# Using ML models for better Dimension precision in FDM printing components

Abstract:

Additive manufacturing (AM), also known as three-dimensional printing, is gaining increasing attention from academia and industry due to the unique advantages it has in comparison with traditional subtractive manufacturing. However, AM processing parameters are difficult to tune, since they can exert a huge impact on the printed microstructure and on the performance of the subsequent products. It is a difficult task to build a process–structure–property–performance (PSPP) relationship for AM using traditional numerical and analytical models. Today, the machine learning (ML) method has been demonstrated to be a valid way to perform complex pattern recognition and regression analysis without an explicit need to construct and solve the underlying physical models. Among ML algorithms, the neural network (NN) is the most widely used model due to the large dataset that is currently available, strong computational power, and sophisticated algorithm architecture. This paper overviews the progress of applying the NN algorithm to several aspects of the AM whole chain, including model design, in situ monitoring, and quality evaluation. Current challenges in applying NNs to AM and potential solutions for these problems are then outlined. Finally, future trends are proposed in order to provide an overall discussion of this interdisciplinary area.

Introduction:

In this Experiment our goal was to create a Genetic Algorithm based model which would give us the best optimal values of the selected factors for minimum variation in Length, Width, and Height of the sample FDM piece, significant process parameters (layer thickness, build orientation, infll density and number of contours) are optimized for enhancing the magnitude/dimensional preciseness of fused deposition modelling (FDM) devise units. An attempt has been made to build up a mathematical model in order to set up an indirect correlation between various FDM process parameters and magnitude preciseness. Sequentially to verify the different developed models and the optimum process parameters setting validation tests were also performed.

Methodology:

Using **Modelling and parametric optimization of FDM 3D printing process using hybrid techniques for enhancing dimensional preciseness** paper by Sandeep Deswal, Rajan Narang, Deepak Chhabra

As a reference the initial dimension of 30 sample are taken and the error in dimension measured by hand are taken as an input for training the NN MATLAB model and Using the Design Expert software the Mathematical model and the design matrix is created.

Relation between and effect of different process parameters are studied and the equation for

1. Percentage variation in length
2. Percentage variation in width
3. Percentage variation in height

These Equation are then used as fitness function in the Genetic Algorithm model made with the use of pyGAD library to predict the possible optimal values for selected factors.

Experiment:

From above discussed literature, it is found that the magnitude preciseness of FDM fabricated parts depends upon various process parameters. Four parameters layer thickness, build orientation, number of contours and infill density selected with their range and levels are shown below. The low and high value range of the selected parameters is set in terms of alpha according to the FDM machine specifications and other parameters are kept fixed. Central composite design with five levels for each process parameter was used to enhance the magnitude preciseness of FDM devise parts. Each process parameter is defined like so:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Sr.no. | Process Parameters | Notation | Unit | Lower limit | Upper limit |
| 1 | Layer Thickness | A | mm | 0.12 | 0.4 |
| 2 | Build Orientation | B | Degree | 0 | 90 |
| 3 | Infill Density | C | % | 0 | 100 |
| 4 | No of contours | D | No. | 2 | 10 |

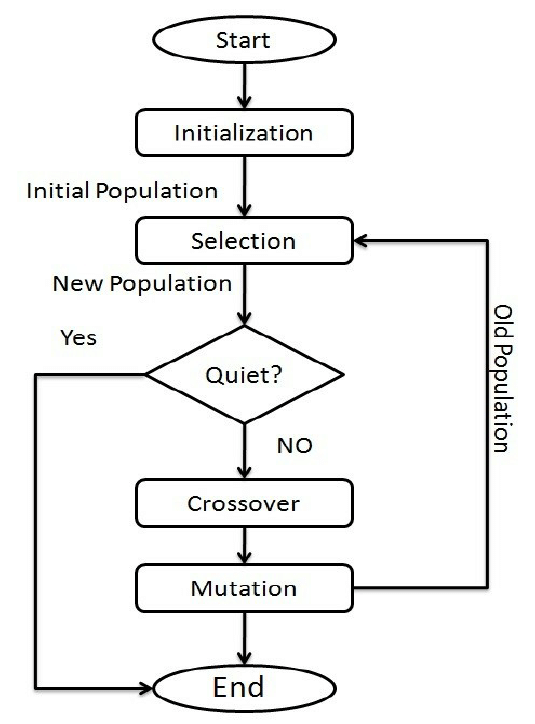
Genetic Algorithm:

Nature has always been a great source of inspiration to all mankind. Genetic Algorithms (GAs) are search based algorithms based on the concepts of natural selection and genetics. GAs are a subset of a much larger branch of computation known as **Evolutionary Computation**.

GAs were developed by John Holland and his students and colleagues at the University of Michigan, most notably David E. Goldberg and has since been tried on various optimization problems with a high degree of success.

In GAs, we have a **pool or a population of possible solutions** to the given problem. These solutions then undergo recombination and mutation (like in natural genetics), producing new children, and the process is repeated over various generations. Each individual (or candidate solution) is assigned a fitness value (based on its objective function value) and the fitter individuals are given a higher chance to mate and yield more “fitter” individuals. This is in line with the Darwinian Theory of “Survival of the Fittest”.

In this way we keep “evolving” better individuals or solutions over generations, till we reach a stopping criterion.



Genetic Algorithms are sufficiently randomized in nature, but they perform much better than random local search (in which we just try various random solutions, keeping track of the best so far), as they exploit historical information as well.

### **Advantages of GAs**

GAs have various advantages which have made them immensely popular. These include −

* Does not require any derivative information (which may not be available for many real-world problems).
* Is faster and more efficient as compared to the traditional methods.
* Has very good parallel capabilities.
* Optimizes both continuous and discrete functions and also multi-objective problems.
* Provides a list of “good” solutions and not just a single solution.
* Always gets an answer to the problem, which gets better over the time.
* Useful when the search space is very large and there are a large number of parameters involved.

### **Limitations of GAs**

Like any technique, GAs also suffer from a few limitations. These include −

* GAs are not suited for all problems, especially problems which are simple and for which derivative information is available.
* Fitness value is calculated repeatedly which might be computationally expensive for some problems.
* Being stochastic, there are no guarantees on the optimality or the quality of the solution.
* If not implemented properly, the GA may not converge to the optimal solution.

## **GA – Motivation**

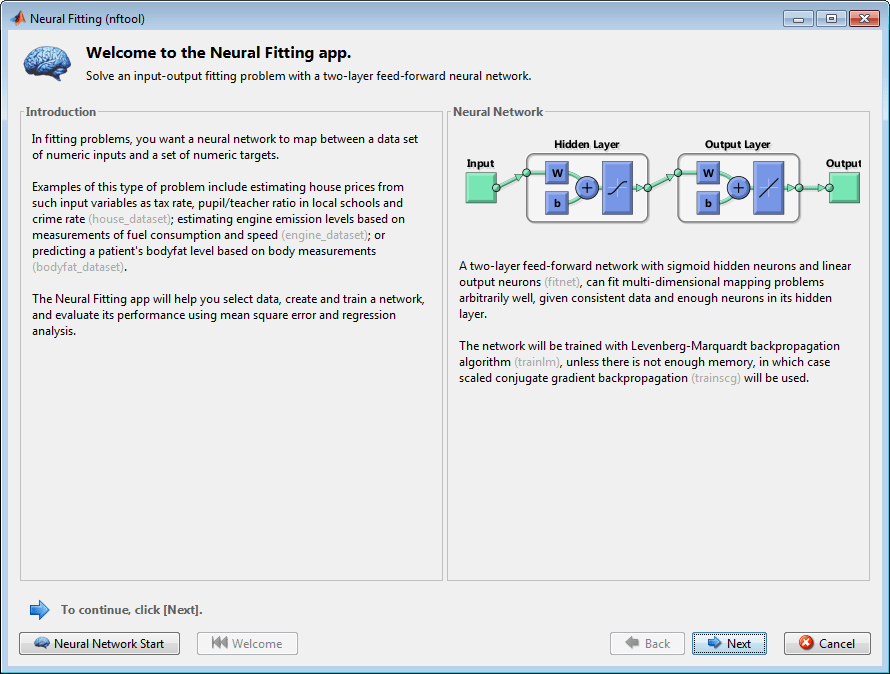
Genetic Algorithms have the ability to deliver a “good-enough” solution “fast-enough”. This makes genetic algorithms attractive for use in solving optimization problems. The reasons why GAs are needed are as follows −

### **Solving Difficult Problems**

In computer science, there is a large set of problems, which are **NP-Hard**. What this essentially means is that, even the most powerful computing systems take a very long time (even years!) to solve that problem. In such a scenario, GAs proves to be an efficient tool to provide **usable near-optimal solutions** in a short amount of time.

NN model Development:

The ANN technique is collected work of data which is used for creating the network, configuring the network, initializing the weights and biases, network training, validation of network and investigate the data. Artificial neural network systems, being great calculations for information investigation, have given wide chances to making utilization of the information contained in mechanical databases. The sigmoid 10 hidden neurons with two level feed-forward network fitted the multidimensional mapping problem. The ANN mechanism to generate the best ft model has used inputs, outputs and neurons three-layer model. The input and output data were investigated from the RSM design matrix. Every input has four variables and six replications. The various process parameters such as input, output, net and fitness obtained from ANN model were used in GA to find out the optimum process parameters.



Design Matrix:

The following design matrix is developed with taking base value from the given paper and using design expert and response surface methodology the variation in dimensions for each run is predicted followed with NN predictions.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Sr  .No | Run no. | Thickness | Orientation | Infill density | Counters | Length Variation | Width  Variation | Thickness  Variation |
| 1 | 25 | 0.33 | 22.5 | 25 | 8 | 1.7 | 1.6 | 1.1 |
| 2 | 23 | 0.19 | 22.5 | 25 | 8 | 0.5 | 0.47 | 2.4 |
| 3 | 15 | 0.33 | 22.5 | 75 | 8 | 1.4 | 1.93 | 1.5 |
| 4 | 29 | 0.19 | 67.5 | 75 | 4 | 1.4 | 0.8 | 1 |
| 5 | 16 | 0.26 | 45 | 0 | 6 | 0.3 | 0.13 | 1.2 |
| 6 | 24 | 0.26 | 0 | 50 | 6 | 2 | 1.8 | 1.7 |
| 7 | 19 | 0.26 | 45 | 50 | 6 | 1.3 | 0.87 | 1.6 |
| 8 | 5 | 0.26 | 45 | 50 | 6 | 1.2 | 0.67 | 1.6 |
| 9 | 13 | 0.19 | 22.5 | 75 | 4 | 0.5 | 1.07 | 2 |
| 10 | 11 | 0.33 | 67.5 | 25 | 4 | 0.7 | 0.47 | 1.5 |
| 11 | 17 | 0.26 | 45 | 50 | 6 | 1.1 | 0.67 | 1.6 |
| 12 | 27 | 0.33 | 22.5 | 25 | 4 | 1.5 | 0.93 | 1 |
| 13 | 8 | 0.19 | 67.5 | 25 | 8 | 1.7 | 0.07 | 0.9 |
| 14 | 1 | 0.19 | 22.5 | 25 | 4 | 0.5 | 0.47 | 2.4 |
| 15 | 30 | 0.33 | 67.5 | 25 | 8 | 1 | 0.87 | 1.7 |
| 16 | 21 | 0.33 | 67.5 | 75 | 8 | 2 | 1.07 | 2.5 |
| 17 | 2 | 0.4 | 45 | 50 | 6 | 1.3 | 1.13 | 1.5 |
| 18 | 10 | 0.33 | 67.5 | 75 | 4 | 1.8 | 0.73 | 2 |
| 19 | 26 | 0.26 | 45 | 50 | 6 | 1.4 | 0.87 | 1.5 |
| 20 | 20 | 0.19 | 67.5 | 25 | 4 | 1 | 0.27 | 1.1 |
| 21 | 14 | 0.26 | 45 | 100 | 6 | 0.7 | 0.67 | 1.7 |
| 22 | 9 | 0.26 | 45 | 50 | 2 | 1.2 | 0.8 | 1.4 |
| 23 | 6 | 0.26 | 45 | 50 | 6 | 1.3 | 0.67 | 1.6 |
| 24 | 28 | 0.12 | 45 | 50 | 6 | 0.1 | 0.2 | 1.6 |
| 25 | 3 | 0.26 | 45 | 50 | 10 | 1.6 | 1.07 | 1.9 |
| 26 | 4 | 0.26 | 45 | 50 | 6 | 1 | 0.8 | 1.7 |
| 27 | 7 | 0.19 | 67.5 | 75 | 8 | 1.9 | 0.27 | 1 |
| 28 | 18 | 0.33 | 22.5 | 75 | 4 | 2.2 | 1.33 | 1.1 |
| 29 | 12 | 0.19 | 22.5 | 75 | 8 | 0.3 | 1 | 2.4 |
| 30 | 22 | 0.25 | 90 | 50 | 6 | 3.1 | 0.87 | 1.4 |

Table Continued….

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Sr  .No | Run no. | RSM Length Variation | RSM Width  Variation | RSM Thickness Variation | NN  Length Variation | NN Width Variation | NN Thickness Variation |
| 1 | 25 | 1.49962 | 1.60357 | 1.099407 | 1.96 | 1.061 | 1.1 |
| 2 | 23 | 0.634195 | 0.505255 | 2.34864 | 1.1024 | 0.8708 | 2.4 |
| 3 | 15 | 1.57045 | 1.88149 | 1.56907 | 2.5032 | 1.0379 | 1.3201 |
| 4 | 29 | 1.50856 | 0.765746 | 0.969215 | 1.3021 | 0.6398 | 1.4053 |
| 5 | 16 | 0.47325 | 0.054616 | 1.39132 | 0.6849 | 0.5818 | 1.0396 |
| 6 | 24 | 2.09033 | 1.8494 | 1.8014 | 1.862 | 1.6166 | 1.7 |
| 7 | 19 | 1.27463 | 0.758289 | 1.56182 | 1.183 | 0.8366 | 1.575 |
| 8 | 5 | 1.21656 | 0.758289 | 1.59132 | 1.183 | 0.8366 | 1.575 |
| 9 | 13 | 0.400861 | 1.07359 | 1.99864 | 0.4528 | 1.6761 | 1.9702 |
| 10 | 11 | 0.719372 | 0.484944 | 1.57834 | 1.2278 | 0.607 | 1.5 |
| 11 | 17 | 1.21655 | 0.66549 | 1.59132 | 1.183 | 0.8366 | 1.575 |
| 12 | 27 | 1.62045 | 0.891489 | 0.869074 | 1.0849 | 1.2552 | 1 |
| 13 | 8 | 1.74189 | 0.067413 | 0.919215 | 1.2895 | 0.3243 | 0.9 |
| 14 | 1 | 0.480028 | 0.469951 | 2.37364 | 0.1227 | 0.7616 | 2.4 |
| 15 | 30 | 1.22354 | 0.899528 | 1.70334 | 1.1751 | 0.818 | 1.7327 |
| 16 | 21 | 1.91937 | 1.00994 | 2.47834 | 1.7138 | 0.7664 | 2.5 |
| 17 | 2 | 1.25756 | 1.19413 | 1.5061 | 1.6187 | 1.1395 | 1.3602 |
| 18 | 10 | 1.79021 | 0.727861 | 2.05334 | 1.7531 | 0.7295 | 2 |
| 19 | 26 | 1.21656 | 0.775478 | 1.579132 | 1.183 | 0.8366 | 1.575 |
| 20 | 20 | 0.962724 | 0.355329 | 1.04421 | 1.1793 | 0.4942 | 1.1 |
| 21 | 14 | 0.792044 | 0.74295 | 1.79132 | 0.5392 | 0.9631 | 1.7 |
| 22 | 9 | 1.3059 | 0.78795 | 1.39132 | 1.2529 | 0.9013 | 1.4 |
| 23 | 6 | 1.21656 | 0.672589 | 1.59132 | 1.183 | 0.8366 | 1.575 |
| 24 | 28 | 0.110488 | 0.133699 | 1.67654 | 0.507 | 0.3591 | 1.6 |
| 25 | 3 | 1.52538 | 1.07962 | 1.79132 | 1.5484 | 0.5763 | 1.9 |
| 26 | 4 | 1.09475 | 0.758289 | 1.59132 | 1.183 | 0.8366 | 1.575 |
| 27 | 7 | 1.919272 | 0.345329 | 1.14421 | 1.0871 | 0.3652 | 1.8547 |
| 28 | 18 | 2.06629 | 1.30191 | 1.04407 | 1.3884 | 1.3019 | 1.1 |
| 29 | 12 | 0.180028 | 0.950672 | 2.27364 | 0.3311 | 1.1573 | 2.4 |
| 30 | 22 | 2.9771 | 0.81817 | 1.24171 | 2.0787 | 0.0515 | 1.6733 |

Results:

Analysis was carried out in DOE software from where the mathematical model was created which in turn was used in the pyGAD code to get optimal solution for length width and height.

In present study quadratic and 2FI (two-factor interaction) models were analysed and selected according to three different tests- the sequential model sum of squares, lack-of-ft and the adequacy model. For percentage variation in length and width quadratic model has the maximum value of R2, adjusted R2 and predicted R2 with very fine concord with each other, but for percentage deviation in the thickness 2FI model has the maximum values for all these correlation coefficient as represented in Table 3. So smaller p values and insignificant lack-of-ft for quadratic and 2FI models in comparison to other models gives an admirable clarification among FDM process parameters and magnitude preciseness.

DOE: RSM summary of model:

### Variation in Length:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Source** | **Std. Dev.** | **R²** | **Adjusted R²** | **Predicted R²** | **PRESS** |  |
| Linear | 0.5944 | 0.2920 | 0.1787 | -0.1020 | 13.75 |  |
| 2FI | 0.5322 | 0.5686 | 0.3416 | 0.2567 | 9.27 |  |
| **Quadratic** | **0.1594** | **0.9694** | **0.9409** | **0.8609** | **1.74** | **Suggested** |
| Cubic | 0.1628 | 0.9873 | 0.9384 |  | \* | Aliased |

Variation in width:

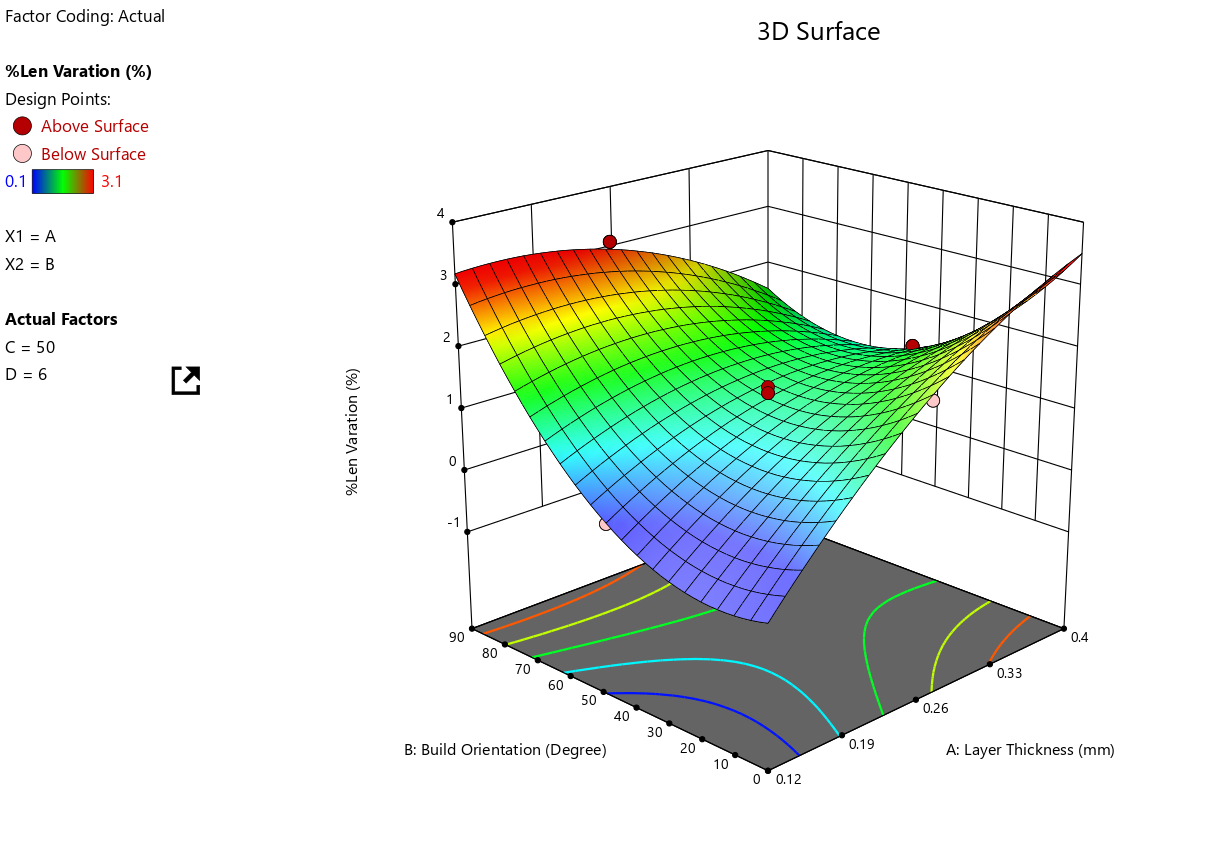
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Source** | **Std. Dev.** | **R²** | **Adjusted R²** | **Predicted R²** | **PRESS** |  |
| Linear | 0.2722 | 0.6857 | 0.6354 | 0.5151 | 2.86 |  |
| 2FI | 0.2411 | 0.8126 | 0.7140 | 0.6891 | 1.83 |  |
| **Quadratic** | **0.0828** | **0.9826** | **0.9663** | **0.9365** | **0.3743** | **Suggested** |
| Cubic | 0.0954 | 0.9907 | 0.9552 |  | \* | Aliased |

Variation in Thickness:

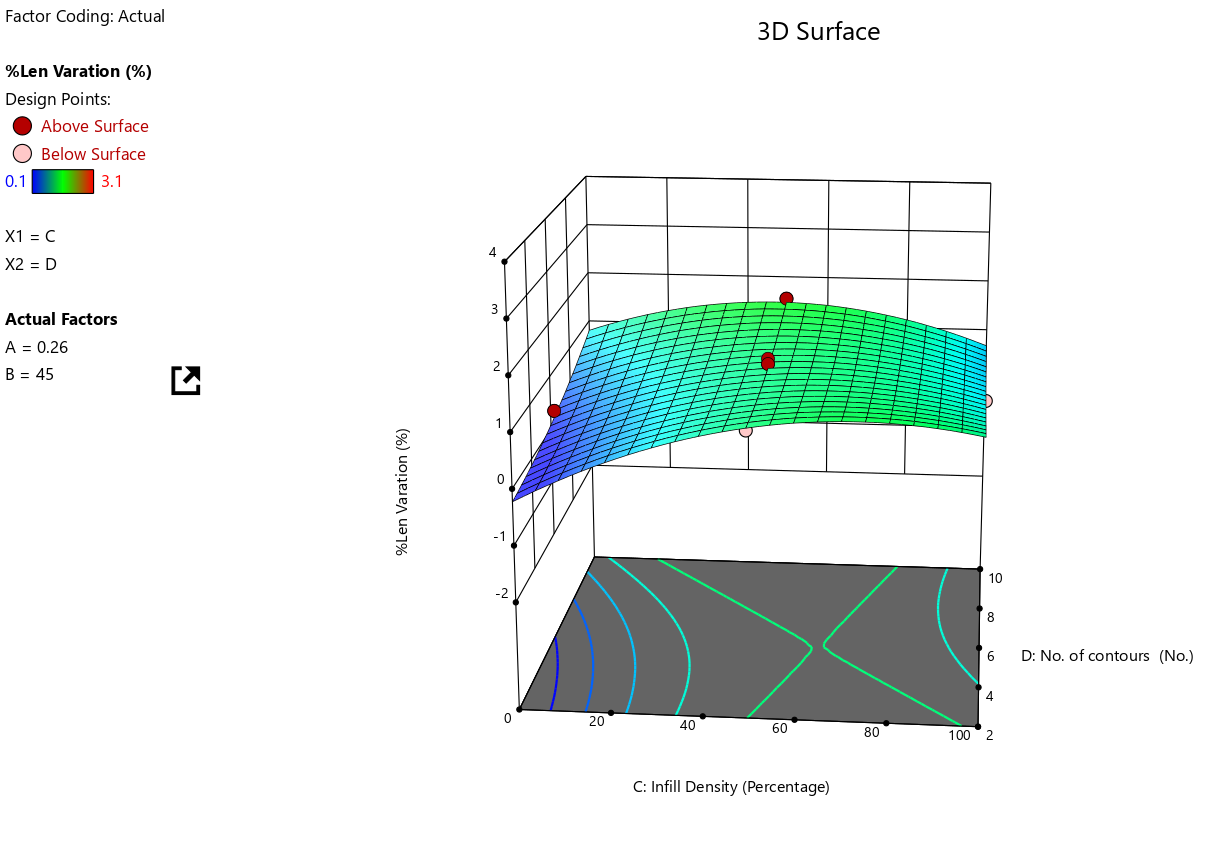
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Source** | **Std. Dev.** | **R²** | **Adjusted R²** | **Predicted R²** | **PRESS** |  |
| Linear | 0.4439 | 0.1470 | 0.0105 | -0.3312 | 7.69 |  |
| **2FI** | **0.1020** | **0.9658** | **0.9478** | **0.8942** | **0.6108** | **Suggested** |
| Quadratic | 0.0999 | 0.9741 | 0.9499 | 0.8659 | 0.7743 |  |
| Cubic | 0.0612 | 0.9961 | 0.9812 |  | \* | Aliased |

Length Varation Graphs:

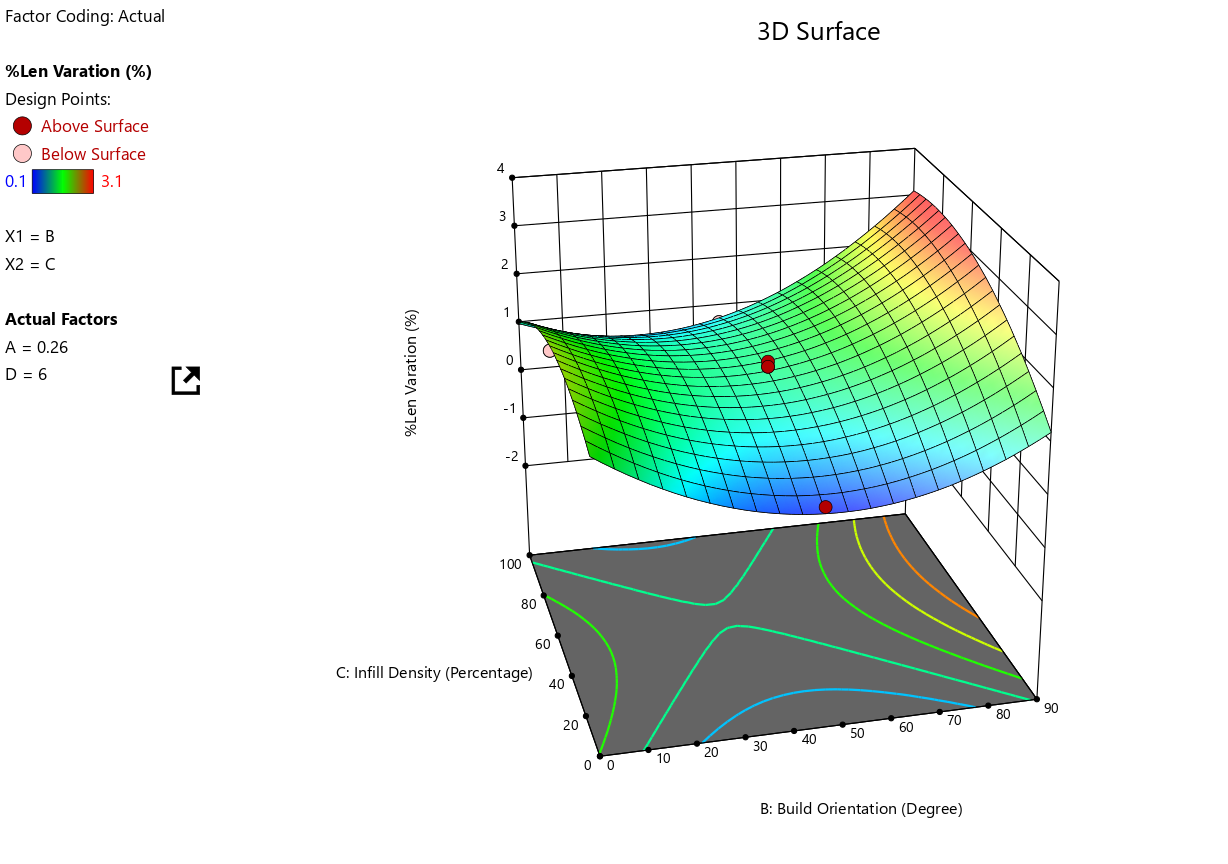
1. Layer Thickness vs Build Orientation



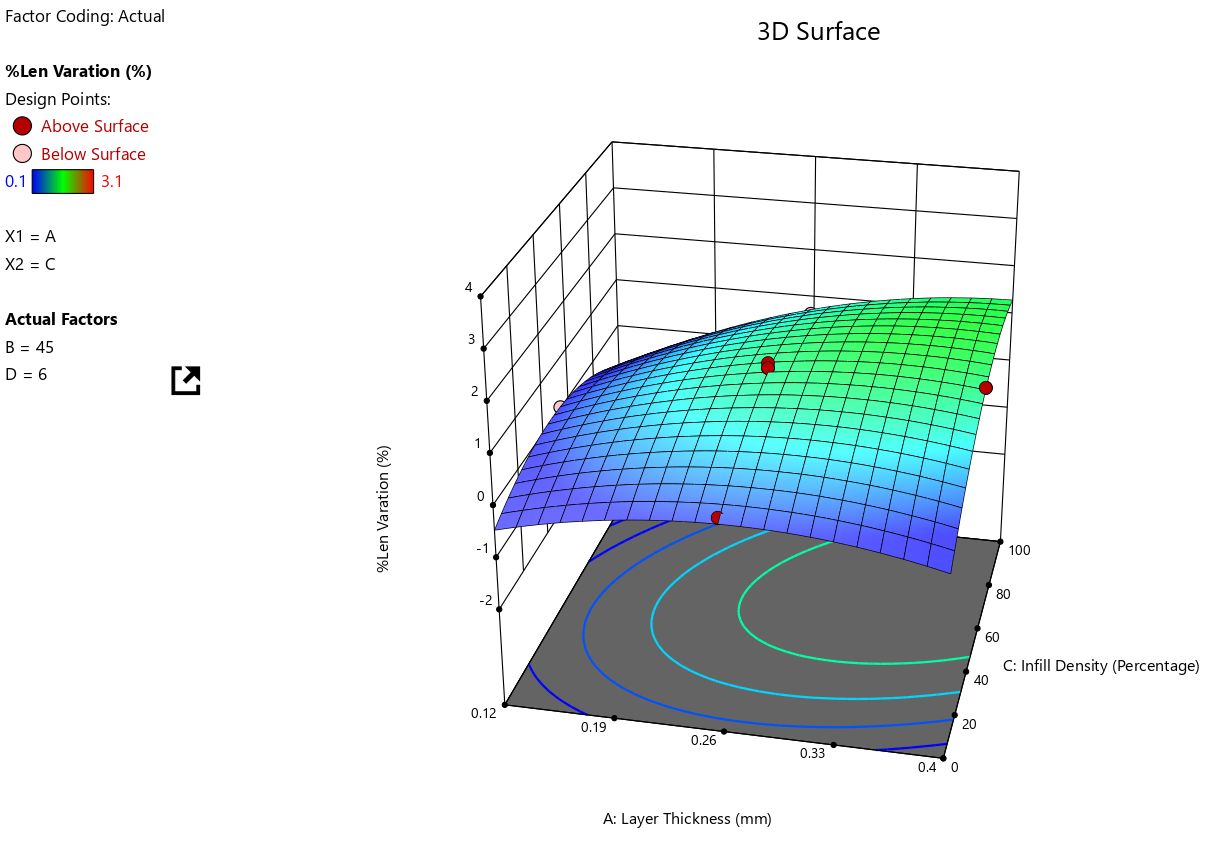
1. Build Orientation vs Infill Density



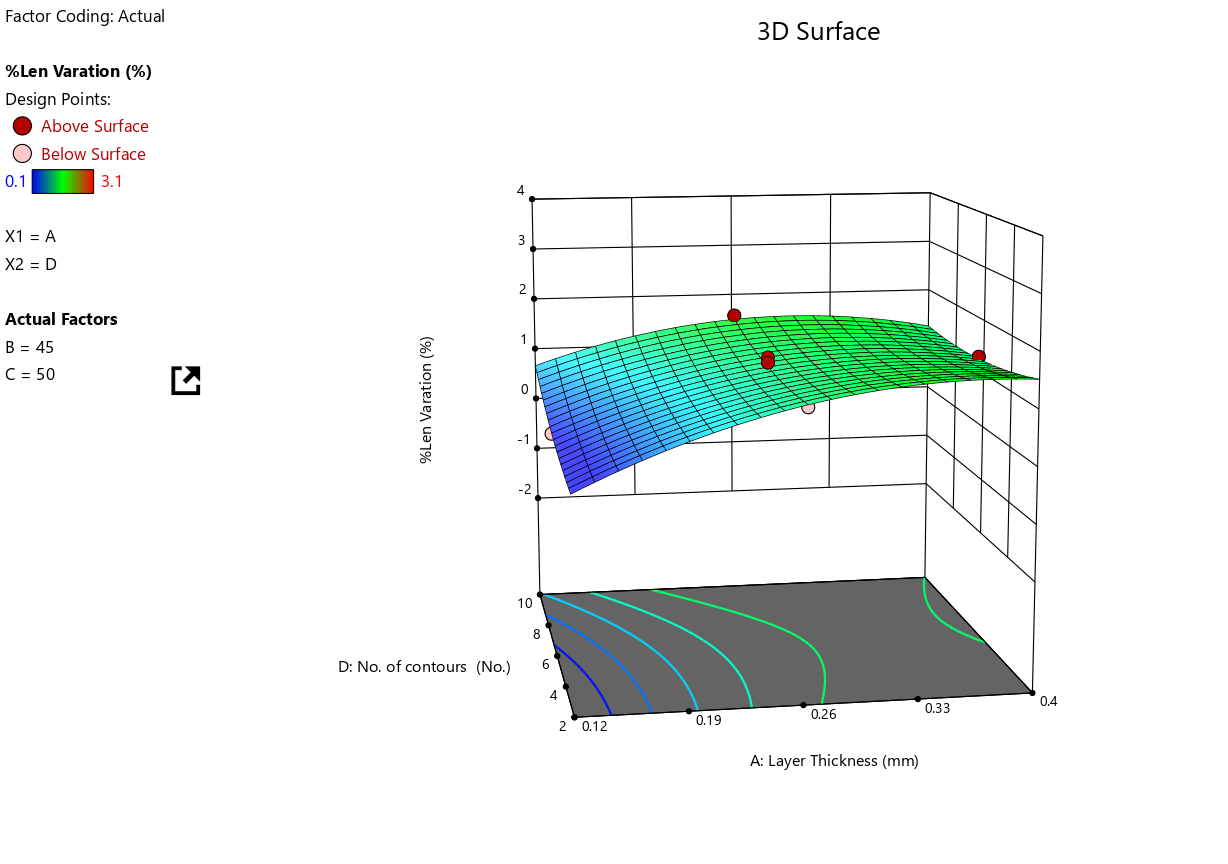
1. Build Orientation vs Infill Density



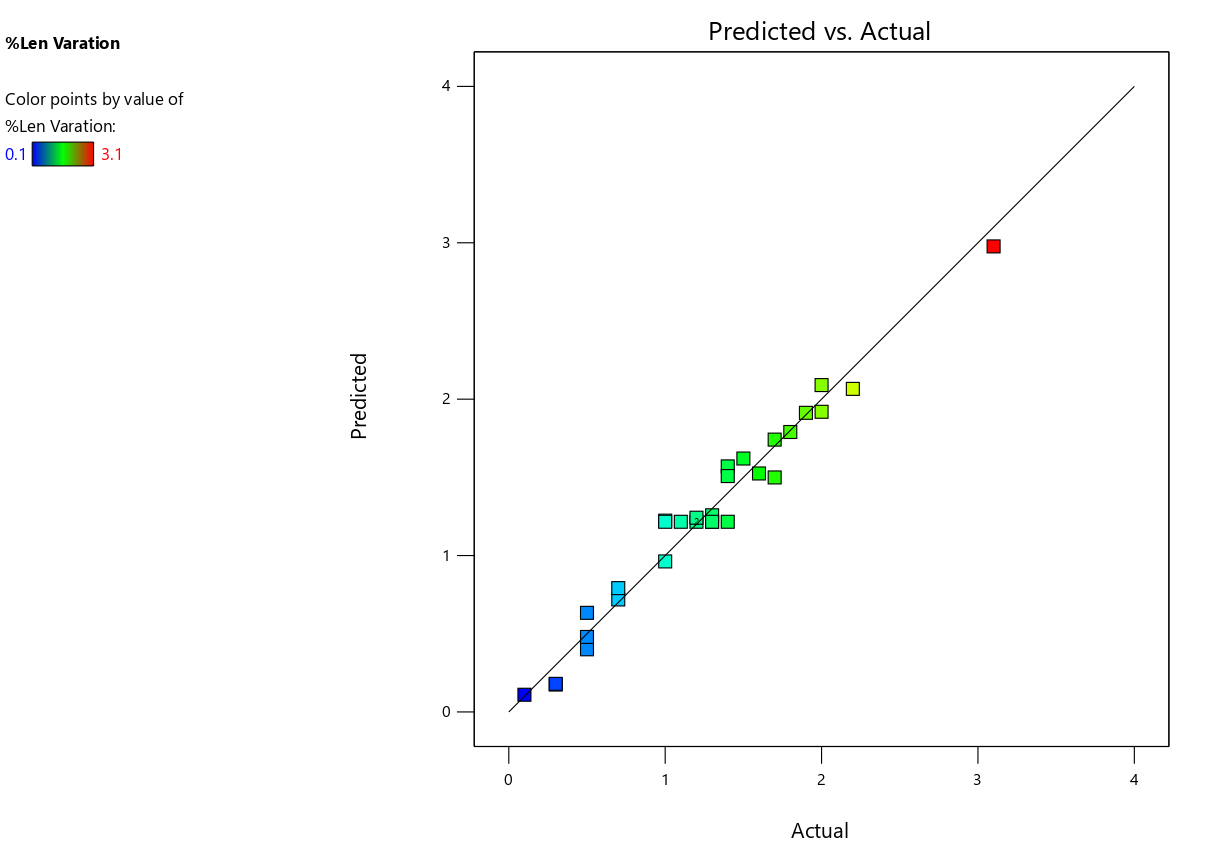
1. Layer Thickness vs Infill Density



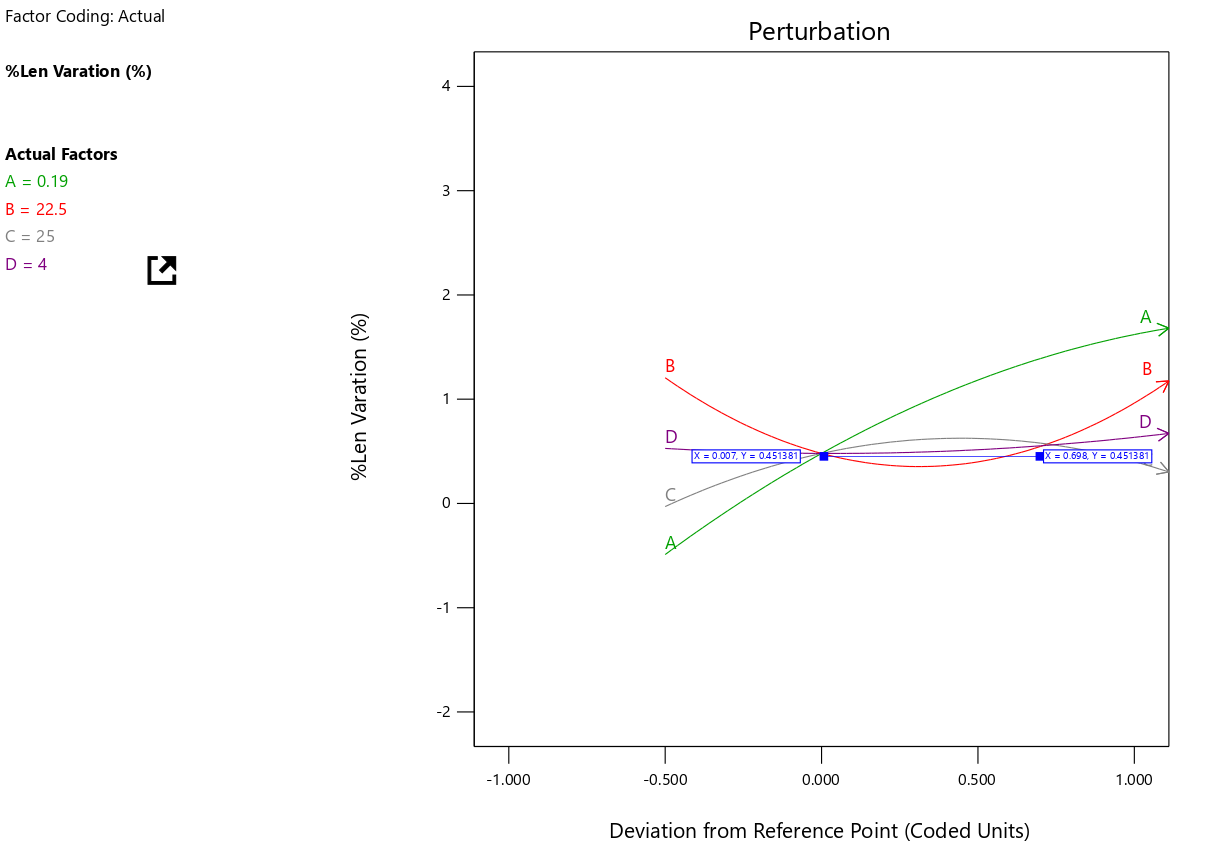
1. Layer Thickness vs No. of Countours



Predicted RSM length Variation vs Actual Value

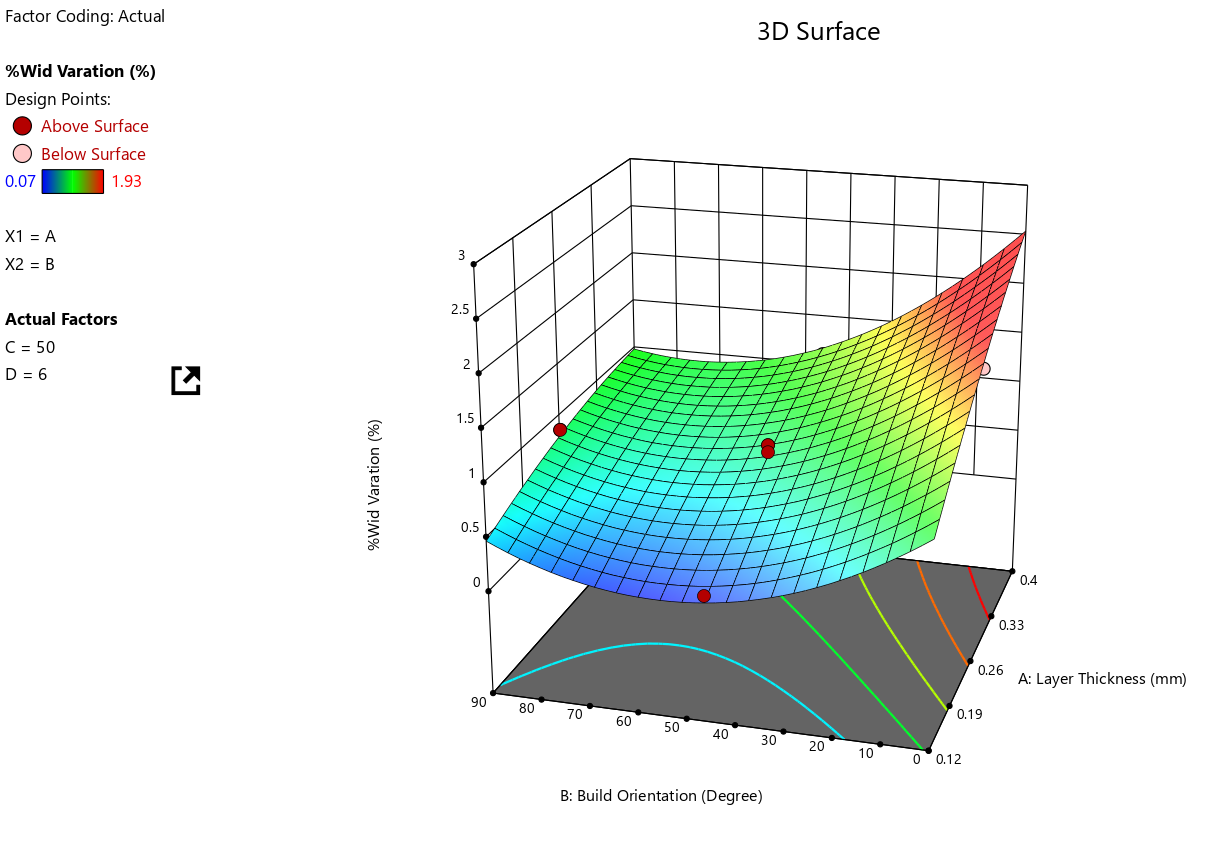


**The mean variation in length is 0.727838 and from the graph below it can be inferred that the minimum length variation is around 0.7% by adjusting the factors**

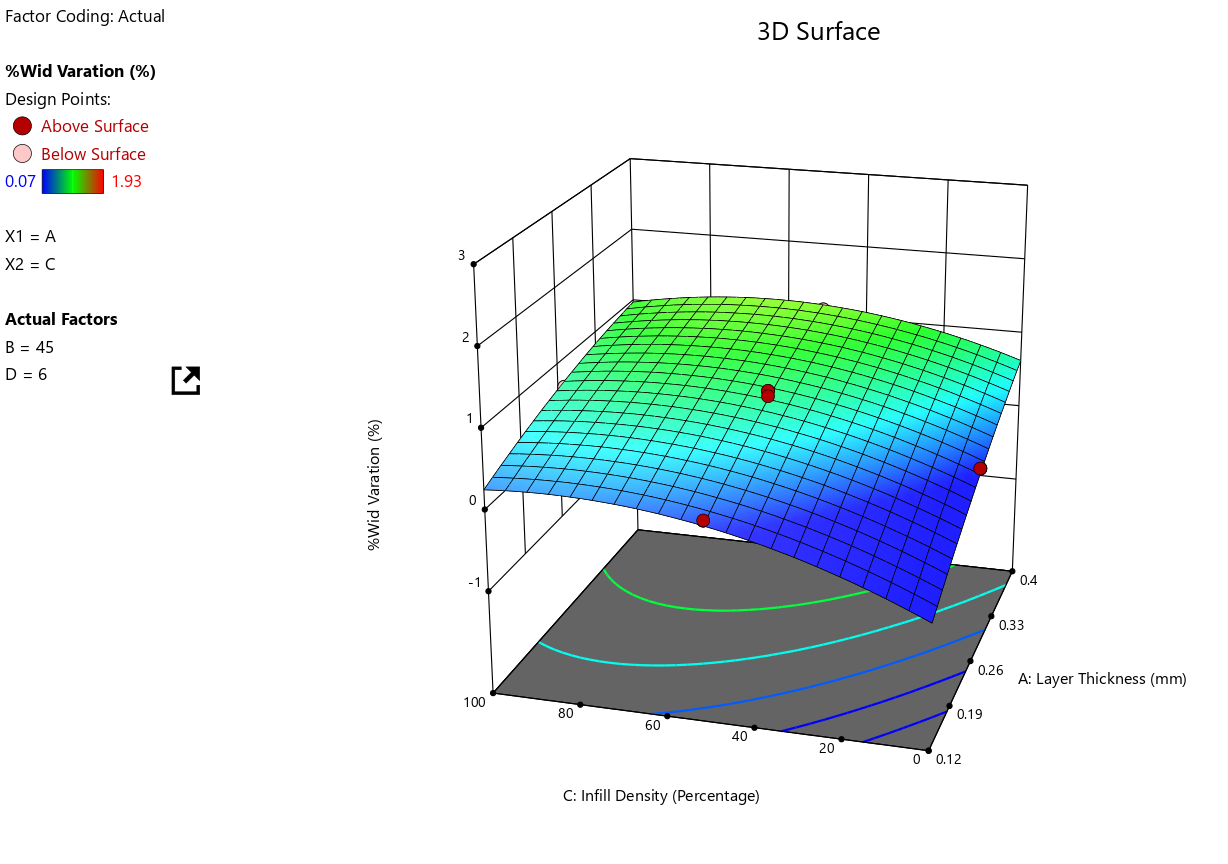


Width variation graphs

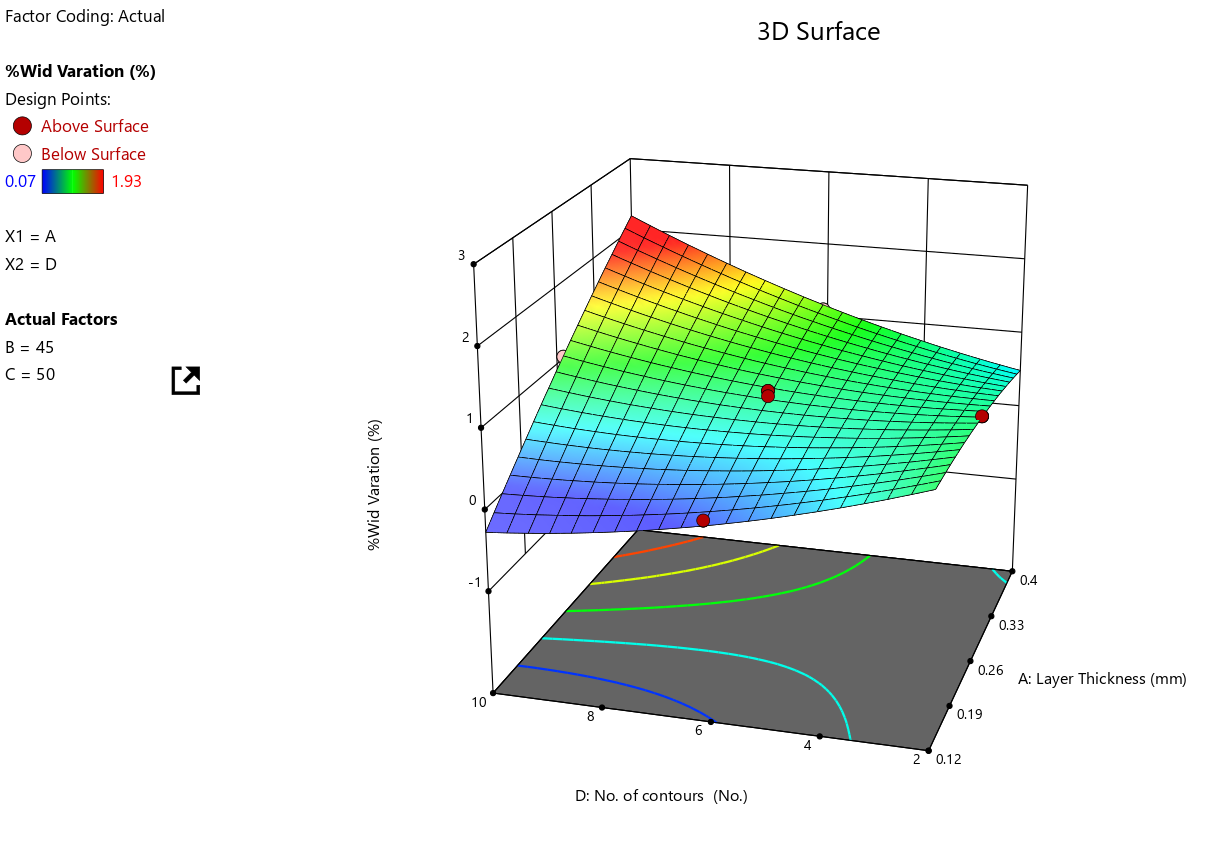
1. Layer Thickness vs Build Orientation



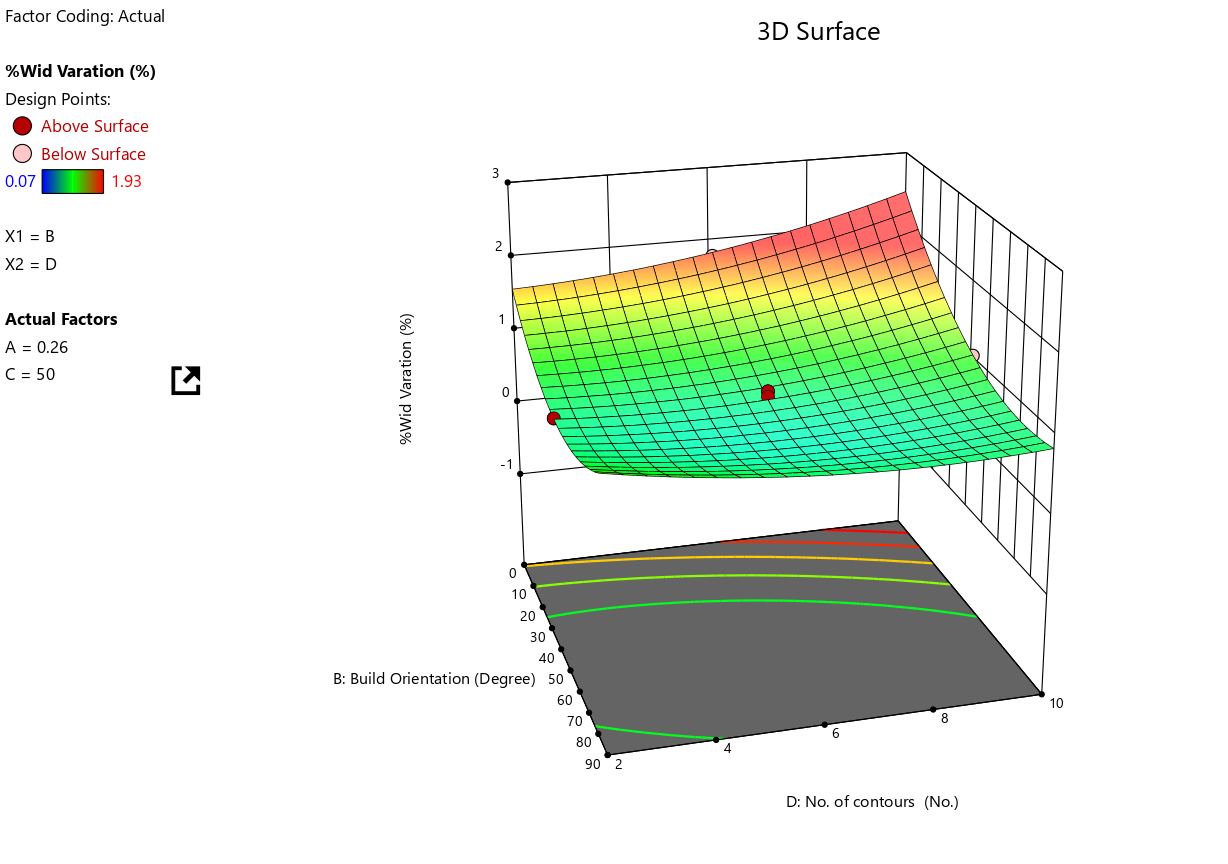
1. Layer Thickness vs Infill Density



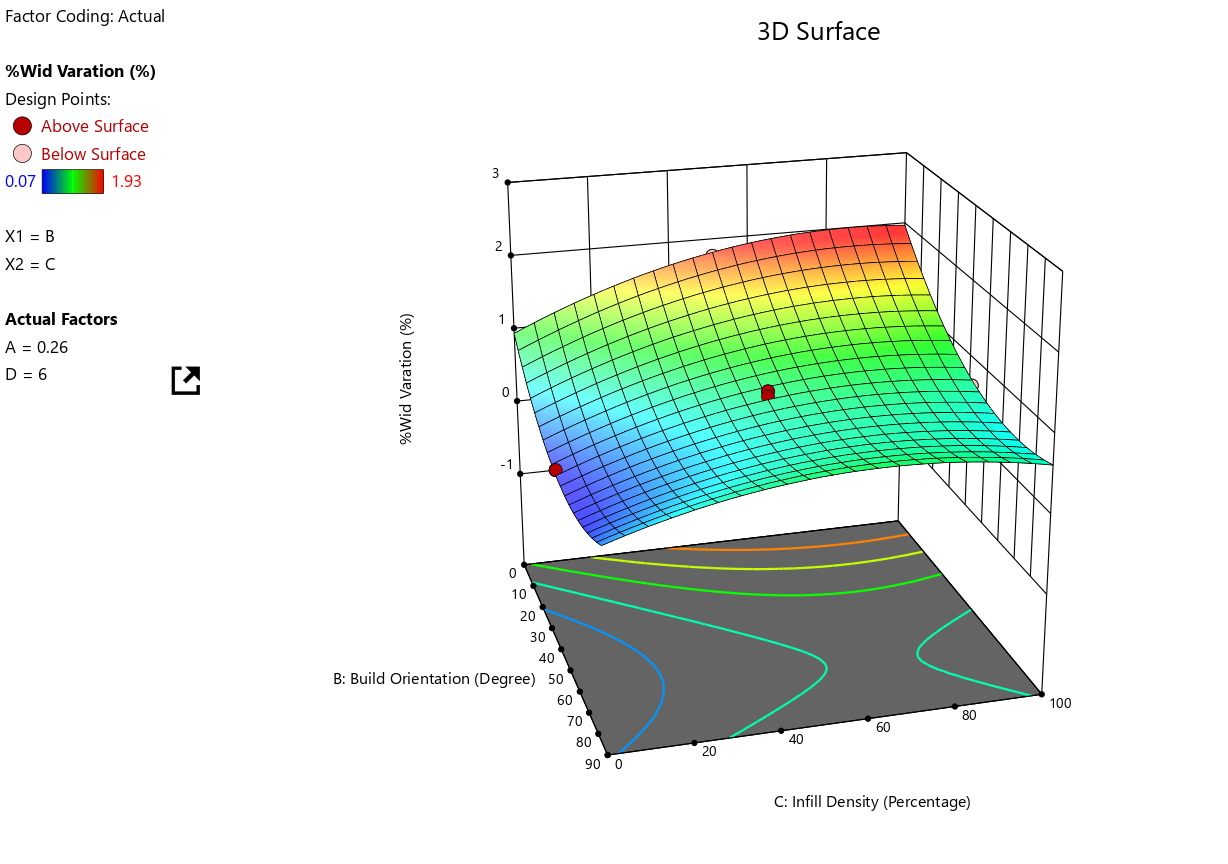
1. Layer Thickness vs No of Countours



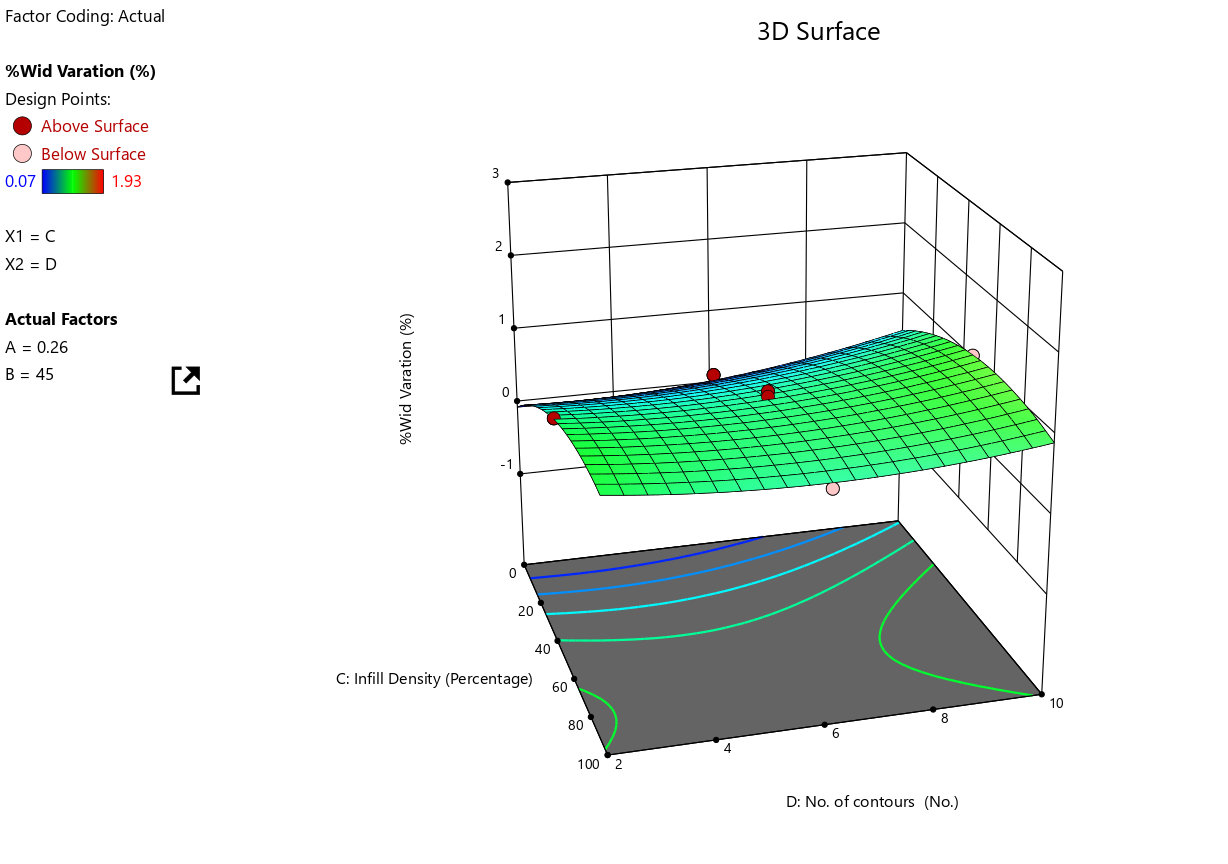
1. Build Orientation vs No of countours



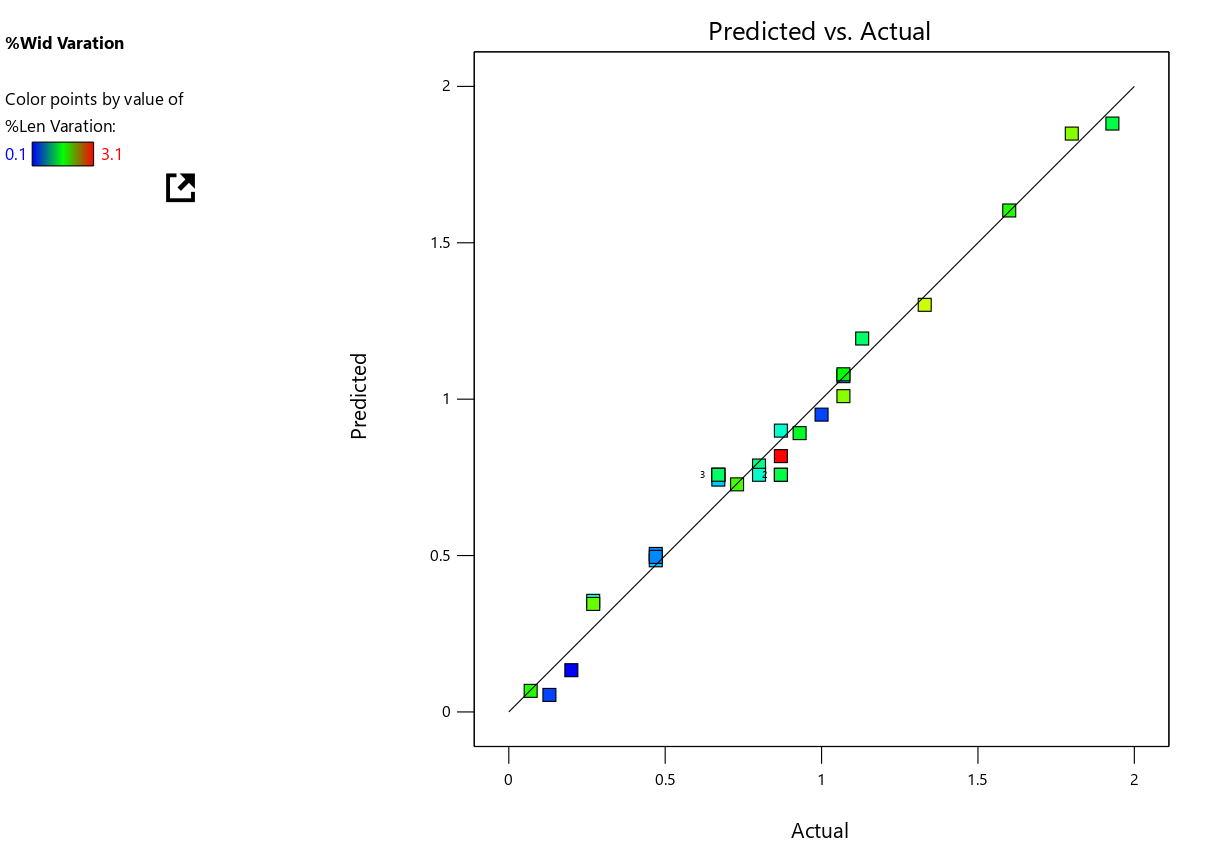
1. Build orientation vs Infill Density



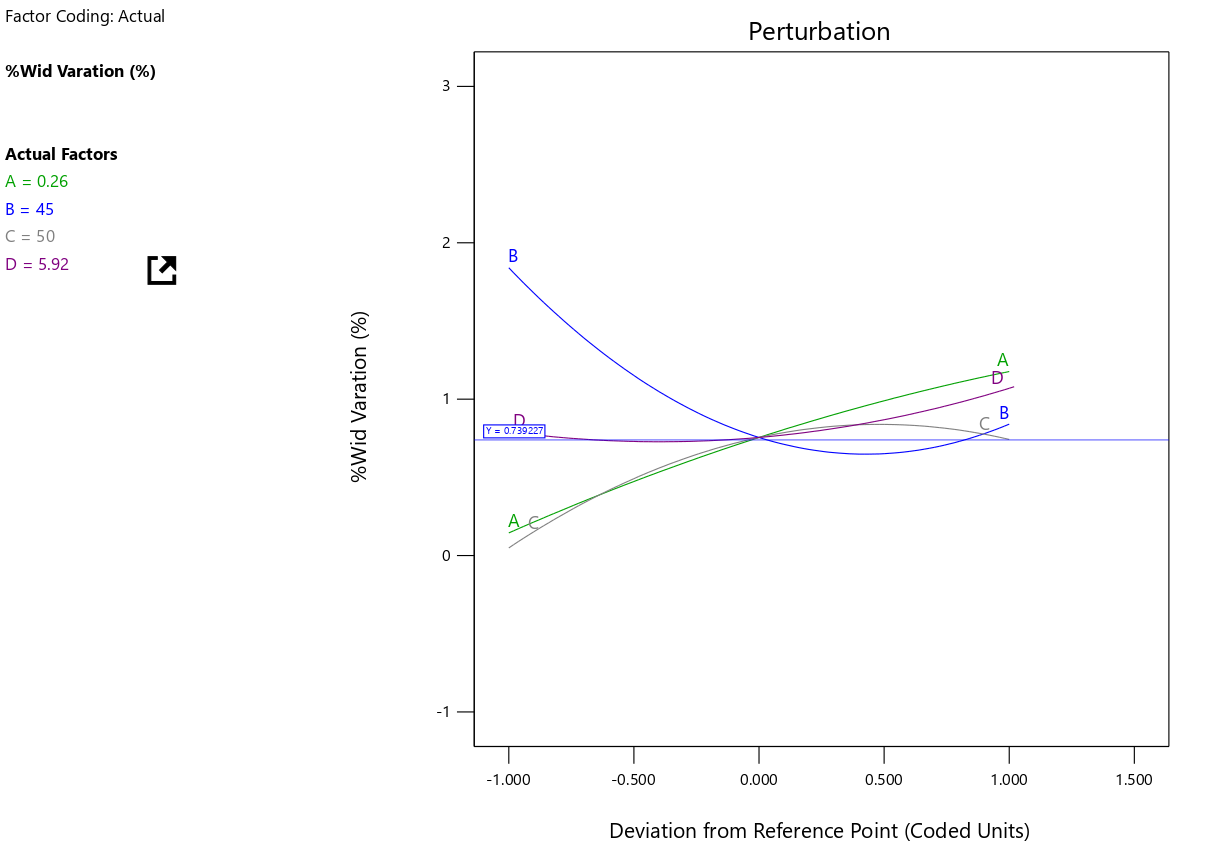
1. Infill density vs No of Countours



RSM prediction vs Actual values(coded)

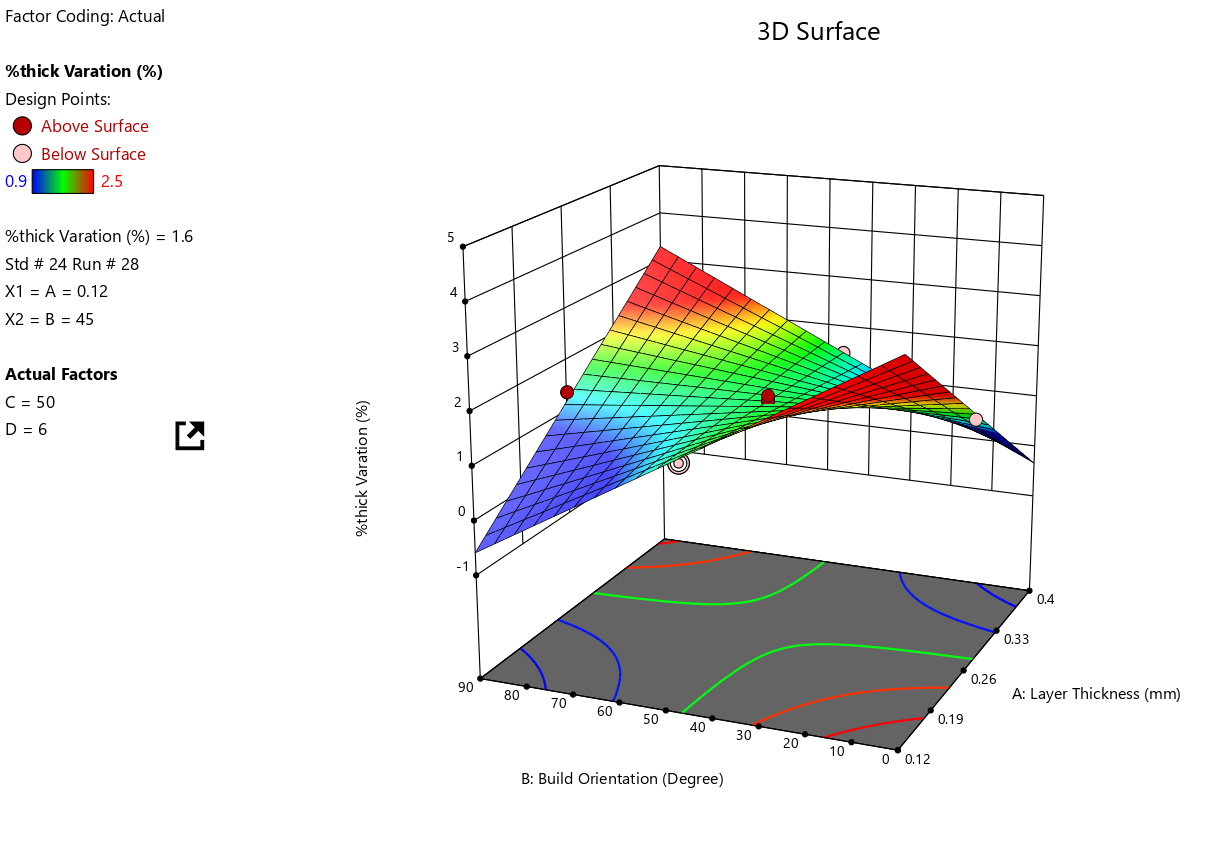


**The mean variation in width by RSM is 0.429572. From the below curve infill density palys a major role in determination of length variation along with build orientation the miniumn width variation is 0.3842 for build orientation 45 and infill density of 50**

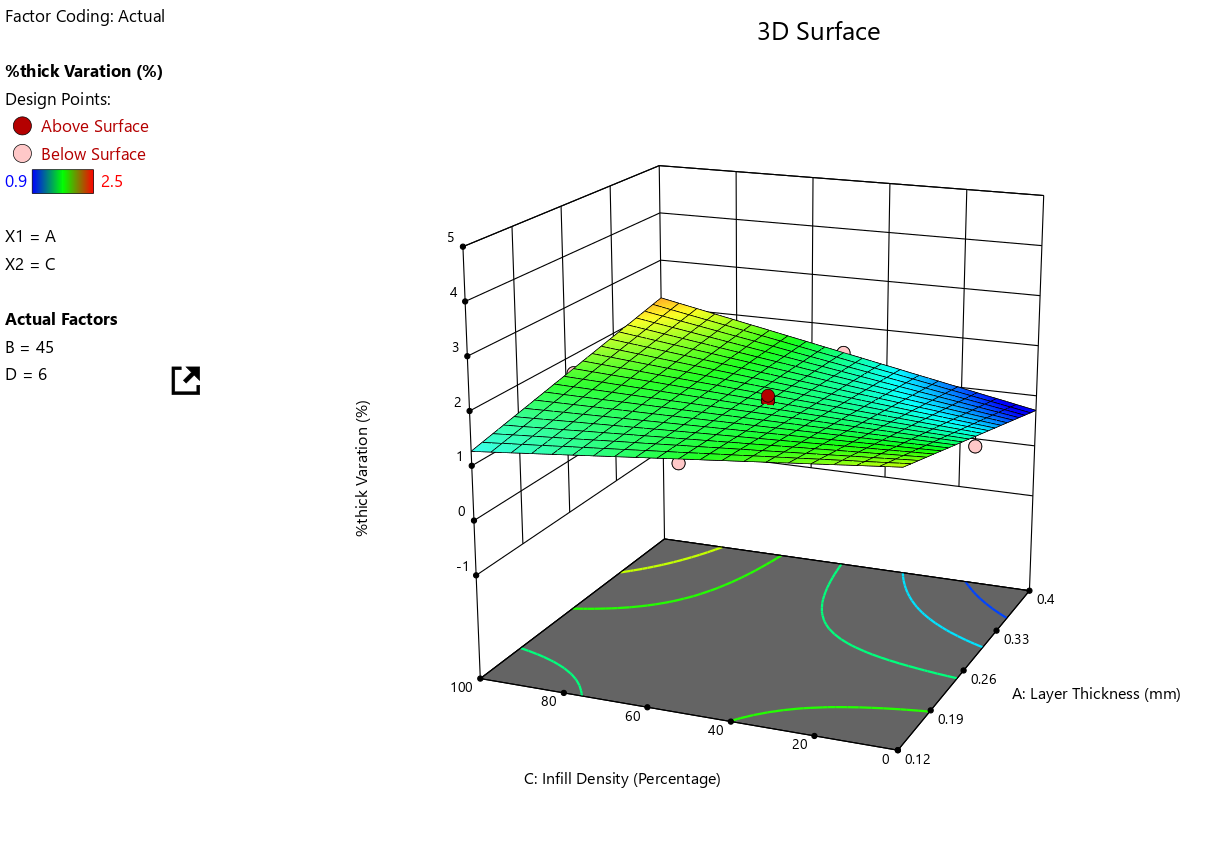


Thickness Variation

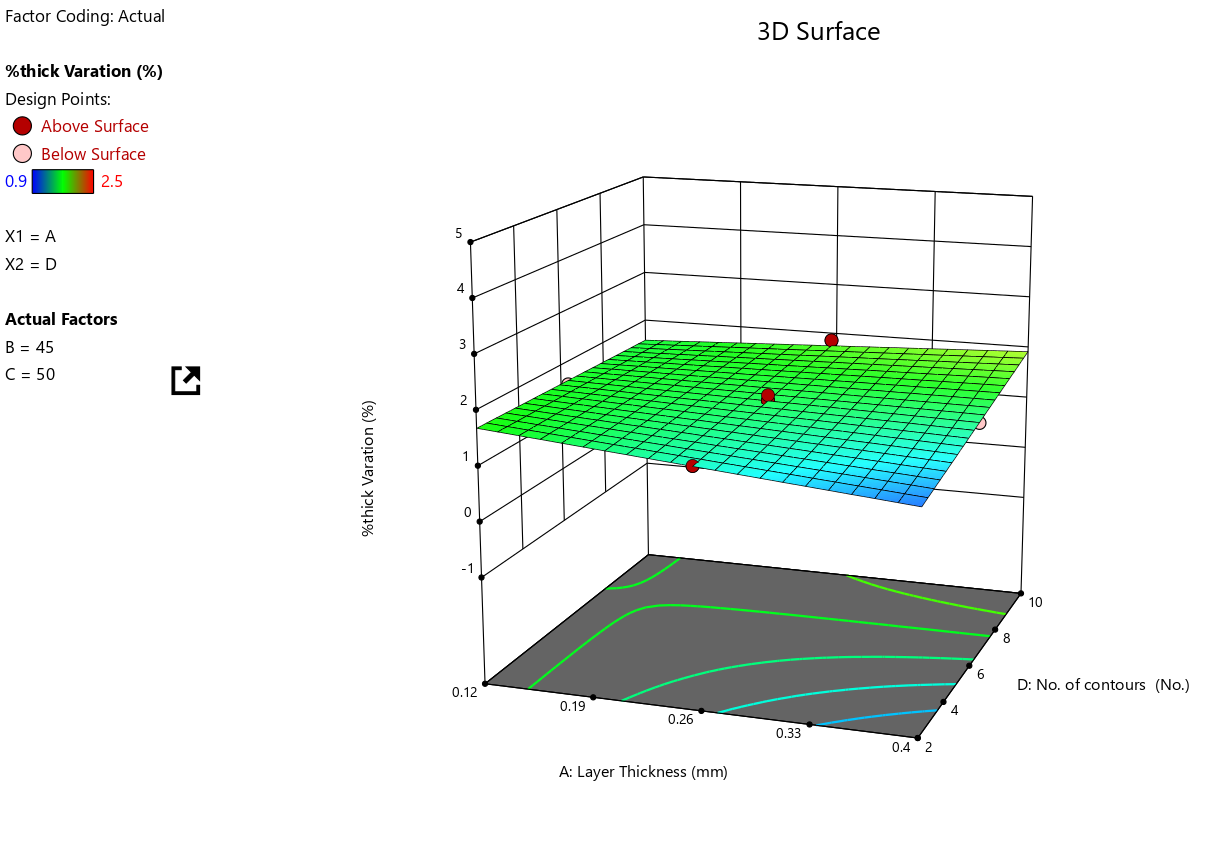
1. Layer Thickness vs Build Orientation



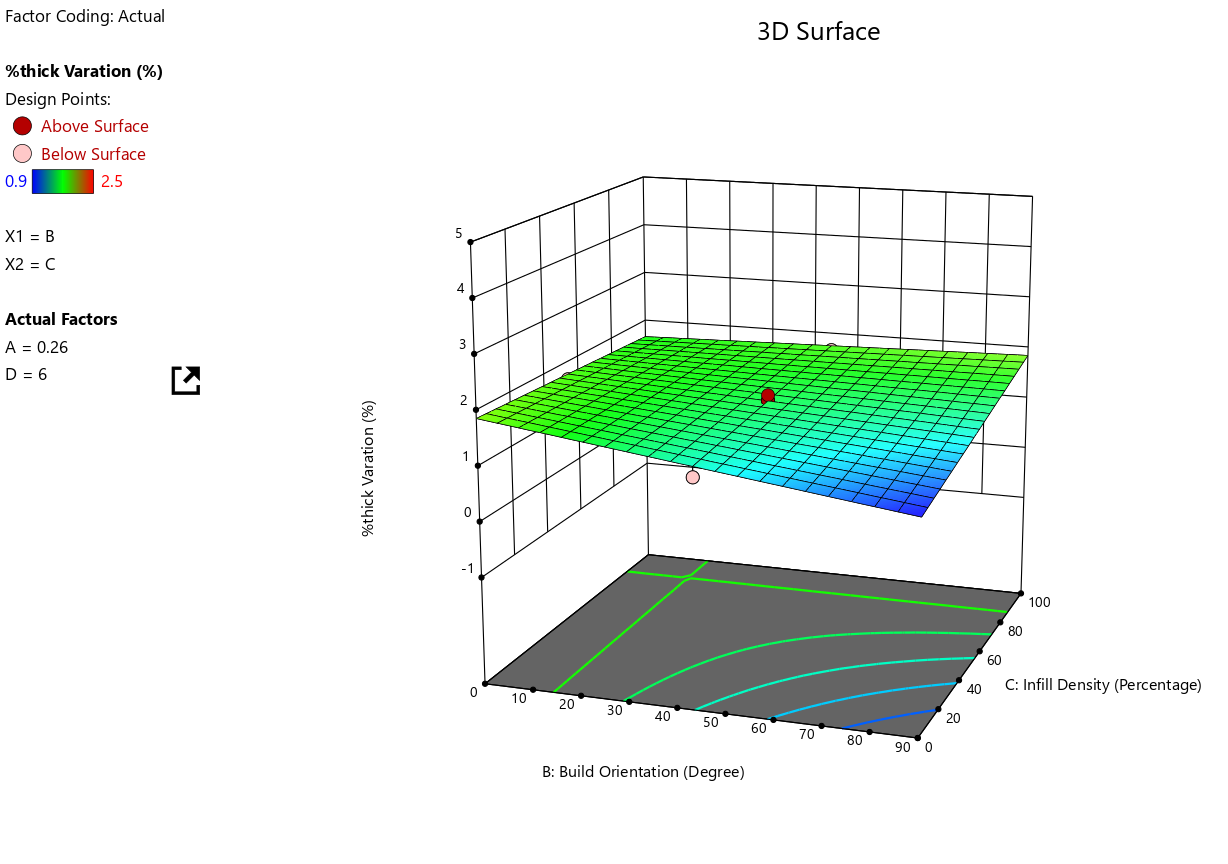
1. Layer Thickness vs Infill Density



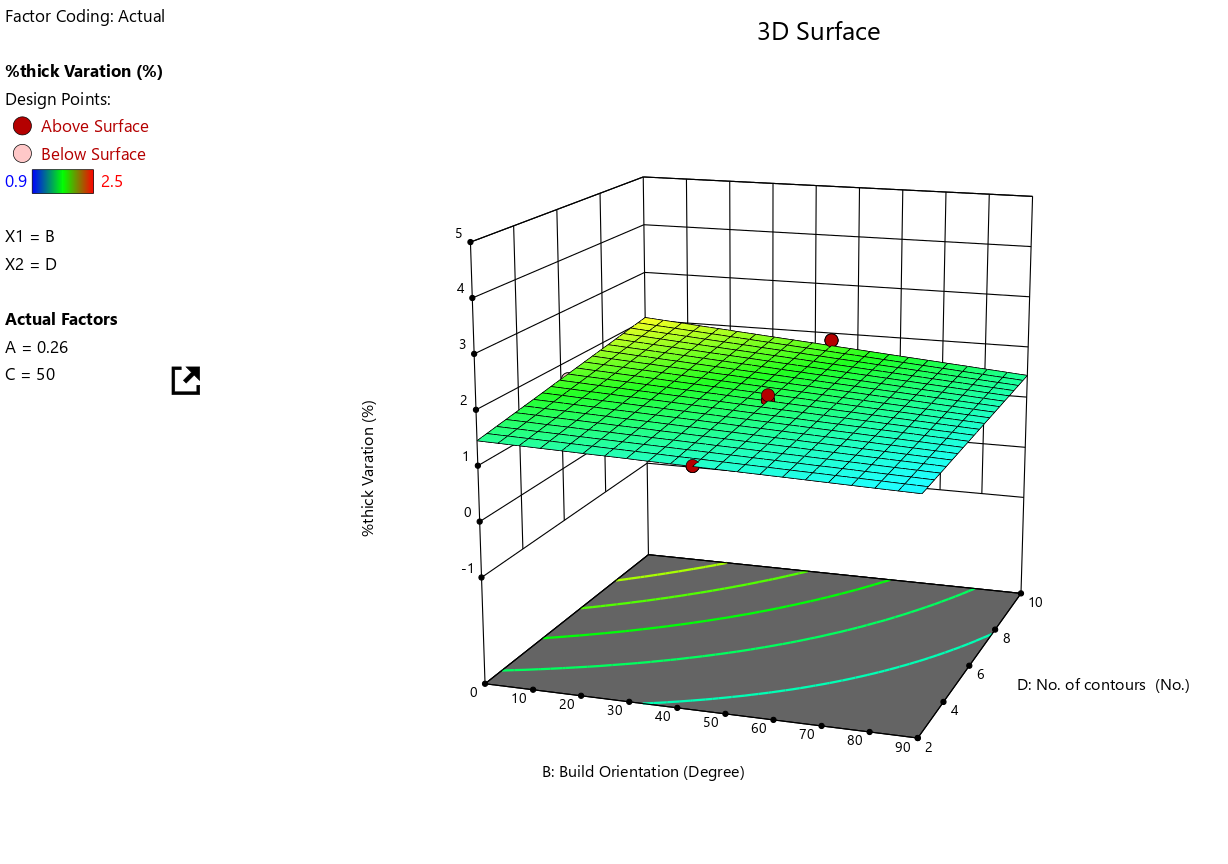
1. Layer Thickness vs No of Countours



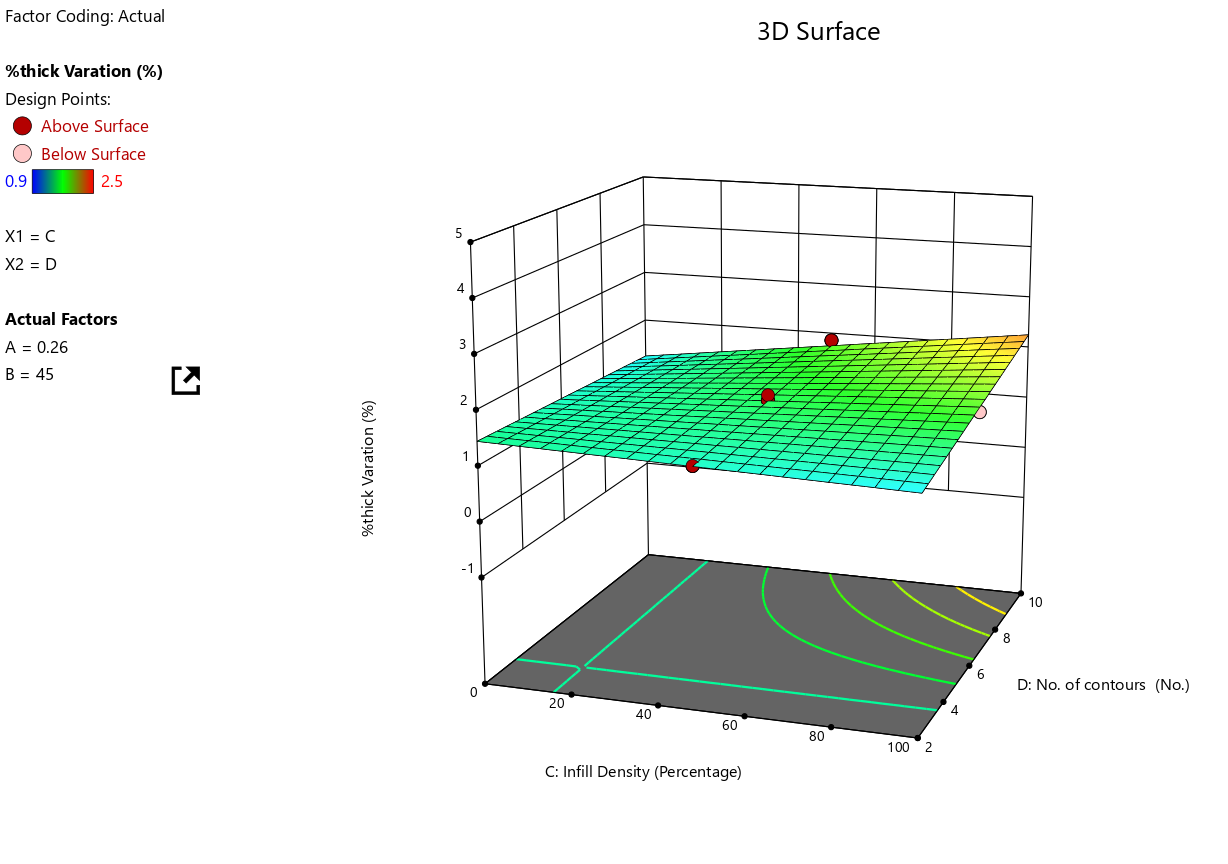
1. Build Orientation vs Infill density



1. Build Orientatio vs No of Countours



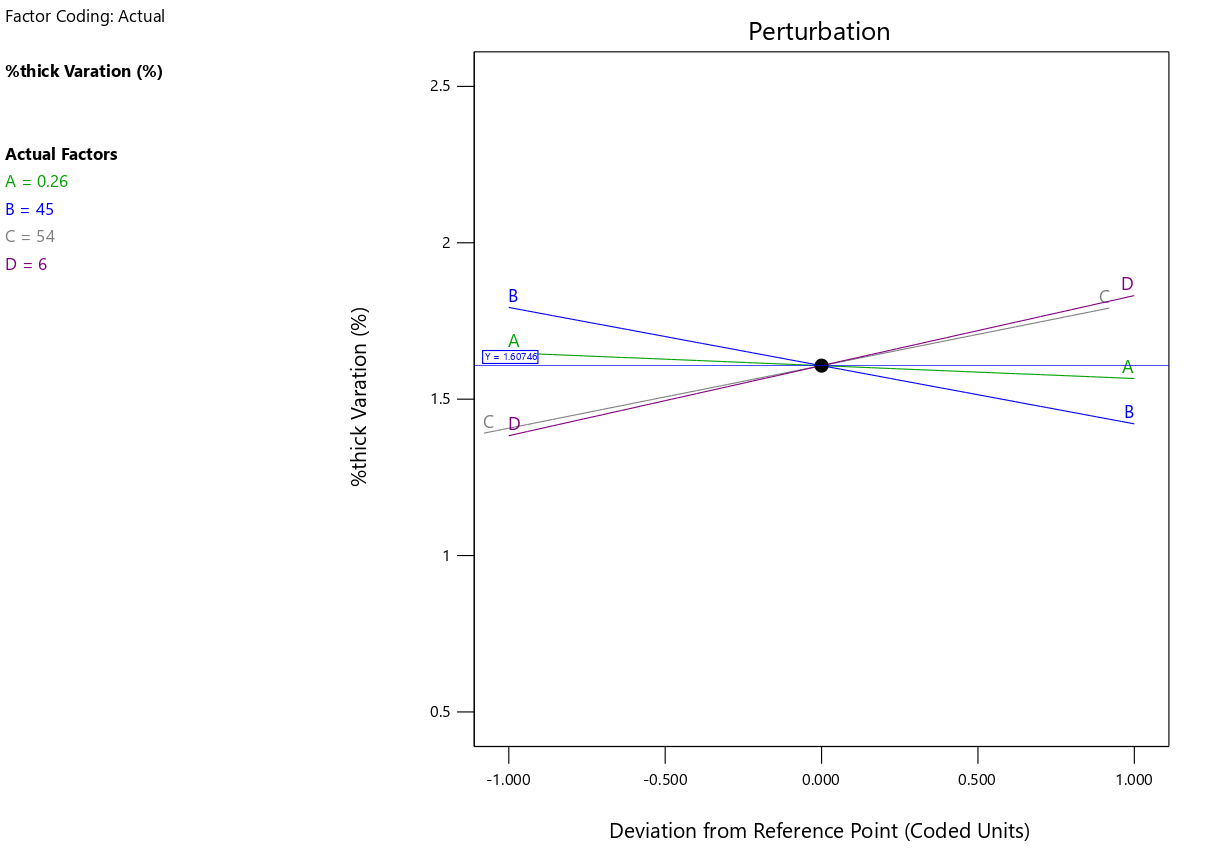
1. Infill Density vs No of Countours



RSM predicted values for width varu=iation vs actual values



**The mean variation for Thickness Variation is 1.6753. Thickness variation has the highest variation. One reason is that thickness is the smallest dimension in the model in magnitude hence its more tough to get better percentage dimension precision.From the graph below minimum thickness variation is 1.60734.**



ANOVA models:

In this study different process parameters selected for optimization to enhance the magnitude preciseness were layer thickness, Build orientation, Density and Number of contours. The process parameters range as given in Table 1 was selected on the basis of study literature and developed a design matrix by using RSM based central composite design (CCD) in the design of expert software 6.0.8. Responses in terms of percentage variation in length, width and thickness were measured by performing experiments according to Table 2. For each response value mathematical models are generated according to the second order equations (ii, iii and iv) as given below:

Precentage variation in lenght=

**(-3.05007 + 27.3058 \* A + -0.025688 \* B + 0.0147307 \* C + -0.0247663 \* D + -0.219647 \* AB + 0.075 \* AC + -0.491071 \* AD + 0.000277778 \* BC + 0.00347222 \* BD + -0.001875 \* CD + -27.1703 \* A^2 + 0.000636825 \* B^2 + -0.000293141 \* C^2 + 0.0104468 \* D^2)**

### **Response 1: %Len Varation**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Source | Sum of Squares | df | Mean Square | F-value | p-value |  |
| Model | 12.09 | 14 | 0.8638 | 34.00 | < 0.0001 | significant |
| A-Layer Thickness | 1.97 | 1 | 1.97 | 77.71 | < 0.0001 |  |
| B-Build Orientation | 1.04 | 1 | 1.04 | 40.79 | < 0.0001 |  |
| C-Infill Density | 0.5704 | 1 | 0.5704 | 22.45 | 0.0003 |  |
| D-No. of contours | 0.1204 | 1 | 0.1204 | 4.74 | 0.0459 |  |
| AB | 1.92 | 1 | 1.92 | 75.53 | < 0.0001 |  |
| AC | 0.2756 | 1 | 0.2756 | 10.85 | 0.0049 |  |
| AD | 0.0756 | 1 | 0.0756 | 2.98 | 0.1050 |  |
| BC | 0.3906 | 1 | 0.3906 | 15.37 | 0.0014 |  |
| BD | 0.3906 | 1 | 0.3906 | 15.37 | 0.0014 |  |
| CD | 0.1406 | 1 | 0.1406 | 5.53 | 0.0327 |  |
| A² | 0.4857 | 1 | 0.4857 | 19.12 | 0.0005 |  |
| B² | 2.85 | 1 | 2.85 | 112.08 | < 0.0001 |  |
| C² | 0.9205 | 1 | 0.9205 | 36.23 | < 0.0001 |  |
| D² | 0.0479 | 1 | 0.0479 | 1.88 | 0.1900 |  |
| Residual | 0.3811 | 15 | 0.0254 |  |  |  |
| Lack of Fit | 0.2728 | 10 | 0.0273 | 1.26 | 0.4224 | not significant |
| Pure Error | 0.1083 | 5 | 0.0217 |  |  |  |
| Cor Total | 12.47 | 29 |  |  |  |  |

Factor coding is **Coded**. Sum of squares is **Type III - Partial**

The **Model F-value** of 34.00 implies the model is significant. There is only a 0.01% chance that an F-value this large could occur due to noise.

**P-values** less than 0.0500 indicate model terms are significant. In this case A, B, C, D, AB, AC, BC, BD, CD, A², B², C² are significant model terms. Values greater than 0.1000 indicate the model terms are not significant. If there are many insignificant model terms (not counting those required to support hierarchy), model reduction may improve your model.

The **Lack of Fit F-value** of 1.26 implies the Lack of Fit is not significant relative to the pure error. There is a 42.24% chance that a Lack of Fit F-value this large could occur due to noise. Non-significant lack of fit is good -- we want the model to fit.

Percentage Variation in width =

**(0.35067 + 1.86215 \* A + -0.0126254 \* B + 0.03481 \* C + -0.313823 \* D + -0.0422543 \* AB + -0.0239286 \* AC + 1.25446 \* AD + -7.44444e-05 \* BC + -0.00165278 \* BD + -0.0006625 \* CD + -4.815 \* A^2 + 0.00028897 \* B^2 + -0.000143803 \* C^2 + 0.0109684 \* D^2)**

**Response 2: %Wid Varation**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Source | Sum of Squares | df | Mean Square | F-value | p-value |  |
| Model | 5.79 | 14 | 0.4136 | 60.36 | < 0.0001 | significant |
| A-Layer Thickness | 1.69 | 1 | 1.69 | 246.24 | < 0.0001 |  |
| B-Build Orientation | 1.53 | 1 | 1.53 | 223.91 | < 0.0001 |  |
| C-Infill Density | 0.7107 | 1 | 0.7107 | 103.71 | < 0.0001 |  |
| D-No. of contours | 0.1276 | 1 | 0.1276 | 18.62 | 0.0006 |  |
| AB | 0.0710 | 1 | 0.0710 | 10.36 | 0.0057 |  |
| AC | 0.0281 | 1 | 0.0281 | 4.09 | 0.0612 |  |
| AD | 0.4935 | 1 | 0.4935 | 72.02 | < 0.0001 |  |
| BC | 0.0281 | 1 | 0.0281 | 4.09 | 0.0612 |  |
| BD | 0.0885 | 1 | 0.0885 | 12.92 | 0.0027 |  |
| CD | 0.0176 | 1 | 0.0176 | 2.56 | 0.1303 |  |
| A² | 0.0153 | 1 | 0.0153 | 2.23 | 0.1564 |  |
| B² | 0.5863 | 1 | 0.5863 | 85.56 | < 0.0001 |  |
| C² | 0.2215 | 1 | 0.2215 | 32.33 | < 0.0001 |  |
| D² | 0.0528 | 1 | 0.0528 | 7.70 | 0.0141 |  |
| Residual | 0.1028 | 15 | 0.0069 |  |  |  |
| Lack of Fit | 0.0527 | 10 | 0.0053 | 0.5262 | 0.8187 | not significant |
| Pure Error | 0.0501 | 5 | 0.0100 |  |  |  |
| Cor Total | 5.89 | 29 |  |  |  |  |

Factor coding is **Coded**.  
Sum of squares is **Type III - Partial**

The **Model F-value** of 60.36 implies the model is significant. There is only a 0.01% chance that an F-value this large could occur due to noise.

**P-values** less than 0.0500 indicate model terms are significant. In this case A, B, C, D, AB, AD, BD, B², C², D² are significant model terms. Values greater than 0.1000 indicate the model terms are not significant. If there are many insignificant model terms (not counting those required to support hierarchy), model reduction may improve your model.

The **Lack of Fit F-value** of 0.53 implies the Lack of Fit is not significant relative to the pure error. There is a 81.87% chance that a Lack of Fit F-value this large could occur due to noise. Non-significant lack of fit is good -- we want the model to fit.

Precentage Variation in thickness =

**(7.56366 + -21.7779 \* A + -0.0921383 \* B + -0.0314286 \* C + -0.116071 \* D + 0.323602 \* AB + 0.0785714 \* AC + 0.446429 \* AD + 0.000133333 \* BC + -0.000555556 \* BD + 0.0015 \* CD)**

**Response 3: %thick variation**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Source | Sum of Squares | df | Mean Square | F-value | p-value |  |
| Model | 5.58 | 10 | 0.5577 | 53.63 | < 0.0001 | significant |
| A-Layer Thickness | 0.0436 | 1 | 0.0436 | 4.19 | 0.0547 |  |
| B-Build Orientation | 0.2645 | 1 | 0.2645 | 25.44 | < 0.0001 |  |
| C-Infill Density | 0.2400 | 1 | 0.2400 | 23.08 | 0.0001 |  |
| D-No. of contours | 0.2400 | 1 | 0.2400 | 23.08 | 0.0001 |  |
| AB | 4.17 | 1 | 4.17 | 401.31 | < 0.0001 |  |
| AC | 0.3025 | 1 | 0.3025 | 29.09 | < 0.0001 |  |
| AD | 0.0625 | 1 | 0.0625 | 6.01 | 0.0241 |  |
| BC | 0.0900 | 1 | 0.0900 | 8.65 | 0.0084 |  |
| BD | 0.0100 | 1 | 0.0100 | 0.9616 | 0.3391 |  |
| CD | 0.0900 | 1 | 0.0900 | 8.65 | 0.0084 |  |
| Residual | 0.1976 | 19 | 0.0104 |  |  |  |
| Lack of Fit | 0.1776 | 14 | 0.0127 | 3.17 | 0.1045 | not significant |
| Pure Error | 0.0200 | 5 | 0.0040 |  |  |  |
| Cor Total | 5.77 | 29 |  |  |  |  |

Factor coding is **Coded**.  
Sum of squares is **Type III - Partial**

The **Model F-value** of 53.63 implies the model is significant. There is only a 0.01% chance that an F-value this large could occur due to noise.

**P-values** less than 0.0500 indicate model terms are significant. In this case B, C, D, AB, AC, AD, BC, CD are significant model terms. Values greater than 0.1000 indicate the model terms are not significant. If there are many insignificant model terms (not counting those required to support hierarchy), model reduction may improve your model.

The **Lack of Fit F-value** of 3.17 implies the Lack of Fit is not significant relative to the pure error. There is a 10.45% chance that a Lack of Fit F-value this large could occur due to noise. Non-significant lack of fit is good -- we want the model to fit.

Further optimization of the FDM process parameters to enhance magnitude preciseness is necessary as the above discussed models’ output is not much significant as our objective and also developed mathematical equations cannot find out the accurate results. So further hybrid techniques were applied to optimize the FDM process parameters.

Response Surface Methodology – Genetic Algorithm

In GA epochs is a measure which shows the performance and the number of cycles that processed by the genetic algorithm for various numbers of generations. It was performed for experiments using 500 generations. Single -point crossover was used as a crossover function and adaptive feasible mutation was taken as random.

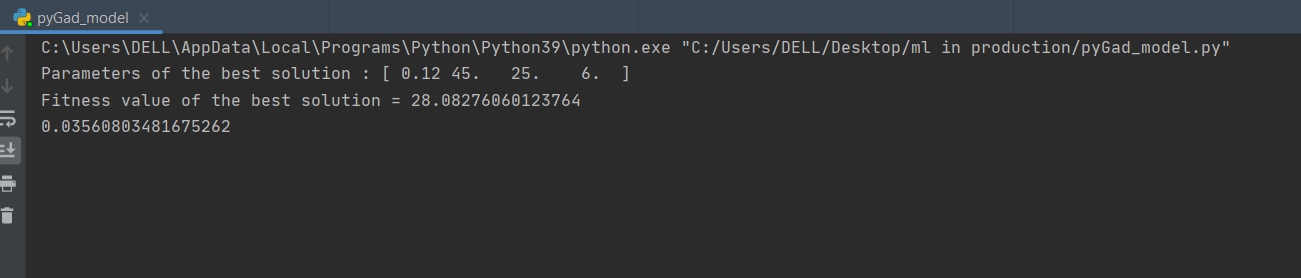
num\_generations = 500  
num\_parents\_mating = 4  
  
sol\_per\_pop = 10  
num\_genes = 4  
  
parent\_selection\_type = "sus"  
keep\_parents = 1

crossover\_type = "single\_point"

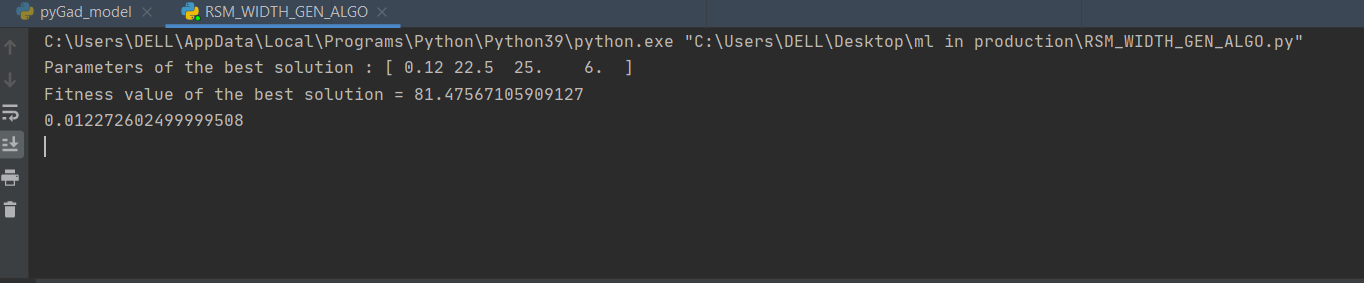
mutation\_type = "random"  
mutation\_percent\_genes = 10

The hybrid RSM-GA has been run to get optimum process parameters for least percentage variation in different magnitudes in the below code by fixing upper and lower bound parameter limits as found out from the design matrix of RSM.

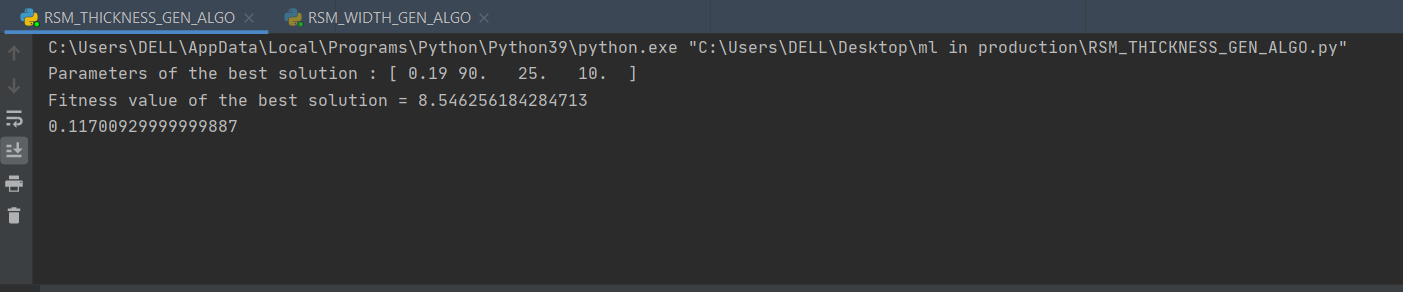
The minimum percentage variation in length obtained with RSM-GA is 0.03560803481675262% at process parameters (layer thickness 0.12 mm, build orientation 45°, infll density 25% and number of contours 6)

****

The minimum percentage variation in Width obtained with RSM-GA 0.012272602499999508% at process parameters (layer thickness 0.12 mm, build orientation 22.5°, infll density 25% and number of contours 6)

****

The minimum percentage variation in Width obtained with RSM-GA 0.11700929999999887% at process parameters (layer thickness 0.19 mm, build orientation 90°, infll density 25% and number of contours 10)



**Code Archive:**

**Code for RSM GA Lengh optimization:**

import pygad  
import numpy  
  
lst = []  
def fitness\_func(solution, solution\_idx):  
 a = solution[0]  
 b = solution[1]  
 c = solution[2]  
 d = solution[3]  
  
 output = -3.0500731235557 + (27.305807814185 \* a) - (0.02568795511004 \* b) + \  
 (0.01473071848973 \* c) - (0.024766290222257 \* d) - (0.21964743016038 \* a \* b) \  
 + (0.075 \* a \* c) - (0.49107142857143 \* a \* d) + (0.00027777777777778 \* b \* c) \  
 + (0.0034722222222222 \* b \* d) - (0.001875 \* c \* d) - (27.170294216721 \* a \* a) \  
 + (0.00063682469472212 \* b \* b) - (0.00029314051823067 \* c \* c) + \  
 (0.010446794026458 \* d \* d)  
 if output > 0 and output < 0.03560803481675263 :  
 lst.append(solution)  
 if output < 0:  
 fitness = -100000  
 else:  
 fitness = 1.0 / (numpy.abs(output - 0) + 0.000001)  
 return fitness  
  
  
fitness\_function = fitness\_func  
  
num\_generations = 500  
num\_parents\_mating = 4  
  
sol\_per\_pop = 10  
num\_genes = 4  
  
parent\_selection\_type = "sus"  
keep\_parents = 1  
  
crossover\_type = "single\_point"  
#gene\_space=[{'low': 0.12, 'high': 0.4}, {'low': 0, 'high': 90}, {'low': 0, 'high': 100},{'low': 2, 'high': 10}]  
  
#gene\_space = [[0.12, 0.19, 0.26, 0.33],[0,22.5,45,67.5,90],[25,50,75,100],[2,4,6,8,10]]  
  
mutation\_type = "random"  
mutation\_percent\_genes = 10  
  
ga\_instance = pygad.GA(num\_generations=num\_generations,  
 num\_parents\_mating=num\_parents\_mating,  
 fitness\_func=fitness\_function,  
 sol\_per\_pop=sol\_per\_pop,  
 init\_range\_low=0,  
 init\_range\_high=100,  
 num\_genes=num\_genes,  
 parent\_selection\_type=parent\_selection\_type,  
 keep\_parents=keep\_parents,  
 gene\_space = [[0.12, 0.19, 0.26, 0.33],[22.5,45,67.5,90],[25,50,75,100],[2,4,6,8,10]],  
 crossover\_type=crossover\_type,  
 mutation\_type=mutation\_type,  
 mutation\_num\_genes=4,  
 mutation\_percent\_genes=mutation\_percent\_genes,  
 allow\_duplicate\_genes=False)  
  
ga\_instance.run()  
  
solution, solution\_fitness, solution\_idx = ga\_instance.best\_solution()  
print("Parameters of the best solution : "+ str(solution))  
print("Fitness value of the best solution = {solution\_fitness}".format(solution\_fitness=solution\_fitness))  
  
#print(len(lst))  
#print(lst)  
  
a = solution[0]  
b = solution[1]  
c = solution[2]  
d = solution[3]  
  
output = -3.0500731235557 + (27.305807814185 \* a) - (0.02568795511004 \* b) + \  
 (0.01473071848973 \* c) - (0.024766290222257 \* d) - (0.21964743016038 \* a \* b) \  
 + (0.075 \* a \* c) - (0.49107142857143 \* a \* d) + (0.00027777777777778 \* b \* c) \  
 + (0.0034722222222222 \* b \* d) - (0.001875 \* c \* d) - (27.170294216721 \* a \* a) \  
 + (0.00063682469472212 \* b \* b) - (0.00029314051823067 \* c \* c) + \  
 (0.010446794026458 \* d \* d)  
  
print(output)

**Code for RSM GA Width optimization:**

import pygad  
import numpy  
  
lst = []  
def fitness\_func(solution, solution\_idx):  
 A = solution[0]  
 B = solution[1]  
 C = solution[2]  
 D = solution[3]  
  
 output = 0.35067 + (1.86215 \* A) + (-0.0126254 \* B) + (0.03481 \* C) + (-0.313823 \* D) + (-0.0422543 \* A\*B) + \  
 (-0.0239286 \* A\*C) + (1.25446 \* A\*D) + (-7.44444e-05 \* B\*C) + (-0.00165278 \* B\*D) + (-0.0006625 \* C\*D) + \  
 (-4.815 \* A\*A) + (0.00028897 \* B\*B) + (-0.000143803 \* C\*C) + (0.0109684 \* D\*D)  
  
 if output > 0 and output < 0.03560803481675263 :  
 lst.append(solution)  
 if output < 0:  
 fitness = -100000  
 else:  
 fitness = 1.0 / (numpy.abs(output - 0) + 0.000001)  
 return fitness  
  
  
fitness\_function = fitness\_func  
  
num\_generations = 500  
num\_parents\_mating = 4  
  
sol\_per\_pop = 10  
num\_genes = 4  
  
parent\_selection\_type = "sus"  
keep\_parents = 1  
  
crossover\_type = "single\_point"  
#gene\_space=[{'low': 0.12, 'high': 0.4}, {'low': 0, 'high': 90}, {'low': 0, 'high': 100},{'low': 2, 'high': 10}]  
  
#gene\_space = [[0.12, 0.19, 0.26, 0.33],[0,22.5,45,67.5,90],[25,50,75,100],[2,4,6,8,10]]  
  
mutation\_type = "random"  
mutation\_percent\_genes = 10  
  
ga\_instance = pygad.GA(num\_generations=num\_generations,  
 num\_parents\_mating=num\_parents\_mating,  
 fitness\_func=fitness\_function,  
 sol\_per\_pop=sol\_per\_pop,  
 init\_range\_low=0,  
 init\_range\_high=100,  
 num\_genes=num\_genes,  
 parent\_selection\_type=parent\_selection\_type,  
 keep\_parents=keep\_parents,  
 gene\_space = [[0.12, 0.19, 0.26, 0.33],[22.5,45,67.5,90],[25,50,75,100],[2,4,6,8,10]],  
 crossover\_type=crossover\_type,  
 mutation\_type=mutation\_type,  
 mutation\_num\_genes=4,  
 mutation\_percent\_genes=mutation\_percent\_genes,  
 allow\_duplicate\_genes=False)  
  
ga\_instance.run()  
  
solution, solution\_fitness, solution\_idx = ga\_instance.best\_solution()  
print("Parameters of the best solution : "+ str(solution))  
print("Fitness value of the best solution = {solution\_fitness}".format(solution\_fitness=solution\_fitness))  
  
#print(len(lst))  
#print(lst)  
  
A = solution[0]  
B = solution[1]  
C = solution[2]  
D = solution[3]  
  
output = 0.35067 + (1.86215 \* A) + (-0.0126254 \* B) + (0.03481 \* C) + (-0.313823 \* D) + (-0.0422543 \* A\*B) + \  
 (-0.0239286 \* A\*C) + (1.25446 \* A\*D) + (-7.44444e-05 \* B\*C) + (-0.00165278 \* B\*D) + (-0.0006625 \* C\*D) + \  
 (-4.815 \* A\*A) + (0.00028897 \* B\*B) + (-0.000143803 \* C\*C) + (0.0109684 \* D\*D)  
  
print(output)

**Code for RSM GA Thickness optimization:**

import pygad  
import numpy  
  
lst = []  
def fitness\_func(solution, solution\_idx):  
 A = solution[0]  
 B = solution[1]  
 C = solution[2]  
 D = solution[3]  
  
 output = 7.56366 + (-21.7779 \* A) + (-0.0921383 \* B) + (-0.0314286 \* C) + (-0.116071 \* D) + (0.323602 \* A\*B) + \  
 (0.0785714 \* A\*C) + (0.446429 \* A\*D) + (0.000133333 \* B\*C) + (-0.000555556 \* B\*D) + (0.0015 \* C\*D)  
  
  
 if output > 0 and output < 0.03560803481675263 :  
 lst.append(solution)  
 if output < 0:  
 fitness = -100000  
 else:  
 fitness = 1.0 / (numpy.abs(output - 0) + 0.000001)  
 return fitness  
  
  
fitness\_function = fitness\_func  
  
num\_generations = 500  
num\_parents\_mating = 4  
  
sol\_per\_pop = 10  
num\_genes = 4  
  
parent\_selection\_type = "sus"  
keep\_parents = 1  
  
crossover\_type = "single\_point"  
#gene\_space=[{'low': 0.12, 'high': 0.4}, {'low': 0, 'high': 90}, {'low': 0, 'high': 100},{'low': 2, 'high': 10}]  
  
#gene\_space = [[0.12, 0.19, 0.26, 0.33],[0,22.5,45,67.5,90],[25,50,75,100],[2,4,6,8,10]]  
  
mutation\_type = "random"  
mutation\_percent\_genes = 10  
  
ga\_instance = pygad.GA(num\_generations=num\_generations,  
 num\_parents\_mating=num\_parents\_mating,  
 fitness\_func=fitness\_function,  
 sol\_per\_pop=sol\_per\_pop,  
 init\_range\_low=0,  
 init\_range\_high=100,  
 num\_genes=num\_genes,  
 parent\_selection\_type=parent\_selection\_type,  
 keep\_parents=keep\_parents,  
 gene\_space = [[0.12, 0.19, 0.26, 0.33],[22.5,45,67.5,90],[25,50,75,100],[2,4,6,8,10]],  
 crossover\_type=crossover\_type,  
 mutation\_type=mutation\_type,  
 mutation\_num\_genes=4,  
 mutation\_percent\_genes=mutation\_percent\_genes,  
 allow\_duplicate\_genes=False)  
  
ga\_instance.run()  
  
solution, solution\_fitness, solution\_idx = ga\_instance.best\_solution()  
print("Parameters of the best solution : "+ str(solution))  
print("Fitness value of the best solution = {solution\_fitness}".format(solution\_fitness=solution\_fitness))  
  
#print(len(lst))  
#print(lst)  
  
A = solution[0]  
B = solution[1]  
C = solution[2]  
D = solution[3]  
  
output = 7.56366 + (-21.7779 \* A) + (-0.0921383 \* B) + (-0.0314286 \* C) + (-0.116071 \* D) + (0.323602 \* A\*B) + \  
 (0.0785714 \* A\*C) + (0.446429 \* A\*D) + (0.000133333 \* B\*C) + (-0.000555556 \* B\*D) + (0.0015 \* C\*D)  
  
print(output)

Results Gathering:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Quantity | Minimim Value | Layer Thickness | Build Orientiation | Infill Density | No of Countours |
| % Length Variation | 0.03560803481675262% | 0.12 mm | 45° | 25% | 6 |
| % Width Variation | 0.012272602499999508% | 0.12 mm | 22.5° | 25% | 6 |
| % Thickness  Variation | 0.11700929999999887% | 0.19 mm | 90° | 25% | 10 |

Remaining work:

Printing at least 3 parts for the above-mentioned parameter values respectively and cross checking the minimum variation for the corresponding dimension variation.