# volesti: sampling efficiently from high dimensional distributions

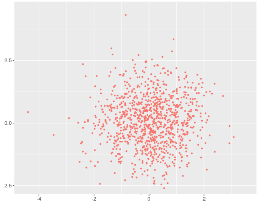
Vissarion Fisikopoulos

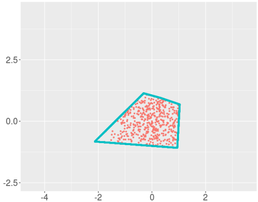
data analytics @ FOSDEM 2025

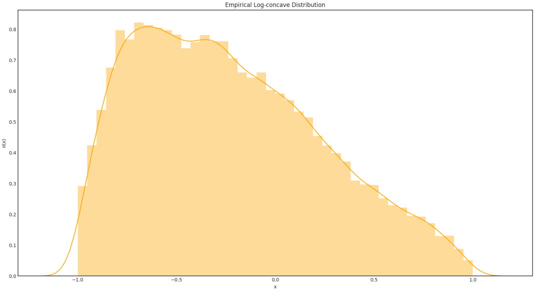
GeomScale

# Truncated distributions

- Multivariate probability distribution truncated to a convex set
- Distributions: uniform, gaussian, logconcave, etc
- Convex sets: polygons/polytopes etc



# Sampling from (truncated) distributions

**Problem**

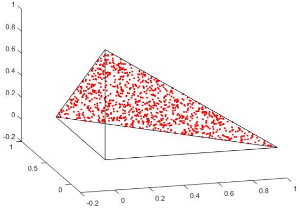
Sample (efficiently) from a (truncated) distribution

**Why?**

- Fundamental problem in mathematics and computer science
- Building block for integration & volume computation
- Applications
  - Bayesian inference (estimation of constraint parameters)
  - Constrained optimization
  - Finance (portfolio contraints)
  - Computational biology (metabolic networks)

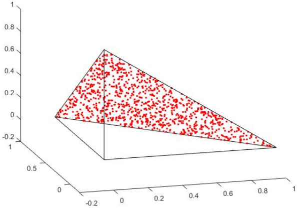
# Simple cases and simplistic approaches

- Fundamental shapes (hypercube, hypershpere, simplex) admit efficient methods

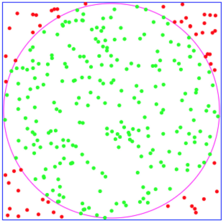


# Simple cases and simplistic approaches

- Fundamental shapes (hypercube, hypershpere, simplex) admit efficient methods

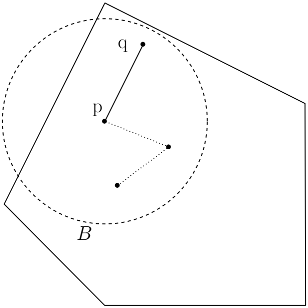



- Acception/rejection sampling does not scale to high dimensions

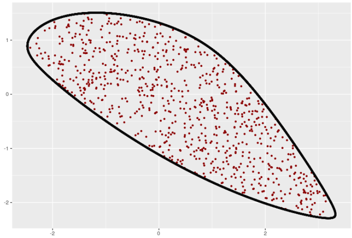


# How to sample efficiently?

- A Geometric Random Walk starts at some interior point and at each step moves to a "neighboring" point, chosen according to some distribution depending only on the current point.
- A Marcov Chain that converges to some target distribution after a number of steps




Steps of a ball walk.



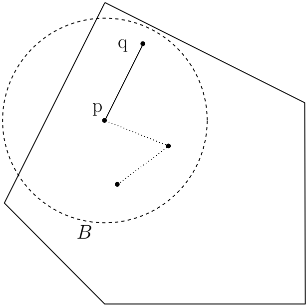

Uniform target distribution

# Three basic walks: (1) Ball walk

**Ball Walk**$(K, p, \delta, f)$: convex $K \subset \mathbb{R}^d$, $p \in P$, radius $\delta$, $f : \mathbb{R}^d \to \mathbb{R}_+$

1. Pick a uniform random point $x$ in $B(p, \delta)$.
2. **return** $x$ with probability $\min\left\{1, \frac{f(x)}{f(p)}\right\}$;

   **return** $p$ with the remaining probability.

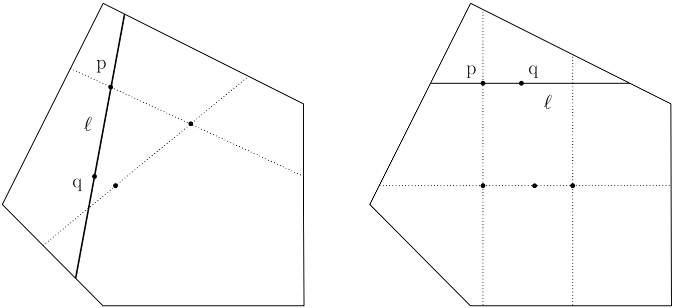



*If the density is not restricted in $K$, then it is the* **Metropolis-Hastings** *algorithm*

# Three basic walks: (2) Hit-and-Run

**Hit and Run**$(K, p, f)$: convex $K \subset \mathbb{R}^d$, point $p \in P$, $f : \mathbb{R}^d \to \mathbb{R}_+$

1. Pick uniformly a line $\ell$ through $p$.
2. **return** a random point on the chord $\ell \cap K$ chosen from the distribution $\pi_{\ell,f}$ restricted in $K \cap \ell$.

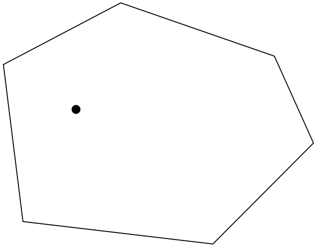



- **Q**: How do we compute $\ell \cap K$? Can we do it *exactly*?

# Three basic walks: (3) Billiard walk - Uniform case

**BW**$(K, p_i, \tau, R)$ [Polyak'14]

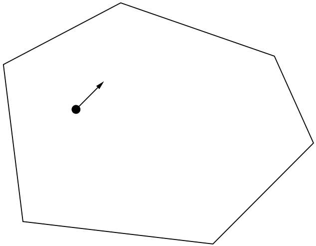
1. Generate the length of the trajectory $L = -\tau \ln \eta$, $\eta \sim U(0,1)$.
2. Pick a uniform direction $v$ to define the trajectory. then the direction becomes $v \leftarrow v - 2\langle v, s \rangle$.
3. If the trajectory meets a boundary with internal normal $s$, $||s|| = 1$,
4. **return** the end of the trajectory as $p_{i+1}$.
   If the number of reflections exceeds $R$, then **return** $p_{i+1} = p_i$.



# Three basic walks: (3) Billiard walk - Uniform case

**BW**$(K, p_i, \tau, R)$ [Polyak'14]

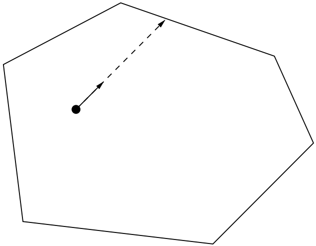
1. Generate the length of the trajectory $L = -\tau \ln \eta$, $\eta \sim U(0,1)$.
2. Pick a uniform direction $v$ to define the trajectory. then the direction becomes $v \leftarrow v - 2\langle v, s \rangle$.
3. If the trajectory meets a boundary with internal normal $s$, $||s|| = 1$,
4. **return** the end of the trajectory as $p_{i+1}$.
   If the number of reflections exceeds $R$, then **return** $p_{i+1} = p_i$.



# Three basic walks: (3) Billiard walk - Uniform case

**BW**$(K, p_i, \tau, R)$ [Polyak'14]

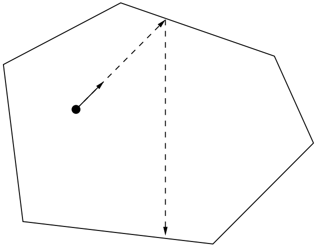
1. Generate the length of the trajectory $L = -\tau \ln \eta$, $\eta \sim U(0,1)$.
2. Pick a uniform direction $v$ to define the trajectory. then the direction becomes $v \leftarrow v - 2\langle v, s\rangle$.
3. If the trajectory meets a boundary with internal normal $s,\ ||s|| = 1$,
4. **return** the end of the trajectory as $p_{i+1}$.
   If the number of reflections exceeds $R$, then **return** $p_{i+1} = p_i$.



# Three basic walks: (3) Billiard walk - Uniform case

**BW**$(K, p_i, \tau, R)$ [Polyak'14]

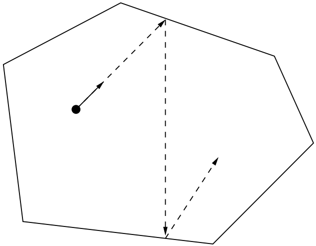
1. Generate the length of the trajectory $L = -\tau \ln \eta$, $\eta \sim U(0,1)$.
2. Pick a uniform direction $v$ to define the trajectory. then the direction becomes $v \leftarrow v - 2\langle v, s\rangle$.
3. If the trajectory meets a boundary with internal normal $s$, $||s|| = 1$,
4. **return** the end of the trajectory as $p_{i+1}$.
   If the number of reflections exceeds $R$, then **return** $p_{i+1} = p_i$.



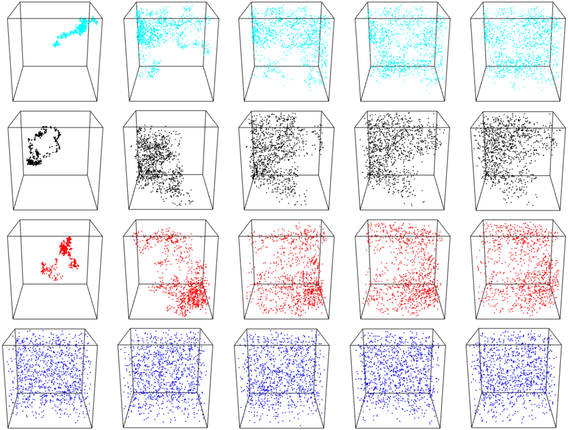
# Three basic walks: (3) Billiard walk - Uniform case

**BW**$(K, p_i, \tau, R)$ [Polyak'14]

1. Generate the length of the trajectory $L = -\tau \ln \eta$, $\eta \sim U(0,1)$.
2. Pick a uniform direction $v$ to define the trajectory. then the direction becomes $v \leftarrow v - 2\langle v, s \rangle$.
3. If the trajectory meets a boundary with internal normal $s,\ ||s|| = 1$,
4. **return** the end of the trajectory as $p_{i+1}$.
   If the number of reflections exceeds $R$, then **return** $p_{i+1} = p_i$.



# How many steps are needed to converge?



- Uniform sampling from the hypercube $[-1, 1]^{200}$ and projection to $\mathbb{R}^3$.
- Rows: Ball Walk, Coordinate Directions Hit and Run, Random Directions Hit and Run, Billiard Walk.
- Columns: walk length, $\{1, 50, 100, 150, 200\}$

# Convergence rate (or when is the right time to stop)

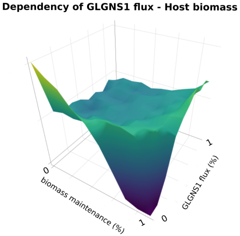
- Theoretical bounds (pessimistic) $\neq$ practice
- Statistical tests: effective sample size (ESS), potential scale reduction factor (psrf)
- Challenge: error guarantees in practice where sampling is used as a subroutine (e.g. Monte-Carlo integration)

## sampling from high dimensional distributions

- C++ library
  - R (CRAN:1.1.2, github:1.2.0)
  - Python interfaces (only github, todo: pip)

## Volesti sampling from high dimensional distributions

- C++ library
  - R (CRAN:1.1.2, github:1.2.0)
  - Python interfaces (only github, todo: pip)
- Algorithms for sampling, integration/volume, copulas

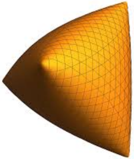


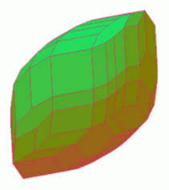
# Volesti sampling from high dimensional distributions

- C++ library
  - R (CRAN:1.1.2, github:1.2.0)
  - Python interfaces (only github, todo: pip)

- Algorithms for sampling, integration/volume, copulas

- Optimizations for different (non) convex bodies (polytopes, spectahedra, zonotopes)





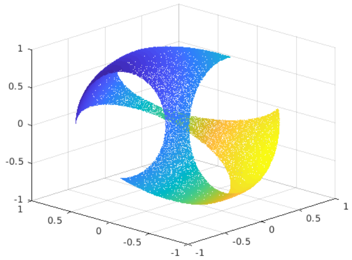
# Volesti sampling from high dimensional distributions

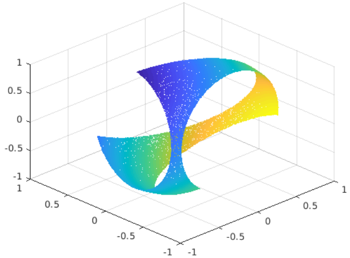
- C++ library
  - R (CRAN:1.1.2, github:1.2.0)
  - Python interfaces (only github, todo: pip)
- Algorithms for sampling, integration/volume, copulas
- Optimizations for different (non) convex bodies (polytopes, spectahedra, zonotopes)
- Utilities for financial and biolgical applications

# Applications in finance

## Portfolio analysis

- The set of portfolios (investments in a collection of stocks) is a simplex.
- Constraints on investments yield a general polytope.
- Portfolios with same volatility (the degree of variation of a trading price series over time) lie on an ellipsoid.



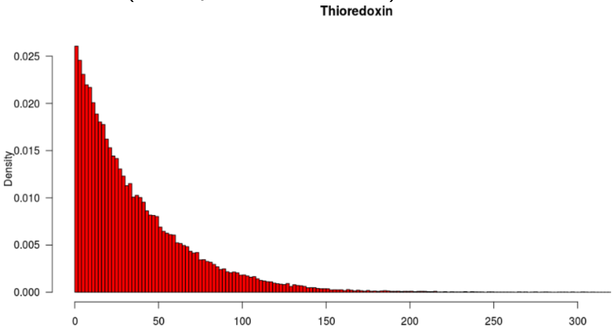




Randomized geometric tools for anomaly detection in stock markets [Bachelard,Chalkis,F,Tsigaridas'23]

# Applications in structural biology

[Chalkis,F, Tsigaridas, Zafeiropoulos]

- Metabolic networks model the reactions of metabolites in an organim or system.
- Each reaction has a flow or rate called flux.
- The set of states of the network where fluxes are in balance (rate of production = rate of consumption) is a convex polytope.
- Sampling from polytope yield probability densities for reaction fluxes (example: thioredoxin)



# GeomScale org

C++ library: sampling, integration/volume from convex bodies

Python interface with extra utilities for metabolic network analysis (FBA, copulas, visualization)

R interface with extra utilities for finance (portfolio analysis)

---

NumFOCUS Affiliated Project.

Support from an open source community.

**Thank you!**