3. Report

Introduction:

In this homework, we deal with the problem of 3-D point data set registration. Point set registration, or point matching, is a method to find the best spatial transformation to align two point data set or point clouds in space. We are given an input data set X, and another data set Y, which is a rotated, translated and scaled version or a subset of X and we are asked to align the data set Y to X. We have used the paper, “A Method for Registration of 3-D Shapes”[1], by Paul J. Besl and Neil D. McKay, as the base paper for this assignment and used the method describe in it for efficient registration of 3-D shapes including free-form curves and surfaces. The paper describes a method based on the Iterative Closest Point (ICP) algorithm for the registration. Here, we have a version of the algorithm implemented. We were supposed to

a) Understand and explain the method described in the paper
b) Explore various data set to analyze pros and cons of the method as well as the code.
c) Add error analysis to the code and show the error change with respect to the iterations of the algorithm.
d) Produce a write-up to show understanding from the homework

A) Iterative Closest Point (ICP) Algorithm

The algorithm, as the name suggest, does exactly the same. The method computes closest point on a geometric shape to any given point and aligns the point to that point iteratively. The method to find the closest point on a point data set to any given shape, is described in Section III of Besl and McKay et. al. The section describes finding the closest distance of a point from a point, a line segment and a triangle as the base to find the distance between a point and any given 3-D entity since any 3-D entity can be describe in terms of a set of points, line segment or triangles. In the paper, section IIIA, IIIB and IIIC describes point to parametric entity distance, implicit entity distance and point set registration respectively.

We have used the point to point set registration method to describe the find closest point function, which is a quaternion-based algorithm. There are a couple of more algorithm for the given task, of which, Besl and McKay mention Single Value Decomposition (SVD) method and state that quaternion based method is preferred over SVD method in two and three dimensions, since reflections are not desired. The SVD method is preferred for generalization of the algorithm to any n>3 dimensions. The basic quaternion based method is described in the paper, “Closed-form solution of absolute orientation using quaternions”, by Berthold K. P. Horn[2].

In the paper, Horn talks about the photogrammetric problem of recovering the transformation between two systems from measurements given in two different co-ordinate systems. He explains that, “the transformation between two Cartesian co-ordinate systems can be thought of as the result of a rigid-body motion and can thus be decomposed into a **rotation** and a **translation.** In addition, the **scale** may not be known.” He proposes a closed-form solution to the least-square problem of absolute orientation. The advantages of a closed form solution are, a) they give the best possible transformation in one step and b) we do not require a good initial get for the best possible results. Horn states, “information may be already available which obtains such an initial guess of the transformation parameters that a single step iteration brings them close enough to the solution.”

The Horn method uses quaternions to describe the rotation. There are other methods, like , Euler Angles, Gibbs Vector, Orthonormal Matrices and Hamilton’s quaternions that Horn mentions. The advantages of a quaternions over the others can be listed as,

* it is simpler to enforce that a quaternion have unit magnitude, difficult for orthonormal matrices
* unit quaternions are closely allied to the geometrically intuitive axis and angle notation.
* it takes fewer arithmetic operations to multiply two quaternions than it does for two 3X3 matrices
* it is trivial to find nearest unit quaternion than to find nearest orthonormal matrix

The basic transformation between two point set r_r and r_l can be described of the form,

$$r_r = sR(r_l) + r_0$$

where, s is the scale factor, $R(r_l)$ is the rotated version of the vector r_l and r_0 is the translational offset.

The steps described to find the given transformation is given as,

1. Find the centroids of the two sets of measurements in the left and right coordinate system.
2. The centroid are subtracted from all measurements, to make all measurements relative to the centroids
3. For each pair of co-ordinates, we compute the nine possible products. It is mentioned earlier that we need at least three points in both coordinate system to get the constraints to solve for s, R and r_0.
4. These nine products are used to find the solution.

Essentially, in Besl and McKay paper, they define rotation in terms of a unit quaternion, $q_R = [q_0\ q_1\ q_2\ q_3]$. The 3X3 rotation matrix is described in equation 21 of Besl and McKay et al. And the above mentioned steps are described in equation 22-26. For our problem, in context of two point data set X and P, of which X is input and we are trying to align P to X. Finally, we get the least square quaternion operation, which is $O(N_p)$ and is denoted as,

$$(q, d_{ms}) = Q(P, X)$$

where, d_{ms} is the mean square point matching error. The notation q(P) is used to denote the point set P after transformation by the registration vector q.

Once we know the method for computing the closest point on a geometric shape, the ICP algorithm can be described in terms of X, any abstract geometric shapes. Given any point set, P with N_p points $\{p_i\}$ from the data shape, and the model shape X, the steps involved in ICP algorithm are,

1. Initialize $P_0 = P$, $q_0 = [1\ 0\ 0\ 0\ 0\ 0\ 0]$ and k = 0. Iterate over point 2-5 until convergence.
2. Compute the closest points: $Y_k = C(P_k, X)$. In our implementation, we use findClosest method, which takes the point sets X corresponding to X, $Ynew_k$ corresponding to P_k, for each iteration and updates $Ynew_{k+1}$.

3. Compute the registration: $(q_k, d_k) = Q(P_0, Y_k)$, which is implemented using the Horn algorithm implemented in Besl & McKay paper, described above, using the cross covariance matrix of the the dataset described as,

$$Q(\Sigma_{px}) = \begin{pmatrix} tr(\Sigma_{px}) & \Delta^T \\ \Delta & \Sigma_{px} + \Sigma_{px}^T - tr(\Sigma_{px}) I_3 \end{pmatrix}$$

where, $\Delta = [A_{23}\ A_{31}\ A_{12}]^T$, $A_{ij} = (\Sigma_{px} - \Sigma_{px}^T)_{ij}$ and I_3 is 3X3 identity matrix.

The maximum eigen value of the given matrix is selected as the optimal rotation. The rotation matrix R is the calculated using qRotate, which takes the maximum eigen value. The translation transformation is calculates using the qTranslate method which takes the R matrix and the centroid of the two data set, using,

$$q_T = \mu_x - R(q_r)\,\mu_p$$

Thus, we get R and T matrix, that we look for the registration.

4. Apply the registration: $P_{k+1} = q_k(P_0)$, After finding the Ynew, we calculate the Mean Squared error from the new Ynew and the old Ynew, data set. We will discuss error analysis in section C.

5. Terminate the iteration when the change in mean-squared error falls below a preset threshold $\tau > 0$ specifying the desired precision of the registration $d_k - d_{k+1} < \tau$. In our implementation, we fix the number of iteration beforehand to study the registration evolving with iterations.

Thus, we get a very good alignment of the two 3-D point data set with the given method.

B) Data Sets Used

In the implementation of Besl and McKay paper, they demonstrated the algorithm using three types of data sets viz,

a. Point data set Matching

They list out a point set with 8 points and match it against 11 points. Set 1 is a subset of Set 2. They briefly explains the advantage of the algorithm against the brute force approach, explaining quite convincingly how the brute force approach is computationally too intensive to be practical as well as time taken by the approach for a mere 2500 point data matched with 4200 points data set registration is more than 10^{250} universe lifetimes!.

b. Curve Matching

They use a 3D parametric space curve created using B-Splines and showed the registration over them.

c. Surface Matching

We have used Point data set and Surface data set for our implementation. We have used 4 types of data files,

1. Random point data,
2. Triangulated .off files,
3. Mathematica pre-built models, and
4. Triangulated .obj files.

1. Random point data: We have a 2-D and 3-D random data selected as ipick ϵ [11, 16] as follows,
    - 11: 2D without noise data, 5 points
    - 12: 3D without noise data, 5 points
    - 13: 2D noisy data, 5 points
    - 14: 3D noisy data, 5 points
    - 15: Random data, 15 points
    - 16: Random noisy data, 15 points
    - 17: Random data, with Y taken as X subset, 15 points

2. Triangulated .off files: ipick ϵ [21, 29]
    - 21: “Cross” 162 vertices
    - 22: “Dodecagon” 962 vertices
    - 23: “Dragon” 403 vertices
    - 24: “Helix” 505 vertices
    - 25: “King” 314 vertices
    - 26: “Mushroom” 226 vertices
    - 27: “epcot” 770 vertices
    - 28: “Space Shuttle” 2376 vertices
    - 29: “Pear” 867 vertices

3. Mathematica pre-built shapes: ipick ϵ [31,39]
    - 31: “MoebiusStrip” 215 vertices
    - 32: “SpaceShuttle” 310 vertices
    - 33: “Cone” 403 vertices
    - 34: “UtahTeapot” 480 vertices
    - 35: “Seashell” 915 vertices
    - 36: “UtahVWBug” 1148 vertices
    - 37: “BassGuitar” 1375 vertices
    - 38: “Zeppelin” 1792 vertices
    - 39: “HammerheadShark” 2564 vertices

4. Triangulated .obj files: ipick ϵ [41, 44]
    - 41: “Diamond” 6 vertices
    - 42: “Humanoid” 64 vertices
    - 43: “Violin Case” 1080 vertices
    - 44: “Skull” 3386 vertices

Observations:

We show Input X using Blue color, Transformed Y in red and the registered Ynew in Green. We turn the opacity to 1 for X and Ynew to best see how the registration is coming up. Add to the previous observations, these are few of the observations made:

a. After convergence, the green model completely covers the blue model, showing that the registration is complete, for the given shape. [21, 22, 41, 42]

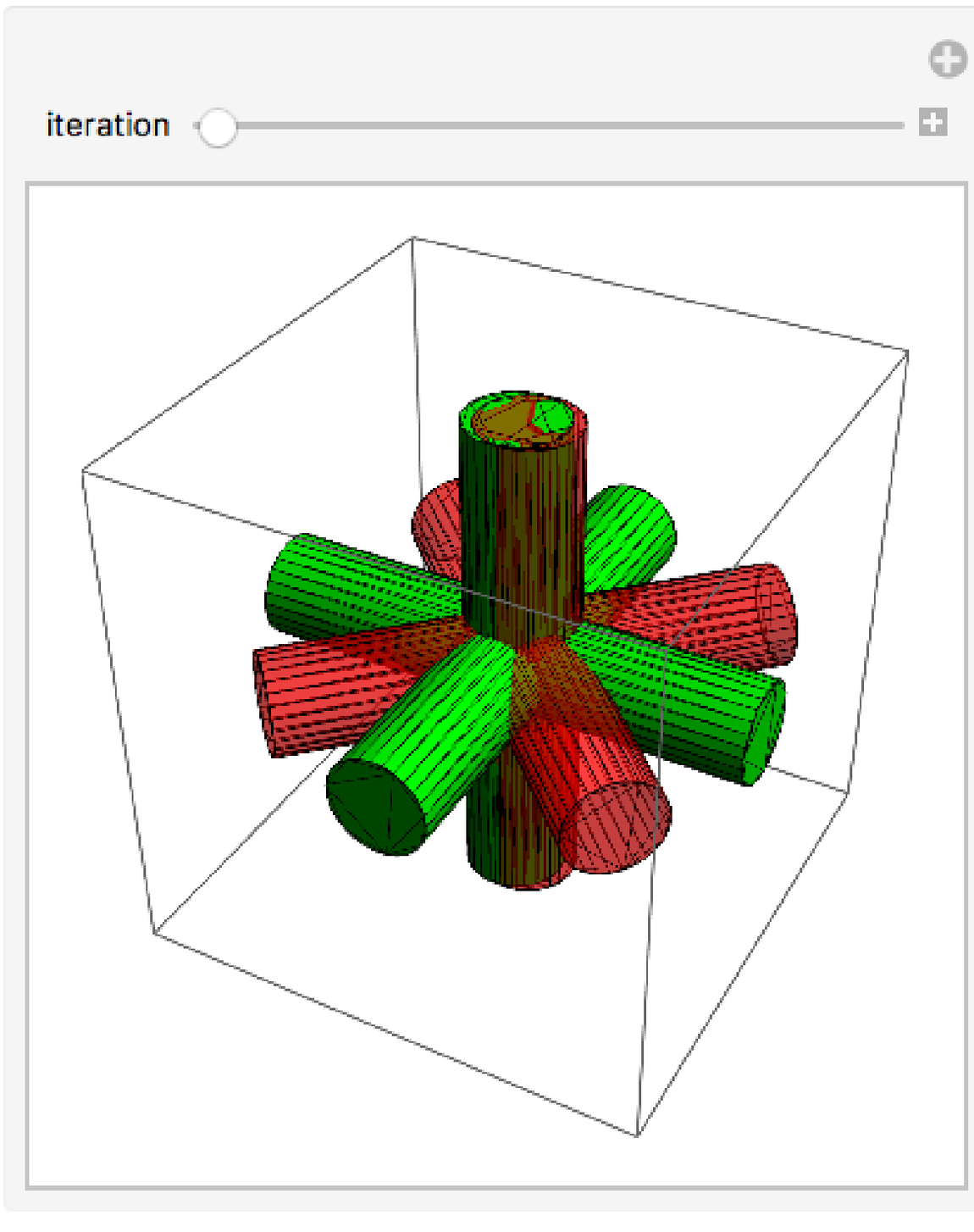
b. Models do not conforms to the original model, although the error graph shows the convergence have been achieved.[24, 31, 32, 33, 44]

c. Many models have gaps in them, models having large number of vertices, showing the triangulation is not done properly. [29, 34, 36, 39, 43]

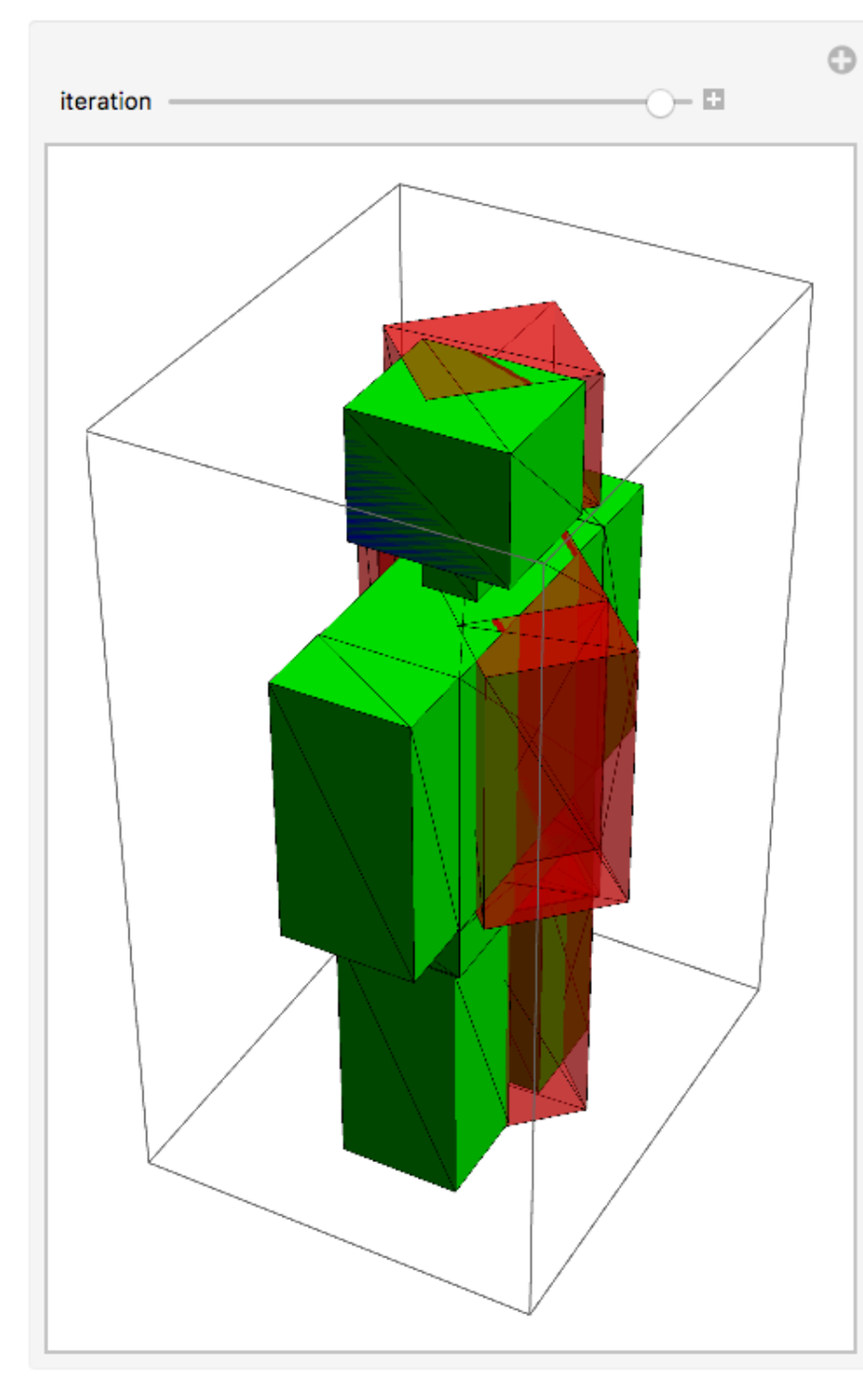
d. The first data set, 2D points after 2 iterations are not able to find the correct correspondence, maybe due to faulty correspondence code. The problem is resolved in 3D points data set. [11, 12,13, 14]

e. Models break there triangulation after a few iteration and becomes meaningless, again showing the correspondence problem in d. [34]

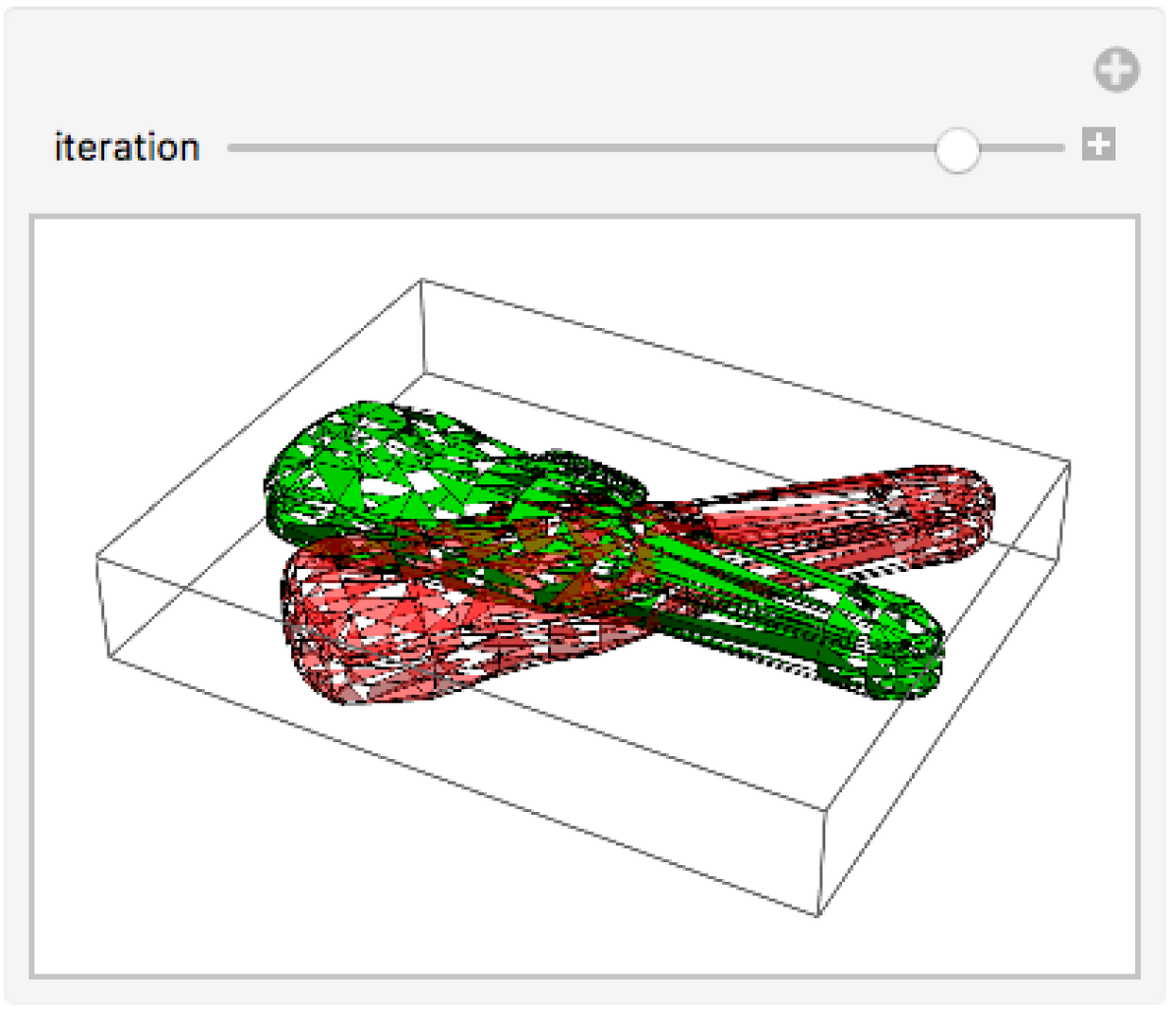
f. We can not take a subset of points to register with the original data set (Case not handled??). [16]



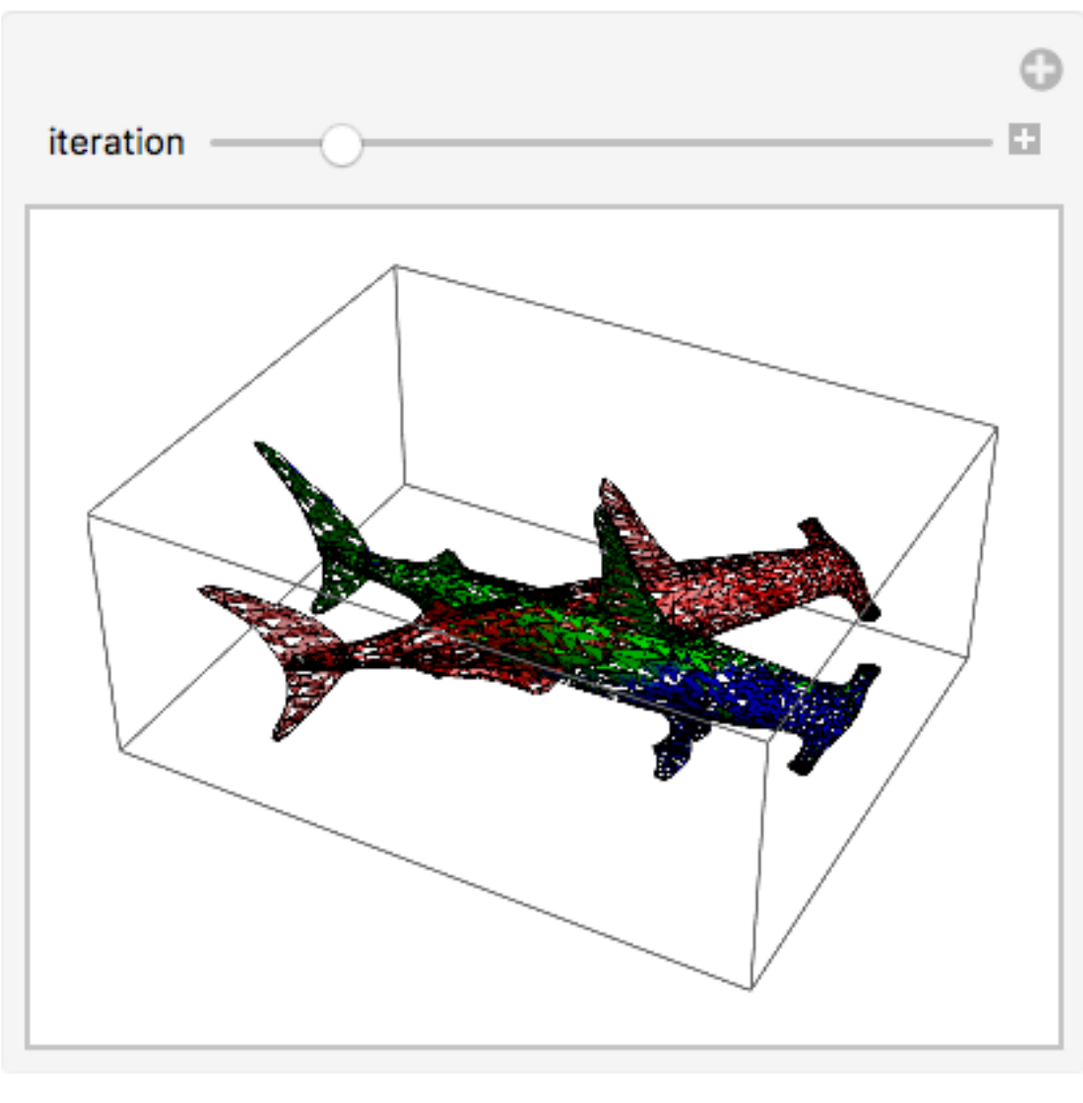
Cross (.off) Data set, 162 Vertices, Iteration = 1



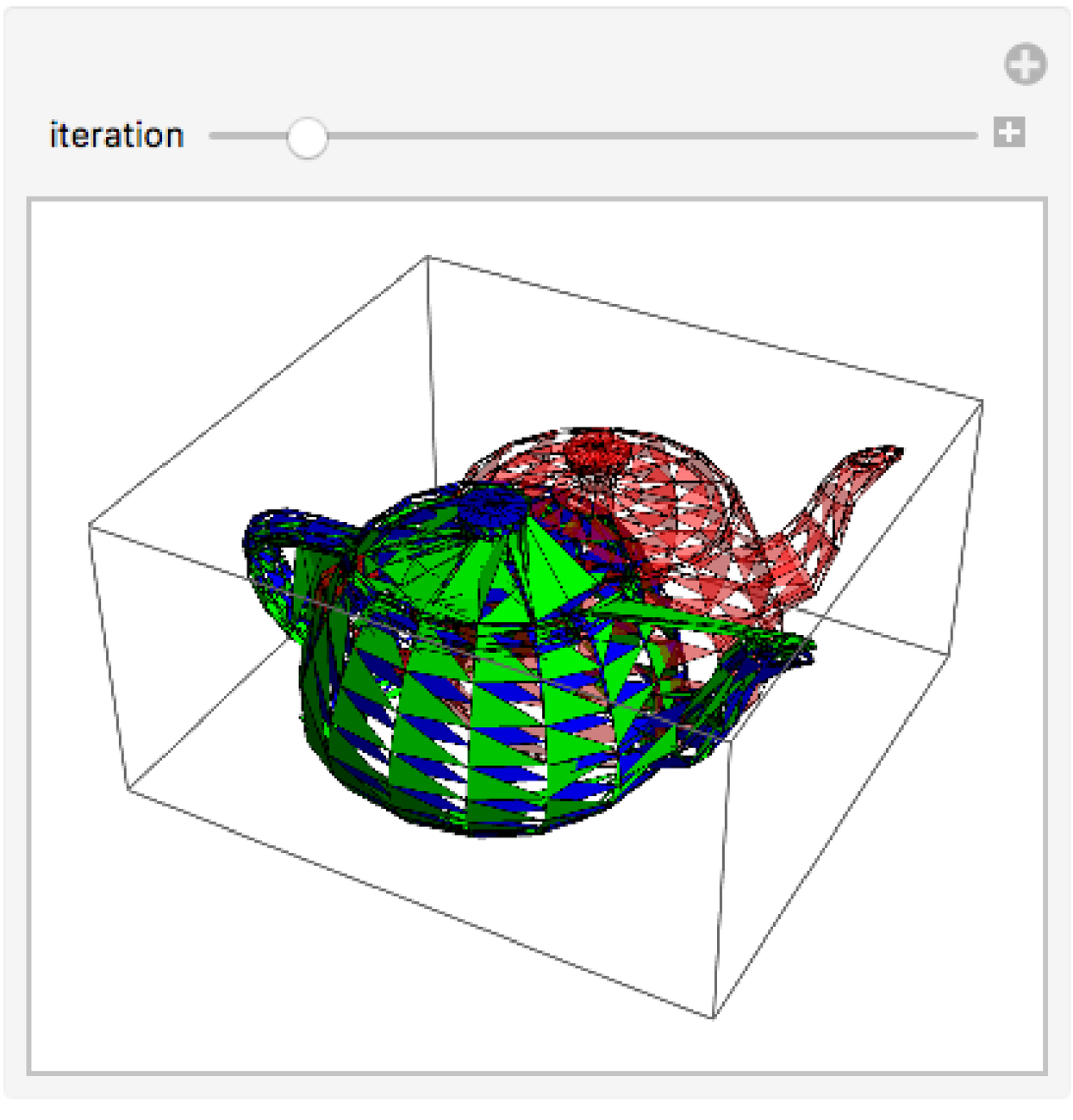
Humanoid (.obj) Data set, 64 Vertices, Iteration = 97



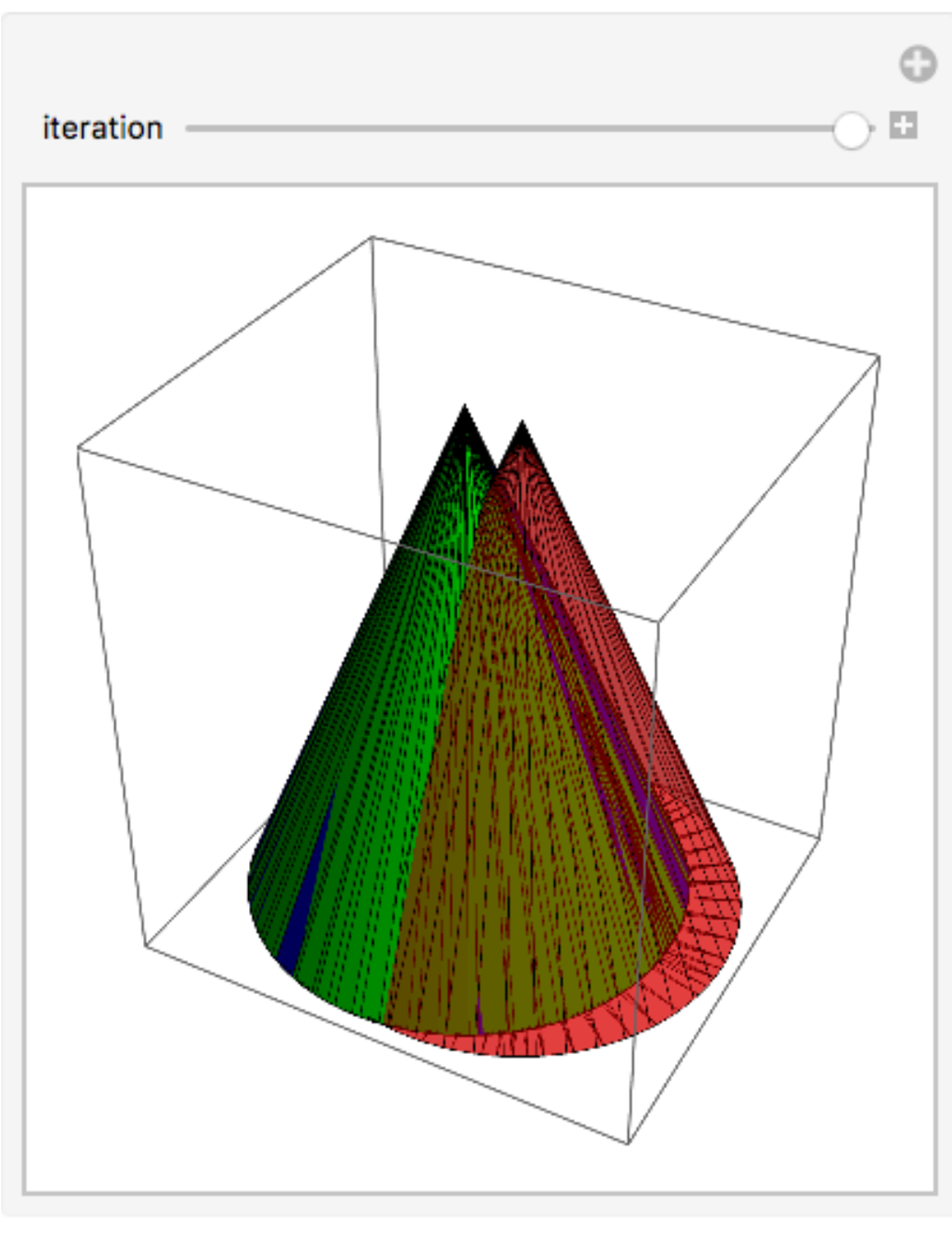
Violin Case (.obj) Data set, 1080 Vertices, Iteration = 45



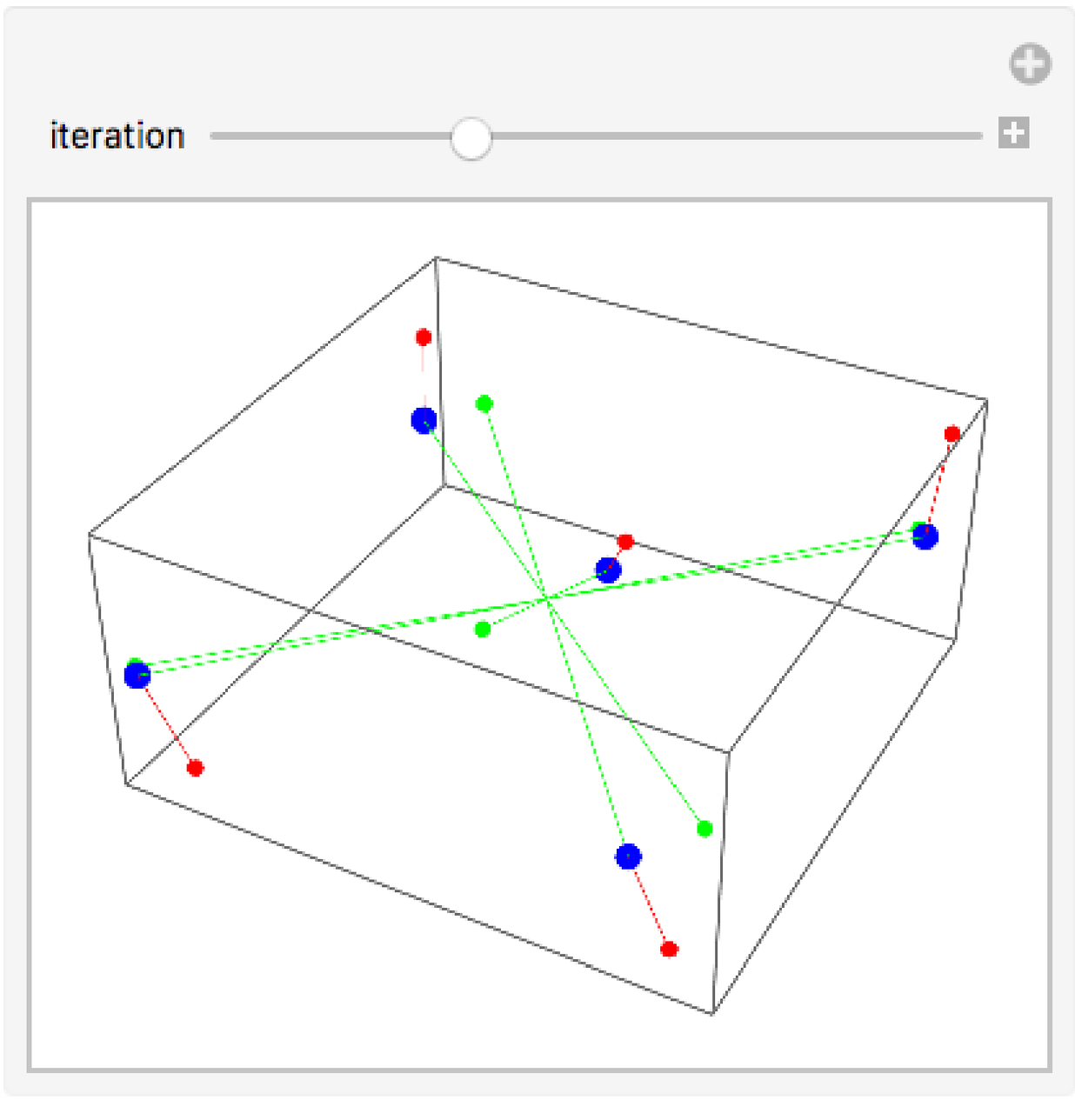
Hammerhead Shark (.off) Data set, 2564 Vertices, Iteration = 8



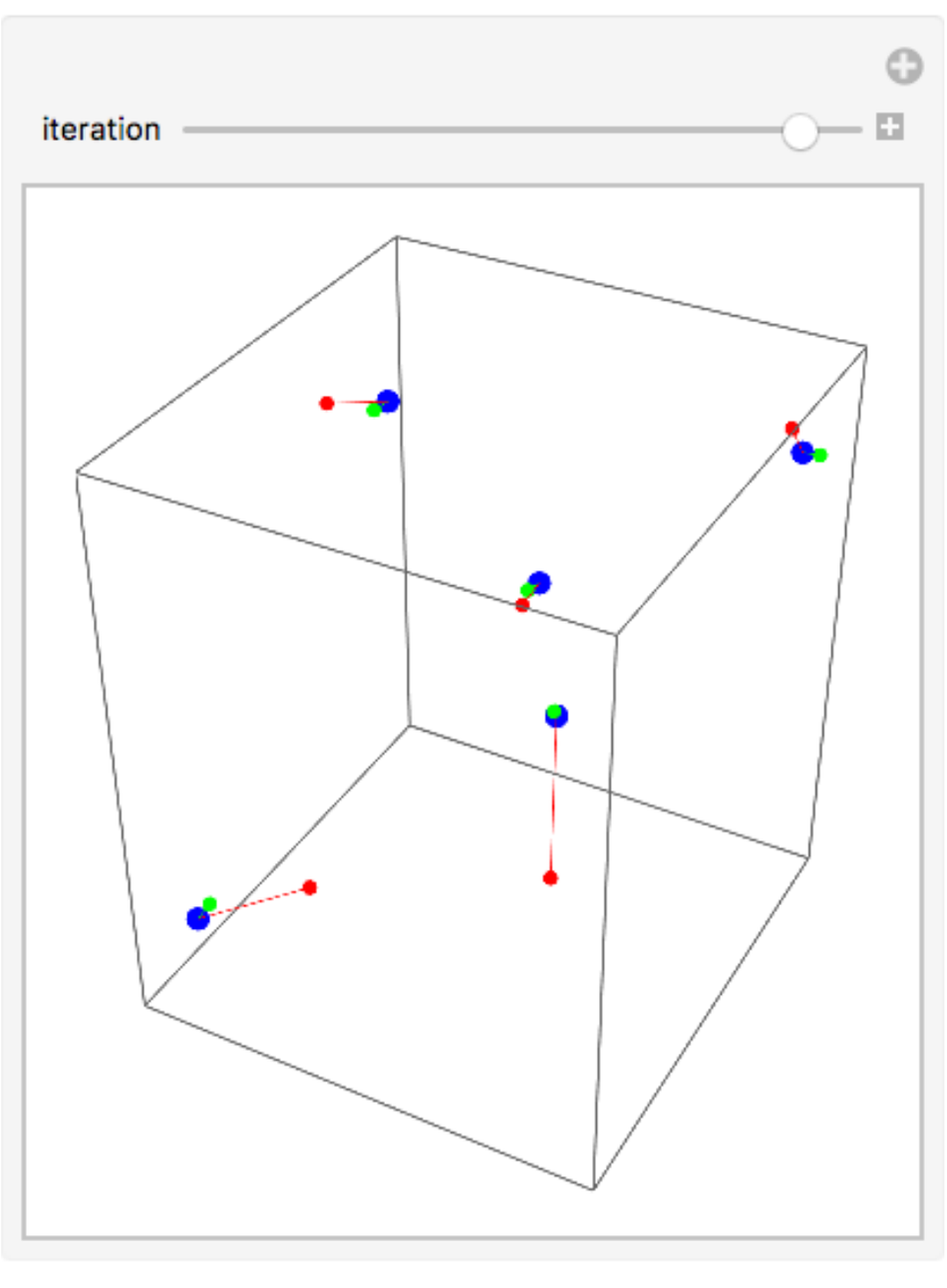
Utah TeaPot (.off) Data set, 480 Vertices, Iteration = 12



Cone (.off) Data set, 403 Vertices, Iteration = 40



2D Point Data set, 5 Vertices, Iteration = 19



3D Point Data set, 5 Vertices, Iteration = 47

C) Mean Squared Error Analysis: Convergence Theorem

The most important ideas regarding the convergence theorem are a) least square registration generally reduces the average distance between corresponding points during each iteration, and b) the closest point determination generically reduces the distance for each each individually. I will not go into the proof of the theorem which is described in Section IVB of the Besl and McKay paper. The mean squared error e_k of the correspondence is given as,

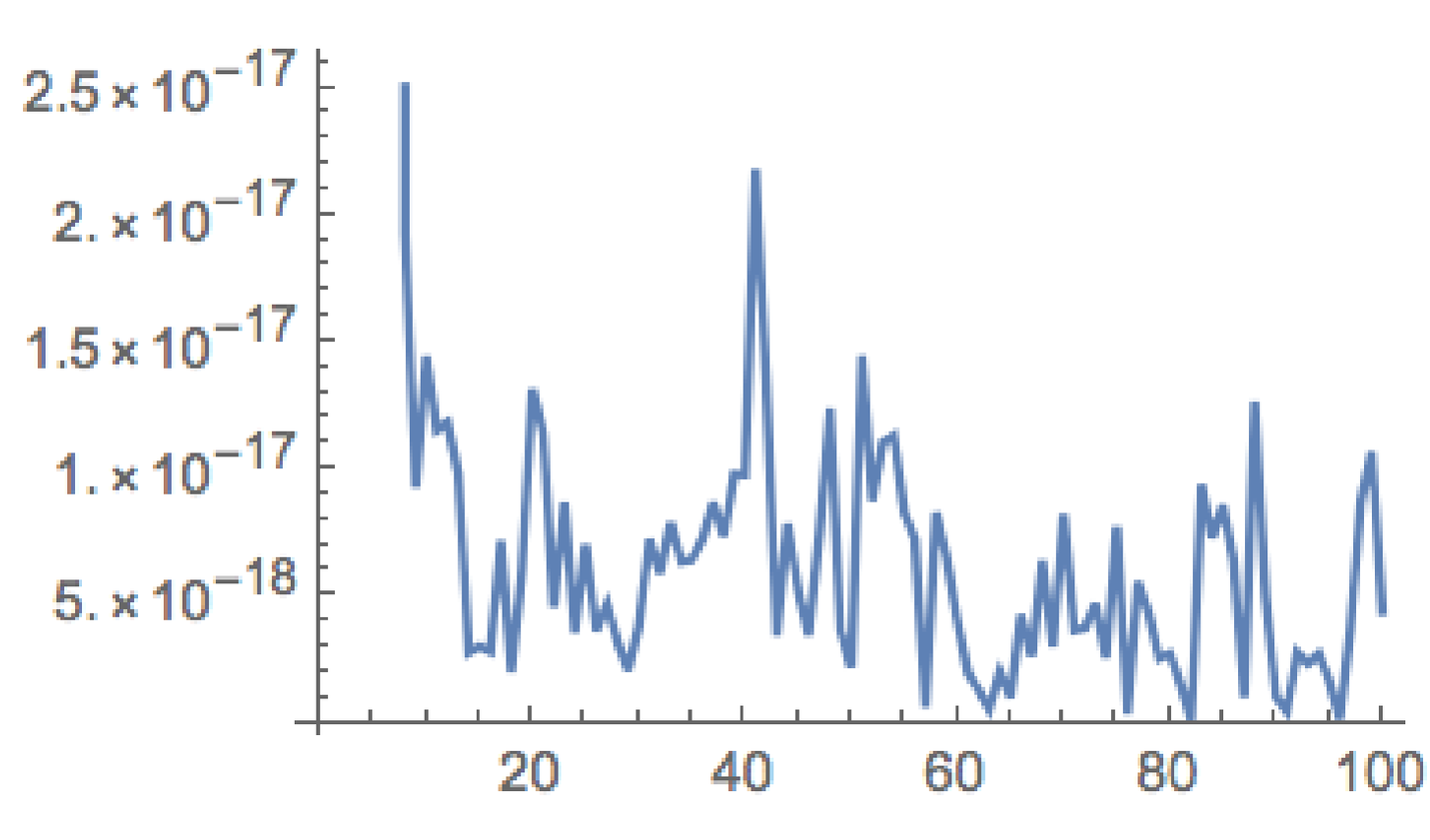
$$e_k = 1/N_p\ \Sigma\ ||y_{ik} - p_{ik}||^2 \text{ where } i \in [1, N_p]$$

The Q operator is applied and we get q_k and d_k from the correspondence, using which the mean square error between the two point data set is d_k,

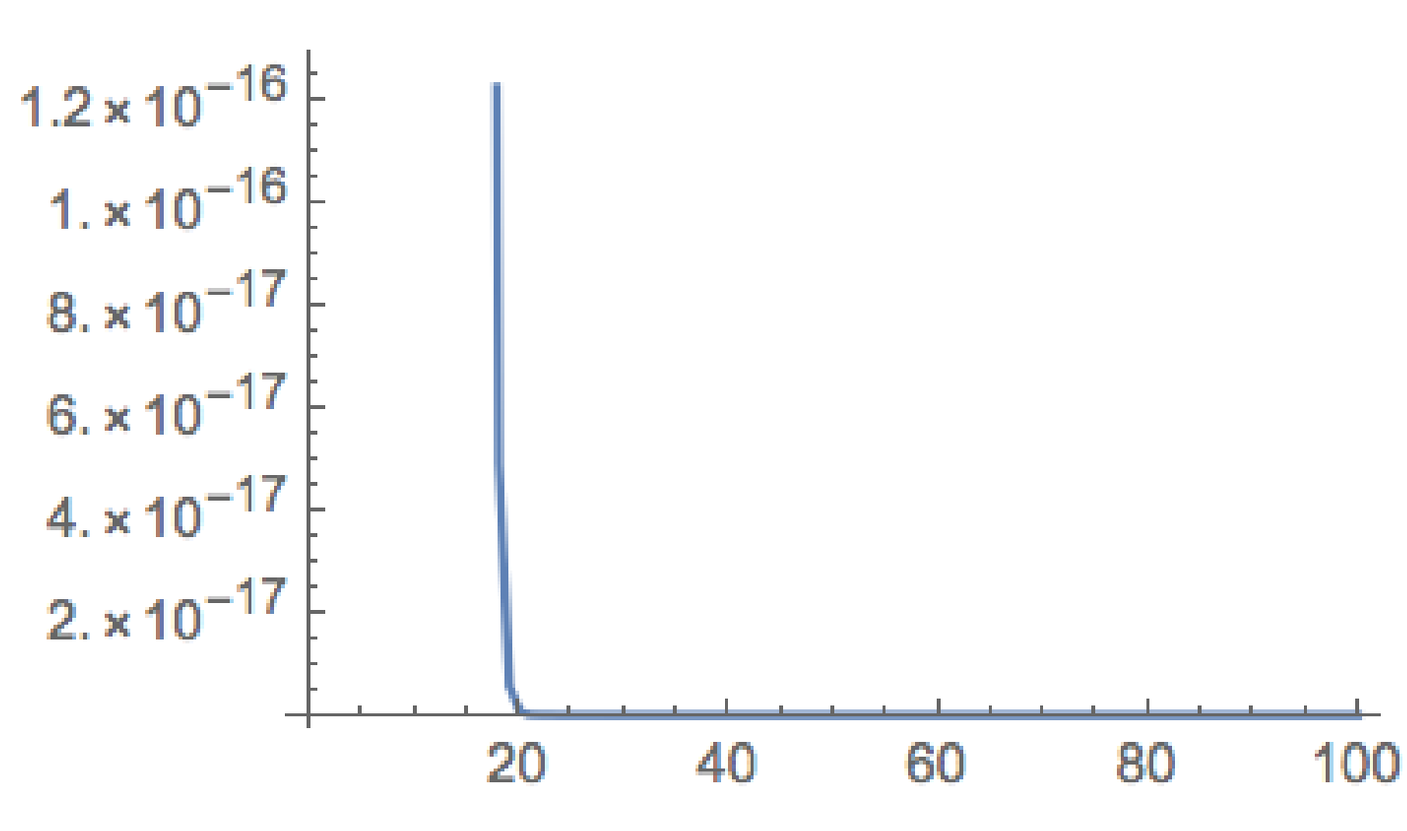
$$d_k = 1/N_p\ \Sigma\ ||y_{ik} - p_{i,k+1}||^2 \text{ where } i \in [1, N_p]$$

In our implementation, the error variable gives us the difference between the two point data set, Ynew and Yold corresponding to y_{ik} and $p_{i,k+1}$ and sum sums up the square of errors. Finally, we calculate the mean sum of square error by dividing it by the number of point in the dataset and display it for each iteration. Additionally, we create a list of the MSE for each iteration and plot it. The plots obtained shows clearly how the algorithm converges with iterations.

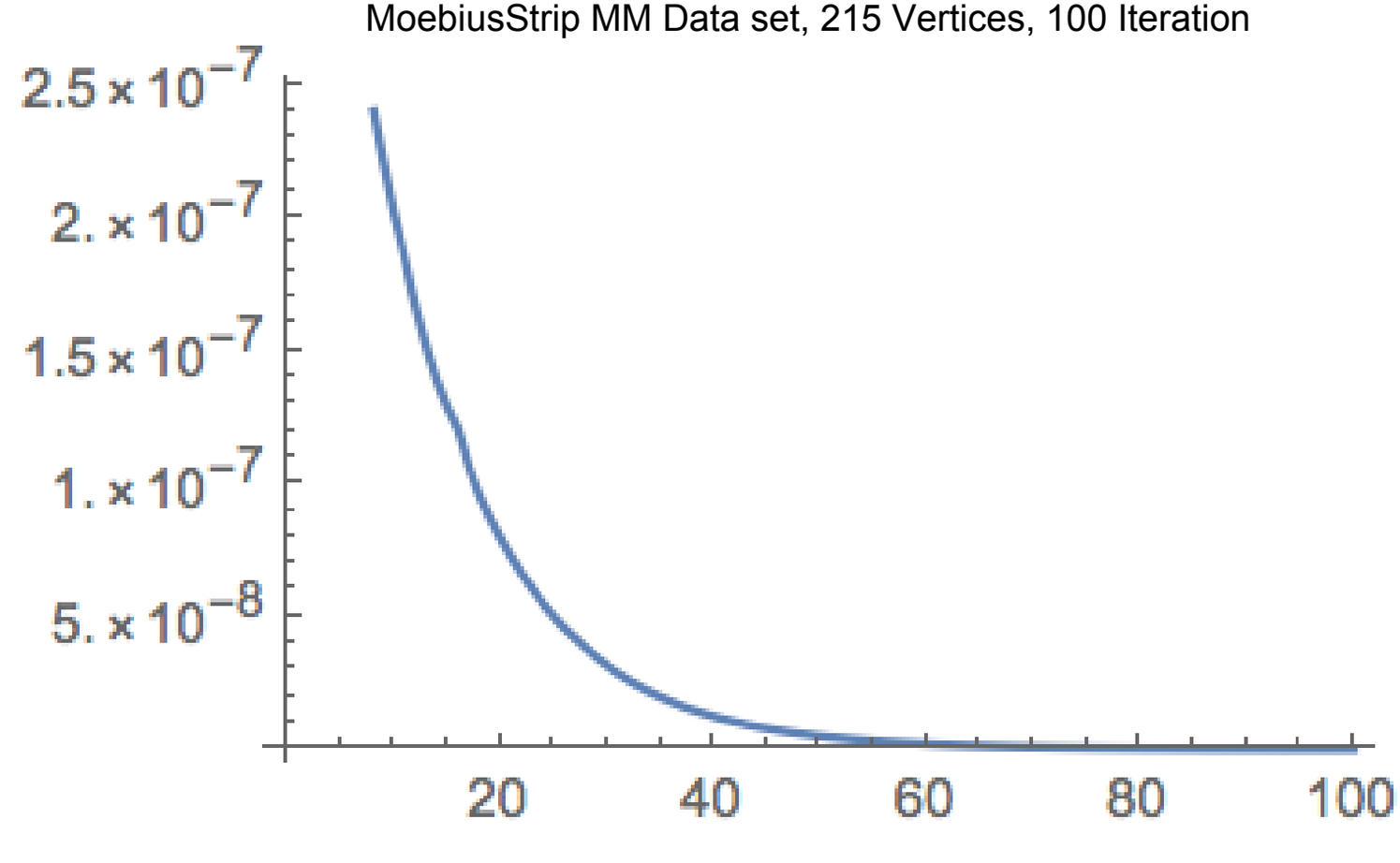
Some of the plots for the data sets are shown below.



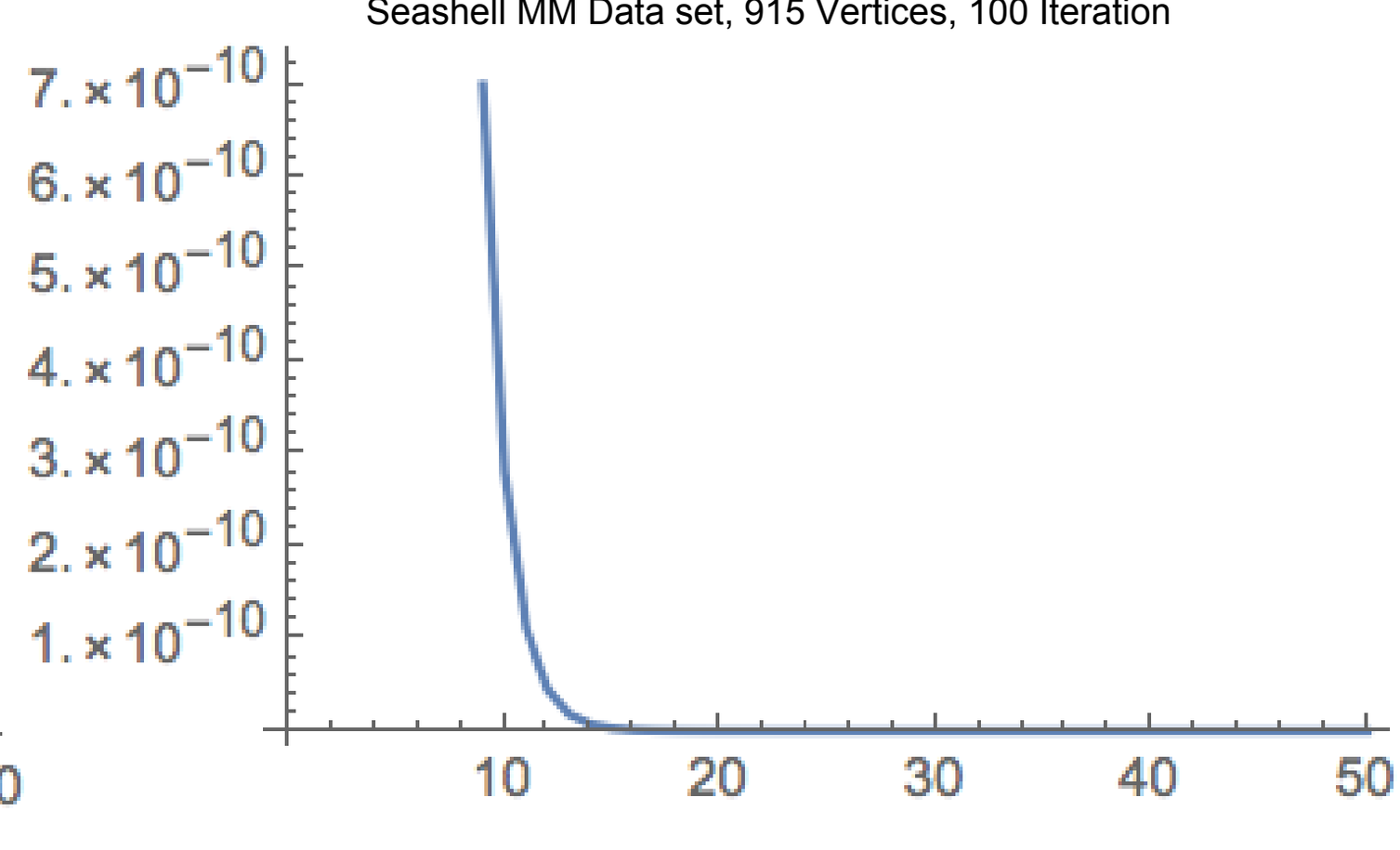
MoebiusStrip MM Data set, 215 Vertices, 100 Iteration



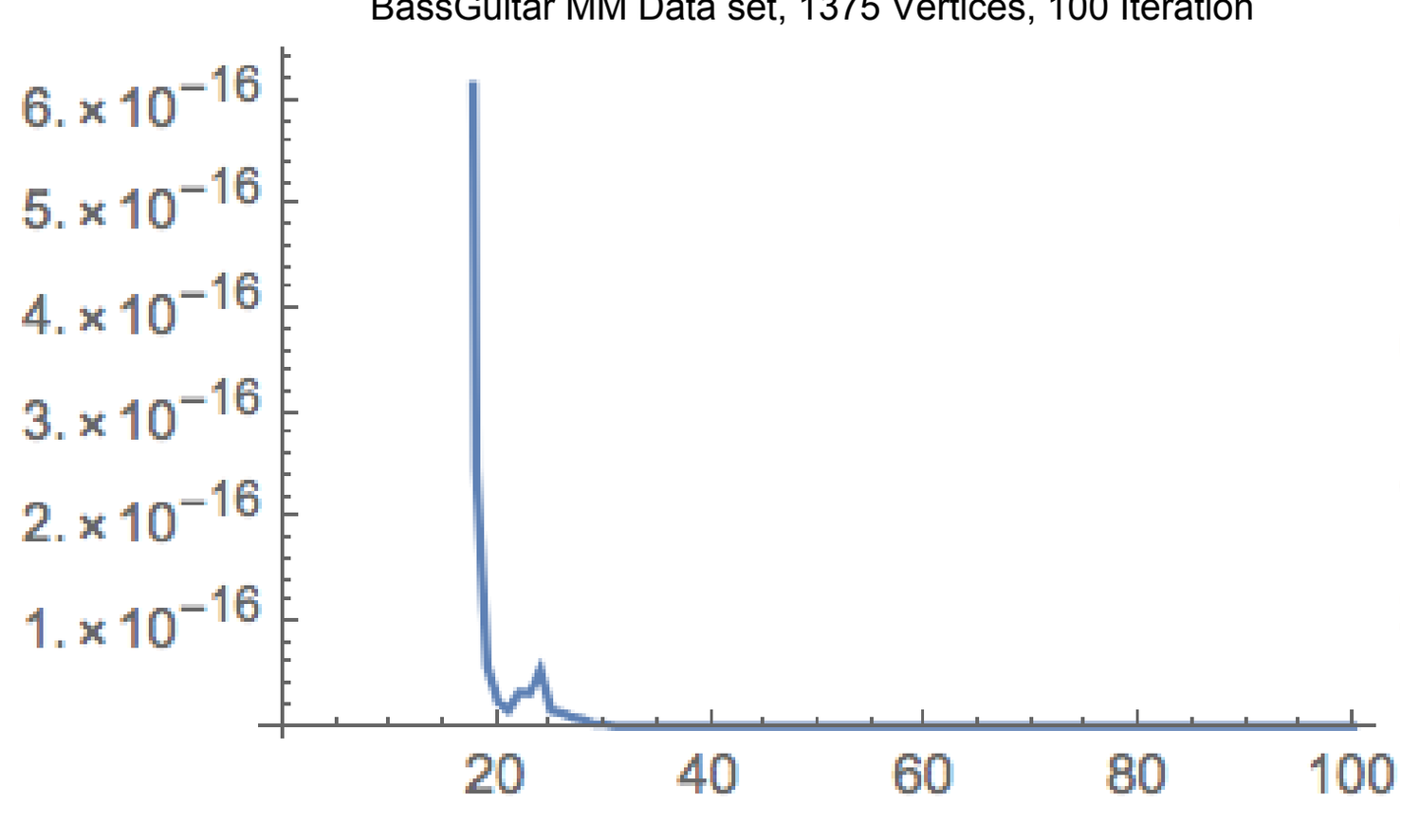
Seashell MM Data set, 915 Vertices, 100 Iteration



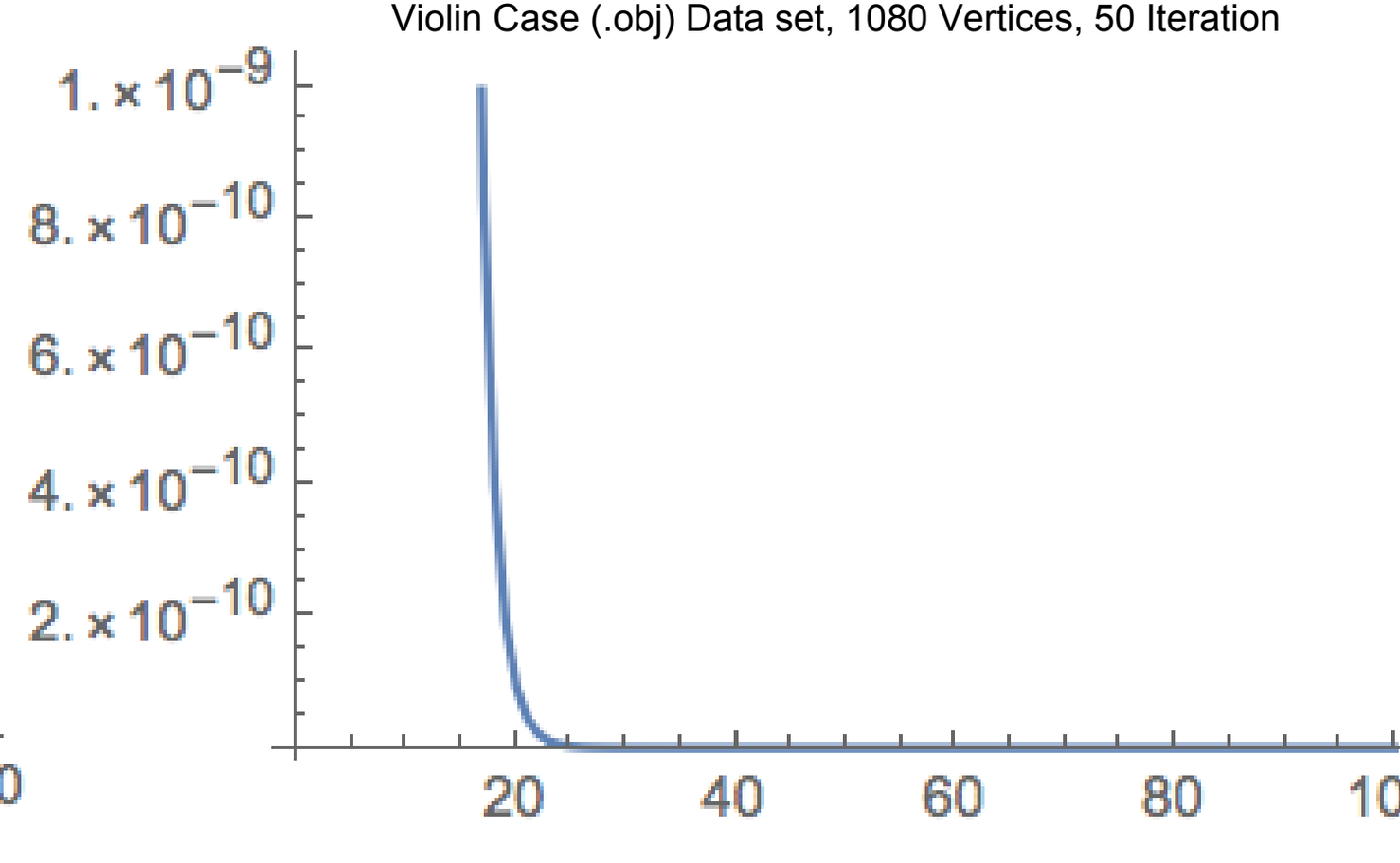
BassGuitar MM Data set, 1375 Vertices, 100 Iteration



Violin Case (.obj) Data set, 1080 Vertices, 50 Iteration



Random 3D Point Noisy Data set, 5 Vertices, 100 Iteration



Pear (.off) Data set, 867 Vertices, 100 Iteration

All these registration took less than 2-3 minutes* for 100 iterations. As the number of vertices increases, the time taken to compute the matrix increases. For a data set well under 1000-5000 vertices range, it takes approx. 2-5 minutes* for 50-100 iterations. We may note, in Besl & McKay paper, it is mentioned that we get excellent results somewhere between 30 and 50 iterations with d_k = 0.1% of model or shape size. We find out these results aptly match our observations, proving the convergence and the correctness of the implementation.

*All approximate times are given for execution done on a single-processor 2.7Ghz Intel Core i5 Processor, with 8GB RAM

Reference

1 Besl, Paul J., and Neil D. McKay. "Method for registration of 3-D shapes." Robotics-DL tentative. International Society for Optics and Photonics, 1992.

2 Horn, Berthold KP. "Closed-form solution of absolute orientation using unit quaternions." JOSA A 4.4 (1987): 629-642.