Ugrc report

Lakshmiram S, ee22b117

December 2025

Summary of the Problem and Methods

The Underlying Sets

A qubit state is a 2×2 density matrix

$$\mathcal{D}_2 = \{\rho \in C^{2\times 2} : \rho = \rho^\dagger,\ \rho \succeq 0,\ \mathrm{Tr}(\rho) = 1\},$$

ensuring a valid probabilistic interpretation.

Every $\rho \in \mathcal{D}_2$ has a Bloch representation

$$\rho = \tfrac{1}{2}\big(I + \vec{r}\cdot\vec{\sigma}\big), \qquad \vec{r} \in R^3,\ \|\vec{r}\| \le 1,$$

so $\mathcal{D}_2$ is the Bloch ball: pure states satisfy $\|\vec{r}\| = 1$, mixed states $\|\vec{r}\| < 1$.

Product States. For two qubits,

$$\rho_A, \rho_B \in \mathcal{D}_2 \quad \Rightarrow \quad \rho_A \otimes \rho_B \in C^{4\times 4},$$

and the set of product states is

$$S = \{\rho_A \otimes \rho_B : \rho_A, \rho_B \in \mathcal{D}_2\}.$$

Here $\mathcal{D}_2$ is convex but S is not, so S is a nonlinear manifold inside the 15-dimensional space of 4×4 Hermitian trace-1 matrices.

Separable States. Separable two-qubit states form the convex hull

$$\mathrm{conv}(S) = \Big\{\sum_{i=1}^{k} p_i\, \rho_A^{(i)} \otimes \rho_B^{(i)} : p_i \ge 0,\ \sum_i p_i = 1,\ \rho_A^{(i)}, \rho_B^{(i)} \in \mathcal{D}_2\Big\},$$

i.e., classical mixtures of product states. Non-separable states are entangled.

In the 2×2 case, separability is exactly characterized by the PPT condition:

$$\rho \in \mathrm{conv}(S) \quad \Longleftrightarrow \quad \rho^{T_B} \succeq 0.$$

Problem Statement

Let $P \in C^{4\times 4}$ be an unknown two-qubit density matrix

$$P = P^\dagger, \quad P \succeq 0, \quad \mathrm{Tr}(P) = 1.$$

We wish to decide whether $P \in \mathrm{conv}(S)$ (separable). If $P \notin \mathrm{conv}(S)$, we seek a supporting hyperplane separating P from $\mathrm{conv}(S)$, i.e. an entanglement witness.

Instead of an explicit matrix, we have an inner-product oracle

$$\mathcal{O}_P(A) = \mathrm{Tr}(A^\dagger P),$$

returning Hilbert–Schmidt inner products with arbitrary A.

The real vector space of 4×4 Hermitian matrices is 16-dimensional with orthogonal basis

$$\{\sigma_\alpha \otimes \sigma_\beta : \alpha, \beta = 0, 1, 2, 3\}.$$

Oracle queries yield

$$c_{\alpha\beta} = \mathrm{Tr}\big[(\sigma_\alpha \otimes \sigma_\beta)P\big],$$

so

$$P = \frac{1}{4} \sum_{\alpha,\beta=0}^{3} c_{\alpha\beta} \, (\sigma_\alpha \otimes \sigma_\beta),$$

which fully specifies P.

This enables:

- PPT-based separability tests for $\mathrm{conv}(S)$,
- projection of P onto $\mathrm{conv}(S)$ via SDP,
- construction of separating entanglement witnesses.

Membership Test via the PPT Criterion

Deciding separability is hard in general, but for 2×2 systems the Peres–Horodecki (PPT) criterion gives a necessary and sufficient condition.

Partial Transpose. For ρ on $C^2 \otimes C^2$, the partial transpose with respect to subsystem B is

$$(\rho^{T_B})_{ij,k\ell} = \rho_{i\ell,kj},$$

i.e. transpose on the second subsystem only.

PPT Criterion. For 2×2,

$$\rho \in \mathrm{SEP} \quad \Longleftrightarrow \quad \rho^{T_B} \succeq 0,$$

so ρ is separable iff its partial transpose is positive semidefinite.

Practical Membership Test. Given a two-qubit density matrix P, to test $P \in \mathrm{conv}(S)$:

1. Compute P^{T_B}.
2. Check whether P^{T_B} has negative eigenvalues.

If all eigenvalues are nonnegative, P is separable; otherwise it is entangled and lies outside $\mathrm{conv}(S)$.

Projection Onto $\mathrm{conv}(S)$

Once a state P has been identified as lying outside $\mathrm{conv}(S)$, it is natural to quantify "how far" it is from the set of separable states. One principled way to do this is to compute the *orthogonal projection* of P onto $\mathrm{conv}(S)$ with respect to the Frobenius norm.

Due to the equivalence

$$\mathrm{conv}(S) = \mathrm{SEP} = \{X : \; X \succeq 0, \; \mathrm{Tr}(X) = 1, \; X^{T_B} \succeq 0\},$$

the projection problem can be formulated as the semidefinite program (SDP)

$$X^\star = \operatorname*{argmin}_X \; \|X - P\|_F^2 \quad \text{subject to} \quad X \succeq 0, \; X^{T_B} \succeq 0, \; \mathrm{Tr}(X) = 1.$$

This is a convex optimization problem: the objective function is strictly convex, and the feasible region is a closed, convex, spectrahedral set.

Interpretation. The optimizer $X^\star$ is the closest separable state to P. The residual vector

$$P - X^\star$$

acts as the normal vector to the supporting hyperplane that separates P from $\mathrm{conv}(S)$, and is later used to construct entanglement witnesses.

Warm Starting the SDP. Although the above SDP can be solved directly, practical performance can be improved by initializing the optimization with a point that approximately satisfies the constraints. Two useful warm-start strategies are:

- **PSD projection warm start:** Compute the eigen-decomposition $P = U\Lambda U^\dagger$, set $\Lambda_+ = \max(\Lambda, 0)$ (replacing negative eigenvalues with 0), and define
$$X_0 = \frac{U\Lambda_+ U^\dagger}{\mathrm{Tr}(U\Lambda_+ U^\dagger)}.$$
This yields a valid density matrix close to P.

- **PPT projection warm start:** Compute the partial transpose P^{T_B}, filter its negative eigenvalues, and back-transform:
$$Y = (P^{T_B})_+, \qquad X_0 = \frac{Y^{T_B}}{\mathrm{Tr}(Y)}.$$
This directly produces a state satisfying the key constraint $X_0^{T_B} \succeq 0$.

Warm-starting is especially beneficial in high-accuracy computations or in iterative methods where multiple projections are required.

Constructing a Supporting Hyperplane

If P lies outside SEP, a separating (and, at the optimum, supporting) hyperplane is:

$$W = P - X^\star.$$

This Hermitian matrix satisfies:
$$\langle W, X^\star \rangle = \min_{X \in \mathrm{SEP}} \langle W, X \rangle, \qquad \langle W, P \rangle > \langle W, X^\star \rangle,$$
and therefore W acts as an entanglement witness and the equation of the separating hyperplane is given by $Z : <W, Z> = <W, X*>$

Summary of the Full Workflow

1. Reconstruct P from Pauli expectation values (if needed).
2. Check separability via the PPT criterion.
3. If $P \notin$ SEP, compute the SDP projection $X^\star$.
4. Form the separating/supporting hyperplane $W = P - X^\star$.
5. Use W as an entanglement witness satisfying $\mathrm{Tr}(WP) < \mathrm{Tr}(W\rho)$ for all $\rho \in$ SEP.

Generic problem statement

Given a set of points S(not necessarily finite), and a point P, do the following:

1. if P lies inside conv(S) then report that
2. else give a supporting hyperplane that separates P and conv(S).

Algorithm 1 Check if point P lies in $\text{conv}(S)$ by solving a feasibility problem.

Require: Point $P \in R^d$, set $S = \{x_1, x_2, \ldots, x_n\}$
Ensure: **TRUE** if $P \in \text{conv}(S)$, else **FALSE**

1: Define variables $\lambda_1, \lambda_2, \ldots, \lambda_n$
2: Define dummy objective:

$$\min_{\lambda} \quad 0$$

3: Subject to constraints:

1. Convex combination (one equality for each coordinate):

$$\sum_{i=1}^{n} \lambda_i \, x_i[j] = P[j], \quad \forall j = 1, \ldots, d$$

2. Sum-to-one constraint:

$$\sum_{i=1}^{n} \lambda_i = 1$$

3. Non-negativity:

$$\lambda_i \geq 0, \quad \forall i$$

4: Solve the linear program.
5: **if** the LP returns a feasible solution **then**
6: **return TRUE** ▷ P lies in the convex hull
7: **else**
8: **return FALSE** ▷ P lies outside the convex hull
9: **end if**

Abbasov, 2017 (link to paper)

Algorithm 2 Charged Ball Method — projection of point o (origin) onto convex set $X = \{x : f(x) \le 0\}$

Require: A twice continuously differentiable convex function $f : R^n \to R$ with $\nabla f(x) \neq 0$ for $x \in \operatorname{bd} X$, initial guess $x_0 \in X$, $z_0 = 0$, parameters $p_1, p_2 > 0$, initial step $\delta_0 > 0$, tolerances $\varepsilon, \varepsilon_1, \varepsilon_2$.

Ensure: Approximate orthogonal projection $\hat{x} \in \operatorname{bd} X$ of the origin (or given point) onto X.

1: Set $k \leftarrow 1$, $\delta \leftarrow \delta_0$.

2: **while** true **do**

3: // Euler-type predictor (move along velocity z_{k-1})

$$\tilde{x}_{k-1} \;=\; x_{k-1} + \delta\, z_{k-1}.$$

4: // Local orthogonal projection of $\tilde{x}_{k-1}$ onto the boundary $\operatorname{bd} X$

$$x_k \;=\; \tilde{x}_{k-1} \;-\; \frac{\nabla f(\tilde{x}_{k-1})}{\|\nabla f(\tilde{x}_{k-1})\|^2}\; f(\tilde{x}_{k-1}).$$

5: // Compute tangential driving vector $\psi(x_{k-1})$ and correction $\chi(x_{k-1}, z_{k-1})$:

$$\psi(x) \;=\; -\frac{1}{\|x\|^3}x + \frac{\langle x, \nabla f(x)\rangle}{\|x\|^3\|\nabla f(x)\|^2}\nabla f(x),$$

$$\chi(x, z) \;=\; \frac{\langle H(x)z, z\rangle}{\|\nabla f(x)\|^2}\nabla f(x),$$

6: // Velocity update (Euler)

$$z_k \;=\; z_{k-1} + \delta\big(p_1\psi(x_{k-1}) - p_2 z_{k-1} - \chi(x_{k-1}, z_{k-1})\big).$$

7: // Stopping test (tangential component small)

8: **if** $\|\psi(x_k)\| \le \varepsilon$ **then**

9: **return** x_k ▷ approximate projection; algorithm converged

10: **end if**

11: // Relative step control: compute relative move measure

$$r \;:=\; \frac{\|x_k - \tilde{x}_{k-1}\|}{\|x_k - x_{k-1}\|} \;=\; \frac{\|\nabla f(\tilde{x}_{k-1})\|^{-2}\,\|\nabla f(\tilde{x}_{k-1})\, f(\tilde{x}_{k-1})\|}{\|x_k - x_{k-1}\|}.$$

12: **if** $r < \varepsilon_1$ **then**

13: $\delta \leftarrow 2\delta$ ▷ increase step and repeat this iteration

14: **goto** line 3

15: **else if** $r > \varepsilon_2$ **then**

16: $\delta \leftarrow \delta/2$ ▷ decrease step and repeat

17: **goto** line 3

18: **else**

19: $k \leftarrow k + 1$ ▷ accept step and continue

20: **end if**

21: **end while**

Algorithm 3 Face-Walking Projection onto conv(S)

Require: Point set $S = \{s_1, \ldots, s_m\} \subset R^d$, external point p, initial guess x_0 near boundary
Require: tolerances tol, maximum iterations $K_{\max}$
Ensure: Approximate projection x_{proj}, path of visited points, exit information
Build the convex hull $\mathcal{H} = \text{ConvexHull}(S)$
$x \leftarrow x_0$
Find facet (simplex) F of $\mathcal{H}$ that contains x using FindContainingFacet($\mathcal{H}, S, x$)
if F not found **then**
    fail ("no facet found")
end if
Compute orthogonal projection q of x onto the affine hull of F
if $\|x - q\| > \varepsilon$ **then**
    if q lies inside simplex F **then**
        $x \leftarrow q$
    else
        Snap x to nearest vertex of F
    end if
end if
Path $\leftarrow [x]$
reason $\leftarrow$ "max_iters"
final_face $\leftarrow$ None

Algorithm 4 Face-Walking Projection onto conv(S) continued

for $k = 1 \dots K_{\max}$ **do**
  Determine facet F of $\mathcal{H}$ containing x
  if F not found **then**
    reason $\leftarrow$ "no_facet_found"
    break
  end if
  Let vertices of F be $\{v_0, \dots, v_{r-1}\}$ $(r = |F|)$
  Compute barycentric coordinates $\beta(x)$ of x w.r.t. F
  $p_{\text{aff}} \leftarrow \text{ProjAffine}(p, F)$
  Check if p_{aff} lies in simplex F
  if p_{aff} is inside F **then**
    $x \leftarrow p_{\text{aff}}$
    reason $\leftarrow$ "found_projection_on_face"
    final_face $\leftarrow F$
    Path.append(x)
    break
  end if
  $g \leftarrow x - p$ ▷ Gradient of $\frac{1}{2}\|x - p\|^2$
  Build tangent matrix $A = [\, v_1 - v_0, \dots, v_{r-1} - v_0 \,]$ (dimension $d \times (r-1)$)
  if A has zero columns **then** ▷ Face is a vertex
    Check if this vertex is the closest hull vertex to p
    if yes **then**
      reason $\leftarrow$ "projection_is_vertex"
      final_face $\leftarrow F$
      break
    else
      Move x to closest hull vertex and continue
      Path.append(x)
      continue
    end if
  end if
  Compute orthogonal projection $g_{\tan}$ of g onto column space of A
  if $\|g_{\tan}\| \le \text{tol}$ **then**
    reason $\leftarrow$ "stationary_in_face"
    final_face $\leftarrow F$
    break
  end if
  $d_{\text{dir}} \leftarrow -g_{\tan}/\|g_{\tan}\|$ ▷ Descent direction within face
  Solve $Ac = d_{\text{dir}}$ for c
  Compute barycentric direction vector γ:

$$\gamma_0 = -\sum_{i=1}^{r-1} c_i, \qquad \gamma_i = c_i \text{ for } i \ge 1$$

  Initialize $s^{\leftarrow \infty}$
  for each vertex index j of face **do**
    if $\gamma_j < 0$ and $\beta_j(x) > 0$ **then**
      Compute step $s = -\beta_j(x)/\gamma_j$
      if $s > 0$ and $s < s$ **then**
        $s^{\leftarrow s}$
      end if
    end if
  end for
  if $s^{=\infty}$ **then**
    $s^{\leftarrow 10^{-6}}$ ▷ safeguard
  end if
  $x \leftarrow x + s^{d_{\text{dir}}}$
  Path.append(x)
end for
$x_{\text{proj}} \leftarrow x$
info $\leftarrow$ {iterations: k, reason: reason, final_face: final_face}
return x_{proj}, Path, info

Algorithm 5 Projection of point P onto conv(S) via active-set QP

Require: A set of points $S = \{s_1, \ldots, s_n\} \subset R^d$, target point $P \in R^d$
Require: tolerance tol, maximum iterations $K_{\max}$
Ensure: projected point x_{proj}, barycentric weights $w \in R^n$, active index set A

Compute squared distances $d_i = \|s_i - P\|^2$ for all i
Let $k = \arg\min_i d_i$
Initialize active set $A = \{k\}$, weights $w = [1]$, point $x = s_k$
$t \leftarrow 0$
while $t < K_{\max}$ **do**
  $t \leftarrow t + 1$
  Compute $x - P$ and directional scores

$$\Delta_i = (x - P)^\top (s_i - x)$$

  Let $j = \arg\min_i \Delta_i$
  if $\Delta_j \geq -\text{tol}$ **then**
    break ▷ KKT conditions satisfied; optimal
  end if
  Add index j to active set A (if not already present)
  repeat
    Let $V = [\, s_i : i \in A \,]$ as a $d \times m$ matrix
    Form QP matrices for objective

$$\min_w \|Vw - P\|^2 \quad \text{s.t.} \ \sum w_i = 1,\ w_i \geq 0$$
$$H = 2V^\top V, \qquad c = -2V^\top P$$

    Solve QP for candidate weights w^{cand}
    if $w_i^{cand} \geq -\text{tol}$ for all i **then**
      Project w^{cand} onto simplex (set negatives to 0, renormalize)
      $w \leftarrow w^{cand}$; $x \leftarrow Vw$
      break ▷ successful add step
    end if
    Solve auxiliary QP to represent current x on A:

$$\min_w \|Vw - x\|^2, \quad \sum w_i = 1,\ w_i \geq 0$$

  yielding w^{old}
    Compute step ratios

$$t_i = \frac{w_i^{old}}{w_i^{old} - w_i^{cand}} \quad \text{for all } w_i^{cand} < 0$$

    Let $t^{=\min t_i}$, clipped to $[0, 1]$
    Interpolate

$$w^{interp} = w^{old} + t^{(w^{cand} - w^{old})}$$

    Remove all indices i from A for which $w_i^{interp} = 0$
    Update $w \leftarrow w^{interp}$ and continue QP solve on reduced A
  until active-set update is successful
end while
Construct full weight vector $w \in R^n$:

$$w_i = \begin{cases} \text{current weight on index } i, & i \in A \\ 0, & \text{otherwise} \end{cases}$$

Compute projection $x_{\text{proj}} = \sum_{i=1}^n w_i s_i$
return x_{proj}, w, A

Projection Verification Results

Vertices of the Sampling Set S. The convex hull is generated by the following 12 sampled points on the unit circle:

$$S = \begin{bmatrix} 1.0000 & 0.0000 \\ 0.8660 & 0.5000 \\ 0.5000 & 0.8660 \\ 0.0000 & 1.0000 \\ -0.5000 & 0.8660 \\ -0.8660 & 0.5000 \\ -1.0000 & 0.0000 \\ -0.8660 & -0.5000 \\ -0.5000 & -0.8660 \\ 0.0000 & -1.0000 \\ 0.5000 & -0.8660 \\ 0.8660 & -0.5000 \end{bmatrix}.$$

Algorithm Output. The projection computed by the active–set algorithm is

$$\hat{x}_{\text{alg}} = (0.908488,\ 0.341529),$$

with barycentric weights

$$w_{\text{alg}} = [\,0.316943,\ 0.683057,\ 0, \ldots, 0\,],$$

and active vertex set

$$A_{\text{alg}} = \{1,\ 0\}.$$

True Projection. Solving the exact convex projection problem via constrained minimization yields

$$\hat{x}_{\text{true}} = (0.908494,\ 0.341506),$$

with weights

$$w_{\text{true}} = [\,0.316987,\ 0.683013,\ 0, \ldots, 0\,].$$

Comparison Metrics. The difference between the algorithmic projection and the true projection is:

$$\|\hat{x}_{\text{alg}} - \hat{x}_{\text{true}}\| = 2.32 \times 10^{-5},$$

$$\|w_{\text{alg}} - w_{\text{true}}\|_1 = 8.96 \times 10^{-5}.$$

These results confirm that the custom active–set algorithm produces a projection extremely close to the true Euclidean projection onto the convex hull.

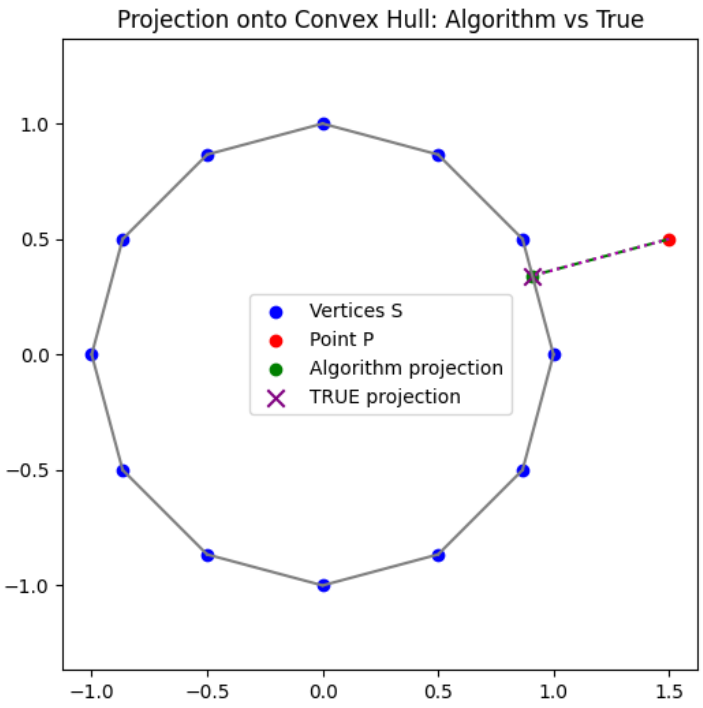

Figure 1: sampled circle with projection and estimate