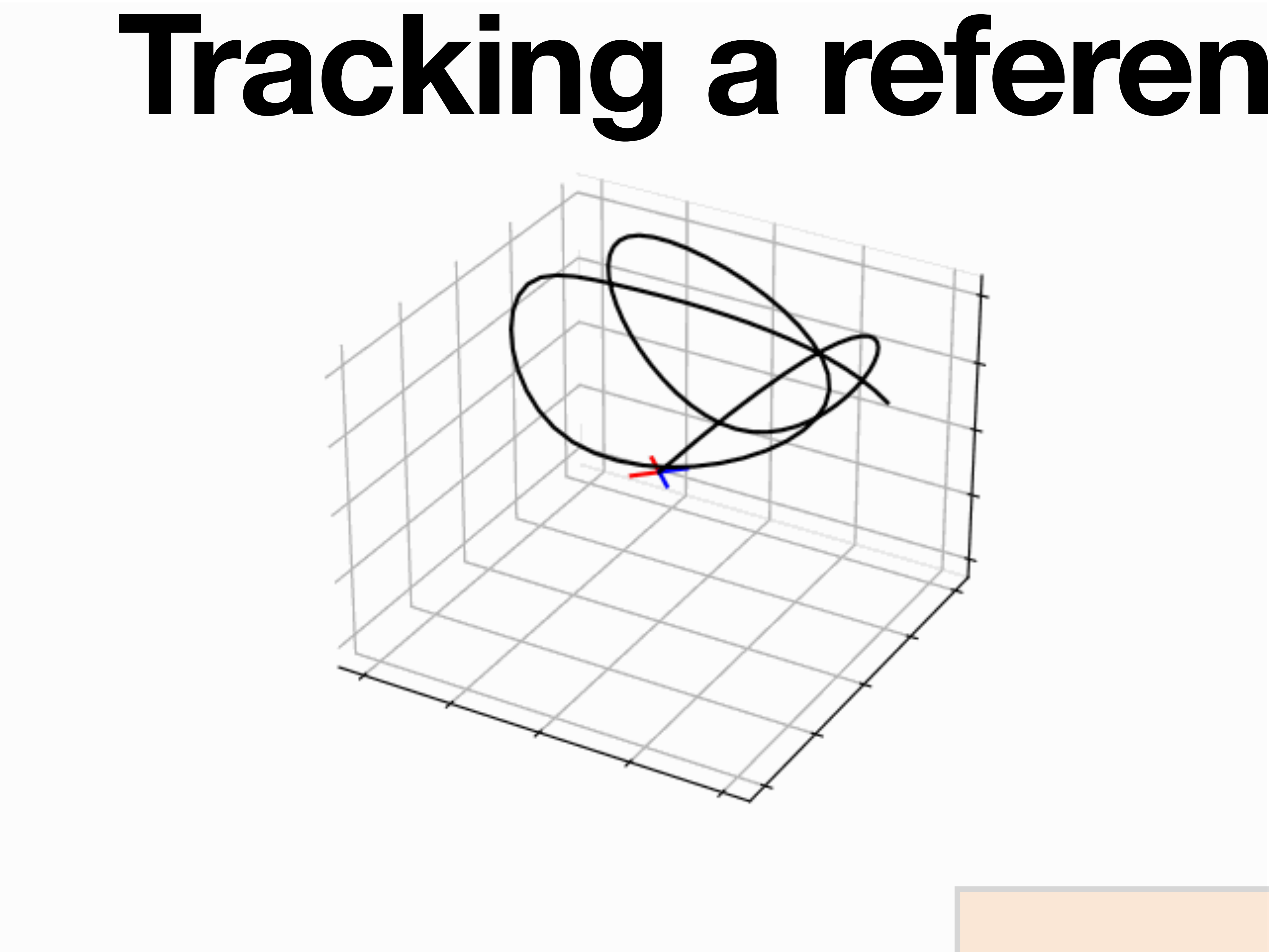
Learning to Warm-Start Fixed-Point Optimization Algorithms

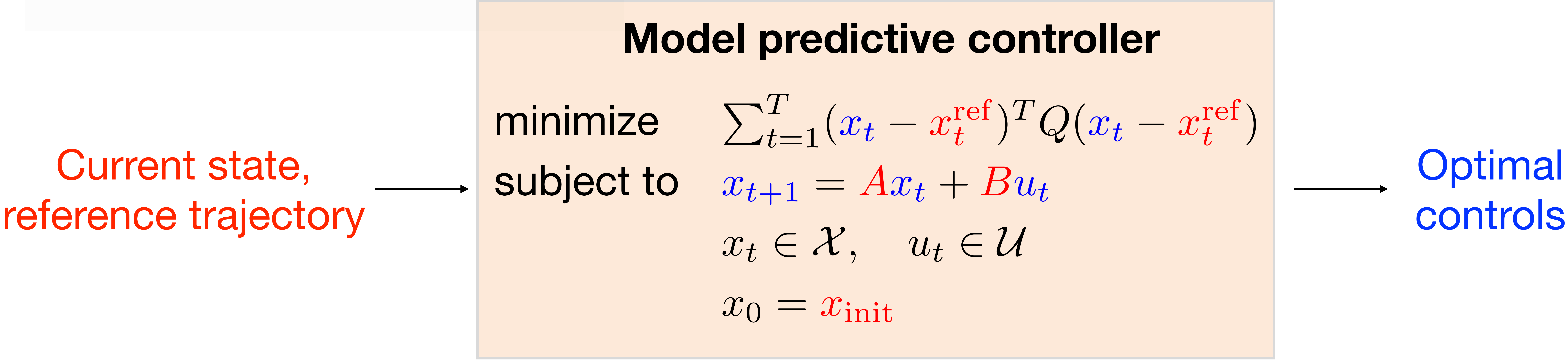
Yale Robotics Seminar 2023

Rajiv Sambharya

Tracking a reference trajectory with a quadcopter



Model predictive control



Current state, reference trajectory

Model predictive controller

minimize $\sum_{t=1}^{T}(x_t - x_t^{\mathrm{ref}})^T Q (x_t - x_t^{\mathrm{ref}})$

subject to $x_{t+1} = Ax_t + Bu_t$

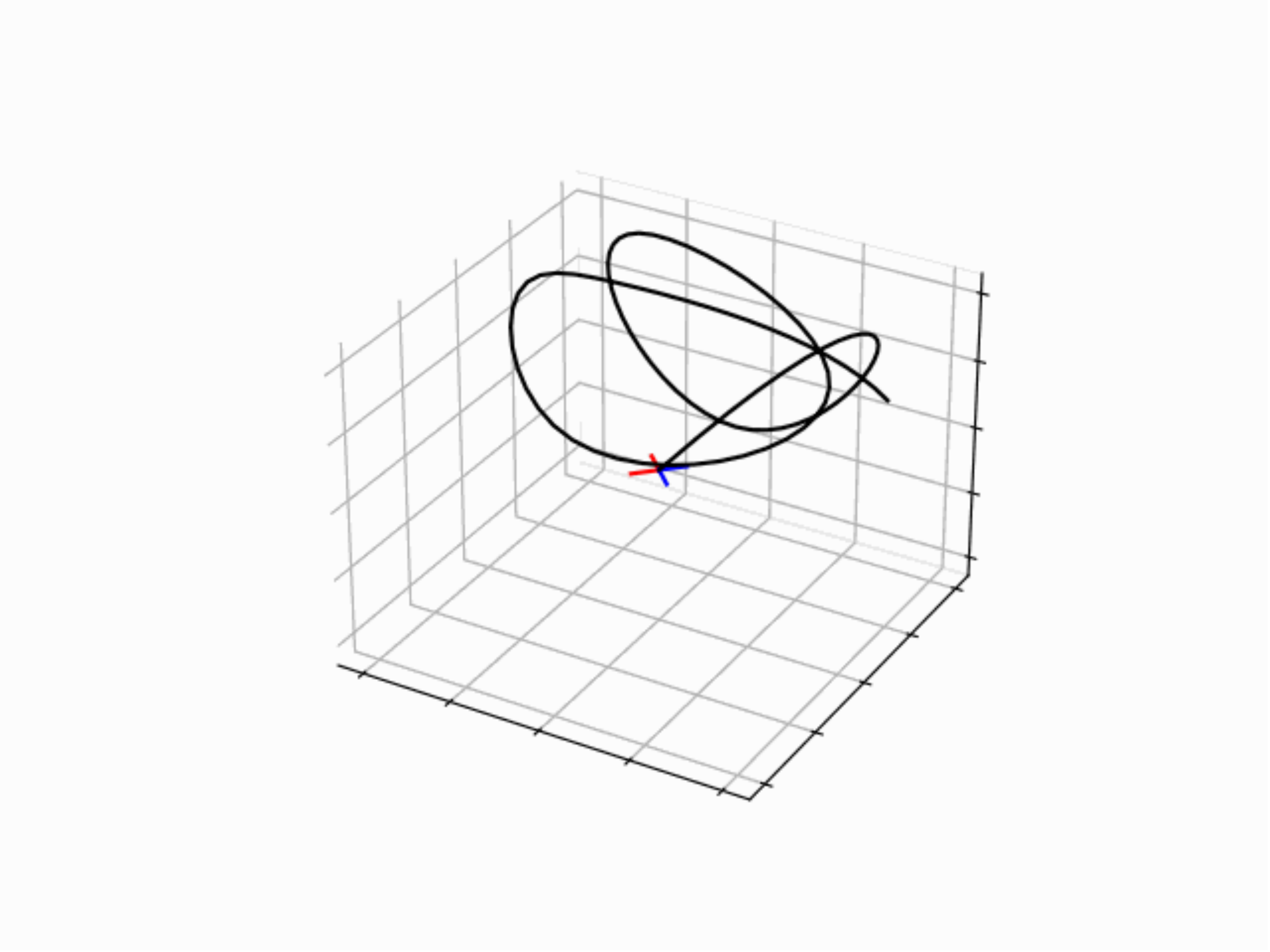
$x_t \in \mathcal{X}, \quad u_t \in \mathcal{U}$

$x_0 = x_{\mathrm{init}}$

Optimal controls

Challenge: we need faster methods to solve optimization problems

Robotics and control

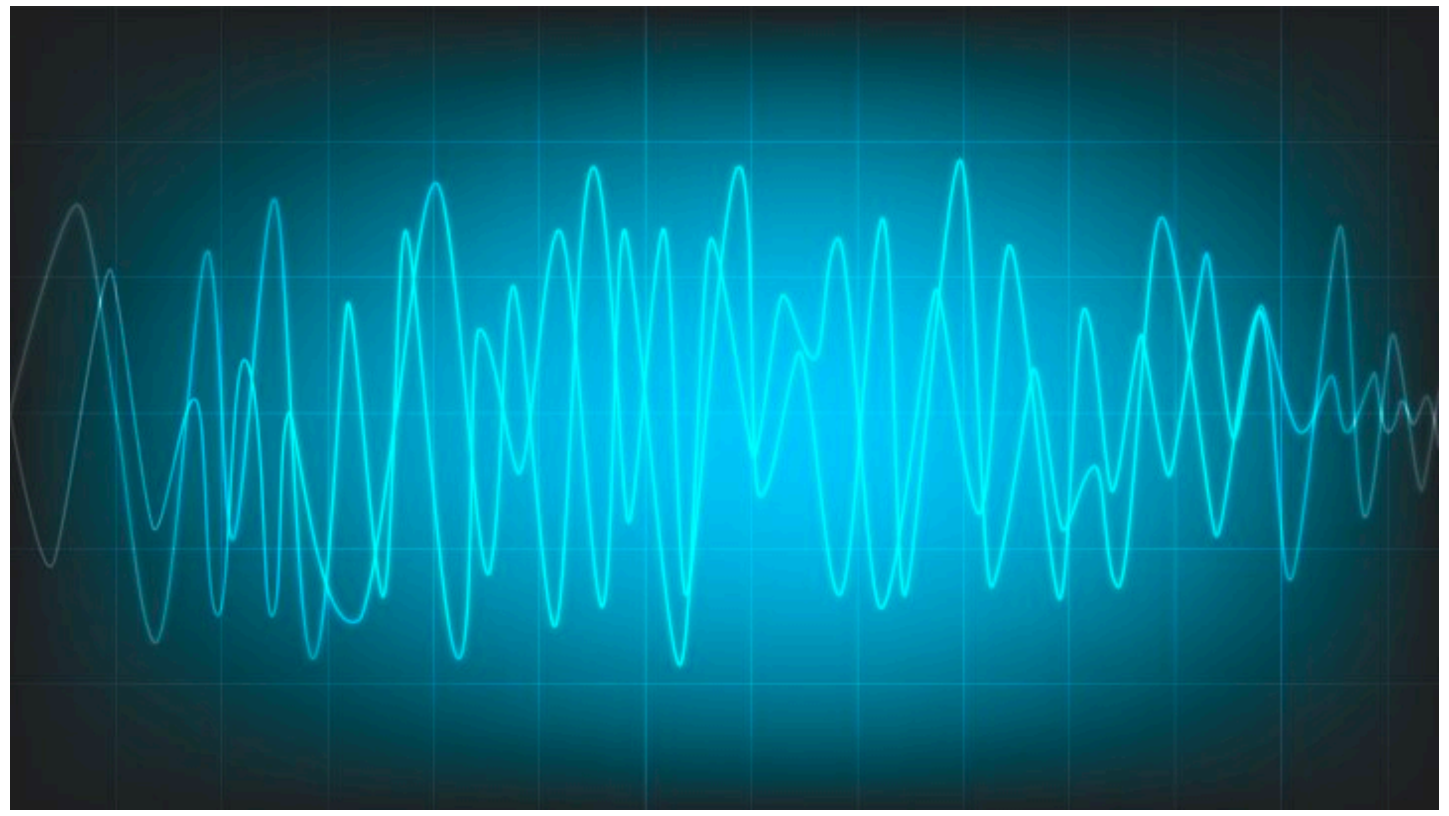


We sometimes need solutions in ~10 milliseconds or less

Claim: Real-world optimization is parametric

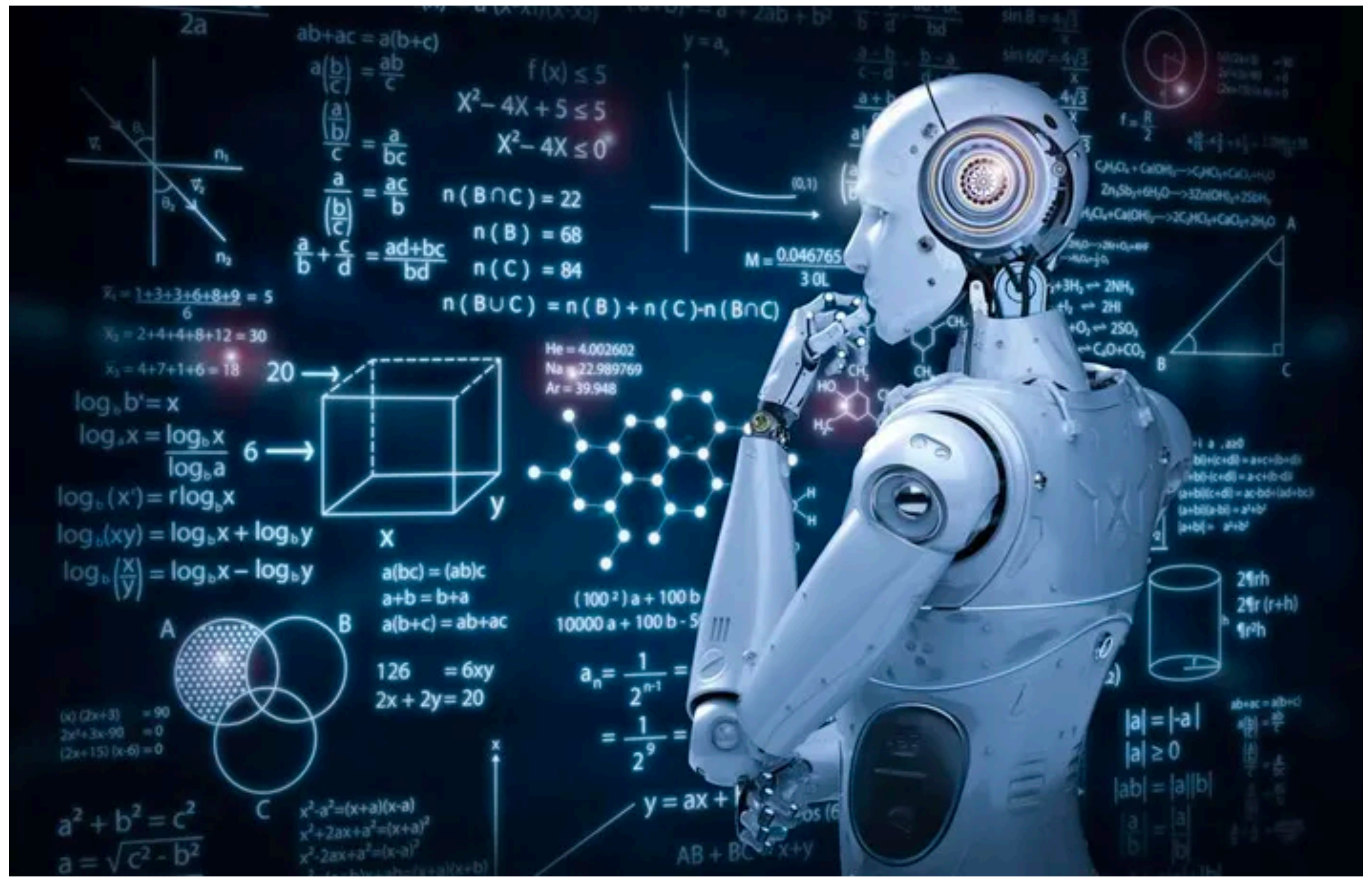
Robotics and control

Signal processing



Energy

Machine learning



Can machine learning speed up parametric optimization?

Often, we solve **parametric** optimization problems from the same family

Goal: Do mapping quickly and accurately

Parameter

Optimal solution

$$\theta \longrightarrow \boxed{\begin{array}{ll}\text{minimize} & f_\theta(z)\\ \text{subject to} & g_\theta(z) \leq 0\end{array}} \longrightarrow z^\star(\theta)$$

$\theta \longrightarrow$ Only Optimization 🐢 $\longrightarrow \hat{z}^{\mathrm{Opt}}(\theta)$

$\theta \longrightarrow$ Only Machine Learning 🐇 $\longrightarrow \hat{z}^{\mathrm{ML}}(\theta)$

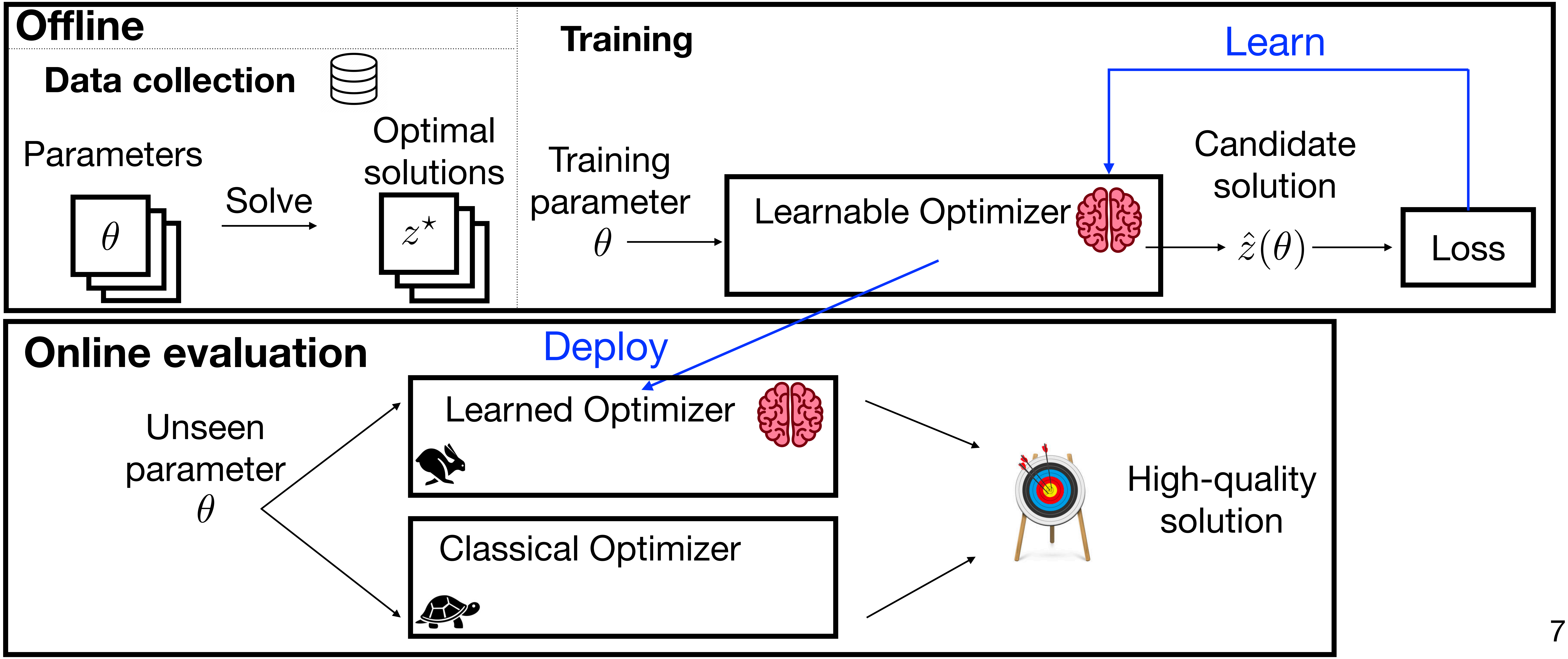
$\theta \longrightarrow$ Optimization 🤝 Machine Learning 🐇 $\longrightarrow \hat{z}^{\mathrm{Opt/ML}}(\theta)$

Learning to Optimize

The learning to optimize paradigm

Goal: solve the parametric optimization problem fast

$$\begin{array}{ll} \text{minimize} & f_\theta(z) \\ \text{subject to} & g_\theta(z) \leq 0 \end{array}$$



Offline

Data collection

Parameters θ Solve → Optimal solutions $z^\star$

Training

Training parameter θ → Learnable Optimizer → Candidate solution $\hat{z}(\theta)$ → Loss

Learn

Deploy

Online evaluation

Unseen parameter θ

Learned Optimizer

Classical Optimizer

High-quality solution

Learning to optimize is a growing research area

Inverse problems

(Gregor and LaCun 2010)

(Liu et. al 2018)

(Wu et. al 2020)

Convex optimization

(Venkataraman and Amos 2021)

(Ichnowski et. al 2021)

(Heaton et. al 2020)

(Jung et. al 2022)

Integer programming

Excellent tutorials

Learning to Optimize: A Primer and a Benchmark (Chen et. al 2021)

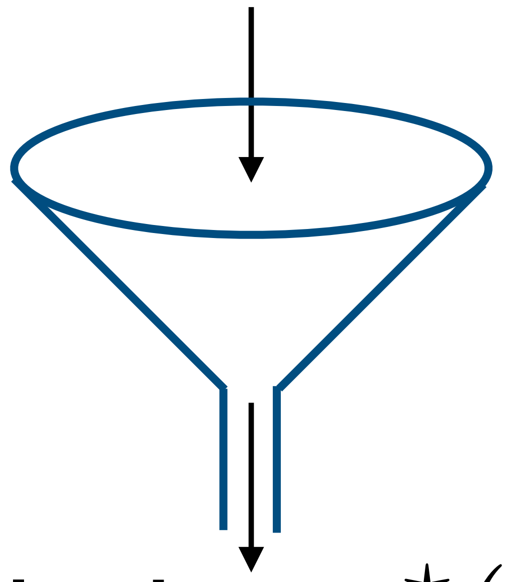
Tutorial on Amortized Optimization (Amos 2022)

Issues in learning to optimize (L2O) methods

- **I: Lack convergence guarantees**
- **II: Lack generalization guarantees**
- **III: Incompatibility with state-of-the-art solvers**

We need reliable L2O methods

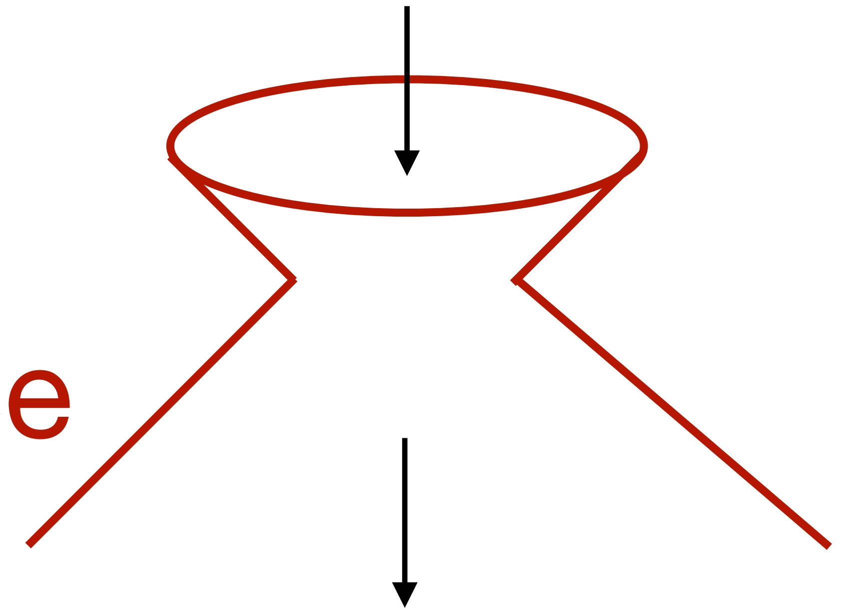
L2O Challenge I: Convergence guarantees



Parameter θ

Solver with convergence

Solution $z^\star(\theta)$



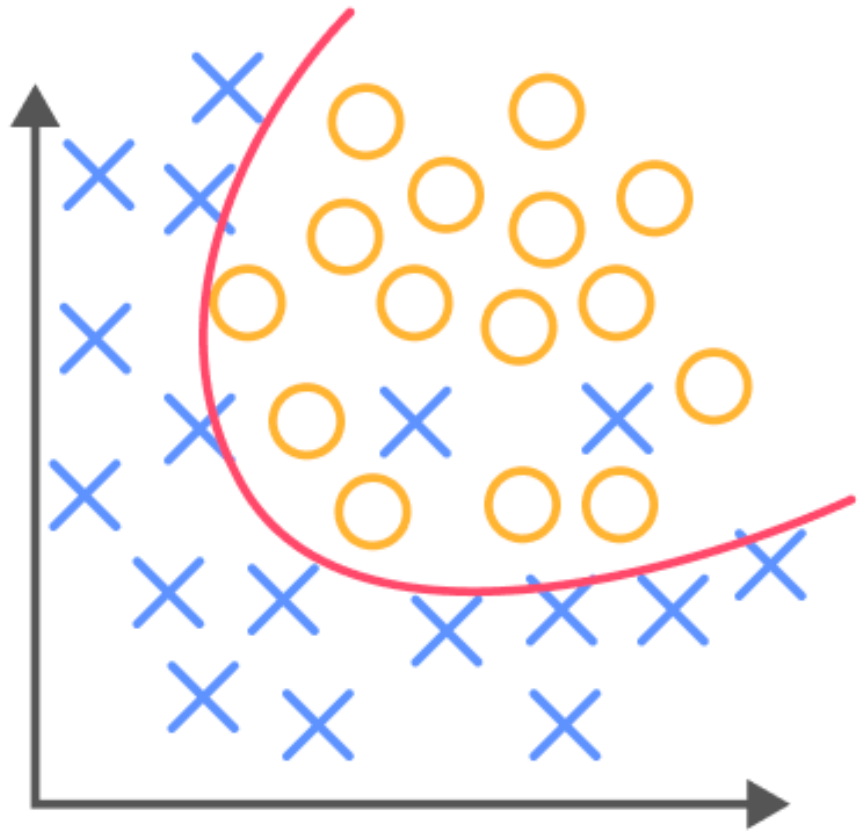
Parameter θ

Learned solver w/out convergence

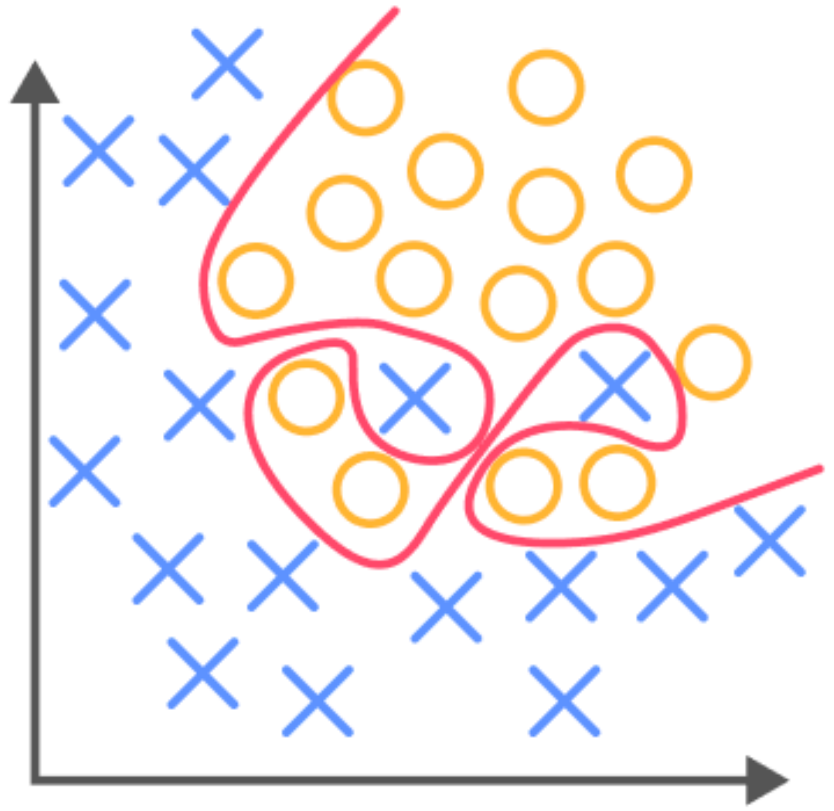
Poor candidate solution

L2O Challenge II: Generalization guarantees

To unseen problems



Appropriate fitting



Overfitting

L2O Challenge III: Incompatibility with state-of-the-art solvers

Existing state-of-the-art solvers are highly optimized

Written in low-level languages

OSQP
UNIVERSITY OF OXFORD

SCS
SPLITTING CONIC SOLVER

Clarabel.jl

GUROBI
OPTIMIZATION

pyOptSparse

mosek

When learning the algorithm steps, we cannot use these solvers

Learning to Warm-Start Fixed-Point Optimization Algorithms

Collaborators

Georgina
Hall

INSEAD

Brandon
Amos

Meta

Bartolomeo
Stellato

PRINCETON
UNIVERSITY

Fixed-point optimization problems are ubiquitous

Parametric fixed-point problem: find z such that $z = T_\theta(z)$

Convex optimization

Problem

minimize $f_\theta(z)$

subject to $g_\theta(z) \leq 0$

→

Optimality conditions

(KKT conditions)

→

Fixed-point operator

Unconstrained, smooth convex optimization

Problem

minimize $f_\theta(z)$

Smooth, convex

→

Optimality conditions

$\nabla f_\theta(z) = 0$

→

Fixed-point operator

$T_\theta(z) = z - \alpha \nabla f_\theta(z)$

Many optimization algorithms are fixed-point iterations

Fixed-point iterations: $z^{i+1} = T_\theta(z^i)$

Initialize with z^0 (a warm-start)

Terminate when $f_\theta(z^i) = \|T_\theta(z^i) - z^i\|_2$ is small **Fixed point residual**

Example: Proximal gradient descent

$$\text{minimize} \quad g_\theta(z) + h_\theta(z)$$

Convex
Smooth

Convex
Non-smooth

Iterates $z^{i+1} = \textbf{prox}_{\alpha h_\theta}(z^i - \alpha \nabla g_\theta(z^i))$

$$\textbf{prox}_s(v) = \arg\min_x \left(s(x) + \frac{1}{2}\|x - v\|_2^2 \right)$$

Operator $T_\theta(z) = \textbf{prox}_{\alpha h_\theta}(z - \alpha \nabla g_\theta(z))$

Problem: limited iteration budget

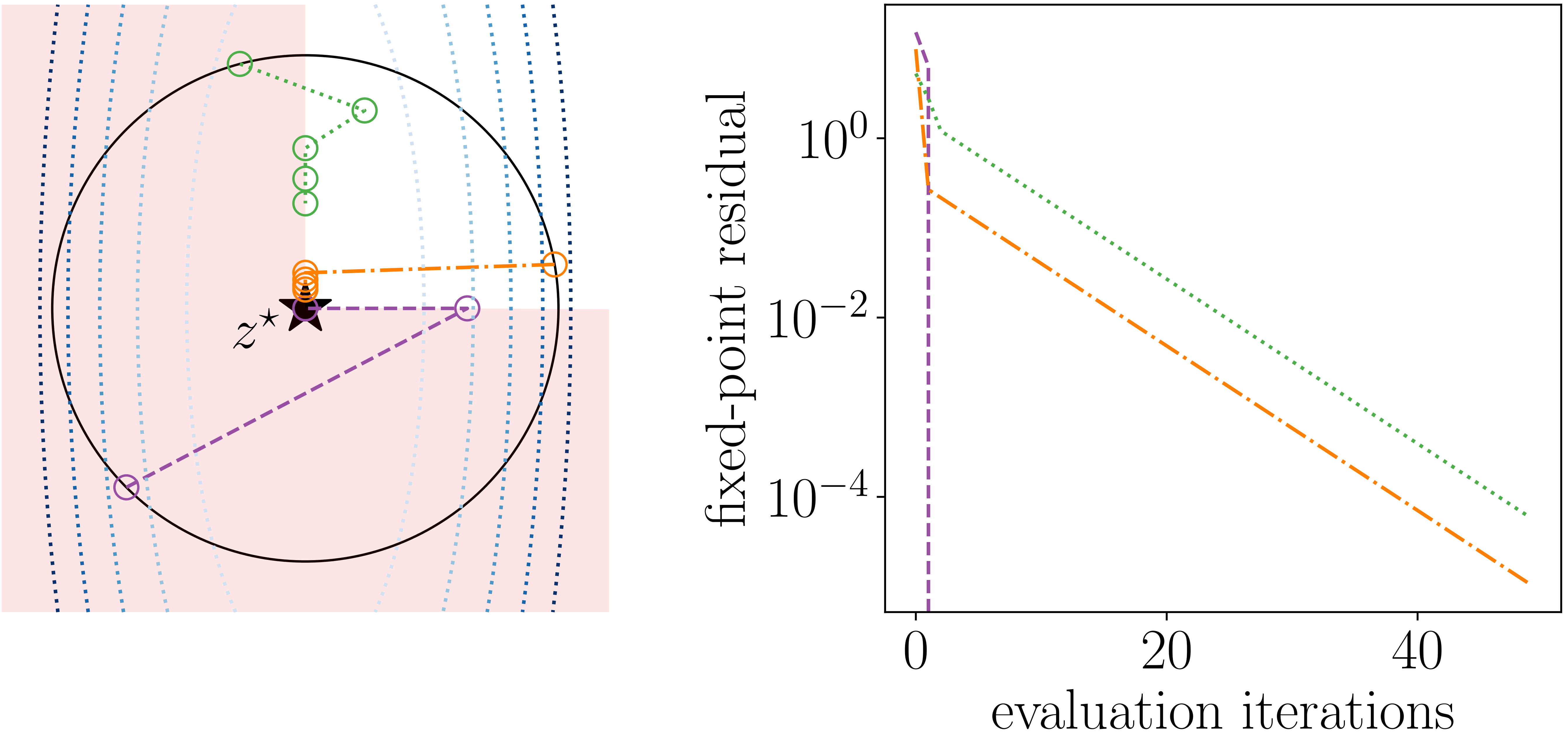
Solution: learn the warm-start to improve the solution within budget

Some warm-starts are better than others

$$\begin{array}{ll} \text{minimize} & 10z_1^2 + z_2^2 \\ \text{subject to} & z \geq 0 \end{array}$$

Optimal solution at the origin

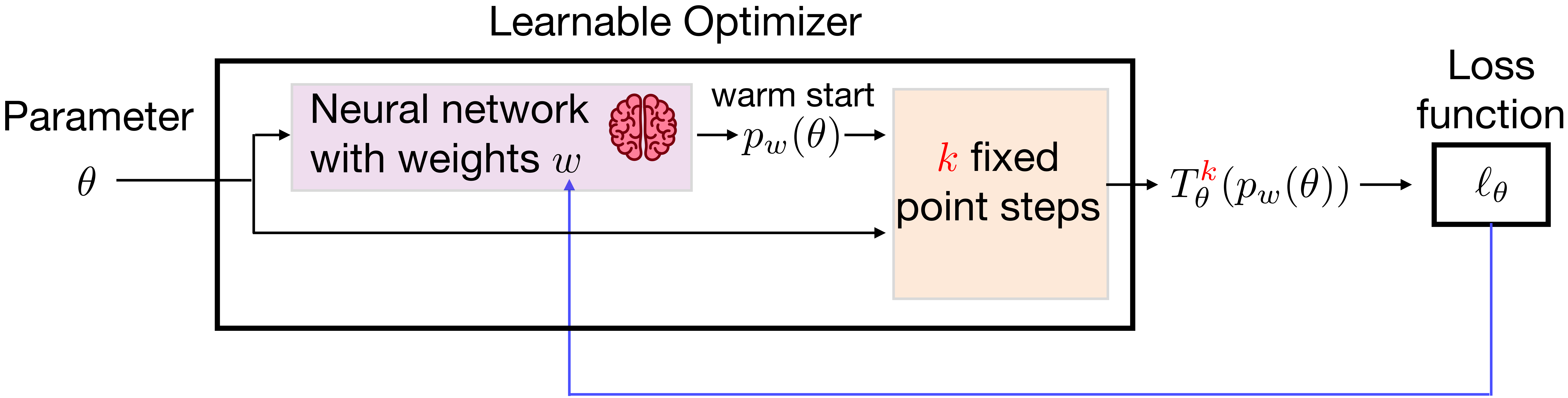
Run proximal gradient descent to solve



All three warm starts appear to be equally suboptimal but converge at very different rates

Learning Framework

End-to-end learning architecture



Learnable Optimizer

Parameter

θ

Neural network with weights w

warm start

$p_w(\theta)$

k fixed point steps

$T_\theta^k(p_w(\theta))$

Loss function

ℓ_θ

Learn with $\nabla_w \ell_\theta$ through the fixed point steps

Loss function: $\ell_\theta(z) = \|z - z^\star(\theta)\|_2$ Ground truth solution

Learned warm start tailored for downstream algorithm

An example architecture

minimize $\quad 1/2 z^T P z + \theta^T z$

subject to $\quad z \geq 0$

Projection onto $\mathbf{R}_+$

Step size

Fixed-point operator: $T_\theta(z) = \Pi_+\Big((I - \alpha P)z - \alpha\theta\Big)$

Computational Graph

🟩 🟢 **Linear components**

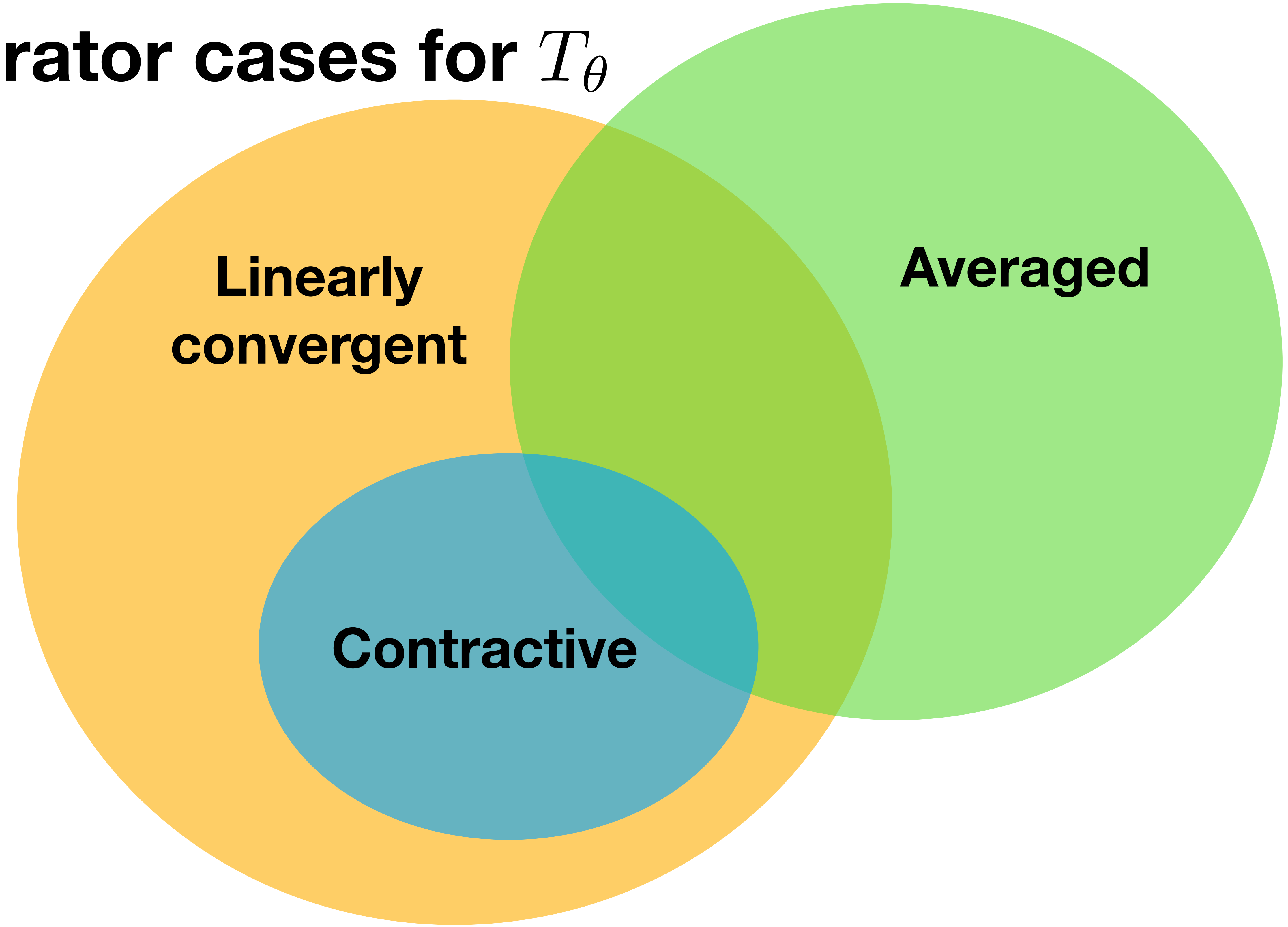
🟦 🔵 **Non-linear components**

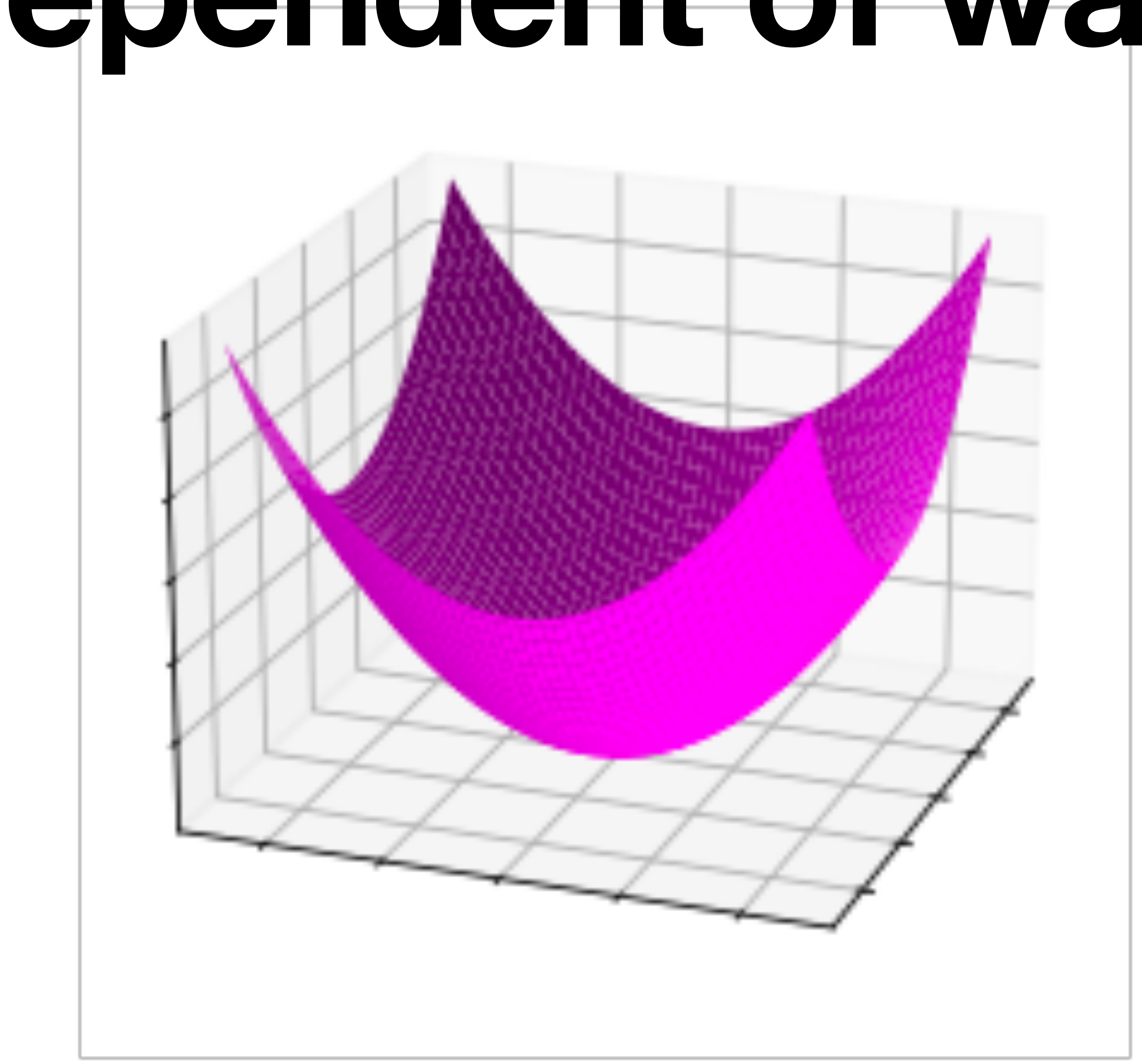
W_1, b_1, W_2, b_2 **Learnable components**

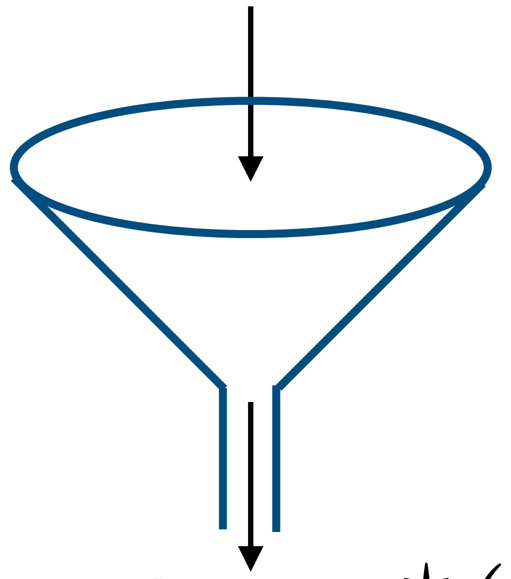
Convergence and Generalization Bounds

Guaranteed convergence independent of warm-start

Operator cases for T_θ







Parameter θ

Learned solver with convergence

Solution $z^\star(\theta)$

⚡ **Convergence always guaranteed for learned warm starts**

Generalization bounds: train for k, evaluate for t

Flexibility: # of evaluation steps can differ from # of train steps

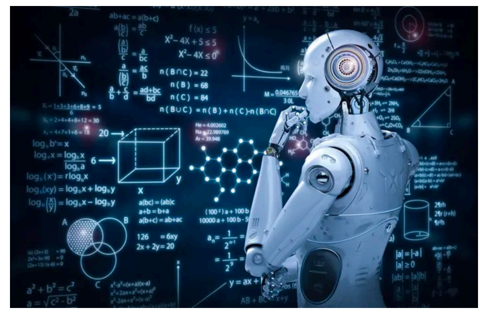
Number of fixed-point steps

Train with k steps

Loss: $\ell_\theta(z) = \|z - z^\star(\theta)\|_2$

Budget is t steps

Goal: min $f_\theta(z) = \|T_\theta(z) - z\|_2$



Guarantees from k training steps to t evaluation steps

β-contractive case $f_\theta(T_\theta^t(z)) \leq 2\beta^{t-k}\ell_\theta(T_\theta^k(z))$

Generalization bounds to unseen data

β-contractive case

Theorem 1. *With high probability over a training set of size N, for any γ,*

$$\underbrace{\mathbf{E} f_\theta(T_\theta^t(p_w(\theta)))}_{\text{Risk}} \leq \underbrace{\frac{1}{N}\sum_{i=1}^{N} f_{\theta_i}(T_{\theta_i}^t(p_w(\theta_i)))}_{\text{Empirical risk}} + \underbrace{2\beta^t\gamma + \mathcal{O}\left(c_1(t)\sqrt{\frac{c_2(w)+\log(\frac{LN}{\delta})}{\gamma^2 N}}\right)}_{\text{Penalty term}}$$

$c_1(t)$: worst-case fixed-point residual after t steps

As $N \to \infty$, the **penalty term** decreases

As $t \to \infty$, the **penalty term** goes to zero

Derived from the PAC-Bayes framework

Non-contractive case: we provide similar bounds

Learned warm-start can easily interface with solvers

Written in

OSQP

UNIVERSITY OF OXFORD

Quadratic programs

$$\begin{array}{ll} \text{minimize} & (1/2)x^T P x + c^T x \\ \text{subject to} & \ell \leq Ax \leq u \end{array}$$

```
sol = osqp_solver.warm_start(x=x0, y=y0)
```

SCS

SPLITTING CONIC SOLVER

Conic programs

$$\begin{array}{ll} \text{minimize} & (1/2)x^T P x + c^T x \\ \text{subject to} & Ax + s = b \\ & s \in \mathcal{K} \end{array}$$

```
sol = scs_solver.solve(warm_start=True,
                       x=x0, y=y0, s=s0)
```

We code exact replicas of OSQP and SCS

Allows us to make timing comparisons for QPs and conic programs

Numerical Experiments

We evaluate the gain over a cold-start

Baseline initializations

1. **Cold-start: initialize at zero** ❄
2. **Nearest neighbor: initialize with solution of nearest training problem**

Metrics plotted

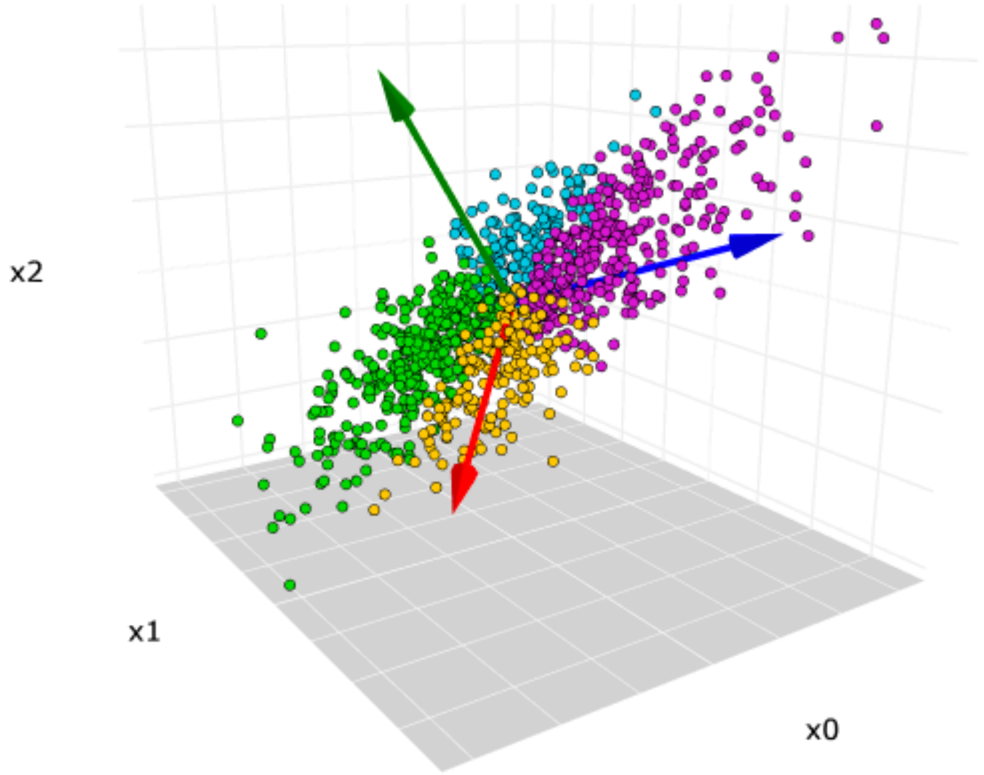
1. **Fixed-point residual**
2. **Gain over the cold-start**

$$\text{gain} = \frac{f_\theta(T_\theta^t(0))}{f_\theta(T_\theta^t(p_w(\theta)))}$$

Cold-start ❄

Learned warm-start 🧠

Sparse PCA



Non-convex problem

$$\begin{array}{ll} \text{maximize} & x^T A x \\ \text{subject to} & \|x\|_2 \leq 1 \\ & \textbf{Card}(x) \leq c \end{array}$$

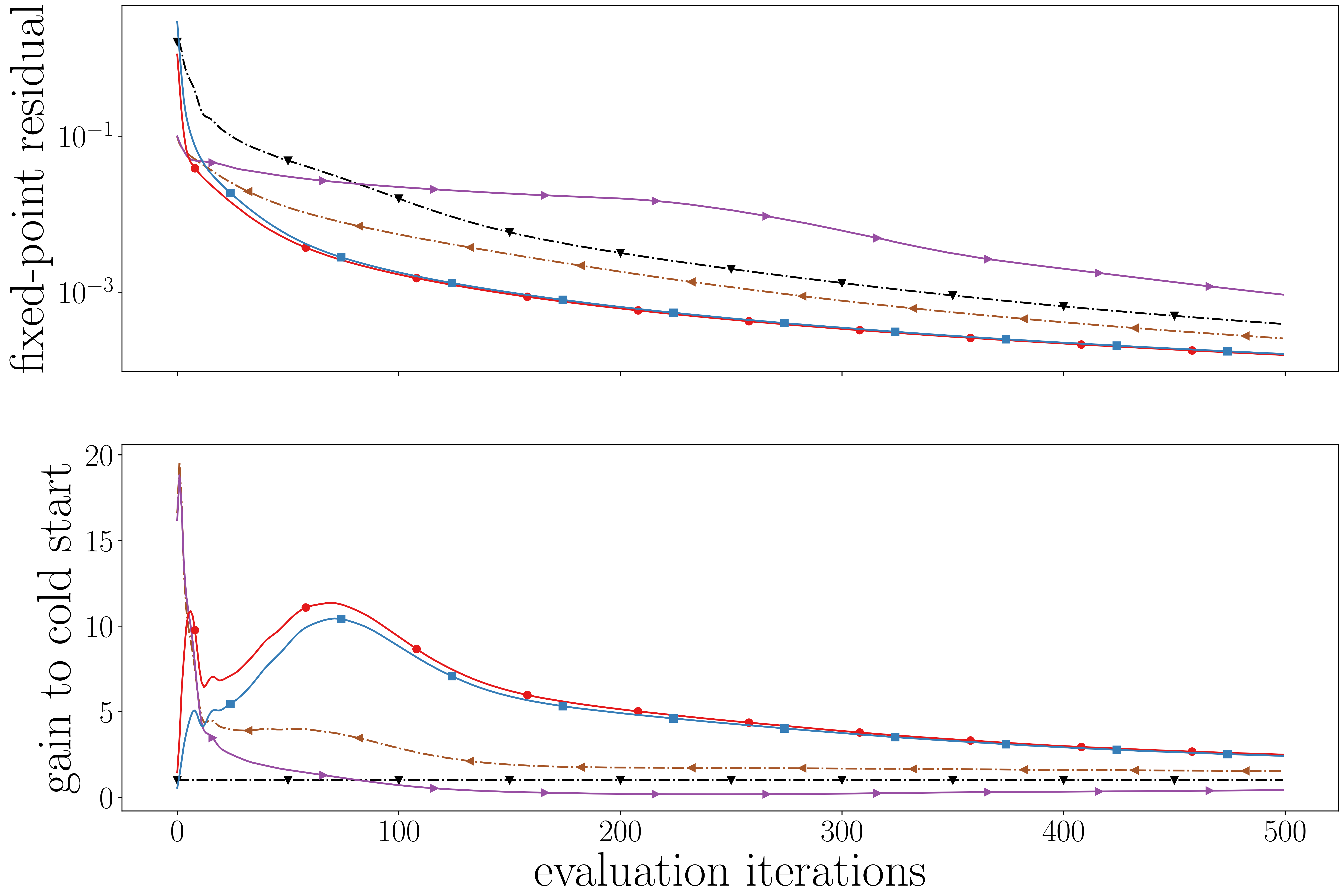
Covariance matrix → **Sparse PCA** → **Sparse principal component**

$\theta = \text{vec}(A)$ → **Semidefinite program**

$$\begin{array}{ll} \text{maximize} & \mathbf{Tr}(AX) \\ \text{subject to} & \mathbf{Tr}(X) = 1 \\ & \mathbf{1}^T |X| \mathbf{1} \leq c \\ & X \succeq 0 \end{array}$$

→ $X^\star$

Sparse PCA results



Different initializations

Baselines

▼ Cold-start

◀ Nearest neighbor

Learned

▶ $k = 0$

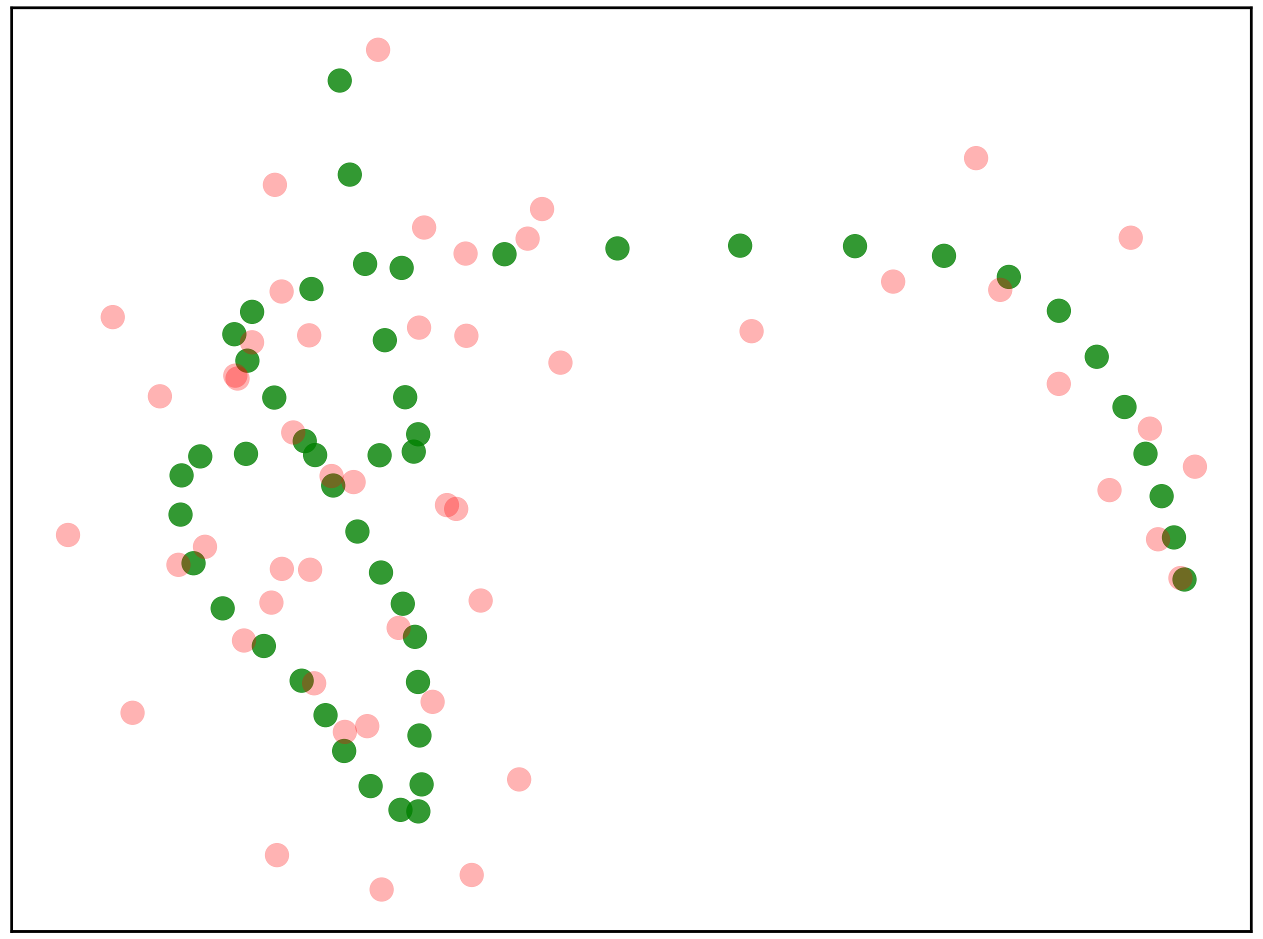
● $k = 5$

■ $k = 15$

Picking k > 0 is essential to improve convergence

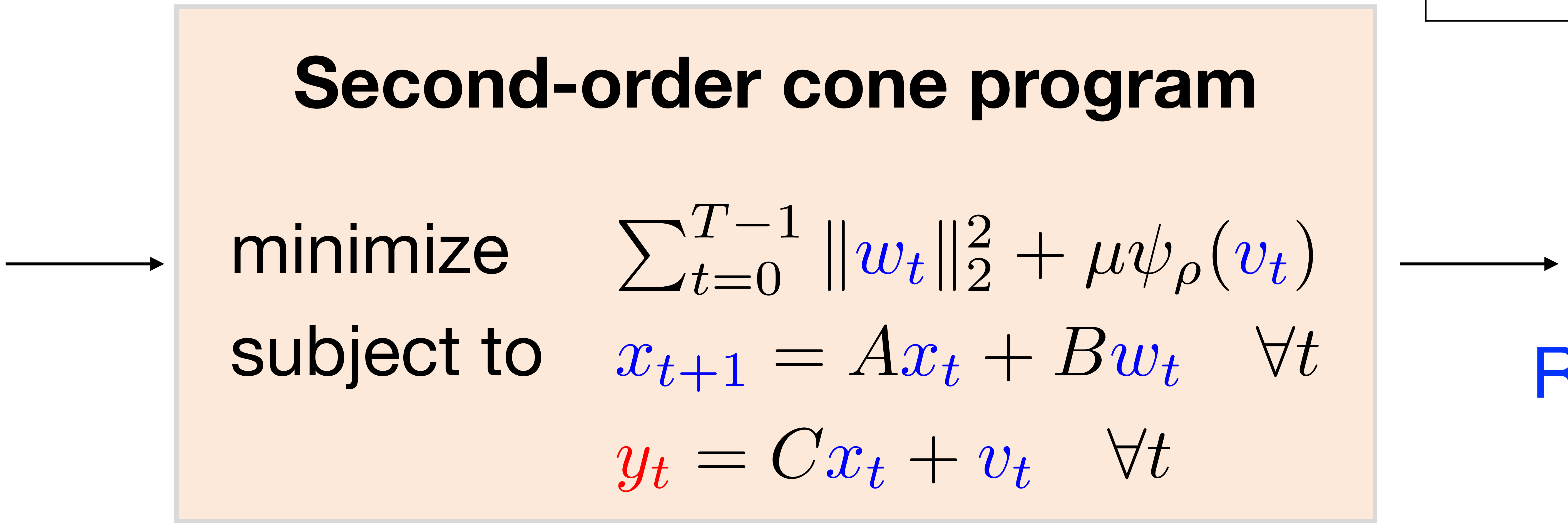
Robust Kalman filtering


Robust Kalman filtering



$\theta = \{y_t\}_{t=0}^{T-1}$

Noisy trajectory


Second-order cone program

minimize $\sum_{t=0}^{T-1} \|w_t\|_2^2 + \mu\psi_\rho(v_t)$

subject to $x_{t+1} = Ax_t + Bw_t \quad \forall t$

$y_t = Cx_t + v_t \quad \forall t$

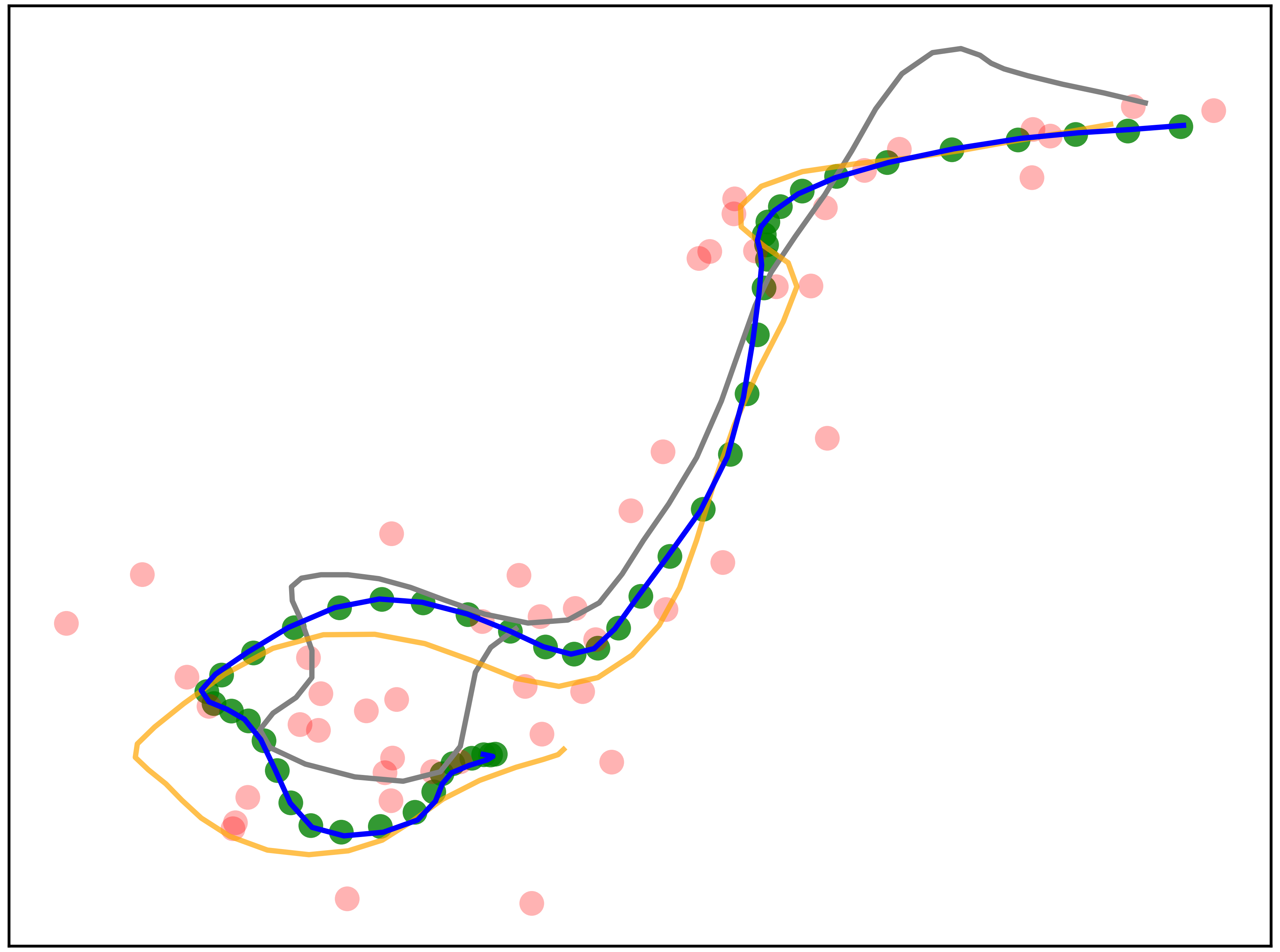
$\{x_t^\star, w_t^\star, v_t^\star\}_{t=0}^{T-1}$

Recovered trajectory

Dynamics matrices: A, B

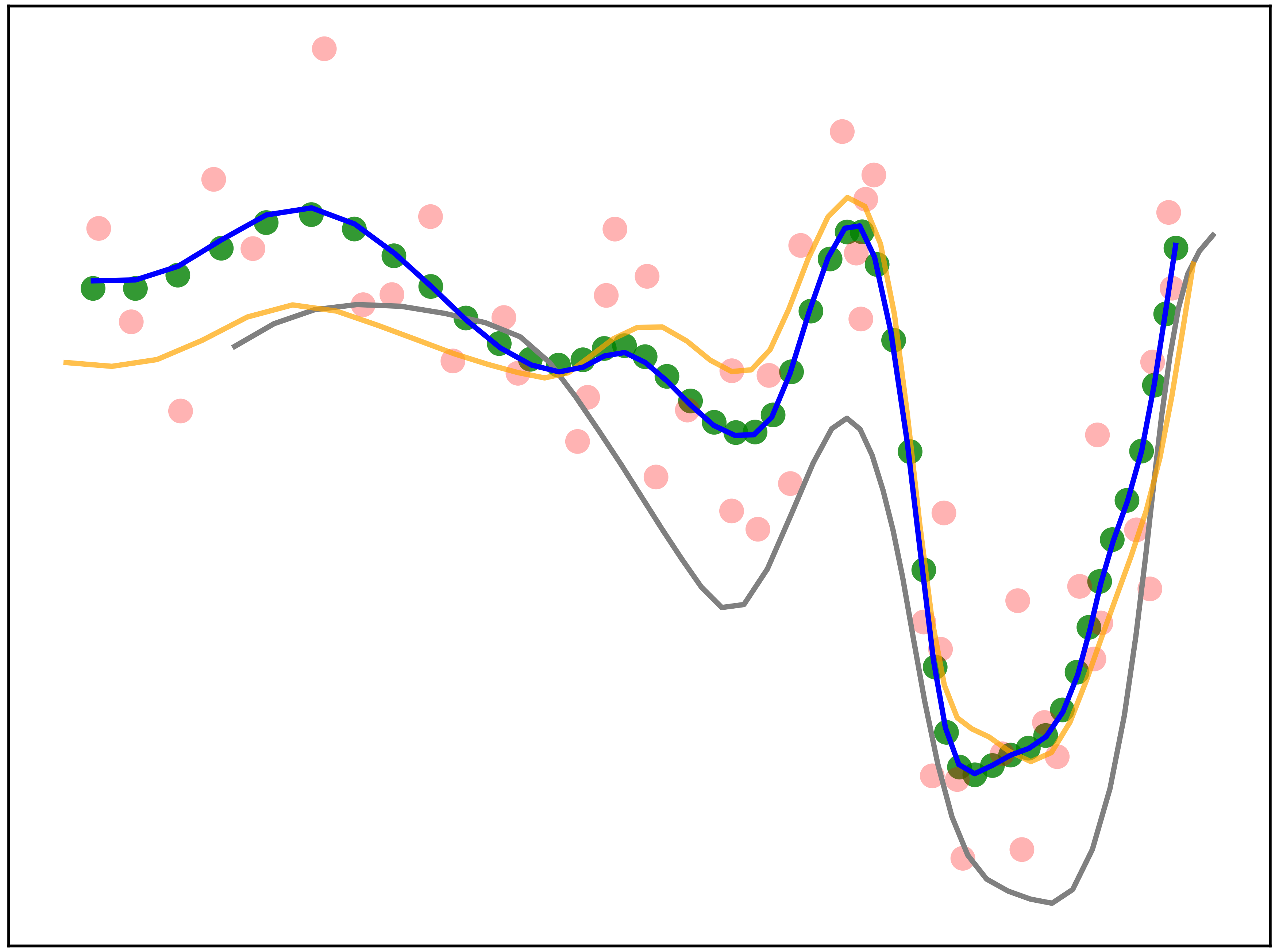
Observation matrix: C

Robust Kalman filtering visuals



🟥 Noisy trajectory

🟩 Optimal solution



Solution after 5 fixed-point steps
with different initializations

🟨 Nearest neighbor

◼ Previous solution

🟦 Learned: $k = 5$

With learning, we can estimate the state well

Model predictive control (MPC) of a quadcopter

Current state,
previous control
reference trajectory

→

Controller

→

Optimal
controls

$\theta = (x_{\text{init}}, u_{\text{prev}}, \{x_t^{\text{ref}}\}_{t=1}^{T})$ →

Quadratic program

minimize $\sum_{t=1}^{T}(x_t - x_t^{\text{ref}})^T Q (x_t - x_t^{\text{ref}}) + \sum_{t=0}^{T-1} u_t{}^T R u_t$

subject to

$$x_{t+1} = A x_t + B u_t$$

$$u_{\text{min}} \leq u_t \leq u_{\text{min}}$$

$$x_{\text{min}} \leq x_t \leq x_{\text{max}}$$

$$|u_{t+1} - u_t| \leq \Delta u$$

$$x_0 = x_{\text{init}}$$

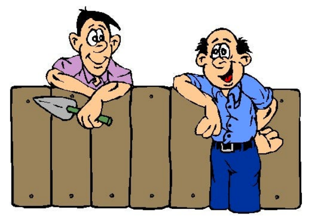
$$u_{-1} = u_{\text{prev}}$$

→ $\{x_t^\star, u_t^\star\}_{t=0}^{T}$

Linearized dynamics

MPC of a quadcopter in a closed loop

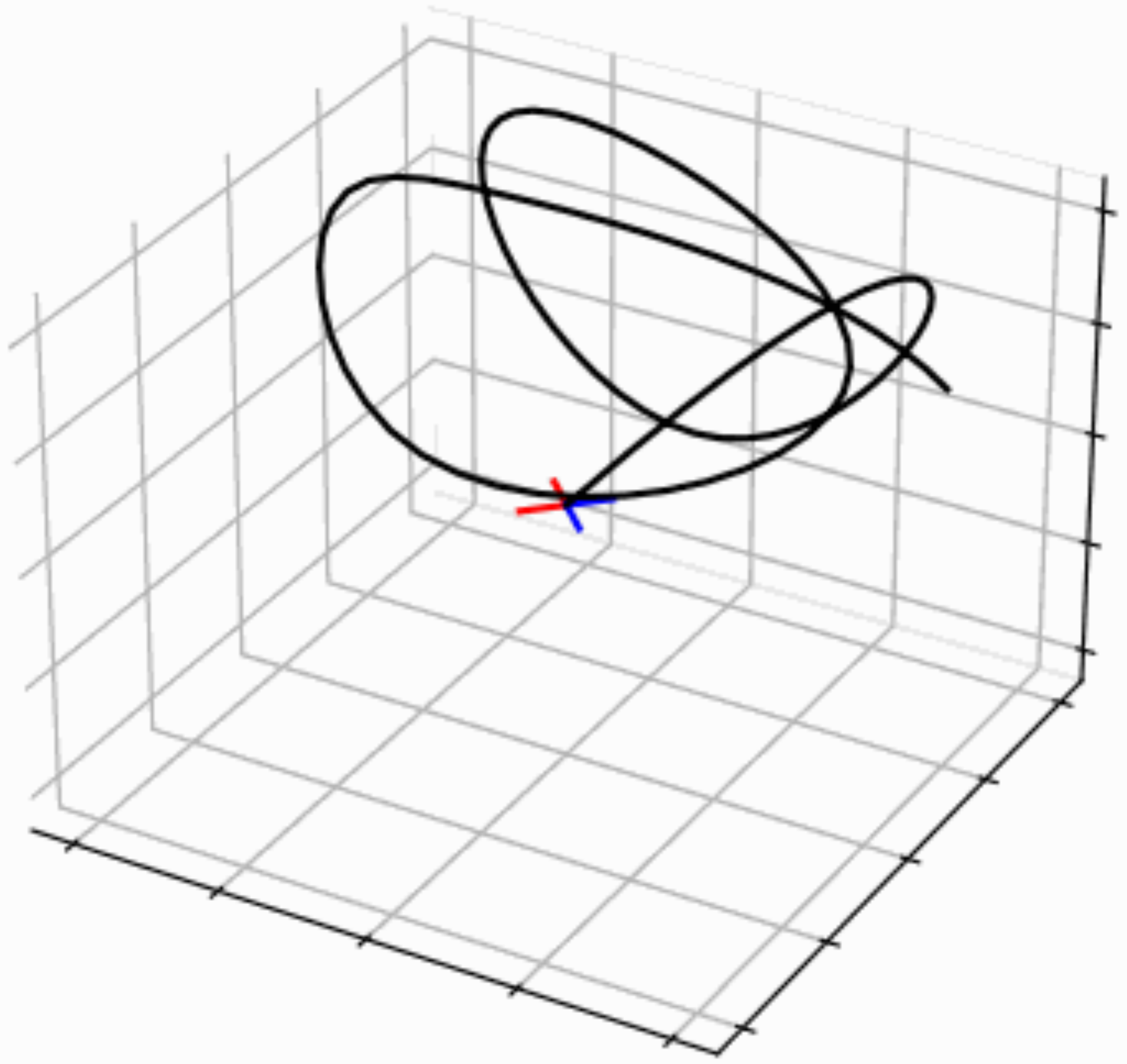
Budget of 15 fixed-point steps

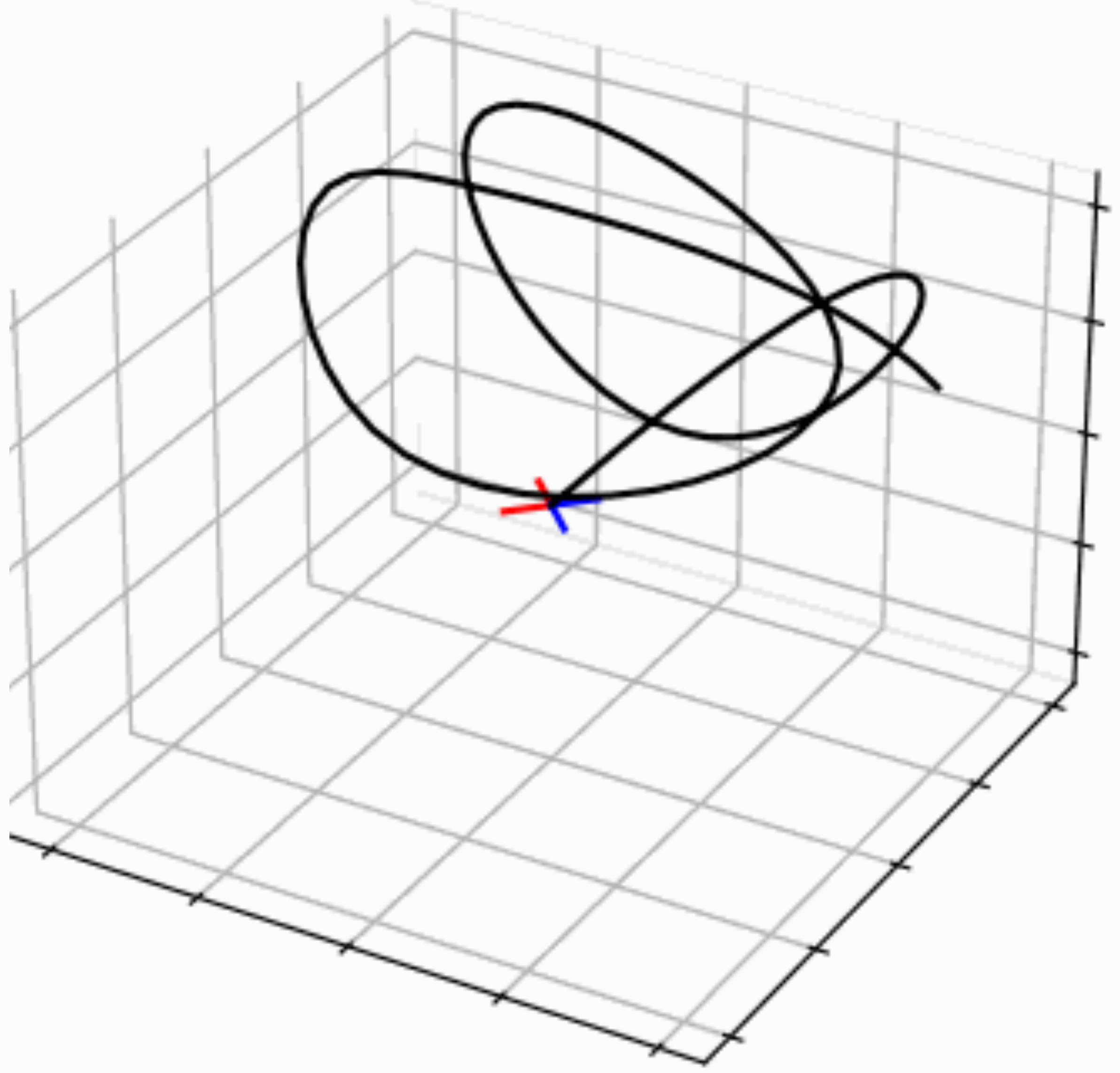


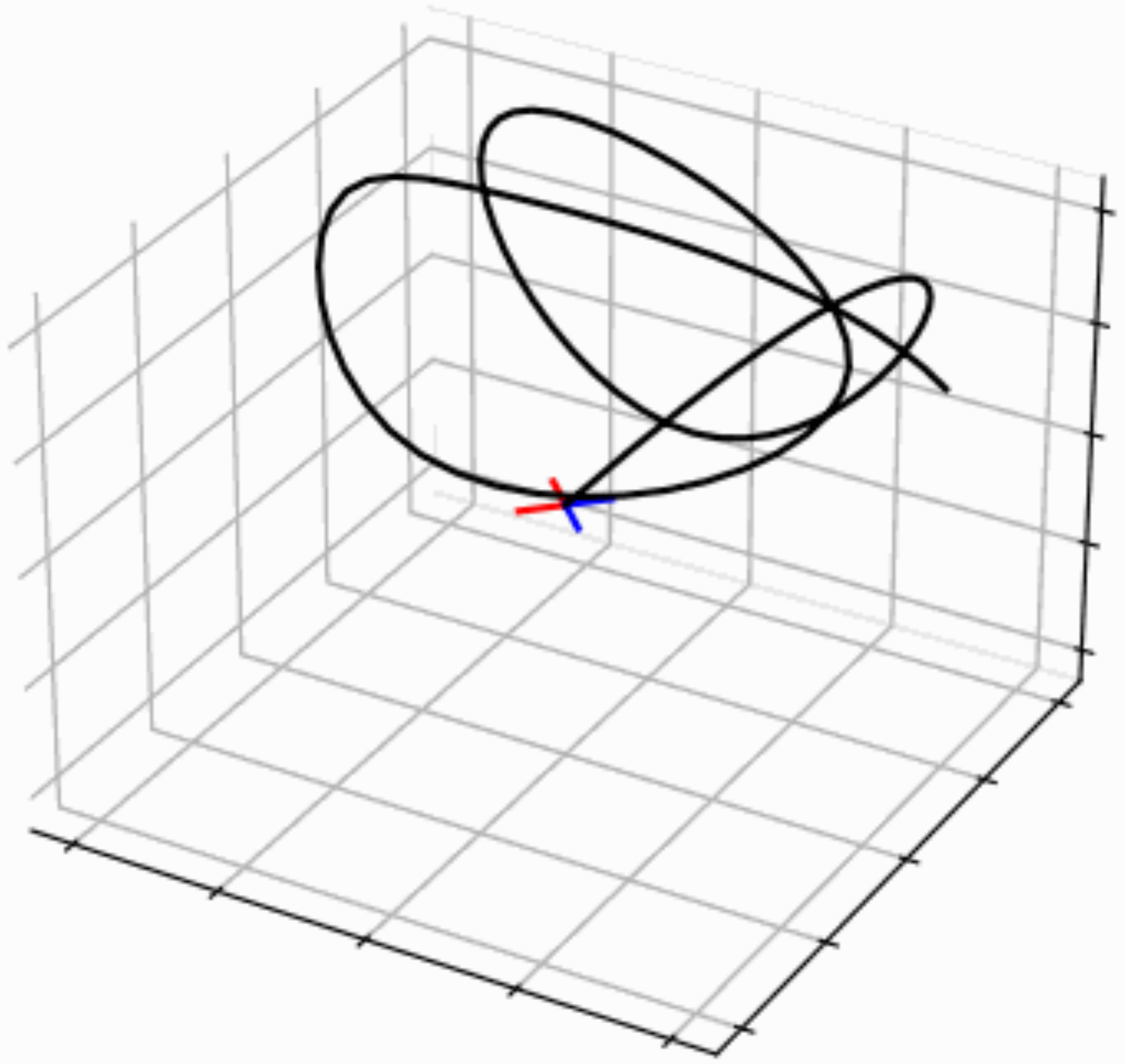
Nearest neighbor

Previous solution

Learned: $k = 5$







With learning, we can track the trajectory well

Image deblurring

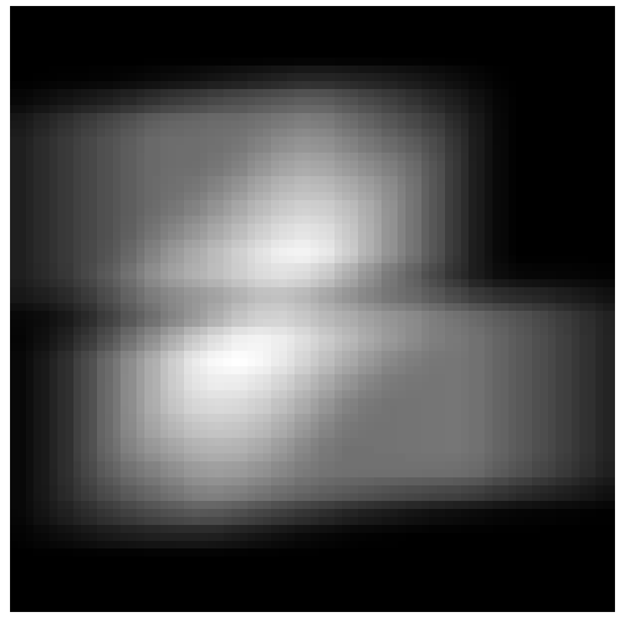


Image deblurring

$\theta = b$

Blurred image

Quadratic program

minimize $\|Ax - b\|_2^2 + \lambda \|x\|_1$

subject to $0 \leq x \leq 1$

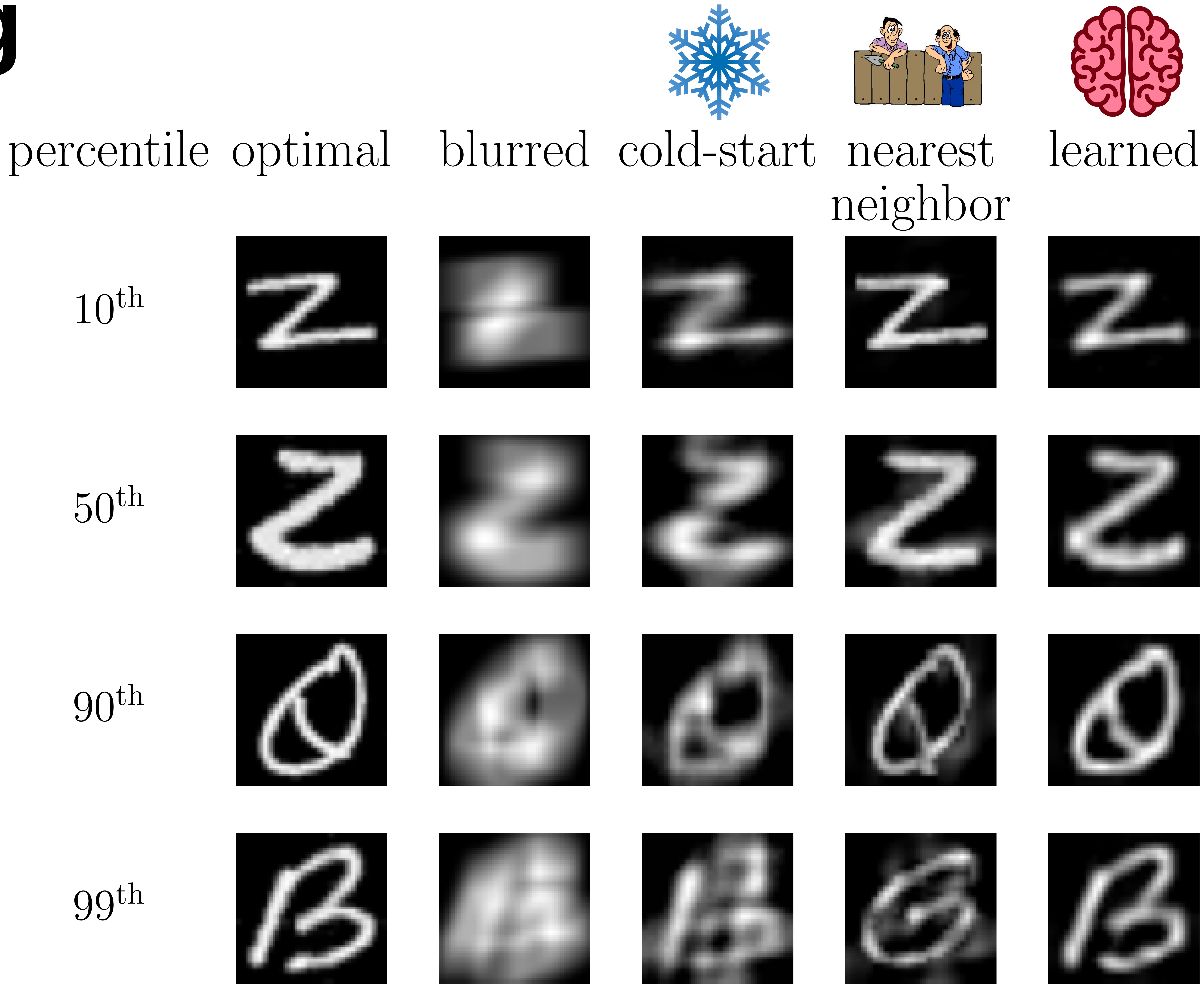
$x^\star$

Deblurred image

Image deblurring

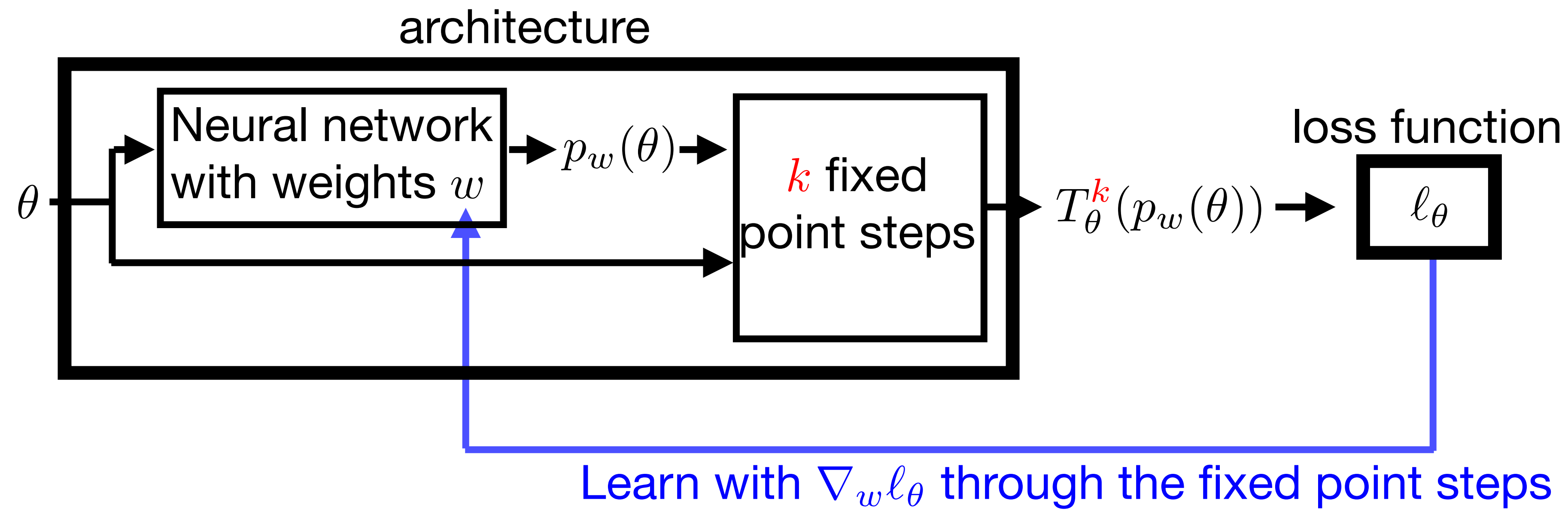
50 fixed-point steps

Distance to nearest neighbor increases



With learning, we can deblur all of the images quickly

Benefits of our learning framework



architecture

θ

Neural network with weights w

$p_w(\theta)$

k fixed point steps

$T_\theta^k(p_w(\theta))$

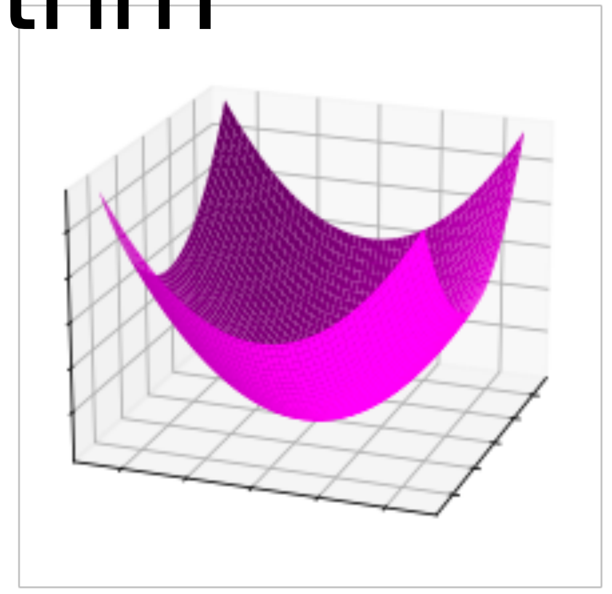
loss function

ℓ_θ

Learn with $\nabla_w \ell_\theta$ through the fixed point steps

End-to-end learning: warm-start predictions tailored to downstream algorithm

Guaranteed convergence



Can interface with state-of-the-art solvers

OSQP
UNIVERSITY OF OXFORD

Quadratic programs

SCS
SPLITTING CONIC SOLVER

Conic programs

Generalization to

Future iterations
Unseen data

rajivs@princeton.edu

rajivsambharya.github.io

arXiv

https://arxiv.org/pdf/2309.07835.pdf