Ian Johnston
ID: 1005604990

Finite element analysis and machine learning to predict mechanical properties of polydisperse glass-fiber composites.

Introduction

Glass-fiber reinforced composite materials continue to attract interest from both industrial and academic investigators for applications in aerospace, automotive, construction, and recreation equipment [1]. To avoid costly experimental work, finite element analysis (FEA) can be used to simulate the material properties of a glass-fiber composite [2]. Recently, machine learning (ML) approaches have been suggested as a less computationally intensive approach for rapid predictions [3]. One recent publication prepared a dataset of 1800 simulated 2-dimensional glass-fiber composites, found material properties with FEA, and trained a Gradient Boosted Tree Regressor (GBT) on the responses. To engineer the features for the GBT, the authors described the microstructure using a 2-point correlation function, and then used principal component analysis (PCA) to reduce the dimension of the data set to 10-50 features. The authors of [3] limited their investigation to fibers within each microstructure to be monodisperse diameter, a significant deviation from actual experimental conditions [4].

We sought to extend [3] to better mimic experimental realities by generating 2D microstructures with fiber diameters that are unimodally normally distributed, and bimodal with two superimposed normal distributions. We used a similar workflow to that of the previous authors, except for generating FEA results. Lacking expertise in FEA and access to a fully featured software package, we instead simulated a range of response functions to demonstrate the robustness of our GBT algorithm.

Computational Procedure

All computations were completed on a system with Intel i7-8700 CPU with 6 cores at a speed of 3.2GHz and 32 GB of RAM. Full code to replicate the results is available in the supporting information.

Ian Johnston
ID: 1005604990

Generation of microstructures

To more closely mimic experimental conditions, we departed from [3] by expanding the diversity of microstructures considered. More specifically, sets of 2D microstructures with diameters that are 1) monodisperse, 2) unimodal, normally disturbed polydisperse, and 3) bimodal polydisperse with two overlapping normal distributions. In [3], fiber fill ratios (ϕ) of 10-60% were considered; we considered fiber fill ratios up to 50%. For each fill case, 400 microstructures were generated, resulting 2000 structures with a total computational time of 4.7 seconds.

The procedure for generating the 400 by 400 pixel microstructures departs from the previous work and does not use the algorithm described in [5], rather it uses a simple circle drawing function [6] modified to include a draw circle function from the OpenCV2 package. Briefly, a circle of radius 50 pixels was drawn at a random location on the image. To ensure that a new circle (x_1, y_1, R_1) does not intersect with a previously drawn circle (x_0, y_0, R_0), the equation below can be used as a yes/no check on intersection [7]:

$$(R_0 - R_1)^2 \leq (x_0 - x_1)^2 + (y_0 - y_1)^2 \leq (R_0 + R_1)^2$$

To extend [3], we considered a polydisperse distribution of fiber radii. We first tested a normal distribution with a radius of 50 pixels and a standard deviation of 15 pixels. We then examined a bimodal distribution composed of two super-imposed normal distributions with μ_1 = 30 pixels, μ_2 = 60 pixels, and standard deviations of 5 and 15 pixels respectively. Normal distributions in both extension cases were obtained using `numpy.random.normal` function. For the bimodal case, an additional `numpy.random.choice` function is used to randomly generate a radius value from one of the two distributions (Figure 1).

Ian Johnston
ID: 1005604990

Two-point correlation function

We calculated the fiber-fiber two-point correlation function using the computationally efficient Materials Knowledge System in Python (PyMKS) correlation function. This calculation required 60- 70 milliseconds per microstructure to determine the 2-point correlation function, totaling 2 minutes; significantly faster than described in the original publication. Example results of the 2-point correlation function are seen in Figure 1.

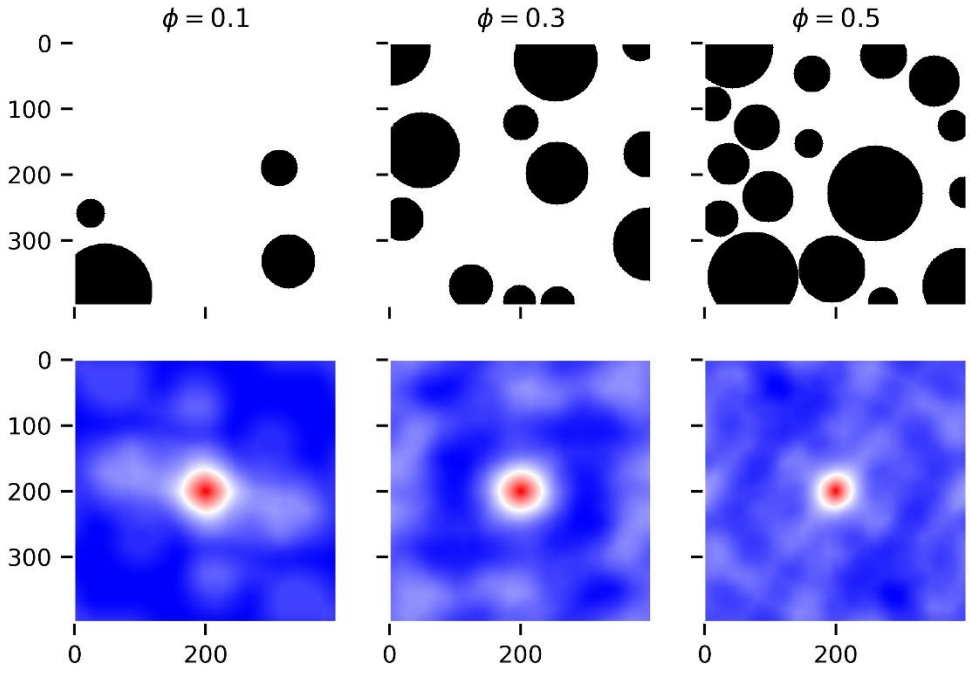

Figure 1: microstructures and corresponding 2-point correlation function plot of fibers with a bimodal diameter distribution. (Top Row) Representative microstructure of various fill percentages (ϕ) at 10, 30, and 50 %. (Bottom row).

Dimensionality reduction

After determination of the 2-point correlation function, the matrix was reshaped from a (400, 400, 2000) matrix to a (160000, 2000) matrix. The transformation was rapid, taking 4.2 seconds to complete on the previously described system. More explicitly, in this format each row of the transformed

matrix contains the data from one microstructure. As the transformed matrix has dimension (160000, 2000) with the number of parameters far in excess of number of microstructures, principal component analysis (PCA) was applied. We selected between 2 and 50 principal components (PCs) to see the affect of including more parameters affects the training of the ML model. The PCA was performed using the `PCA` function of the `sklearn.decomposition` package with the default settings used.

Generation of pseudo-FEA results

We were unable to obtain a fully featured FEA software[1] to determine the mechanical properties of the microstructures. As a substitute, we generated a pseudo-FEA data sets by assuming that % fiber loading strongly influenced the mechanical properties. To add robustness to this assumption, we developed several underlying response functions to capture a range of material responses: linear, linear with additional simulated noise, natural logarithmic, quadratic, and quartic. Inputting fiber fill ratios of 0 to 1, the functions output pseudo-FEA results of 0 to 20 GPa, as seen below in Figure 2.

Table 1: Equations for underlying response functions

Function	Equation
Linear	$y = 20x$
Logarithmic	$y = abs(\log(x))$
Quadratic	$y = -20x(x-2)$
Quartic	$y = -70x(x-1.1)(x+0.2)(x+0.4)^2$

[1] Authors note: I was able to test a few structures with an FEA software package (ABAQUS Student edition); however, the student edition limits the number of nodes in a mesh convergence study to 1000. Presumably limiting the accuracy of the simulations in microstructures with a large number of small fibers. Addition, the lack of expert feedback on how correct the simulations were, suggested that devoting significant time to this portion of the project was infeasible. Personally, I would really enjoy gaining access to the full featured version of ABAQUS, receiving expert feedback, and the ability to redo this portion of the project to get accurate predictions of the mechanical properties.

Ian Johnston
ID: 1005604990

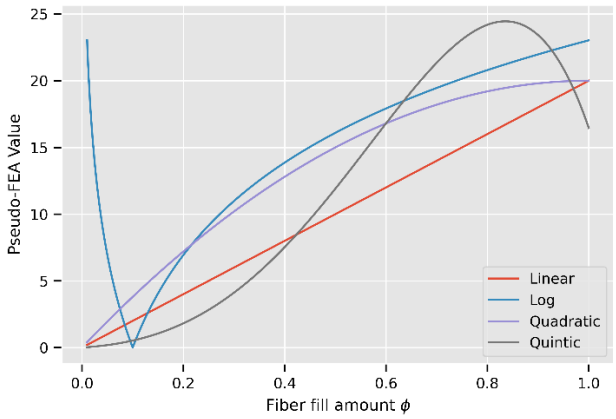

Figure 2: plots of underlying response functions

Machine learning application

We next analyzed the PCs and pseudo-FEA values with a GBR model that is available in the `sklearn.ensemble` python package. Prior to training the model, a training test split in the data was done, using the `train_test_split` function of `sklearn.model_selection`, with a test size of 25% of the data set. For simplicity, we used the optimal parameters described in [3]: the learning rate was 0.1; number of estimators was 800; and max depth of the decision tree was 3. All other parameters were left as function defaults. To assess the accuracy of the model, R^2 of the predicted y values as a function of the actual y values was calculated using the built in `r2_scores` function found in the `sklearn.metrics` package. We then undertook an exhaustive grid search to test the affect that number of PCs and underlying tend in pseudo-FEA data has.

Results and Discussion

Given the impetus for [3] is the reduction of computation time for FEA, the time requirements of any potential ML algorithm are important. We found the PCA portion of the ML workflow to have an outsized influence on the total time. More specifically, PCA takes 10-20 seconds, making the approximately 100 ms training/testing time of GBR relatively insignificant (supporting information). To state the obvious, the

selection of more PCs during PCA requiring more time comes as no surprise and mirrors the results obtained in [3] (Figure 3). The 10-20s computational time of the PCA must still be contrasted with FEA (Figure 3). During preliminary studies on the same computer that was used for this work, FEA using ABAQUS took between 5 to 10 seconds per microstructure. It can be thus calculated that for the full sample set of 2000 microstructures, the computational time would be approximately 7 days of computational time; clearly significant savings were realized.

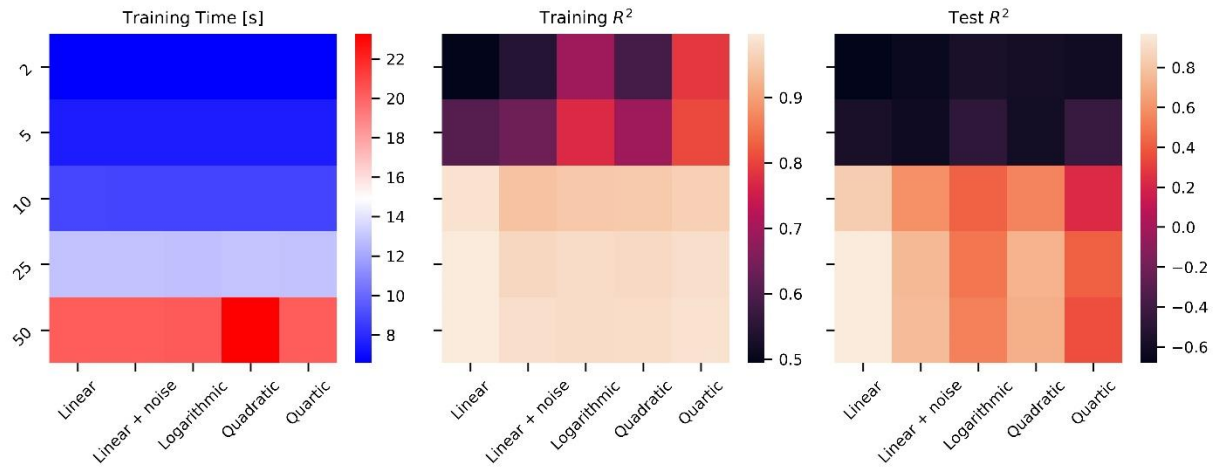

Figure 3: Machine learning results for polydisperse fiber diameter. (Left) Time required for PCA and GBT training as a function of number of PCs and underlying function. PCA calculation dominates the time needed to train the model. (Middle) R^2 score for training of the model as a function of both number of PCs and underlying function. (Right) R^2 score for the test data set.

The featuring engineering in this work was accomplished by varying the number of PCs used prior to fitting the GBR. Foremost, it was found that at least 10 PCs were needed to give a reasonable training error (Figure 1). However, in the case of a bimodal fiber radius distribution, 25 PCs were needed to give a reasonable training error (supporting information). In all three fiber cases, the training error was higher than the test error, suggesting that the GBT model maybe overfitting on the available data. Considering that pseudo-FEA data is a function of the fiber fill percentage and the lower order PCs are more heavily loaded with the fiber fill percentage data, it suggests that adding additional PCs maybe counter productive

after a given point (Figure 4, Figure 5). The use of actual FEA calculations to determine material properties would be helpful in determining how few PCs are needed for accurate predictions.

Using pseudo-FEA data as a replacement for FEA, the ML model successfully handles a range of underling response functions. While the response functions depend on fiber fill percentage only, the ML model was able to accurately predict results using >10 PCs for a polynomial of degree five and a logarithmic function that includes a discontinuity in the form of a cusp (Figure 2,Figure 4, Figure 5). Given that the loading of the largest PCs is predominantly fiber fill percentage [3], it suggests that this first order approximation is reasonable. However, the highest R^2 found in this work is 0.96, which less than the 0.995 found in [3]. Improved accuracy maybe realized by using an underling function that incorporates more subtle features like clustering and inter-fiber distance (features that primarily load higher order PCs). The logical follow up of this work is to include FEA data that has been cross-validated with real world experimental data.

Three different fiber diameter distributions were tested in this study. Given that the pseudo-FEA results only take into account fiber fill percentage and that the algorithm adds fibers, regardless of size, until a predetermined fill ratio is achieved, the pseudo-FEA results should be reasonably similar for all three groups of fibers. The biggest difference between the three groups would be in the higher order PCs, which a loaded with parameters such as inter-fiber distance and cluster. This suggests that the ML model overcomes these extraneous features in the case of unimodal polydisperse and bimodal polydisperse fibers. Ideally, this study would be extended to incorporate actual FEA data, which may begin to show differences in goodness of fit when the fiber populations deviate from monodisperse.

Ian Johnston
ID: 1005604990

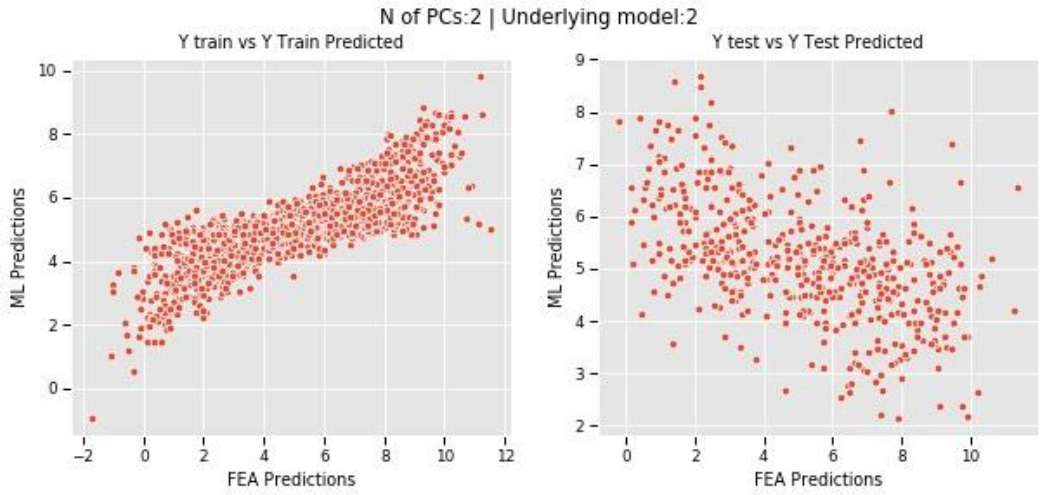

Figure 4: ML predictions vs (Pseudo-) FEA Predictions for 2 PCs with an underlying linear model with noise on a monodisperse fiber diameter population

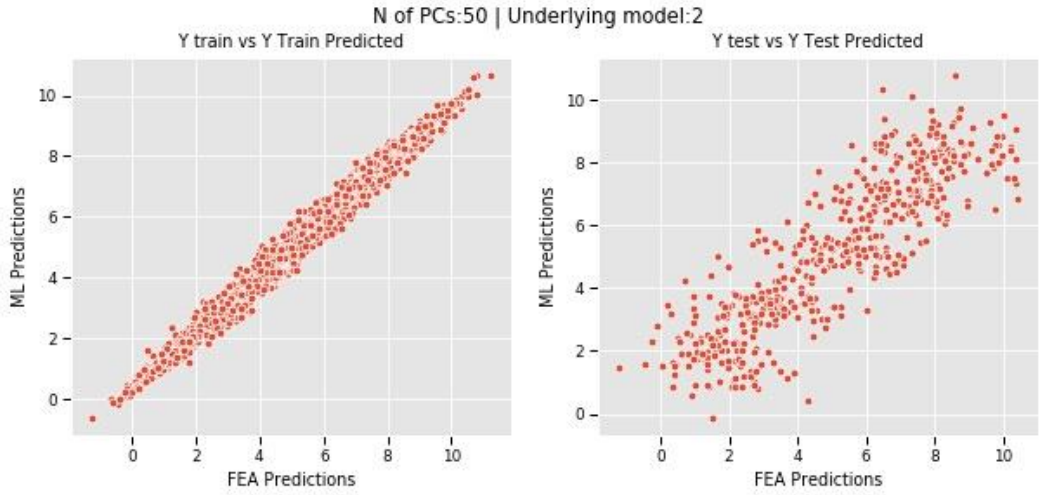

Figure 5: ML predictions vs (Pseudo-) FEA Predictions for 50 PCs with an underlying linear model with noise on a monodisperse fiber diameter population

Conclusions and Future Work

This work extended previous efforts to use machine learning to alleviate the computation burden of determining mechanical properties of glass-fiber composites by introducing fibers that depart from monodispersity. Several response functions were generated to simulate a range of material properties that might be encountered. It was found that a PCA-GBR algorithm was able to correctly model all of the underlying response functions given a sufficient number of PCs were provided to the

GBR. Extensions of this work including thorough FEA data and supplementing simulation data with real world experimental results.

References

[1] A. N. Dickson, J. N. Barry, K. A. McDonnell, and D. P. Dowling, "Fabrication of continuous carbon, glass and Kevlar fibre reinforced polymer composites using additive manufacturing," *Addit. Manuf.*, vol. 16, pp. 146–152, Aug. 2017.

[2] A. Pegoretti, L. Fambri, G. Zappini, and M. Bianchetti, "Finite element analysis of a glass fibre reinforced composite endodontic post," 2002.

[3] M. V. Pathan *et al.*, "Predictions of the mechanical properties of unidirectional fibre composites by supervised machine learning," *Sci. Rep.*, vol. 9, no. 1, Dec. 2019.

[4] F. Ulrych, M. Sova, and B. Tureic, "Empirical Relations of the Mechanical Properties of Polyamide 6 Reinforced With Short Glass Fibers."

[5] M. V. Pathan, V. L. Tagarielli, S. Patsias, and P. M. Baiz-Villafranca, "A new algorithm to generate representative volume elements of composites with cylindrical or spherical fillers," *Compos. Part B Eng.*, vol. 110, pp. 267–278, Feb. 2017.

[6] SirAleXbox and bgfvdu3w, "Python Pygame randomly draw non overlapping circles," *Stack Overflow*, 2017. [Online]. Available: https://stackoverflow.com/questions/46702987/python-pygame-randomly-draw-non-overlapping-circles?noredirect=1&lq=1. [Accessed: 20-Jul-2020].

[7] Dasblinkenlight, "How do I detect intersections between a circle and any other circle in the same plane?," *Stack Overflow2*, 2011. [Online]. Available: https://stackoverflow.com/questions/8367512/how-do-i-detect-intersections-between-a-circle-and-any-other-circle-in-the-same/8367547#8367547. [Accessed: 20-Jul-2020].