Multi-view triangulation and Non-linear optimization

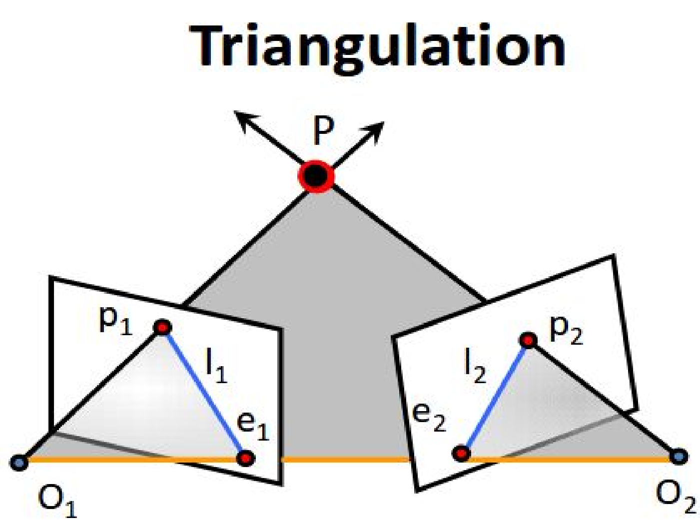
By: Gulshan Kumar

1. Multi-view triangulation

Multi-view triangulation is a straight forward extension of 2-view triangulation which I have already coded in the previous assignment.

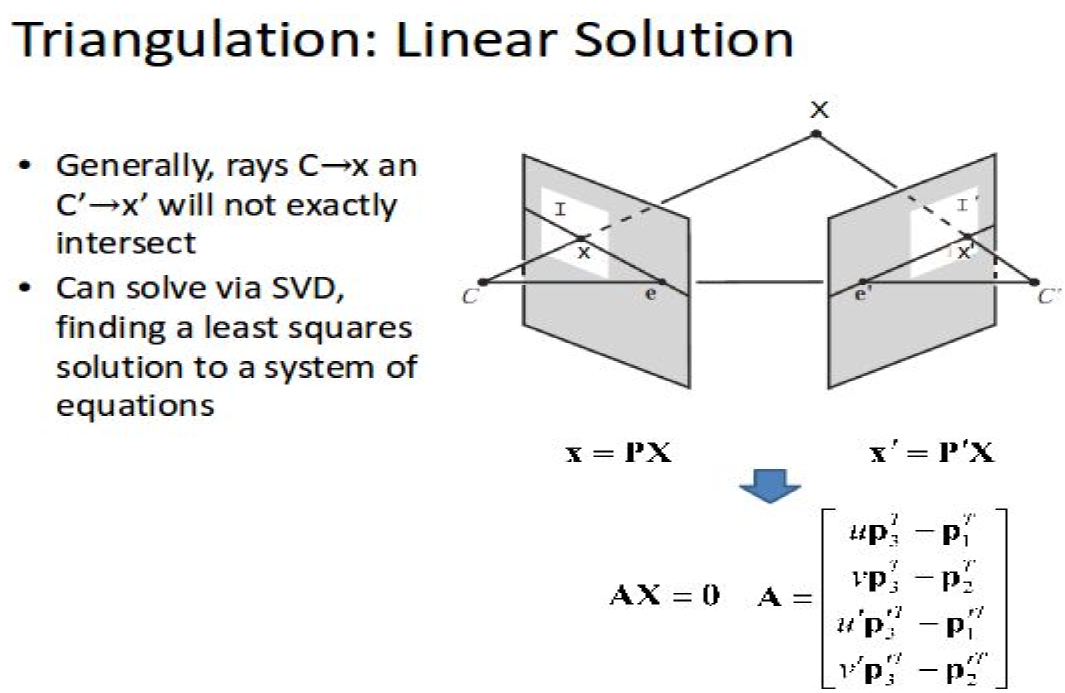
We have used projection matrices of all the 8 views and setup a least square system of the form Ax = b and then solve it using SVD. For example 3D point X3 must satisfy the following constraints P1*X3 = x13, P2*X3 = x23, ..., P8*X3 = x83, where x13 denotes the 2D projection of X3 in image 1, x23 denotes the 2D projection of X3 in image 2,..., x83 denotes the 2D projection of X3 in image 8.

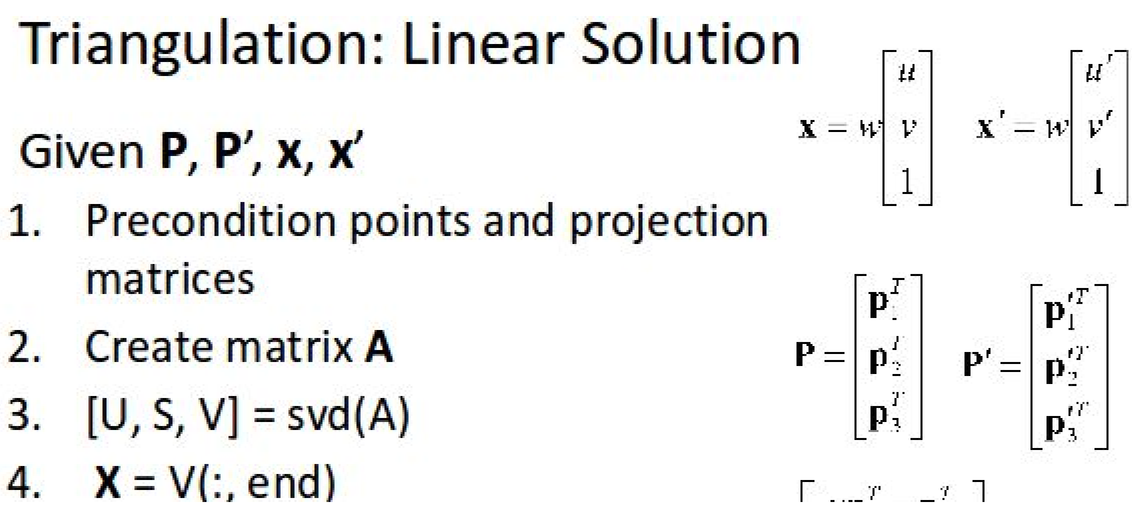
Triangulation



Parameter Solving for All View triangulation

Similar to 2 view triangulation we will make A matrix for 8 views provided and take the SVD to get the homogenous world coordinates(4 x 1 vector). We will divide the fourth coordinate to get our 3d world coordinates.(3 x 1 vector)



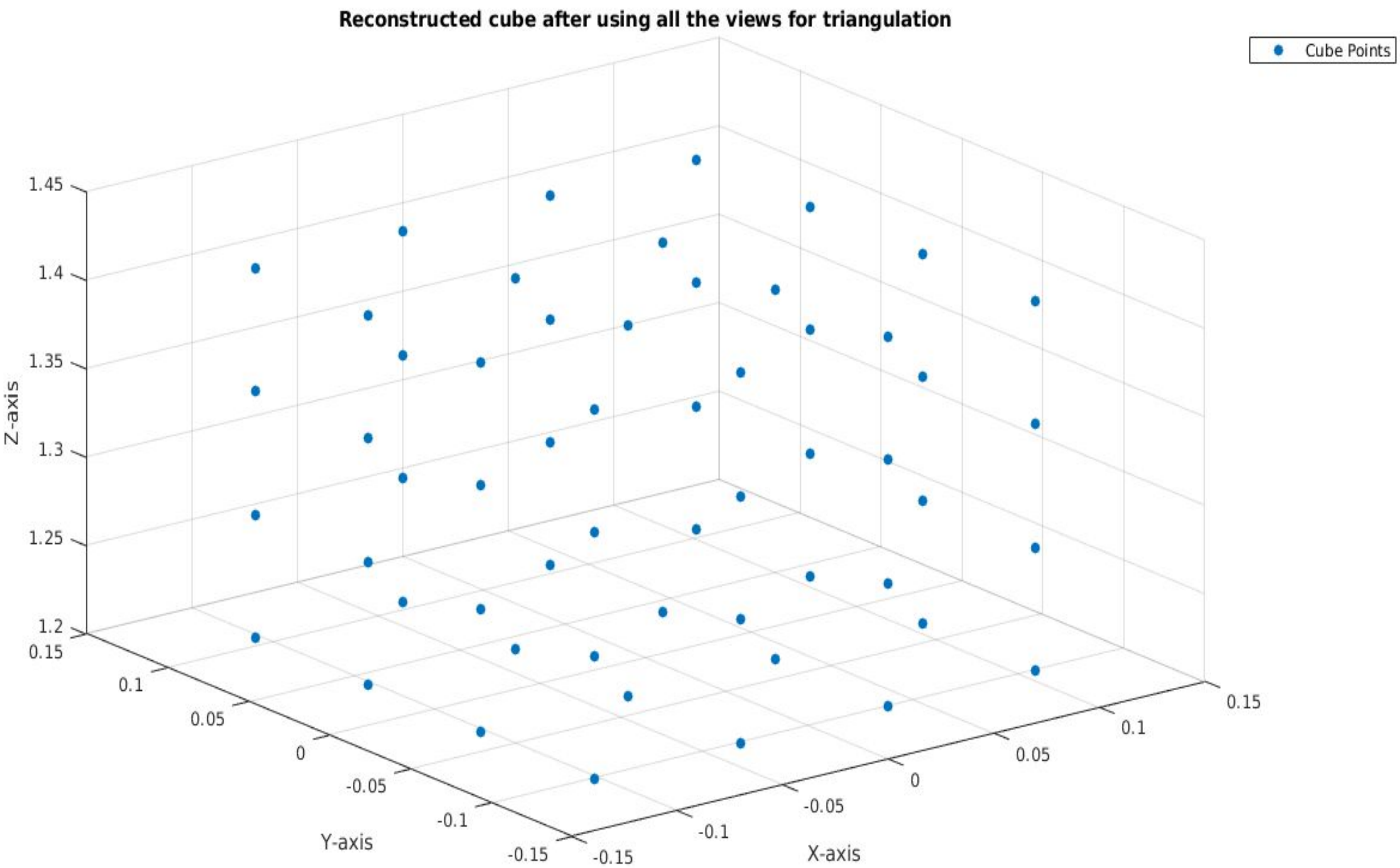


ALL View Triangulation Function

```
function [pts3D] = allviewsalgebraicTriangulation(pts2D, ProjMat_1,
ProjMat_2, ProjMat_3,ProjMat_4,ProjMat_5,ProjMat_6,ProjMat_7,ProjMat_8)
      %% Calculating the 3D coordinates using Linear Triangulation %%
      A = [pts2D(1,1)*ProjMat_1(3,:) - ProjMat_1(1,:);
pts2D(1,2)*ProjMat_1(3,:) - ProjMat_1(2,:);
      pts2D(2,1)*ProjMat_2(3,:) - ProjMat_2(1,:); pts2D(2,2)*ProjMat_2(3,:)
- ProjMat_2(2,:);
      pts2D(3,1)*ProjMat_3(3,:) - ProjMat_3(1,:); pts2D(3,2)*ProjMat_3(3,:)
- ProjMat_3(2,:);
      pts2D(4,1)*ProjMat_4(3,:) - ProjMat_4(1,:); pts2D(4,2)*ProjMat_4(3,:)
- ProjMat_4(2,:);
      pts2D(5,1)*ProjMat_5(3,:) - ProjMat_5(1,:); pts2D(5,2)*ProjMat_5(3,:)
- ProjMat_5(2,:);
      pts2D(6,1)*ProjMat_6(3,:) - ProjMat_6(1,:); pts2D(6,2)*ProjMat_6(3,:)
- ProjMat_6(2,:);
      pts2D(7,1)*ProjMat_7(3,:) - ProjMat_7(1,:); pts2D(7,2)*ProjMat_7(3,:)
- ProjMat_7(2,:);
      pts2D(8,1)*ProjMat_8(3,:) - ProjMat_8(1,:); pts2D(8,2)*ProjMat_8(3,:)
- ProjMat_8(2,:);];
      [~, ~, V ] = svd(A);
      pts3D = V(:,end);
end
```

Given P1,P2,....,P8 , we have solved for 3D location of all points using least squares .

The reconstructed points are as given below :



2. Levenberg-Marquardt (LM) Algorithm for non-linear least square

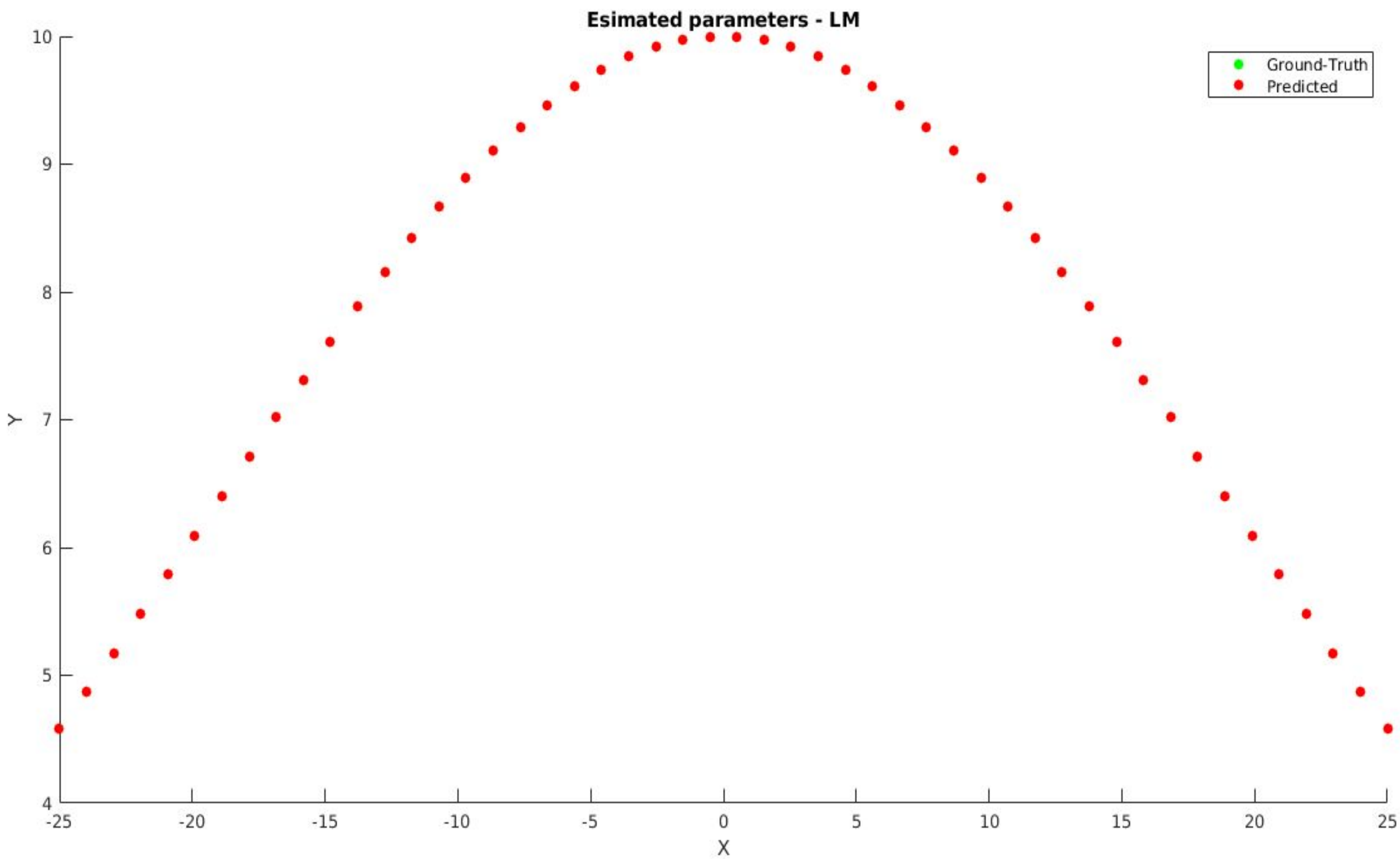
We have implemented ONLY the trust-region strategy of Levenberg-Marquardt (LM) method for non-linear least square problems.The required code for the trust region of LM algorithm has been written in the file '**testLevenbergMarquardt.m**' .

The error or difference in outputs obtained was of the order 8.2510e-16 . The obtained parameters after some 22 iterations (after which the program stops) is same as input parameters i.e

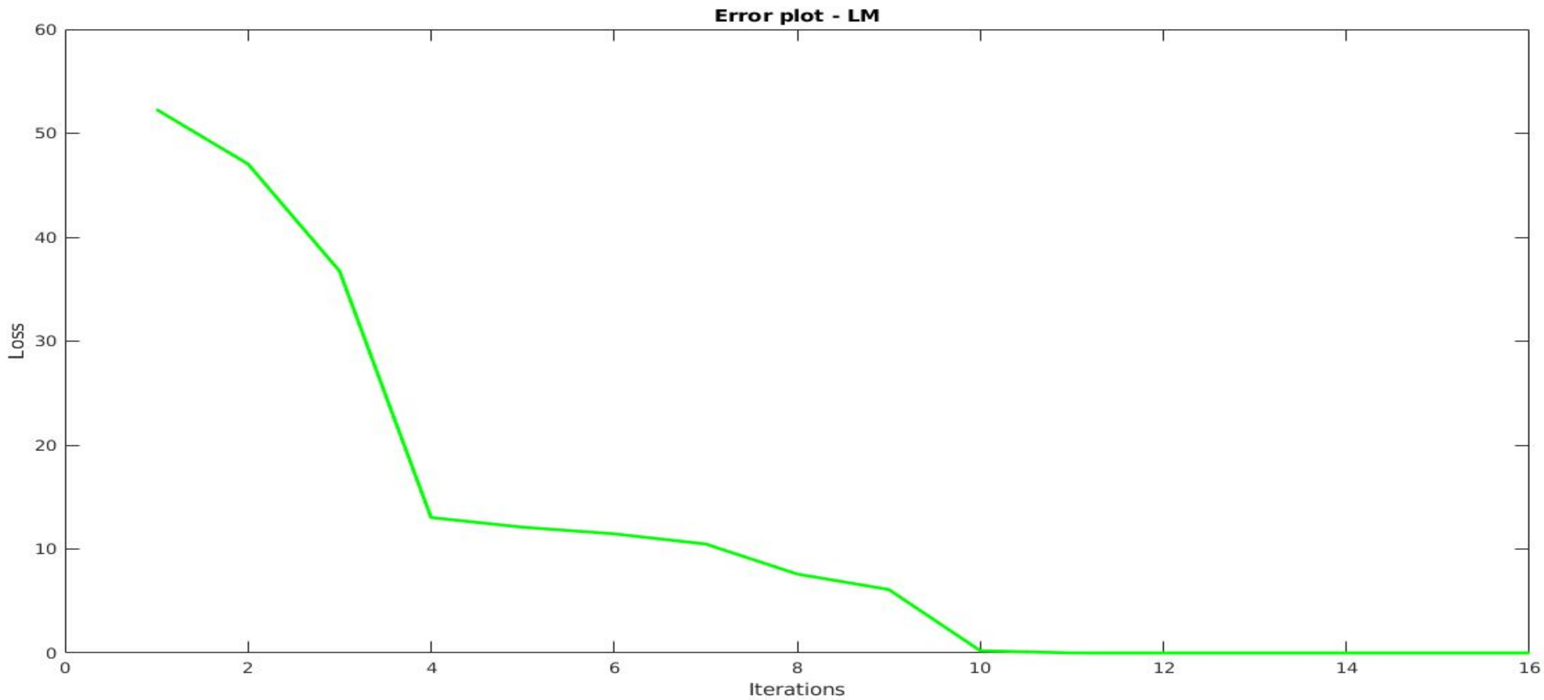
a = 10 , mean = 0 , variance = 20 .

Levenberg-Marquardt (LM) is quite better than gauss Newton method where difference/error in outputs is about 0.5694.

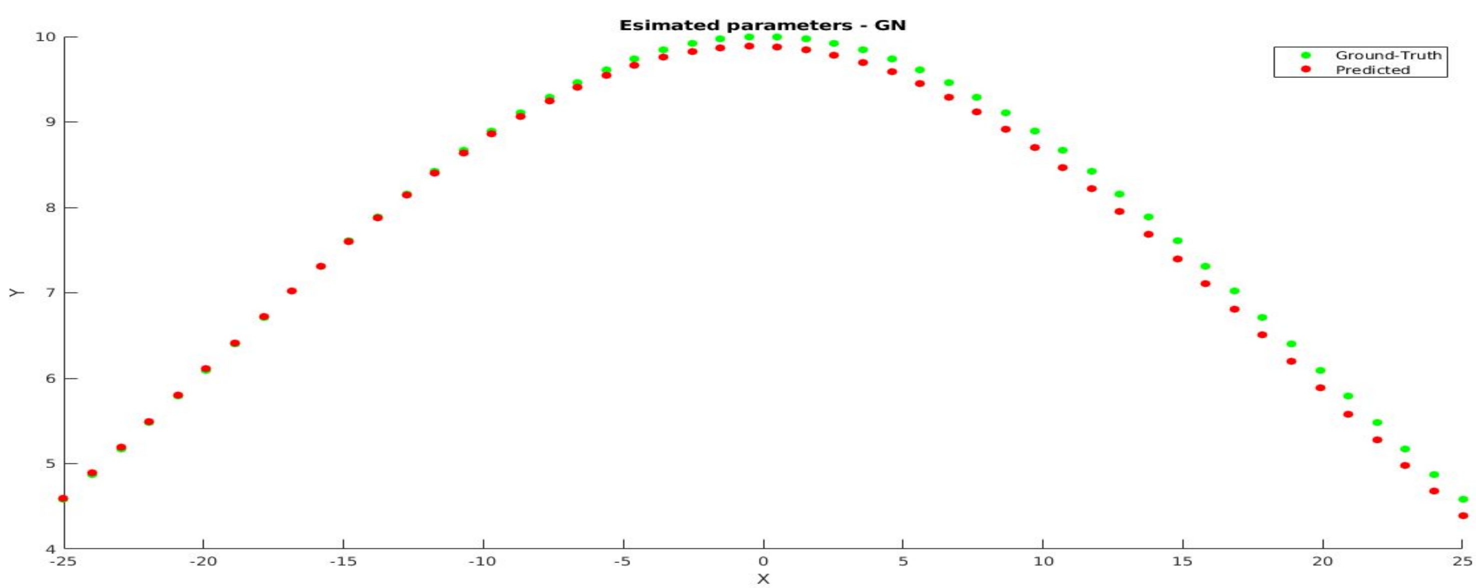
Estimated parameters- LM



Error Plot LM



Estimated Parameters Gaussian Newton



Error Plot Gaussian Newton