Deep FlexQP: Accelerated Nonlinear Programming via Deep Unfolding

Authors: Alex Oshin, Rahul Vodeb Ghosh,
Augustinos D. Saravanos*, Evangelos A. Theodorou

Georgia Institute of Technology

arXiv:2512.01565

Presenter: Zhi-Long Han

University of Electronic Science and Technology of China

December 17, 2025

Background: QP and Nonlinear Programming

Quadratic Programs (QP)

$$\min_x \tfrac{1}{2} x^\top P x + q^\top x \quad \text{s.t. } Gx \le h,\ Ax = b,$$

where $P \in \mathbb{R}^{n \times n}$, $h \in \mathbb{R}^m$, and $b \in \mathbb{R}^p$.

- Appear as subproblems in model predictive control, portfolio optimization, etc.
- Large-scale settings: n, m, p can be 10^4–10^5.

Role in Nonlinear Programming (NLP)

- Many constrained NLP problems are solved by *sequential local approximations*.
- Quality and speed of QP solvers directly affect overall NLP performance.

Background: Sequential Quadratic Programming (SQP)

Nonlinear Programming Problem

$$\min_x \ f(x) \quad \text{s.t. } c(x) = 0,\ d(x) \le 0.$$

SQP Idea

- At iterate x_k, quadraticize the Lagrangian and linearize constraints to solve for step w:

$$\min_w \ \tfrac{1}{2} w^\top H_k w + \nabla f(x_k)^\top w \quad \text{s.t. } \begin{cases} \nabla c(x_k) w + c(x_k) = 0 \\ \nabla d(x_k) w + d(x_k) \le 0 \end{cases}$$

  where $H_k = \nabla^2_{xx} \mathcal{L}$ is the Hessian of the Lagrangian.
- This constitutes a QP subproblem.
- Update $x_{k+1} = x_k + \alpha_k w_k$ via line-search / trust-region.

Why Robust QP Matters

- Linearization can make constraints inconsistent $\Rightarrow$ infeasible QP subproblems.
- Need a QP solver that remains meaningful even when subproblems are infeasible.

Challenges in Practical QP Solving

Core Challenges

- **Infeasible QP subproblems:** SQP linearization yields inconsistent constraints.
- **Hyperparameter tuning:** ADMM/OSQP involve many hyperparameters.
- **Scalability:** need efficient methods for 10^4-scale problems.
- **Robustness:** solvers should handle model mismatch and ill-conditioning.

Motivation

- Combine robust convex optimization with learned parameter policies.
- Preserve convergence guarantees while accelerating iterations.

Related Work: ADMM-based QP & Learning to Optimize

ADMM-based QP Solvers

- OSQP: operator splitting QP solver with factorization reuse.[2]
- Good scalability but sensitive to choice of ρ, relaxation α, regularization σ.
- Typically assume the QP is feasible; infeasible subproblems only yield "failure" status.[1]

Learning to Optimize (Deep Unfolding)

- Deep unfolding for sparse coding and inverse problems.[3]
- Deep OSQP: learn penalty parameters as functions of ADMM residuals.[4]
- RLQP: use reinforcement learning to tune ADMM parameters.[5]
- *Limitations:* mostly scalar penalties; lack unified treatment of infeasibility and constraint-wise behavior.

---

[1] Boyd S. et al., "Distributed Optimization and Statistical Learning via ADMM," 2011.
[2] Stellato B. et al., "OSQP: An Operator Splitting Solver for Quadratic Programs," MPC, 2020.
[3] Gregor K., LeCun Y., "Learning Fast Approximations of Sparse Coding," ICML, 2010.
[4] Saravanos, Augustinos D., et al. "Deep Distributed Optimization for Large-Scale Quadratic Programming." arXiv preprint arXiv:2412.12156 (2024).
[5] Ichnowski, Jeffrey, et al. "Accelerating quadratic optimization with reinforcement learning." Advances in Neural Information Processing Systems 34 (2021): 21043-21055.

Target Problem and Design Goals

Target

- Build a QP solver that:
  - Handles both feasible and infeasible problems in a unified way.
  - Is suitable as an inner loop of SQP/NLP.
  - Can be accelerated via learned parameter policies.

Design Goals

- **Robustness:** returns a meaningful solution with interpretable constraint violations.
- **Structure-preserving:** retain ADMM splitting structure.
- **Learnable:** treat penalty/relaxation parameters as outputs of neural networks.

FlexQP: Elastic ℓ_1 Relaxation of QP

From Standard QP to Elastic QP

Introduce slack $s \geq 0$ for inequalities ($Gx + s - h = 0$) and add exact ℓ_1 penalties:

$$\min_{x,\, s \geq 0} \tfrac{1}{2} x^\top P x + q^\top x + \mu_I \|Gx + s - h\|_1 + \mu_E \|Ax - b\|_1.$$

Key Property

- For sufficiently large (μ_I, μ_E):

  Elastic QP optimum = Original QP optimum.

- If original QP is infeasible, the solution minimizes ℓ_1 constraint violation.

Operator Splitting & ADMM: Problem

Optimization problem

$$\begin{aligned} \min_{x,s,z_I,z_E} \quad & \tfrac{1}{2}x^\top Px + q^\top x + \mu_I\|z_I\|_1 + \mu_E\|z_E\|_1, \\ \text{subject to} \quad & z_I = Gx + s - h, \\ & z_E = Ax - b, \\ & s \geq 0. \end{aligned} \tag{1}$$

Final splitting with copy variables

$$\min_{\tilde{x},x} \quad \underbrace{\tfrac{1}{2}x^\top Px + q^\top x + \mathcal{I}_I(\tilde{x}) + \mathcal{I}_E(\tilde{x}) + \mathcal{I}_s(x)}_{=:f(\tilde{x})} + \underbrace{\mu_I\|z_I\|_1 + \mu_E\|z_E\|_1}_{=:g(x)} \tag{2}$$

$$\text{subject to} \quad \tilde{x} = x,$$

where $\tilde{\boldsymbol{x}} = (\tilde{x}, \tilde{s}, \tilde{z}_I, \tilde{z}_E)$ and $\boldsymbol{x} = (x, s, z_I, z_E)$ for notational simplicity. The indicator functions $\mathcal{I}_I, \mathcal{I}_E$, and $\mathcal{I}_s$ are defined as

$$\mathcal{I}_I(\boldsymbol{x}) = \begin{cases} 0, & z_I = Gx + s - h, \\ +\infty, & \text{otherwise}, \end{cases} \quad \mathcal{I}_E(\boldsymbol{x}) = \begin{cases} 0, & z_E = Ax - b, \\ +\infty, & \text{otherwise}, \end{cases} \quad \mathcal{I}_s(\boldsymbol{x}) = \begin{cases} 0, & s \geq 0, \\ +\infty, & \text{otherwise}. \end{cases}$$

Operator Splitting & ADMM: Solution

Let the dual variables for $\tilde{x} = x$ be $y = (w_x, w_s, y_I, y_E)$. The ADMM updates are

$$
\begin{aligned}
\tilde{x}^{k+1} &= \arg\min_{\tilde{x}} \; f(\tilde{x}) + \tfrac{\sigma_x}{2}\|\tilde{x} - x^k + \sigma_x^{-1} w_x^k\|_2^2 + \tfrac{\sigma_s}{2}\|\tilde{s} - s^k + \sigma_s^{-1} w_s^k\|_2^2 \\
&\quad + \tfrac{\rho_I}{2}\|\tilde{z}_I - z_I^k + \rho_I^{-1} y_I^k\|_2^2 + \tfrac{\rho_E}{2}\|\tilde{z}_E - z_E^k + \rho_E^{-1} y_E^k\|_2^2, \\
x^{k+1} &= \alpha\tilde{x}^{k+1} + (1-\alpha)x^k + \sigma_x^{-1} w_x^k, \\
s^{k+1} &= \left(\alpha\tilde{s}^{k+1} + (1-\alpha)s^k + \sigma_s^{-1} w_s^k\right)_+, \\
z_I^{k+1} &= \mathcal{S}_{\mu_I/\rho_I}\left(\alpha\tilde{z}_I^{k+1} + (1-\alpha)z_I^k + \rho_I^{-1} y_I^k\right), \\
z_E^{k+1} &= \mathcal{S}_{\mu_E/\rho_E}\left(\alpha\tilde{z}_E^{k+1} + (1-\alpha)z_E^k + \rho_E^{-1} y_E^k\right), \\
w_x^{k+1} &= w_x^k + \sigma_x(\tilde{x}^{k+1} - x^{k+1}), \\
w_s^{k+1} &= w_s^k + \sigma_s(\tilde{s}^{k+1} - s^{k+1}), \\
y_I^{k+1} &= y_I^k + \rho_I(\tilde{z}_I^{k+1} - z_I^{k+1}), \\
y_E^{k+1} &= y_E^k + \rho_E(\tilde{z}_E^{k+1} - z_E^{k+1}).
\end{aligned}
$$

where $\sigma_x, \sigma_s, \rho_I, \rho_E > 0$ are the augmented Lagrangian penalty parameters, $\alpha \in (0, 2)$ is the ADMM relaxation parameter, $(s)_+ = \max(s, 0)$ is the ReLU activation function, and $S_\kappa(z) = (z - \kappa)_+ - (-z - \kappa)_+$is the soft thresholding operator, which is the proximal operator of the ℓ_1 norm.

FlexQP: Exact Penalty and Always-Feasible Behavior

Exact Penalty Theorem (informal)

Let $(x^\star, y_I^\star, y_E^\star)$ be a KKT point of the original QP. If

$$\mu_I \geq \|y_I^\star\|_\infty, \quad \mu_E \geq \|y_E^\star\|_\infty,$$

then any minimizer of the elastic problem is also optimal for the original QP.

Implications

- FlexQP is always feasible: it always returns some (x, s).
- For infeasible QP, elastic solution gives minimum ℓ_1 constraint violation.
- Dual variables saturating $|y_{I,i}| = \mu_{I,i}$ identify the most violated constraints.

ADMM Iterations as a Deep Network

Layer = One ADMM Iteration (k)

- Solve equality QP $\Rightarrow (\tilde{x}^{k+1}, \tilde{s}^{k+1}, \tilde{z}^{k+1})$.
- Element-wise updates:
  - $x^{k+1} = (1-\alpha^k)x^k + \alpha^k \tilde{x}^{k+1}$.
  - $s^{k+1} = \max\left(0, (1-\alpha^k)s^k + \alpha^k \tilde{s}^{k+1}\right)$.
  - $z^{k+1} = S_{\mu^k/\rho^k}(\cdot)$ (soft-threshold on z_I, z_E).
- Dual variable updates w^{k+1}, y^{k+1}.

Deep FlexQP: Stack K such layers. Parameters $(\mu, \rho, \sigma, \alpha)$ are treated as learnable, iteration-dependent quantities.

ADMM Iterations as a Deep Network

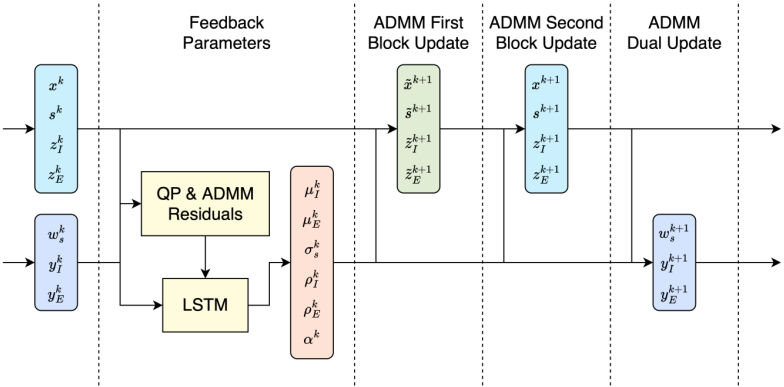


Figure: One layer of our proposed Deep FlexQP architecture. We learn dimension-agnostic feedback policies for the parameters while the propagation from one layer to the next is defined by the ADMM updates Equation 9.

Learning Per-Constraint Parameter Policies

Classical ADMM Heuristic

- Use *primal / dual residuals* r^k, s^k to hand-tune penalties:

$$r^k := Ax^k + Bz^k - c, \qquad s^k := \rho A^\top B(z^k - z^{k-1}).$$

- OSQP and Deep OSQP use scalar (or vector) rules of this kind.

Deep FlexQP Policies

- **Inequality policy** π_I:
  - *Inputs:* $(s^k, z_I^k, w_s^k, y_I^k)$, relaxed QP residual ζ_I^k, and $\|\zeta_{\text{dual}}^k\|_\infty$
    $\zeta_I^k = Gx^k + s^k - h - z_I^k$ and $\zeta_{\text{dual}}^k = Px^k + q + G^\top y_I^k + A^\top y_E^k$.
  - *Outputs:* $(\mu_I^k, \rho_I^k, \sigma_s^k)$
- **Equality policy** π_E:
  - *Inputs:* (z_E^k, y_E^k), relaxed QP residual ζ_E^k, and $\|\zeta_{\text{dual}}^k\|_\infty$
    $\zeta_E^k = Ax^k - b - z_E^k$
  - *Outputs:* (μ_E^k, ρ_E^k)
- **Relaxation policy** π_α:
  - *Inputs:* $\|\zeta_{\text{QP}}^k\|_\infty$ and $\|\zeta_{\text{ADMM}}^k\|_\infty$
  - *Output:* relaxation parameter α^k

Training Objective and Supervision

Supervised Targets

For training QPs with known solutions $\xi^\star = (x^\star, y_I^\star, y_E^\star)$:

Normalized Optimality Loss

$$\mathcal{L}_{\text{opt}}(\theta) = \sum_{k=1}^{K} \frac{\|\xi^k(\theta) - \xi^\star\|_2^2}{\|\xi^\star\|_2^2}, \quad \xi^k = (x^k, y_I^k, y_E^k).$$

- Includes both primal x and dual (y_I, y_E).
- Encourages learned penalties (μ_I^k, μ_E^k) to be compatible with exact-penalty theory:

$$|y_{I,i}^\star| \le \mu_{I,i}^k, \quad |y_{E,j}^\star| \le \mu_{E,j}^k.$$

- Later, this supervised model serves as the **prior** in PAC-Bayes analysis.

KKT Residuals and Log-Scaled Loss

KKT Residual Vector

$$R(\xi) = \begin{bmatrix} \nabla_x L(x, y_I, y_E) \\ \text{primal constraint residuals} \\ \text{complementarity residuals} \end{bmatrix}.$$

Why Residual-Based Loss?

- Standard optimality gap can become numerically saturated when residuals are tiny.
- Need a loss that remains sensitive in the "high-accuracy" regime.

Log-Scaled Residual Loss (for PAC-Bayes)

$$\mathcal{L}_{\log}(\theta) = \mathsf{clip}\Big(1 - \frac{\log \|R(\xi_K(\theta))\|_2}{\log \|R(\xi^\star)\|_2}, 0, 1\Big).$$

PAC-Bayes Training Pipeline

Stage 1: Supervised Training (Deterministic)

- Train Deep FlexQP on a base QP class (e.g., random QPs with equalities).
- Use $\mathcal{L}_{\text{opt}}$ to obtain a good parameter vector θ_0.

Stage 2: Define a Prior over Parameters

- Construct a stochastic model by placing a distribution over θ:

$$P_0 = \mathcal{N}(\theta_0, \sigma_0^2 I).$$

- This is the *prior* for PAC-Bayes: captures a "good neighborhood" around θ_0.

Stage 3: Task-Specific Posterior P

- For a new QP task (e.g., LASSO, control), choose a parametric family P_ϕ.
- Optimize a PAC-Bayes bound:

$$\ell_S(P_\phi) + \sqrt{\frac{\mathrm{KL}(P_\phi \| P_0) + \log(2\sqrt{N}/\delta)}{2N}},$$

where $\ell_S(P_\phi)$ is empirical loss using $\mathcal{L}_{\log}$.

Key Theory I: Exact Penalty & Dual Bounds

Theorem 1 (Exact ℓ_1 Penalty)

For the original QP with KKT point $(x^\star, y_I^\star, y_E^\star)$, define

$$\mu_I^\star = \|y_I^\star\|_\infty, \quad \mu_E^\star = \|y_E^\star\|_\infty.$$

If $\mu_I \geq \mu_I^\star, \mu_E \geq \mu_E^\star$, then any minimizer of the elastic QP coincides with $x^\star$.

Theorem 2 (Dual Bounds at FlexQP Optimum)

Let $(\hat{x}, \hat{s}, \hat{y}_I, \hat{y}_E)$ solve the elastic QP. Then

$$|\hat{y}_{I,i}| \leq \mu_{I,i}, \quad |\hat{y}_{E,j}| \leq \mu_{E,j} \quad \forall i, j.$$

If the original QP is infeasible, active violated constraints satisfy $|\hat{y}_{I,i}| = \mu_{I,i}$.

Key Theory II: ADMM Convergence & PAC-Bayes Bound

Theorem 3 (FlexQP-ADMM Convergence)

Under mild coercivity assumptions on the elastic objective, the two-block ADMM updates (Eq. (9)) converge to a saddle point of the augmented Lagrangian, regardless of whether the original QP is feasible.

Theorem 4 (PAC-Bayes Generalization)

Consider a distribution P over Deep FlexQP parameters. With probability at least $1-\delta$ over the training set S:

$$\underbrace{L_{\mathcal{D}}(P)}_{\text{true risk}} \leq \underbrace{L_S(P)}_{\text{empirical risk}} + \sqrt{\frac{\mathrm{KL}(P\|P_0) + \log(2\sqrt{N}/\delta)}{2N}}.$$

- $L_S(P)$ computed using log-scaled residual loss $\mathcal{L}_{\log}$.
- $\mathrm{KL}(P\|P_0)$ penalizes deviation from the supervised prior.

Experimental Methodology and Baselines

Experimental Goal

- Evaluate *convergence speed*, *robustness*, and *scalability* of learned QP solvers.
- Focus on settings with mixed constraints and potential infeasibility.

Baselines

ADMM-based QP Solvers

- OSQP: operator-splitting QP solver with adaptive penalties.[2]
- FlexQP: ℓ_1-relaxed always-feasible QP solver (non-learned).

Learning to Optimize

- Deep OSQP: scalar penalty learning via deep unfolding.[4]
- Deep OSQP (Improved): vector penalties + learned relaxation α.
- RLQP: reinforcement learning for ADMM parameter tuning.[5]

Proposed Method

- **Deep FlexQP**: constraint-wise learned penalties with explicit ℓ_1 relaxation for infeasible QPs.

Benchmark Problems: QP Reformulations

We evaluate on three distinct problem classes transformed into standard QP form:

$$\min_z \tfrac{1}{2} z^\top P z + q^\top z \quad \text{s.t.} \quad Gz \leq h,\ Az = b$$

1. Portfolio Optimization

Maximize risk-adjusted return using a **factor model** $\Sigma = FF^\top + D$ to maintain sparsity.

Original: $\max \mu^\top x - \gamma x^\top \Sigma x$

QP Form:

$$\begin{aligned} \min_{x,y} \quad & x^\top D x + y^\top y - \gamma^{-1} \mu^\top x \\ \text{s.t.} \quad & y = F^\top x \\ & \mathbf{1}^\top x = 1 \\ & x \geq 0 \end{aligned}$$

Variables: Assets x, Factors y.

2. Support Vector Machines

Linear classification minimizing hinge loss with regularization λ.

Original: Hinge Loss Minimization

QP Form:

$$\begin{aligned} \min_{x,t} \quad & x^\top x + \lambda \mathbf{1}^\top t \\ \text{s.t.} \quad & t \geq \text{diag}(b) A x + \mathbf{1} \\ & t \geq 0 \end{aligned}$$

Variables: Weights x, Hinge Slacks t.

3. Huber Fitting

Robust regression. Residuals split into quadratic part (u) and linear part (r, s).

Original: $\min \sum \phi_{hub}(a_i^\top x - b_i)$

QP Form:

$$\begin{aligned} \min_{x,u,r,s} \quad & u^\top u + 2\delta \mathbf{1}^\top (r + s) \\ \text{s.t.} \quad & Ax - b - u = r - s \\ & r, s \geq 0 \end{aligned}$$

Variables: Weights x, Residual components u, r, s.

Ref: Appendix G.2, G.3, and G.5 of "Deep FlexQP: Accelerated Nonlinear Programming via Deep Unfolding".

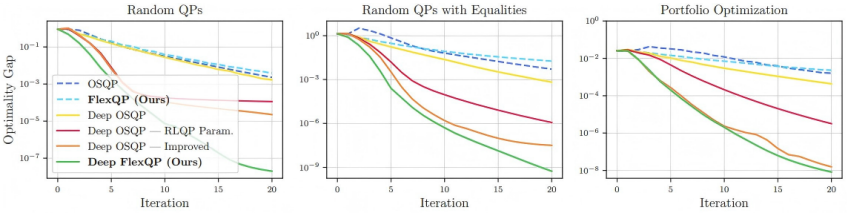
Experiments 5.2: Small and Large-Scale QPs

Problem Classes

- Random QPs (with / without equalities)
- Portfolio optimization
- SVM, LASSO, Huber fitting

Evaluation Protocol

- Train on 500–2000 QPs, test on 1000 unseen instances.
- Metric: optimality gap vs. iteration.
- All methods use best-tuned hyperparameters.



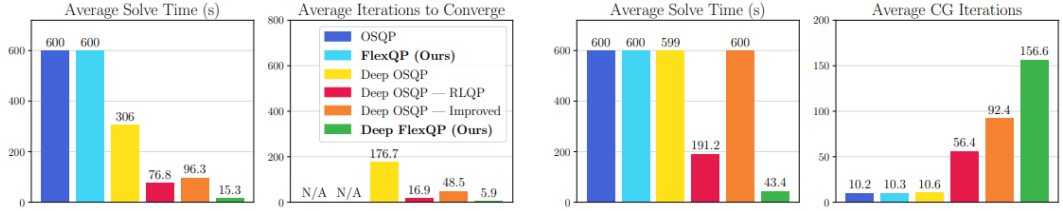
Experiment 5.2: Large-Scale QPs

Setup

- Large-scale problems with up to 10^4 variables.
- Portfolio optimization and large-margin SVMs.
- Indirect (CG-based) ADMM updates.

Methodology

- Fine-tune models trained on small QPs.
- Stop when $\|\text{QP residual}\|_\infty \leq 10^{-3}$ or timeout.



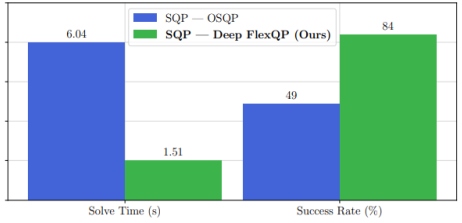
Experiment 5.3: SQP with Deep FlexQP

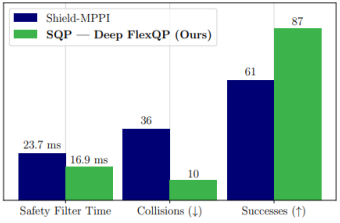
Motivation

- Sequential Quadratic Programming (SQP) generates *infeasible QP subproblems*.
- Standard QP solvers often fail or stall in this regime.

Setup

- Use Deep FlexQP as the QP subsolver inside SQP.
- Compare SQP + OSQP vs. SQP + Deep FlexQP.
- Applications: trajectory optimization and safety filters.





Experimental Takeaways

- Deep FlexQP outperforms classical and learned baselines across problem scales and constraint structures.
- Explicit ℓ_1 relaxation enables robust handling of infeasible QPs.
- Constraint-wise learned penalties yield faster and more stable convergence.
- Particularly effective when used as a subsolver in SQP.

Conclusion: Deep FlexQP is not only a faster QP solver, but a robust and scalable optimization module for learning-based and nested optimization pipelines.

Summary, Limitations and Future Directions

Main Ideas

- FlexQP: elastic ℓ_1 reformulation yielding an always-feasible QP solver.
- Deep FlexQP: deep unfolding with learned, per-constraint parameter policies.
- PAC-Bayes: provides generalization guarantees for the learned optimizer.

Key Contributions

- Unified treatment of feasible and infeasible QPs with exact-penalty guarantees.
- Dimension-agnostic LSTM policies acting on constraint-wise residuals.
- Strong empirical gains on convex QPs and nonlinear control tasks.

Thank you!