# For the random effects
(random_effects <- ranef(fit_brms))

# Extract posterior samples
(post_samples <- posterior_samples(fit_brms))

post_samples$`r_Run__Scale1[1,Intercept]`


Doe_10_factor_Parameter_Table_Weibull_Model <- read_excel("C:/Users/Gary/
    Downloads/Doe 10 factor Parameter Table Weibull Model.xlsx")

Doe_10_factor_Parameter_Table_Weibull_Model

# Create the Time sequence
Time <- seq(from = 0.7, to = 335.4, length.out = 24)

# Add Time column to the dataframe
Doe_10_factor_Parameter_Table_Weibull_Model$Time <- Time

# Display the updated dataframe
print(Doe_10_factor_Parameter_Table_Weibull_Model)

# Rename columns in the data
names(Doe_10_factor_Parameter_Table_Weibull_Model) <- gsub("\\.", "", names(
    Doe_10_factor_Parameter_Table_Weibull_Model))
names(Doe_10_factor_Parameter_Table_Weibull_Model) <- gsub("_", "", names(
    Doe_10_factor_Parameter_Table_Weibull_Model))
names(Doe_10_factor_Parameter_Table_Weibull_Model) <- gsub(" ", "", names(
    Doe_10_factor_Parameter_Table_Weibull_Model))


# Display the data
colnames(Doe_10_factor_Parameter_Table_Weibull_Model)

# Create the formula
poly_formula <- bf(
  Y ~ 0 + PredFormulaAsymptote + PredFormulaInflectionPoint +
      PredFormulaGrowthRate +
    PredFormulaAsymptote:PredFormulaInflectionPoint + PredFormulaAsymptote:
        PredFormulaGrowthRate +
    PredFormulaInflectionPoint:PredFormulaGrowthRate +
    PredFormulaAsymptote:PredFormulaInflectionPoint:PredFormulaGrowthRate +
    (0 + PredFormulaAsymptote + PredFormulaInflectionPoint +
        PredFormulaGrowthRate | p | RunID),
  nl = FALSE
)

# Specify priors
prior1 <- prior(normal(0, 100), class = "b")
prior2 <- prior(normal(0, 100), class = "sd", group = "RunID")

# Run the model
fit_brms <- brm(
  formula = poly_formula,
  data = Doe_10_factor_Parameter_Table_Weibull_Model,
  family = gaussian(),
  prior = c(prior1, prior2),
  control = list(adapt_delta = 0.9999, max_treedepth = 17),
  cores = 4,
  iter = 4000
```

```
)

# Display pairs plot
pairs(fit_brms)

# Get summary of model
summary_brms <- summary(fit_brms)
print(summary_brms)

# For the fixed effects
fixed_effects <- fixef(fit_brms)
print(fixed_effects)

# For the random effects
random_effects <- ranef(fit_brms)
print(random_effects)

# Extract posterior samples
post_samples <- posterior_samples(fit_brms)

# View the posterior samples
print(head(post_samples))

posterior_means <- colMeans(post_samples)
print(posterior_means)


# Read the data
Fly_Ash_Parameter_Table_Gaussian_Peak_Model <- read_excel("C:/Users/Gary/
    Downloads/Fly Ash Parameter Table Gaussian Peak Model.xlsx")

# Create the Time sequence
Time <- seq(from = 0.1, to = 23.9, length.out = 15)

# Add Time column to the dataframe
Fly_Ash_Parameter_Table_Gaussian_Peak_Model$Time <- Time

# Display the updated dataframe
print(Fly_Ash_Parameter_Table_Gaussian_Peak_Model)

# Rename columns in the data
names(Fly_Ash_Parameter_Table_Gaussian_Peak_Model) <- gsub("\\.", "", names(
    Fly_Ash_Parameter_Table_Gaussian_Peak_Model))
names(Fly_Ash_Parameter_Table_Gaussian_Peak_Model) <- gsub("_", "", names(
    Fly_Ash_Parameter_Table_Gaussian_Peak_Model))
names(Fly_Ash_Parameter_Table_Gaussian_Peak_Model) <- gsub(" ", "", names(
    Fly_Ash_Parameter_Table_Gaussian_Peak_Model))


# Display the data
colnames(Fly_Ash_Parameter_Table_Gaussian_Peak_Model)

# Create the formula
poly_formula <- bf(
  TempC ~ 0 + PeakValue + CriticalPoint + GrowthRate +
    PeakValue:CriticalPoint + PeakValue:GrowthRate +
    CriticalPoint:GrowthRate +
    PeakValue:CriticalPoint:GrowthRate +
    (0 + PeakValue + CriticalPoint + GrowthRate | p | RunID),
  nl = FALSE
)
```

```
# Specify priors
prior1 <- prior(normal(0, 100), class = "b")
prior2 <- prior(normal(0, 100), class = "sd", group = "RunID")

# Run the model
fit_brms <- brm(
  formula = poly_formula,
  data = Fly_Ash_Parameter_Table_Gaussian_Peak_Model,
  family = gaussian(),
  prior = c(prior1, prior2),
  control = list(adapt_delta = 0.9999, max_treedepth = 17),
  cores = 4,
  iter = 4000
)


# Display pairs plot
pairs(fit_brms)

# Get summary of model
summary_brms <- summary(fit_brms)
print(summary_brms)

# For the fixed effects
#fixed_effects <- fixef(fit_brms)
#print(fixed_effects)

# For the random effects
#random_effects <- ranef(fit_brms)
#print(random_effects)

# Extract posterior samples
post_samples <- posterior_samples(fit_brms)

# View the posterior samples
print(head(post_samples))

posterior_means <- colMeans(post_samples)
print(posterior_means)
```

9 References

References

[1] Del Castillo, E., Colosimo, B. M., Alshraideh, H., *Bayesian Modeling and Robust Optimization of Functional Responses affected by Noise Factors*, JQT, 44(2), 2012.

[2] Ullah, Shoriat, Seo, K., *Prediction of Lithium-Ion Battery Capacity by Functional Principal Component of Monitoring Data*, Appl. Sci., 12, 4296, 2022. https://doi.org/10.3390/app12094296

[3] Lemkus, Trent, Gotwalt, Christopher, Ramsey, Philip, Weese, Maria L., *Self-validated ensemble models for design of experiments*, Chemometrics and Intelligent Laboratory Systems, 219, 104439, 2021. https://doi.org/10.1016/j.chemolab.2021.104439

[4] Peterson, J. J., *A Posterior Predictive Approach to Multiple Response Surface Optimization*, Journal of Quality Technology, 36(2), 139–153, 2004.

[5] Miro, G., Del Castillo, E., Peterson, J. J., *A Bayesian Approach for Multiple Response Surface Optimization in the Presence of Noise Variables*, Journal of Applied Statistics, 31(3), pp. 251–270, 2004.

[6] Govaerts, B., Noel, J., *Analyzing the Results of a Designed Experiment when the Response Is a Curve: Methodology and Application in Metal Injection Moulding*, Quality and Reliability Engineering International, 21, pp. 509–520, 2005.