Hit and Run and Stuff

Brian Cohn May Szedlák

March 24, 2015

Abstract

The brain must select its control strategies among an infinite set of possibilties, thereby solving an optimization problem. While this set is infinite and lies in high dimensions, it is bounded by kinematic, neuromuscular, and anatomical constraints, within which the brain must select optimal solutions. We use data from a human index finger with 7 muscles, 4DOF, and 4 output dimensions. For a given force vector at the endpoint, the feasible activation space is a 3D convex polytope, embedded in the 7D unit cube. It is known that explicitly computing the volume of this polytope can become too computationally complex in many instances. We generated random points in the feasible activation space using the Hit-and-Run method, which converged to the uniform distribution. After generating enough points, we computed the distribution of activation across each muscle, shedding light onto the structure of these solution spaces- rather than simply exploring their maximal and nimimal values. We also visualize the change in these activation distributions as we march toward maximal feasible force production in a given direction. Using the parallel coordintes method, we visualize the connection between the muscle activations. Once can then explore the feasible activation space, while constraining certain muscles. Although this paper presents a 7 dimensional case of the index finger, our methods extend to systems with up to at least 40 muscles. We challenge the community to map the shapes distributions of each variable in the solution space, thereby providing important contextual information into optimization of motor cortical function in future research.

1 Author Summary

2 Introduction

Optimal control of a musculoskeletal system is intrinsically related to mechanical constraints. An endpoint's end effector forces are highly dependent upon tendon force ranges, the leverage of each tendon insertion point across each joint, and the planes of motion each degree of freedom (DOF). In spite of the complexity of alpha-gamma neuromuscular drive models, every system exists under limitations intrinsic to any moving system. As such, limbs have been modeled to behave under these constraints with stunning realism[cite], and have been accurate in predicting kinetics when different muscles are activated [todorov's mujoco]. Some attention has been given to the constraints that physical systems impart on control itself ['nice try' citations]. As limbs exist under physical constraints, neuromuscular control must strategize within the same laws of physics. A breadth of modeling techniques have been applied to physical systems to model and understand CNS control under the constaints of a given task, and many have been able to visualize some of the limitations animals must abide by in optimization.

Optimal control theory must be implemented in a way such that it is computationally tractable. Control systems of designed (robotic) and evolved (neurophysiologic) origins can afford only a small measure of latency. Identifying how optimal control works within the framework of constraints could bring rise to more efficient algorithms, and this contextual understanding could introduce new ways to visualize how neuromuscular systems learn to improve over training.

In dynamic systems we have seen ¡do research on this¿[cite].

In a static system, every possible combination of independent muscle activations exists within the unit-n-cube, where N is set to however many independently-controlled muscles a system has. Prior work has highlighted the relationship between the feasible force space and the set of all activation solutions.[cite papers in the last 10 years] In effect, adding constraints on the FFS (e.g. requiring only force in a given plane) adds constraints to the FAS

The effect of each muscle on each joint has been represented by the moment arm matrix [citations], the relationship of each DOF on end-effector output directions The feasible force set (described in detail in [cite]) is an M-dimensional polytope containing all possible force vectors an endpoint can output.

neurons do alot of stuff, and much work has been put into understanding how neural drive results in force, motion, and kinetics. physical description of a musculoskeletal system

Functional performance is defined by the ability for a system to identify optimal solutions in a set of suboptimal solutions. ¡talk about local and global maxima and minima in neuro optimization control theory¿

The feasible force set represents every possible output force an end effector can impart on an endpoint.

Described in a mathematical way the feasible activation set is expressed as follows.

For a given force vector $f \in \mathbb{R}^m$, which are the activations that satisfy

$$\mathbf{f} = A\mathbf{a}, \mathbf{a} \in [0,1]^n?$$

In our 7-dimensional example $m = 4$ and $n = 7$, typically n is much larger than m. The constraint $\mathbf{a} \in [0,1]^n$ describes that the feasible activation space lies in the n-dimensional unit cube (also called the n-cube). Each row of the constraint $\mathbf{f} = A\mathbf{a}$ is a $n-1$ dimensional hyperplane. Assuming that the rows in A are linearly independent (which is a safe assumption in the muscle system case), the intersection of all m equality constraints constraints is a $(n-m)$-dimensional hyperplane. Hence the feasible activation set is the polytope given by the intersection of the n-cube and an $(n-m)$-dimensional hyperplane. Note that this intersection is empty in the case where the force f can not be generated.

We first describe the stochastic method of hit-and-run, and illustrate its use on a fabricated 3-muscle, 1-DOF system with a desired force output of 1N. We designed this schematic (but mathematically viable) linear system of constraints to help readers understand the mechanics of hit-and-run mathematics. Our index-finger model has too many dimensions to show how the process works, so we hope this will help readers understand what is going on in n dimensions (7 in the case of the index-finger model). We also used this model to perform unit tests on our code in thoroughly validating our hit-and-run implementation.

We investigated the distributions of the feasible activation set across each muscle. State the purpose of the work in the form of the hypothesis, question, or problem you investigated; and, Briefly explain your rationale and approach and, whenever possible, the possible outcomes your study can reveal.

3 Materials and Methods

3.1 Polytope representation of the feasible activation space

Exact volume calculations for polygons can only be done in reasonable time in up to 10 dimensions [2, 6, 7]. We therefore use the so called Hit-and-Run approach, which samples a series of points in a given polygon. Given the points for a feasibale activation space, this method gives us a deeper understanding of its underlying structure.

3.2 Hit-and-Run

In this section we introduce the Hit-and-Run algorithm used for uniform sampling in a convex body K, was introduced by Smith in 1984 [10]. The mixing time is known to be $\mathcal{O}^*(n^2R^2/r^2)$, where R and r are the radii of the inscribed and cicumscribed ball of K respectively [1, 8]. I.e., after $\mathcal{O}^*(n^2R^2/r^2)$ steps of the Hit-and-Run algorithm we are at a uniformly at random point in the convex body. In the case of the muscles of a limb, we are interested in the polygon P that is given by the set of all possible activations $\mathbf{a} \in \mathbb{R}^n$ that satisfy

$$\mathbf{f} = A\mathbf{a}, \mathbf{a} \in [0,1]^n,$$

where $\mathbf{f} \in \mathbb{R}^m$ is a fixed force vector and $A = J^{-T}RF_m \in \mathbb{R}^{m\times n}$. P is bounded by the unit n-cube since all variables a_i, $i \in [n]$ are bounded by 0 and 1 from below, above respectively. Consider the following 1×3 example.

$$1 = \frac{10}{3}a_1 - \frac{53}{15}a_2 + 2a_3$$
$$a_1, a_2, a_3 \in [0, 1],$$

the set of feasible activations is given by the shaded set in Figure 1.

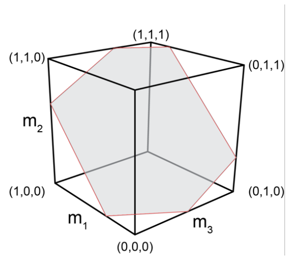


Figure 1: Feasible Activation

The Hit-and-Run walk on P is defined as follows (it works analogously for any convex body).

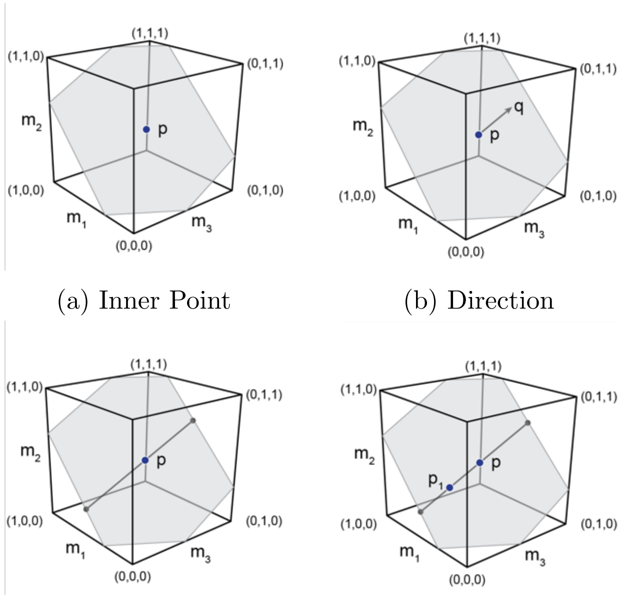
1. Find a given starting point $\mathbf{p}$ of P (Figure 2a) .
2. Generate a random direction through $\mathbf{p}$ (uniformly at random over all directions) (Figure 2b).
3. Find the intersection points of the random direction with the n-unit cube (Figure 2c).
4. Choose the next point of the sampling algorithm uniformly at random from the segment of the line in P (Figure 2d).
5. Repeat from (b) the above steps with the new point as the starting point .

The implementation of this algorithm is straight forward except for the choice of the random direction. How do we sample uniformly at random (u.a.r.) from all directions in P? Suppose that $\mathbf{q}$ is a direction in P and $p \in P$. Then by definition of P, $\mathbf{q}$ must satisfy $\mathbf{f} = A(\mathbf{p} + \mathbf{q})$. Since $\mathbf{p} \in P$, we know that $\mathbf{f} = A\mathbf{p}$ and therefore

$$\mathbf{f} = A(\mathbf{p} + \mathbf{q}) = \mathbf{f} + A\mathbf{q}$$

and hence

$$A\mathbf{q} = 0.$$



(a) Inner Point (b) Direction

(c) Endpoints (d) New Point

Figure 2: Hit-and-Run Step

We therefore need to choose directions uniformly at random from all directions in the vectorspace

$$V = \{\mathbf{q} \in \mathbb{R}^n | A\mathbf{q} = 0\}.$$

As shown by Marsaglia this can be done as follows [9].

1. Find an orthonormal basis $b_1, \ldots, b_r \in \mathbb{R}^n$ of $A\mathbf{q} = 0$.
2. Choose $(\lambda_1, \ldots, \lambda_r) \in \mathcal{N}(0,1)^n$ (from the Gaussian distribution).
3. $\sum_{i=1}^{r} \lambda_i b_i$ is a u.a.r. direction.

A basis of a vectorspace V is a minimal set of vectors that generate V, and it is orthonormal if the vectors are pairwise orthogonal (perpendicular) and have unit length. Using basic linear algebra one can find a basis for $V = \{A\mathbf{q} = 0\}$ and orthogonalize it with the well known Gram-Schmidt method (for details see e.g. [3]). Note that in order to get the desired u.a.r. sample the basis needs to be orthonormal. For the limb case we can safely assume that the rows of A are linearly independent and hence the number of basis vectors is $n - m$.

3.3 Mixing Time

How many steps are necessary to reach a uniformly at random point in the polytope? The theoretical bound $\mathcal{O}^*(n^2R^2/r^2)$ given in [8] has a very large hidden coefficient (10^30) which makes the algorithm almost infeasible in lower dimensions.

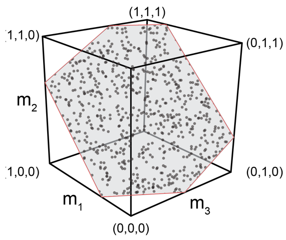
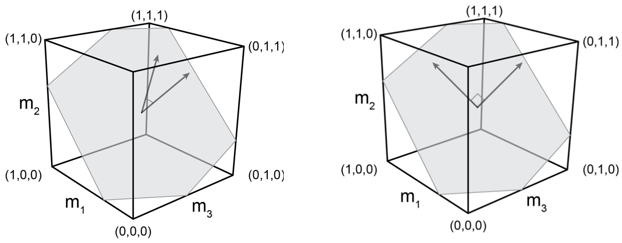


Figure 3: Uniform Distribution



(a) Some Basis (b) Direction

Figure 4: Find Orthonormal Basis

These bounds hold for general convex sets. For convex polygons in higher dimensions, experimental results suggest that $\mathcal{O}(n)$ steps of the Hit-and-Run algorithm are sufficient. In particular Emiris and Fisikopoulos paper suggest that $(10 + 10\frac{n}{)}n$ steps are enough to have a close to uniform distribution [4]. In all cases tested, sampling more point did not make accuarcy significantly higher.

Ge et al. showed experimentally that up to about 40 dimensions, ??? random points seem to suffice to get a close to uniform discussion [5].

Therfore for given output force we execute the Hit-and-Run algorithm 1000 times on 100 points. The experimental results propose that those 1000 points are uniformly distributed on the polygon.

As a additional control, for each muscle we observe that the theoretical upper and lower bound of the feasible activation match the observed corresponding bounds (difference max ??). To find the theoretical upperbound (lowerbound) of a given muscle activation we solve two linear programs maximizing (minimizing) a_i over the polytope.

3.4 Starting Point

To find a starting point in

$$\mathbf{f} = A\mathbf{a}, \mathbf{a} \in [0,1]^n,$$

we only need to find a feasible activation vector. For the hit and run algorithm to mix faster, we do not want the starting point to be in a vertex of the activation space. We use the following standard trick using slack variables ϵ_i.

$$\begin{array}{lrcl} \text{maximize} & \sum_{i=1}^{n} \epsilon_i & & \\ \text{subject to} & \mathbf{f} & = & A\mathbf{a} \\ & a_i & \in & [\epsilon_i, 1-\epsilon_i], \quad \forall i \in \{1, \dots, n\} \\ & \epsilon_i & \geq & 0, \quad \forall i \in \{1, \dots, n\}. \end{array} \tag{1}$$

This approach can still fail in theory, but this method has the choose $\epsilon_i > 0$ and therefore $a_i \neq 0$ or 1. Since for all vertices of the feasible activation space lie on the boundary of the n-cube, at least $n - m$ muscles must have activation 0 or 1. Documentation is included in our supplementary information.

3.5 Parallel Coordinates: Visualization of the Feasible Activation Space

Citation A common way to visualize higher dimensional data is using parallel coordinates. To show our sample set of points in the feasible activation space we draw n parallel lines, which representing the activations of the n muscles. Each point is then represented by connecting their coordinates by $n-1$ lines.

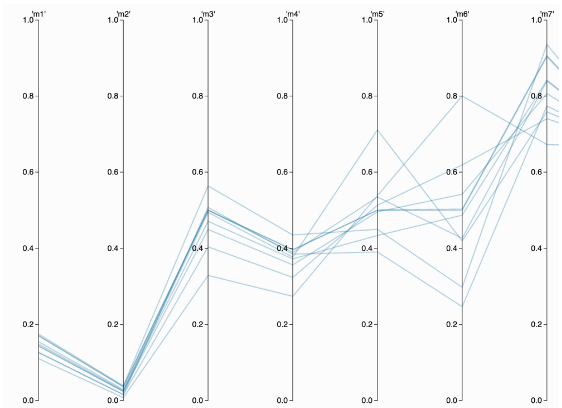


Figure 5: Feasible Activation

Using an interactive surface one can now restrict each muscle function to any desired interval, e.g., figure ??.

NICE FIGURE OF RESTRICTED PARALLEL COORDINATES

For the l_1, l_2 and l_3 norm respectively, we added an additional line to represent the corresponding weight. E.g. for a given point $\mathbf{a} \in \mathbb{R}^n$ we are interested in $\sum_{i=1}^{n} a_i$, $\sqrt{\sum_{i=1}^{n} a_i^2}$ and $\sqrt[3]{\sum_{i=1}^{n} a_i^3}$. As for the muscles one can restrict the intervals of the weight functions, to explore the corresponding feasible activation space.

NICE PICTURE WITH WEIGHTS INCLUDED

4 Results

Many nice figures

1. Histograms
2. Histograms 3 directions
3. PC

4.1 Activation Distribution on a Fixed Force Vector

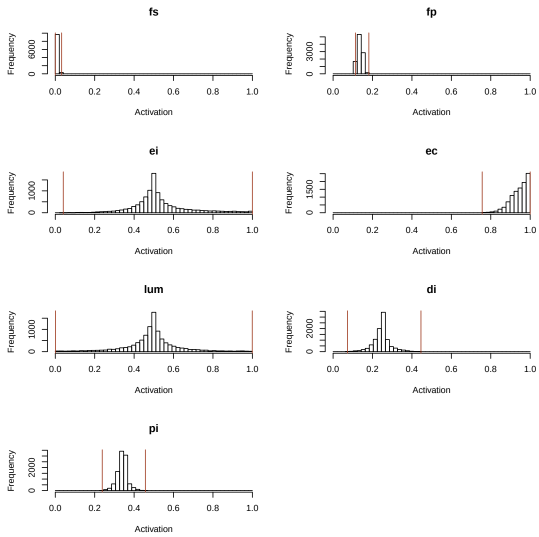


Figure 6: Histogram for Fixed Force

4.2 Changing Output Force in 3 Directions

We discuss different forces into three different directions, which are given by the palmar direction (x-direction), the distal direction (y-direction) and the sum of them. The maximal forces into each direction are given by ??, ?? and ?? respectively. For $\alpha = 0.1, 0.2, \ldots, 0.9$, we give the histograms where the force is $\alpha \cdot F_{\max}$, where $F_{\max}$ is the maxium output force in the corresponding direction.

4.3 Parallel Coordinates

Muscle 5 and 6 same direction and same strength $\Rightarrow$ Does not matter which one we activate for low cost

5 Discussion

Mostly to be written by Brian

5.1 Distributions

- Bounding box away from 0 and 1 means muscle is really needed $\rightarrow$ Already known from the bounding boxes
- High density $\rightarrow$ most solutions in that area

5.2 Parallel Coordinates

- Parallel lines in PC indicate opposite direction of muscles
- Crossing lines indicate similar direction

5.3 Running Time

The step of the algorithm which are time consuming are finding a starting point, which solves a linear program and can take exponential running time in worst case. For each fixed force vector we only have to find a starting point and an orthonormal basis once, and are hence not of concern for the running time.

Running one loop of the hit and run algorithm only needs linear time, therefore the method will extend to higher dimesions with only linear factor of additinal running time needed.

TODO: add part on how we didn't deal with a dynamical system, but the feasible force space still exists, just with constraints on muscle activations instantaneously, and the momentums present in each of the limbs about their joints.

TODO: discuss how the FAS is an overestimate of the total number of possible activation solutions, because of synergies. Our mathematical approach considers 100 percent independent control of every muscle, with no correlations or inverse correlations between muscle activations. Then slightly discuss how synergies can limit activation capabilities.

6 Acknowledgments

References

[1] M. Dyer, A. Frieze, and R. Kannan. A random polynomial time algorithm for approximating the volume of convex bodies. *Proc. of the 21st annual ACM Symposium of Theory of Computing*, pages 375–381, 1989.

[2] M. Dyer and M. Frieze. On the complexity of computing the volume of a polyhedron. *SIAM Journal on Computing*, 17:967–974, 1988.

[3] T. S. Blyth E. F. Robertson. *Basic Linear Algebra.* Springer, 2002.

[4] I.Z. Emiris and V. Fisikopoulos. Efficient random-walk methods for approximating polytope volume. *Preprint: arXiv:1312.2873v2*, 2014.

[5] C. Ge, F. Ma, and J. Zhang. A fast and practical method to estimate volumes of convex polytopes. *Preprint: arXiv:1401.0120*, 2013.

[6] L.G. Khachiyan. On the complexity of computing the volume of a polytope. *Izvestia Akad. Nauk SSSR Tekhn. Kibernet*, 216217, 1988.

[7] L.G. Khachiyan. The problem of computing the volume of polytopes is np-hard. *Uspekhi Mat. Nauk 44*, 179180, 1989.

[8] L. Lovász. Hit-and-run mixes fast. *Math. Prog.*, 86:443–461, 1998.

[9] G. Marsaglia. Choosing a point from the surface of a sphere. *Ann. Math. Statist.*, 43:645–646, 1972.

[10] R.L. Smith. Efficient monte-carlo procedures for generating points uniformly distributed over bounded regions. *Operations Res.*, 32:1296–1308, 1984.

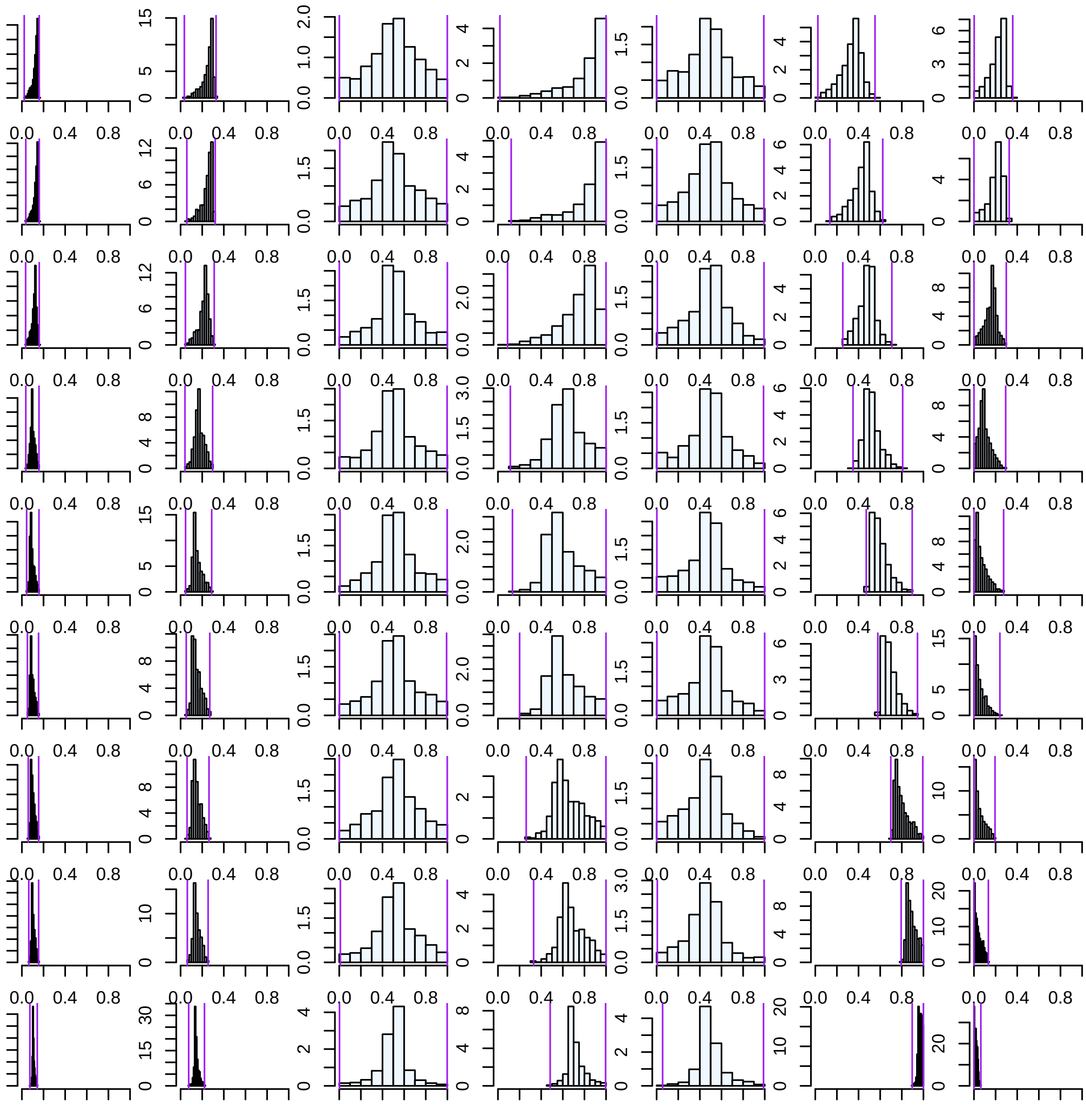


Figure 7: Histogram for x-Direction

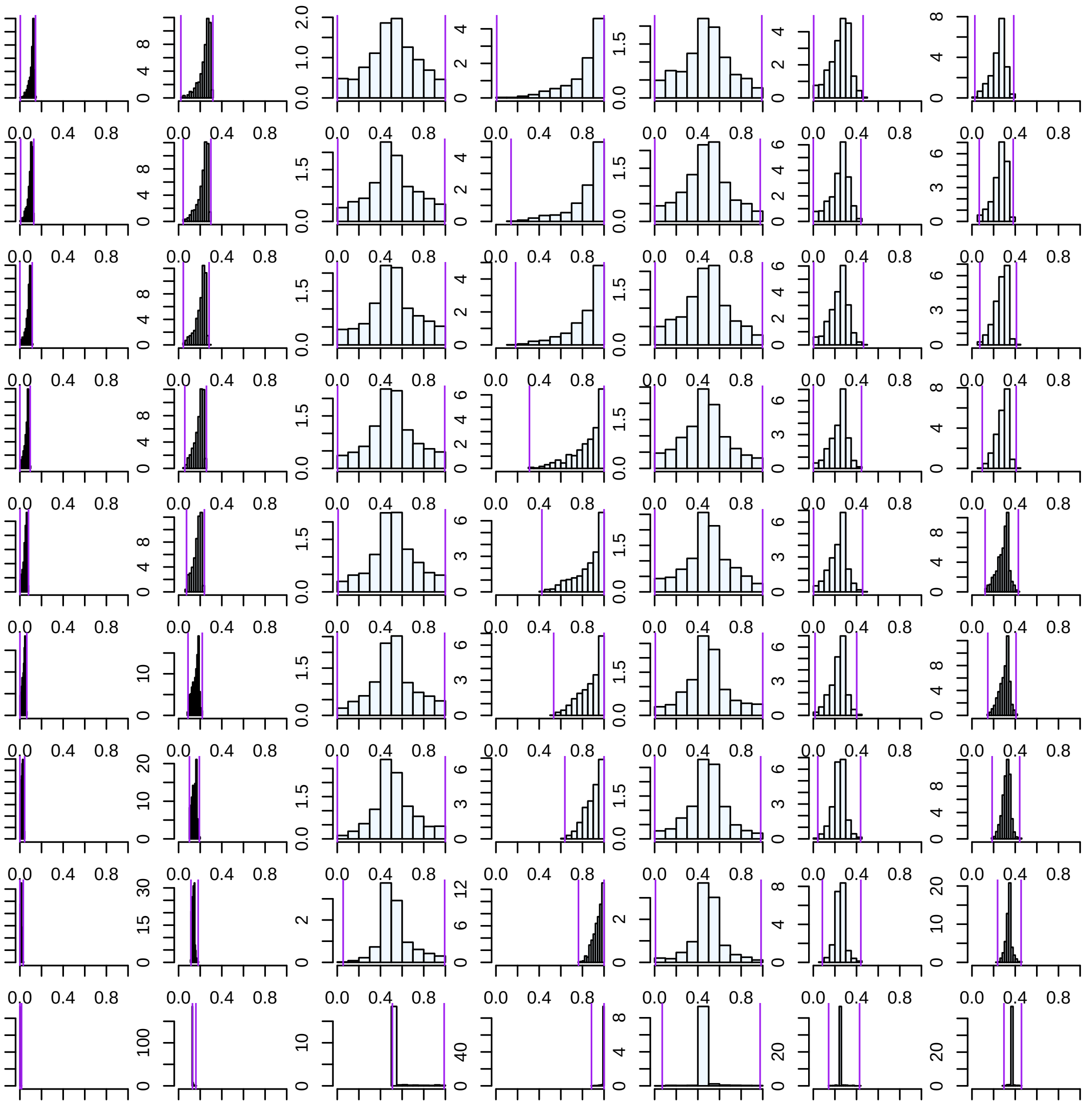


Figure 8: Histogram for xy-Direction

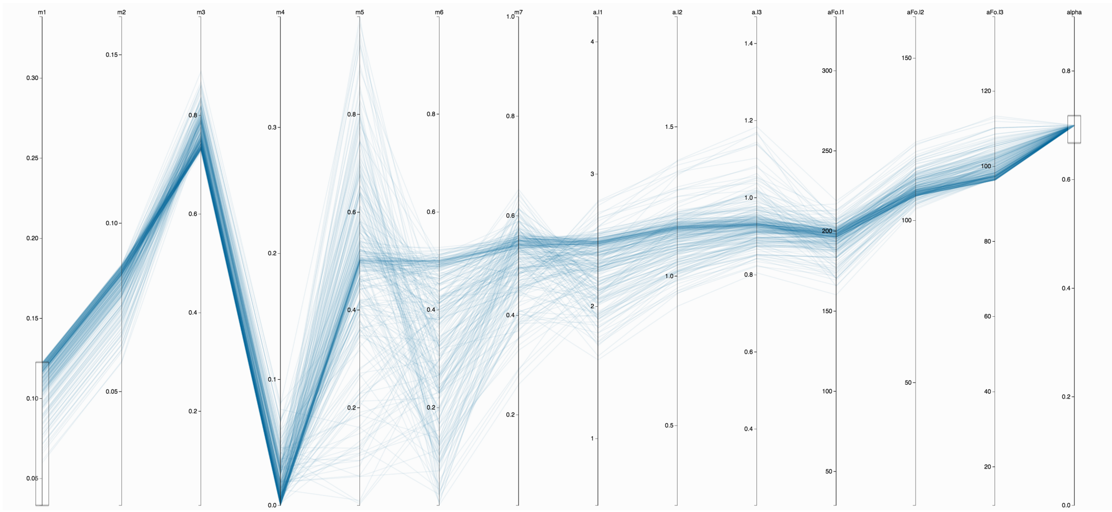


Figure 9: Low for Muscle 1

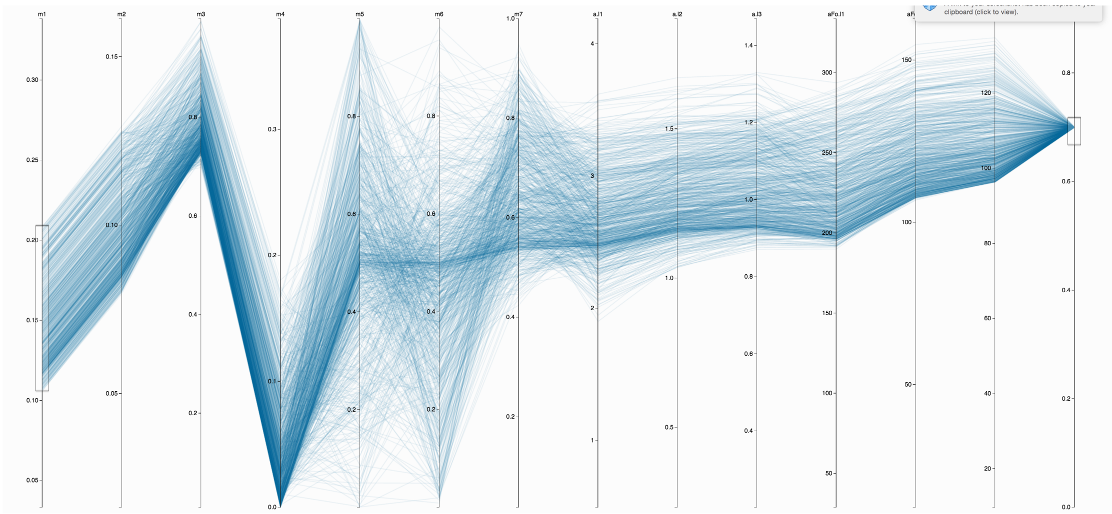


Figure 10: Middle for Muscle 1

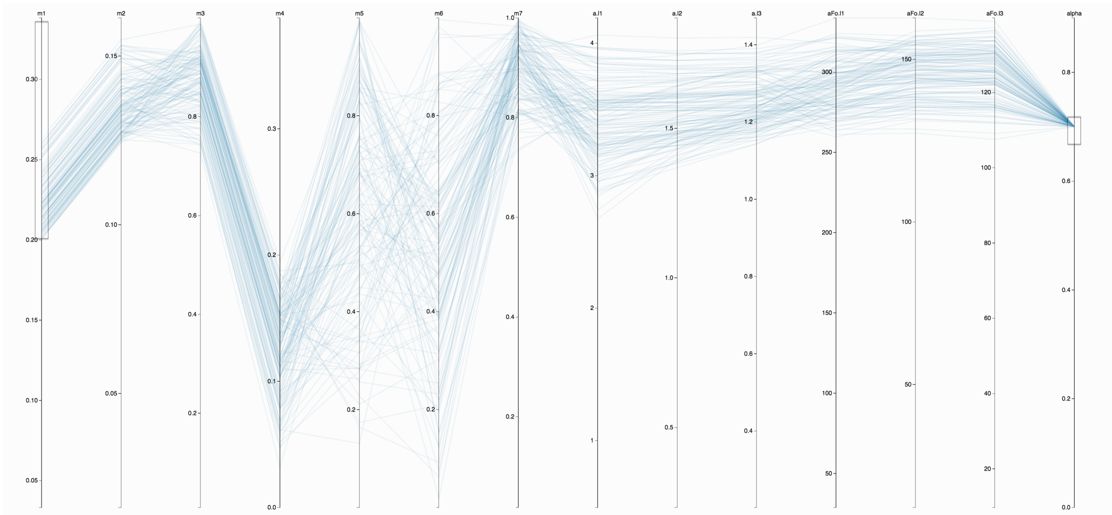


Figure 11: Upper for Muscle 1

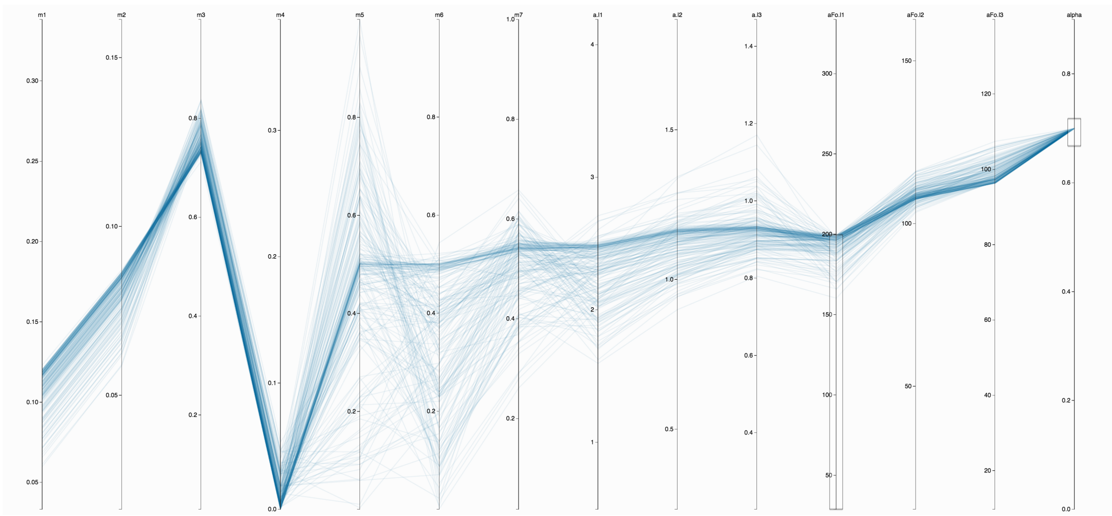


Figure 12: Weighted Cost