EFFICIENT IMPLEMENTATION OF INTERIOR-POINT METHODS FOR QUANTUM RELATIVE ENTROPY

MEHDI KARIMI AND LEVENT TUNÇEL

Abstract. Quantum Relative Entropy (QRE) programming is a recently popular and challenging class of convex optimization problems with significant applications in quantum computing and quantum information theory. We are interested in modern interior point (IP) methods based on optimal self-concordant barriers for the QRE cone. A range of theoretical and numerical challenges associated with such barrier functions and the QRE cones have hindered the scalability of IP methods. To address these challenges, we propose a series of numerical and linear algebraic techniques and heuristics aimed at enhancing the efficiency of gradient and Hessian computations for the self-concordant barrier function, solving linear systems, and performing matrix-vector products. We also introduce and deliberate about some interesting concepts related to QRE such as symmetric quantum relative entropy (SQRE). We design a two-phase method for performing facial reduction that can significantly improve the performance of QRE programming. Our new techniques have been implemented in the latest version (DDS 2.2) of the software package DDS. In addition to handling QRE constraints, DDS accepts any combination of several other conic and non-conic convex constraints. Our comprehensive numerical experiments encompass several parts including 1) a comparison of DDS 2.2 with Hypatia for the nearest correlation matrix problem, 2) using DDS 2.2 for combining QRE constraints with various other constraint types, and 3) calculating the key rate for quantum key distribution (QKD) channels and presenting results for several QKD protocols.

Mehdi Karimi: Department of Mathematics, Illinois State University, Normal, IL, 61761. (e-mail: **mkarim3@ilstu.edu**). Research of this author was supported in part by the National Science Foundation (NSF) under Grant No. CMMI-2347120.

Levent Tunçel: Department of Combinatorics and Optimization, Faculty of Mathematics, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada (e-mail: **levent.tuncel@uwaterloo.ca**). Research of this author was supported in part by Discovery Grants from the Natural Sciences and Engineering Research Council (NSERC) of Canada.

1. Introduction

In this manuscript, we propose and evaluate various techniques to efficiently solve convex optimization problems involving the quantum relative entropy (QRE) cone using interior-point methods. Optimization over problems involving the QRE cone has several applications in quantum computing. These applications include calculating the key rate of quantum key distribution (QKD) channels ([40, 49]) or calculating the quantum rate-distortion function ([10, 20]). QKD is a commercialized secure communication method that distributes a secret key between two honest parties in the presence of an eavesdropper. The rate-distortion function is a fundamental concept in information theory that quantifies the minimum achievable compression rate for transmitting a source signal within a specified distortion or reconstruction error bound ([8]). The quantum relative entropy function is the matrix extension of vector relative entropy or Kullback-Leibler (KL) divergence ([33, 8, 46, 19, 6]) of two vectors which is defined as $KL : \mathbb{R}^n \oplus \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$:

$$KL(x,y) := \begin{cases} \sum_{i=1}^n x_i \ln(x_i) - x_i \ln(y_i), & x, y \in \mathbb{R}^n_+, \mathrm{supp}(x) \subseteq \mathrm{supp}(y), \\ +\infty & \text{o.w.} \end{cases} \tag{1}$$

where $\mathrm{supp}(x) := \{i : x_i \neq 0\}$ denotes the support of x. KL divergence, mostly used to measure the difference of two probability distributions, is an important function in statistics, information theory, optimization, and machine learning. KL divergence is widely used in machine learning for tasks like information retrieval, clustering, generative modeling, and variational autoencoders ([11, 17]). KL divergence is also used for convergence proof in classic and quantum zero-sum games by [9, 5]. To define the quantum version of the KL divergence, we need the definition of the matrix extension of a univariate function. Consider a function $f : \mathbb{R} \to \mathbb{R} \cup \{+\infty\}$ and let $X \in \mathbb{H}^n$ ($\mathbb{H}^n$ is the set of n-by-n Hermitian matrices with complex entries) with a spectral decomposition $X = U\mathrm{Diag}(\lambda_1, \ldots, \lambda_n)U^*$, where Diag returns a diagonal matrix with the given entries on its diagonal and U^* is the conjugate transpose of a unitary matrix U. We define the *matrix extension* F of f as $F(X) := U\mathrm{Diag}(f(\lambda_1), \ldots, f(\lambda_n))U^*$. Then, we define the trace of this extension function as

$$\mathrm{Tr}(F(X)) := \begin{cases} \mathrm{Tr}(U\mathrm{Diag}(f(\lambda_1), \ldots, f(\lambda_n))U^*) & \text{if } f(\lambda_i) \in \mathbb{R},\ \forall i, \\ +\infty & \text{o.w.} \end{cases} \tag{2}$$

For the special case of $f(x) = x\ln(x)$, we use the convention that $f(0) := 0$, so in this special case, $\mathrm{Tr}(F(X))$ has a real value for every positive semidefinite matrix. For two matrices

$X, Y \in \mathbb{H}^n_+$, the quantity $\mathrm{Tr}(X\ln(Y))$ is real if the intersection of the null space of Y and the range of X is the zero vector (equivalently, range of X is contained in the range of Y). Then, we can define the quantum relative entropy function $qre : \mathbb{H}^n \oplus \mathbb{H}^n \to \mathbb{R} \cup \{+\infty\}$ as

$$qre(X,Y) := \begin{cases} \mathrm{Tr}(X\ln(X) - X\ln(Y)) & \text{if } X, Y \in \mathbb{H}^n_+ \text{ and } \mathrm{range}(X) \cap \mathrm{null}(Y) = \{0\}, \\ +\infty & \text{o.w.} \end{cases}$$

The QRE cone is defined as the epigraph of the qre function:

$$QRE^n := \left\{(t, X, Y) \in \mathbb{R} \oplus \mathbb{H}^n_+ \oplus \mathbb{H}^n_+ : qre(X,Y) \leq t\right\}.$$

QRE programming concerns with optimization problems over the intersection of one or more QRE cones with an affine subspace and potentially many other simpler convex sets. One approach to solve QRE programming is approximating it with other tractable optimization classes such as semidefinite programming (SDP) ([14, 2]). These SDP approximations are expensive and do not scale well; therefore, work only for small size problems. We are interested in using modern interior-point (IP) algorithms for convex optimization based on the theory of self-concordant (s.c.) functions and barriers ([36]). The intriguing theoretical properties exhibited by the s.c. barriers for symmetric cones [37, 38], along with their extensive practical applications, have positioned the primary research emphasis of IP algorithms on symmetric cones. After decades of successful implementation of interior-point algorithms for optimization over symmetric cones (see, for instance, [43, 42, 35, 27]), there have been several recent efforts to create efficient codes for handling other convex sets with available computationally efficient self-concordant (s.c.) barriers. The available modern IP codes for solving convex optimization problems (beyond symmetric cones) using s.c. barriers are: a MATLAB-based software package Alfonso ([39]), a software package Hypatia in the Julia language ([7]), and a MATLAB-based software package DDS ([27]). DDS has some major differences from the other two, including: 1) accepting both conic and non-conic constraints, 2) utilizing the Legendre-Fenchel conjugate of the s.c. barriers when available, and 3) using conservative strategies for the implementation of each iteration, and strict stopping criteria (for improved robustness, see [31]).

[36] presented a theoretical s.c. function (universal barrier) for any closed convex set, which is very costly to compute in general. To apply modern IP methods to optimization problems involving the QRE cone or any other convex set, a computationally efficient s.c. barrier in needed, where its gradient and Hessian can be calculated in a reasonable time. DDS has been using the following barrier function since 2019 for solving problems involving

quantum relative entropy constraints $\Phi : \mathbb{R} \oplus \mathbb{H}^n \oplus \mathbb{H}^n \to \mathbb{R} \cup \{+\infty\}$:

$$(3)\ \Phi(t, X, Y) := \begin{cases} -\ln(t - qre(X, Y)) - \ln\det(X) - \ln\det(Y), & \text{if } X, Y \in \mathbb{H}^n_{++}, \\ +\infty & \text{o.w.} \end{cases}$$

which was recently proved to be self-concordant by Fawzi and Saunderson ([12]). The first available code for QRE programming was CVXQUAD ([13]), which is a collection of matrix functions to be used on top of CVX ([18]). The package CVXQUAD is based on the paper ([14]), which approximates the matrix logarithm with functions that can be described by SDPs. CVXQUAD does not scale well and the SDP approximation becomes too large for available SDP solvers even for matrices of size 15. Note that the SDP formulation of the quantum relative entropy function in [14] involves linear matrix inequalities of size $n^2 \times n^2$. [15] proved self-concordance results for some functions related to quantum entropy and then [16] designed some interior-point algorithms for various problems involving quantum entropy. They also consider minimizing the $qre(X, Y)$ function for the special case of calculating the key rate for quantum key distribution (QKD) channels, which is one of the most popular applications of QRE programming. For this special case, both X and Y are linear functions of another matrix ρ and the problem formulation is significantly simpler than our general QRE setup[1]. [25] also proposed an interior-point method, not using the s.c. barriers for the QRE cone, for solving QKD key rate using the simplified formulation. As far as we know, Hypatia and DDS are the only publicly available codes for QRE programming where both use the s.c. barrier in (3). The main differences of DDS (and Hypatia) with [16] and [25] are:

- DDS QRE programming is not just for the simplified QKD key rate computation, but is for any problem with a combination of an arbitrary number of QRE cones, with all the other function/set constraints available in DDS.
- DDS is using the optimal s.c. barrier in (3). [16] use their conjectured s.c. function for the simplified problem, and [25] do not utilize self-concordance.

Several computational and theoretical challenges related to (3) have hindered the scalability of Quantum Relative Entropy (QRE) optimization solvers, specifically DDS 2.1 and Hypatia. Among the primary issues are the complexity of evaluating the gradient and Hessian of Φ in (3), and also solving the linear systems involving the Hessian, which are needed in implementing the second-order IP methods. In this paper, we present a set of numerical and theoretical techniques aimed at enhancing the performance of IP methods,

[1]The code of [16] is not publicly available.

and then evaluate the effectiveness of these techniques through a series of numerical experiments. The new techniques have been implemented in DDS 2.2 ([28]), which is introduced and released by this paper. Here are the contributions of this paper:

- Introducing numerical and linear algebraic techniques and heuristics to improve the calculation of the gradient and Hessian of Φ, solving the needed linear systems, and calculating the matrix-vector products. These techniques improved DDS 2.2, and enabled us to solve much larger instances compared to DDS 2.1 and Hypatia (Sections 2 and 3).
- Introducing and deliberating about the concept of *symmetric quantum relative entropy* (Section 4).
- Introducing a two-phase approach for QRE programming to improve the running time and condition of the problems (Section 5).
- Developing a comprehensive setup (including a two-phase approach and facial reduction) for calculating quantum key distribution (QKD) channel rates (Section 6), and discussing how to handle complex Hermitian matrices (Section 7).
- A comprehensive numerical experiment including: 1) comparison of DDS 2.2 with Hypatia for the nearest correlation matrix, 2) using DDS for combination of QRE and many other types of constraints, and 3) examples to elaborate on the two-phase method and its performance improvement, 4) Solving symmetric QRE programming problems, 5) calculating the key rate for QKD channels and presenting results for several QKD protocols (Section 8).

1.1. **Notations.** The sets $\mathbb{S}^n$, $\mathbb{S}^n_+$, and $\mathbb{S}^n_{++}$ are the set of n-by-n symmetric matrices, positive semidefinite matrices, and positive definite matrices, respectively. For a multivariate function f, both f' and ∇f are used for the gradient, and both f'' and $\nabla^2 f$ are used for the Hessian.

2. Evaluating the Derivatives for Quantum Relative Entropy

In this section, we discuss how to calculate the gradient and Hessian for the s.c. barrier function in (3). The details of our Domain-Driven infeasible-start predictor-corrector algorithms are given in [30, 31]. At each iteration of the algorithm, for both the predictor

and corrector steps, we solve a linear system of the form:

$$\left(U^\top \underbrace{\begin{bmatrix} \bar{H} & 0 \\ 0 & \left[\hat{H}\right]^{-1} \end{bmatrix}}_{\mathcal{H}(\bar{H},\hat{H})} U \right) d = r_{RHS}, \tag{4}$$

where $\bar{H}$ and $\hat{H}$ are positive definite matrices based on the Hessians of the s.c. barrier Φ representing the feasible set, U is a fixed matrix, and r_{RHS} is the right-hand-side vector that is different for the predictor and corrector steps. The following pseudocode gives a framework for the algorithm ([30, 31]):

Framework for the Interior-Point Algorithm in DDS

INPUT: Matrix U, the oracles for calculating Φ and its derivatives. A proximity measure Ω and the constants $0 < \delta_1 < \delta_2$.
while (the stopping criteria are not met)

Predictor Step

Calculate the predictor search direction d by solving (4) with the proper r_{RHS}. Update the current point using the search direction and a step size that guarantees $\Omega \leq \delta_2$.

Corrector Step

Modify the system and the r_{RHS} in (4) and calculate the corrector search direction d. Apply the corrector step at least once so that the updated point satisfies $\Omega < \delta_1$.

Fine Tuning

Modify the dual iterate to make sure it approximately satisfies dual feasibility.

end while

For QRE programming, the main challenge in forming the linear system (4) is calculating the gradient and Hessian for the $qre(X,Y)$ function in an efficient and numerically stable way. For this, we need to calculate the derivatives for $\mathrm{Tr}(X\ln(X))$ and $\mathrm{Tr}(-X\ln(Y))$. $X\ln(X)$ is the matrix extension of $x\ln(x)$. For the trace of a general matrix extension function $\mathrm{Tr}(F(X))$ defined in (2), the gradient can be calculated by the following theorem:

Theorem 2.1 (see, for example, [21]-Section 3.3)**.** *Let X and H be self-adjoint matrices and $f:(a,b)\mapsto\mathbb{R}$ be a continuously differentiable function defined on an interval. Assume that the eigenvalues of $X+\alpha H$ are in (a,b) for an interval around $\alpha_0\in\mathbb{R}$. Then,*

$$\frac{d}{d\alpha}\mathrm{Tr}F(X+\alpha H)\Big|_{\alpha=\alpha_0}=\mathrm{Tr}HF'(X+\alpha_0 H). \tag{5}$$

By putting $\alpha_0=0$ in (5), we get the directional derivative of F in the direction of H, and we can write:

$$\frac{d}{d\alpha}\mathrm{Tr}F(X+\alpha H)\Big|_{\alpha=0}=\mathrm{Tr}HF'(X)=\mathrm{vec}(F'(X))^\top\mathrm{vec}(H),$$

where vec changes a matrix into a vector by stacking the columns on top of one another. This implies that if we look at $(\mathrm{Tr}(F(X)))'$ as a vector of length n^2, we have

$$(\mathrm{Tr}(F(X)))'=\mathrm{vec}(F'(X)). \tag{6}$$

For the rest of our discussion, we need two definitions for univariate functions similar to the derivative. For a continuously differentiable function $f:(a,b)\mapsto\mathbb{R}$, we define the first and second divided differences as

$$f^{[1]}(\alpha,\beta) := \begin{cases} \frac{f(\alpha)-f(\beta)}{\alpha-\beta} & \alpha\neq\beta,\\ f'(\alpha) & \alpha=\beta.\end{cases}$$

$$f^{[2]}(\alpha,\beta,\gamma) := \begin{cases} \frac{f^{[1]}(\alpha,\beta)-f^{[1]}(\alpha,\gamma)}{\beta-\gamma} & \beta\neq\gamma,\\ \frac{f^{[1]}(\alpha,\beta)-f'(\beta)}{\alpha-\beta} & \beta=\gamma\neq\alpha,\\ -\frac{1}{2}f''(\alpha) & \beta=\gamma=\alpha.\end{cases} \tag{7}$$

To calculate the Hessian of $\mathrm{Tr}(F(X))$, we can use the following theorem:

Theorem 2.2 (see, for example, [21]-Theorem 3.25)**.** *Assume that $f:(a,b)\mapsto\mathbb{R}$ is a $\mathcal{C}^1$-function and $T=\mathrm{Diag}(t_1,\ldots,t_n)$ with $t_i\in(a,b)$, $i\in\{1,\ldots,n\}$. Then, for a Hermitian*

matrix H, we have

$$\frac{d}{d\alpha}F(T+\alpha H)\Big|_{\alpha=0} = T_f \odot H, \tag{8}$$

where $\odot$ is the Hadamard product and T_f is the divided difference matrix defined as

$$[T_f]_{ij} := f^{[1]}(t_i, t_j), \quad \forall i,j \in \{1,\ldots,n\}. \tag{9}$$

T is diagonal in the statement of the theorem, which is without loss of generality. Note that by the definition of functional calculus in (2), for a Hermitian matrix X and a unitary matrix U, we have

$$F(UXU^*) = UF(X)U^*. \tag{10}$$

Therefore, for a matrix $T = U\mathrm{Diag}(t_1,\ldots,t_n)U^*$, we can update (8) as

$$\frac{d}{d\alpha}F(T+\alpha H)\Big|_{\alpha=0} = U\left(T_f \odot (U^*HU)\right)U^*, \tag{11}$$

where we extend the definition of T_f in (9) to non-diagonal matrices by defining that $T_f := (U^*TU)_f$. In other words, for a non-diagonal matrix $T = U\mathrm{Diag}(t_1,\ldots,t_n)U^*$, we use $\mathrm{Diag}(t_1,\ldots,t_n)$ to calculate T_f using (9). Now we can use Theorems 2.2 and 2.1 to calculate the Hessian of the function $\mathrm{Tr}(F(X))$.

Theorem 2.3. *Let X, H, and $\tilde{H}$ be self-adjoint matrices and $f : (a,b) \mapsto \mathbb{R}$ be a continuously differentiable function defined on an interval. Assume that the eigenvalues of $X+tH$ and $X+t\tilde{H}$ are in (a,b) for an interval around $t=0$. Assume that $X = U_X\mathrm{Diag}(\lambda_1,\ldots,\lambda_n)U_X^*$. Then,*

$$\nabla^2\mathrm{Tr}(F(X))[H,\tilde{H}] = \mathrm{Tr}\left(\left(X_{f'} \odot (U_X^*HU_X)\right)U_X^*\tilde{H}U_X\right). \tag{12}$$

Proof. Proof. We can write

$$\begin{aligned}
\nabla^2\mathrm{Tr}(F(X))[H,\tilde{H}] &= \tfrac{d}{d\beta}\tfrac{d}{d\alpha}\mathrm{Tr}(F(X+\beta H+\alpha\tilde{H}))\Big|_{\alpha=0}\Big|_{\beta=0} \\
&= \tfrac{d}{d\beta}\mathrm{Tr}(HF'(X+\beta H))\Big|_{\beta=0}, && \text{using (5)} \\
&= \mathrm{Tr}(\tilde{H}\ \tfrac{d}{d\beta}F'(X+\beta H)\Big|_{\beta=0}) \qquad (13)\\
&= \mathrm{Tr}(\tilde{H}U_X\left(X_{f'} \odot (U_X^*HU_X)\right)U_X^*), && \text{using (11)} \\
&= \mathrm{Tr}\left(\left(X_{f'} \odot (U_X^*HU_X)\right)U_X^*\tilde{H}U_X\right).
\end{aligned}$$

□

To find a formula for the matrix form of the Hessian, note that by using the properties of the Hadamard product, we have

$$\mathrm{vec}(X_{f'} \odot (U_X^* H U_X)) = \mathrm{Diag}(\mathrm{vec}(X_{f'}))\mathrm{vec}(U_X^* H U_X).$$

Using this, we have

$$\begin{aligned} \mathrm{Tr}\left((X_{f'} \odot (U_X^* H U_X))\, U_X^* \tilde{H} U_X\right) &= \mathrm{vec}(X_{f'} \odot (U_X^* H U_X))^\top \mathrm{vec}(U_X^* \tilde{H} U_X) \\ &= \mathrm{vec}(U_X^* H U_X)^\top \mathrm{Diag}(\mathrm{vec}(X_{f'}))\mathrm{vec}(U_X^* \tilde{H} U_X) \\ &= \mathrm{vec}(H)^\top (U_X \otimes U_X)\mathrm{Diag}(\mathrm{vec}(X_{f'}))(U_X^* \otimes U_X^*)\mathrm{vec}(\tilde{H}). \end{aligned} \tag{14}$$

So the matrix form of the Hessian is

$$(\mathrm{Tr}(F))''(X) = (U_X \otimes U_X)\mathrm{Diag}(\mathrm{vec}(X_{f'}))(U_X^* \otimes U_X^*). \tag{15}$$

For the other components of the Hessian of qre, we need to differentiate $\mathrm{Tr}(-X\ln(Y))$ in terms of X and Y. In terms of Y, for a fixed matrix X and a continuously differentiable function $f : (a, b) \mapsto \mathbb{R}$, let us define

$$F_X(Y) := \mathrm{Tr}(XF(Y)). \tag{16}$$

Let $Y = U_Y \mathrm{Diag}(\gamma_1, \ldots, \gamma_n) U_Y^*$ be the spectral decomposition of Y. The gradient of $F_X(Y)$ can be calculated using Theorem 2.2 as:

$$F_X'(Y) = U_Y \left(Y_f \odot (U_Y^* X U_Y)\right) U_Y^*. \tag{17}$$

The Hessian of $F_X(Y)$ is calculated as follows in [16]:

$$F_X''(Y) = (U_Y \otimes U_Y) S (U_Y^* \otimes U_Y^*), \tag{18}$$

where S is the n^2-by-n^2 second divided difference matrix. If we assume that S is a block matrix of size n-by-n where each block is again a matrix of size n-by-n, then we can show the entries of S as $S_{ij,kl}$ where ij denotes the place of the block, and kl denotes the rows and columns inside the block. We have:

$$S_{ij,kl} = \delta_{kl} \tilde{X}_{ij} f^{[2]}(\gamma_i, \gamma_j, \gamma_l) + \delta_{ij} \tilde{X}_{kl} f^{[2]}(\gamma_j, \gamma_k, \gamma_l), \tag{19}$$

where $\tilde{X} := U_Y^* X U_Y$ and δ_{ij} is an indicator function which is 1 if $i = j$, and 0 otherwise. Putting together all these results, we have

$$qre''(X, Y) = \begin{bmatrix} H_{11} & H_{12} \\ H_{12}^\top & H_{22} \end{bmatrix},$$

$$H_{11} = (U_X \otimes U_X)(\mathrm{Diag}(\mathrm{vec}(X_{\ln})))(U_X \otimes U_X),$$

$$H_{12} = -(U_Y \otimes U_Y)(\mathrm{Diag}(\mathrm{vec}(Y_{\ln})))(U_Y \otimes U_Y),$$

$$H_{22} = -(U_Y \otimes U_Y)S(U_Y^* \otimes U_Y^*).$$

By having the derivatives of the qre function, we can calculate the derivatives for the s.c. barrier function Φ in (3). For simplicity, we define $T := t - qre(X,Y)$. We have

$$\Phi'(t,X,Y) = \begin{bmatrix} \frac{-1}{T} \\ \frac{1}{T}h + \mathrm{vec}(-X^{-1}) \\ \frac{1}{T}\bar{h} + \mathrm{vec}(-Y^{-1}) \end{bmatrix}, \quad \begin{array}{l} h := \mathrm{vec}(I + \ln(X) - \ln(Y)) \\ \bar{h} := \mathrm{vec}((U_Y\,(Y_f \odot (U_Y^* X U_Y))\,U_Y^*)). \end{array} \tag{20}$$

We can write the Hessian as:

$$\Phi''(t,X,Y) = \underbrace{\begin{bmatrix} \frac{1}{T^2} & -\frac{1}{T^2}h^\top & \frac{1}{T^2}\bar{h}^\top \\ -\frac{1}{T^2}h & \frac{1}{T}H_{11} + (X^{-1}\otimes X^{-1}) & \frac{1}{T}H_{12} \\ \frac{1}{T^2}\bar{h} & \frac{1}{T}H_{12}^\top & \frac{1}{T}H_{22} + (Y^{-1}\otimes Y^{-1}) \end{bmatrix}}_{\bar{H}} + \begin{bmatrix} 0 \\ \frac{1}{T}h \\ \frac{1}{T}\bar{h} \end{bmatrix} \begin{bmatrix} 0 \\ \frac{1}{T}h \\ \frac{1}{T}\bar{h} \end{bmatrix}^\top. \tag{21}$$

2.1. **Other numerical techniques.** For calculating the gradient and Hessian of $qre(X,Y)$, numerical instability happens in calculating $X_{\ln}$ and $Y_{\ln}$, where for a matrix $X = U_X \mathrm{Diag}(\lambda_1,\ldots,\lambda_n)U_X^*$, using (9), we have:

$$[X_{\ln}]_{ij} = \ln^{[1]}(\lambda_i,\lambda_j) \;\; = \;\; \begin{cases} \frac{1}{\lambda_i} & \lambda_i = \lambda_j \\ \frac{\ln(\lambda_i)-\ln(\lambda_j)}{\lambda_i-\lambda_j} & \lambda_i \neq \lambda_j. \end{cases}$$

To make the calculation more stable, [15] used the following equivalent formula given in [23]:

$$\ln^{[1]}(\lambda_i,\lambda_j) \;\; = \;\; \begin{cases} \frac{1}{\lambda_i} & \lambda_i = \lambda_j \\ \frac{\ln(\lambda_i)-\ln(\lambda_j)}{\lambda_i-\lambda_j} & \lambda_i < \frac{\lambda_j}{2} \;\text{ or }\; \lambda_j < \frac{\lambda_i}{2} \\ \frac{2\tanh^{-1}(z)}{\lambda_i-\lambda_j} & \text{o.w.} \end{cases}$$

where $z = (\lambda_i - \lambda_j)/(\lambda_i + \lambda_j)$. Numerical experiments with DDS 2.2 shows that this formula indeed works better in terms of numerical stability.

3. Solving the Linear System

In QRE programming with DDS, other than solving the linear system (4), for calculating the proximity measure Ω and the fine-tuning step, we need to solve linear systems that directly have the Hessian on the left-hand-side. In this section, we discuss how to solve these systems in DDS to significantly speed up the algorithm. Writing the Hessian as in (21) is efficient since the Hessian is the summation of a simpler matrix and a rank one matrix, and we can use Sherman–Morrison formula[2] to solve the linear systems involving the Hessian. By the Sherman–Morrison formula, the linear system reduces to a linear system with $\bar{H}$ defined in (21). With some linear algebra, the main part of the reduced linear system is the one involving the matrix

$$\left[\begin{array}{cc} \frac{1}{T}H_{11} + (X^{-1}\otimes X^{-1}) & \frac{1}{T}H_{12} \\ \frac{1}{T}H_{12}^{\top} & \frac{1}{T}H_{22} + (Y^{-1}\otimes Y^{-1}) \end{array}\right]. \tag{22}$$

Calculating $U_X \otimes U_X$ and $U_Y \otimes U_Y$ is the main computational challenge in forming this matrix. U_X and U_Y are dense matrices, but have the nice property that both are unitary matrices, which we can exploit. By defining $1/\lambda := \mathrm{Diag}(1/\lambda_1, \ldots, 1/\lambda_n)$ and $1/\gamma := \mathrm{Diag}(1/\gamma_1, \ldots, 1/\gamma_n)$, we can write the diagonal block matrices of (22) as

$$\begin{aligned} \frac{1}{T}H_{11} + (X^{-1}\otimes X^{-1}) &= (U_X\otimes U_X)(\mathrm{Diag}(\frac{1}{T}\mathrm{vec}(X_{\ln}) + 1/\lambda \otimes 1/\lambda))(U_X\otimes U_X) \\ \frac{1}{T}H_{22} + (Y^{-1}\otimes Y^{-1}) &= (U_Y\otimes U_Y)(-\frac{1}{T}S + \mathrm{Diag}(1/\gamma\otimes 1/\gamma))(U_Y^*\otimes U_Y^*). \end{aligned}$$

The approximation we made in DDS 2.2 for solving the linear systems with the Hessian of Φ on the left hand side is ignoring the off-diagonal block matrices in (22). Doing this, the matrix can be factorized as

$$U_{XY}\left[\begin{array}{cc} \mathrm{Diag}(\frac{1}{T}\mathrm{vec}(X_{\ln}) + 1/\lambda\otimes 1/\lambda) & 0 \\ 0 & -\frac{1}{T}S + \mathrm{Diag}(1/\gamma\otimes 1/\gamma) \end{array}\right] U_{XY}^*,$$

$$U_{XY} := \left[\begin{array}{cc} U_X\otimes U_X & 0 \\ 0 & U_Y\otimes U_Y \end{array}\right].$$

[2]Sherman–Morrison formula has been employed in solving convex optimization problems for a long time. These algorithms go back to the implementation of quasi-Newton methods, implementation of ellipsoid methods (see [3] and references therein), as well as interior-point methods (see [32], [44] and references therein)

By this simplification, solving the linear system is reduced to solving a system with the matrix in the middle and mostly a system involving S. Note that in DDS 2.2, this simplification is not used for the linear system (4) for calculating the search directions of the IP method. For those systems, the matrix on the left hand side is a quadratic function of U, and the Hessian does not directly appear in the left-hand-side.

4. Symmetric Quantum Relative Entropy

As vector relative entropy or Kullback-Leibler divergence is not symmetric, there is a natural way to symmetrize the KL function defined in (1) as $J : \mathbb{R}^n \oplus \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$:

$$J(x,y) := \begin{cases} KL(x,y) + KL(y,x), & x, y \in \mathbb{R}^n_+, \text{supp}(x) = \text{supp}(y), \\ +\infty & \text{o.w.} \end{cases} \tag{23}$$

$J(x, y)$ is called Jeffreys divergence or symmetrized Kullback-Leibler divergence ([33, 26]). In a similar way, we define the symmetric QRE function as $sqre : \mathbb{H}^n \oplus \mathbb{H}^n \to \mathbb{R} \cup \{\infty\}$:

$$sqre(X,Y) := \begin{cases} qre(X,Y) + qre(Y,X) & X, Y \in \mathbb{H}^n_+, \text{ range}(X) = \text{range}(Y), \\ +\infty & \text{o.w.} \end{cases} \tag{24}$$

SQRE is a straightforward extension of Jeffreys divergence, and it was also suggested in the context of QRE ([41]). Clearly $sqre$ is also a convex function in (X, Y). A code that accepts QRE constraints can also handle SQRE constraints using the following reformulation:

$$qre(X,Y) + qre(Y,X) \le t \quad \equiv \begin{cases} t_1 + t_2 & \le & t \\ qre(X,Y) & \le & t_1 \\ qre(Y,X) & \le & t_2. \end{cases} \tag{25}$$

The only issue with this approach is that by this reformulation, the s.c. barrier assigned to two QRE constraints has parameter $2n$. However, it is plausible that there exists an efficient s.c. barrier with a better parameter for the epigraph of $sqre$. In Subsection 8.4, we present some numerical results to show that formulation (25) works well in practice.

5. Two-phase Methods

In this section, we propose two-phase methods for solving QRE programming which can significantly improve the running time and condition of the problem. This two-phase

approach is different than, for example, the one for solving LP problems or SDP problems where the two phases are roughly equivalent in terms of size and complexity. Here, phase one for the QRE programming is a convex optimization problem that can be solved more efficiently and much faster compared to the QRE problem. The solution of Phase-I is used to reformulate the QRE problem to make it a smaller size and well-conditioned (or, at least, better conditioned) QRE programming problem.

Framework for the Two-Phase QRE Programming

INPUT: A QRE optimization problem (P_{QRE}).
Phase-I: Create an auxiliary optimization problem (P_{AUX}), which is more robust and much faster to solve by DDS.
Reformulation: Use the solution of (P_{AUX}) to reformulate (P_{QRE}) as a smaller size and better conditioned QRE programming problem ($\bar{P}_{QRE}$).
Phase-II: Use DDS to solve ($\bar{P}_{QRE}$).
Solution Reconstruction: Use the solution of ($\bar{P}_{QRE}$) to create a corresponding solution for (P_{QRE}).

Consider an optimization problem of the form

$$\begin{aligned} \min \quad & qre\left(\sum_{i=1}^{k} x_i A_i, M\right) \\ & \ell \leq x \leq u, \end{aligned} \tag{26}$$

where $M \in \mathbb{S}_+^n$ is given. Assume that the set of points $\sum_{i=1}^{k} x_i A_i \in \mathbb{S}_+^n$ lie on a smaller *face* of the $\mathbb{S}_+^n$ cone (see, for instance, Chapter 2 of [45]). In other words, there exist a positive integer $r < n$ and an n-by-r matrix V with orthonormal columns such that

$$\left\{ X : X = \sum_{i=1}^{k} x_i A_i \right\} \cap \mathbb{S}_+^n \subset V \mathbb{S}_+^r V^\top.$$

We have the freedom to model the phase-I problem to not just calculate a matrix V, but also calculate some other useful information. We present two methods for problem (26):

5.1. **Dual Method.** We can find a facial reduction matrix V by solving the following optimization problem

$$\begin{aligned} \text{(Phase-I)} \quad \inf \quad & -\ln(\det(Y)) \\ & \langle Y, A_i \rangle = 0, \quad i \in \{1, \ldots, k\}. \end{aligned} \tag{27}$$

In the above, $-\ln(\det) : \mathbb{S}^n \to \mathbb{R} \cup \{+\infty\}$ is defined as

$$-\ln(\det(Y)) := \begin{cases} -\sum_{j=1}^{n} \ln(\lambda_j(Y)), & \text{if } Y \in \mathbb{S}^n_{++}, \\ +\infty & \text{o.w.} \end{cases}$$

Indeed, problem (27) may be unbounded and generally may not have an optimal solution. However, whenever (27) has a nontrivial feasible solution Y, we expect a robust interior-point algorithm to generate iterates $Y^{(l)} \in \mathbb{S}^n_{++}$, approximately satisfying the equations in (27) such that $Y^{(l)}$ approach maximum rank positive semidefinite solutions of (27). Therefore, using such $Y^{(l)}$ we can approximately identify the minimal face of $\mathbb{S}^n_+$ which contains all optimal solutions of QRE problem (26).

Let $Y^* \in \mathbb{S}^n_+$ be a solution of (27), then we can show that the columns of V can be chosen as an orthonormal basis for the null space of Y^*, since we have

$$Y^* \left(\sum_{i=1}^{k} x_i A_i \right) = 0, \quad \forall x. \tag{28}$$

Note however that due to the fact that strict complementarity can fail for SDPs, the complementary face identified by Y^* may be a strict super-set of the minimal face for (26) (and one might need to do a full facial reduction to obtain a description of the minimal face for (26) which in the worst case may require $(n-1)$ recursive applications of the above process, see a family of examples in [45]-Page 43). In our experiments we only performed the above described facial reduction step once.

In defining problem (27), we ignored the primal constraints $\ell \leq x \leq u$. In some applications, it might be worthwhile to include such constraints in deriving the corresponding Phase-I problem such that the underlying cone is not $\mathbb{S}^n_+$ but $\mathbb{S}^n_+ \oplus \mathbb{R}^k_+ \oplus \mathbb{R}^k_+$.

5.2. **Primal Method.** We can use the following phase-I problem with the same feasible region as the QRE problem:

$$\begin{aligned} \text{(Phase-I)} \quad \inf \quad & -\ln(\det(\textstyle\sum_{i=1}^{k} x_i A_i)) \\ & \ell \leq x \leq u. \end{aligned} \tag{29}$$

As we mentioned in Subsection 5.1, here too, a robust interior-point algorithm generates a sequence of vectors $x^{(l)}$ such that $\ell \leq x^{(l)} \leq u$, $\sum_{i=1}^{k} x_i^{(l)} A_i \in \mathbb{S}^n_{++}$, and $\sum_{i=1}^{k} x_i^{(l)} A_i$ approach the relative interior of the minimal face of $\mathbb{S}^n_+$ containing all optimal solutions of

the QRE problem (26). Additionally, solving (29) gives us a feasible solution that can be given to the QRE solver as an initial point.

One important feature of Phase-I problems (27) and (29) is that both problems are minimizing a s.c. barrier, which typically is a very well-behaved convex optimization problem with local quadratic convergence ([36]). Additionally, both problems can be reformulated as SDPs, which are still more robust and faster compared to the QRE problem. Since the current version of DDS does not support minimizing a s.c. barrier (although the ingredients of such an algorithm are already present in the DDS code), we use the SDP reformulations for the numerical results in Subsection 8.3. The numerical results confirm that the two-phase approach can significantly improve the conditioning and speed of QRE problems.

In the following section, we propose similar two-phase methods designed for calculating the rate of QKD channels. In the numerical result section, we show that two-phase methods can significantly improve the size and condition of the QRE problems, in general and also in the context of QKD channel rate calculations.

6. Quantum Key Distribution Rate

One application of minimizing qre function is calculating the rate of quantum key distribution (QKD) channels. QKD is a secure communication method between two parties involving components of quantum mechanics ([48]). The security of the QKD channel depends on the exact calculation of its key rate. There are different protocols for QKD and for many of them, the main non-trivial component of calculating the key rate is an optimization problem of the form:

$$\begin{aligned} \min \quad & qre(\mathcal{G}(\rho), \mathcal{Z}(\mathcal{G}(\rho))) \\ \text{s.t.} \quad & A(\rho) = b, \\ & \rho \succeq 0, \end{aligned} \tag{30}$$

where A is a linear map on Hermitian matrices and $\mathcal{G}$ and $\mathcal{Z}$ are Kraus operators. The Linear map $\mathcal{G} : \mathbb{H}^n \to \mathbb{H}^k$ is defined as

$$\mathcal{G}(\rho) := \sum_{j=1}^{n_g} K_j \rho K_j^\dagger, \tag{31}$$

where $K_j \in \mathbb{C}^{k\times n}$ and $\sum_{j=1}^{n_g} K_j K_j^\dagger \preceq I$, and the self-adjoint linear map $\mathcal{Z} : \mathbb{H}^k \to \mathbb{H}^k$ is defined as

$$\mathcal{Z}(\delta) := \sum_{j=1}^{n_z} Z_j \delta Z_j, \tag{32}$$

where $Z_j = Z_j^2 = Z_j^\dagger \in \mathbb{H}^k_+$ and $\sum_{j=1}^{n_z} Z_j = I$. [25] used *facial reduction* ([4]) for calculating the QKD rate. Their approach restricts the feasible region of the problem into a smaller face and reduces the dimension of the matrices, which improves the performance of interior-point methods. Another simplification by [25] is using the special structure of $\mathcal{Z}$ to prove the following equation for every $\delta \succeq I$:

$$\mathrm{Tr}(\delta \ln(\mathcal{Z}(\delta)) = \mathrm{Tr}(\mathcal{Z}(\delta)\ln(\mathcal{Z}(\delta)). \tag{33}$$

This can simplify the QRE function as the difference of two quantum entropy (QE) functions, which makes calculating the gradients and Hessians much easier. Using these techniques, [25] designed an interior-point algorithm specialized just for computing the QKD rate.

6.1. **Two-phase approach.** The analytic facial reduction techniques by [25] can be used with DDS 2.2 as well. Here, we propose a two-phase approach for finding the minimal face of the positive semidefinite cone for the feasible region of the problem. A rationale is that solving QRE optimization is much costlier than solving SDPs. If we can use a Phase-I SDP to find a better-conditioned formulation for the QRE optimization problem, the overall cost will be lower. Our Phase-I SDP is based on the following lemma:

Lemma 6.1. *Consider the spectrahedron defined by the following constraints:*

$$\begin{aligned} &\langle A_i, \rho\rangle = b_i, \quad i = \{1,\dots,m\} \\ &\rho \succeq 0. \end{aligned} \tag{34}$$

Let $y \in \mathbb{R}^m$ be such that

$$\begin{aligned} &Y := \sum_{i=1}^{m} y_i A_i \succeq 0 \\ &y^\top b = 0. \end{aligned} \tag{35}$$

Then, for every ρ in the spectrahedron defined by (34)*, we have $\rho Y = 0$.*

Indeed, as in Section 5, we want a maximum rank solution to the system (35). Even then, this will correspond to a single iteration of full facial reduction. Consider a $Y \in \mathbb{S}^n_+$ from the lemma that has $\bar{n}$ zero eigenvalues with spectral decomposition

$$Y = [U \quad V]\text{Diag}(\lambda_1, \ldots, \lambda_{n-\bar{n}}, 0, \ldots, 0)[U \quad V]^\top.$$

Then, $\rho Y = 0$ and $\rho \succeq 0$ imply that $\rho = V\bar{\rho}V^\top$ for some $\bar{\rho} \in \mathbb{S}^{\bar{n}}_+$, and the feasible region can be equivalently written as:

$$\begin{aligned} &\langle V^\top A_i V, \bar{\rho}\rangle = b_i, \quad i \in \{1, \ldots, m\} \\ &\bar{\rho} \succeq 0, \end{aligned} \tag{36}$$

where the size of the $\bar{\rho}$ matrix was reduced to $\bar{n}$. For Phase-I of the optimization process (Dual method of Section 5), we can approximately solve the following problem:

$$\begin{aligned} \inf \quad & -\ln(\det(Y)) \\ & Y := \sum_{i=1}^m y_i A_i \succeq 0 \\ & y^\top b = 0. \end{aligned} \tag{37}$$

The effect of using phase-I on the three groups of the QKD problems are shown in Table 7. The main bottleneck in the speed of the code is the dimension of $\mathcal{G}(\rho)$ and $\mathcal{Z}(\mathcal{G}(\rho))$ as the arguments of *qre*. We can significantly reduce this dimension by the following lemma:

Lemma 6.2. *Consider the Kraus operator $\mathcal{G}$ and assume $\mathcal{Z}(\mathcal{G}(I))$ has $\bar{n}$ non-zero eigenvalues with the spectral decomposition*

$$\mathcal{Z}(\mathcal{G}(I)) = \sum_{i=1}^{n_g}\sum_{j=1}^{n_z} Z_i K_j K_j^\dagger Z_i^\dagger = [U \quad V]\text{Diag}(\lambda_1, \ldots, \lambda_{\bar{n}}, 0, \ldots, 0)[U \quad V]^\dagger. \tag{38}$$

Then, we have

$$qre(\mathcal{G}(\rho), \mathcal{Z}(\mathcal{G}(\rho))) \;=\; qre(U^\dagger\mathcal{G}(\rho)U, U^\dagger\mathcal{Z}(\mathcal{G}(\rho))U). \tag{39}$$

Proof. Proof. Note that using (33), we have

$$\begin{aligned} qre(\mathcal{G}(\rho), \mathcal{Z}(\mathcal{G}(\rho))) &= \text{Tr}G(\rho)\ln(G(\rho)) - \text{Tr}\mathcal{Z}(\mathcal{G}(\rho))\ln(\mathcal{Z}(\mathcal{G}(\rho))) \\ &= \text{Tr}(F(\mathcal{Z}(\mathcal{G}(\rho)))) - \text{Tr}(F(U^\dagger\mathcal{Z}(\mathcal{G}(\rho))U)), \end{aligned} \tag{40}$$

where F is the matrix extension of $f(x) := x\ln(x)$. Therefore, to show (39), it suffices to show that $U^\dagger \mathcal{G}(\rho)U$ has the same non-zero eigenvalues as $\mathcal{G}(\rho)$ and $U^\dagger \mathcal{Z}(\mathcal{G}(\rho))U$ has the same non-zero eigenvalues as $\mathcal{Z}(\mathcal{G}(\rho))$. Consider the columns of $V = [v_1 \quad \dots \quad v_{n-\bar{n}}]$. We claim that

$$K_j^\dagger Z_i^\dagger v_t = 0, \quad j \in \{1,\dots,n_g\}, i \in \{1,\dots,n_z\}, t \in \{1,\dots,n-\bar{n}\}$$

$$K_j^\dagger v_t = 0, \quad j \in \{1,\dots,n_g\}, t \in \{1,\dots,n-\bar{n}\}. \tag{41}$$

For the first equation, note that for each $t \in \{1,\dots,n-\bar{n}\}$ we have

$$\begin{aligned} 0 &= v_t^\dagger \mathcal{Z}(\mathcal{G}(I))v_t = \sum_{i=1}^{n_g}\sum_{j=1}^{n_z} v_t^\dagger Z_i K_j K_j^\dagger Z_i^\dagger v_t \\ &= \sum_{i=1}^{n_g}\sum_{j=1}^{n_z} \|K_j^\dagger Z_i^\dagger v_t\|^2, \end{aligned} \tag{42}$$

which implies the first equation in (41). For the second equation, we can use the first equation and the fact that $\sum_{i=1}^{n_z} Z_i = I$: For each fixed j and t, we can add the the equations for all $i \in \{1,\dots,n_z\}$. Equation (41) is important since it shows that for any ρ, $\mathcal{G}(\rho)$ and $\mathcal{Z}(\mathcal{G}(\rho))$ have the columns of V in their null space. Therefore, the range of U contains the ranges of $\mathcal{G}(\rho)$ and $\mathcal{Z}(\mathcal{G}(\rho))$ for any matrix ρ. This implies the statement of the lemma. Specifically, if γ is a non-zero eigenvalue of $\mathcal{Z}(\mathcal{G}(\rho))$ with eigenvector w, there exists $\bar{w}$ such that $w = U\bar{w}$, then

$$U^\dagger \mathcal{Z}(\mathcal{G}(\rho))U\bar{w} = U^\dagger \mathcal{Z}(\mathcal{G}(\rho))w = U^\dagger(\gamma w) = \gamma U^\dagger w.$$

□

In Subsection 8.5, we use the methods we developed here to calculate the QKD rate for multiple protocols.

7. Handling Complex Matrices

For some applications, including key rate calculations for QKD channels, we need to handle complex Hermitian matrices. For software packages such as DDS that only accept real symmetric matrices, we need an equivalent formula for $qre(X,Y)$ based on the real and imaginary parts of X and Y. For a unitary matrix $U = U_r + \iota U_i$ (where $\iota = \sqrt{-1}$),

we have

$$UU^* = U^*U = I \Leftrightarrow \begin{cases} U_r U_r^\top + U_i U_i^\top = I \\ -U_r U_i^\top + U_i U_r^\top = 0 \end{cases}. \tag{43}$$

For any complex n-by-n matrix $X = X_r + \iota X_i$, we define a $2n$-by-$2n$ matrix $\bar{X}$ as

$$\bar{X} = \begin{bmatrix} X_r & -X_i \\ X_i & X_r \end{bmatrix}.$$

If X is Hermitian, then $\bar{X}$ is symmetric. By using (43), we can show that if U is a unitary matrix, then $\bar{U}$ is also a real unitary matrix. We can also show that if $X = UDU^*$ is a spectral decomposition for X, then

$$\bar{X} = \bar{U}\mathrm{Diag}(D, D)\bar{U}^\top,$$

is a spectral decomposition for $\bar{X}$. This implies that for every function $f : \mathbb{R} \to \mathbb{R} \cup \{+\infty\}$ we have

$$\mathrm{Tr}(F(\bar{X})) = 2\mathrm{Tr}(F(X)).$$

Now we can prove the following lemma:

Lemma 7.1. *For two Hermitian matrices $X, Y \in \mathbb{H}^n_+$ we have*

$$qre(\bar{X}, \bar{Y}) = 2qre(X, Y).$$

Proof. Proof. First we show that for two Hermitian matrices $X, W \in \mathbb{H}^n_+$, we have

$$\mathrm{Tr}(\bar{X}\bar{W}) = 2\mathrm{Tr}(XW). \tag{44}$$

Assume that $X = X_r + \iota X_i$ and $W = W_r + \iota W_i$. Then, we have

$$\mathrm{Tr}(XY) = \mathrm{Tr}(X_r W_r - X_i W_i + \iota(X_r W_i + X_i W_r)) = \mathrm{Tr}(X_r W_r - X_i W_i),$$

where for the last equation we used the fact that $\mathrm{Tr}(XY)$ is a real number. Then, we have

$$\mathrm{Tr}(\bar{X}\bar{Y}) = \mathrm{Tr}\left(\begin{bmatrix} X_r & -X_i \\ X_i & X_r \end{bmatrix}\begin{bmatrix} W_r & -W_i \\ W_i & W_r \end{bmatrix}\right) = 2\mathrm{Tr}(X_r W_r - X_i W_i).$$

The last two equations confirm (44). To complete the proof, we claim that for every function F, we have

$$\overline{F(X)} = F(\bar{X}). \tag{45}$$

Assume that the spectral decomposition of X is $X = UDU^*$.

$$\begin{aligned} F(X) = UF(D)U^* &= (U_r + \iota U_i)F(D)(U_r^\top - \iota U_i^\top) \\ &= U_rF(D)U_r^\top + U_iF(D)U_i^\top + \iota(-U_rF(D)U_i^\top + U_iF(D)U_r^\top). \end{aligned}$$

Therefore, we have

$$\overline{F(X)} = \begin{bmatrix} U_rF(D)U_r^\top + U_iF(D)U_i^\top & U_rF(D)U_i^\top - U_iF(D)U_r^\top \\ -U_rF(D)U_i^\top + U_iF(D)U_r^\top & U_rF(D)U_r^\top + U_iF(D)U_i^\top \end{bmatrix}.$$

The spectral decomposition of $\bar{X}$ is $\bar{X} = \bar{U}\mathrm{Diag}(D, D)\bar{U}^\top$. Therefore, $F(\bar{X}) = \bar{U}\mathrm{Diag}(F(D), F(D))\bar{U}^\top$. By expanding this term, we can confirm that (45) holds. Now, (44) and (45) imply the result of the lemma. If we define $F(X) := \ln(X)$, then

$$\begin{aligned} qre(\bar{X}, \bar{Y}) &= \mathrm{Tr}(\bar{X}F(\bar{X})) - \mathrm{Tr}(\bar{X}F(\bar{Y})) \\ &= \mathrm{Tr}(\bar{X}\overline{F(X)}) - \mathrm{Tr}(\bar{X}\overline{F(Y)}), \qquad \text{by using (45)} \\ &= 2\mathrm{Tr}XF(X) - 2\mathrm{Tr}(XF(Y)), \qquad \text{by using (44)} \\ &= 2qre(X, Y). \end{aligned}$$

□

8. Numerical Results

The techniques designed here have been used to improve the performance of the newest version of the software package DDS (namely DDS 2.2), which can be downloaded from [28].

DDS accepts every combination of the following function/set constraints: (1) symmetric cones (LP, SOCP, and SDP); (2) quadratic constraints that are SOCP representable; (3) direct sums of an arbitrary collection of 2-dimensional convex sets defined as the epigraphs of univariate convex functions (including as special cases geometric programming and entropy programming); (4) generalized Koecher (power) cone; (5) epigraphs of matrix norms (including as a special case minimization of nuclear norm over a linear subspace); (6) vector relative entropy; (7) epigraphs of quantum entropy and quantum relative entropy; and (8) constraints involving hyperbolic polynomials. The command in MATLAB that calls DDS has the following form (see [28])

(46) `[x,y,info]=DDS(c,A,b,cons,OPTIONS).`

Input Arguments:
`cons`: A cell array that contains the information about the type of constraints.
`c,A,b`: Input data for DDS: A is the coefficient matrix, c is the objective vector, b is the shift vector.
`OPTIONS` (optional): An array which contains information about the tolerance and initial points.

Output Arguments:
`x`: Primal point.
`y`: Dual point which is a cell array.
`info`: A structure array containing performance information such as `info.time`, which returns the CPU time for solving the problem.

In this section, we present several numerical examples of running DDS 2.2 for QRE programming. We performed computational experiments using the software MATLAB R2022a, on a 1.7 GHz 12th Gen Intel Core i7 personal computer with 32GB of memory. All the numerical results in this section are by using the default settings of DDS, including the tolerance of $tol = 10^{-8}$. Some of the examples in this section are included in a developing work by the authors to create a library for multiple classes of modern convex optimization problems ([29]).

8.1. **Nearest correlation matrix.** The nearest correlation matrix in the classical sense has been heavily studied for years, for example by [22, 24]. Here, we are interested in the nearest correlation matrix in the *quantum sense*, which was introduced in the example folder of CVXQUAD ([13]) and then adopted by Hypatia ([7]) for numerical experiments. For a fixed matrix $M \in \mathbb{S}^n_+$, the nearest correlation matrix in the quantum sense is defined as a matrix Y_M with all diagonals equal to 1 that minimizes $qre(M, Y)$. In other words:

$$\begin{aligned} Y_M = \text{argmin} \quad & qre(M, Y) \\ & Y_{ii} = 1, \qquad i \in \{1, \ldots, n\}. \end{aligned} \tag{47}$$

To make a comparison with Hypatia which uses the exact formulation for the Hessian, we consider a fixed matrix $M \in \mathbb{S}^n$ and change n to see how DDS 2.2 and Hypatia scale in this problem. For the numerical experiments, we assume that Y is a tridiagonal matrix with all the diagonals equal to one. We consider two cases for M: 1) $M = 2I$, and 2) M is a random positive definite matrix constructed as

$$M := M_0 M_0^\top / \|\text{diag}(M_0 M_0^\top)\|_\infty, \tag{48}$$

where $M_0 :=$ `rand(n)` is an n-by-n matrix of uniformly distributed random numbers over $(0, 1)$. Table 1 shows the iterations and time that both DDS 2.2 and Hypatia take to solve the problem for different values of n. For the random M cases, the results are averaged over 10 instances. As can be seen, the running time explodes fast by increasing the dimension

TABLE 1. Finding the nearest correlation matrix in the quantum sense using DDS 2.2 and Hypatia. Times are in seconds.

	$M = 2I$		Random M (average)	
n	DDS Itr/time	Hypatia Itr/time	DDS Itr/time	Hypatia Itr/time
25	11/ 0.8	15 / 6.6	30/ 5	19/ 4
50	14/ 3.6	15/ 43	37/ 20	27 / 29
75	17/ 13.7	17/ 248	51 / 65	33 / 152
100	19/ 32	18/ 900	58 / 168	40 / 829
125	21/ 50.3	20/ 1993	62 / 375	43 / 2701
150	22/ 92.53	20 / 5627	66 / 693	46 / 6079
175	24/ 139.8	time $> 10^4$	71 / 1184	time $> 10^4$
200	26/ 237	time $> 10^4$	75 / 1760	time $> 10^4$
250	30/ 550	time $> 10^4$	78 / 3501	time $> 10^4$
300	32/ 1080	time $> 10^4$	80 / 6980	time $> 10^4$

of the matrices for Hypatia, where for DDS 2.2, the increase rate is more reasonable due to the techniques used in the paper. For more numerical examples, we use the matrices introduced in [24] for classical correlation matrix problem defined as

$$M_0 := \begin{bmatrix} A & Y \\ Y^\top & B \end{bmatrix}, \tag{49}$$

where $A \in \mathbb{S}^m_{++}$ is a random correlation matrix, $B \in \mathbb{S}^n$ is a random matrix with uniformly distributed numbers and all 1 diagonal, and Y is an m-by-n random matrix with uniformly distributed numbers. Since for the quantum nearest correlation matrix M needs to be positive definite, we define M using M_0 and equation (48).

Table 2 shows the iterations and time that both DDS 2.2 and Hypatia take to solve the problem for different values of (m, n). Note that for the largest instances reported in Table 2, the arguments of the *qre* function are 400-by-400 matrices.

8.2. **QRE with other type of convex constraints.** DDS is a software package for convex optimization which accepts a combination of multiple conic and non-conic constraints

TABLE 2. Finding the nearest correlation matrix in the quantum sense for M defined by (49) and (48) using DDS 2.2 and Hypatia. Times are in seconds.

(m,n)	DDS Itr/time	Hypatia Itr/time
(25,25)	31/ 16	19 / 25
(40,40)	37/ 52.8	24/ 437
(40,60)	40/ 134	24/ 980
(60,90)	49/ 305	time $> 10^4$
(100,100)	57/ 1033	time $> 10^4$
(200,100)	64/ 4111	time $> 10^4$
(200,200)	72/ 9660	time $> 10^4$

([28]). Considering QRE programming, DDS lets us solve problems with QRE constraints combined with several other constraints. As far as we know, DDS is the only available software that can solve QRE problems of these sizes combined with other types of constraints. Moreover, DDS is the only available code to handle some types of constraints such as the ones involving hyperbolic polynomials. Preliminary results of QRE programming was reported for DDS 2.1 ([27]). To compare DDS 2.1 and 2.2 for QRE programming, we run the same table in ([27]) for DDS 2.1 and re-run it for DDS 2.2 ([28]). The results are given in Table 3. We also added the results of using CVXQUAD created by [14], which uses SDP approximation for the QRE programming. As can be seen, CVXQUAD does not scale well, and we have size error even for problem instances with 20-by-20 matrices. Due to the major improvements in DDS 2.2 which we are reporting here, we can now solve

TABLE 3. Results for problems involving Quantum Relative Entropy using DDS 2.1, DDS 2.2, and CVXQUAD (with SDPT3 as the solver)

Problem	size of A	Itr/time(sec) DDS 2.1	Itr/time(sec) DDS 2.2	Itr/time(sec) CVXQUAD
QuanReEntr-6	$73 * 13$	9/ 1.0	12/ 0.8	20/ 1.7
QuanReEntr-10	$201 * 21$	12/ 11.2	12/ 1.2	25/ 95
QuanReEntr-20	$801 * 41$	15/ 34.4	15/ 1.2	Size error
QuanReEntr-LP-6	$79 * 13$	29/ 1.7	25/ 0.7	24/ 4.8
QuanReEntr-LP-6-infea	$79 * 13$	30/ 1.7	28/ 0.8	28/ 3.8
QuanReEntr-LP-10	$101 * 21$	27/ 4.6	27/ 1.3	38 / 678.9

much larger instances. Consider an optimization problem of the form

$$\begin{array}{lll} \min & & qre\left(A_0+\sum_{i=1}^k x_iA_i, B_0+\sum_{i=1}^k x_iB_i\right) \\ & (I) & x \geq b_L, \\ & (II) & \|x-b_N\|_p \leq \alpha, \\ & (III) & p(x+b_H) \geq 0. \end{array} \tag{50}$$

For the created examples, A_i and B_i, $i \in \{1,\ldots,k\}$, are sparse 0,1 random symmetric matrices. Table 4 shows the results of running DDS 2.2 on some instances of the form (50). QRE-LP problems only have (I) as the constraint. QRE-LP-POW3 and QRE-LP-SOCP problems have (I)-(II) as constraints, with respectively $p = 3$ and $p = 2$. The problems with infeas in the name are infeasible. Exploiting duality in an efficient way makes DDS robust in detecting the infeasibility of the problems ([31, 27]). The problems QRE-Vamos has (III) as the constraint where p is a hyperbolic polynomial created by Vamos-like matroids as explained in [27]. QRE-KL problems are of the form:

$$\begin{array}{ll} \min & qre\left(A_0+\sum_{i=1}^k x_iA_i, B_0+\sum_{i=1}^k y_iB_i\right) \\ & KL(x,y) \leq \gamma, \end{array} \tag{51}$$

where KL is defined in (1).

8.3. **Two-phase methods for QRE programming.** We proposed two-phase methods for QRE programming in Section 5. For numerical experiments, we have synthesized some problems by fixing r and n, and choosing a n-by-r matrix V which is all zero except the main diagonal is all ones. Let $E_i \in \mathbb{S}^r$ be a matrix of all zeros except a 1 on the ith diagonal entry. Consider problem (26) where $M := I$ and the linear constraints are $x_i \leq 1$ for $i \in \{1,\ldots,k\}$. We define

$$A_i := VE_iV^\top, \quad i \in \{1,\ldots,k\}.$$

Table 5 shows the results of solving problem (26) using DDS 2.2 for different values of r and n. We explained that problems (27) and (29) are minimizing a s.c. barrier which can be done very efficiently. Since the current version of DDS does not support this, we use the SDP reformulations for Phase-I. As can be seen, the reformulated QRE after phase-I is not only smaller in size, but the much fewer number of iterations shows that it is more well-conditioned. The overall running time of the two-phase method is smaller than the 1-phase one, and the gap grows by increasing n. Note that both Phase-I SDP and Phase-II QRE programming are solved by using DDS.

TABLE 4. Results for problems involving Quantum Relative Entropy using DDS 2.2. The types of the constraints are included in the name of the problem. The number is the size of the matrices in the QRE constraint. Vamos stands for hyperbolic polynomials created by Vamos-like matroids.

Problem	size of `cell2mat(A)` in (46)	Itr/time(sec) DDS 2.2
QRE-LP-100	20101×101	17/ 42
QRE-LP-200-infeas	80201×201	50/ 966
QRE-LP-Pow3-20	842×21	37/ 6.7
QRE-LP-Pow3-20-infeas	842×21	25/ 3.8
QRE-LP-Pow3-100	20201×101	56/ 170
QRE-LP-Pow3-100-infeas	20201×101	44/ 74
QRE-LP-SOCP-20	842×21	39/ 4.2
QRE-LP-SOCP-20-infeas	842×21	24/ 3.9
QRE-LP-SOCP-100	20202×101	66/ 180
QRE-LP-SOCP-100-infeas	20202×101	28/ 90
QRE-LP-SOCP-200-infeas	80402×201	36/ 544
QRE-LP-SOCP-200	80402×201	103/ 1191
QRE-Vamos-20	811×6	22/ 3.6
QRE-Vamos-20-infeas	811×6	32/ 7.8
QRE-Vamos1-100	20011×6	27/ 48
QRE-Vamos2-100	20011×6	44/ 129
QRE-KL-100	20202×201	49/ 109
QRE-KL-100-infeas	20202×201	22/ 37
QRE-KL-200	80402×401	76/ 1153
QRE-KL-200-infeas	80402×401	25/ 278

8.4. **Symmetric QRE.** Handing symmetric QRE constraints using QRE ones discussed in Section 4. Consider the following two optimization problems:

$$\begin{array}{ll} \min & qre\left(I + \sum_{i=1}^{k} x_i A_i, I + \sum_{i=1}^{k} x_i B_i\right) \\ & x \geq \ell, \end{array} \qquad \begin{array}{ll} \min & sqre\left(I + \sum_{i=1}^{k} x_i A_i, I + \sum_{i=1}^{k} x_i B_i\right) \\ & x \geq \ell, \end{array}$$

where $A_i \in \mathbb{S}^n$ and $B_i \in \mathbb{S}^n$, $i \in \{1, \ldots, k\}$, are sparse 0-1 random matrices (each matrix is all zero except for two off-diagonal entries). $x = 0$ is a feasible solution for both problems. We want to compare the number of iterations and running time of solving these problems by changing n, the size of matrices where we choose $k = n$. Let us choose $\ell = -2\mathbb{1}$, where $\mathbb{1}$ is the vector of all ones. Table 6 shows the results of the iteration count and running

TABLE 5. The effect of the primal and dual two-phase approachs on the running time and condition of the QRE problem. For the primal two-phase method, a feasible initial point is also given as input to DDS.

n	r	Dual two-phase (27)		Primal two-phase (29)		1-phase method
		Phase-I time	iter/time	Phase-I time	iter/time	iter/time
25	5	0.96	13 / 0.39	0.25	12/ 0.35	90/ 5.1
50	5	0.91	12 / 0.45	0.3	11 / 0.3	108/ 29
100	5	5.8	12 / 0.42	0.4	11 / 0.4	172/ 178
200	5	70	13 / 0.44	0.7	10/ 0.3	226/ 1527
300	5	654	12/ 0.74	2.9	11/ 0.44	313/ 8560
25	10	1.1	14 / 0.6	0.3	14/ 0.4	78/ 4.0
50	10	0.9	14 / 0.63	0.38	13/0.53	97/ 26.6
100	10	5.3	15 / 0.68	0.49	13/0.55	164/ 180
200	10	72	15 / 0.68	0.78	14/ 0.49	214/ 1695
300	10	736	15 / 0.72	4.5	13/ 0.57	304/ 8663
25	20	0.7	12 / 1	0.3	11/ 0.9	48/ 3.2
50	45	0.87	15 / 4.3	0.32	13/ 3.4	50/ 14
100	95	4.5	25 / 46	0.56	16/ 23	54/ 81.4
200	195	63	32 / 342	1.07	23/ 188	50/ 478
300	295	621	38 / 1620	2.04	28/ 785	52/ 1412

time for both problems using DDS 2.2. As can be seen in Table 6, the ratio of iteration counts are growing from 1 to 1.53, while the ratio of running times are growing from 1.7 to 2.45.

8.5. **Quantum key distribution rate.** We developed a two-phase approach for calculating the QKD rate in Section 6. In this section, we consider some QKD protocols such as some variants of the Bennett-Brassard 1984 (BB84) protocol ([1]): entanglement-based (ebBB84), prepare-and-measure (pmBB84), and measurement-device-independent (mdiBB84) ([34]). OpenQKDSecurity is a platform for numerical key rate calculation of QKD ([47]), where several examples for different regimes can be created. One protocol for QKD is Entanglement-Based BB84. The parameters of the problem $\mathcal{G}$, $\mathcal{Z}$, and Γ are calculated based on the probability of performing measurement in one of the two possible basis p_z, and the error rate e. As we use natural logarithm in DDS, the key rate R for this

TABLE 6. Comparing SQRE and QRE using DDS 2.2

n	Itr/time(sec) QRE	Itr/time(sec) SQRE
10	9 / 0.3	9 / 0.5
25	12 / 1.7	15 / 3.3
50	13 / 4.6	13 / 8.1
100	14 / 28	15 / 47
150	14 / 81	16 / 159
200	14 / 166	17 / 333
250	15 / 344	23 / 858
300	15 / 698	23 / 1712

protocol is calculated by the formula

$$R = \frac{p}{\ln(2)} - \delta_{EC},$$

where p is the optimal value of (30) and δ_{EC} is a constant caused by performing error-correction.

By applying the phase-I optimization discussed above and Lemma 6.2, not only do we significantly reduce the size of the involved matrices, but we also improve the condition of the problem. Some examples are shown in Table 7. Without using Phase-I, the problem is ill-conditioned and DDS cannot achieve the desired accuracy.

TABLE 7. The effect of phase-I and 6.2 on reducing the size of ρ

protocol (p_z, e)	(n,k)	$(\bar{n},\bar{k})$	Phase-I and Lemma 6.2		Just Lemma 6.2
			Phase-I time	iter/time	iter/time
pmBB84 (0.5, .09)	(32,8)	(8,4)	1	19/ 2.7	81/14
pmBB84 (0.9, .09)	(32,8)	(8,4)	0.9	19/ 2.7	ill-conditioned
mdiBB84 (0.5, .09)	(96,48)	(8,12)	1.4	25/4.7	ill-conditioned
mdiBB84 (0.9, .09)	(96,48)	(8,12)	1.4	25/4.2	ill-conditioned

For the protocols eeBB84, pmBB84, and mdiBB84, the QRE program is setup by using two parameters: p_z is the probability of choosing the Z basis, and e is the observed error rate. The iteration counts and running times of using DDS 2.2 for solving the QRE optimization problems for these three protocols are given in Tables 8, 9, and 10.

TABLE 8. Numerical Report for ebBB84 Instances.

Protocol	Parameters (p_z, e)	Size	Iter	Time
ebBB84	(0.5, .01)	(16,4)	23	0.8
ebBB84	(0.5, .03)	(16,4)	20	0.7
ebBB84	(0.5, .05)	(16,4)	18	0.65
ebBB84	(0.5, .07)	(16,4)	17	0.57
ebBB84	(0.5, .09)	(16,4)	14	0.5
ebBB84	(0.7, .01)	(16,4)	21	0.8
ebBB84	(0.7, .03)	(16,4)	21	0.74
ebBB84	(0.7, .05)	(16,4)	17	0.65
ebBB84	(0.7, .07)	(16,4)	16	0.6
ebBB84	(0.7, .09)	(16,4)	17	0.65
ebBB84	(0.9, .01)	(16,4)	22	0.8
ebBB84	(0.9, .03)	(16,4)	21	0.7
ebBB84	(0.9, .05)	(16,4)	17	0.65
ebBB84	(0.9, .07)	(16,4)	17	0.65
ebBB84	(0.9, .09)	(16,4)	17	1.7

9. Conclusion

We developed novel numerical techniques to enhance the performance of interior-point (IP) methods which use the optimal self-concordant (s.c.) barrier function in (3). Extensive numerical results demonstrate that DDS 2.2, which incorporates these techniques, can effectively solve significantly larger instances compared to its predecessor, DDS 2.1, and Hypatia. The two-phase approaches proposed in this paper warrants further computational investigation in future research. The Phase-I problem can be tailored in various ways to modify the problem to help speed up Phase-II. Currently, the primary bottleneck in the speed of DDS 2.2 for QRE programming lies in calculating the matrix S defined in (19). A significantly more efficient and numerically stable algorithm to implicitly or explicitly compute the matrix could significantly accelerate DDS and other IP solvers for QRE programming. Additionally, exploring the duality setup for the QRE cone presents an intriguing open question. A numerically robust characterization of the dual cone or the LF conjugate of the s.c. barrier could be leveraged in DDS and other solvers to further enhance their performance.

TABLE 9. Numerical Report for pmBB84 Instances.

Protocol	Parameters (p_z, e)	Size	Iter	Time
pmBB84	(0.5, .01)	(32,8)	24	1.8
pmBB84	(0.5, .03)	(32,8)	22	1.4
pmBB84	(0.5, .05)	(32,8)	21	1.3
pmBB84	(0.5, .07)	(32,8)	18	1.3
pmBB84	(0.5, .09)	(32,8)	17	1.1
pmBB84	(0.7, .01)	(32,8)	24	1.6
pmBB84	(0.7, .03)	(32,8)	22	1.4
pmBB84	(0.7, .05)	(32,8)	20	1.4
pmBB84	(0.7, .07)	(32,8)	18	1.3
pmBB84	(0.7, .09)	(32,8)	18	1.1
pmBB84	(0.9, .01)	(32,8)	25	1.6
pmBB84	(0.9, .03)	(32,8)	23	1.5
pmBB84	(0.9, .05)	(32,8)	21	1.1
pmBB84	(0.9, .07)	(32,8)	20	1.1
pmBB84	(0.9, .09)	(32,8)	21	1.4

REFERENCES

[1] C. H. BENNETT AND G. BRASSARD, *Quantum cryptography: Public key distribution and coin tossing*, Theoretical computer science, 560 (2014), pp. 7–11.

[2] D. BERTSIMAS, R. CORY-WRIGHT, AND J. PAUPHILET, *A new perspective on low-rank optimization*, Mathematical Programming, (2023), pp. 1–46.

[3] R. G. BLAND, D. GOLDFARB, AND M. J. TODD, *The ellipsoid method: a survey*, Oper. Res., 29 (1981), pp. 1039–1091.

[4] J. M. BORWEIN AND H. WOLKOWICZ, *Facial reduction for a cone-convex programming problem*, Journal of the Australian Mathematical Society, 30 (1981), pp. 369–380.

[5] A. BOULAND, Y. M. GETACHEW, Y. JIN, A. SIDFORD, AND K. TIAN, *Quantum speedups for zero-sum games via improved dynamic gibbs sampling*, in International Conference on Machine Learning, PMLR, 2023, pp. 2932–2952.

[6] V. CHANDRASEKARAN AND P. SHAH, *Relative entropy optimization and its applications*, Mathematical Programming, 161 (2017), pp. 1–32.

[7] C. COEY, L. KAPELEVICH, AND J. P. VIELMA, *Solving natural conic formulations with hypatia. jl*, INFORMS Journal on Computing, 34 (2022), pp. 2686–2699.

[8] T. M. COVER, *Elements of information theory*, John Wiley & Sons, 1999.

[9] C. DASKALAKIS AND I. PANAGEAS, *Last-iterate convergence: Zero-sum games and constrained min-max optimization*, arXiv preprint arXiv:1807.04252, (2018).

[10] N. DATTA, M.-H. HSIEH, AND M. M. WILDE, *Quantum rate distortion, reverse shannon theorems, and source-channel separation*, IEEE Transactions on Information Theory, 59 (2012), pp. 615–630.

[11] C. DOERSCH, *Tutorial on variational autoencoders*, arXiv preprint arXiv:1606.05908, (2016).

TABLE 10. Numerical Report for mdiBB84 Instances.

Protocol	Parameters (p_z, e)	Size	Iter	Time
mdiBB84	(0.5, .01)	(96,48)	31	3.0
mdiBB84	(0.5, .03)	(96,48)	28	2.4
mdiBB84	(0.5, .05)	(96,48)	28	3.1
mdiBB84	(0.5, .07)	(96,48)	26	2.7
mdiBB84	(0.5, .09)	(96,48)	18	1.6
mdiBB84	(0.7, .01)	(96,48)	29	2.6
mdiBB84	(0.7, .03)	(96,48)	23	2.2
mdiBB84	(0.7, .05)	(96,48)	24	2.4
mdiBB84	(0.7, .07)	(96,48)	27	2.6
mdiBB84	(0.7, .09)	(96,48)	26	2.6
mdiBB84	(0.9, .01)	(96,48)	31	3.2
mdiBB84	(0.9, .03)	(96,48)	28	2.6
mdiBB84	(0.9, .05)	(96,48)	24	2.3
mdiBB84	(0.9, .07)	(96,48)	27	2.5
mdiBB84	(0.9, .09)	(96,48)	26	2.7

[12] H. FAWZI AND J. SAUNDERSON, *Optimal self-concordant barriers for quantum relative entropies*, SIAM J. Optim., 33 (2023), pp. 2858–2884.

[13] H. FAWZI, J. SAUNDERSON, AND P. A. PARRILO, *Semidefinite approximations of the matrix logarithm*, Foundations of Computational Mathematics, (2018). Package cvxquad at `https://github.com/hfawzi/cvxquad`.

[14] H. FAWZI, J. SAUNDERSON, AND P. A. PARRILO, *Semidefinite approximations of the matrix logarithm*, Foundations of Computational Mathematics, 19 (2019), pp. 259–296.

[15] L. FAYBUSOVICH AND C. ZHOU, *Self-concordance and matrix monotonicity with applications to quantum entanglement problems*, Applied Mathematics and Computation, 375 (2020), p. 125071.

[16] ——, *Long-step path-following algorithm for quantum information theory: Some numerical aspects and applications*, Numerical Algebra, Control and Optimization, 12 (2022), pp. 445–467.

[17] I. GOODFELLOW, Y. BENGIO, AND A. COURVILLE, *Deep learning*, MIT press, 2016.

[18] M. GRANT AND S. BOYD, *CVX: Matlab software for disciplined convex programming, version 2.2.* `http://cvxr.com/cvx`, Mar. 2020.

[19] P. E. HART, D. G. STORK, AND R. O. DUDA, *Pattern classification*, Wiley Hoboken, 2000.

[20] K. HE, J. SAUNDERSON, AND H. FAWZI, *Efficient computation of the quantum rate-distortion function*, arXiv preprint arXiv:2309.15919, (2023).

[21] F. HIAI AND D. PETZ, *Introduction to matrix analysis and applications*, Springer Science & Business Media, 2014.

[22] N. J. HIGHAM, *Computing the nearest correlation matrix—a problem from finance*, IMA journal of Numerical Analysis, 22 (2002), pp. 329–343.

[23] ——, *Functions of matrices: theory and computation*, SIAM, 2008.

[24] N. J. HIGHAM, N. STRABIĆ, AND V. ŠEGO, *Restoring definiteness via shrinking, with an application to correlation matrices with a fixed block*, SIAM Rev., 58 (2016), pp. 245–263.

[25] H. Hu, J. Im, J. Lin, N. Lütkenhaus, and H. Wolkowicz, *Robust interior point method for quantum key distribution rate computation*, Quantum, 6 (2022), p. 792.
[26] H. Jeffreys, *The theory of probability*, OuP Oxford, 1998.
[27] M. Karimi and L. Tunçel, *Domain-Driven Solver (DDS) Version 2.1: A MATLAB-based software package for convex optimization problems in domain-driven form*, Math. Program. Comput., 16 (2024), pp. 37–92.
[28] M. Karimi and L. Tunçel, *Efficient Implementation of Interior-Point Methods for Quantum Relative Entropy*, 2024. Available for download at https://github.com/INFORMSJoC/2024.0570.
[29] M. Karimi and L. Tunçel, *Library for Modern Convex Optimization*. `https://github.com/mehdi-karimi-math/Library-for-Modern-Convex-Optimization`, Jan. 2024.
[30] M. Karimi and L. Tunçel, *Primal–dual interior-point methods for domain-driven formulations*, Mathematics of Operations Research, 45 (2020), pp. 591–621.
[31] ——, *Status determination by interior-point methods for convex optimization problems in Domain-Driven form*, Mathematical Programming, 194 (2022), pp. 937–974.
[32] N. Karmarkar, *A new polynomial-time algorithm for linear programming*, Combinatorica, 4 (1984), pp. 373–395.
[33] S. Kullback and R. A. Leibler, *On information and sufficiency*, The annals of mathematical statistics, 22 (1951), pp. 79–86.
[34] H.-K. Lo, M. Curty, and B. Qi, *Measurement-device-independent quantum key distribution*, Physical review letters, 108 (2012), p. 130503.
[35] MOSEK ApS, *The MOSEK optimization toolbox for MATLAB manual. Version 9.0.*, 2019. `http://docs.mosek.com/9.0/toolbox/index.html`.
[36] Y. Nesterov and A. Nemirovski, *Interior-Point Polynomial Algorithms in Convex Programming*, SIAM Series in Applied Mathematics, SIAM: Philadelphia, 1994.
[37] Y. E. Nesterov and M. J. Todd, *Self-scaled barriers and interior-point methods for convex programming*, Mathematics of Operations research, 22 (1997), pp. 1–42.
[38] ——, *Primal-dual interior-point methods for self-scaled cones*, SIAM Journal on optimization, 8 (1998), pp. 324–364.
[39] D. Papp and S. Yi ldiz, *Alfonso: Matlab package for nonsymmetric conic optimization*, INFORMS J. Comput., 34 (2022), pp. 11–19.
[40] V. Scarani, H. Bechmann-Pasquinucci, N. J. Cerf, M. Dušek, N. Lütkenhaus, and M. Peev, *The security of practical quantum key distribution*, Reviews of modern physics, 81 (2009), p. 1301.
[41] M. L. Stone. personal communication, 2022.
[42] J. F. Sturm, *Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones*, Optimization Methods and Software, 11 (1999), pp. 625–653.
[43] K.-C. Toh, M. J. Todd, and R. H. Tütüncü, *SDPT3– a MATLAB software package for semidefinite programming, version 1.3*, Optimization Methods and Software, 11 (1999), pp. 545–581.
[44] L. Tunçel and L. Vandenberghe, *Linear optimization over homogeneous matrix cones*, Acta Numer., 32 (2023), pp. 675–747.
[45] L. Tunçel, *Polyhedral and Semidefinite Programming Methods in Combinatorial Optimization*, American Mathematical Soc., 2010.
[46] T. Van Erven and P. Harremos, *Rényi divergence and Kullback-Leibler divergence*, IEEE Transactions on Information Theory, 60 (2014), pp. 3797–3820.
[47] W. Wang and N. Lütkenhaus, *OpenQKDSecurity platform*. https://github.com/nlutkenhaus/openQKDsecurity, 2021.
[48] A. Winick, N. Lütkenhaus, and P. J. Coles, *Reliable numerical key rates for quantum key distribution*, arXiv preprint arXiv:1710.05511v2, (2018).

[49] F. Xu, X. Ma, Q. Zhang, H.-K. Lo, and J.-W. Pan, *Secure quantum key distribution with realistic devices*, Reviews of Modern Physics, 92 (2020), p. 025002.