B.Tech Project Report

Creation of Autocad Plugin to draw Block model of an ore based on Bore hole log data

Under the guidance of:

DR. ASHOK JAISWAL
(Associate Professor)
(DEPT. OF MINING ENGINEERING IIT BHU Varanasi)

Submitted by:

Rajat Kumar Khandelwal
Roll no. 17155069
B.Tech (Part -4)
(DEPT. OF MINING ENGINEERING)

Department of Mining Engineering
IIT (BHU), Varanasi

November, 2020

Acknowledgements

In the accomplishment of this project successfully, many people have best owned upon me their blessings and the heart pledged support, this time I am utilizing to thank all the people who have been concerned with this project.

I would like to express my deep gratitude to **Dr. Ashok Jaiswal**, my research supervisors, for their patient guidance, enthusiastic encouragement and useful critiques of this project work even at the time of global pandemic Covid19 through virtual meetings and telephonic conversations.

My grateful thanks are also extended to my friends of my project group for their help in keeping my progress on schedule.

Finally, I wish to thank my parents for their support and encouragement throughout my study.

INDIAN INSTITUTE OF TECHNOLOGY (BHU), VARANASI

Certificate

This is to certify that RAJAT KUMAR KHANDELWAL(Roll no. 17155069), a student of Department of Mining Engineering- Batch 2017, has successfully completed the B.Tech project "*Preparation of Plugin file for AutoCAD that will generate block model of an ore using Bore hole log data"* under my guidance during the session 2019- 2020.

Date:

Place:

Dr. Ashok Jaiswal
Department of Mining Engineering
Indian Institute of Technology (BHU) Varanasi

Contents

Abstract

Understanding the geological controls on grade distribution throughout a mineral deposit is key to the development of a robust assessment of mining potential. A 2D or 3D computerised model of geology and grade is fundamental to modern Mineral Resource estimation. The Block modelling of an ore is one of the important geological interpretations for mineral reserve estimation and ore simulation. This is carried out by mines simulation software. These software are highly effective and robust for mine planning and mineral exploration and mostly use by mining professionals working in mine companies.

But the main disadvantage is their lack of access and availability for large open source communities of university students, researchers and professors. These software are not freely available online and difficult to use for a beginner. They are not generally used among student communities during their graduations.

So to provide knowledge about Ore block modeling, we have developed a plugin(DLL library file) that runs on AutoCAD software, take bore-hole log data from user and generate a live 3D block model of an ore in AutoCAD by following very simple steps. This block model represents the grade of an ore in different-different location under influential region.

Me and my mentor's research finding indicate that that this plugin can be develop using visual studio .Net core and AutoCAD API. I research a lot on how to code on .Net C# platform, and learn to draw AutoCAD drawing using C# programming language and finally developed a required plugin library file. For grade estimation I researched on various methods and came with a solution that inverse distance method is highly effective and also easily implement by programming. And also find that AutoCAD software is popular, easily and freely available on web, and widely use for design and modeling. It provides simple user interface, 3D space with many rotations that help in easily analyzing of 3D objects and data.

I believe, this project will create great contribution impact in the mine simulation field and mining education and also motivate researchers across the globe to work on this aspect also come with their valuable contribution for mining education and society.

Introduction

Background Information:

Ore reserves can be classified as the following: Massive deposits; deep and very large laterally such that dumping of the waste within the pit is not possible. Stratified vein-type deposits with an inclination steeper than the natural angle of repose of the material so that waste cannot be tipped inside the pit; and Relatively horizontal stratified reserves with a thin/thick covering of overburden.

Open cast mine planning is done by developing the block models and then dividing the deposit into smaller pits which contain both ore and waste blocks which are to be mined in order to reach the pit limit and these operations are done keeping in the mind the overall optimization of the pit and reaching ultimate pit limit design.

Ore body block models are computerized representations of portions of the earth's crust based on geological and geophysical observations made on and below the earth's surface. Ore body models are numerical equivalent of a 3-D geological map complemented by description of physical quantities in the domain of interest.

Geological ore block models are used to generate economical block models by using unit costs and income. With known volume of a block, thickness and grade of ore at each particular block, it becomes possible to convert this information to economical aspect. (Volume *tonnage factor * grade = block reserve.) Economical block models have visual and numerical results; 3D appearances of them give an idea if and where an ore body is rich and how quality changes.

Ore-reserve estimates include the determination of (1) tonnages of ore and (2) average grade or value per ton. Since the grade or content of valuable metal establishes the difference between rock that may and may not be classed as ore, tonnage cannot be estimated without considering the question of grade.

Most operating companies make periodical ore-reserve estimates, usually at least annually, to determine their ore-reserve position as a basis for controlling development and exploration and allocation of funds therefore; for determining deferred, depletion, and depreciation charges per ton; or as a basis for deciding

upon operating policy—expansion or contraction of operations, capital expenditures, and the like.

Mines simulation software:

Mines simulation software is a comprehensive and easy to use exploration and mine design solution, which offers integrated tools for modelling, estimation, design, optimization and scheduling.

They provide users with an in-depth understanding of your projects, so prospective regions can be targeted more accurately, increasing the chance of their project's success. Miners are provided with interactive and easy-to-use modelling, estimation, and design tools to simplify your day-to-day design and production tasks. **Block modelling of an ore with grade estimation** is one of the important features of these software.

Use of Computer Software for ore body modeling:

Generally, size of the database is too bulky to manage studies with hand effort. Thus, numerical algorithms and mathematical approaches necessitate computer applications to overcome huge computational time and processes. Currently, many computer aided systems and software serve for geological modeling. The accuracy and speed of computers enable evaluation of various scenarios within reasonably short times. Computer programs are ready for ore body modeling after building a healthy database structure. Visual appearance of geological body is supported by numerical data such as ore reserve amount and quality composition, which are vital parameters for mine design and scheduling. Thus, computer systems are very important for mining and geological studies.

Objective

To prepare Block model Plugin that's runs in AutoCAD platform and using single command fetch Bore Hole log datasets from CSV files and generate:-

- A Bore hole log depict by a line to particular depth, inclination with their title.
- A block model of an ore around bore log that represent estimated grades concentration at different-different regions using colors shapes using inverse square method.

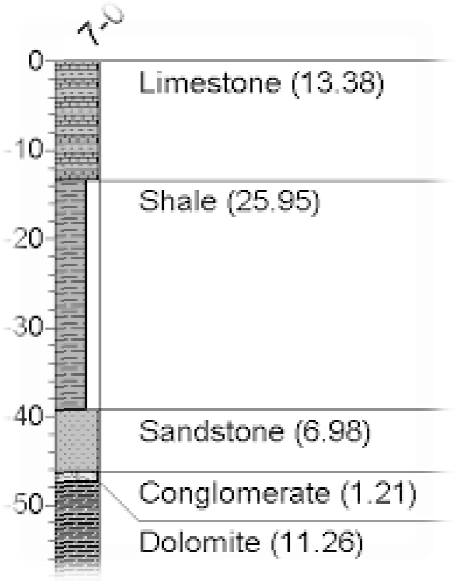
All these computations are dynamic and customized according to the inputs given by user following very simple dialog box instruction.

Literaures and Theory

1. BORE HOLE LOG :

Borehole logging is the practice of making a detailed record (a well log) of the geologic formations penetrated by a borehole. The log may be based either on visual inspection of samples brought to the surface (geological logs) or on physical measurements made by instruments lowered into the hole (geophysical logs). Some types of geophysical well logs can be done during any phase of a well's history: drilling, completing, producing, or abandoning. Well logging is performed in boreholes drilled for the oil and gas, groundwater, mineral and geothermal exploration, as well as part of environmental and geotechnical studies.

BORE HOLE DATA: The interpretations stored in the Borehole Geology database are made from borehole logs that show the geology encountered at depth within each borehole. In many cases, these logs were created by the geotechnical Companies responsible for drilling the holes, and supplied to the BGS. In other cases, the logs may have been made by our own geologists, either at the time of drilling, or subsequently from core samples.



Fig(1.1)

2. Mineral resource estimation:

Resource estimation is used to determine and define the ore tonnage and grade of a geological deposit, from the developed block model. There are different estimation methods (see below) used for different scenarios dependent upon the ore boundaries, geological deposit geometry, grade variability and the amount of time and money available. A typical resource estimation involves the construction of a geological and resource model with data from various sources.

3. Geological Block Models (by Erarslan, 2001):

Geological Block Models are used to generate economical block models by using unit costs and income. As volume of a block, thickness and grade of ore at each particular block is known, then it becomes possible to convert this information to economical aspect. Multiplication of volume, tonnage factor and grade give block reserve. Economical block models have visual and numerical results. 3D appearances of them give an idea where ore body is rich and how quality changes.

4. Ore body Modeling : An Integrated Geological-Geostatistical Approach (by Indranil Roy and B. C. Sarkar)

They concluded that ore body modeling is a reflection of geological and geometrical reality of an ore deposit. Geologists and mining engineers can benefit from such an integrated modeling approach by honoring the deposit geology, understanding the statistical distribution and emphasizing the spatial continuity studies. The model can act as principal guides for development of mineral inventory model and grade-tonnage curves which ultimately can lead to a value model in terms of economic extraction of the ore body.

5. Block Model Estimation:

Once the geological modeling is completed, the geological envelopes are divided into block models. Subsequently, the estimation of these blocks is done from "composites" that are point measures of the grade of ore in the rock. Several different mathematical methods can be used to do the estimation depending on the desired degree of precision, quality and quantity of data and of their nature:

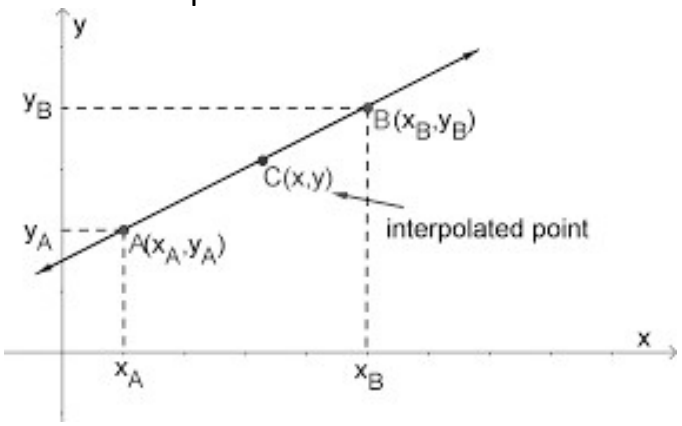
- Nearest Neighbour Method
- Inverse Distance Weighting Method

- Kriging Method

In this project we are going to use **Inverse Square Distance Weighting Method** for ore grade reserve estimation.

6. Interpolation:

Interpolation is the process of estimating unknown values that fall between known values. In this example, a straight line passes through two points of known value. You can estimate the point of unknown value because it appears to be midway between the other two points.



It is used to create continuous surfaces of elevation, rainfall, temperature, chemical dispersion, or other spatially- based phenomena. It is more realistic representation of a geographic phenomenon. It is widely use in ore grade estimation.

7. Ore grade estimation using Inverse Distance Weighting Method (IDW):

The name given to this type of method was motivated by the **weighted average** applied, since it resorts to the inverse of the distance to each known point ("amount of proximity") when assigning weights.
The simplest weighting function in common usage is based upon the inverse of the distance of the sample from the point to be estimated, usually raised to the second power, although higher or lower powers may be useful.

Samples closer to the point of interest get a higher weighting than samples farther away. Samples closer to the point of estimation are more likely to be

similar in grade. Such inverse distance techniques introduce issues such as sample search and declustering decisions, and cater for the estimation of blocks of a defined size, in addition to point estimates.

Inverse Distance Method (IDW) is given by:

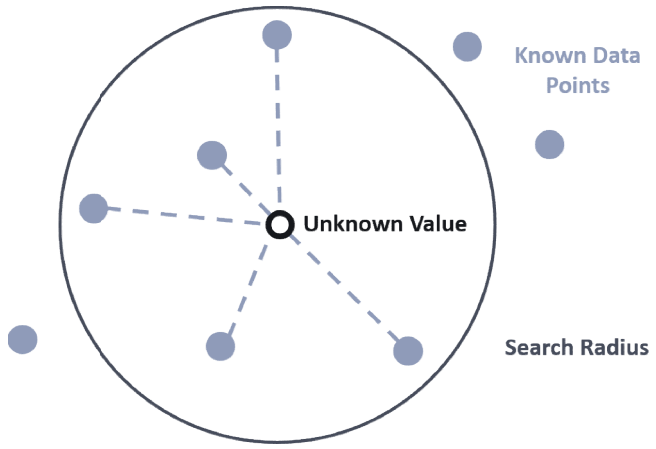
$$Z_j = \frac{\sum_i \frac{Z_i}{d^n_{ij}}}{\sum_i \frac{1}{d^n_{ij}}}$$

Z_i is value of known point
D_{ij} is distance to known point
Z_j is the unknown point
n is a user selected exponent
(often 1,2 or 3)

Any number of points may be used up to all points in the sample; typically 3 or more

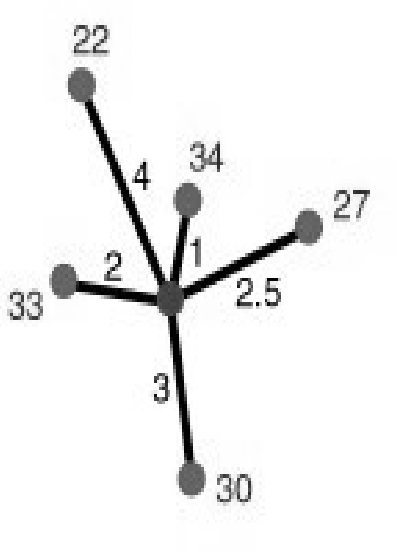
For inverse square distance method, the value of n = 2 is taken.

Search Radius: The characteristics of the interpolated surface can also be controlled by applying a search radius (fixed or variable), which limits the number of input points that can be used for calculating each interpolated cell. You limit the number of points for each cell's calculation to improve processing speeds. You can also limit the number of points, because points far from the cell location where the prediction is being made may have no spatial correlation.



Example:

Here known values associated with different points are given. We have to find the value associated by mid interior point using inverse square distance method.



$$Z(x) = \frac{\sum w_i z_i}{\sum w_i} = \frac{\frac{34}{1^2} + \frac{33}{2^2} + \frac{27}{2.5^2} + \frac{30}{3^2} + \frac{22}{4^2}}{\frac{1}{1^2} + \frac{1}{2^2} + \frac{1}{2.5^2} + \frac{1}{3^2} + \frac{1}{4^2}} = 32.38$$

This method is highly effective in grade calculation of unknown points given grades associated with other known points and there distance from these points.

Advantages of Inverse Distance Weight Method:

- Computationally simple.
- Exponent gives flexibility. The same estimation procedure can be used to create very smooth estimates (like a moving average) or very variable estimates (like nearest neighbor).

Disadvantage:

- Preferential sampling makes estimates unreliable.
- Requires decision on which sample to use.
- Extremes create large halos of great estimates.
- Choice of exponent introduces arbitrariness.

8. Error evaluation for IDW mineral reserve estimates (by Li Zhanglin, Wang Ping)

They concluded that Inverse distance weighting (IDW) is a significant and well known mineral reserve estimation method. In their proposed method, errors of IDW estimated mineral reserves can be evaluated by comparing with actual values in a developed mine, estimates resulted from other methods or other parameters in the framework of cross validation. Based on the analysis of the basic principle of IDW mineral reserve estimation method, the complete process in the implementation of IDW mineral reserve estimation and error estimation has be discussed and implemented in computer software. According to compare the IDW mineral reserve estimates with the actual developed situations, ore tonnage and metal tonnage estimates have shown clearly clear stability and robustness, which can strongly prove the validity and practicability of the proposed method. Thus, both theoretical analysis and practical application can show the validity and practicability of the proposed method.

Approach and Methodology

1. First of all it all started after the brief introduction about the project and concepts to work on with my mentor Dr. Ashok Jaiswal sir.

2. I started my research work on Bore-Hole, Grade estimation method, and Block model of an ore by bore-hole log data.

3. After that, I started on software part by getting familiarity with AutoCAD and Microsoft visual studio .Net core platform and learned to write programs to draw simple geometries like lines, circles etc.

4. After that I started plugin(DLL library file) development coding work in .Net Core platform.

5. Create a main class file, write standard C# starting codes and gave a AutoCAD plugin command (**BlockModel**) and attach Autocad api reference files to visual studio IDE. This command is used to load plugin library file in Autocad.

6. After that I created a window form, designed it by adding two buttons, file dialog setup and gridView window, and attached it with **BlockModel** command. So that when user run this command, a window form displayed on user screen which ask user to upload Bore hole data.

7. The provided borehole data must be in .csv format and must be in standardized format.

8. After upload step, the window form give a preview of bore-log data data so that user can verify it.

9. After that we started backend coding of this window form. We followed functional programming. Implement methods to two buttons.

10. The first task of project was designing bore-log according to data taken from csv fille that was given by user earlier. Many lines of C# code was written under Form1.cs file that included user-oriented programming that implement logic consisted of loops and condition statements devised by us.

11. Now test plugin that it will load bore-hole view or not. After successful testing, I started next phase project i.e. grade calculation at different surrounding points by Inverse Square Distance method and generation of block model around them. I created another window form, designed it by adding one buttons, and GridView window, that display relevant known grade points with their x, y, z values, when user completed 1st form.

12. It started by first stored x, y, z co-ordinates of midpoint of ore region of different-different bore-hole logs along with their grade and displaying them in another form. I found min and max values of x, y, and z co-ordinates. Then I setup 3 For loops that draw and store coordinates of random points about 3 units away bounded by min and max values of x, y, and z co-ordinates.

13. Inside body deepest loop, I declare an **array**, **calculate_relev_distance** function, and **calculate_grade** function.

14. The **calculate_relev_distance** function calculate distance of each known points from this unknown point, checked if these points were inside search radius(i.e. was given by user in further development), if yes then store the distance along with known grade in an **array.**

15. The **calculate_grade** function take this array as an argument, calculate distance of unknown point using **Inverse Distance Square Method** of grade estimation and return it to interpolation (main) function.

16. If grade calculated is greater than zero then we called another function to draw a solid block having centroid equal to co-ordinates of calculated points having calculated grade value and give color to the block according to magnitude of calculated grade interval.

17. Finally after successful build and testing our **BlockModel plugin file** was ready and successfully loaded and implemented in AutoCAD.

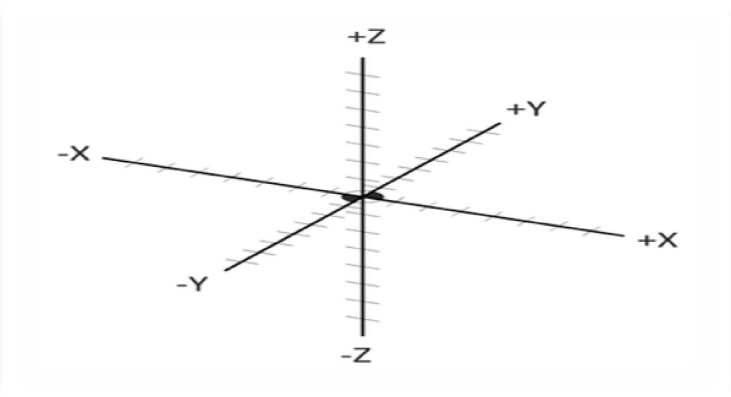
18. In this project future develop involve more systematic UI design and giving all complete user specified features.

Bore Hole data CSV file format

A1 | Name of BoreHole

	A	B	C	D	E	F	G	H	I
1	Name of BoreHole	Length	Rock type	Angle_alpha	Angle_beta	Grade	Starting_x	Starting_y	Starting_z
2	BH1	15	sand	0.349066	0.0872665	0	0	0	0
3	BH1	10	ore	0.349066	0.0872665	0.6			
4	BH1	5	ore	0.349066	0.0872665	0.7			
5	BH1	15	mud	0.349066	0.0872665	0			
6	BH2	15	sand	0.436332	0.0872665	0	5	0	0
7	BH2	10	ore	0.436332	0.0872665	0.7			

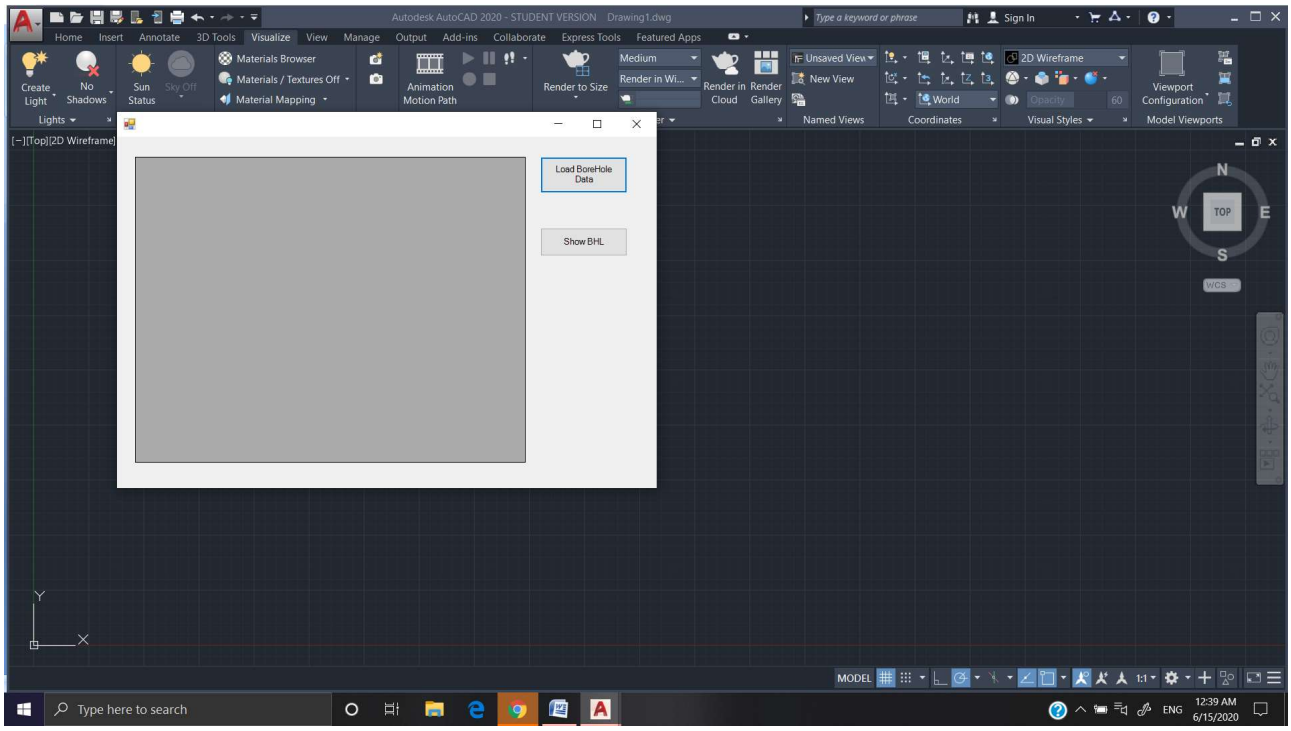
- First column A Bore-Hole log name/code.
- B column represent length of log core.
- C column the rock typre associated with log core.
- D column, here we have to input angle of bore log core with z-axis i.e. vertical depth.
- E column, here we have to input angle of bore log core with x-axis i.e. horizontal length along the ground surface.
- F represent grade of ore associated with respective log core.
- And at last Column G, H, and I represent co-ordinates of collar at ground surface below which we drilled a bore-hole log.



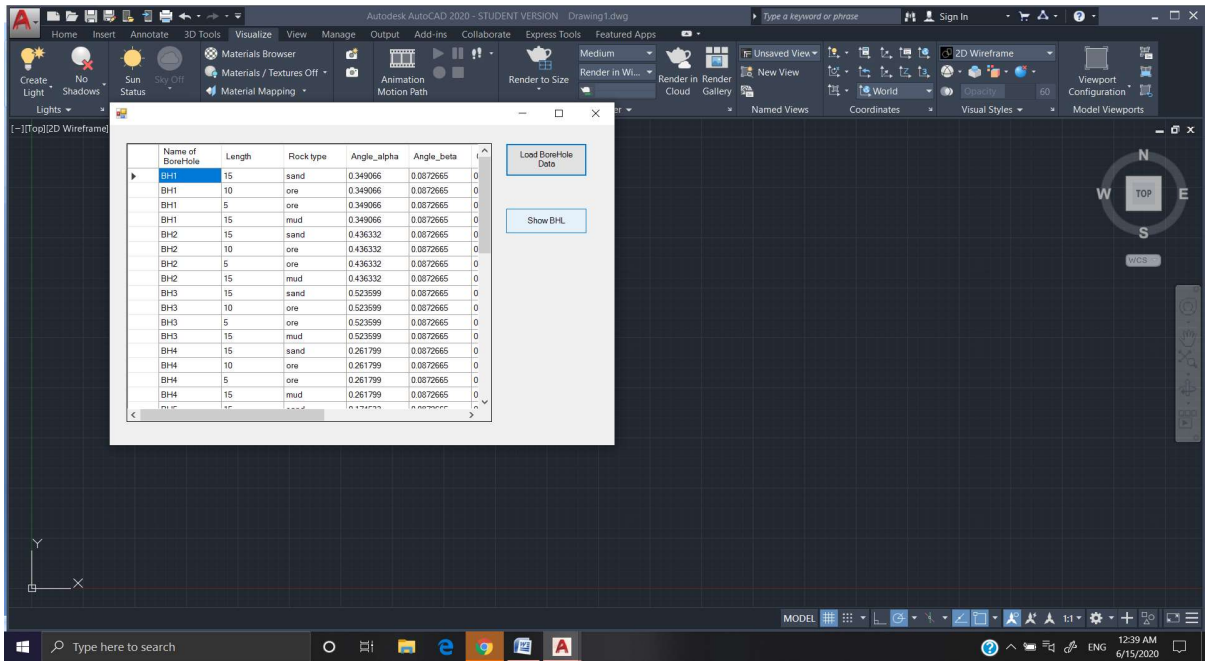
Steps to run BlockModel Plugin

To use the requird Block Model plugin file, please follow these steps:-

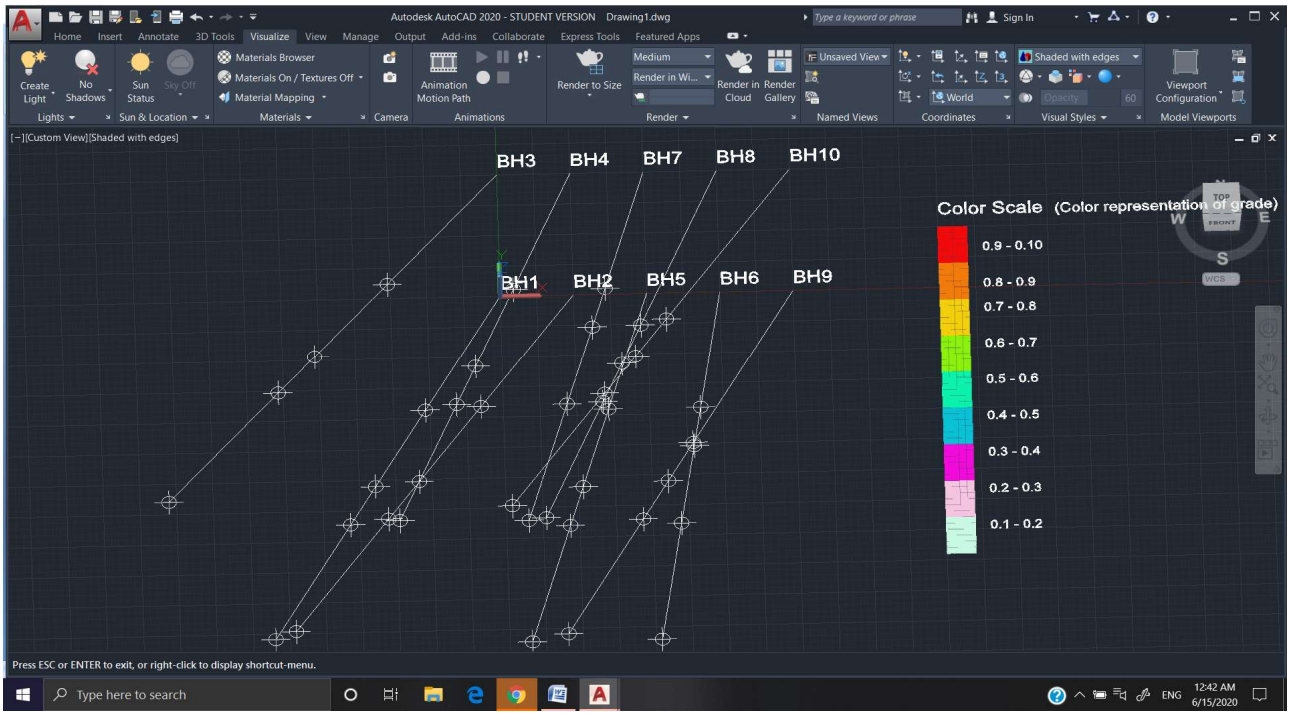
- Collect and Process Bore-Hole log data and save them in CSV (Common separated file) format.
- Run **AutoCAD** software. Click on Start Drawing option under Getting Started menu.
- After that type and run **NETLOAD** command on drawing dashboard window.
- Select **BlockModel plugin** (DLL file) and click **Open.**
- After that type and run **blockModel** command on drawing dashboard window. A dialog box will display.



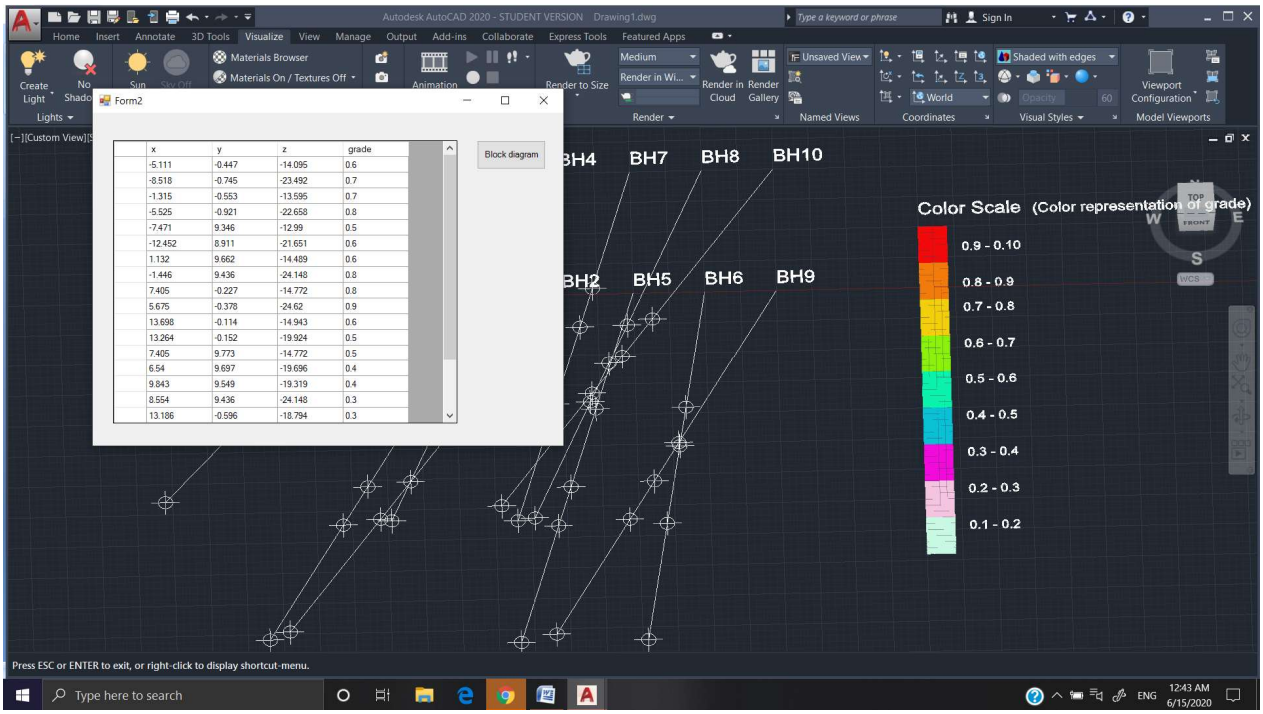
- Click on **Load BHL button** and select Bore-Hole log data csv file. A BHL data will display on left side of dialog box.



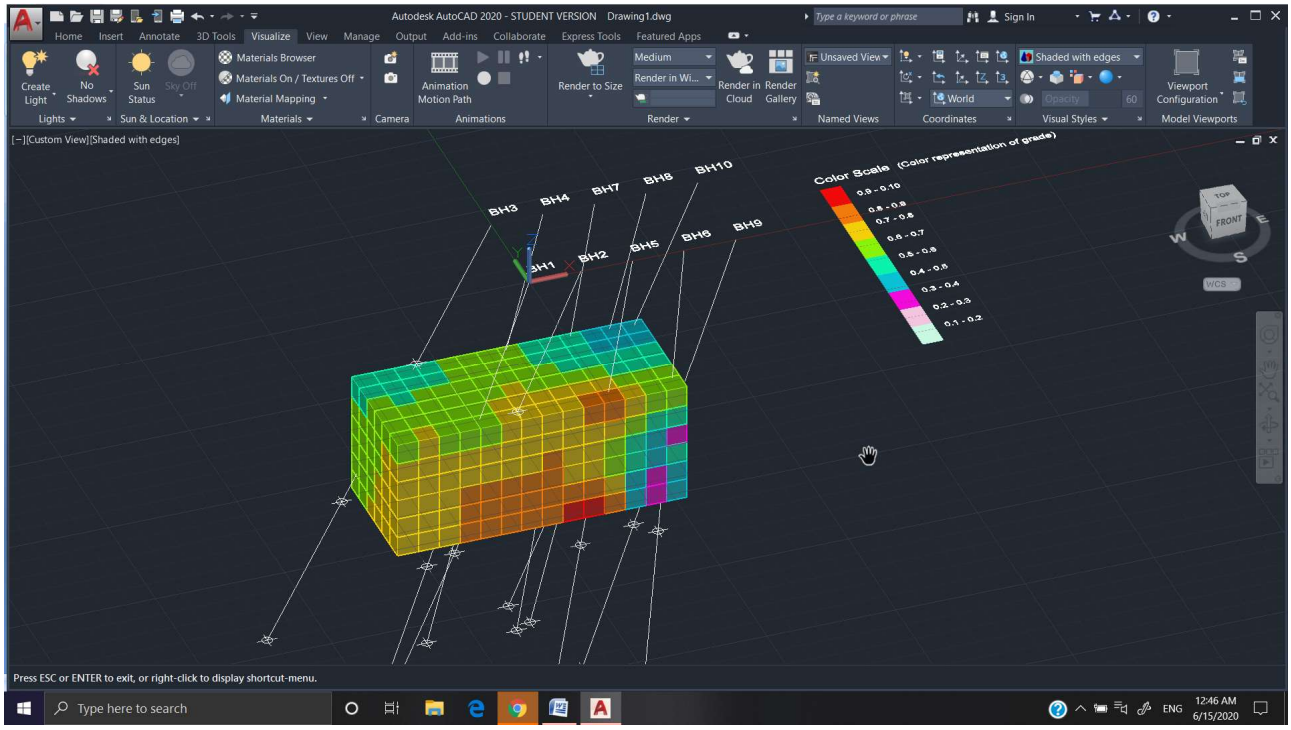
- Then click on Show **BHL button**. Now AutoCAD dashboard is displaying graphical **3D line representation of Bore-Log** in space along with their name and u/g material changing points with depth.



- And also another Dialog box display. The left GridView window is showing x, y, z co-ordinates of known relevance ore points with their grade value and right side contain a Generate Block Model button.



- Finally click on this button. A 3D block model representation of ore surrounding will display on screen. The different colors of wireframe block represent different magnitude of ore grade at that points.



Advantage of Block-Model Plugin project

The main advantage of this plugin is that it is very cost effective and can be used for learning purpose and at small scale modelling because Autocad software easily available on internet free of cost and it is widely popular for modelling purpose but mining software like Surpac are very expensive and not easily free available on internet.

Result and Summary

The Project on “Creation of AutoCAD Plugin to draw Block model of an ore based on Bore Hole log data” is completed successfully.

The final outcome of project is creation of single ClassLibrary2.dll file.

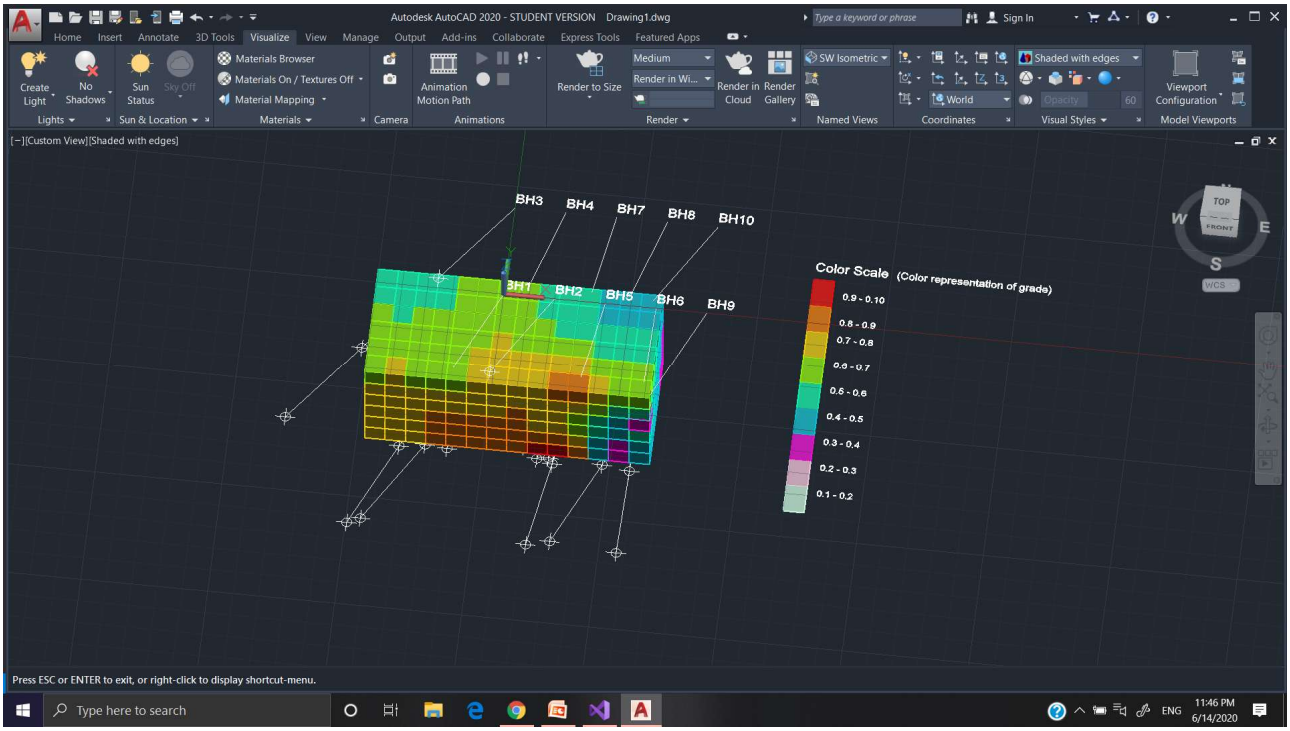
When it run on AutoCad after NETLOAD command, it perform its function as describe above and create 3D Block Grid Model of ore easily.

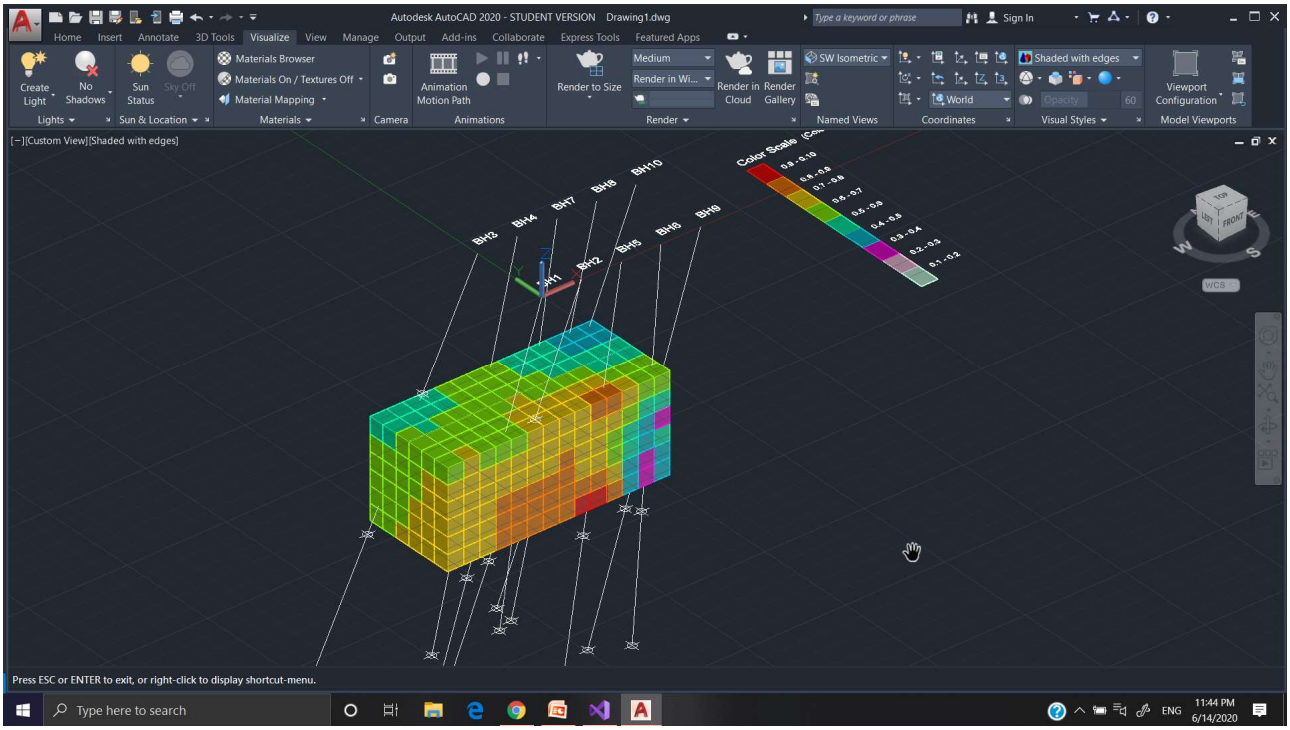
It ensure complete dynamic and data adaptive programming logic i.e. when you give any Bore-hole log data as per prescribed csv format, it will successfully create block model according to the given data.

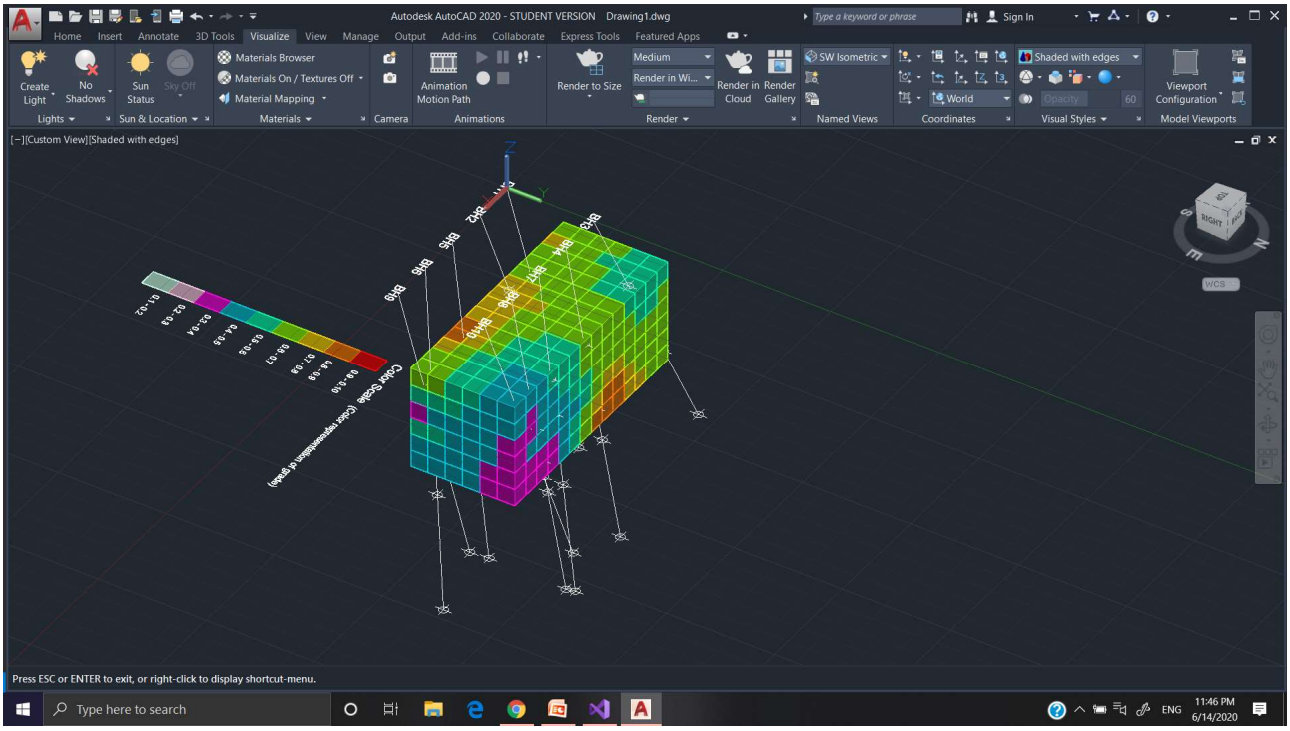
For Block Modeling, A high end Personal computer is recommended; otherwise it can take much time to render the graphics on AutoCAD.

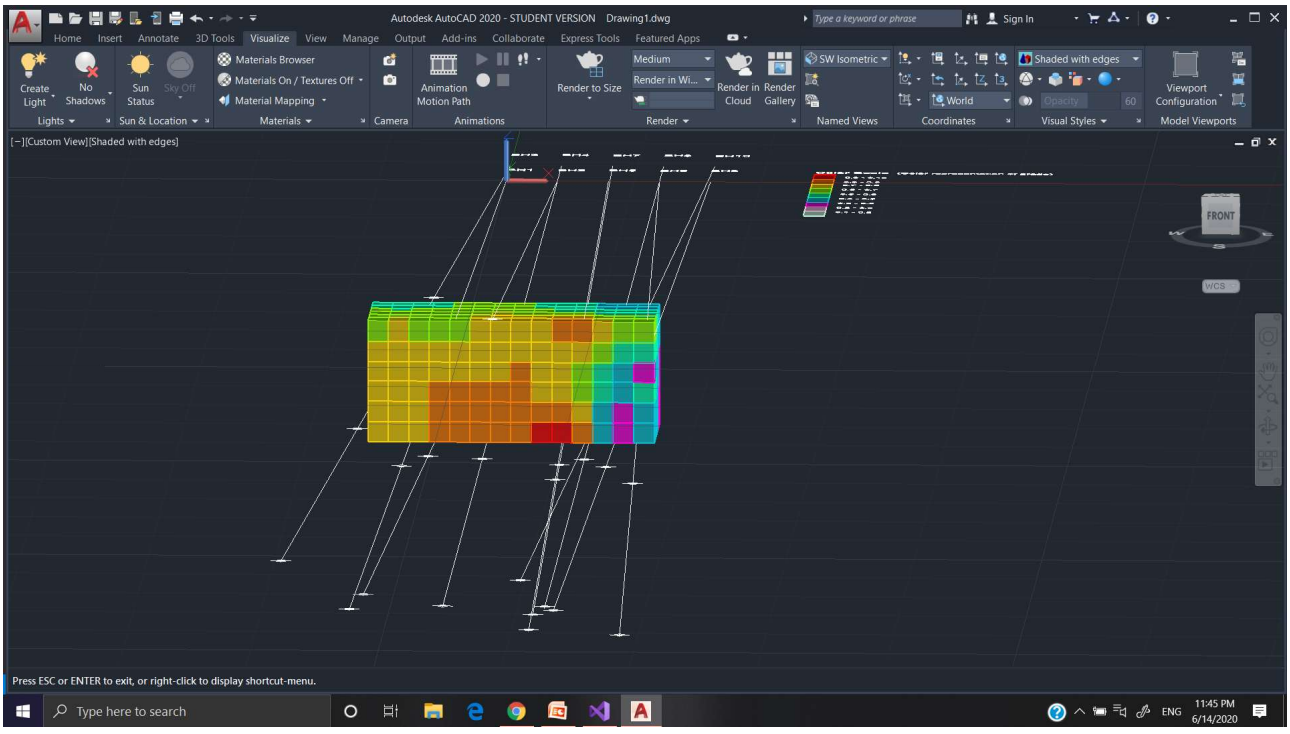
Example Screenshots

Below are some screenshots for the Final Block Model images at different views generated by BlockModel Plugin:






Autodesk AutoCAD 2020 - STUDENT VERSION Drawing1.dwg
[−][Custom View][Shaded with edges]
Press ESC or ENTER to exit, or right-click to display shortcut-menu.
Type here to search
11:46 PM
6/14/2020


Autodesk AutoCAD 2020 - STUDENT VERSION Drawing1.dwg
[−][Custom View][Shaded with edges]
FRONT
Press ESC or ENTER to exit, or right-click to display shortcut-menu.
Type here to search
11:45 PM
6/14/2020

Further Work

Further works include getting more Bore-hole log data that help in accurate grade prediction and smoother grid modeling.

Give option to user to choose region under which he want to view block model and also decide size of blocks for grid modeling.

And implementation of specific features that give model more realistic and robust feature.

References

Related to Theory and concept:

[1] https://en.wikipedia.org/wiki/Mineral_resource_estimation
[2] Julian Poniewierski , January 2019. BLOCK MODEL KNOWLEDGE FOR MINING ENGINEERS-AN INTRODUCTION.
[3] https://gisgeography.com/inverse-distance-weighting-idw-interpolation
[4] Kaan Erarslan , Feb 2012. Computer Aided Ore Body Modelling and Mine Valuation.
[5] Philip H. Odd, Robert F. Bmullad and Carl P. Lathan, Borehole logging methods for exploration.
[6] https://en.wikipedia.org/wiki/Interpolation
[7] Zhongxue Li ; Xin Li ; Cuiping Li ; Zhiguo Cao. Improvement on inverse distance weighted interpolation for ore reserve estimation.
[8] Indranil Roy and B. C. Sarkar , Ore body Modeling : An Integrated Geological-Geostatistical Approach

Related to .NET Programming with AutoCad API.:

[9] Stephen Preton Autodesk DevTech: AutoCAD .NET Developers Guide
[10] https://through-the-interface.typepad.com/through_the_interface/
[11] http://docs.autodesk.com/ACD/2010/ENU/AutoCAD%20.NET%20Developer% 27s%20Guide/index.html
[12] https://www.autodesk.com/autodesk-university/class/Introduction-AutoCAD-Softwares-NET-API-Using-C-NET-2018
[13] https://www.keanw.com/autocad_net/
[14] https://forums.autodesk.com/
[15] https://www.geogebra.org/3d?lang=en

Appendix (.NET C# Source Code)

Overall project folder contain three code files namely **Class1.cs, Form1.cs, and Form2.cs**

Class1.cs (Main class file)

```
using Autodesk.AutoCAD.Runtime;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace bore_hole1
{
   public class Class1
   {
      [CommandMethod("BlockModel")]
      public void BHL()
      {
         var BHL = new ClassLibrary2.Form1();
         BHL.Show();
      }
   }
}
```

Form1.cs // Contain Logic for generating Bore-Hole line model and color index

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
```

```
using Autodesk.AutoCAD.ApplicationServices;
using Autodesk.AutoCAD.DatabaseServices;
using Autodesk.AutoCAD.Geometry;
using System.Data.OleDb;
using DataTable = System.Data.DataTable;
using System.IO;
using Autodesk.AutoCAD.Colors;
using Autodesk.AutoCAD.EditorInput;

namespace ClassLibrary2
{
   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
      }
      private void button3_Click_1(object sender, EventArgs e)
      {
         if (openFileDialog1.ShowDialog() == DialogResult.OK)
         {
            string path;
            path = openFileDialog1.FileName;
            string ext = Path.GetExtension(path);
            if (ext == ".csv")
            {
               DataTable my_data = new DataTable();
               string[] raw_text = File.ReadAllLines(path);
               string[] data_col = null;
               int x = 0;

               foreach (string test_line in raw_text)
               {
                  //MessageBox.Show(test_line);
                  data_col = test_line.Split(',');
                  if (x == 0)
                  {
                     //header
                     for (int i1 = 0; i1 < data_col.Count(); i1++)
```

```
                {
                    my_data.Columns.Add(data_col[i1]);
                    //dataGridView1.Columns.Add(data_col[i1]);
                }
                x++;
            }
            else
            {
                //data
                my_data.Rows.Add(data_col);
            }

        }
        dataGridView2.DataSource = my_data;
        int a, i;
        a = dataGridView2.Columns.Count;
        for (i = 0; i < a; i++)
        {
            dataGridView2.Columns[i].SortMode = DataGridViewColumnSortMode.NotSortable;

        }
    }
  }
}
private void dataGridView2_CellContentClick(object sender, DataGridViewCellEventArgs e)
{
}

private void button4_Click_1(object sender, EventArgs e)
{
  //this.draw();
  this.draw_line();
  this.draw_scale();
  this.draw_surface();
  this.Close();
}

void draw_surface()
{
```

```
        Point3d p0 = new Point3d(-10,30, 0);
        Point3d p1 = new Point3d(40, 30, 0);
        Point3d p2 = new Point3d(-10, 0, 0);
        Point3d p3 = new Point3d(40, 0, 0);

        Document acDoc =
Autodesk.AutoCAD.ApplicationServices.Application.DocumentManager.MdiActiveDocument;
        Autodesk.AutoCAD.DatabaseServices.Database acCurDb = acDoc.Database;
        // Start a transaction
        using (DocumentLock docLock = acDoc.LockDocument())
        {
          using (Transaction acTrans = acCurDb.TransactionManager.StartTransaction())
          {

            BlockTable acBlkTbl;
            acBlkTbl = acTrans.GetObject(acCurDb.BlockTableId,
            OpenMode.ForRead) as BlockTable;
            // Open the Block table record Model space for write
            BlockTableRecord acBlkTblRec;
            acBlkTblRec = acTrans.GetObject(acBlkTbl[BlockTableRecord.ModelSpace],
            OpenMode.ForWrite) as BlockTableRecord;

            Solid ac2DSolidSqr1 = new Solid(p0, p1, p2, p3);
            acBlkTblRec.AppendEntity(ac2DSolidSqr1);
            acTrans.AddNewlyCreatedDBObject(ac2DSolidSqr1, true);
          }
        }

    }
    void draw_scale()
    {
      //For color of cubes according to thier grade values....
      Autodesk.AutoCAD.Colors.Color[] acColors = new Autodesk.AutoCAD.Colors.Color[9];
      acColors[0] = Autodesk.AutoCAD.Colors.Color.FromRgb(201, 248, 227);
      acColors[1] = Autodesk.AutoCAD.Colors.Color.FromRgb(245, 196, 224);
      acColors[2] = Autodesk.AutoCAD.Colors.Color.FromRgb(243, 11, 220);
      acColors[3] = Autodesk.AutoCAD.Colors.Color.FromRgb(11, 193, 213);
      acColors[4] = Autodesk.AutoCAD.Colors.Color.FromRgb(11, 243, 173);
      acColors[5] = Autodesk.AutoCAD.Colors.Color.FromRgb(139, 243, 11);
```

```
        acColors[6] = Autodesk.AutoCAD.Colors.Color.FromRgb(243, 208, 11);
        acColors[7] = Autodesk.AutoCAD.Colors.Color.FromRgb(243, 123, 11);
        acColors[8] = Autodesk.AutoCAD.Colors.Color.FromRgb(243, 11, 11);

        int x1 = 20;
        int y1 = 5;
        int z1 = 0;

        int x2 = x1 + 2;
        int y2 = 5;
        int z2 = 0;
        for (int i = 0; i < 9; i++)
        {
           int d = 3;
           int x3 = x1;
           int y3 = y1 - d;
           int z3 = 0;

           int x4 = x2;
           int y4 = y3;
           int z4 = 0;

           Point3d p0 = new Point3d(x1, y1, z1);
           Point3d p1 = new Point3d(x2, y2, z2);
           Point3d p2 = new Point3d(x3, y3, z3);
           Point3d p3 = new Point3d(x4, y4, z4);

           Document acDoc =
Autodesk.AutoCAD.ApplicationServices.Application.DocumentManager.MdiActiveDocument;
           Autodesk.AutoCAD.DatabaseServices.Database acCurDb = acDoc.Database;
           Editor ed = acDoc.Editor;
           // Start a transaction
           using (DocumentLock docLock = acDoc.LockDocument())
           {
              using (Transaction acTrans = acCurDb.TransactionManager.StartTransaction())
              {

                 BlockTable acBlkTbl;
                 acBlkTbl = acTrans.GetObject(acCurDb.BlockTableId,
```

```
OpenMode.ForRead) as BlockTable;
// Open the Block table record Model space for write
BlockTableRecord acBlkTblRec;
acBlkTblRec = acTrans.GetObject(acBlkTbl[BlockTableRecord.ModelSpace],
OpenMode.ForWrite) as BlockTableRecord;

using (Solid ac2DSolidSqr1 = new Solid(p0, p1, p2, p3))
{
   //ac2DSolidSqr1.Color = acColors[i];
   Autodesk.AutoCAD.Colors.Color curcolor = new Autodesk.AutoCAD.Colors.Color();
   curcolor = acColors[8-i];
   ac2DSolidSqr1.Color = curcolor;

   // Add the new object to the block table record and the transaction
   acBlkTblRec.AppendEntity(ac2DSolidSqr1);
   acTrans.AddNewlyCreatedDBObject(ac2DSolidSqr1, true);
}

// Create a single-line text object
using (DBText acText = new DBText())
{
   acText.Position = new Point3d(x4 + 1, (y1+y3)/2, 0);
   acText.Height = 0.7;
   String grade = "0." + (10 - i-1) + " - " + "0." +(10-i);
   acText.TextString = grade;

   acBlkTblRec.AppendEntity(acText);
   acTrans.AddNewlyCreatedDBObject(acText, true);
}
y1 = y3;
y2 = y1;
x1 = x3;
x2 = x4;
z1 = z3;
z2 = z4;

if (i == 5)
{
   // Create a single-line text object
```

```
                using (DBText acText = new DBText())
                {
                    acText.Position = new Point3d(20, 6, 0);
                    acText.Height = 1;
                    acText.TextString = "Color Scale";
                    acBlkTblRec.AppendEntity(acText);
                    acTrans.AddNewlyCreatedDBObject(acText, true);
                }
                using (DBText acText = new DBText())
                {
                    acText.Position = new Point3d(27, 6, 0);
                    acText.Height = 0.8;
                    acText.TextString = "(Color representation of grade)";
                    acBlkTblRec.AppendEntity(acText);
                    acTrans.AddNewlyCreatedDBObject(acText, true);
                }
            }

            // Changing Visual style
            DBDictionary dict =
(DBDictionary)acTrans.GetObject(acCurDb.VisualStyleDictionaryId, OpenMode.ForRead);
            ViewportTable vt = (ViewportTable)acTrans.GetObject(acCurDb.ViewportTableId,
OpenMode.ForRead);
            ViewportTableRecord vtr = (ViewportTableRecord)acTrans.GetObject(vt["*Active"],
OpenMode.ForWrite);
            vtr.VisualStyleId = dict.GetAt("Shaded");

            acTrans.Commit();
          }
          ed.UpdateTiledViewportsFromDatabase();
        }
      }

    }

    void draw_line()
    {
      int i;
      double xi = 0;
```

```
        double yi = 0;
        double x1 = 0;
        double y1 = 0;
        double z1 = 0;
        int L = 0;

        string prev = Convert.ToString(dataGridView2.Rows[0].Cells[1].Value);
        var BHL_data = new ClassLibrary2.Form2();
        BHL_data.Show();
        for (i = 0; i < dataGridView2.Rows.Count - 1; i++)
        {
           int count = 0;
           int len = Convert.ToInt32(dataGridView2.Rows[i].Cells[1].Value);
           double angle_alpha = Convert.ToDouble(dataGridView2.Rows[i].Cells[3].Value);    // by z-
axis
           double angle_beta = Convert.ToDouble(dataGridView2.Rows[i].Cells[4].Value);     // by x-
axis
           Point3d p0 = new Point3d(x1, y1, z1);

           L =  len;
           double x2 = -L*Math.Sin(angle_alpha)*Math.Cos(angle_beta);
           double y2 = -L * Math.Sin(angle_alpha) * Math.Sin(angle_beta); ;
           double z2 = -L*Math.Cos(angle_alpha);
           Point3d p1 = new Point3d(x2+x1, y2+y1, z2+z1);

           if (Convert.ToString(dataGridView2.Rows[i].Cells[2].Value) == "ore")
           {
              double grade = Convert.ToDouble(dataGridView2.Rows[i].Cells[5].Value);        //grade
              string[] row = new string[] {x1.ToString("0.###"),y1.ToString("0.###"),
z1.ToString("0.###"), grade.ToString("0.###") } ;

              BHL_data.add(row);
           }

           Document acDoc =
Autodesk.AutoCAD.ApplicationServices.Application.DocumentManager.MdiActiveDocument;
           Autodesk.AutoCAD.DatabaseServices.Database acCurDb = acDoc.Database;
           // Start a transaction
           using (DocumentLock docLock = acDoc.LockDocument())
```

```
{
   using (Transaction acTrans = acCurDb.TransactionManager.StartTransaction())
   {
      BlockTable acBlkTbl;
      acBlkTbl = acTrans.GetObject(acCurDb.BlockTableId,
      OpenMode.ForRead) as BlockTable;
      // Open the Block table record Model space for write
      BlockTableRecord acBlkTblRec;
      acBlkTblRec = acTrans.GetObject(acBlkTbl[BlockTableRecord.ModelSpace],
      OpenMode.ForWrite) as BlockTableRecord;

      Line ln = new Line(p0, p1);
      ln.ColorIndex = 7;
      // set the lineweight for it
      ln.LineWeight = LineWeight.LineWeight080;
      acCurDb.LineWeightDisplay = true;

      acBlkTblRec.AppendEntity(ln);
      acTrans.AddNewlyCreatedDBObject(ln, true);

      // Create a point at (4, 3, 0) in Model space
      using (DBPoint acPoint = new DBPoint(p1))
      {
         // Add the new object to the block table record and the transaction
         acBlkTblRec.AppendEntity(acPoint);
         acTrans.AddNewlyCreatedDBObject(acPoint, true);
      }
      // Set the style for all point objects in the drawing
      acCurDb.Pdmode = 34;
      acCurDb.Pdsize = 1;

      x1 = x1 + x2;
      y1 = y1 + y2;
      z1 = z1 + z2;

      if (Convert.ToString(dataGridView2.Rows[i].Cells[0].Value) !=
Convert.ToString(dataGridView2.Rows[i + 1].Cells[0].Value))
      {
```

```
                //  x1 = xi + 5;
                //  xi = x1;
                  //  y1 = 0;
                  //   z1 = 0;
                  x1 = Convert.ToDouble(dataGridView2.Rows[i+1].Cells[6].Value);
                  y1 = Convert.ToDouble(dataGridView2.Rows[i+1].Cells[7].Value);
                  z1 = Convert.ToDouble(dataGridView2.Rows[i+1].Cells[8].Value);
                  count = count + 1;
                  L = 0;
                }

                if (count > 0)
                {
                  // Create a single-line text object
                  using (DBText acText = new DBText())
                  {
                    acText.Position = new Point3d(x: xi, y: yi, z: 1);
                    xi = x1;
                    yi = y1;
                    acText.Height = 1;
   acText.TextString = Convert.ToString(dataGridView2.Rows[i].Cells[0].Value);

                    acBlkTblRec.AppendEntity(acText);
                    acTrans.AddNewlyCreatedDBObject(acText, true);
                  }
                  // Save the new object to the database
                  // acTrans.Commit();
                }
                acTrans.Commit();
              }
            }
          }
        }
      }
    }
```

Form2.cs: //Contain logic for inverse square distance method and block generation at relevant ore area

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using Autodesk.AutoCAD.ApplicationServices;
using Autodesk.AutoCAD.DatabaseServices;
using Autodesk.AutoCAD.Geometry;
using System.Data.OleDb;
using DataTable = System.Data.DataTable;
using System.IO;
using Autodesk.AutoCAD.Colors;
using Autodesk.AutoCAD.EditorInput;

namespace ClassLibrary2
{
   public partial class Form2 : Form
   {
      public Form2()
      {
         InitializeComponent();
         dataGridView1.ColumnCount = 4;
         dataGridView1.Columns[0].Name = "x";
         dataGridView1.Columns[1].Name = "y";
         dataGridView1.Columns[2].Name = "z";
         dataGridView1.Columns[3].Name = "grade";

      }
      double x_min = 1000000, x_max = -1000000;
      double y_min = 1000000, y_max = -1000000;
      double z_min = 1000000, z_max = -1000000;
      int i = 0;
```

```
public void add(string[] row)
{
   dataGridView1.Rows.Add(row);
   if(x_min > Convert.ToDouble(dataGridView1.Rows[i].Cells[0].Value))
   {
      x_min = Convert.ToDouble(dataGridView1.Rows[i].Cells[0].Value);
   }
   if (x_max < Convert.ToDouble(dataGridView1.Rows[i].Cells[0].Value))
   {
      x_max = Convert.ToDouble(dataGridView1.Rows[i].Cells[0].Value);
   }
   if (y_min > Convert.ToDouble(dataGridView1.Rows[i].Cells[1].Value))
   {
      y_min = Convert.ToDouble(dataGridView1.Rows[i].Cells[1].Value);
   }
   if (y_max < Convert.ToDouble(dataGridView1.Rows[i].Cells[1].Value))
   {
      y_max = Convert.ToDouble(dataGridView1.Rows[i].Cells[1].Value);
   }
   if (z_min > Convert.ToDouble(dataGridView1.Rows[i].Cells[2].Value))
   {
      z_min = Convert.ToDouble(dataGridView1.Rows[i].Cells[2].Value);
   }
   if (z_max < Convert.ToDouble(dataGridView1.Rows[i].Cells[2].Value))
   {
      z_max = Convert.ToDouble(dataGridView1.Rows[i].Cells[2].Value);
   }
   i = i + 1;
}

private void button1_Click(object sender, EventArgs e)
{
   this.interpolation();
  // this.drawCube();
   this.Close();
}

private void dataGridView1_CellContentClick(object sender, DataGridViewCellEventArgs e)
```

```
{
}

public void interpolation()
{
   //For color of cubes according to thier grade values....
   Autodesk.AutoCAD.Colors.Color[] acColors = new Autodesk.AutoCAD.Colors.Color[9];
   acColors[0] = Autodesk.AutoCAD.Colors.Color.FromRgb(201, 248, 227);
   acColors[1] = Autodesk.AutoCAD.Colors.Color.FromRgb(245, 196, 224);
   acColors[2] = Autodesk.AutoCAD.Colors.Color.FromRgb(243, 11, 220);
   acColors[3] = Autodesk.AutoCAD.Colors.Color.FromRgb(11, 193, 213);
   acColors[4] = Autodesk.AutoCAD.Colors.Color.FromRgb(11, 243, 173);
   acColors[5] = Autodesk.AutoCAD.Colors.Color.FromRgb(139, 243, 11);
   acColors[6] = Autodesk.AutoCAD.Colors.Color.FromRgb(243, 208, 11);
   acColors[7] = Autodesk.AutoCAD.Colors.Color.FromRgb(243, 123, 11);
   acColors[8] = Autodesk.AutoCAD.Colors.Color.FromRgb(243, 11, 11);

   for (double z = z_min; z <= z_max; z = z + 3)
   {
      for (double y = y_min; y <= y_max; y = y + 3)
      {
         for (double x = x_min; x <= x_max; x = x + 3)
         {
            List<double[]> myarray = new List<double[]>();
            calculate_relev_distance(x, y, z,ref myarray);
            double grade =  calculate_grade(ref myarray);
            if(grade > 0)
            {
               drawPoint(x,y,z,grade,ref acColors);
            }
            myarray.Clear();
         }
      }
   }
}

   public void drawPoint(double x, double y, double z, double grade, ref
Autodesk.AutoCAD.Colors.Color[] acColors)
   {
```

```
        Document acDoc =
Autodesk.AutoCAD.ApplicationServices.Application.DocumentManager.MdiActiveDocument;
        Autodesk.AutoCAD.DatabaseServices.Database acCurDb = acDoc.Database;
        Editor ed = acDoc.Editor;
        // Start a transaction
        using (DocumentLock docLock = acDoc.LockDocument())
        {
          using (Transaction acTrans = acCurDb.TransactionManager.StartTransaction())
          {
            // Open the Block table record for read
            BlockTable acBlkTbl;
            acBlkTbl = acTrans.GetObject(acCurDb.BlockTableId,
                                  OpenMode.ForRead) as BlockTable;

            // Open the Block table record Model space for write
            BlockTableRecord acBlkTblRec;
            acBlkTblRec = acTrans.GetObject(acBlkTbl[BlockTableRecord.ModelSpace],
                                    OpenMode.ForWrite) as BlockTableRecord;

            // Create a single-line text object
            using (DBText acText = new DBText())
            {
              acText.Position = new Point3d(x, y, z);
              acText.Height = 0.5;
              acText.TextString = grade.ToString("0.###");
              acBlkTblRec.AppendEntity(acText);
              acTrans.AddNewlyCreatedDBObject(acText, true);
            }
            // Create a 3D solid box
            Solid3d acSol3D = new Solid3d();
            acSol3D.SetDatabaseDefaults();
            acSol3D.CreateBox(3, 3, 3);
            // Position the center of the 3D solid at (5,5,0)
            acSol3D.TransformBy(Matrix3d.Displacement(new Point3d(x, y, z) - Point3d.Origin));

            //Color
            Autodesk.AutoCAD.Colors.Color curcolor = new Autodesk.AutoCAD.Colors.Color();

            if (grade.ToString("0.#") == "0.9")
```

```
            { curcolor = acColors[8]; }
        else if (grade.ToString("0.#") == "0.8")
            { curcolor = acColors[7]; }
        else if (grade.ToString("0.#") == "0.7")
            { curcolor = acColors[6]; }
        else if (grade.ToString("0.#") == "0.6")
            { curcolor = acColors[5]; }
        else if (grade.ToString("0.#") == "0.5")
            { curcolor = acColors[4]; }
        else if (grade.ToString("0.#") == "0.4")
            { curcolor = acColors[3]; }
        else if (grade.ToString("0.#") == "0.3")
            { curcolor = acColors[2]; }
        else if (grade.ToString("0.#") == "0.2")
            { curcolor = acColors[1]; }
        else if (grade.ToString("0.#") == "0.1")
            { curcolor = acColors[0]; }

        acSol3D.Color = curcolor;

        //Block thickness
        acSol3D.LineWeight = LineWeight.LineWeight020;
        acCurDb.LineWeightDisplay = true;

        // Add the new object to the block table record and the transaction
        acBlkTblRec.AppendEntity(acSol3D);
        acTrans.AddNewlyCreatedDBObject(acSol3D, true);
        acTrans.Commit();
      }
    }
}

public double calculate_grade(ref List<double[]> myarray)
{
    double sum_up = 0;
    double sum_down = 0;
    for(int i = 0; i < myarray.Count(); i++)
    {
        if (myarray[i][0] == 0)
```

```
            return myarray[i][1];
        sum_up = sum_up + myarray[i][1] / (myarray[i][0]* myarray[i][0]);
        sum_down = sum_down + (1 / (myarray[i][0] * myarray[i][0]));
      }
      if (sum_down == 0)
        return 0;

      double grade = sum_up / sum_down;
      return grade;
    }

    public void calculate_relev_distance(double x,double y,double z,ref List<double[]> myarray)
    {
      for (i = 0; i < dataGridView1.Rows.Count ; i++)
      {
        double x1 = Convert.ToDouble(dataGridView1.Rows[i].Cells[0].Value);
        double y1 = Convert.ToDouble(dataGridView1.Rows[i].Cells[1].Value);
        double z1 = Convert.ToDouble(dataGridView1.Rows[i].Cells[2].Value);
        double g = Convert.ToDouble(dataGridView1.Rows[i].Cells[3].Value);

        double d = Math.Sqrt((x1 - x) * (x1 - x) + (y1 - y) * (y1 - y) + (z1 - z) * (z1 - z));
        if(d < 10)   //Setting inference zone
        {
          double[] dis = new double[] { d, g };
          myarray.Add(dis);
        }
      }
    }
  }
}
```