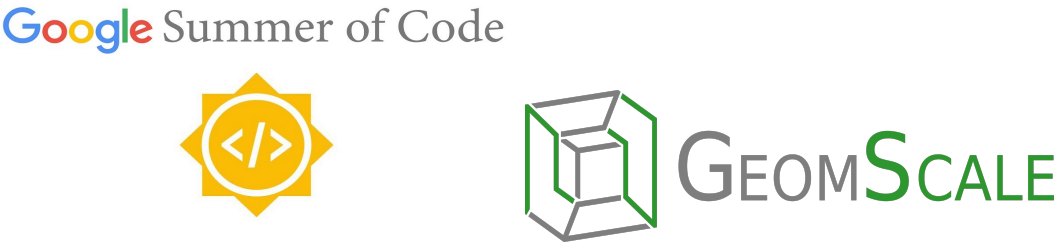
Google Summer of Code 2022 Project Proposal
GeomScale Project
Randomized SDP solver with Riemannian Hamiltonian Monte Carlo

Ioannis Iakovidis
National Kapodistrian University of Athens
iakoviid@gmail.com

April 17, 2022



Abstract

The goal of this project is to equip the VolEsti [1] library with a Riemannian Hamiltonian Monte Carlo sampler constrained on convex bodies based on [2]. The implemented sampler should return samples from log-concave distributions constrained in polytopes as well as spectrahedra. Afterwards, the above implementation will be used as a subroutine for a Semidefinate Program (SDP) solver based on the methods [3], [4]. We will also include benchmark tests to compare the efficiency of our code against other state of the art libraries.

Keywords: sampling, geometry, semidefinate programming, spectrahedron, hamiltonian, monte carlo

Contents

1 Contant Information

Name	Ioannis Iakovidis
Affiliation	National Kapodistrian University of Athens
Program	Algorithms, Logic and Discrete Mathematics (ALMA)
Mentors	Apostolos Chalkis, Vissarion Fisikopoulos, Marios Papachristou and Elias Tsigaridas
email	iakoviid@gmail.com
github	iakoviid
blog	iakoviid.github.io
Phone	+306944267793
Address	Mpiglistas 13, Athens-Greece

2 Introduction

The problem of sampling from a given distribution appears in many areas of computer science and mathematics. Its correspondence with the field of optimization is one that has gained an increasing amount of interest in recent years. In this project we are gone concern ourselves with the use of sampling for the solution of Semidefinate Programs (SDPs).

Definition (Semidefinate Programming)**.** A Semidefinate program is a optimization problem of the following form:

$$\begin{aligned} \min_{x} \quad & c^T x \\ \text{s.t.} \quad & A_0 + \sum_{0}^{n} A_i x_i \succcurlyeq 0 \\ & x \in \mathbb{R}^n \end{aligned}$$

We will implement methods that using samples from a distribution constrained in spectrahedra [5], the feasible region of a semidefinate program, will approximate the solution of an SDP. Specifically our implementation is gone sample from a log-concave distribution constrained in a convex shape.

Definition (log-concave distribution)**.** A distribution P is log-concave if its probability density function g can be written in the form:

$$g(x) \propto \epsilon^{-f(x)}$$

where f is a convex function.

2.1 Riemannian Hamiltonian Monte Carlo

A popular sampler that generates samples from log-concave distributions is the Riemannian Hamiltonian Monte Carlo (RHMC). The RHMC algorithm extends parametric space from $x \in \mathbb{R}^d$ to $(x, v) \in \mathbb{R}^{2d}$. In this new space, it uses the momentum v to move with dynamics that preserve some distribution ($\epsilon^{-H(x,v)}$, $H(x, v)$ to be defined later), related to the desired one. Specifically the RHMC is defined by the algorithm bellow where $H(x, v)$ is the Hamiltonian:

$$H(x, v) = f(x) + \frac{1}{2} v^T M(x)^{-1} v + \frac{1}{2} \log \det M(x)$$

And $M(x)$ is a position-dependent positive definite $n \times n$ matrix, called the mass matrix.

Algorithm 1: RHMC(x_0, T)

Result: Sequence $x_0, x_1, \ldots, x_l$ of RHMC sampled points

for *k=1,... l* **do**

  Sample v: $v \sim \mathcal{N}(0, M(x_{k-1}))$;

  Solve the ODEs

$$\frac{dx}{dt} = \frac{\partial H(x,v)}{\partial v} \ , \ \frac{dv}{dt} = -\frac{\partial H(x,v)}{\partial x}$$

  with starting point v, x_{k-1};

  Let $x_k = x(T)$ and $v_k = v(T)$;

end

One can prove that this algorithm samples from the distribution $\epsilon^{-H(x,v)}$. To get samples from $\epsilon^{-f(x)}$ we have to choose M such that marginal with respect to v is proportional to $\epsilon^{-H(x,v)}$. Notice that to implement this above system computationally we have to solve the ODEs numerically, which can be done using the Leapfrog integrator, Euler's Method and Implicit Midpoint Integrator (which is provided on the PolytopeSampler Package). However, since ODEs can not be solved exactly, because they are discretized to the applications accuracy, the stationary distribution can not be preserved at every step. To ensure convergence to the stationary distribution an additional correction procedure should be performed known as the Metropolis filter. Details about the discretization process are mentioned in paragraph 2.3.

2.2 Constrained Riemannian Hamiltonian Monte Carlo

Now we will define a constrained version of RHMC named Constrained Riemannian Hamiltonian Monte Carlo (CRHMC). In this version, Yin Tat Lee et al. [2] propose a change in the definition of the Hamiltonian and prove that the resulting random walk does not violate the constrains. Let the constrains being defined as $c(x) = 0$ where $c : \mathbb{R}^d \to \mathbb{R}^m$ and $x \in K$, where $K = \prod_i K_i$ additional constraints with barrier function $\phi(x)$ and K_i of the form $l_i \leq x_i \leq u_i, l_i$ where , $u_i \in \mathbb{R}$.

To ensure that we do not violate the constraints we will choose $M(x)$ so that we have Range(M(x)) = Null$Dc(x)$, where $Dc(x)$ is the Jacobian of the constraints. Notice that M should not be invertible in this case. In other words $M(x)$ is chosen such the the sampled momentum will be from directions do not change the value of $c(x)$. One such choice is $M(x) = Q(x)^T g(x) Q(x)$ where $Q(x) = I - Dc(x)^T (Dc(x)Dc(x)^T)^{-1} Dc(x)$ and $g(x) = \nabla^2 \phi(x)$.

Then $H(x, v)$ is defined as:

$$H(x,v) = \bar{H}(x,v) + \lambda(x,v)^T c(x) \text{ where } \bar{H}(x,v) = f(x) + \frac{1}{2} v^T M(x)^\dagger v + \frac{1}{2} \log \text{pdet} M(x)$$

where pdet and † are the pseudo-determinant and pseudoinverse operators. And λ is picked so that $v \in \text{Null}(Dc(x))$ namely $\lambda(x,v) = (Dc(x)Dc(x)^T)^{-1}(D^2 c(x)[v, \frac{dx}{dt}] - Dc(x)\frac{\partial \bar{H}(x,v)}{\partial x})$.

Theorem 1 (CRHMC for Linear Constraints)**.** *For linear constraints $c(x) = Ax - b$ the above expression for the Hamiltonian can be simplified to:*

$$H(x,v) = f(x) + \frac{1}{2} v^T g^{-\frac{1}{2}} (I - P) g^{-\frac{1}{2}} v + \frac{1}{2} (\log\det g - \log\det A g^{-1} A^T)$$

where $P = g^{-\frac{1}{2}} A^T (A g^{-1} A^T)^{-1} A g^{-\frac{1}{2}}$.

2.3 Discretization

2.3.1 Implicit Midpoint Integrator

For numerically solving the ODEs [2] uses the implicit midpoint method. This is a method that solves an ODE of the form $\frac{dy}{dt} = f(t, y)$ by producing approximate values for $y(t_n), t_n = t_0 + nh$:

$$y_{n+1} = y_n + hf(t_n + \frac{h}{2}, \frac{y_n + y_{n+1}}{2}).$$

For our case by writing $H(x, v)$ as $H(x, v) = H_1(x, v) + H_2(x, v)$ with:

$$H_1(x, v) = f(x) + \frac{1}{2}(\log\det g - \log\det Ag^{-1}A^T)$$
$$H_2(x, v) = \frac{1}{2}v^T g^{-\frac{1}{2}}(I - P)g^{-\frac{1}{2}}v$$

We can apply the implicit midpoint method to starting point x_0, v_0 by first computing:

$$x_{1/3} = x_0 + \frac{h}{2}\frac{\partial H_1}{\partial v}(x_0, v_0) = x_0$$
$$v_{1/3} = v_0 - \frac{h}{2}\frac{\partial H_1}{\partial x}(x_0, v_0)$$

then solving:

$$x_{2/3} = x_{1/3} + h\frac{\partial H_2}{\partial v}(\frac{x_{1/3} + x_{2/3}}{2}, \frac{v_{1/3} + v_{2/3}}{2})$$
$$v_{2/3} = v_{1/3} - h\frac{\partial H_2}{\partial x}(\frac{x_{1/3} + x_{2/3}}{2}, \frac{v_{1/3} + v_{2/3}}{2})$$

and returning:

$$x = x_{2/3}$$
$$v = v_{2/3} - \frac{h}{2}\frac{\partial H_1}{\partial x}(x_1)$$

2.3.2 Overall Method

Algorithm 2: RHMC(Initial point $x(0)$, velocity $v(0)$, record frequency T , step size h, ODE steps K)

Result: Sequence $x_0, x_1, \ldots, x_l$ of RHMC sampled points

for *k=1,... l* **do**
- Sample v: $v \sim \mathcal{N}(0, M(x_{k-1}))$;
- Solve $\frac{dx}{dt} = \frac{\partial H(x,v)}{\partial v}$, $\frac{dv}{dt} = -\frac{\partial H(x,v)}{\partial x}$ via the implicit midpoint method
- with starting point x_{k-1}, v to get point x', v';
- Use Metropolis Filter to update:
- With probability $\min(1, \frac{e^{-H(x',v')}}{e^{-H(x_{k-1},v)}})$, set $x_t = x'$, otherwise set $x(t) = x$.

end

2.4 SDP Optimization

2.4.1 Simulated Annealing

This method stems from the fact that if we sample from $\pi_T(x) \propto \epsilon^{-c^T x/T}$ samples should most likely come from a region close to the optimum, if T is very small. More precisely we will use a

sequence of decreasing values of T, temperatures, to get close to the optimum. So we will generate samples from l different distributions. Each time a random point drawn from the i'th distribution is feed to distribution $i+1$ for a warm start, for faster convergence of the sampler. A simplified overview of the algorithm is given bellow, R is the radius of a ball that contains the feasible region X and d is its dimensionality.

Algorithm 3: SimulatedAnnealing(X, c, R)

Result: Point x_l close to the optimum.
let x_i uniform sample from X **for** $i = 1 \dots l$ **do**
  $T_i = R(1 - \sqrt{d})^i$;
  Sample x_i from π_{T_i} constrained on X given x_{i-1} to the sampler;
end

It can be proven that with $l = O(\sqrt{d}\log(\frac{Rd}{\epsilon\delta}))$ the algorithm above returns with probability $1-\delta$ an ϵ approximation for the global optimum. An other view of this algorithm is that it starts form a uniform distribution and slowly (any two subsequent distributions are almost the same) converges to a distribution that is concentrated close to the optimum. Bellow we see an illustration of this method.

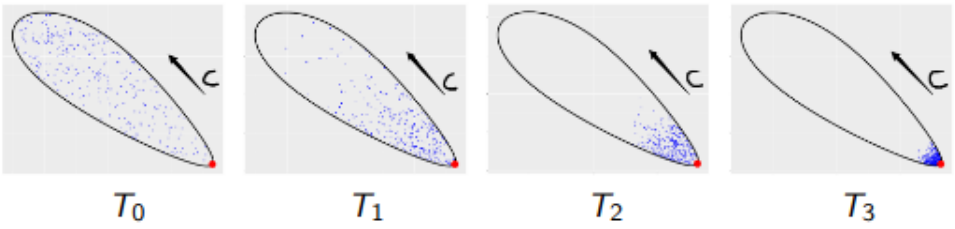


Figure 1: Simulated annealing illustration on a Spectrahedron.

2.4.2 Randomized plane cutting

Cutting plane methods have been very popular in the field of optimization especially in linear and semidefinate optimization. These methods solve the feasibility problem "Deciding whether the feasible region is empty or not", which is essentially equivalent to the problem of optimization. In a nutshell, these methods after bounding the feasible region, they repeatedly divide the bounding region by separation oracle calls. Separation oracle is a routine that given a point returns whether it is feasible or not and if it not feasible it returns a separating hyperplane. For an efficient convergence to the feasible set these method's select the center of the bounding region to feed in the separation oracle. However computing this center is expensive (NP-hard) so they result in more complicated approximate methods [6].

In this project we are gone implement a randomized variant of these methods [3]. First of all in contrast with these methods described above our algorithm will repeatedly divide the feasible region until we are sufficiently close to the optimal. Specifically it is gone sample N points from the feasible region. And we will use the best of them, with respect to the objective function, to reduce the feasible region (can be viewed as adding a constraint). Bellow we see the algorithm for a linear objective function, this is a feature of both linear and semidefinate programs, along with an illustration.

Algorithm 4: RandomizedPlaneCutting(X, c)

Result: Region X of points close to the optimum

while *StoppingRule(X)==false* **do**
  Sample N uniform random samples from X: $S = \{x_1, \ldots, x_N\}$;
  $y = \text{argmin}_{x \in S} = c^T x$;
  $X = X \cap \{x : c^T x \leq c^T y\}$;
end

A number of theoretical guarantees can be proven for this method [3]. A very important result one can prove is that the expected number times one needs to execute the loop in order to get a ϵ-approximation to the optimum is $\lceil \frac{1}{\ln N+1} d \ln(R/e) \rceil$, where d is the dimension of X (number of variables) and R is the radius of the smallest ball that contains X.

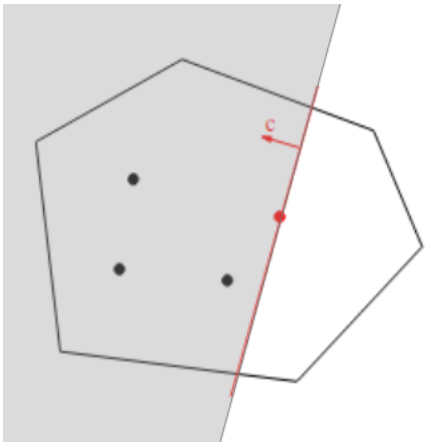


Figure 2: Randomized Plane cutting illustration.

3 Project Milestones

The primary goals of the project can be partitioned into milestones in the following way. It is important to test code after each milestone for both correctness and efficiency to have a more structured development pipeline. Bellow we see each milestone, its estimated completion time (that was carefully thought out by me and the project mentors) as well as the tasks that it is composed of.

- **Milestone I (4.5 weeks) CRHMC sampler**: Implement CRHMC sampler for log-concave sampling constrained in polytopes according to [2] (PolytopeSampler). Specifically:
  - The polytope_preprocessing function implements the preprocessing procedure of the PolytopeSampler package as described in 4.3. Functions will be added in include/preprocess. (1.5 weeks)
  - The implicit_midpoint function will solve a system of ODEs with the implicit midpoint method. And will be added to include/ode_solvers folder. (0.5 weeks)
  - Logarithmic barrier auxilary functions. (0.5 weeks)
  - The remannian_hamiltonian_monte_carlo will be the entry fucntion for the CRHMC sampler. It will be added to include/random_walks. (2 weeks)

- **Milestone II (3 weeks) CRHMC for Spectrahedra sampling** : Extend the functionality of the CRHMC code to handle spectrahedra. Make use of VolEsti's existing functions and classes for Spectrahedra. Main changes are expected to be added in the CRHMC entry function, remannian_hamiltonian_monte_carlo and also in the preprocessing step.
- **Milestone III (1.5 weeks) Spectrahedra optimization** : Use the above as a subroutine for spectrahedra optimization. All functions added will be placed in the include/optimization.
  - Use (and if needed modify) VolEsti's Simulated Annealing with CRHMC to solve SDPs [4]. (1.5 weeks)
- **Milestone IV (2 weeks) Testing** : Compare the implementation with for polytopes and test the efficiency of the SDP solvers.
  - Compare polytope sampling with PolytopeSampler. (0.5 weeks)
  - Compare SDP optimization methods to SDPA. (1.5 weeks)
  - Use testing results to improve efficiency.
- **Milestone V (1 week) R interface** : Make these capabilities available to the R interface through Rcpp and write detailed documentation.
- **Bonus Beyond GSoC**:
  - Implement the Randomized Plane Cutting algorithm for SDPs according to [3]. The randomized_plane_cutting will be the function that implements this procedure.
  - Solve the ODEs with different methods and compare the efficiency results.
  - Test the use of the preprocessing method in VolEsti's other functions.
  - Implement time consuming highly parallelizable functions in CUDA to make use of the GPU.

4 Technical Details

4.1 Tools

The development is gone be done in C++. Also the main routines will also become available to the R interface through Rcpp. For structured testing of our code we will use the following development tools:

- doctest for C++ testing.
- testthat for R testing.
- comparing with existing software (PolytopeSampler and SDPA) to ensure correctness.
- benchmarking existing software (PolytopeSampler and SDPA) to compare efficiency of our implementation.

4.2 Function Oracles

In order to apply CRHMC and simulate the dynamics the user needs to input what the function f and its first derivative are. Likely, this has been previously implemented in VolEsti i(n include/ode_solvers/oracle_functors.hpp) and using external routines the user can define oracles for the function and its derivative.

4.3 Preprocessing

For better efficiency [2] preprocesses the polytope data prior to sampling. With this preprocessing they:

- Simplify the polytope and scale for numerical stability.
- Find a feasible starting point.
- Split dense columns into several sparser columns by introducing additional variables.
- Fix variables that have range less than some tolerance by the use of the Dikin ellipsoid and by the use of the analytic center (if a coordinate of the analytic center is to close to the boundary).
- In the end remove dependent rows with the Cholesky decomposition.

This step has a large number of linear algebra calls that accessible as primitives in the original Matlab project. In our case we plan to make use of the Airpack library which is one of VolEsti's external dependencies.

4.4 ODE solvers

For the numerical solution of dynamic ODEs there are a number of routines already available in VolEsti (as it supports a constrained Hamiltonian walk). The PolytopeSampler package supports Implicit Midpoint Integrator which we are going to implement. Furthermore, one of our goals will be to use other ODE solvers (e.g. leapfrog integrator) and compare the results.

4.5 Relevant Packages

There are a number of relevant open source packages that implement related functionalities. Specifically we will compare and base our sampler implementation with the PolytopeSampler Package. This Package implements CRHMC only for polytopes our aim is to bring its capabilities to VolEsti and later extend it to handle spectrahedra. The package is mainly implemented in matlab with some routines running in C++ through MEX.

Also as for SDP optimization one of the most efficient open source software is SDPA [7]. This will be used in order to test the correctness and efficiency of our implementation.

4.6 Testing

Efficiency testing is gone be performed using synthetic data like Hypercube, Simplices, Birkhoff Polytope as well as real data like the Recon datasets [8] that represent large human metabolic network. Often matrices produced by real datasets (e.g. the matrix or matrices that are involved in the conditions of an linear or semidefinate program) have a very large condition number because they are nearly singular [9]. And so we expect for the new CRHMC sampler to perform better than VolEsti's existing samplers. For further benchmarks real data for polytopes as well as SDP problems will be taken from Plato Project Arizona State University.

Furthermore, an other goal of importance to VolEsti is to perform more tests for the cases presented in [2], to better understand the relation between efficiency and the condition number. As only experiments for the Coordinate Random Walk were presented and therefore further comparisons would be beneficial.

5 Timeline and Schedule Conflicts

The project is gone be stretched over 13 weeks and planed from 15 of July to 14 of October. I will have finished my Master's exams shortly before this date. So I will have no schedule conflict

during this time interval and this project will be my full time job. One week is left as leeway in order tackle deviations from the program or the Beyond GSoC tasks if it is not needed.

- Acceptance-Start of GSoC (15 of July)
  - Communicate with the mentors.
  - Test VolEsti's existing code.
  - Familiarization with the literature.
  - Familiarization with related software.
- 15th July - 16th August
  - Implementation of Milestone I
  - Documentation and testing.
- 16th August - 6th September
  - Implementation of Milestone II
  - Documentation and testing.
- 6th September - 16th September
  - Implementation of Milestone III
  - Documentation and testing.
- 16th September - 30th September
  - Implementation of Milestone IV
  - Documentation and testing.
- 30th September - 7th October
  - Implementation of Milestone V
  - Documentation and testing.

6 Personal Motivation

In this section I list relevant background information, this project's significance in my personal growth as well as why I am fit for it. Detailed resume can be found in this page CV.

6.1 Backround

I am a Master's student at the National and Kapodistrian University of Athens in the Department of Informatics and Telecommunications. Initially, I was attracted to the field of Mathematics and Computer Science in the context of high school problem solving contests. Having some distinctions at a local level, I aspired to continue participating as a university student. After graduating high school, being unsure of what I should do, I entered through National Exams in 2013 the Corporate Branch of the Evelpidon Military Academy. This position is very appealing to Greek high school students because of the job stability it offers. After a year and a half in the academy, I had fulfilled my year long mandatory military service and I was ready to focus on my university studies. So in the fall of 2015 I entered the Department of Electrical and Computer Engineering of Aristotle University of Thessaloniki. Concerning my studies, it has always been my plan to pursue a research career in Computer Science and Mathematics, so I focused on related courses.

I am interested in several subjects of Computer Science and Mathematics. More specifically, I enjoy learning about and working on Data Science, Algorithm Design and Optimization. During

my studies, I have worked on a number of Parallel Programming projects, optimizing algorithms that are used in Machine learning (ex. MPI/CUDA k Nearest Neighbors). Last year, I was working on finishing my Diploma Thesis under the supervision of Associate Professor Nikolaos Pitsianis. The subject of my thesis was Performance Optimization of the t-SNE algorithm for the GPU, a problem that has generated a lot of exciting research in recent years.

As an extracurricular activity, during the first four years of my studies I participated in several mathematics competitions winning multiple awards. This helped me refine my scientific interests and gave me a strong background in mathematics and problem solving. My distinctions were one silver and two bronze medals in the International Mathematics Competition for University students (IMC) and two silver medals in the South Eastern European Mathematical Olympiad for University Students (SEEMOUS). This experience is of particular importance to me, as it has given me the chance to compete and most importantly bond with some very talented and passionate people from universities across the globe.

During the second and third year of my undergraduate studies I voluntarily assisted in teaching the course that selects and prepares the university team for SEEMOUS. Later this course became part of the curriculum in the Mathematics Department as an elective course. In the Fall semester of 2018 I worked as a teaching assistant for the course with Associate Professor Romanos Diogenes Malikiosis.

6.2 Motivation

Regarding the project I apply for. I believe that working in this project as an intern with Geomscale through GSoC 2022 is a very exiting opportunity. Because it combines two fields that I am very passionate about, programming and mathematics. Moreover, It is very appealing to me that the problem involves coding an scientific computing like procedure, as it is something that I am very interested in and that I have previous experience, through my diploma thesis. Coding large projects is very exiting for me as I like how from some primitives, building block by building block people can build very useful libraries. Furthermore, writing open source software involves more collaborative effort and this is something I am very passionate about. I always enjoy sharing ideas and collaborating with new people. Last but not least I would be very interested to work in the field of Geometric Statistics permanently and I believe that this internship would be an amazing start to such a goal.

Concerning my professional and career goals, I would like to be able to complete a PhD and do research in the field of Computer Science and Mathematics. In my opinion this internship could help me explore the subject on a deeper level. And last but not list, I think that it would be a great opportunity for me to work with some very experienced and talented researchers.

7 Projects Impact and Difficulties

The main advantage of the CRHMC is that its efficiency is independent of the condition number of the shape that it is constrained in. Since in practice many shapes that arise are usually ill-conditioned, this property is of crucial importance and leads to orders of magnitude better results than existing Packages. Also the PolytopeSampler project has a well thought out preprocessing scheme 4.3 for polytopes that improves performance. This is a function that can be used by VolEsti, if profitable, not only for the implementation of CRHMC but also for other procedures. In this way the user can first load the data with preprocessing and then do use multiple of VolEsti's functionalities with the modified data, while preprocessing is only used once. This is very helpful as in statistics and analytics one often applies a large number of method's to a single dataset.

One of the main difficulties of this project in my opinion will be the efficient implementation of Milestone II. Because the authors of CRHMC do a lot of calculations for the specialized case of polytopes and the general case is not that well studied, extensions are expected to be difficult and

need the understanding of the mathematics of CRHMC. Likely because of my background I am confident in by ability to complete this task.

8 Tests

To ensure the understanding of the project the mentors assigned me with 3 task that were successfully completed. Code written for the task can be found in Github.

8.1 Easy Task

The objective of this task is using the R-interface of VolEsti to visualize sampling over a polytope using any random walk. Bellow we see such a visualization for a random pentagon.

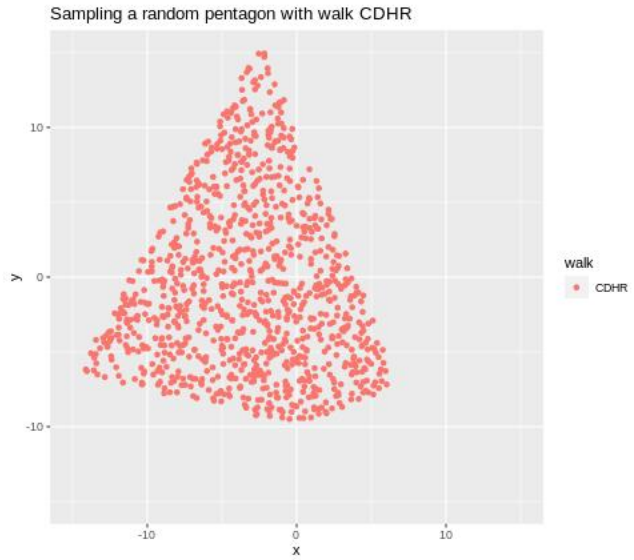


Figure 3: Sampling from a random polytope

8.2 Medium Task

For this task I was asked to extend the Hit and Run algorithm so that it can sample from the boundary of a Spectrahedra. Consulting [10], [11] I implemented the simple following algorithm and then executed it for set 2-dimensional (visualizable) examples which then I ploted together with the actual boundary.

Algorithm 5: BoundarySampling(P, n)

Result: Samples $x_1, \ldots, x_n$ from the boundary of Spectrahedron P.

```
p=InteriorPoint(P);
v=SampleUnifDirection();;
for i = 1 ... n do
    d=PositiveIntersectionDistance(P,p,v);
    x_i = p + d · v;
    nt=BoundaryNormal(P,x_i);
    v=SampleUnifDirection();
    if v · nt < 0 then
        v = −v;
    end
end
```

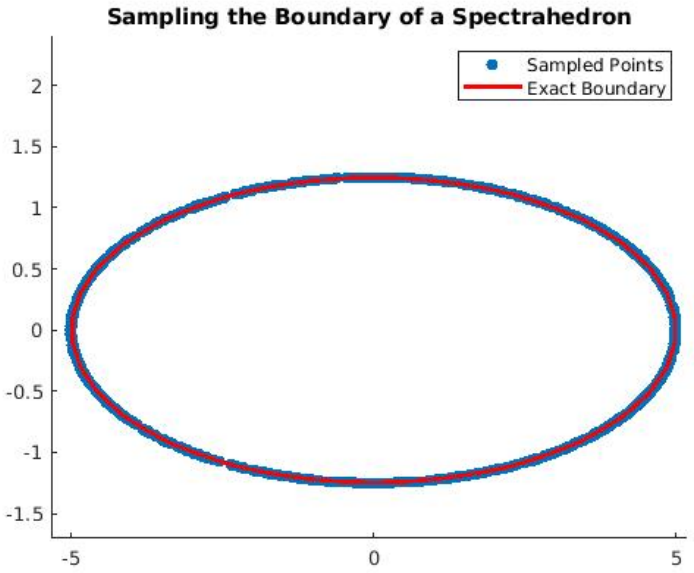


Figure 4: Sampling from the boundary of a Spectrahedron

8.3 Hard Task

For this task I was asked to implement an interior point method for linear programming. I selected to implement Newton's method with a logarithmic barrier [12]. Essentially with this technique in order to solve the problem $\min c^T \cdot x$ s.t $Ax \leq b$ we minimize the function $F_t(x) = \min t \cdot c^T \cdot x - \sum_{i=0}^{m}(b_i - (Ax)_i)$ for a sequence of increasing values of t. Notice that increasing t gives greater utility to the objective function and for a large enough value of t we can obtain an ϵ good approximation of the optimum. Using Newton's method one can increase t multiplicatively and thus have a faster convergence. Bellow we see the duality gap of the function F with the number of iterations. The value duality gap can be used for a stopping criterion as we know that we are at least this close to the optimum f^*, or in other words $|f - f^*| \leq \text{DualityGap} := |f - \text{dualf}|$.

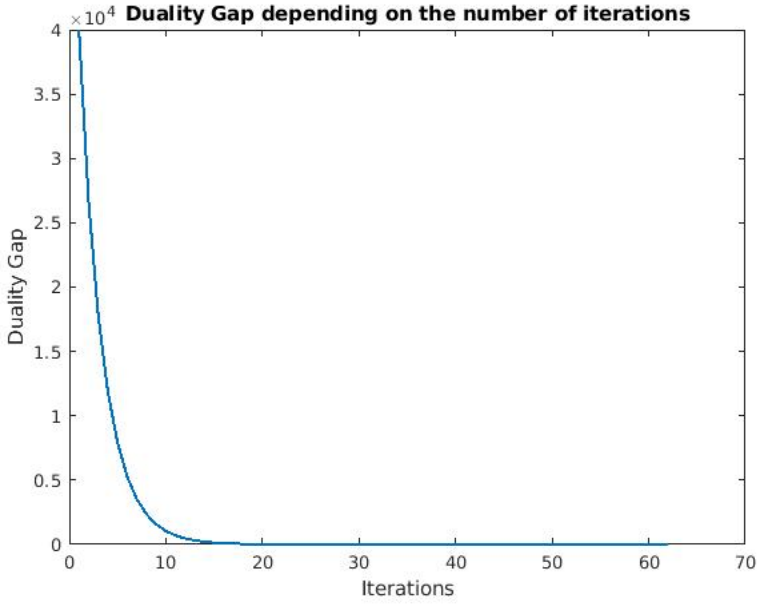


Figure 5: Duality Gap of F

8.4 Code Listings

Here for completeness we include the code written for the main routines of the tasks.

8.4.1 Easy Task

```
walk="CDHR"
b=c(10,10,10,10,10)
A = matrix(c(1,0,−0.25,−1,2.5,1,0.4,−1,−0.9,0.5), nrow=5, ncol=2, byrow = TRUE)
P = Hpolytope(A = A, b = b)
points = sample_points(P, 1000)
plot(ggplot(data.frame( x=points[1,], y=points[2,] )) +
geom_point( aes(x=x, y=y, color=walk)) + coord_fixed(xlim = c(−15,15),
ylim = c(−15,15)) + ggtitle( sprintf("Sampling a random pentagon with walk %s", walk)))
```

8.4.2 Medium Task

```
/*Pick a uniform direction on the halfplane of the normal vector */
void unifDirection(spectrahedron P, Point &p, Point &v, RNGType &rng){
    unsigned int d = P.dimension();
    VT nt(d);
    (P.lmi).normalizedDeterminantGradient(p.getCoefficients(), (P.precomputedValues).eigenvector, nt);
    v=getDirection(d,rng);
    if(v.dot(nt)>0){v=−1*v;}
}
/*Stochastic Billiards algorithm */
```

```
std :: vector<Point> StochasticBilliard(spectrahedron &P, Point& q, unsigned int num_points, RNGType
     &rng){
    unsigned int d = P.dimension();
    double tau;
    Point p = q, v;
    std :: vector<Point> randPoints;
        v = getDirection(d, rng);
        int it = 0;
        //For each sample
        while (it < num_points)
        {
          //Find the intersection
            tau=P.positiveLinearIntersection(p. getCoefficients (), v. getCoefficients ());
            randPoints.push_back(p+tau*v);
            tau = 0.995 * tau;
            //Move to it
            p += tau * v;
            //pick a uniform direction
            unifDirection(P, p, v,rng);
            it++;
        }
    return randPoints;
}
//Entry function
std :: vector<Point> boundarySampler(unsigned int d, unsigned int num_points){
    RNGType rng(d);
    Point p(d);
    // Create a spectrahedron
    std :: vector<MT> lmi_mat = Condition1();
    LMI<double, MT, VT> lmi(lmi_mat);
    spectrahedron spectra(lmi);
    spectra. set_interior_point (p);
    return StochasticBilliard (spectra, p, num_points, rng);
}
```

8.4.3 Hard Task

```
double t0 = 0.0001;
double nu = 1.5;
double tolerance=0.000001;

double IPMprimaldual(mat &A, vec &b, vec &c, vec &p) {
  int max_iter = 150;
  int iter = 0;
  double t = t0;
  double dualf, f;
  vec dualArg;
  while (iter < max_iter) {
    vec temp = c * t;
    NewtonsMethod(A, b, temp, p,t,dualf,dualArg,f);
    double dualityGap=f-dualf;
    //Compair the Duality Gap against some preset tolerance value
    if(dualityGap<tolerance){
      break;
    }
    //Multiplicatevely Update t
    t = t * nu;
    iter++;
```

```
  }
  return as_scalar(c.t() * p);
}
double alphaparam=0.1;
double betaparam=0.5;
double threshold = 0.00000001;
double e = numeric_limits<double>::epsilon();
//Newtons Method
void NewtonsMethod(mat &A, vec &b, vec &c, vec &p,double T,double& dualf,vec& dualArg,double& f)
     {
  int max_iter = 100;
  int iter = 0;
  int n=A.n_cols;
  int m=A.n_rows;
  vec Grad = vec(n);
  Grad.fill(1);
  mat H = mat(n, n);
  vec temp,dsq;
  while (iter < max_iter && norm(Grad) > threshold) {
    temp = b-A * p;
    temp.transform( [](double val) { return 1/max(val,e); } );
    // Compute the Gradient
    Grad=c+A.t()*temp;
    // Compute the Hessian matrix
    dsq=temp;
    dsq.transform( [](double val) { return val*val; } );
    H=A.t()*diagmat(dsq)*A;

    vec dx = -solve(H, Grad, solve_opts::fast);
    double lambda = -as_scalar(Grad.t() * dx);
    if (lambda / 2 < threshold) {
      break;
    }
    //Backtracking Line Search
    f=ComputeF(A,b,c,p);
    double mu=1;
    while (ComputeF(A,b,c,p+mu*dx)>f+alphaparam*mu*as_scalar(Grad.t()*dx)){
      mu=mu*betaparam;
    }
    // update p
    p = p + mu*dx;
    iter++;
  }
  f=ComputeF(A,b,c,p);
  dualf=f-m/T;
  dualArg=1/T*temp;

}
```

References

[1] A. Chalkis and V. Fisikopoulos, "Volesti: Volume approximation and sampling for convex polytopes in r," *The R Journal*, vol. 13, no. 2, p. 561, 2021. DOI: 10.32614/rj-2021-077. [Online]. Available: https://doi.org/10.32614%2Frj-2021-077.

[2] Y. Kook, Y. T. Lee, R. Shen, and S. S. Vempala, "Sampling with riemannian hamiltonian monte carlo in a constrained space," *CoRR*, vol. abs/2202.01908, 2022. arXiv: 2202.01908. [Online]. Available: https://arxiv.org/abs/2202.01908.

[3] F. Dabbene, P. Shcherbakov, and B. Polyak, "A randomized cutting plane method with probabilistic geometric convergence," *SIAM Journal on Optimization*, vol. 20, pp. 3185–3207, Jan. 2010. DOI: 10.1137/080742506.

[4] A. T. Kalai and S. Vempala, "Simulated annealing for convex optimization," *Mathematics of Operations Research*, vol. 31, no. 2, pp. 253–266, 2006. DOI: 10.1287/moor.1060.0194. eprint: https://doi.org/10.1287/moor.1060.0194. [Online]. Available: https://doi.org/10.1287/moor.1060.0194.

[5] C. Vinzant, "What is...a spectrahedron?" *Notices of the American Mathematical Society*, vol. 61, p. 1, May 2014. DOI: 10.1090/noti1116.

[6] H. Jiang, Y. T. Lee, Z. Song, and S. C.-w. Wong, *An improved cutting plane method for convex optimization, convex-concave games and its applications*, 2020. DOI: 10.48550/ARXIV.2004.04250. [Online]. Available: https://arxiv.org/abs/2004.04250.

[7] M. Yamashita, K. Fujisawa, M. Fukuda, K. Kobayashi, K. Nakata, and M. Nakata, "Latest developments in the sdpa family for solving large-scale sdps," in *Handbook on Semidefinite, Conic and Polynomial Optimization*, M. F. Anjos and J. B. Lasserre, Eds. Boston, MA: Springer US, 2012, pp. 687–713, ISBN: 978-1-4614-0769-0. DOI: 10.1007/978-1-4614-0769-0_24. [Online]. Available: https://doi.org/10.1007/978-1-4614-0769-0_24.

[8] Z. King, J. Lu, A. Dräger, *et al.*, "Bigg models: A platform for integrating, standardizing and sharing genome-scale models," *Nucleic acids research*, vol. 44, Oct. 2015. DOI: 10.1093/nar/gkv1049.

[9] M. Udell and A. Townsend, *Why are big data matrices approximately low rank?* 2017. DOI: 10.48550/ARXIV.1705.07474. [Online]. Available: https://arxiv.org/abs/1705.07474.

[10] A. B. Dieker and S. Vempala, *Stochastic billiards for sampling from the boundary of a convex set*, 2014. DOI: 10.48550/ARXIV.1410.5775. [Online]. Available: https://arxiv.org/abs/1410.5775.

[11] C. G. E. Boender, R. J. Caron, J. F. McDonald, *et al.*, "Shake-and-bake algorithms for generating uniform points on the boundary of bounded polyhedra," *Operations Research*, vol. 39, no. 6, pp. 945–954, 1991, ISSN: 0030364X, 15265463. [Online]. Available: http://www.jstor.org/stable/171241 (visited on 04/07/2022).

[12] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004, ISBN: 0521833787. [Online]. Available: http://www.amazon.com/exec/obidos/redirect?tag=citeulike-20\&path=ASIN/0521833787.