Cargèse 2015

Nisheeth K. Vishnoi

Preface

These notes are intended for the attendees of the Sixth Cargèse Workshop on Combinatorial Optimization to accompany the lectures delivered by the author in the Fall of 2015. These notes draw heavily from a set of surveys/lecture notes written by the author on the underlying techniques for fast algorithms (see [Vis13, SV14, Vis14]) and are intended for an audience familiar with the basics. Many thanks to the many people who have contributed to these notes and to my understanding. Please feel free to email your comments and suggestions.

Nisheeth K. Vishnoi
September 2015, EPFL
nisheeth.vishnoi@epfl.ch

Contents

1

Solving Linear Equations via the Conjugate Gradient Method

We discuss the problem of solving a system of linear equations. We first present the Conjugate Gradient method which works when the corresponding matrix is positive definite. We then give a simple proof of the rate of convergence of this method by using low-degree polynomial approximations for x^s.[1]

Given a matrix $A \in \mathbb{R}^{n\times n}$ and a vector $v \in \mathbb{R}^n$, our goal is to find a vector $x \in \mathbb{R}^n$ such that $Ax = v$. The exact solution $x^\star \stackrel{\text{def}}{=} A^{-1}v$ can be computed by Gaussian elimination, but the fastest known implementation requires the same time as matrix multiplication (currently $O(n^{2.737})$). For many applications, the number of non-zero entries in A (denoted by m), or its *sparsity*, is much smaller than n^2 and, ideally, we would like linear solvers which run in time $\tilde{O}(m)$ [2], roughly the time it takes to multiply a vector with A. While we are far from this goal for general matrices, *iterative methods*, based on techniques such as gradient descent or the Conjugate Gradient method reduce the problem of solving a system of linear equations to the computation of a small number of matrix-vector products with the matrix A when A is symmetric and positive definite (PD). These methods often produce only approximate solutions. However, these approximate solutions suffice for most applications. While the running time of the gradient descent-based method varies linearly with the condition number of A, that of the Conjugate Gradient

[1] This is almost identical to Chapter 9 from Sachdeva and Vishnoi [SV14].
[2] The $\tilde{O}$ notation hides polynomial factors in $\log n$.

method depends on the square-root of the condition number; the quadratic saving occurring precisely because there exist $\sqrt{s}$- degree polynomials approximating x^s. The guarantees of the Conjugate Gradient method are summarized in the following theorem:

Theorem 1.1. Given an $n \times n$ symmetric matrix $A \succ 0$, and a vector $v \in \mathbb{R}^n$, the Conjugate Gradient method can find a vector x such that $\|x - A^{-1}v\|_A \leq \delta \|A^{-1}v\|_A$ in time $O((t_A + n) \cdot \sqrt{\kappa(A)} \log 1/\delta)$, where t_A is the time required to multiply A with a given vector, and $\kappa(A)$ is the condition number of A.

1.1 A Gradient Descent Based Linear Solver

The gradient descent method is a general method to solve convex programs; here we only focus on its application to linear systems. The PD assumption on A allows us to formulate the problem of solving $Ax = v$ as a convex programming problem: For the function

$$f_A(x) \stackrel{\text{def}}{=} \|x - x^\star\|_A^2 = (x - x^\star)^\top A(x - x^\star) = x^\top Ax - 2x^\top v + x^{\star\top} Ax^\star,$$

find the vector x that minimizes $f_A(x)$. Since A is symmetric and PD, this is a convex function, and has a unique minimizer $x = x^\star$.

When minimizing f_A, each iteration of the gradient descent method is as follows: Start from the current estimate of $x^\star$, say x_t, and move along the direction of maximum rate of decrease of the function f_A, *i.e.*, against its gradient, to the point that minimizes the function along this line. We can explicitly compute the gradient of f_A to be $\nabla f_A(x) = 2A(x - x^\star) = 2(Ax - v)$. Thus, we have

$$x_{t+1} = x_t - \alpha_t \nabla f_A(x_t) = x_t - 2\alpha_t(Ax_t - v)$$

for some α_t. Define the *residual* $r_t \stackrel{\text{def}}{=} v - Ax_t$. Thus, we have

$$\begin{aligned} f_A(x_{t+1}) = f_A(x_t + 2\alpha_t r_t) &= (x_t - x^\star + 2\alpha_t r_t)^\top A(x_t - x^\star + 2\alpha_t r_t) \\ &= (x_t - x^\star)^\top A(x_t - x^\star) + 4\alpha_t (x_t - x^\star)^\top Ar_t + 4\alpha_t^2 r_t^\top Ar_t. \\ &= (x_t - x^\star)^\top A(x_t - x^\star) - 4\alpha_t r_t^\top r_t + 4\alpha_t^2 r_t^\top Ar_t \end{aligned}$$

is a quadratic function in α_t. We can analytically compute the α_t that minimizes f_A and find that it is $\frac{1}{2} \cdot \frac{r_t^\top r_t}{r_t^\top Ar_t}$. Substituting this value of α_t, and using $x^\star - x_t = A^{-1}r_t$, we obtain

$$\|x_{t+1} - x^\star\|_A^2 = \|x_t - x^\star\|_A^2 - \frac{(r_t^\top r_t)^2}{r_t^\top Ar_t} = \|x_t - x^\star\|_A^2 \left(1 - \frac{r_t^\top r_t}{r_t^\top Ar_t} \cdot \frac{r_t^\top r_t}{r_t^\top A^{-1}r_t}\right).$$

Now we relate the rate of convergence to the optimal solution to the condition number of A. Towards this, note that for any z, we have

$$z^\top Az \leq \lambda_1 z^\top z \quad \text{and} \quad z^\top A^{-1}z \leq \lambda_n^{-1} z^\top z,$$

where λ_1 and λ_n are the largest and the smallest eigenvalues of A respectively. Thus,

$$\|x_{t+1} - x^\star\|_A^2 \leq (1 - \kappa^{-1})\|x_t - x^\star\|_A^2,$$

where $\kappa \stackrel{\text{def}}{=} \kappa(A) = {}^{\lambda_1}/_{\lambda_n}$ is the condition number of A. Hence, assuming we start with $x_0 = \mathbf{0}$, we can find an x_t such that $\|x_t - x^\star\|_A \leq \delta\|x^\star\|_A$ in approximately $\kappa \log {}^1/_\delta$ iterations, with the cost of each iteration dominated by $O(1)$ multiplications of the matrix A with a given vector (and $O(1)$ dot product computations). Thus, this gradient descent-based method allows us to compute a δ approximate solution to $x^\star$ in time $O((t_A + n)\kappa \log {}^1/_\delta)$.

1.2 The Conjugate Gradient Method

Observe that at any step t of the gradient descent method, we have $x_{t+1} \in \mathsf{Span}\{x_t, Ax_t, v\}$. Hence, for $x_0 = \mathbf{0}$, it follows by induction that for any positive integer k,

$$x_k \in \mathsf{Span}\{v, Av, \ldots, A^k v\}.$$

The running time of the gradient descent-based method is dominated by the time required to compute a basis for this subspace. However, this vector x_k may not be a vector from this subspace that minimizes f_A. On the other hand, the essence of the Conjugate Gradient method is that it finds the vector in this subspace that minimizes f_A, in essentially the same amount of time required by k iterations of the gradient descent-based method. We must address two important questions about the Conjugate Gradient method: (1) Can the best vector be computed efficiently? and (2) What is the approximation guarantee achieved after k iterations? We show that the best vector can be found efficiently, and prove, using a low-degree polynomial approximations to x^k, that the Conjugate Gradient method achieves a quadratic improvement over the gradient descent-based method in terms of its dependence on the condition number of A.

Finding the best vector efficiently. Let $\{v_0, \ldots, v_k\}$ be a basis for $\mathcal{K} \stackrel{\text{def}}{=} \mathsf{Span}\{v, Av, \ldots, A^k v\}$ (called the *Krylov subspace of order k*). Hence, any vector in this subspace can be written as $\sum_{i=0}^k \alpha_i v_i$. Our objective then becomes

$$\|x^\star - \sum_i \alpha_i v_i\|_A^2 = (\sum_i \alpha_i v_i)^\top A(\sum_i \alpha_i v_i) - 2(\sum_i \alpha_i v_i)^\top v + \|x^\star\|_A^2 .$$

Solving this optimization problem for α_i requires matrix inversion, the very problem we set out to mitigate. The crucial observation is that if the v_is are A-orthogonal, *i.e.*, $v_i^\top A v_j = 0$ for $i \neq j$, then all the cross-terms disappear. Thus,

$$\|x^\star - \sum_i \alpha_i v_i\|_A^2 = \sum_i (\alpha_i^2 v_i^\top A v_i - 2\alpha_i v_i^\top v) + \|x^\star\|_A^2 ,$$

and, as in the gradient descent-based method, we can analytically compute the set of values α_i that minimize the objective to be given by $\alpha_i = \frac{v_i^\top v}{v_i^\top A v_i}$.

Hence, if we can construct an A-orthogonal basis $\{v_0, \ldots, v_k\}$ for $\mathcal{K}$ efficiently, we do at least as well as the gradient descent-based method. If we start with an arbitrary set of vectors and try to A-orthogonalize them via the Gram-Schmidt process (with inner products with respect to A), we need to compute k^2 inner products and, hence, for large k, it is not more efficient than the gradient descent-based method. An efficient construction of such a basis is one of the key ideas here. We proceed iteratively, starting with $v_0 = v$. At the i^{th} iteration, we compute Av_{i-1} and A-orthogonalize it with respect to $v_0, \ldots, v_{i-1}$, to obtain v_i. It is trivial to see that the vectors $v_0, \ldots, v_k$ are A-orthogonal. Moreover, it is not difficult to see that for every i, we have

$$\mathsf{Span}\{v_0, \ldots, v_i\} = \mathsf{Span}\{v, Av, \ldots, A^i v\}.$$

Now, since $Av_j \in \mathsf{Span}\{v_0, \ldots, v_{j+1}\}$ for every j, and A is symmetric, A-orthonormality of the vectors implies $v_i^\top A(Av_j) = v_j^\top A(Av_i) = 0$ for all j such that $j + 1 < i$. This implies that we need to A-orthogonalize Av_i only to vectors v_i and v_{i-1}. Hence, the time required for constructing this basis is dominated by $O(k)$ multiplications of the matrix A with a given vector, and $O(k)$ dot-product computations. Hence, we can find the best vector in the Krylov subspace efficiently enough.

Approximation guarantee. We now analyze the approximation guarantee achieved by this vector. Note that the Krylov subspace $\mathcal{K} = \mathsf{Span}\{v, Av, \ldots, A^k v\}$ consists of exactly those vectors which can be expressed as $\sum_{i=0}^{k} \beta_i A^i v = p(A)v$, where p is a degree-k polynomial defined by the coefficients β_i. Let Σ_k denote the set of all degree-k polynomials. Since the output vector x_k is the vector in the subspace that achieves the best possible error guarantee, we have $\|x_k - x^\star\|_A^2$

$$= \inf_{x \in \mathcal{K}} \|x^\star - x\|_A^2 = \inf_{p \in \Sigma_k} \|x^\star - p(A)v\|_A^2 \le \|x^\star\|_A^2 \cdot \inf_{p \in \Sigma_k} \|I - p(A)A\|^2.$$

Observe that the last expression can be written as

$$\|x^\star\|_A^2 \cdot \inf_{q \in \Sigma_{k+1}, q(0)=1} \|q(A)\|^2,$$

where the minimization is now over degree-$(k+1)$ polynomials q that evaluate to 1 at 0. Since A is symmetric and, hence, diagonalizable, we know that

$$\|q(A)\|^2 = \max_i |q(\lambda_i)|^2 \le \sup_{\lambda \in [\lambda_n, \lambda_1]} |q(\lambda)|^2, \tag{1.1}$$

where $0 < \lambda_n \le \cdots \le \lambda_1$ denote the eigenvalues of the matrix A. Hence, in order to prove that an error guarantee of $\|x_k - x^\star\|_A \le \delta \|x^\star\|_A$ is achieved after k rounds, it suffices to show that there exists a polynomial of degree $k+1$ that takes value 1 at 0, and whose magnitude is less than δ on the interval $[\lambda_n, \lambda_1]$.

As a first attempt, we consider the degree-s polynomial

$$q_0(x) \stackrel{\text{def}}{=} \left(1 - \tfrac{2x}{(\lambda_1 + \lambda_n)}\right)^s.$$

The maximum value attained by q_0 over the interval $[\lambda_n, \lambda_1]$ is $({}^{(\kappa-1)}/_{(\kappa+1)})^s$. Hence, $d_0 \stackrel{\text{def}}{=} \lceil \kappa \log {}^1/_\delta \rceil$ suffices for this value to be less than δ. Or equivalently, approximately $\kappa \log {}^1/_\delta$ rounds suffice for error guarantee $\|x - x^\star\|_A \leq \delta \|x^\star\|_A$, recovering the guarantee provided by the gradient descent-based method.

However, for a better guarantee, we can apply the polynomial approximation to x^{d_0}. The proof is simple and appears in Chapter 3 of Sachdeva and Vishnoi [SV14].

Theorem 1.2. For any positive integers s and d, there is a degree-d polynomial $p_{s,d}$ such that

$$\sup_{x\in[-1,1]} |p_{s,d}(x) - x^s| \leq 2e^{-d^2/2s}.$$

Hence, for any $\delta > 0$, and $d \geq \left\lceil \sqrt{2s \log ({}^2/_\delta)} \right\rceil$, we have $\sup_{x\in[-1,1]} |p_{s,d}(x) - x^s| \leq \delta$.

Let $z \stackrel{\text{def}}{=} 1 - {}^{2x}/_{(\lambda_1+\lambda_n)}$. Hence, $q_0(x) = z^s$. As x ranges over $[0, \lambda_n + \lambda_1]$, the variable z varies over $[-1, 1]$. Theorem 1.2 implies that for $d \stackrel{\text{def}}{=} \left\lceil \sqrt{2d_0 \log {}^2/_\delta} \right\rceil$, the polynomial $p_{d_0,d}(z)$ approximates the polynomial z^{d_0} up to an error of δ over $[-1, 1]$. Hence, the polynomial $q_1(x) \stackrel{\text{def}}{=} p_{d_0,d}(z)$ approximates $q_0(x)$ up to δ for all $x \in [0, \lambda_1 + \lambda_n]$. Combining this with the observations from the previous paragraph, $q_1(x)$ takes value at most 2δ on the interval $[\lambda_n, \lambda_1]$, and at least $1 - \delta$ at 0. Thus, the polynomial ${}^{q_1(x)}/_{q_1(0)}$ is a polynomial of degree $d = O(\sqrt{\kappa} \log {}^1/_\delta)$ that takes value 1 at 0, and at most ${}^{2\delta}/_{(1-\delta)} = O(\delta)$ on the interval $[\lambda_n, \lambda_1]$. Or equivalently, $O(\sqrt{\kappa} \log {}^1/_\delta)$ rounds suffice for an error guarantee $\|x - x^\star\|_A \leq O(\delta)\|x^\star\|_A$, which gives a quadratic improvement over the guarantee provided by the gradient descent-based method. This completes the proof of Theorem 1.1.

1.3 Matrices with Clustered Eigenvalues

As another corollary of the characterization of (1.1) we show that the rate of convergence of the Conjugate Gradient method is better if the eigenvalues of A are *clustered*. This partly explains why it is attractive in practice; it is used in the next lecture to construct fast Laplacian solvers.

Corollary 1.3. For a matrix A, suppose all eigenvalues are contained in a range $[l, u]$ and the rest are more than u but there are at most c of them. Then, after $t \geq c + O(\sqrt{{}^u/_l} \log {}^1/_\varepsilon)$ iterations,

$$\|x_t - x^\star\|_A \leq \varepsilon \|x^\star\|_A.$$

Proof. Let $q(x) \stackrel{\text{def}}{=} q_{l,u,j}(x) \cdot \Pi_{i=1}^c (1 - {}^x/_{\lambda_i})$ where $\lambda_1, \ldots, \lambda_c$ are the c eigenvalues more than u, and $q_{l,u,j}$ is the degree-j polynomial for the interval $[l, u]$ as described in the previous section. Since the value of $q(\lambda_i) = 0$ for all $i = 1, \ldots, c$, we only need to consider the maximum value of $|q(x)|^2$ in the range $[l, u]$. By the properties of $q_{l,u,j}$ described above, if we pick $j = \Theta(\sqrt{u/l} \log 1/\varepsilon)$, for all $x \in [l, u]$, $|q(x)|^2 \leq \varepsilon$. Therefore, the Conjugate Gradient method returns an ε-approximation in $c + O(\sqrt{u/l} \log 1/\varepsilon)$ steps; note that this could be much better than $\sqrt{\kappa(A)}$ since no restriction is put on the c eigenvalues. $\square$

2

Preconditioning for Laplacian Systems

We introduce the notion of preconditioning: Instead of solving $Ax = b$, here one tries to solve $PAx = Pb$ for a matrix P such that $\kappa(PA) \ll \kappa(A)$ where it is not much slower to compute Pv than A^+v, thus speeding up iterative methods such as the conjugate gradient method. As an application, preconditioners are constructed for Laplacian systems from low-stretch spanning trees that result in a $\tilde{O}(m^{4/3})$ time Laplacian solver.[1]

2.1 Preconditioning

We saw in the previous lecture that using the Conjugate Gradient method for a symmetric matrix $A \succ 0$, one can solve a system of equations $Ax = b$ in time $O(t_A \cdot \sqrt{\kappa(A)} \log 1/\varepsilon)$ up to an error of ε. Let us focus on the two quantities in this running time bound: t_A which is the time it takes to multiply a vector with A, and $\kappa(A)$, the condition number of A. Note that the algorithm requires only restricted access to A: Given a vector v, output Av. Thus, one strategy to reduce this running time is to find a matrix P s.t. $\kappa(PA) \ll \kappa(A)$ and $t_P \sim t_A$. Indeed, if we had access to such a matrix, we could solve the equivalent system of equations $PAx = Pb$.[2] Such a matrix P is called a *preconditioner* for A. One choice for P is A^+. This reduces the condition number to 1 but reduces the problem of computing A^+b to itself, rendering it useless. Surprisingly, as we will see in this chapter, for a Laplacian system one can often find preconditioners by using the graph structure,

[1] This is almost identical to Chapter 17 from Vishnoi [Vis13].

[2] One might worry that the matrix PA may not be symmetric; one normally gets around this by preconditioning by $P^{1/2}AP^{1/2}$. This requires $P \succ 0$.

thereby reducing the running time significantly. The following theorems are the main result of this chapter.

Theorem 2.1. For any undirected, unweighted graph G with m edges, a vector b with $\langle b, 1\rangle = 0$, and $\varepsilon > 0$, one can find an x such that $\|x - L_G^+ b\|_{L_G} \leq \varepsilon \|L_G^+ b\|_{L_G}$ in $\tilde{O}(m^{4/3} \log 1/\varepsilon)$ time.

There are two crucial ingredients to the proof. The first is the following simple property of the Conjugate Gradient method.

Lemma 2.2. Suppose A is a symmetric positive definite matrix with minimum eigenvalue $\lambda_1 \geq 1$ and trace $\mathrm{Tr}(A) \leq \tau$. Then the Conjugate Gradient method converges to an ε-approximation in $O(\tau^{1/3} \log 1/\varepsilon)$ iterations.

Proof. Let Λ be the set of eigenvalues larger than τ/γ, where γ is a parameter to be set later. Note that $|\Lambda| \leq \gamma$. Therefore, apart from these γ eigenvalues, all the rest lie in the range $[1, \tau/\gamma]$. From Corollary 1.3, we get that the Conjugate Gradient method finds an ε-approximation in $\gamma + O(\sqrt{\tau/\gamma} \log 1/\varepsilon)$ iterations. Choosing $\gamma \stackrel{\text{def}}{=} \tau^{1/3}$ completes the proof of the lemma. $\square$

The second ingredient is a construction of a combinatorial preconditioner for L_G. The choice of preconditioner is L_T^+ where T is a spanning tree of G. It is an easy exercise to see that this preconditioner satisfies the property that $L_T^+ v$ can be computed in $O(n)$ time. Thus, the thing we need to worry about is how to construct a spanning tree T of G such that $\kappa(L_T^+ L_G)$ is as small as possible.

2.2 Combinatorial Preconditioning via Trees

Let G be an unweighted graph with an arbitrary orientation fixed for the edges giving rise to the vectors b_e which are the rows of the corresponding incidence matrix. We start by trying to understand why, for a spanning tree T of a graph G, L_T^+ might be a natural candidate for preconditioning L_G. Note that

$$L_T = \sum_{e \in T} b_e b_e^\top \preceq \sum_{e \in G} b_e b_e^\top = L_G.$$

Thus, $I \prec L_T^+ L_G$, which implies that $\lambda_1(L_T^+ L_G) \geq 1$. Thus, to bound the condition number of $\kappa(L_T^+ L_G)$, it suffices to bound $\lambda_n(L_T^+ L_G)$. Unfortunately, there is no easy way to bound this. Since $L_T^+ L_G$ is PSD, an upper bound on its largest eigenvalue is its trace, $\mathrm{Tr}(L_T^+ L_G)$. If this upper bound is τ, Lemma 2.2 would imply that the conjugate gradient method applied to $L_T^+ L_G$ takes time approximately $\tau^{1/3} t_{L_T^+ L_G} \sim O(\tau^{1/3}(m+n))$. Thus, even though it sounds wasteful to bound the trace by τ rather than by the largest eigenvalue by τ, Lemma 2.2 allows us to improve the dependency on τ from $\tau^{1/2}$ to $\tau^{1/3}$. We proceed to bound the trace of $L_T^+ L_G$:

$$\mathrm{Tr}(L_T^+ L_G) = \mathrm{Tr}\left(L_T^+ \sum_{e \in G} b_e b_e^\top\right) = \sum_e b_e^\top L_T^+ b_e,$$

where we use $\mathrm{Tr}(A+B) = \mathrm{Tr}(A) + \mathrm{Tr}(B)$ and $\mathrm{Tr}(ABC) = \mathrm{Tr}(CAB)$. Note that $b_e^\top L_T^+ b_e$ is a scalar and is precisely the effective resistance across the endpoints of e in the tree T where each edge of G has a unit resistance. The effective resistance across two nodes i, j in a tree is the sum of effective resistances along the unique path $P(i, j)$ on the tree. Thus, we get

$$\mathrm{Tr}(L_T^+ L_G) = \sum_{e \in G} |P(e)|.$$

A trivial upper bound on $\mathrm{Tr}(L_T^+ L_G)$, thus, is nm. This can also be shown to hold when G is weighted. This leads us to the following definition.

Definition 2.3. For an unweighted graph G, the *stretch* of a spanning T is defined to be $\mathrm{str}_T(G) \stackrel{\text{def}}{=} \mathrm{Tr}(L_T^+ L_G)$.

Thus, to obtain the best possible bound on the number of iterations of the Conjugate Gradient method, we would like a spanning tree T of G which minimizes the average length of the path an edge of G has to travel in T. The time it takes to construct T is also important.

2.3 An $\tilde{O}(m^{4/3})$-Time Laplacian Solver

In this section we complete the proof of Theorem 2.1. We start by stating the following non-trivial structural result about the existence and construction of low-stretch spanning trees whose proof is graph-theoretic and outside the scope of this monograph. The theorem applies to weighted graphs as well.

Theorem 2.4. For any undirected graph G, a spanning tree T can be constructed in $\tilde{O}(m \log n + n \log n \log\log n)$ time such that $\mathrm{str}_T(G) = \tilde{O}(m \log n)$. Here $\tilde{O}(\cdot)$ hides $\log\log n$ factors.

This immediately allows us to conclude the proof of Theorem 2.1 using Lemma 2.2 and the discussion in the previous section. The only thing that remains is to address the issue that the matrix $L_T^+ L_G$ is symmetric. The following trick is used to circumvent this difficulty. One can compute the Cholesky decomposition of L_T in $O(n)$ time. In particular, let $L_T = EE^\top$, where E is a lower triangular matrix with at most $O(n)$ non-zero entries. The idea is to look at the system of equations $E^+ L_G E^{+\top} y = E^+ b$ instead of $L_G x = b$. If we can solve for y then we can find $x = E^{+\top} y$, which is computationally fast since E is lower triangular with at most $O(n)$ non-zero entries. Also, for the same reason, $E^+ b$ can be computed quickly.

Now we are in good shape. Let $A \stackrel{\text{def}}{=} E^{+}L_G E^{+\top}$ and $b' \stackrel{\text{def}}{=} E^{+}b$; note that A is symmetric. Also, the eigenvalues of A are the same as those of $E^{\top +}AE^{\top} = L_T^{+}L_G$. Thus, the minimum eigenvalue of A is 1 and the trace is $\tilde{O}(m)$ by Theorem 2.4. Thus, using the conjugate gradient method, we find an ε-approximate solution to $Ay = b'$ in $\tilde{O}(m^{1/3})$ iterations.

In each iteration, we do $O(1)$ matrix-vector multiplications. Note that for any vector v, Av can be computed in $O(n+m)$ operations since $E^{+}v$ takes $O(n)$ operations (since E is a lower triangular matrix with at most $O(n)$ non-zero entries) and $L_G v'$ takes $O(m)$ operations (since L_G has at most $O(m)$ non-zero entries).

3

Newton's Method and the Interior Point Method

In this lecture we continue our journey towards faster (and better) algorithms for convex programs. The algorithms introduced till now assume access only to an oracle for the value of the convex function and its gradient at a specified point. In this lecture we assume that we are also given a *second order* access to f: namely, given vectors x and y, we could obtain $(\nabla^2 f(x))^{-1} y$. The resulting method would be Newton's method for solving unconstrained programming and will have this property that if one starts close enough to the optimal solution the convergence would be in $\log \log {}^1\!/\!_\varepsilon$ iterations! Finally, we present the application of Newton's method to solving constrained convex programs. This is achieved by moving from constrained to unconstrained optimization via a *barrier* function. The resulting methods are broadly termed as interior point methods. We analyze one such method, referred to as the *primal path-following interior point method*, for linear programs. We end this lecture by a discussion on *self-concordant* barrier functions which allow us to go beyond linear programs to more general convex programs.[1]

3.1 Newton's Method and its Quadratic Convergence

Our starting point is the versatile Newton's method which we first explain in the simplest setting of finding a root for a univariate polynomial.

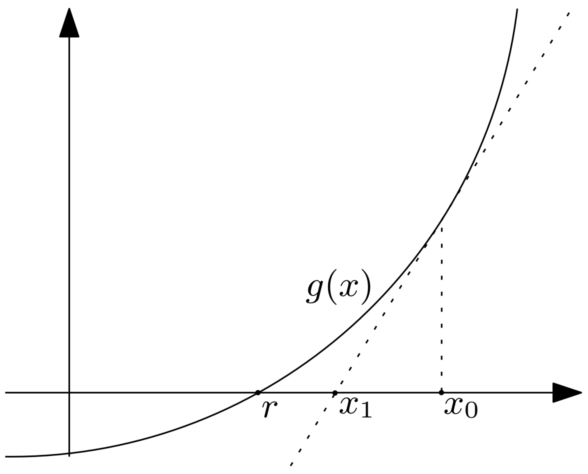

Fig. 3.1 One step of Newton's Method

3.1.1 Newton-Raphson method

In numerical analysis, Newton's method (also known as the Newton-Raphson method), named after Isaac Newton and Joseph Raphson, is a method for finding successively better approximations to the roots (or zeroes) of a real-valued function. Suppose we are given a function $g : \mathbb{R} \mapsto \mathbb{R}$ and we want to find its root (or one of its roots). Assume we are given a point x_0 which is likely to be close to a zero of g. We consider the point $(x_0, g(x_0))$ and draw a line through it which is tangent to the graph of g. Let x_1 be the intersection of the line with the x-axis (see Figure 3.1.1). Then it is reasonable (at least if one were to believe the figure above) to suppose that by moving from x_0 to x_1 we have made progress in reaching a zero of g. First note that:

$$x_1 \stackrel{\text{def}}{=} x_0 - \frac{g(x_0)}{g'(x_0)}$$

From x_1, by the same method we obtain x_2, then x_3 etc. Hence, the general formula is:

$$x_{k+1} \stackrel{\text{def}}{=} x_k - \frac{g(x_k)}{g'(x_k)} \qquad \text{for all } k \geq 0. \tag{3.1}$$

Of course we require differentiability of g, in fact we will assume even more – that g is twice continuously differentiable. Let us now analyze how fast the distance to the root decreases with k.

Let r, be the root of g, that is $g(r) = 0$. Expand g into Taylor series at the point

[1] This lecture appears as Lecture 3 in Vishnoi [Vis14].

x_k and, use the Mean Value Theorem, to obtain the following:

$$g(r) = g(x_k) + (r - x_k)g'(x_k) + \frac{1}{2}(r - x_k)^2 g''(\theta)$$

for some θ in the interval $[r, x_k]$. From (3.1) we know that

$$g(x_k) = g'(x_k)(x_k - x_{k+1}).$$

Recall also that $g(r) = 0$. Hence, we get:

$$0 = g'(x_k)(x_k - x_{k+1}) + (r - x_k)g'(x_k) + \frac{1}{2}(r - x_k)^2 g''(\theta)$$

which implies that

$$g'(x_k)(r - x_{k+1}) = \frac{1}{2}(r - x_k)^2 g''(\theta).$$

This gives us the relation between the new distance from the root in terms of the old distance from it:

$$|r - x_{k+1}| = \left|\frac{g''(\theta)}{2g'(x_k)}\right| |r - x_k|^2.$$

This can be summarized in the following theorem:

Theorem 3.1. Suppose $g : \mathbb{R} \mapsto \mathbb{R}$ is a $\mathcal{C}^2$ function, [2] $r \in \mathbb{R}$ is a root of g, $x_0 \in \mathbb{R}$ is a starting point and $x_1 = x_0 - \frac{g(x_0)}{g'(x_0)}$, then:

$$|r - x_1| \leq M|r - x_0|^2$$

where $M = \sup_{x \in [r, x_0]} \left|\frac{g''(\theta)}{2g'(x)}\right|$.

Thus, assuming that M is a small constant, say $M \leq 1$ (and remains so throughout the execution of this method) and that $|x_0 - r| < 1$, we obtain *quadratically* fast convergence of x_k to r. For the error $|x_k - r|$ to became less then ε one needs to take $k = \log\log {}^1\!/\!_\varepsilon$. As one can imagine, for this reason Newton's Method is very efficient and powerful. In practice, it gives very good results even when no reasonable bounds on M or $|x_0 - r|$ are available.

3.1.2 Newton's Method for Convex Optimization

How could the benign looking Newton-Raphson method be useful to solve convex programs? The key lies in the observation from the first lecture that the task of minimization of a differentiable convex function in the unconstrained setting is equivalent to *finding a root of its derivative*. In this section we abstract out the method from the previous section and present Newton's method for convex programming.

Recall that the problem is to find

$$x^\star \stackrel{\text{def}}{=} \arg \inf_{x \in \mathbb{R}^n} f(x).$$

[2] The function is twice differentiable and the second derivative is continuous.

where f is a convex (smooth enough) function. The gradient ∇f of f is a function $\mathbb{R}^n \mapsto \mathbb{R}^n$ and its derivative $\nabla^2 f$ maps $\mathbb{R}^n$ to $n \times n$ symmetric matrices. Hence, the right analog of the update formula (3.1) to our setting can be immediately seen to be:

$$x_{k+1} \overset{\text{def}}{=} x_k - (\nabla^2 f(x_k))^{-1} \nabla f(x_k) \qquad \text{for all } k \geq 0. \tag{3.2}$$

For notational convenience we define *the Newton step* at point x to be

$$n(x) \overset{\text{def}}{=} -(\nabla^2 f(x))^{-1} \nabla f(x),$$

then (3.2) gets abbreviated to $x_{k+1} = x_k + n(x_k)$. One may convince themselves that (3.2) is meaningful by applying it to f being a strictly convex quadratic function (i.e. $f(x) = x^\top Mx$ for M positive definite). Then, no matter which point we start, after one iteration we land in the unique minimizer. This phenomenon can be explained as follows: suppose $\tilde{f}$ is the second order approximation of f at point x,

$$\tilde{f}(y) = f(x) + (y-x)^\top \nabla f(x) + \frac{1}{2}(y-x)^\top \nabla^2 f(x)(y-x)$$

If f is sctrictly convex then its Hessian is positive definite, hence the minimizer of $\tilde{f}(y)$ is

$$y^\star = x - (\nabla^2 f(x))^{-1} \nabla f(x) = x + n(x)$$

For this reason Newton's method is called a second-order method, because it takes advantage of the second order approximation of a function to make a step towards the minimum. All the algorithms we looked at in the previous lecture were first-order methods. They used only the gradient (first order approximation) to perform steps. However, computationally our task has increased as now we would need a second order oracle to the function: given x and y, we would need to solve the system of equations $\nabla^2 f(x) y = \nabla f(x)$.

The next question is if, and, under what conditions x_k converges to the minimizer of f. It turns out that it is possible to obtain a similar quadratic convergence guarantee assuming that x_0 is sufficiently close to the minimizer $x^\star$. We can prove the following theorem whose hypothesis and implications should be compared to Theorem 3.1.

Theorem 3.2. Let $f : \mathbb{R}^n \mapsto \mathbb{R}$ be a $\mathcal{C}^2$ function and $x^\star$ be its minimizer. Denote the gradient $\nabla^2 f(x)$ by $H(x)$ and assume that the following hold:

- There is some constant $h > 0$ and a ball $B(x^\star, r)$ around $x^\star$ such that, whenever $x \in B(x^\star, r)$, $\|H(x)^{-1}\| \leq \frac{1}{h}$.
- There is some constant $L > 0$ and a ball $B(x^\star, r)$ around $x^\star$ such that, whenever $x, y \in B(x^\star, r)$, $\|H(x) - H(y)\| \leq L\|x - y\|$.

If x_0 is a starting point, sufficiently close to $x^\star$ and $x_1 = x_0 + n(x)$ then:

$$\|x_1 - x^\star\| \leq M \|x_0 - x^\star\|^2$$

for some constant M. For example $M = \frac{L}{2h}$ will do.

Thus, if $M \leq 1$, then with a starting point close enough to the optimal solution, Newton's method converges quadratically fast. One can present a rough analogy of this theorem with Theorem 3.1. There, for the method to have quadratic convergence, $|g'(x)|$ should be bigger in comparison to $|g''(x)|$ (to end up with small M). Here, the role of g is played by the derivative f' of f (the gradient in the one-dimensional case). The first condition on $H(x)$ says basically that $|f''(x)|$ is big. The second condition may be a bit more tricky to decipher, it says that $f''(x)$ is Lipschitz-continuous, and upper-bounds the Lipschitz constant. Assuming f is thrice continuously differentiable, this essentially gives an upper bound on $|f'''(x)|$. Note that this intuitive explanation does not make any formal sense, since $f'(x), f''(x), f'''(x)$ are not numbers, but vectors, matrices and 3-tensors respectively. We only wanted to emphasize that the spirit of Theorem 3.2 still remains the same as Theorem 3.1.

Proof. [Proof of Theorem 3.2] The basic idea of the proof is the same as in 3.1. We need a similar tool as the Taylor expansion used in the previous chapter. To obtain such, we consider the function $\phi : [0,1] \to \mathbb{R}^n$, $\phi(t) = \nabla f(x + t(y - x))$. Applying the fundamental theorem of calculus to ϕ (to every coordinate separately) yields:

$$\begin{aligned} \phi(1) - \phi(0) &= \int_0^1 \nabla\phi(t)dt \\ \nabla f(y) - \nabla f(x) &= \int_0^1 H(x + t(y - x))(x - y)dt. \end{aligned} \tag{3.3}$$

Let $x = x_0$ for notational convenience and write $x_1 - x^\star$ in a convenient form:

$$\begin{aligned} x_1 - x^\star &= x - x^\star + n(x) \\ &= x - x^\star - H(x)^{-1}\nabla f(x) \\ &= x - x^\star + H(x)^{-1}(\nabla f(x^\star) - \nabla f(x)) \\ &= x - x^\star + H(x)^{-1}\int_0^1 H(x + t(x^\star - x))(x - x^\star)dt \\ &= H(x)^{-1}\int_0^1 (H(x + t(x^\star - x)) - H(x))(x - x^\star)dt. \end{aligned}$$

Now take norms:

$$\begin{aligned} \|x_1 - x^\star\| &\leq \|H(x)^{-1}\| \int_0^1 \|(H(x + t(x^\star - x)) - H(x))(x - x^\star)\|dt \\ &\leq \|H(x)^{-1}\|\|x - x^\star\| \int_0^1 \|(H(x + t(x^\star - x)) - H(x))\|dt. \end{aligned} \tag{3.4}$$

We use the Lipschitz condition on H to bound the integral:

$$\begin{aligned} \int_0^1 \|(H(x + t(x^\star - x)) - H(x))\|dt &\leq \int_0^1 L\|t(x^\star - x)\|dt \\ &\leq L\|x^\star - x\| \int_0^1 tdt \\ &= \frac{L}{2}\|x^\star - x\|. \end{aligned}$$

Together with (3.4) this implies:

$$\|x_1 - x^\star\| \leq \frac{L\|H(x)^{-1}\|}{2}\|x^\star - x\|^2 \tag{3.5}$$

which completes the proof. We can take $M = \frac{L\|H(x)^{-1}\|}{2} \leq \frac{L}{2h}$. $\square$

3.2 Constrained Convex Optimization via Barriers

In this section return to constrained convex optimization problems of the form:

$$\inf_{x \in K} f(x) \tag{3.6}$$

where f is a convex, real valued function and $K \subseteq \mathbb{R}^n$ is a convex set. [3] In the first lecture we discussed how gradient-descent type methods could be adapted in this setting by projecting onto K at every step. We took an improvement step $x_k \mapsto x'_{k+1}$ with respect to f and then we projected x'_{k+1} onto K to obtain x_{k+1}. There are a few problems with this method. One of them is that in most cases computing projections is prohibitively hard. Furthermore, even if we ignore this issue, the projection-based methods are not quiet efficient. To get an ε-approximation to the solution, the number of iterations depends polynomially on ε^{-1} (i.e. the number of iterations is proportional to $1/\varepsilon^{O(1)}$). Unfortunately such dependence on ε is often unsatisfactory. For example, to obtain the optimal solution for a linear program we need to take ε of the form 2^{-L}, where L is the size of the instance.[4] Hence, to get a polynomial time algorithm, we need the running time dependence on ε to be $\log^{O(1)}(1/\varepsilon)$. Today we will see an interior point algorithm which achieve such a guarantee.

3.2.1 Following the Central Path

We are going to present one very general idea for solving constrained optimization problems. Recall that our aim is to minimize a given convex function $f(x)$ subject to $x \in K$. To simplify our discussion, we assume that the objective function is linear, [5] i.e. $f(x) = c^\top x$ and the convex body K is bounded and full-dimensional (it has positive volume).

Suppose we have a point $x_0 \in K$ and we want to perform an improvement step maintaining the condition of being inside K. The simplest idea would be to move in the direction $-c$ to decrease our objective value as much as possible. Our step will then end up on the boundary of K. The second and further points would lie very close to the boundary, which will force our steps to be short and thus inefficient. In case

[3] K is given to us either explicitly – by a collection of constraints defining it, or by a separation oracle.

[4] One should think of L as the total length of all the binary encodings of numbers in the description of the linear program. In the linear programming setting, $L^{O(1)}$ can be shown to bound the number of binary digits required to represent the coordinates of the optimal solution.

[5] Actually we are not losing on generality here. Every convex problem can be stated equivalently with a linear objective function.

of K being a polytope, such a method would be equivalent to the simplex algorithm, which as known to have an exponential worst case running time. For this reason we need to set some force, which would repel us from the boundary of K. More formally, we want to move our constraints to the objective function and consider $c^\top x + F(x)$,[6] where $F(x)$ can be regarded as a "fee" for violating constraints. $F(x)$ should become big for x close to ∂K. Of course, if we would like the methods developed in the previous sections for unconstrained optimization to be applicable here, we would also like f to be strongly convex. Thus, this is another route we could take to convert a constrained minimization problem into unconstrained minimization, but with a slightly altered objective function. To formalize this approach we introduce the notion of a *Barrier Function.* Instead of giving a precise definition, we list some properties, which we wish to hold for a barrier function F:

- F is defined in the interior of K, i.e. $\mathrm{dom}(F) = \mathrm{int}(K)$, (3.7)
- for every point $b \in \partial K$ we have: $\lim_{x \to b} F(x) = +\infty$, (3.8)
- F is strictly convex. (3.9)

Suppose F is such a barrier function, let us define a perturbed objective function f_η, where $\eta > 0$ is a real parameter:

$$f_\eta(x) \stackrel{\text{def}}{=} \eta c^\top x + F(x) \tag{3.10}$$

We may imagine that f_η is defined on all of $\mathbb{R}^n$ but attains finite values only on $\mathrm{int}(K)$. Intuitively, making η bigger and bigger reduces the influence of $F(x)$ on the optimal value of $f_\eta(x)$. Furthermore, observe that since $c^\top x$ is a linear function, the second order behavior of f_η is completely determined by F, that is $\nabla^2 f_\eta = \nabla^2 F$. In particular, f_η is strictly convex and it has a unique minimizer $x^\star_\eta$. The set

$$\{x^\star_\eta : \eta \geq 0\}$$

can be seen to be continuous due to the Implicit Function Theorem and is referred to as a *central path* starting at $x^\star_0$ and approaching $x^\star$ – the solution to our convex problem 3.6. In other words:

$$\lim_{\eta \to \infty} x^\star_\eta = x^\star.$$

A method which follows this general approach is called a *path-following interior point method.* Now, the key question is: "how fast is this convergence?". Needless to say, this would depend on the choice of the barrier function and the method for solving the unconstrained optimization problem. We answer it in the next section for linear programs: using the logarithmic barrier function along with Newton's method from the previous section we can give an algorithm to solve linear programs in time polynomial in the encoding length.

[6] One seemingly perfect choice of F would be a function which is 0 on K and $+\infty$ on the complement of K. This reduces our problem to unconstrained minimization of $c^\top x + F(x)$. However, note that we have not gained anything by this reformulation, even worse: our objective is not continuos anymore.

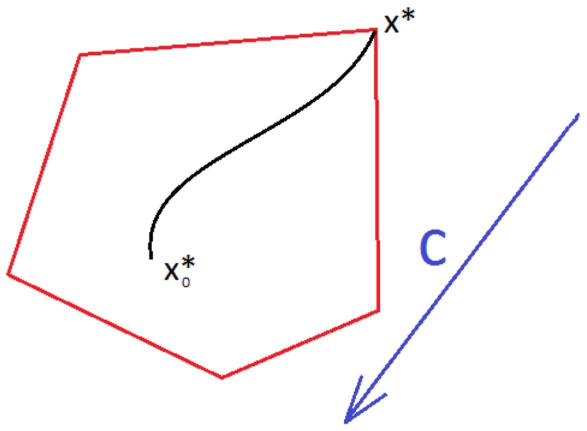

Fig. 3.2 Example of a central path

3.3 Interior Point Method for Linear Programming

3.3.1 A Brief History of IPMs

Interior point methods (IPMs), as alluded to in the previous section, first appear in a recognizable form in the Ph.D. thesis of Ilya Dikin in the 60s. However, there was no analysis for the time needed for convergence. In 1984 Narendra Karmarkar announced a polynomial-time algorithm to solve linear programs using an IPM. At that point there was already a known polynomial time algorithm for solving LPs, namely the Ellipsoid Algorithm from 1979. Further, the method of choice in practice, despite known to be inefficient in the worst case, was the Simplex method. However, in his paper, he also presented empirical results which showed that his algorithm was consistently 50 times faster than the simplex method. This event, which received publicity around the world throughout the popular press and media, marks the beginning of the interior-point revolution. For a nice historical perspective on this revolution, we refer the reader to the survey of Wright [Wri05].

Karmarkar's algorithm needed roughly $O(m \log {}^1\!/\!_\varepsilon)$ iterations to find a solution (which is optimal up to an additive error of ε) to a linear program, where m is the number of constraints. Each such iteration involved solving a linear system of equations, which could be easily performed in polynomial time. Thanks to the logarithmic dependence on the error ε it can be used to find the exact, optimal solution to a linear program in time polynomial with respect to the encoding size of the problem. Thus, Karmarkar's algorithm was the first efficient algorithm with provable worst case polynomial running time. Subsequently, James Renegar proposed an interior point algorithm with reduced number of iterations: $O(\sqrt{m}\log({}^1\!/\!_\varepsilon))$. Around the same time, Nesterov and Nemirovski abstracted out the essence of interior point methods and came up with the the notion of *self-concordance*, which in turn was used to provide efficient, polynomial time algorithms for many nonlinear convex problems such as semi-definite programs.

We begin our discussion by introducing the logarithmic barrier function and then Newton's Method, which is at a core of all interior-point algorithms. Then we present Renegar's primal path following method. We give a full analysis of this algorithm, i.e. we prove that after $\widetilde{O}(\sqrt{m}\log(1/\varepsilon))$ it outputs a solution which is ε-close to the optimum.

3.3.2 The Logarithmic Barrier

In this section we switch from a general discussion on constrained convex optimization to a particular problem: linear programming. We will use ideas from the previous section to obtain an efficient algorithm for solving linear programs in the form:

$$\begin{aligned} &\min_{x\in\mathbb{R}^n} c^\top x \\ &\text{s.t. } Ax \le b \end{aligned} \tag{3.11}$$

where x is the vector of variables of length n, $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$ and $A \in \mathbb{R}^{m\times n}$. We will denote the rows of A by $a_1, a_2, \ldots, a_m$ (but treat them as column vectors). We will assume that the set of constraints $Ax \le b$ defines a bounded polytope P of nonzero volume in $\mathbb{R}^n$.[7]

We are going to use the following barrier function which is often called the *logarithmic barrier* function:

$$F(x) \stackrel{\text{def}}{=} -\sum_{i=1}^{m} \log(b_i - a_i^\top x)$$

It is easy to see that $F(x)$ is well defined on $\text{int}(P)$ and tends to infinity when approaching the boundary of P. Let us now write down formulas for the first and second derivative of F, they will prove useful a couple of times in the analysis. To simplify the notation, we will often write $s_i(x)$ for $b_i - a_i^\top x$.

Fact 3.3. If x is a point in the interior of P, then:

(1) $\nabla F(x) = \sum_{i=1}^m \frac{a_i}{s_i(x)}$
(2) $\nabla^2 F(x) = \sum_{i=1}^m \frac{a_i a_i^\top}{s_i(x)^2}$

[7] Of course this is not always the case for general linear programs, however if the program is feasible then we can perturb the constraints by an exponentially small $\varepsilon > 0$ to force the feasible set to be full dimensional. Moreover, it will turn out soon that infeasibility is not a serious issue.

Using the above formulas we can investigate the convexity of F. It is clear that $\nabla^2 F(x) \succeq 0$, which implies convexity. Strong convexity of F (i.e. $\nabla^2 F(x) \succ 0$) is equivalent to the fact that $a_1, a_2, \ldots, a_n$ span the whole $\mathbb{R}^n$, in our case this is true, because of the assumption that P is full dimensional and bounded. Which is which should be clear from the context.

From now on, whenever we talk about a function F we will write $H(x)$ for its Hessian $\nabla^2 F(x)$ and $g(x)$ for the gradient $\nabla F(x)$. Note however that at some places in the text we refer to F as a general function and sometimes we fix F to be a specific one: the logarithmic barrier.

3.3.3 Local Norms

In this section we introduce an important concept of a local norm. It plays a crucial role in understanding interior point methods. Rather than working with the Euclidean norm, the analysis of our algorithm will be based on bounding the local norm of the Newton step. Let $A \in S^n$ be a positive definite matrix, we associate with it the inner product $\langle \cdot, \cdot \rangle_A$ defined as:

$$\langle x, y \rangle_A \stackrel{\text{def}}{=} x^\top A y$$

and the norm $\| \cdot \|_A$:

$$\|x\|_A \stackrel{\text{def}}{=} \sqrt{x^\top A x}$$

Note that a ball of radius 1 with respect to such a norm corresponds to an ellipsoid in the Euclidean space. Formally, we define the ellipsoid associated with the matrix A, centered at $x_0 \in \mathbb{R}^n$ as:

$$\mathcal{E}_{x_0}(A) \stackrel{\text{def}}{=} \{x : (x - x_0)^\top A (x - x_0) \leq 1\}.$$

Matrices of particular interest are for us Hessians of strictly convex functions $F : \mathbb{R}^n \mapsto \mathbb{R}$ (such as the logarithmic barrier). For them we usually write in short:

$$\|z\|_x \stackrel{\text{def}}{=} \|z\|_{H(x)}$$

whenever the function F is clear from the context. This is called a local norm at x with respect to F. From now on, let $F(x) = \sum_{i=1}^m - \log(b_i - a_i^\top x)$ be the logarithmic barrier function. For the logarithmic barrier, $\mathcal{E}_x(\nabla^2 F(x))$ is called the Dikin Ellipsoid centered at x. An important property of this ellipsoid is that it is contained in P and that the Hessian does not change much. Thus, the curvature of the central path with respect to the local norm does not change much and, hence, it is close to a straight line. Thus, taking a short Newton step inside this ellipsoid from the center keeps one close to the central path.

We return to this intimate connection between the Dikin Ellipsoid and IPMs in the appendix. We conclude this section by noting that there is another way to motivate measuring progress in the local norm: that Newton's method is *affinely invariant*. This means that if, for a invertible linear transformation A, we do a change of variables $x = Ay$, then the Newton step is just a transformation by A: $n(x) = An(y)$. On the other hand the Lipschitz condition on the Hessian in

Theorem 3.2 is not affine invariant w.r.t. the Euclidean norm. However, it would be if we redefine it as

$$\|H(x) - H(y)\| \leq L\|x - y\|_x,$$

this remains affine invariant.

3.3.4 A Primal Path-Following IPM

We come back to the description of the path following algorithm. Recall that we defined the central path as: $\{x^\star_\eta : \eta > 0\}$ consisting of optimal solutions to the perturbed linear program. We want to start at $x_0 = x^\star_{\eta_0}$ for some small $\eta_0 > 0$ and move along the path by taking discrete steps of certain length. This corresponds essentially to increasing η in every step. So one idea for our iterative procedure would be to produce a sequence of pairs $(x_0, \eta_0), (x_1, \eta_1), \ldots$ such that $\eta_0 < \eta_1 < \cdots$ and $x_k = x^\star_{\eta_k}$ for every k. We should finish at the moment when we are close enough to the optimum, that is when $c^\top x_k < c^\top x^\star + \varepsilon$. Our first lemma gives a bound on when to stop:

Lemma 3.4. For every $\eta > 0$ we have $c^\top x^\star_\eta - c^\top x^\star < \frac{m}{\eta}$.

Proof. Calculate first the derivative of $f_\eta(x)$:

$$\nabla f_\eta(x) = \nabla(\eta c^\top x + F(x)) = \eta c + \nabla F(x) = \eta c + g(x)$$

The point $x^\star_\eta$ is the minimum of f_η, hence $\nabla f_\eta(x^\star_\eta) = 0$ and so:

$$g(x^\star_\eta) = -\eta c \tag{3.12}$$

Using this observation we obtain that

$$c^\top x^\star_\eta - c^\top x^\star = -\left\langle c, x^\star - x^\star_\eta \right\rangle = \frac{1}{\eta}\left\langle g(x^\star_\eta), x^\star - x^\star_\eta \right\rangle$$

To complete the proof it remains to argue that $\left\langle g(x^\star_\eta), x^\star - x^\star_\eta \right\rangle < m$. We will show even more, that for every two points x, y in the interior of P, we have $\langle g(x), y - x\rangle < m$. This follows by a simple calculation:

$$\begin{aligned}
\langle g(x), y - x\rangle &= \sum_{i=1}^m \frac{a_i^\top (y - x)}{s_i(x)} \\
&= \sum_{i=1}^m \frac{(b_i - a_i^\top x) - (b_i - a_i^\top y)}{s_i(x)} \\
&= \sum_{i=1}^m \frac{s_i(x) - s_i(y)}{s_i(x)} = m - \sum_{i=1}^m \frac{s_i(y)}{s_i(x)} < m
\end{aligned}$$

Where in the last inequality we make use of the fact that our points x, y are strictly feasible, i.e. $s_i(x), s_i(y) > 0$ for all i.

□

This lemma tells us that if want an ε-additive solution, we could stop our path following procedure when

$$\eta = \Omega\left(\frac{m}{\varepsilon}\right).$$

At this point one may wonder why instead of working in an iterative fashion we do not just set $\eta = {}^{m}/_{\varepsilon}$ and solve the perturbed linear problem to optimality. It turns out that it is hard to compute $x^\star_\eta$ when some arbitrary $\eta > 0$ is given (essentially, it is at least as hard as solving linear optimization problems). What we are able to do is given $x^\star_\eta$ for some $\eta > 0$ calculate $x^\star_{\eta'}$ for η' a little bit larger than η. This immediately rouses another question: then how do we find $x^\star_{\eta_0}$ at the very beginning of the algorithm? We will discuss this problem in detail later, for now let us assume that it is possible to provide some $\eta_0 > 0$ and the corresponding point $x_0 = x^\star_{\eta_0}$.

One step of our algorithm will essentially correspond to one step of Newton's method. Before we give a full description of the algorithm recall that $n(x)$ is the Newton step at point x with respect to some underlying function f. Whenever we write $n(x_k)$, we have the function $f_{\eta_k}(x) = \eta_k c^\top x + F(x)$ in mind. So

$$n(x_k) = -H(x_k)^{-1} \nabla f_{\eta_k}(x_k).$$

We now give the algorithm.

```
Primal Path Following IPM for Linear Programming:

  (1) Find an initial $\eta_0$ and $x_0$ with $\|n(x_0)\|_{x_0} \leq \frac{1}{2}$.
  (2) At iteration $k$ $(k = 0, 1, 2, \ldots)$:
        • compute $x_{k+1}$ according to the rule:
              $x_{k+1} \overset{\text{def}}{=} x_k + n(x_k)$
        • set $\eta_{k+1} \overset{\text{def}}{=} \eta_k \left(1 + \frac{1}{8\sqrt{m}}\right)$.
  (3) Stop when for the first time $K$, $\eta_K > \frac{m}{\varepsilon}$.
  (4) Calculate $\hat{x} \overset{\text{def}}{=} x^\star_{\eta_K}$ by Newton's method (starting at $x_K$), output
      $\hat{x}$.
```

In this algorithm we do not ensure that $x_k = x^\star_{\eta_k}$ at every step. This means that as opposed to what we have said before, our points do not lie on the central path. All we care about is that the points are *close enough to the central path.* By close enough, we mean $\|n(x_k)\|_{x_k} \leq {}^1/_2$. It may not be clear why this is the right notion of the distance from central path. We will discuss this in the Section 3.4. Let us conclude the description of the algorithm by the following remark.

Remark 3.5. At step 4. of the *Primal Path Following IPM* we apply Newton's method to obtain a point on the central path. At this moment it is not obvious that it will converge and, if yes, how quickly. We will see later that the point x_K is in the quadratic convergence region of Newton's method, so in fact we will need only a few iterations to reach $x^\star_{\eta_K}$. Instead of doing this final computation, one could also output simply x_K. It can be shown that the optimality guarantee at point x_K is $O(\varepsilon)$, see Appendix 3.6

We summarize what we prove about the algorithm above:

Theorem 3.6. The primal path following algorithm, given a linear program with m constraints and a precision parameter $\varepsilon > 0$, performs $O(\sqrt{m}\log {}^m/_{\eta_0 \cdot \varepsilon})$ iterations and outputs a point $\hat{x}$ satisfying:

$$c^\top \hat{x} \le c^\top x^\star + \varepsilon.$$

3.3.5 Analysis of the Algorithm

This section is devoted to the proof of Theorem 3.6. The statement does not provide any bounds on the cost of a single iteration. It is easy to see that the only expensive operation we perform at every iteration is computing the Newton step. This computation can be viewed as solving a linear system of the form $H(x)z = g(x)$, where z is the vector of variables. Of course, a trivial bound would be $O(n^3)$ or $O(n^\omega)$, but these may be significantly improved for specific problems. For example, computing a Newton step for an LP max-flow formulation can be done in $\widetilde{O}(|E|)$ time.

Recall that we initialize our algorithm with some $\eta_0 > 0$. Then, at every step we increase η by a factor of $(1 + \frac{1}{8\sqrt{m}})$, thus after $O(\sqrt{m}\log {}^m/_{\eta_0 \cdot \varepsilon})$ we reach $\Omega({}^m/_\varepsilon)$. This establishes the bound on the number of iterations. It remains to prove the correctness (the optimality guarantee).

The following lemma plays a crucial role in the proof of correctness. It asserts that the points we produce lie close to the central path:

Lemma 3.7. For every $k = 0, 1, \ldots, K$ it holds that $\|n(x_k)\|_{x_k} \le \frac{1}{2}$.

To prove this, we consider one iteration and show that, if started with $\|n(x_k)\|_{x_k} \le {}^1/_2$, then we will end up with $\|n(x_{k+1})\|_{x_{k+1}} \le {}^1/_2$. Every iteration consists of two steps: the Newton step w.r.t. f_{η_k} and the increase of η_k to η_{k+1}. We formulate another two lemmas explaining what happens when performing those steps.

Lemma 3.8. After taking one Newton step at point x w.r.t. the function f_η , the new point x' satisfies:

$$\|n(x')\|_{x'} \le \|n(x)\|_x^2.$$

Lemma 3.9. For every two positive $\eta, \eta' > 0$, we have:

$$\|H^{-1}(x)\nabla f_{\eta'}(x)\|_x \leq \frac{\eta'}{\eta}\|H^{-1}(x)\nabla f_\eta(x)\|_x + \sqrt{m}\left|\frac{\eta'}{\eta} - 1\right|.$$

It is easy to verify that the above two lemmas together imply that if $\|n(x_k)\|_{x_k} \leq \frac{1}{2}$, then

$$\|n(x_{k+1})\|_{x_{k+1}} \leq \frac{1}{4} + \frac{1}{8} + o(1) < \frac{1}{2}.$$

Hence, Lemma 3.7 follows by induction.

It remains to prove Lemmas 3.8 and 3.9. We start with the latter, since its proof is a bit simpler.

Proof. [Proof of Lemma 3.9] We have

$$\begin{aligned} H^{-1}(x)\nabla f_{\eta'}(x) &= H^{-1}(x)(\eta' c + g(x)) \\ &= \frac{\eta'}{\eta}H^{-1}(x)(\eta c + g(x)) + (1 - \frac{\eta'}{\eta})H^{-1}g(x) \\ &= \frac{\eta'}{\eta}H^{-1}(x)\nabla f_\eta(x) + \left(1 - \frac{\eta'}{\eta}\right)H^{-1}g(x). \end{aligned}$$

After taking norms and applying triangle inequality w.r.t. $\|\cdot\|_x$:

$$\|H^{-1}(x)\nabla f_{\eta'}(x)\|_x \leq \frac{\eta'}{\eta}\|H^{-1}(x)\nabla f_\eta(x)\|_x + \left|1 - \frac{\eta'}{\eta}\right| \|H^{-1}g(x)\|_x.$$

Let us stop here for a moment and try to understand what is the significance of the specific terms in the last expression. In our analysis of the algorithm, the term $\|H^{-1}(x)\nabla f_\eta(x)\|_x$ is a small constant. The goal is to bound the left hand side by a small constant as well. We should think of η' as $\eta(1+\delta)$ for some small $\delta > 0$. In such a setting $\frac{\eta'}{\eta}\|H^{-1}(x)\nabla f_\eta(x)\|_x$ will be still a small constant, so what prevents us from choosing a large δ is the second term $(1 - \frac{\eta'}{\eta})\|H^{-1}g(x)\|_x$. We need to derive an upper bound on $\|H^{-1}g(x)\|_x$. We show that $\|H^{-1}g(x)\|_x \leq \sqrt{m}$. Let us denote $z \stackrel{\text{def}}{=} H^{-1}g(x)$. We get:

$$\|z\|_x^2 = z^\top g(x) = \sum_{i=1}^m \frac{z^\top a_i}{s_i(x)} \leq \sqrt{m}\sqrt{\sum_{i=1}^m \frac{(z^\top a_i)^2}{s_i(x)^2}}. \tag{3.13}$$

The last inequality follows from Cauchy-Schwarz. Further:

$$\sum_{i=1}^m \frac{(z^\top a_i)^2}{s_i(x)^2} = z^\top \left(\sum_{i=1}^m \frac{a_i a_i^\top}{s_i(x)^2}\right) z = z^\top H(x) z = \|z\|_x^2. \tag{3.14}$$

Putting (3.13) and (3.14) together we obtain that $\|z\|_x^2 \leq \sqrt{m}\|z\|_x$, so in fact $\|z\|_x \leq \sqrt{m}$.

□

Before we proceed with the proof of Lemma 3.8 let us remark that it can be seen as an assertion that x belongs to the quadratic convergence region of Newton's method.

Proof. [Proof of Lemma 3.8] We know that the point x' is the minimizer of the second order approximation of f_η at point x, hence:

$$\nabla f_\eta(x) + H(x)(x' - x) = 0.$$

which implies that:

$$\sum_{i=1}^m \frac{a_i}{s_i(x)} + \sum_{i=1}^m \frac{a_i a_i^\top}{s_i(x)^2}(x' - x) = -\eta c.$$

We use it to compute $\nabla f_\eta(x')$:

$$\begin{aligned}
\nabla f_\eta(x') &= \eta c + \sum_{i=1}^m \frac{a_i}{s_i(x')} \\
&= -\left(\sum_{i=1}^m \frac{a_i}{s_i(x)} + \sum_{i=1}^m \frac{a_i a_i^\top}{s_i(x)^2}(x' - x)\right) + \sum_{i=1}^m \frac{a_i}{s_i(x')} \\
&= \sum_{i=1}^m \left(\frac{a_i}{s_i(x')} - \frac{a_i}{s_i(x)} - \frac{a_i a_i^\top (x' - x)}{s_i(x)^2}\right) \\
&= \sum_{i=1}^m \left(\frac{a_i a_i^\top (x' - x)}{s_i(x) s_i(x')} - \frac{a_i a_i^\top (x' - x)}{s_i(x)^2}\right) \\
&= \sum_{i=1}^m \frac{a_i (a_i^\top (x' - x))^2}{s_i(x)^2 s_i(x')}.
\end{aligned}$$

Our goal is to show that $\|n(x')\|_{x'} \leq \|n(x)\|_x^2$. Instead [8], we will prove that for every vector z, we have that $\langle z, n(x')\rangle_{x'} \leq \|z\|_{x'} \|n(x)\|_x^2$.
Indeed:

$$\begin{aligned}
\langle z, n(x')\rangle_{x'} &= z^\top \nabla f_\eta(x') = \sum_{i=1}^m \frac{z^\top a_i (a_i^\top (x' - x))^2}{s_i(x)^2 s_i(x')} \\
&\overset{\text{Cauchy}-\text{Schwarz}}{\leq} \left(\sum_{i=1}^m \frac{(z^\top a_i)^2}{s_i(x')^2}\right)^{1/2} \cdot \left(\sum_{i=1}^m \frac{(a_i^\top (x' - x))^4}{s_i(x)^4}\right)^{1/2} \\
&\leq \|z\|_{x'} \cdot \left(\sum_{i=1}^m \frac{(a_i^\top (x' - x))^2}{s_i(x)^2}\right) = \|z\|_{x'} \|n(x)\|_x^2
\end{aligned}$$

which completes the proof.

□

[8] The following fact is true for every Hilbert space $\mathcal{H}$: the norm of an element $u \in \mathcal{H}$ is given by the formula $\|u\| = \max\{\frac{\langle z, u\rangle}{\|z\|} : z \in \mathcal{H} \setminus \{0\}\}$. We work with $\mathcal{H} = \mathbb{R}^n$ but with nonstandard inner product: $\langle u, v\rangle_{x'} = u^\top H(x') v$

3.3.6 The Starting Point

In this section we give a method for finding a valid starting point. More precisely, we show how to find efficiently some $\eta_0 > 0$ and x_0 such that $\|n(x_0)\|_{x_0} \leq 1/2$.

Before we start, we would like to remark that this discussion provides a very small η_0, of order 2^{-L}. This enables us to prove that in fact IPM can solve linear programming in polynomial time, but does not seem promising when trying to apply IPM to devise fast algorithms for combinatorial problems. Indeed, there is a factor of $\log \eta_0^{-1}$ in the bound on number of iterations, which translates to L for such a tiny η_0. To make an algorithm fast, we need to have $\eta_0 = \Omega(1/\mathrm{poly(m)})$. However, it turns out that for specific problems (such as maximum flow) we can often devise some specialized methods for finding satisfying η_0 and x_0 and thus solve this issue.

First, we will show how given a point $x' \in \mathrm{int}(P)$ we can find some starting pair (η_0, x_0). Then finally we show how to obtain such point $x' \in \mathrm{int}(P)$. Let us assume now that such a point is given. Furthermore, we assume that each of its coordinates is written using $O(L)$ bits and each constraint is satisfied with slack at least 2^{-L}, that is $b_i - a_i^\top x' \geq 2^{-L}$. Our procedure for finding x' will provide such an x' based on our assumption that P is full dimensional. [9]

Recall that we want to find a point x_0 close to the central path $\Gamma_c = \{x_\eta^\star : \eta \geq 0\}$, which corresponds to the objective function $c^\top x$. Note that as $\eta \to 0$, $x_\eta^\star \to x_0^\star = x_c$, the analytic center of P. So finding a point x_0, very close to the analytic center and choosing η_0 to be some very tiny number should be a good strategy. In fact it is, but how to find a point close to x_c?

The central path Γ_c is of main interest for us, because it tends to the optimal solution to our linear program. In general, if $d \in \mathbb{R}^n$ we may define Γ_d to be the path consisting of minimizers to the functions $\nu d^\top x + F(x)$ for $\nu \geq 0$. What do all the paths Γ_d have in common? The origin! They all start at the same point: analytic center of P. Our strategy will be to pick one such path on which x' lies and traverse it backwards to reach a point very close to the origin of this path, which at the same time will be a good choice for x_0.

Recall that g is the gradient of the logarithimic barrier F and define $d = -g(x')$. Now it turns out that $x' \in \Gamma_d$. Why is this? Denote $f'_\nu(x) = d^\top x + F(x)$ and let $x'^\star_\nu$ be the minimizer of f'_ν. Then $x'^\star_1 = x'$, since $\nabla f'_1(x') = 0$. We will use $n'(x)$ for the Newton step at x with respect to f'_ν. We see in particular that $n'(x') = 0$.

As mentioned above, our goal is to move along the Γ_d path in the direction of decreasing ν. We will use exactly the same method as in the *Primal Path Following*. We perform steps, in each of them we make one Newton step w.r.t. current ν and then decrease ν by a factor of $(1 - 1/8\sqrt{m})$. At each step it holds that $\|n'(x)\|_x \leq 1/2$, by an argument identical to the proof of Lemma 3.7. It remains to see, how small ν we need to have (how many iterations we need to perform).

Lemma 3.10. If $\|H(x)^{-1}g(x)\|_x \leq 1/4$ and we take $\eta_0 = (4\|H(x)^{-1}c\|_x)^{-1}$, then

[9] In this presentation we will not discuss some details, e.g. the full-dimensionality assumption. These are easy to deal with and can be found in any book on linear programming.

the point x is a good candidate for x_0. The Newton step $n(x)$ w.r.t. f_{η_0} satisfies:

$$\|n(x)\|_x \le \frac{1}{2}.$$

Proof. This is just a simple calculation:

$$\|n(x)\|_x = \|H^{-1}(x)(\eta_0 c + g(x))\|_x \le \eta_0\|H(x)^{-1}c\|_x + \|H(x)^{-1}g(x)\|_x \le \frac{1}{4} + \frac{1}{4}.$$

□

Suppose now we have some very small $\nu > 0$ and a point x such that $\|n'(x)\|_x \le 1/2$. By performing two further Newton steps, we may assume $\|n'(x)\|_x \le 1/16$. We have:

$$n'(x) = H(x)^{-1}(-\nu g(x') + g(x)),$$

hence:

$$\|H(x)^{-1}g(x)\|_x \le \nu\|H(x)^{-1}g(x')\|_x + \|n'(x)\|_x \le \nu\|H(x)^{-1}g(x')\|_x + \frac{1}{16}$$

So it turns out, by Lemma 3.10 that it is enough to make $\nu\|H(x)^{-1}g(x')\|_x$ smaller then a constant. We can make ν as small as we want, but what with $\|H(x)^{-1}g(x')\|_x$? Let us present a technical claim, which we leave without proof:

Claim 3.11. All the calculations during the backward walk along Γ_d can be performed with $2^{O(L)}$ precision.

This means that we may assume that all the points x computed during the procedure have only $O(L)$ places after the comma, and are of absolute value at most $2^{O(L)}$. This allows us to provide an upper bound on $\|H(x)^{-1}g(x')\|_x$.

Claim 3.12. If the description sizes of x' and x are $O(L)$, then $\|H(x)^{-1}g(x')\|_x \le 2^{\mathrm{poly(L)}}$.

This follows from the fact that a solution to a linear system is of polynomial size. Now it remains to take ν so small that $\nu\|H(x)^{-1}g(x')\|_x \le 1/8$. By our previous reasoning this is enough to get suitable (η_0, x_0). To reach such a ν we need to perform $\widetilde{O}(\sqrt{m}\log 1/\nu)$ iterations, but $\log 1/\nu$ is polynomial in L, so we are done.

To complete our considerations we show how to find a suitable $x' \in \mathrm{int}(P)$. To this end, we consider an auxiliary linear program:

$$\begin{aligned} &\min_{(t,x)\in\mathbb{R}^{n+1}} t \\ &a_i^\top x \le b_i + t, \text{ for all } i \end{aligned} \tag{3.15}$$

Taking $x = 0$ and t big enough ($t = 1 + \sum_i |b_i|$), we get a strictly feasible solution. Having such, we may solve 3.15 up to an additive error of $2^{-O(L)}$ in polynomial time (by IPM). This will give us a solution (t', x') with $t' = -2^{-\Omega(L)}$, so $x' \in \mathrm{int}(P)$ is a suitable starting point for our walk along Γ_d.

3.4 Self-Concordant Barrier Functions

Finally, in this section we present the definition of a *self-concordant barrier function* for a convex set P. Armed with this definition, the task of bounding the number of iterations of the path following method for minimizing a linear function over P reduces to analyzing certain structural properties of the self-concordant barrier function. We omit the straightforward details about how, once we have a self-concordant barrier function for a convex set, one can design a path following IPM for it.

Definition 3.13. Let $F : \mathrm{int}(P) \mapsto \mathbb{R}$ be a $\mathcal{C}^3$ function. We say that F is self-concordant with parameter ν if:

(1) F is a barrier function: $F(x) \to \infty$ as $x \to \partial P$.
(2) F is strictly convex.
(3) For all $x \in \mathrm{int}(P)$, $\nabla^2 F(x) \succeq \frac{1}{\nu} \nabla F(x) \nabla F(x)^\top$.
(4) For all $x \in \mathrm{int}(P)$, $\left|\nabla^3 F(x)[h,h,h]\right| \le 2\|h\|_x^3 = 2\left|\nabla^2 F(x)[h,h]\right|^{3/2}$.[10]

With this definition in hand, it is not difficult to give an IPM for P which takes about $\sqrt{\nu}\log {}^1\!/\!_\varepsilon$ iterations. In fact, a careful reader would already have observed that Lemmas 3.9 and 3.8 prove exactly (3) and (4) for the log-barrier function with the complexity parameter m.

Similar to the log-barrier function for the positive real-orthant, for semidefinite programs over the cone of positive-definite matrices (or an affine transformation of it), one can use the following barrier function:

$$F(X) = -\log\det X$$

for $X \succ 0$.

In the next lecture we will discuss self-concordant barrier functions with complexity parameters $\ll m$. The main object of study will be the *volumetric barrier* of Vaidya.

3.5 Appendix: The Dikin Ellipsoid

Recall our setting:

$$\begin{aligned} \min\ & \langle c, x\rangle \\ \text{s.t.}\quad & Ax \le b \end{aligned}$$

where $Ax \le b$ defines a bounded, nonempty and full-dimensional polytope P. Recall that we can write the constrains at $\langle a_i, x\rangle \le b_i$ and then consider the *slacks* $s_i(x) = b_i - \langle a_i, x\rangle$ which capture the extent to which a constraint is satisfied. Clearly, if

[10] Here we interpret $\nabla^3 F(x)[h,h,h]$ as the inner product between the 3-tensor $\nabla^3 F(x)$ and $h\otimes h\otimes h$, namely, $\langle \nabla^3 F(x), h\otimes h\otimes h\rangle)$ and $\nabla^2 F(x)[h,h] = h^\top \nabla^2 F(x) h = \langle \nabla^2 F(x), h\otimes h\rangle$.

x is a feasible point, then $s_i(x) \geq 0$ for all i. Note that, when clear from context, we will often write s_i for $s_i(x)$ for conciseness. The log-barrier function (rather its negative) and its differentials are:

$$F(x) = \sum_i \log(s_i(x))$$

where

$$\begin{aligned} \nabla F(x) &= \frac{a_i}{s_i}, \qquad \text{and} \\ \nabla^2 F(x) &= \sum_i \frac{a_i a_i^\top}{s_i^2}. \end{aligned}$$

We often write $H(x) = \nabla^2 F(x)$ for the *Hessian* of x, and note that $H(x) \succeq 0$ (and in fact $H(x) \succ 0$ when A is fully-dimensional).

Now, we can consider the ellipsoid centred at x and defined by H, namely

$$\mathcal{E}_x(H(x)) = \{y : (y-x)^\top H(x)(y-x) \leq 1\}.$$

This is the *Dikin ellipsoid* centred at x.

Definition 3.14. The *Dikin Ellipsoid* at $x \in \text{int}(P)$ is defined as the ball of radius 1 around x in the local norm defined by $H(x)$ at x.

We will show that the Dikin ellipsoid is always completely contained in the polytope P. Furthermore, if we denote by x_c the *Analytic Center* of P, i.e. the unique minimizer of $F(x)$, then the Dikin ellipsoid at x_c inflated m times contains the whole polytope P (we will also show that $\sqrt{m}$ blow-up is sufficient for symmetric polytopes). We start by proving the first claim.

Theorem 3.15. For every $x \in \text{int}(P)$, the Dikin ellipsoid at x is fully contained in P.[11]

Proof. This is easy to see since, for any $y \in \mathcal{E}_x$, all slacks are non-negative, and hence $y \in P$. Formally, let $y \in \mathcal{E}_x(H(x))$. Then,

$$\begin{aligned} (y-x)^\top H(x)(y-x) &\leq 1 \\ \Rightarrow \sum_i \frac{\langle a_i, y-x\rangle^2}{s_i^2} &\leq 1 \\ \Rightarrow \frac{\langle a_i, y-x\rangle^2}{s_i^2} &\leq 1 \quad \forall i \text{ since all summands are non-negative} \\ \Rightarrow \left(\frac{s_i(x)-s_i(y)}{s_i(x)}\right)^2 &\leq 1 \quad \forall i \\ \Rightarrow \left|1 - \frac{s_i(y)}{s_i(x)}\right| &\leq 1 \quad \forall i. \end{aligned}$$

[11] In fact, this also holds when P is not bounded for certain cases such as if P is the positive orthant.

Hence, for all i we know that $0 \leq {}^{s_i(y)}/_{s_i(x)} \leq 2$, and, in particular, $s_i(y) \geq 0 \quad \forall i$. □

Let us start with a very interesting lemma describing one crucial property of the analytic center of P:

Lemma 3.16. If x_c is the analytic center of P then for every point $x \in \mathrm{int}(P)$ it holds that:

$$\sum_{i=1}^{m} \frac{s_i(x)}{s_i(x_c)} = m.$$

Proof. Let us denote $r(x) = \sum_{i=1}^{m} \frac{s_i(x)}{s_i(x_c)}$. We know that $r(x_c) = m$. To show that a function is constant, it remains to find its derivative and argue that it is zero:

$$\nabla r(x) = \nabla \left(\sum_{i=1}^{m} \frac{s_i(x)}{s_i(x_c)} \right) = \frac{a_i}{s_i(x_c)} = \nabla F(x_c)$$

but x_c is the minimizer of $F(x)$ so the gradient at this point vanishes. □

Now we are ready to proof the second theorem describing the Dikin ellipsoid. This time we look at the Dikin ellipsoid centered at a specific point x_c:

Theorem 3.17. The Dikin ellipsoid at x_c inflated m times contains P inside: $P \subseteq m\mathcal{E}_{x_c}(H(x_c))$.

Proof. Take any point $x \in P$, the goal is to show that $(x-x_c)^\top H(x_c)(x-x_c) \leq m^2$. We compute:

$$\begin{aligned}
&(x - x_c)^\top H(x_c)(x - x_c) \\
=&(x - x_c)^\top \left(\sum_{i=1}^{m} \frac{a_i a_i^\top}{s_i(x_c)^2} \right) (x - x_c) \\
=&\sum_{i=1}^{m} \frac{(a_i^\top (x - x_c))^2}{s_i(x_c)^2} = \sum_{i=1}^{m} \frac{(s_i(x_c) - s_i(x))^2}{s_i(x_c)^2} \\
=&\sum_{i=1}^{m} \frac{s_i(x)^2}{s_i(x_c)^2} + \sum_{i=1}^{m} \frac{s_i(x_c)^2}{s_i(x_c)^2} - 2\sum_{i=1}^{m} \frac{s_i(x_c)s_i(x)}{s_i(x_c)^2}.
\end{aligned}$$

Now, the middle term $\sum_{i=1}^{m} \frac{s_i(x_c)^2}{s_i(x_c)^2}$ is equal to m and $\sum_{i=1}^{m} \frac{s_i(x_c)s_i(x)}{s_i(x_c)^2} = \sum_{i=1}^{m} \frac{s_i(x)}{s_i(x_c)} = m$ by Lemma 3.16. So we obtain:

$$(x - x_c)^\top H(x_c)(x - x_c) = \sum_{i=1}^{m} \frac{s_i(x)^2}{s_i(x_c)^2} - m.$$

Observe that all the terms in the sum $\sum_{i=1}^{m} \frac{s_i(x)^2}{s_i(x_c)^2}$ are nonnegative, so we can use the simple bound $\sum_{i=1}^{m} \frac{s_i(x)^2}{s_i(x_c)^2} \leq \left(\sum_{i=1}^{m} \frac{s_i(x)}{s_i(x_c)}\right)^2 = m^2$ (the last equality again by Lemma 3.16) and obtain finally:

$$(x - x_c)^\top H(x_c)(x - x_c) \leq m^2 - m \leq m^2.$$

□

The last theorem we want to proof about the Dikin ellipsoid asserts that when we restrict ourselves to symmetric polytopes (i.e. symmetric with respect to the center of P) then the Dikin ellipsoid at the center inflated only $\sqrt{m}$ times contains the polytope.

Theorem 3.18. Let P be a symmetric polytope, then $P \subseteq \sqrt{m}\mathcal{E}_{x_c}(H(x_c))$.

Remark 3.19. Note that the Dikin ellipsoid depends not exactly on P but rather on the description of P (on the set of constraints). In the above theorem we assume that the description of P is reasonable. Since P is symmetric (with respect to the origin), we can assume that for every constraint $x^\top a_i \leq b_i$ there is a corresponding constraint $-x^\top a_i \leq b_i$.

Proof. [Proof of Theorem 3.18] We assume that P is symmetric w.r.t. the origin, so $x_c = 0$. Further, since all the b_i's are nonzero, we may rescale the constraints and assume $b_i = 1$ for all i. After this reductions, our ellipsoid is as follows: $\mathcal{E} = \{x : \sum_{i=1}^{m}(x^\top a_i)^2 \leq 1\}$.

Take any point x on the boundary of $\mathcal{E}$, that is $\sum_{i=1}^{m}(x^\top a_i)^2 = 1$. We need to show that $\sqrt{m}x \notin \text{int}(P)$. This will finish the proof.

Since $\sum_{i=1}^{m}(x^\top a_i)^2 = 1$, there exists i with $|x^\top a_i| \geq \frac{1}{\sqrt{m}}$. The set of constraints is symmetric, so we can assume that $x^\top a_i \geq \frac{1}{\sqrt{m}}$. Hence $(\sqrt{m}x)^\top a_i \geq 1$, so $\sqrt{m}x \notin \text{int}(P)$. □

Theorems 3.17 and 3.18 together with Theorem 3.15 demostrate that the Dikin ellipsoid centered at x_c is a very good approximation of the polytope P. We obtain an ellipsoid with a blow-up ratio of m (that is $\mathcal{E}_{x_c}(H(x_c)) \subseteq P \subseteq m\mathcal{E}_{x_c}(H(x_c))$) or $\sqrt{m}$ for the symmetric case. One can ask if this is the best we can do. It turns out that we can obtain better ratio by taking the so called John ellipsoid. It is defined as the largest-volume ellipsoid contained in the polytope P. When we inflate it by a factor of n, it contains P inside (similarly, $\sqrt{n}$ is enough for symmetric polytopes). This means that the John Ellipsoid achieves better blow-up ratio, because $n \leq m$ (we need at least n linear inequalities to define a non-degenerate, bounded polytope in n-dimensional space). One can also prove that this is indeed tight, the best possible blow-up ratio for the n-dimensional simplex is exactly n.

3.5.1 The Dikin Algorithm and some Intuition for $\sqrt{m}$ Iterations

The results above imply an particularly simple greedy IPM: start with the analytic centre x_c of a polytope P and move to the boundary of $\mathcal{E}_{x_c}(H(x_c))$ in the direction of $-c$, where c is the cost vector. Repeat the same from the next point. First note that algorithmically this makes sense as Theorem 3.15 implies that throughout the execution of the algorithm, we are always inside P. The cost is also decreasing. However, we can only argue that the cost becomes $1/\sqrt{m}$ in the symmetric case (or $1/m$ in the non-symmetric case) of the optimal cost in the *first* iteration. To see this assume that P is symmetric. Thus, $x_c = 0$ and the cost of this point is 0. Let x be the point on the boundary of $\mathcal{E}_{x_c}(H(x_c))$ in the direction of $-c$. We know from Theorem 3.15 that $x \in P$. However, we also know from Theorem 3.18 that $\langle c, \sqrt{m}x\rangle \leq \langle c, x^\star\rangle = \text{opt}$, where $x^\star$ is the optimal solution to the linear program. This is because $\sqrt{m}x$ lies on the boundary of $\sqrt{m}\mathcal{E}_{x_c}$ which contains P, and is in the direction of $-c$. Thus, $\langle c, x\rangle \leq \frac{1}{\sqrt{m}}\text{opt}$. If this were to magically continue at every step then one would expect the cost to come around opt in about $\sqrt{m}$ iterations. However, we cannot prove this and this analogy ends here.

3.6 Appendix: The Length of Newton Step

In this section we explain the relation between the length of the Newton step at point x (with respect to the function f_η) and the distance to the optimum $x^\star_\eta$. We show that whenever $\|n(x)\|_x$ is sufficiently small, $\|x - x^\star_\eta\|_x$ is small as well. This together with a certain strenghtening of Lemma 3.4 imply that in the last step of *Primal Path Following IPM* we do not need to go with x_K to optimality. In fact, only 2 additional Newton steps bring us (2ε)-close to the optimum.

Let us start by a generalization of Lemma 3.4, which shows that to get a decent approximation of the optimum we do not neccessarily need to be on the central path, but only close enough to it.

Lemma 3.20. For every point $x \in \text{int}(P)$ and every $\eta > 0$, if $\|x - x^\star_\eta\|_x < 1$ then:

$$c^\top x - c^\top x^\star \leq \frac{m}{\eta}(1 - \|x - x^\star_\eta\|_x)^{-1}.$$

Proof. For every $y \in \text{int}(P)$ we have:

$$\begin{aligned} c^\top x - c^\top y &= c^\top(x-y) \\ &= \langle c, x-y\rangle \\ &= \langle c_x, x-y\rangle_x \\ &\leq \|c_x\|_x \|x-y\|_x \end{aligned}$$

Where $c_x = H^{-1}(x)$ and the last inequality follows from Cauchy-Schwarz. Now, we want to bound $\|c_x\|_x$. Imagine we are at point x and we move in the direction of $-c_x$

until hitting the boundary of Dikin's ellipsoid. We will land in the point $x - \frac{c_x}{\|c_x\|_x}$, which is still inside P by Theorem 3.15. Therefore:

$$\left\langle c, x - \frac{c_x}{\|c_x\|_x} \right\rangle \geq \langle c, x^\star \rangle.$$

Since $\langle c, c_x \rangle = \|c_x\|_x^2$, we get:

$$\|c_x\|_x \leq \langle c, x \rangle - \langle c, x^\star \rangle.$$

We have obtained:

$$c^\top x - c^\top y \leq \|x - y\|_x (c^\top x - c^\top x^\star). \tag{3.16}$$

Now, let us express

$$c^\top x - c^\top x^\star = (c^\top x - c^\top x^\star_\eta) + (c^\top x^\star_\eta - c^\top x^\star)$$

and use 3.16 with $y = x^\star_\eta$. We obtain

$$c^\top x - c^\top x^\star \leq (c^\top x - c^\top x^\star)\|x - y\|_x + (c^\top x^\star_\eta - c^\top x^\star).$$

Thus

$$(c^\top x - c^\top x^\star)(1 - \|x - y\|_x) \leq c^\top x^\star_\eta - c^\top x^\star.$$

By applying Lemma 3.4 we get the result. □

Note that in the algorithm we never literally mention the condition that $\|x - x^\star_\eta\|_x$ is small. However, we will show that it follows from $\|n(x)\|_x$ being small. We will need the following simple fact about logarithmic barrier. A proof appears in the notes on Volumetric Barrier in the next lecture.

Fact 3.21. If $x, z \in \text{int}(P)$ are close to each other, i.e. $\|z - x\|_z \leq \frac{1}{4}$ then $H(x)$ and $H(z)$ are close as well:

$$\frac{4}{9} H(x) \preceq H(z) \preceq 4H(x).$$

We are ready to prove the main theorem of this section.

Theorem 3.22. Let x be any point in $\text{int}(P)$ and $\eta > 0$. Consider the Newton step $n(x)$ at point x with respect to f_η. If $\|n(x)\|_x \leq \frac{1}{18}$, then $\|x - x^\star_\eta\|_x \leq 5\|n(x)\|_x$.

Proof. Pick any h such that $\|h\|_x \leq \frac{1}{4}$. Expand $f_\eta(x+h)$ into a Taylor series around x:

$$f_\eta(x + h) = f_\eta(x) + h^\top \nabla f_\eta(x) + \frac{1}{2} h^\top \nabla^2 f_\eta(\theta) h \tag{3.17}$$

for some point θ lying on the segment $[x, x+h]$. We proceed by lower-bounding the linear term. Note that:

$$\left|h^\top \nabla f_\eta(x)\right| = |\langle h, n(x)\rangle_x| \leq \|h\|_x \|n(x)\|_x \tag{3.18}$$

by Cauchy-Schwarz. Next:

$$h^\top \nabla^2 f_\eta(\theta) h = h^\top H(\theta) h \geq \frac{4}{9} h^\top H(x) h \tag{3.19}$$

where we used Fact 3.21. Applying bounds 3.18, 3.19 to the expansion 3.17 results in:

$$f_\eta(x+h) \geq f_\eta(x) - \|h\|_x \|n(x)\|_x + \frac{2}{9}\|h\|_x^2. \tag{3.20}$$

Set $r = \frac{9}{2}\|n(x)\|_x$ and consider points y satysfing $\|x-y\|_x = r$, i.e. points on the boundary of the local norm ball of radius r centered at x. Then 3.20 simplifies to:

$$f_\eta(x+h) \geq f_\eta(x)$$

which implies that $x_\eta^\star$, the unique minimizer of f_η, belongs to the above mentioned ball and the theorem follows.

□

Combining Lemma 3.20 with Theorem 3.22 we get that if $\eta \geq \frac{m}{\varepsilon}$ and $\|n(x)\|_x \leq \frac{1}{18}$ then $c^\top x - c^\top x^\star < 2\varepsilon$.

4

Volumetric Barrier, Universal Barrier and John Ellipsoids

In this lecture we take a step towards improving the $\sqrt{m}$ in the number of iterations the path-following IPM presented in the previous lecture to $\sqrt{n}$. This will be achieved by considering more involved barrier functions rather than altering the basic framework of IPMs. The log-barrier function was particularly nice as working with it was computationally not harder than solving linear systems. However, it had many shortcomings. An obvious one is that it has no way to take into account that a constraint could be repeated many time. One way to handle this would be to work with *weighted* barrier functions which have the following form: $\sum_i w_i(x) \log s_i(x)$ where we could allow the weights to depend on the current point as well. Analyzing such methods, however, poses great technical challeges as now the gradients and Hessians become more unwieldy. Amazingly, Vaidya laid the foundations of analyzing such weighted barrier functions. He introduced one such function, the *volumetric barrrier* and showed how it allows us to improve the $\sqrt{m}$ to $(mn)^{1/4}$. Computationally, the volumetric barrier was no harder than computing determinants.

Subsequently, Nesterov-Nemirovskii discovered the *Universal Barrier* which achieves the stated goal of $\sqrt{n}$ iterations. Their result held far beyond the setting of LPs and worked for almost any convex body. However, computationally the barrier is at least hard as solving the convex program itself!

The search for a $\sqrt{n}$ barrier which is computationally efficient got a major boost by a recent result of Lee-Sidford who demonstrated a computationally efficient barrier (about the same complexity as solving linear systems) which achieves $\tilde{O}(\sqrt{n})$ iterations. In my opinion, their result is inspired by Vaidya's work: while Vaidya considered the volume of the Dikin ellipsoid as a barrier, Lee-Sidford consider an object which can be considered a *smoothening* of John ellipsoid. It remains an outstanding problem to remove the log-factors from the work of Lee-Sidford.[1]

We begin by presenting yet another proof of $\sqrt{m}$ convergence and then show how the volumetric barrier allows us to improve this to $(mn)^{1/4}$. Towards the end, we give a brief presentation (without proofs) of the $\sqrt{\text{rank}}$ Universal Barrier by Nesterov-Nemirovskii for *any* convex set. As for the work of Lee-Sidford, currently, explaining it simply remains a challenge. However, in the appendix we introduce the John ellipsoid and give some basic intuition about why one may expect the $\sqrt{m}$ to be possibly replaced by $\sqrt{n}$.

4.1 Restating the Interior Point Method for LPs

We start by restating the interior point method in a slightly different but equivalent way.

Theorem 4.1. Let F be a barrier function such that there exists δ and θ as follows:

(1) For all $x, z \in \text{int}(P)$ such that $\|x - z\|^2_{\nabla^2 F(z)} \leq \delta$, we have $\nabla^2 F(z) \overset{O(1)}{\approx} \nabla^2 F(x)$,[2] and
(2) $\|\nabla F(x)\|_{(\nabla^2 F(x))^{-1}} \leq \theta$.[3]

Then, there is an interior point method that produces an ε-approximate solution to LPs of the form $\min c^\top x$, such that $Ax \leq b$ in $O(\theta/\delta \log 1/\varepsilon)$ iterations. Here m is the number of rows of A.

Note that the first property captures the fact that, if two points x and z are close in the local norm, then the Hessian does not change by much. These two requirements along with differentiability are equivalent to self-concordance with complexity parameter θ/ε. We leave the proof to the reader.

4.1.1 The Logarithmic Barrier Revisited

Before we proceed to the volumetric barrier, as an exercise, we revisit the log-barrier funtion and derive the $O(\sqrt{m})$-iteration convergence result for it, yet again. Recall that the log-barrier function is:

$$F(x) \overset{\text{def}}{=} -\sum_{i=1}^{m} \log(b_i - a_i^\top x).$$

We will show that for this function $\theta = \sqrt{m}$ and $\delta = O(1)$. Hence, via Theorem 4.1, it follows that this results in an IPM which converges in roughly $\sqrt{m}$ iterations.

[1] This lecture appears as Lecture 4 in Vishnoi [Vis14].
[2] Formally, this means that there exist constants c_1 and c_2 such that $c_1 \nabla^2 F(z) \preceq \nabla^2 F(x) \preceq c_2 \nabla^2 F(z)$.
[3] Equivalently, we could write $(\nabla F(x))(\nabla F(x))^{-1} \preceq \theta \nabla^2 F(x)$.

4.1.1.1 θ for log-barrier

Let S be the diagonal matrix where $S_{ii} = s_i(x)$. We can now restate the gradient and Hessian of F with more convenient notation as follows:

$$\begin{aligned} \nabla F(x) &= A^\top S^{-1} 1, \text{ and} \\ \nabla^2 F(x) &= A^\top S^{-2} A. \end{aligned}$$

Hence,

$$\begin{aligned} \|\nabla F(x)\|^2_{(\nabla^2 F(x))^{-1})} &= (\nabla F(x))^\top (\nabla^2 F(x))^{-1} (\nabla F(x)) \\ &= (A^\top S^{-1} 1)^\top (A^\top S^{-2} A)^{-1} (A^\top S^{-1} 1) \\ &= 1^\top (S^{-1} A (A^\top S^{-2} A)^{-1} A^\top S^{-1}) 1. \end{aligned}$$

Denote

$$\Pi \stackrel{\text{def}}{=} S^{-1} A (A^\top S^{-2} A)^{-1} A^\top S^{-1},$$

and note that Π is a projection matrix; i.e., $\Pi^2 = \Pi$. Hence,

$$\begin{aligned} \|\nabla F(x)\|^2_{(\nabla^2 F(x))^{-1})} &= 1^\top \Pi 1 = 1^\top \Pi^2 1. \\ &= \|\Pi 1\|_2^2 \le \|1\|_2^2 \\ &= m. \end{aligned}$$

Thus, $\|\nabla F(x)\|_{(\nabla^2 F(x))^{-1})} \le \sqrt{m}$ as desired.

4.1.1.2 δ for the log-barrier

We will show that $\nabla^2 F(z) \overset{O(1)}{\approx} \nabla^2 F(x)$ for $\delta = 1/4$. Let x, z be such that $\|x - z\|_z \le \delta$. Hence,

$$\begin{aligned} (x-z)^\top (\nabla^2 F(z))(x-z) &= \sum_i \frac{\langle a_i, x-z \rangle^2}{s_i^2(z)} \\ &= \sum_i \frac{(a_i^\top (x-z))^2}{s_i^2(z)} \\ &= \sum_i \left(\frac{s_i(x) - s_i(z)}{s_i(z)} \right)^2. \end{aligned}$$

Therefore,

$$\left| \frac{s_i(x) - s_i(z)}{s_i(z)} \right| \le \sqrt{\delta} \quad \forall i.$$

In particular, for all i, $1 - \sqrt{\delta} \le s_i(x)/s_i(z) \le 1 + \sqrt{\delta}$, and hence, for $\delta = 1/4$,

$$\frac{s_i^2(z)}{s_i^2(x)} \le 4 \quad \text{and} \quad \frac{s_i^2(x)}{s_i^2(z)} \le \frac{9}{4}$$

for all i. Since $\min_i \left(\frac{s_i^2(z)}{s_i^2(x)}\right) \cdot \nabla^2 F(z) \preceq \nabla^2 F(x) \preceq \max_i \left(\frac{s_i^2(z)}{s_i^2(x)}\right) \cdot \nabla^2 F(z)$, we have that

$$\frac{4}{9}\nabla^2 F(z) \preceq \nabla^2 F(x) \preceq 4\nabla^2 F(z)$$

which gives $\nabla F(x) \overset{O(1)}{\approx} \nabla F(z)$ as desired.

4.2 Vaidya's Volumetric Barrier

Now, we go *beyond* $\sqrt{m}$; though we would like to get roughly $\sqrt{n}$ iterations, we will not quite reach this target, but will come somewhere in-between. The problem with the log-barrier function is that it weighs each constraint equally. Thus, if, for instance, a constraint is repeated many times, there is no way for the log-barrier function to know this. Vaidya introduced the *volumetric barrier* function which aims to bypass this limitation by assigning weights to each constraint:

$$-\sum_i w_i \log(b_i - a_i^\top x).$$

Of course, a useful choice of weights must depend on the current point x, and this causes several technical difficulties. His work laid the foundations for and developed many techniques to analyze such weighted barrier functions.

4.2.1 The Volumetric Barrier

The proposed barrier function by Vaidya uses the log-volume of the Dikin Ellipsoid.

Definition 4.2. The *volumetric barrier* is defined to be

$$V(x) \overset{\text{def}}{=} -\frac{1}{2} \cdot \log\det\left(\nabla^2 F(x)\right).$$

Note that, whenever we come up with a new barrier function, we must also worry about its computability along with the computability of its first and second order oracle. Using the restatement of the barrier function as the determinant of a positive definite matrix, we see that the volumetric barrier is efficiently computable.

Notation and basic properties. Notation will play an important role in understanding the rather technical sections that are to follow. Towards this, we introduce new notation, some of which may conflict with what we have used previously. Instead of $F(x), V(x), H(x)$, and so on, we denote the same quantities by F_x, V_x, and H_x. Similarly, the slack vector will be denoted by s_x and its components by $s_{x,i} = s_i(x)$. Let the i-th row of A be the vector a_i. For an $x \in \text{int}(P)$, let

$$A_x \overset{\text{def}}{=} S_x^{-1} A$$

where S_x is the diagonal matrix corresponding to the vector s_x. Then, the Hessian of the log-barrier function can be written as

$$H_x = A_x^\top A_x = A^\top S^{-2} A.$$

Thus, if $V_x = -\frac{1}{2} \log \det H_x$, then

$$\nabla V_x = A_x^\top \sigma_x \tag{4.1}$$

where

$$\sigma_{x,i} \overset{\text{def}}{=} \frac{a_i^\top H_x^{-1} a_i}{s_{x,i}^2}.$$

An informed reader could compare $\sigma_{x,i}$ with *leverage scores*, or *effective resistances* that arise when dealing with graph Laplacians. We note some simple properties of σ_x which will allow us to build towards finding the θ and δ necessary for applying Thoerem 4.1.

Fact 4.3.

(1) $\sigma_x \leq 1$.
(2) $\sigma_x^\top 1 = n$.

Proof. The first fact follows from the fact that $\frac{a_i^\top H_x^{-1} a_i}{s_{x,i}^2} \leq 1$ if and only if $\frac{a_i a_i^\top}{s_{x,i}^2} \preceq H_x$. But the latter is true since H_x is the sum over all i of quantites on the left hand side. For the second part note that

$$\begin{aligned} \sigma_x^\top 1 &= \sum_i \frac{a_i^\top H_x^{-1} a_i}{s_{x,i}^2} \\ &= \operatorname{Tr}\left(H_x^{-1} \sum_i \frac{a_i a_i^\top}{s_{x,i}^2} \right) \\ &= \operatorname{Tr}\left(H_x^{-1} H_x \right) \\ &= n. \end{aligned}$$

□

Thus, each $\sigma_{x,i}$ is at most one and they sum up to n. Thus, $\sigma_{x,i}$ assigns a relative importance to each constraint while maintaining a budget of n. This is unlike in the setting of log-barrier where each constraint has a weight of 1 and, hence, requires a budget of m. Further, note the following straightforward fact:

Fact 4.4. $\sigma_{x,i} = \sum_j \frac{(a_i^\top H_x^{-1} a_j)^2}{s_{x,i}^2 s_{x,j}^2}$.

Proof.

$$\begin{aligned} \sigma_{x,i} &= \frac{a_i^\top H_x^{-1} a_i}{s_{x,i}^2} \\ &= \frac{a_i^\top H_x^{-1} H_x H_x^{-1} a_i}{s_{x,i}^2} \\ &= \frac{a_i^\top H_x^{-1} \left(\sum_j \frac{a_j a_j^\top}{s_{x,j}^2} \right) H_x^{-1} a_i}{s_{x,i}^2} \\ &= \sum_j \frac{(a_i^\top H_x^{-1} a_j)^2}{s_{x,i}^2 s_{x,j}^2}. \end{aligned}$$

□

Let Σ_x denote the diagonal matrix corresponding to σ_x. Let

$$P_x \stackrel{\text{def}}{=} A_x H_x^{-1} A_x^\top = A_x (A_x^\top A_x)^{-1} A_x^\top.$$

Note that $P_x \succ 0$, is symmetric and is a projection matrix since

$$P_x^2 = A_x (A_x^\top A_x)^{-1} A_x^\top A_x (A_x^\top A_x)^{-1} A_x^\top = P_x.$$

Further, let $P_x^{(2)}$ denote the matrix whose each entry is the square of the corresponding entry of P_x. Then, Fact 4.4 above can be restated as

$$\sigma_x = P_x^{(2)} 1.$$

Thus, $P_x^{(2)} \geq 0$, $\Sigma_x - P_x^{(2)}$ is symmetric and $(\Sigma_x - P_x^{(2)})1 = 0$. Thus, it is a graph Laplacian. As a consequence we obtain the following fact.

Fact 4.5. $\Lambda_x \stackrel{\text{def}}{=} \Sigma_x - P_x^{(2)} \succeq 0$.

Gradient and the Hessian of the Volumetric Barrier. In the appendix we show the following by a straightforward differentiation of the volumetric barrier:

Lemma 4.6.

(1) $\nabla V_x = A_x^\top \sigma_x$ and
(2) $\nabla^2 V_x = A_x^\top (3\Sigma_x - 2P_x^{(2)}) A_x$.

Thus, using Fact 4.5, we obtain the following corollary which allows us to think of the Hessian of the volumetric barrier as the log-barrier weighted with leverage scores. This lemma greatly simplifies the calculations to come.

Corollary 4.7.
$$A_x^\top \Sigma_x A_x \preceq \nabla^2 V_x \preceq 5A_x^\top \Sigma_x A_x.$$

Proof. The lower bound is a direct consequence of Fact 4.5.

For the upper bound, note that for a graph Laplacian, it follows from the inequality $(a-b)^2 \leq 2(a^2+b^2)$ that
$$\Lambda_x \preceq 2(\Sigma_x + P_x^{(2)}).$$
Thus,
$$\nabla^2 V_x = A_x^\top (3\Sigma_x - 2P_x^{(2)})A_x = A_x^\top (\Sigma_x + 2\Lambda_x)A_x \preceq 5A_x^\top \Sigma_x A_x.$$
□

We now let $Q_x \stackrel{\text{def}}{=} A_x^\top \Sigma_x A_x$. The next lemma relates Q_x back the Hessian of the log-barrier function.

Lemma 4.8. $\frac{1}{4m}H_x \preceq Q_x \preceq H_x$.

This lemma is responsible for us losing the grounds we gained by the weighted barrier function. If we could somehow improve the lower bound to $\frac{1}{100}H_x$, then we would achieve our goal of $\sqrt{n}$. In the next section we show how to alter the barrier function to get a lower bound of $\frac{n}{m}H_x$ which will enable us to get the bound of $(mn)^{1/4}$.

Proof. The upper bound follows straightforwardly from Fact 4.3 which states that $\Sigma_x \preceq I$.

For the lower bound, we first break the sum into two parts as follows
$$Q_x = \sum_i \frac{\sigma_{x,i}a_ia_i^\top}{s_{x,i}^2} = \sum_{i:\sigma_{x,i}\geq 1/2m} \frac{\sigma_{x,i}a_ia_i^\top}{s_{x,i}^2} + \sum_{\sigma_{x,i}<1/2m} \frac{\sigma_{x,i}a_ia_i^\top}{s_{x,i}^2} \succeq \sum_{i:\sigma_{x,i}\geq 1/2m} \frac{\sigma_{x,i}a_ia_i^\top}{s_{x,i}^2}.$$
Thus,
$$Q_x \succeq \frac{1}{2m} \sum_{i:\sigma_{x,i}\geq 1/2m} \frac{a_ia_i^\top}{s_{x,i}^2} = \frac{1}{2m}\left(H_x - \sum_{i:\sigma_{x,i}<1/2m} \frac{a_ia_i^\top}{s_{x,i}^2}\right). \tag{4.2}$$
Now note that by the definition of $\sigma_{x,i}$, we obtain for all i,
$$\frac{a_ia_i^\top}{s_{x,i}^2} \preceq \sigma_{x,i}H_x.$$
Thus,
$$\sum_{i:\sigma_{x,i}<1/2m} \frac{a_ia_i^\top}{s_{x,i}^2} \preceq \sum_{i:\sigma_{x,i}<1/2m} \sigma_{x,i}H_x \preceq \frac{1}{2m}\sum_i H_x \preceq \frac{1}{2}H_x.$$

Thus, plugging in this estimate in (4.2), we obtain that

$$Q_x \succeq \frac{1}{2m}\left(H_x - \frac{1}{2}H_x\right) = \frac{1}{4m}H_x.$$

This completes the proof of the lemma. □

This allows us to get the following crucial bounds on σ_x.

Lemma 4.9. $\frac{a_i^\top Q_x^{-1} a_i}{s_{x,i}^2} \le \min\left\{\frac{1}{\sigma_{x,i}}, 4m\sigma_{x,i}\right\} \le 2\sqrt{m}.$

Proof. From the lower bound estimate of Lemma 4.8, it follows that

$$\frac{a_i^\top Q_x^{-1} a_i}{s_{x,i}^2} \le \frac{4m a_i^\top H_x^{-1} a_i}{s_{x,i}^2} = 4m\sigma_{x,i}.$$

The other inequality follows from the following simple inequality

$$\frac{\sigma_{x,i} a_i a_i^\top}{s_{x,i}^2} \preceq Q_x.$$

□

We can now conclude by stating the key properties of the volumetric barrier function necessary to get a desired bound on the number of iterations of the IPM:

Theorem 4.10.

(1) $\nabla V_x^\top (\nabla^2 V_x)^{-1} \nabla V_x \le n.$
(2) For all $x, y \in \mathrm{int}(P)$ such that $\|x - y\|_x \le \frac{1}{8m^{1/2}}$,

$$\nabla^2 V_x \overset{O(1)}{\approx} \nabla^2 V_y.$$

Proof. For the first part, note that

$$\begin{aligned}
\zeta^\top \nabla V_x \nabla V_x^\top \zeta &= \langle \zeta, A_x^\top \sigma_x \rangle^2 \\
&= \langle \zeta, A_x^\top \Sigma_x 1 \rangle^2 \\
&= \langle \Sigma_x^{1/2} A_x \zeta, \Sigma_x^{1/2} 1 \rangle^2 \\
&\le \left(\zeta^\top A_x^\top \Sigma_x A_x \zeta\right)\left(1^\top \Sigma_x 1\right) \\
&\le \left(\zeta^\top Q_x \zeta\right) n \\
&\le n\left(\zeta^\top \nabla^2 V_x \zeta\right).
\end{aligned}$$

For the second part, we start by noting that $\|x-y\|_x \leq \delta$ is the same as $(x-y)^\top Q_x(x-y) \leq \delta^2$. This in turn implies that

$$|a_i^\top (x-y)|^2 \leq \delta^2 a_i^\top Q_x^{-1} a_i \leq \frac{1}{4} \cdot s_{x,i}^2.$$

Thus, an argument similar to what we did for the log-barrier implies that

$$\left|1 - \frac{s_{y,i}}{s_{x,i}}\right| \leq \frac{1}{2}.$$

This implies that $H_x \approx H_y$ and $\sigma_{x,i} \approx \sigma_{y,i}$ for all i. Thus, $Q_x \approx Q_y$. This implies that $\nabla^2 V_x \approx \nabla^2 V_y$, completing the proof. □

4.2.2 The Hybrid Barrier

Now consider the barrier function

$$G_x = V_x + \frac{n}{m} F_x,$$

where V_x is the volumetric barrier function and F_x the log-barrier function.

$$\nabla^2 G_x = \nabla^2 V_x + \frac{n}{m} H_x.$$

This adding a scaled version of the log-barrier function changes nothing significantly in the previous section. However, crucially this addition implies, via Lemmas 4.7 and 4.8 that

$$\nabla^2 G_x \succeq \frac{n}{m} H_x$$

rather than $\frac{1}{4m} H_x$ in the case of volumetric barrier. This implies that the upper bound in Lemma 4.9 comes down to $\sqrt{m/n}$. This then implies that, while the first part of Theorem 4.10 continues to hold with $4n$ instead of n, the second part holds with $(mn)^{1/4}$ rather than $\sqrt{m}$. Hence, from Theorem 4.1, we obtain an $(mn)^{1/4}$ iteration IPM for LPs.

One can interpret Vaidya's technique as starting with a barrier F and replacing it with

$$-\frac{1}{2} \log\det \nabla^2 F(x) + \frac{n}{m} F(x).$$

One may ask if repeating this helps improve the bound of $(mn)^{1/4}$. In particular, it is an interesting question to understand the *fixed point* of this functional equation and to what extent is it self-concordant.

4.3 Appendix: The Universal Barrier

In this section we introduce Nesterov-Nemirovskii's result that every bounded and convex body K in n-dimensions admits an $O(n)$ -self concordant barrier function. This implies that the number of iterations scale like $\sqrt{n}$. This not only improves the self-concordance results for polytopes we have proved till now, but also significantly generalizes it to *all* convex bodies. However, as we will see, computationally it is not very attractive. Designing a barrier function that does not require much more computation than solving linear system of equations, at least for polytopes, remains an outstanding open problem.

What is their barrier function that works in such a general setting? It is a volumetric barrier; however, not of any ellipsoid, rather of the *polar* of the body K centered at the current point x. More precisely, for a point $x \in \text{int}(K)$, let

$$K_x^\circ \stackrel{\text{def}}{=} \{z : z^\top (y - x) \le 1 \;\; \forall y \in K\}.$$

Then, the Nesterov-Nemirovskii barrier function is defined to be

$$F(x) \stackrel{\text{def}}{=} \log \text{vol} K_x^\circ.$$

Since K is bounded, this is well-defined in the interior of K. Moreover, it is easy to see that as x approaches the boundary of K from the interior, the volume blows up to infinity. For instance, if K is the interval $[0, 1]$ then $K_x^\circ = \left[-\frac{1}{x}, 0\right] \cup \left[0, \frac{1}{1-x}\right]$. Thus, as $x \to 1$, the right end of the polar goes to $+\infty$ and as $x \to 0$, the left end of the polar goes to $-\infty$. In either case, the volume goes to infinity.

In fact this one-dimensional intuition goes a long way in understanding $F(x)$. This is due to the following simple observation which allows us to think of K_x° as a collection of line segments. Let $v \in \mathbb{S}^{n-1}$ be a unit vector and let $p(v)$ denote the maximum over $y \in K$ of $v^\top y$. Thus, we can rewrite the polar as a collection of the following *line segments*; each along a unit vector v.

$$K_x^\circ = \bigcup_{v \in \mathbb{S}^{n-1}} \left[0, \frac{1}{p(v) - v^\top x}\right] v.$$

This representation allows us to easily rewrite the volume of K_x° as

$$f(x) \stackrel{\text{def}}{=} \text{vol}\, K_x^\circ = \frac{1}{n} \int_{v \in \mathbb{S}^{n-1}} (p(v) - v^\top x)^{-n} dS(v)$$

where $dS(v)$ is the $(n-1)$-dimensional volume of the surface of the unit sphere around v.

What we achieve by this transformation is an understanding of the derivatives of $f(x)$ in terms of *moment integrals* over the polar. In fact the following claim can now be seen easily:

$$D^l f(x)[h, h, \ldots, h] = \frac{(-1)^l (n+l)!}{n!} I_l(h)$$

where

$$I_l(h) \stackrel{\text{def}}{=} \int_{K_x^\circ} (z^\top h)^l dz.$$

However, we are interested in the differentials of $F(x) = \log f(x)$. How do we calculate the differentials of this function? This is also easy since

$$\frac{d \log g(x)}{dx} = \frac{1}{g(x)} \frac{dg(x)}{dx}.$$

This allows us to easily derive the following expressions:

(1) $DF(x)[h] = -(n+1)\frac{I_1(h)}{I_0}$.
(2) $D^2F(x)[h,h] = (n+1)(n+2)\frac{I_2(h)}{I_0} - (n+1)^2\frac{I_1^2(h)}{I_0^2}$.
(3) $D^3F(x)[h,h,h] = -(n+3)(n+2)(n+1)\frac{I_3(h)}{I_0} + 3(n+2)(n+1)^2\frac{I_2(h)I_1(h)}{I_0^2} - 2(n+1)^3\frac{I_1(h)^3}{I_0^3}$.

It follows that $F(x)$ is convex since a straightforward calculation shows that

$$D^2F(x)[h,h] > 0.$$

Further, it is not difficult to see that

$$|DF(x)[h,h]| \leq \sqrt{n+1}\sqrt{|D^2F(x)[h,h]|},$$

which establishes the required bound on the complexity parameter. The self-concordance property remains to be proved; that does not seem to have a very illuminating proof and hence we omit the details here.

In summary, we sketched the proof of the fact that log-volume of the polar is a $\sqrt{n}$ self-concordant barrier function. Could this be improved beyond $\sqrt{n}$? The answer turns out to be no in general. This is as far as the theory of self-concordant barriers takes us. However, that is not to say that interior point methods cannot take us further, at least in the case of combinatorial polytopes!

4.4 Appendix: Calculating the Gradient and the Hessian of the Volumetric Barrier

In this section we give a sketch of the gradiant and Hessian calculation of the volumetric barrier. In particular, we prove the followin lemma mentioned earlier.

Lemma 4.11. (1) $\nabla V_x = A_x^\top \sigma_x$ and
(2) $\nabla^2 V_x = A_x^\top(3\Sigma_x - 2P_x^{(2)})A_x$.

Proof. Recall that $H_x = \sum_i \frac{a_i a_i^\top}{s_{x,i}^2}$ and $V_x = \frac{1}{2}\log\det H_x$. Let ζ be a vector and let $t \in \mathbb{R}$ be an infinitesimal. First note that

$$H_{x+t\zeta} - H_x = \sum_i \frac{a_i a_i^\top}{s_{x+t\zeta,i}^2} - \sum_i \frac{a_i a_i^\top}{s_{x,i}^2} = 2t\sum_i \frac{a_i a_i^\top}{s_{x,i}^2}\frac{a_i^\top \zeta}{s_{x,i}} + O(t^2).$$

Let $\Delta \overset{\text{def}}{=} 2t\sum_i \frac{a_i a_i^\top}{s_{x,i}^2}\frac{a_i^\top \zeta}{s_{x,i}}$. Thus, for an infinitesimal t,

$$\begin{aligned}
\frac{1}{2}\log\det H_{x+t\zeta} - \frac{1}{2}\log\det H_x &\approx \frac{1}{2}\log\det(H_x+\Delta) - \frac{1}{2}\log\det H_x \\
&= \frac{1}{2}\log\det H_x^{1/2}(I + H_x^{-1/2}\Delta H_x^{-1/2})H_x^{1/2} - \frac{1}{2}\log\det H_x \\
&= \frac{1}{2}\log\det(I + H_x^{-1/2}\Delta H_x^{-1/2}) \\
&\approx \frac{1}{2}\mathrm{Tr}(H_x^{-1/2}\Delta H_x^{-1/2}) \\
&= t\mathrm{Tr}\left(H_x^{-1/2}\sum_i \frac{a_i a_i^\top}{s_{x,i}^2}\frac{a_i^\top\zeta}{s_{x,i}}H_x^{-1/2}\right) \\
&= t\sum_i \sigma_{x,i}\frac{a_i^\top\zeta}{s_{x,i}}.
\end{aligned}$$

Thus, $\nabla V_x = A_x^\top \sigma_x$.

To get the formula for the Hessian, start by noting that

$$\begin{aligned}
\frac{\partial^2 V_x}{\partial x_k \partial x_l} &= \frac{\partial}{\partial x_k} e_l^\top A_x^\top \sigma_x = \frac{\partial}{\partial x_k}\sum_i a_i^\top H_x^{-1} a_i \frac{A_{il}}{s_{x,i}^3} \\
&= \sum_i \left(\frac{A_{il}}{s_{x,i}^3}\cdot\frac{\partial}{\partial x_k}a_i^\top H_x^{-1}a_i + a_i^\top H_x^{-1}a_i\cdot\frac{\partial}{\partial x_k}\frac{A_{il}}{s_{x,i}^3}\right) \\
&= \sum_i \left(\frac{A_{il}}{s_{x,i}^3}\cdot\frac{\partial}{\partial x_k}a_i^\top H_x^{-1}a_i + 3a_i^\top H_x^{-1}a_i\cdot\frac{A_{il}A_{ik}}{s_{x,i}^4}\right).
\end{aligned}$$

It remains to compute $\frac{\partial}{\partial x_k}a_i^\top H_x^{-1}a_i$. Towards this first note that similar to the calculation we did above,

$$H_{x+t\zeta}^{-1} - H_x^{-1} \approx -H_x^{-1}\Delta H_x^{-1}.$$

Thus, $a_i^\top H_{x+t\zeta}^{-1}a_i - a_i^\top H_x^{-1}a_i = -a_i^\top H_x^{-1}\left(2t\sum_j \frac{a_j a_j^\top}{s_{x,j}^2}\frac{a_j^\top\zeta}{s_{x,j}}\right)H_x^{-1}a_i = -2t\sum_j \frac{(a_i^\top H_x^{-1}a_j)^2}{s_{x,j}^2}\frac{a_j^\top\zeta}{s_{x,j}}$. Thus,

$$\frac{A_{il}}{s_{x,i}^3}\cdot\frac{\partial}{\partial x_k}a_i^\top H_x^{-1}a_i = -2\sum_j \frac{(a_i^\top H_x^{-1}a_j)^2}{s_{x,j}^2 s_{x,i}^2}\frac{A_{jk}}{s_{x,j}}\frac{A_{il}}{s_{x,i}} = -2A_x^\top P_x^{(2)}A_x.$$

Thus taking $\zeta = e_k$ and letting $t \to 0$, we obtain

$$\nabla^2 V_x = A_x^\top(3\Sigma_x - 2P_x^{(2)})A_x.$$

□

4.5 Appendix: John ellipsoid

In this section[4] we introduce the John ellipsoid and contrast it with the Dikin ellipsoid introduced in the previous lecture. In particular, we show a fundamental result that, for symmetric polytopes, the John ellipsoid $\sqrt{n}$ approximates the body (as opposed to the $\sqrt{m}$ achieved by the Dikin ellipsoid). Thus, just as we did with the Dikin ellipsoid, one can in principle define "The John Algorithm" and hope that it converges in $\sqrt{n}$ iterations. Whether it does remains far from clear. Recently, Lee-Sidford show that a *smoothened* version of John ellipsoid actually does converge in $\sqrt{n}\log^{O(1)}(m)$ iterations.

Definition 4.12. Given a bounded polytope $P \stackrel{\text{def}}{=} \{x \in \mathbb{R}^n : a_i^\top x \le b_i \text{ for } i = 1, \ldots, m\}$ and a point $x \in \text{int}(P)$, the *John ellipsoid* of P at x is defined to be the ellipsoid of maximum volume which is centered at x and contained in P.

Let us try to describe the John ellipsoid $\mathcal{E}_x$ centered at a point x using a quadratic form. Towards this, let B_x be a PSD matrix such that

$$\mathcal{E}_x = \{y : \|y - x\|_{B_x}^2 \le 1\}.$$

Then, the constraint that $\mathcal{E}_x$ is contained inside the polytope P would require that for every $i = 1, \ldots, m$

$$\mathcal{E}_x \subseteq \{y : \langle a_i, y\rangle \le b_i\}.$$

In other words for all i,

$$\max_{y \in \mathcal{E}_x} \langle a_i, y\rangle \le b_i$$

or equivalently

$$\max_{y : y^\top B_x y \le 1} \langle a_i, y + x\rangle \le b_i$$

or equivalently

$$\max_{y : y^\top B_x y \le 1} \langle a_i, y\rangle \le b_i - \langle a_i, x\rangle = s_i(x).$$

It can be checked that the left-hand side is equal to $\|a_i\|_{B_x^{-1}}$. Because $s_i(x) \ge 0$, we can square both sides and obtain the constraint

$$a_i^\top B_x^{-1} a_i \le s_i(x)^2.$$

As for the volume of $\mathcal{E}_x$ (which we are maximizing), we have that

$$\text{vol}(\mathcal{E}_x) = \text{vol}(\{y : y^\top B_x y \le 1\}) = \text{vol}(\{B_x^{-1/2} v : \|v\|_2 \le 1\}) = V_n \cdot \Big(\det(B_x^{-1/2})\Big)$$

where V_n is the volume of the unit ℓ_2 ball in $\mathbb{R}^n$. Ignoring the V_n term (by just redefining the volume relative to V_n) we obtain

$$\log \text{vol}\ \mathcal{E}_x = \frac{1}{2} \log \det(B_x^{-1}).$$

[4] Scribed by Jakub Tarnawski.

We take the logarithm here because this will allow us to obtain a convex program.

We can now write the following program, called (John-Primal):

$$\begin{aligned} \min \quad & -\log\det B^{-1} \\ \text{s.t.} \quad & \frac{a_i^\top B^{-1} a_i}{s_i(x)^2} \le 1 \qquad \text{for } i = 1, \ldots, m. \end{aligned}$$

Note that we deal with the constraint $B \succ 0$ (equivalently, $B^{-1} \succ 0$) by encoding it into the domain of the function $\log\det$.

Let us denote $C = B^{-1}$. The program (John-Primal) is convex in the variables $\{C_{ij}\}$.

4.5.1 Duality

Let us now write the dual program (John-Dual). We multiply the i-th constraint by a multiplier $w_i \ge 0$, obtaining the Lagrangian

$$\begin{aligned} L(C, w) &= -\log\det C + \sum_i w_i \left(\frac{a_i^\top C a_i}{s_i(x)^2} - 1 \right) \\ &= -\log\det C + C \bullet \left(\sum_i \frac{w_i}{s_i(x)^2} a_i a_i^\top \right) - \sum_i w_i. \end{aligned}$$

The KKT optimality condition $\nabla_C L(C, w) = 0$ yields

$$-C^{-1} + \sum_i \frac{w_i}{s_i(x)^2} a_i a_i^\top = 0$$

and thus

$$B = C^{-1} = \sum_i \frac{w_i}{s_i(x)^2} a_i a_i^\top.$$

The dual objective function is

$$g(w) = \inf_C L(C, w) = -\log\det \left(\sum_i \frac{w_i}{s_i(x)^2} a_i a_i^\top \right)^{-1} + n - \sum_i w_i,$$

with constraints $w_i \ge 0$ and $\sum_i \frac{w_i}{s_i(x)^2} a_i a_i^\top \succ 0$ (the latter arising from our restriction of $\log\det$ to PD matrices).

Slater's condition holds (just consider a small ball certifying that $x \in \operatorname{int}(P)$) and we have strong duality. Thus, at optimal values of C and w:

$$-\log\det C = -\log\det C - \sum_i w_i + n,$$

and hence

$$\sum_i w_i = n.$$

We can interpret w_i as a measure of how strongly the ellipsoid $\mathcal{E}_x$ is supported on the hyperplane defined by the i-th constraint. For example, from the complementary slackness conditions we have that if $a_i^\top B^{-1} a_i < s_i(x)^2$ (i.e., the ellipsoid does not touch the hyperplane), then $w_i = 0$.

4.5.2 Approximating symmetric polytopes

Now we show that the John ellipsoid improves the $\sqrt{m}$ of the Dikin ellipsoid to $\sqrt{n}$ for symmetric polytopes.

Proposition 4.13. Let P be a polytope which is symmetric around the origin (i.e., $P = -P$), $0 \in \text{int}(P)$, and let $\mathcal{E}$ be the John ellipsoid of P at 0. Then the ellipsoid $\sqrt{n}\mathcal{E}$ contains P.

Proof. The proof resembles the one of a similar statement for Dikin ellipsoids. Without loss of generality assume that all $b_i = 1$. Also, because P is symmetric, we assume that for every constraint $\langle a_i, x\rangle \le 1$ there is a corresponding constraint $\langle -a_i, x\rangle \le 1$. Now all $s_i(x) = 1$ and the John ellipsoid $\mathcal{E}$ is given by

$$B = \sum_i w_i a_i a_i^\top, \quad \mathcal{E} = \{x : x^\top Bx \le 1\} = \left\{x : \sum_i w_i \langle a_i, x\rangle^2 \le 1\right\}.$$

Pick a point x on the boundary of $\mathcal{E}$. It is enough to show that $\sqrt{n}x$ is not inside K. Since $x \in \partial\mathcal{E}$, we have

$$\sum_i w_i \langle a_i, x\rangle^2 = 1.$$

If it were the case that $\sqrt{n}x \in \text{int}(P)$, then for every i we would have $\langle a_i, \sqrt{n}x\rangle < 1$ and $\langle -a_i, \sqrt{n}x\rangle < 1$, so that

$$\langle a_i, \sqrt{n}x\rangle^2 < 1,$$

and multiplying by w_i/n and summing over all i we would get

$$\sum_i w_i \langle a_i, x\rangle^2 = \sum_i \frac{w_i}{n} \langle a_i, \sqrt{n}x\rangle^2 < \frac{\sum_i w_i}{n} = 1,$$

a contradiction. □

4.5.3 The John Algorithm?

Thus, the John ellipsoid provides us with an alternative to the Dikin ellipsoid and one may ask what is stopping us from defining a barrier similar to Vaidya's volumetric barrier. After all, it does give us a barrier of the form

$$\sum_i w_i(x) \frac{a_i a_i^\top}{s_i(x)^2}$$

with weights summing up to n at every point raising the possibility of a $\sqrt{n}$-iteration algorithm. Unfortunately, the weights $w_i(x)$ are not even continuous (let alone differentiable). For example, consider the square $[-1, 1]^2$ defined by the four natural constraints (suppose w_1 corresponds to $x \le 1$ and w_2 corresponds to $-x \le 1$). At a point $(\varepsilon, 0)$, with ε very small, we have $w_1 = 1$ and $w_2 = 0$, but at $(-\varepsilon, 0)$ the converse is true. However, Lee-Sidford were able to find a workaround by *smoothening* the weights at a poly-log cost in the number of iterations. The details are quite involved and omitted here.

References

[SV14] Sushant Sachdeva and Nisheeth K. Vishnoi. Faster algorithms via approximation theory. *Foundations and Trends in Theoretical Computer Science*, 9(2):125–210, 2014. URL: `http://dx.doi.org/10.1561/0400000065`, `doi:10.1561/0400000065`.

[Vis13] Nisheeth K. Vishnoi. Lx = b. *Foundations and Trends in Theoretical Computer Science*, 8(1-2):1–141, 2013. URL: `http://research.microsoft.com/en-us/um/people/nvishno/site/Lxb-Web.pdf`.

[Vis14] Nisheeth K. Vishnoi. Fundamentals of convex optimization, 2014. URL: `http://theory.epfl.ch/vishnoi/Nisheeth-VishnoiFall2014-ConvexOptimization.pdf`.

[Wri05] Margaret H. Wright. The interior-point revolution in optimization: history, recent developments, and lasting consequences. *Bull. Amer. Math. Soc. (N.S*, 42:39–56, 2005.