3 Current Address: Ronald Tutor Hall, RTH-404 3710 S. McClintock Ave
3 Current Address: Ronald Tutor Hall, RTH-404 3710 S. McClintock Ave Los Angeles, CA 90089-2905, USA
☯B.C. and M.S. contributed equally as first authors.
* valero@usc.edu

Abstract

The Hit-and-Run algorithm is defined as follows (it works analogously for any convex body even if not a polytope) [**?**]:

1. Find a point $\mathbf{p}$ in P to use as a starting point by using the linear programming method documented in the Supplementary Note.
2. Generate a line in a random direction q (uniformly-at-random over all directions) from $\mathbf{p}$ in P (Fig. 1a).
3. Find the two intersection points of the line given by the random direction q with the boundary of the polytope (Fig. 1b).
4. Choose a new point uniformly-at-random on the line segment between the intersection points (Fig. 1c).
5. Repeat the above steps from 2. onwards using the new point as the starting point, and generate a new random direction. Continue this process for s iterations, where s is the mixing number. The mixing number is the number of iterations that are expected to result in a point sampled uniformly at random from the space, with respect to the starting point (and any other point in P. Saving just the s^{th} point for many repetitions will result in many uniform-at-random points collected across the feasible activation space (Fig. 1).

Experimentally, it is known to converge to a uniform sampling across any convex body, up to about 40 dimensions [**?**]. The Hit-and-Run method is a generalization of a discrete Markov chain—where the goal is to identify multiple points from the feasible activation polytope P uniformly-at-random.

We describe the mathematical basis and details to replicate our implementation in the Supplementary Note, including our application of slack variables to select a valid starting point within the feasible activation space.

Supplementary Note: Algorithmic Implementaion of Hit-and-Run

The polytope P is defined by the mechanics of the limb and the constraints of the task as

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

and to find a starting point $\mathbf{p}$ we need only to find one feasible activation vector in the space. For the Hit-and-Run algorithm to mix faster we do not want the starting point to be close to a vertex of P [**?**]. Intuitively, this is because we do not want to get stuck in a corner. We use the following standard trick

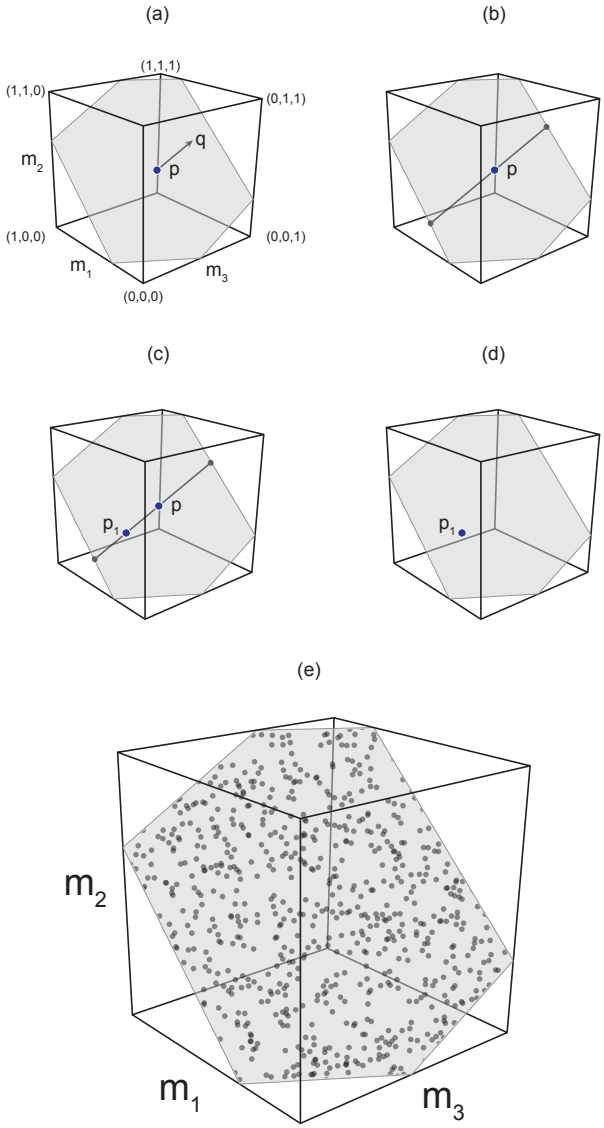

Fig 1. Characterizing the high-dimensional structure of a feasible activation set via the Hit-and-Run algorithm [?]. For a large class of biomechanical tasks, the feasible activation set is a convex polytope. Here we show a 3-muscle system as a schematic example upon which Hit-and-Run can be applied. We select a valid direction at random (a), and project a line to both boundaries (b). After selecting a point at random from the resulting line (c), we get a new valid point p_1 (d). Repeating steps a-d many (e.g., 100,000) times, and down-sampling those points, produces a statistically complete representation of the polytope (e)—and of the high-dimensional structure of the feasible activation set.

with slack variables ϵ_i, which in practice often provides a good starting point for sampling algorithms. In essence, we use the slack variables to find a point far away from the boundaries of the n-cube.

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

Thus running a linear program with these reduced ranges of the variables will produce a starting point that is far from the boundaries of P, and is not stuck in a corner.

The rest of the implementation of the Hit-and-Run algorithm is straightforward except for the choice of the random direction q. How do we sample u.a.r. (uniformly-at-random) from all directions in P? Recall that all points in P are by definition valid solutions to a given task.

The direction of q must be in the null space of the task, which is defined as $V = \{y \in R^n \mid Ay = 0\}$. Hence we need to choose directions uniformly at random across the vector space V.

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

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

A basis of a vector space 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 $\{q \in \mathbb{R}^n | A\mathbf{q} = 0\}$ and orthogonalize with the well known Gram-Schmidt method. Note that in order to get the desired uniform and random sample the basis needs to be orthonormal. The limb model is defined such that the rows of A are linearly independent and hence $r = n - m$.

Mixing Time

While we have rigorous experimental evidence that supports using Hit-and-Run in this setting, we do not have theoretical proof that guarantees our sampling will mix to the uniform distribution. Hit-and-Run is theoretically guaranteed to mix in time $\mathcal{O}(n^3)$ if the polytope is not too "flat", meaning that the ratio between the inscribed circle and enclosing circle is not too small. Such degenerate examples are usually artificially created and we henceforth assume that this does not happen in our system of muscles. From a given starting point, how many iterations of the Hit-and-Run method are necessary to reach a u.a.r. point? For convex polytopes in n dimensions up to 40, experimental results suggest that $\mathcal{O}(n)$ steps of the Hit-and-Run algorithm are sufficient. In particular, the paper [**?**] by Emiris and Fisikopoulos suggests that selecting every $10(n+1)$ points is sufficient to get a uniform distribution [**?**], while in Ge et al.'s paper every single point of the Hit-and-Run algorithm is used in the sample [**?**]. It's worth noting that these results are experimental, not theoretical.

The Hit-and-Run algorithm was specifically developed to address $\#P$-hard problems such as approximating the volume of high-dimensional polytopes [**?**]. The Hit-and-Run algorithms can converge to the uniform sampling across any convex body [**?**] after $\mathcal{O}^*(n^2R^2/r^2)$ steps (r and R are the radii of the inscribed and circumscribed ball of the polytope as mentioned in the Introduction) [**?**, **?**]. Experimental results suggest that a number of points linear with respect to N suffices for convex polytopes embedded in dimensions up to 40. Emiris and Fisikopoulos [**?**] suggest that $(10 + \frac{10}{N})\ N$ iterations suffice to uniformly sample a convex polytope.

Data and code availability

All analyses are replicable and extensible with Open Source libraries. We developed and tested our code in Ubuntu 14.04, Windows 8.1, and OSX Yosemite 10.12.1; implemented Hit-and-Run with Scala 2.11.6, and developed visualizations with R 3.3.0. All figure source data are available within the repository https://github.com/briancohn/space.

Acknowledgments

Author contributions and Correspondence

M.S. derived the mathematics of our Hit-and-Run implementations, B.C. implemented those algorithms and designed the simulations. F.V.C supported the neurobiological interpretations, while B.G. supported work with computational geometry. F.V.C. can be contacted via valero@usc.edu.

Competing Financial Interests

We do not have any competing interest to declare.