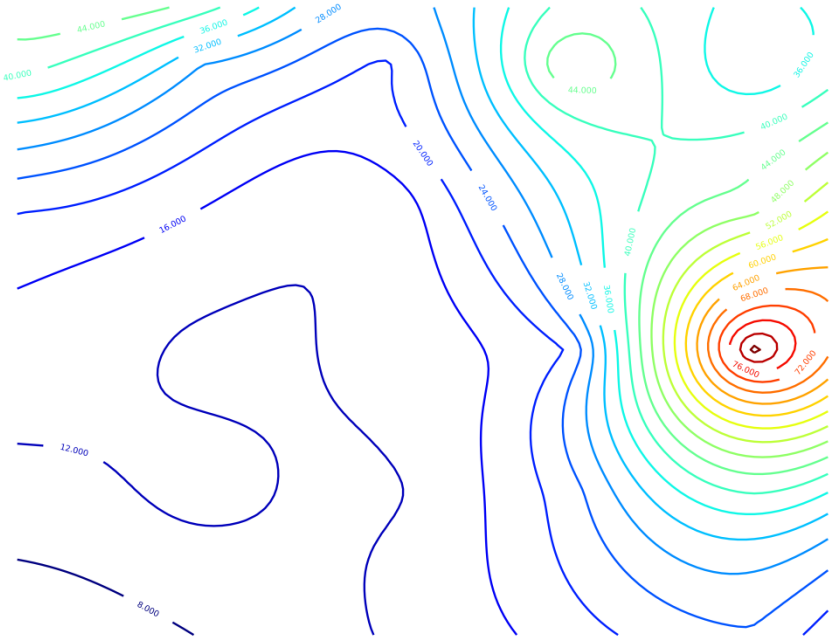
2012

GIS-Contour plotting



Project Members:

Sushant Hiray

Nilesh Satish kulkarni

Vipul Harsh

IIT Bombay

6/22/2012

Objective:

To create and plot elevation contours using sample data points obtained from Google API (includes kriging , interpolation) in less time.

Introduction:

Kriging: It is a group of **geostatistical** techniques to interpolate values of a random field, in this case elevation, at an unobserved location from observations of its value at nearby locations. There are several types of kriging, here we are using ordinary kriging which calculates the weighted average and the weights are decided by minimizing the variance (mean square error).

$$Z(x) = \beta - \sum_{i=1}^{n} variogram(\left|\left|x - x_i\right|\right|)\varepsilon_i$$

Where ε_i and β are the weights to be calculated using the equations

$$Z(x_i) = \beta - \sum_{j=1}^{n} variogram(\left|\left|x_j - x_i\right|\right|)\varepsilon_j$$

$\sum\varepsilon_i$=0

Variogram: It is a function describing the degree of spatial dependence of a spatial random field (elevation in this case). It is defined as the variance of the difference between field values at two locations across realizations of0 the field. If the spatial random field has constant mean, this is equivalent to the expectation for the squared increment of the values between locations. .

Brief Description of the Python Code:

The following are the essential parts of the code:

- ***Variogram:*** this function computes the best polynomial fit (of any specified degree, currently 5) for the variogram using the sample points. It returns the coefficients of the polynomial array.

- ***Kriging:*** this function calculates the elevations at the nodes of a grid of given dimensions using the ordinary method for kriging with the given sample data points. There are 2 versions of this function , the first one is more accurate but extremely slow and hence the second version is being used.

- ***Contour_plot:*** it makes and plots the contours using the predefined routines in python(matplotlib). It takes the elevations and their respective locations(latitude and longitude) as the input.

Brief Description of Matlab Code:

After extracting the krigged data , a Matlab program creates kml files of contour plots. These kml files are then overlayed on google Earth.

The following are the essential parts of the code:

Function **kml_contour(lat, lon, elev, length)**

- The function takes an array of latitudes, longitudes and a 2D array of krigged elevation data and generates corresponding contours.
- The function defines default values for line width, color and the size of font in between the contours .

Description about the running version of the code:

The version on which the website currently functions uses linear best fit for variogram and is a bit slow.

Linear Best is a good choice for plain regions (in our case Thane).But higher degree polynomial fits ought to be used for regions with highly undulating terrain (ex: Mountainous Regions).

Brief Analysis:

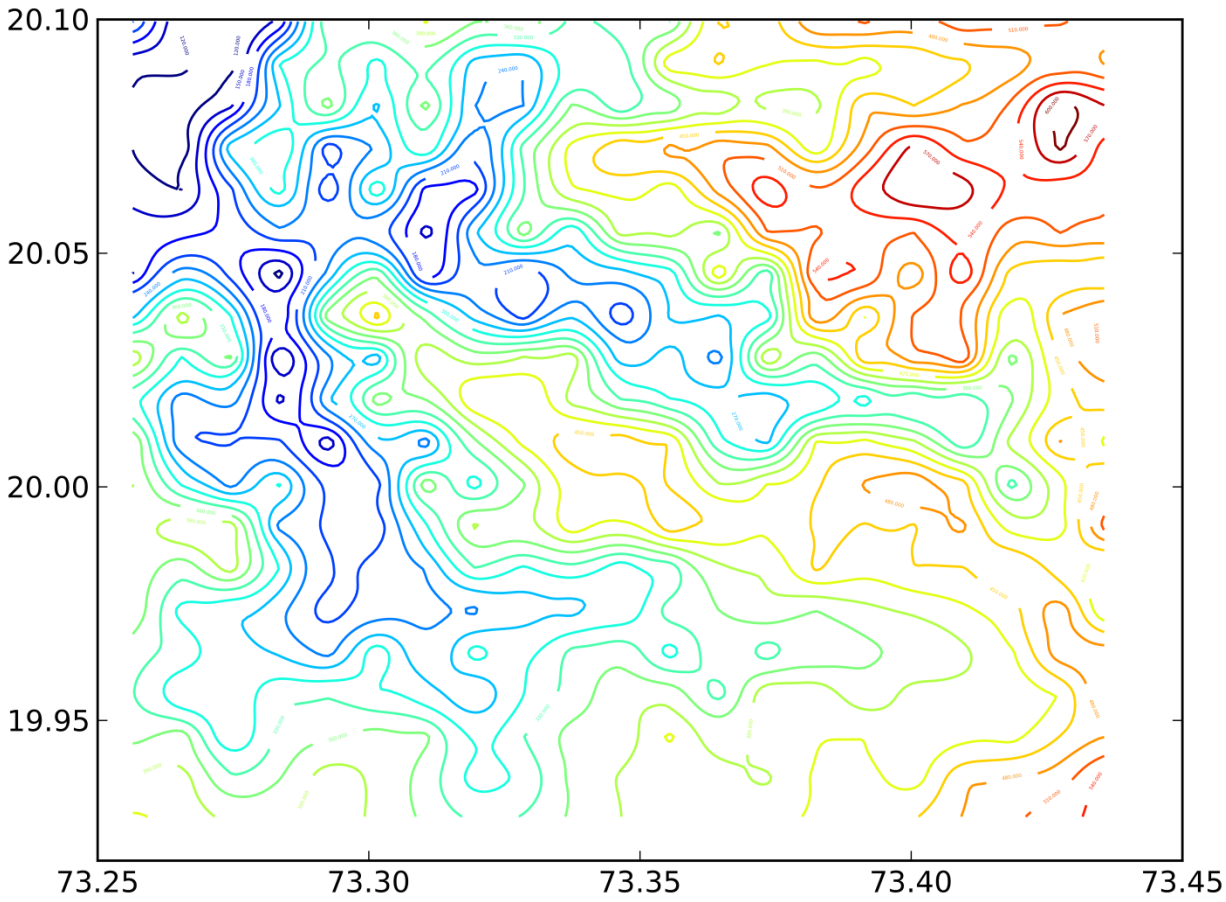
- ### Comparison of the codes :

  The running time of the algorithm currently being implemented for the kriging part and the algorithm which we have used are almost the same.

  The following points give additional functionality to our code:

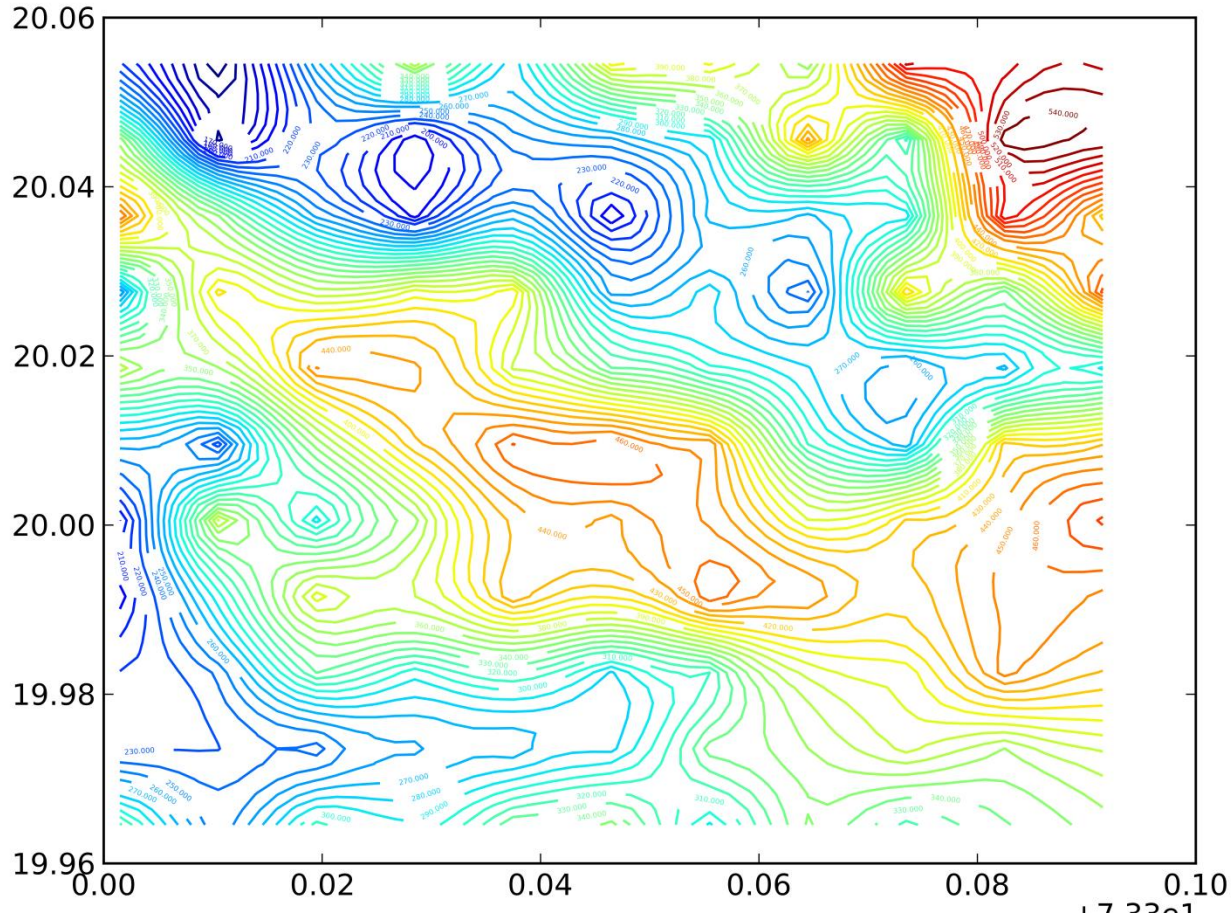
  i. We can specify the number of points for which the user wants to get the data krigged.
  ii. We can specify the degree of the polynomial of the best fit curve.
  iii. The area around which the data has to be queried from **Google API**

  The following images show the contour plot comparison between our code and current code which is being implemented:



Mokhada Region Contours(our code).

T



This is the contour plot done by the code which is currently used on the website.

The following differences are noticeable between the 2 plots:

1. There are a significant number of sharp points on the contour plot by the currently implemented code as compared to our version.

Effect of change in degree of the best-fit polynomial:

As expected the increase in the degree of the best-fit polynomial results in increase of the computing time:

Degree	1	3	5	7	10
Time	119.29	130.09	141.34	150.31	187.7

The computing time shows that it is linear in the degree of the polynomial. There is a trade-off between computing time and the quality of the contours.

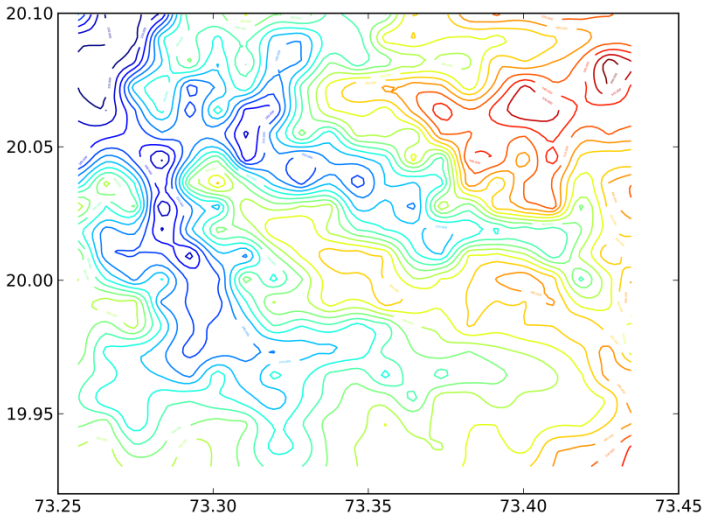


Figure: Contour plot of Mokhada with degree 1

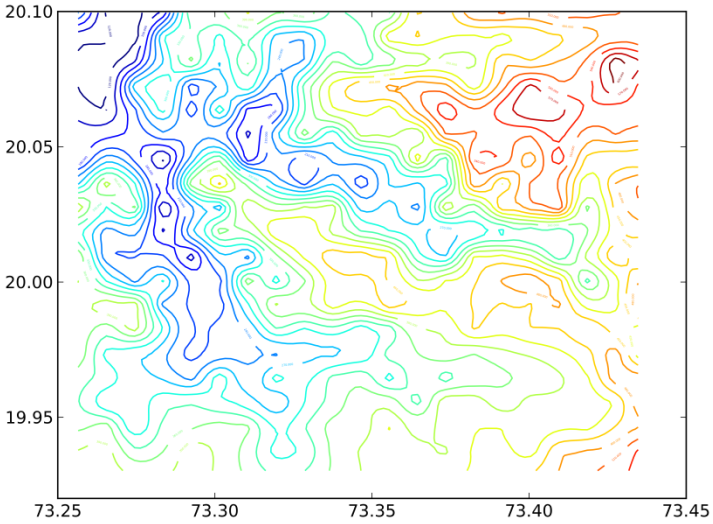


Figure: Contour plot of Mokhada with degree 10

The plots show that there isn't much difference between the two contour plots .But the variation of degree of polynomials is useful in undulating terrains. In such places linear best fit fails terribly.

Effect of the change in the sampled grid size:

- Computing time:
  - The time taken to krig the data and form a 100x100 grid was 145.32 seconds compared to the 500.45 seconds taken when the grid size was 200x200 .
  - This shows that the kriging algorithm is **O(n^2)**
- Contour quality:

  As the grid size increases from 100x100 to 200x200 the quality of the contour increases sharply. There are very few sharp points and the contour generated is very smooth.

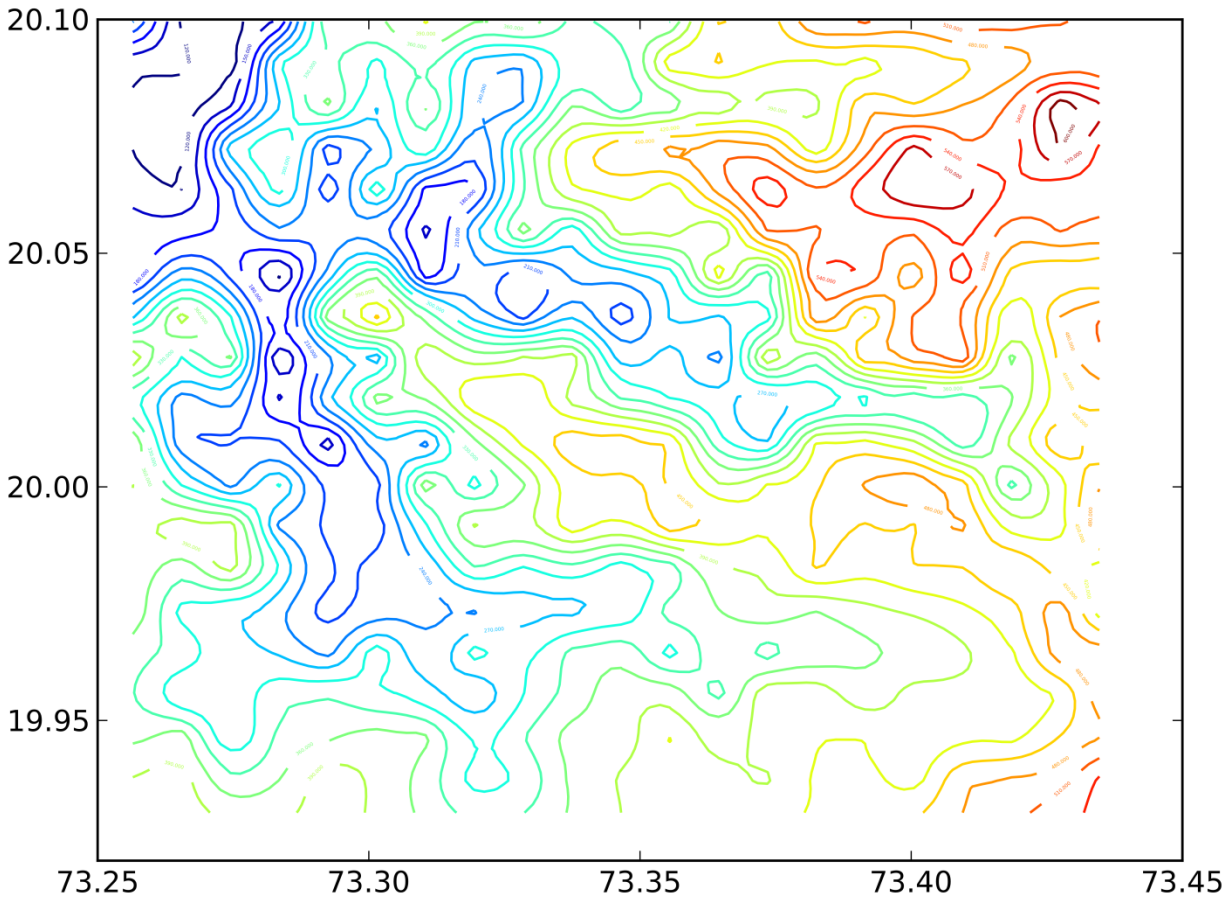


Figure: Contour plot when the krigged grid was 100x100.

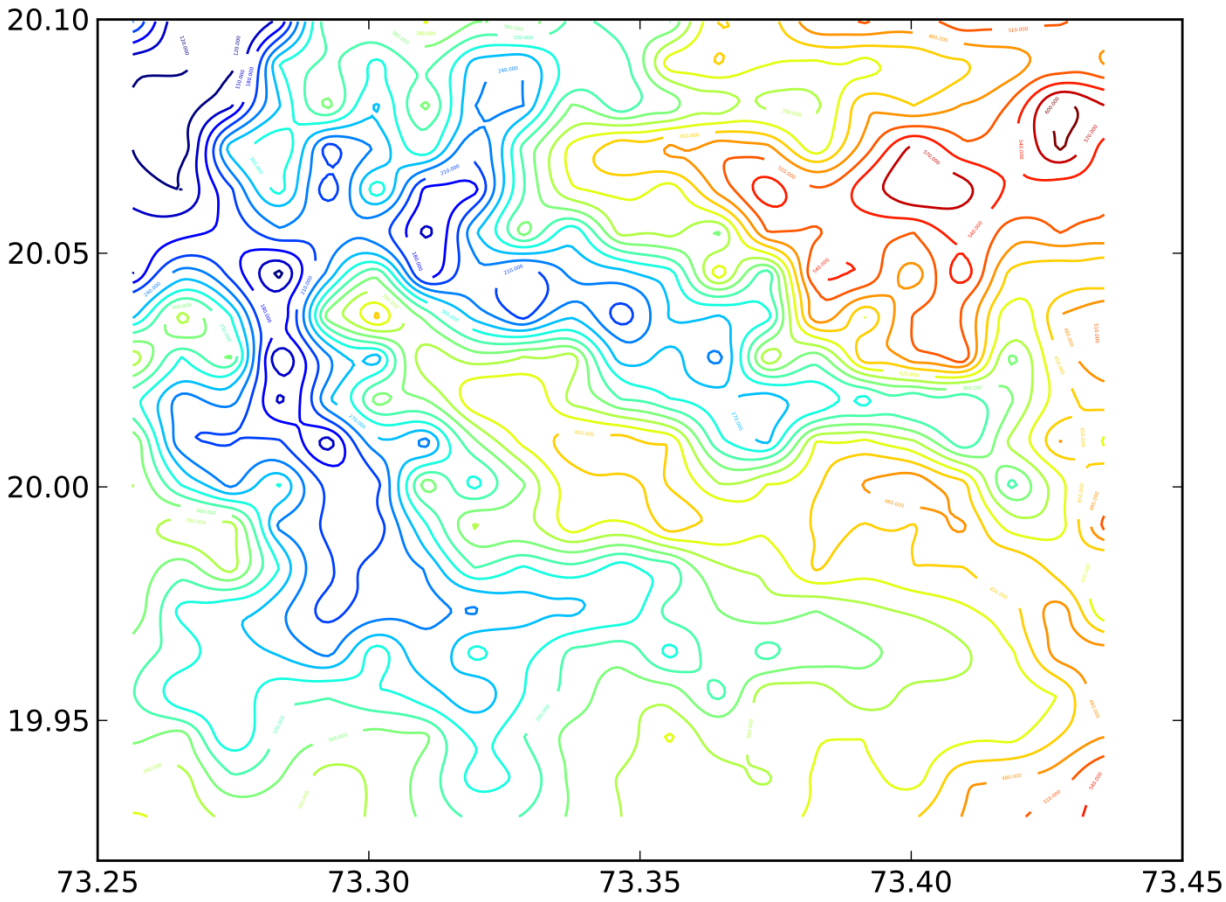
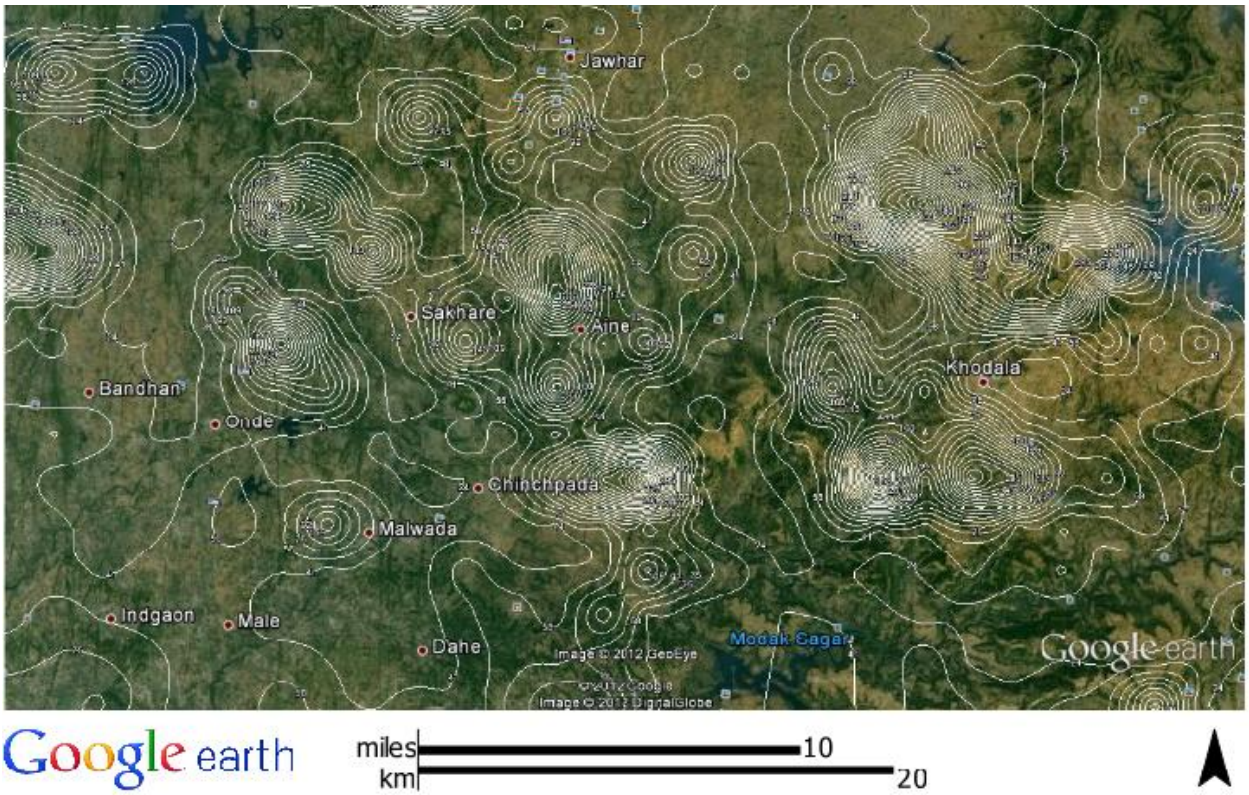


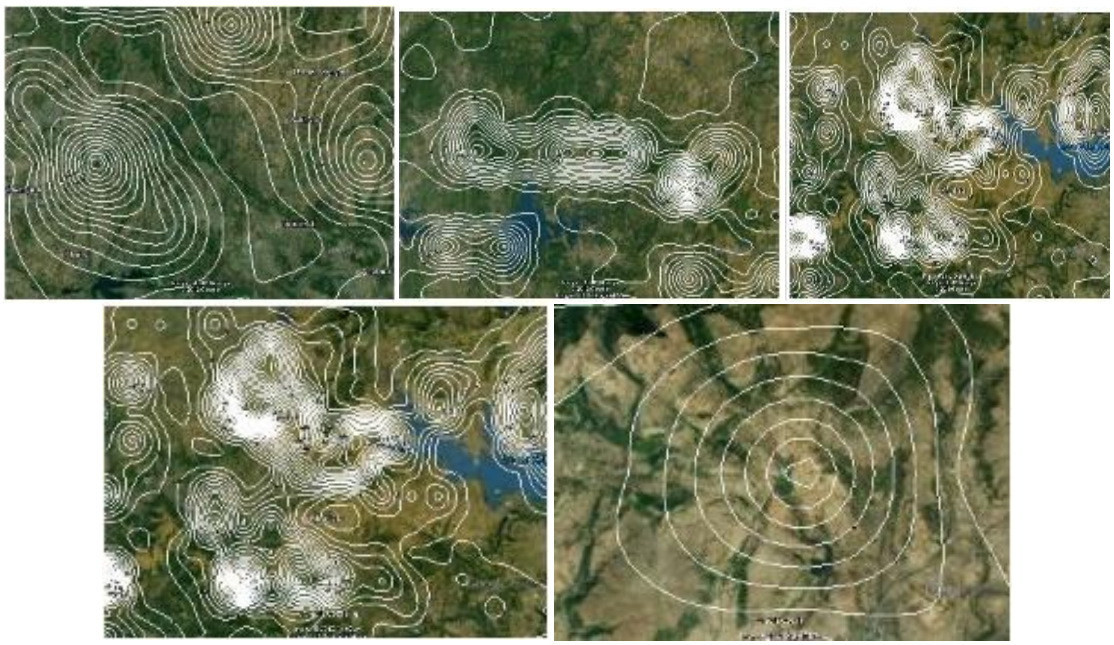
Figure: Contour plot with 200x200 grid.



Screen shot of the KML file overlayed on Google Earth

This is a sample overlay of a kml file on Google Earth. From this sample we notice various shapes of contour which the plotter can plot. Moreover there are very few sharp turns .

The area of the region is 50 km x 30 km (around Jawahar District).



These are some more bitmaps which show the contour plots at various terrains. In picture 1 and 5 we can clearly see that plots are highly accurate. The plots 3 and 4 show some contour lines around the water body so they seem to be satisfactorily correct.

Figure shows an ultra zoomed view of a contour around a water body. The sharpness of the curve arises due to discrepancies in the sampled data.

Acknowledgements:

This project has been a great learning experience for us and hence at this point we want to express our earnest gratitude to all those who were involved in it.

- Prof Bharat Adsul : For granting us this excellent opportunity.
- Prof Milind Sohoni: For giving us constant help and direction as and when required.
- Rahul B Gokhale: For explaining to us minute details about different parts of the project.
- Arvind: For guiding us regarding the contour plotting routines.
- Anuja Shukla:For giving us instructions about handling KML files.
- Python makers: For making such an awesome easy to use programming language.