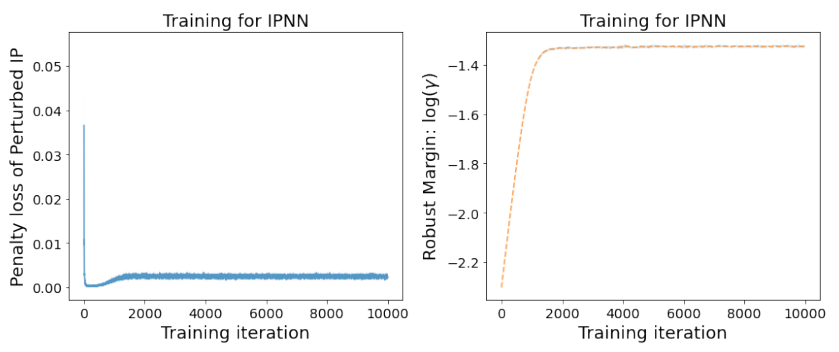
Supplementary Experiment Results for Reviewer h2io

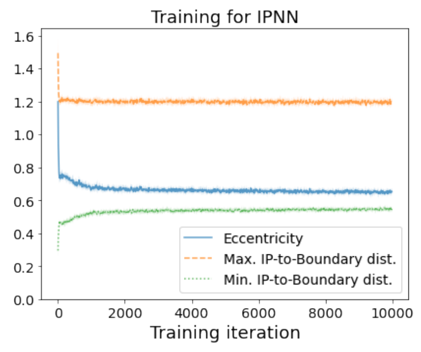
Contents

1. Ability of IPNN to Predict Low-eccentricity Points or Chebyshev centers.

To validate that the proposed training algorithm of IPNN can indeed produce low-eccentricity points or approximated Chebyshev centers, we consider the following non-convex quadratic constraint set $\mathcal{C}_\theta = \{x \mid x^\top Q_i x + q_i^\top x + b_i \leq 0, i = 1, \cdots, 6\}$, where the input parameter is defined as $\theta = \{Q, q, b\}_{i=1}^6$. We then train IPNN over such constraint sets and visualize its training and testing performance:



(a) Training loss of IPNN: the penalty loss of perturbed IP (Left) and the robust margin γ to be maximized (Right)



(b) The eccentricity of IPNN prediction (gap between max. and min. IP to boundary distance).

Figure 1: **Training performance of IPNN**: With the proposed training loss, the IPNN first identifies interior points (the penalty loss decreases in Fig. (a)), then maximizes the robust margin γ to find central points (the $\log\gamma$ increases in Fig. (b)). As a result, the minimum IP-to-boundary distance (Chebyshev radius) increases in Fig. (c) and the corresponding eccentricity decreases, demonstrating the effectiveness of our approach.

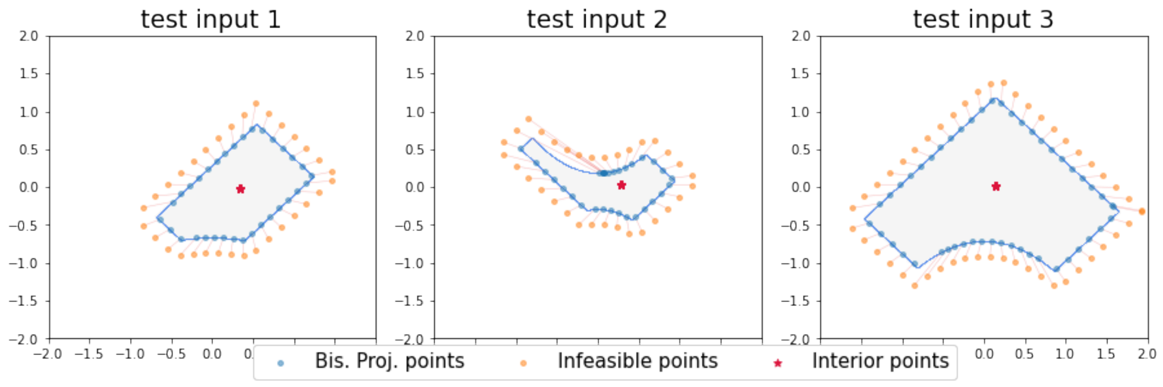


Figure 2: **Testing performance of IPNN**: After training, given unseen input parameters $\theta_1, \theta_2, \theta_3$, IPNN predicts "central" interior points with large IP-to-boundary distance (Chebyshev radius), confirming the generalization capabilities of our method.

2. Actual Baseline Methods Running Time and IPNN Training Time

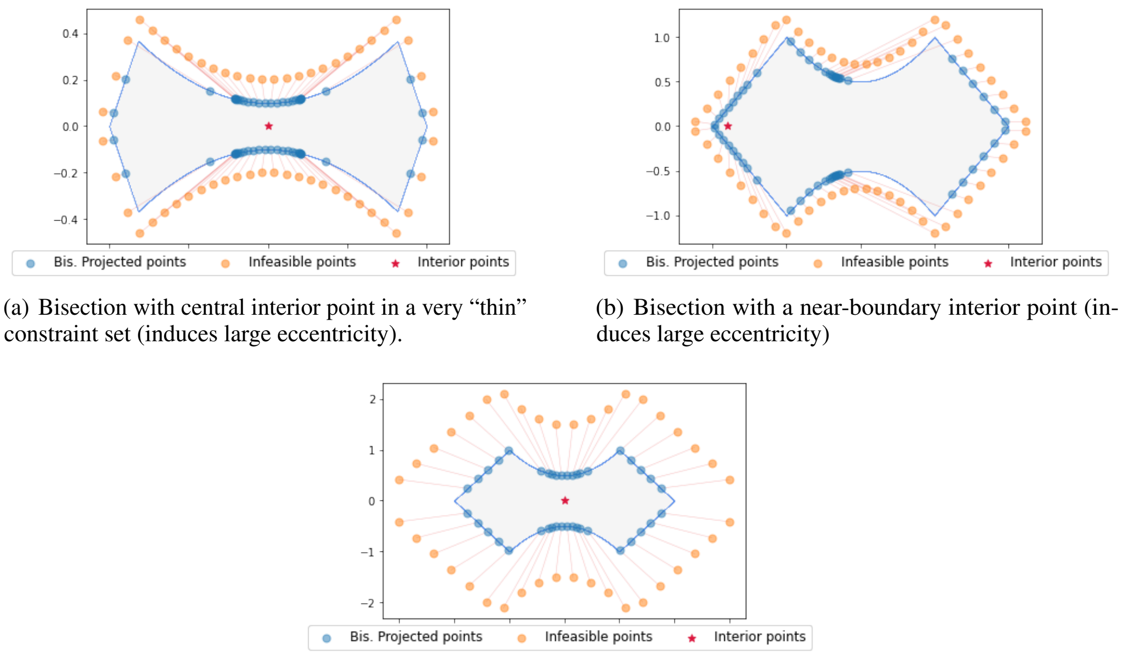
Table 1: **The average computational time** per instance for each problem (i.e., $\frac{\text{Total solving time with parallel execution}}{\text{Num. of instances}}$) for the baseline iterative solvers across different optimization problems, which serves as the base of the speedup calculation for all NN-based methods in the submitted manuscript.

Problems	Iterative solver time (s)
QP ($n = 400$)	0.062
QCQP ($n = 400$)	2.98
SOCP ($n = 400$)	1.69
SDP ($n = 40 \times 40$)	0.21
JCCIM ($n = 400$)	0.70
ACOPF ($n = 476$)	0.12

Table 2: **Total training time** for IPNN models on different problem instances, following parameter settings provided in the submitted manuscript (i.e., 3-layer NN with ReLU activation and residual connections, total iterations 10,000, batch size 64, and Adam optimizer with learning rate 1e-3.)

Problems	IPNN training time (s)
QP ($n = 400$)	31.75
QCQP ($n = 400$)	39.00
SOCP ($n = 400$)	33.97
SDP ($n = 40 \times 40$)	132.11
JCCIM ($n = 400$)	187.01
ACOPF ($n = 476$)	439.06

3. Illustrative Example of Failure case.



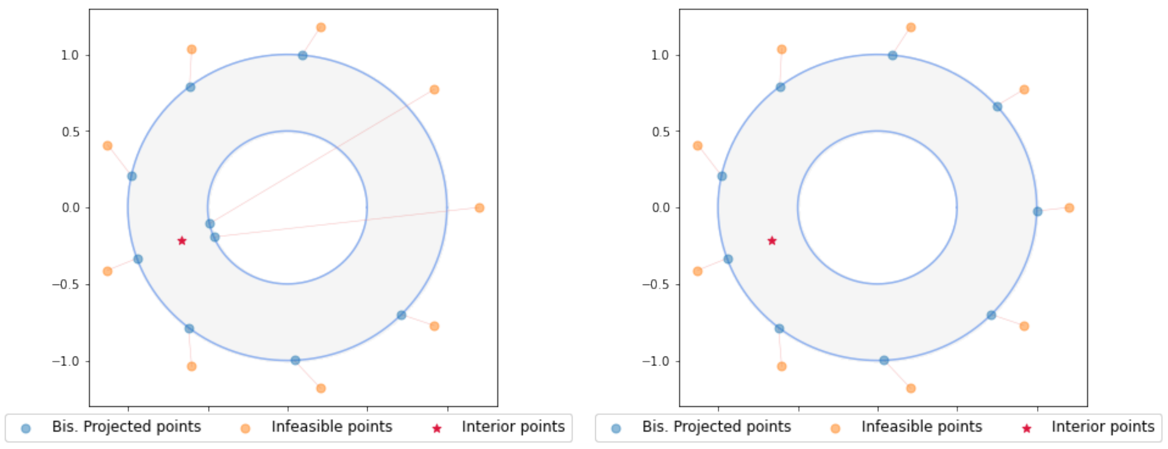
(a) Bisection with central interior point in a very "thin" constraint set (induces large eccentricity).

(b) Bisection with a near-boundary interior point (induces large eccentricity)

(c) Projecting infeasible points far from the boundary (i.e., large initial prediction error of the NN solution)

Figure 3: **Some failure cases for bisection projection that produce significant projection distances:** (a) In highly elongated or thin constraint sets, even central points can have high eccentricity, leading to suboptimal projections. (b) Interior points near the boundary cause larger worst-case projection distances, especially for points on the opposite side of the feasible region. (c) When the initial neural network prediction has large errors, placing infeasible points far from the boundary, the projection distance necessarily increases regardless of interior point quality.

4. Multiple Intersection Scenarios and Mechanism to Avoid it



(a) Bisection with stepsize 0.5, converges in 9 iterations. (b) Bisection with stepsize 0.75, converges in 16 iterations

Figure 4: **The bisection projection with multiple intersections**: the algorithm converges to one of the multiple intersected boundary points. To avoid potentially poor converged points that are far away from the initial infeasible prediction, we may use a larger bisection size (e.g., 0.75) instead of the default choice of 0.5, such that the bisection trajectory will search from the initial infeasible point and move down to the boundary in a more controlled manner. This approach helps ensure that the algorithm converges to a boundary point closer to the initial prediction, preserving the quality of the original NN output, at the cost of more iterations.