LEARNING SURROGATE MODELS FOR OPTIMIZATION AND CONTROL

Alberto Bemporad

imt.lu/ab

Funded by the European Union

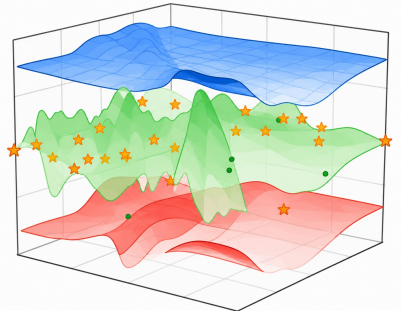
European Research Council
Established by the European Commission

IMT SCHOOL FOR ADVANCED STUDIES LUCCA

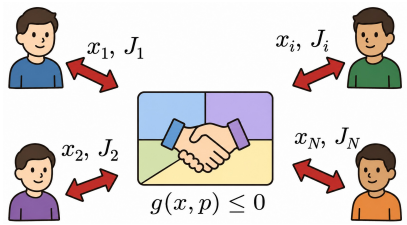
IFAC'26 - Busan - August 23, 2026

CONTENTS OF MY TALK

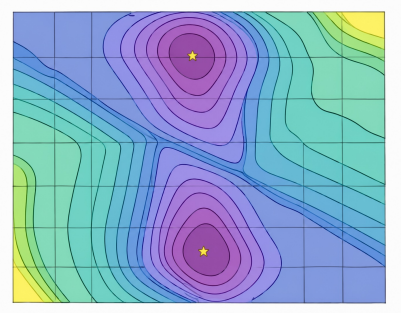
1. An **active learning** method for **worst-case nonlinear regression** with guaranteed **error bounds**



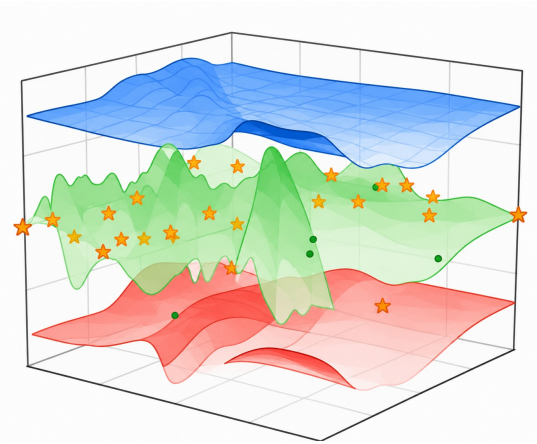
2. **Learning** approximate solutions of **multiparametric generalized Nash equilibrium problems** (and multiparametric nonlinear programming)



3. A **learning-based** approach to build **min-of-convex surrogates** of **parametric nonconvex functions** to ease optimization



WORST-CASE NONLINEAR REGRESSION



Contents based on the following paper:

A. Bemporad, "*Worst-case Nonlinear Regression with Error Bounds*," arXiv preprint 2601.12334, 2026

WORST-CASE NONLINEAR REGRESSION

- In **learning for control**, having a small **worst-case error** (WCE) matters more than a small mean-squared error (MSE):
  - **Model uncertainty**, treated as a bounded disturbance in robust control design
  - Overapproximation of a **constraint set**, to guarantee constraint satisfaction
  - Approx error in **explicit model predictive control** (MPC) laws = input disturbance

- **Goal**: given $f : \mathcal{X} \to \mathbb{R}$ over a compact set $\mathcal{X} \subset \mathbb{R}^n$, fit a **surrogate** $\hat{f}(x;\theta) \in \mathcal{F}$ that minimizes the WCE over the **entire** set $\mathcal{X}$, and quantify it

$$\theta^\star \in \arg\min_\theta \left(r(\theta) + \max_{x\in\mathcal{X}} |f(x) - \hat{f}(x;\theta)| \right)$$

- **Assumption**: f can be **evaluated at any point** (not just on a fixed dataset)

TWO DIFFICULTIES

(i) **Bilevel** optimization, due to the inner maximization over $\mathcal{X}$:

$\Rightarrow$ replace $\mathcal{X}$ by a **finite training set** $\{x_k\}_{k=1}^{N_0}, y_k = f(x_k)$:

$$\theta^\star \in \arg\min_\theta \left(r(\theta) + \max_{k=1,\dots,N_0} |y_k - \hat{f}(x_k;\theta)| \right)$$

$\Rightarrow$ **actively learn** a minimal number of samples x_k to solve the original problem

(ii) The max operator is **nonsmooth**

$\Rightarrow$ replace it with the **log-sum-exp smoothing** of the max operator

$$\theta^\star \in \arg\min_\theta \left(r(\theta) + \frac{1}{\gamma} \log \left(\sum_{k=1}^{N_0} \left(e^{\gamma(y_k - \hat{f}(x_k;\theta))} + e^{\gamma(\hat{f}(x_k;\theta) - y_k)} \right) \right) \right)$$

which can be solved efficiently, e.g., by **L-BFGS** + automatic differentiation

(same problem for $\gamma \to +\infty$)

ACTIVE-LEARNING ALGORITHM

- Collect $N = N_0$ samples $(x_k, f(x_k))$ **randomly** in $\mathcal{X}$
- Repeat until max error $e_N \leq \epsilon_{\text{err}}$ or no more budget:
  - **Fit** $\hat{f}(\cdot;\theta)$ on current dataset and get θ_N
  - Find the point x_{N+1} of **largest error** e_N by **global optimization**:

$$x_{N+1} \in \arg\max_{x\in\mathcal{X}} |f(x) - \hat{f}(x;\theta_N)| =: e_N$$

  - Add $(x_{N+1}, f(x_{N+1}))$ to the training set, increase N
- Select best model $\theta^\star = \theta_N$ with the smallest max global error e_N

```
pip install maxfit
```

MaxFit

GUARANTEES UNDER INEXACT GLOBAL OPTIMIZATION

- The bound $\bar{e}^\star = \max_{x\in\mathcal{X}} |f(x) - \hat{f}(x;\theta^\star)|$ is valid only if the global optimization is solved **exactly**

- **Proposition**: if f and $\hat{f}(\cdot;\theta^\star)$ Lipschitz continuous $\Rightarrow \forall \delta > 0, \exists T$ such that the **DIRECT** global solver (Jones et al., 1998) achieves

$$\max_{x\in\mathcal{X}} |f(x) - \hat{f}(x;\theta^\star)| \leq \bar{e}_T + \delta$$

where $\bar{e}_T$ is the best value found after T iterations

POSTPROCESSING: TIGHTER, INPUT-DEPENDENT ERROR BOUNDS

- Besides the uniform bound $\bar{e}^\star$, we can compute **asymmetric** constant bounds $\bar{e}^\star_{\min}, \bar{e}^\star_{\max} \leq \bar{e}^\star$ by global optimization of the **one-sided errors**

- **Even tighter**: learn an **envelope function** $\varepsilon(x;\psi)$ (e.g., a small NN) that upper-bounds $\{|e_k|\}$ as tightly as possible and set

$$\bar{e}^\star_{\min}(x) = \bar{e}^\star_{\max}(x) = \min(\bar{e}^\star, \kappa^\star \varepsilon(x;\psi^\star))$$

  where $\kappa^\star$ is globally-optimized to **rescale** ε and make it valid $\forall x \in \mathcal{X}$

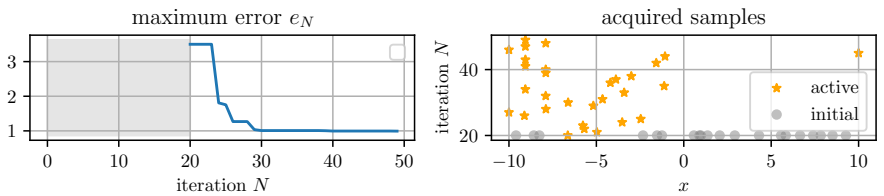
- Related to **conformal prediction**, but $\kappa^\star$ is set by **worst-case** global optimization over $\mathcal{X}$ rather than by calibration on a held-out dataset

- Extends to **asymmetric** input-dependent upper/lower bounds using **two envelope functions** $\varepsilon^\ell(x;\psi^\star), \varepsilon^u(x;\psi^\star)$

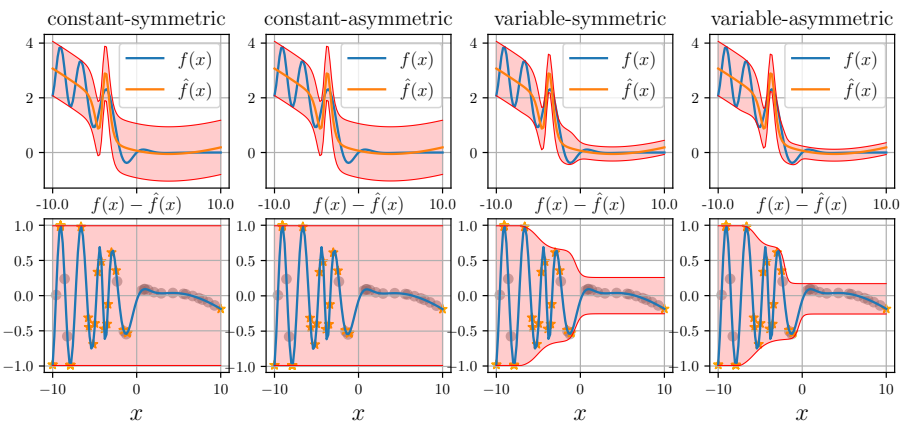
ILLUSTRATIVE EXAMPLE: A SCALAR FUNCTION

- Approximate $f(x) = \left(\sin(x - \frac{x^2}{10}) + (\frac{x}{10})^3 - \frac{4x}{10}\right)\frac{e^{-x}}{1+e^{-x}}, x \in [-10, 10]$ with a **tiny** NN ($\tanh$, layers of 2 and 1 neurons)

- Run the algorithm with $N_0 = 20$ initial samples, max budget $M = 30$ samples



Max error e_N decreases as corner cases are actively added



Learned model with **uniform** and **input-dependent** envelope bounds

EMBEDDING KNOWN STRUCTURE

- **Exact match on a set**: to promote $\hat{f}(x;\theta) \equiv w(x)$ whenever $g(x) \leq 0$ and $h(x) = 0$, we train the **blended** model

$$\hat{f}(x;\theta) = \hat{\delta}(x;\beta)w(x) + (1-\hat{\delta}(x;\beta))\hat{v}(x;\theta) \quad \beta \gg 1$$

where $\hat{\delta}(x;\beta) = e^{-\beta\left(\sum_{i=1}^{n_g}\max\{g_i(x),0\}+\sum_{j=1}^{n_h}|h_j(x)|\right)}, \hat{\delta} \in (0,1)$

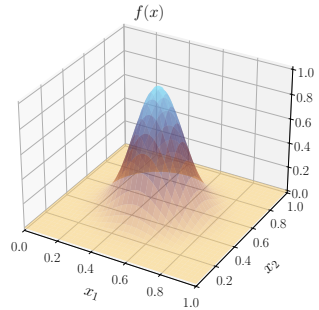
- **Saturation**: use **smooth saturation** function to bound $\hat{f}$ in $[y_{\min}, y_{\max}]$:

$$\text{sat}_\eta(y; y_{\min}, y_{\max}) = y_{\max} + \frac{1}{\eta}\log\left(\frac{1+e^{-\eta(y-y_{\min})}}{1+e^{-\eta(y-y_{\max})}}\right) \quad \eta \gg 1$$

- **Example**: **approximate explicit MPC**, with $\{g(x) \leq 0\}$ = unconstrained region, $w(x)$ = unconstrained solution, and $y_{\min}, y_{\max}$ = actuator saturation limits

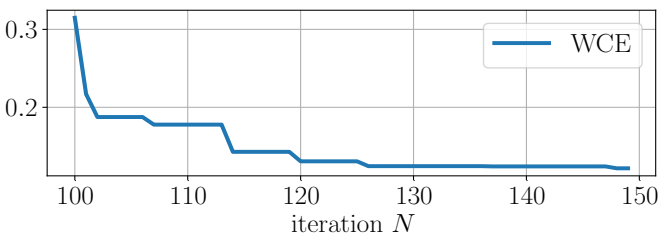
EXAMPLE: APPROXIMATING A NONLINEAR FUNCTION

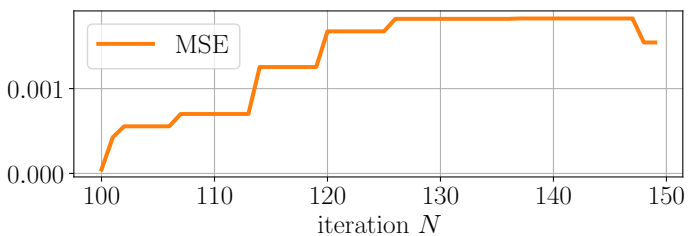
- **PWA approximation** of the **Gaussian function**
  $f(x) = e^{-30((x_1-0.5)^2+(x_2-0.5)^2)}$ on $\mathcal{X} = [0,1]^2$

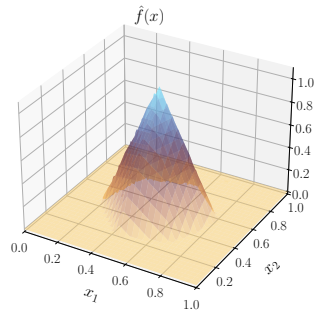


- $\hat{f}(x;\theta)$ = NN with two layers of 10 and 5 **leaky-ReLU** neurons

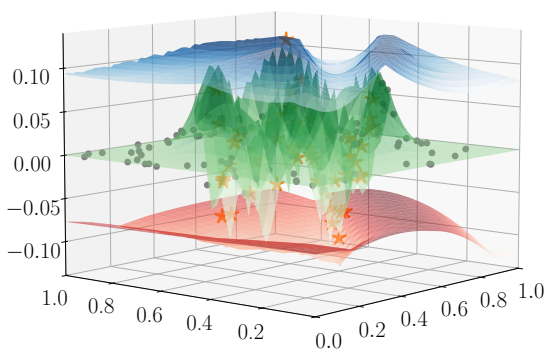
- Active learning from $N_0 = 100$ initial samples, $M = 50$ steps:
  114.3 s total (106.0 s training, 7.8 s global optimization)







uncertainty interval



- **Note**: minimizing the WCE does **not** minimize the MSE!

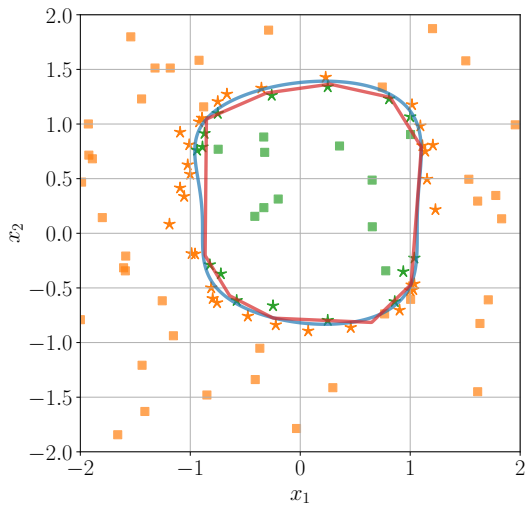
- **Envelope bound function**:
  leaky-ReLU NN with two layers of 20 and 10 neurons

EXAMPLE: CONVEX INNER APPROXIMATION OF A SET

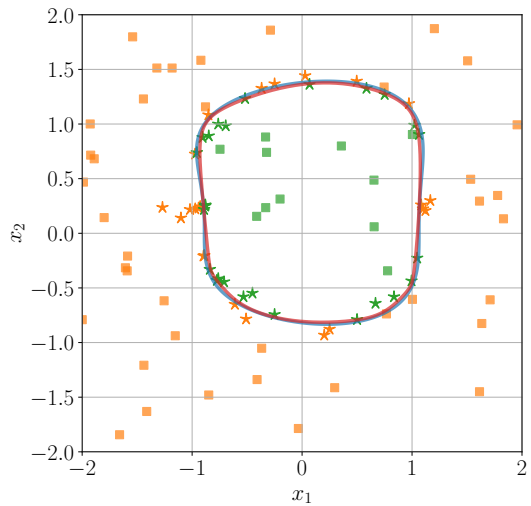
- Replace $\{f(x) \leq 0\}$ with a **simpler convex set** $\{\bar{f}(x) \leq 0\}$ such that $\bar{f}(x) \leq 0 \Rightarrow f(x) \leq 0, \forall x \in \mathcal{X}$, as tight as possible

- **Worst-case regression**: fit smooth surrogate of sign function to $\text{sign}(f(x))$

- **Shift** the learned model by its worst-case offset Δf to guarantee the implication



PWA function

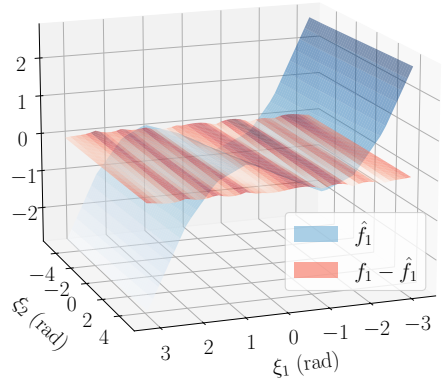


input-convex NN

EXAMPLE: UNCERTAIN NONLINEAR MODEL

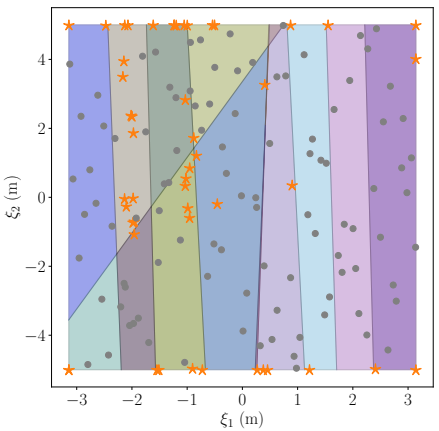
- Continuous-time **pendulum model** w/ stiffnes + friction:

$$J\ddot{\xi} = u - b\dot{\xi} - mg\ell_c \sin\xi - k_1\xi - k_3\xi^3$$



- **Goal**: learn a discrete-time model $(\xi, \dot{\xi}, u) \mapsto (\xi^+, \dot{\xi}^+)$ with a **guaranteed additive uncertainty** bound $\mathcal{W}$ ($T_s = 0.1$ s)

- **Fit** an NN with 10 ReLU neurons plus a linear term in u, ground truth from Heun integration ($\Rightarrow$ PWA model)



- **Active learning** from $N_0 = 100$ LHS samples, $M = 50$ steps

- **Result**: $\mathcal{W} = [-0.0810, 0.0810] \times [-2.6368, 2.6368]$
(= 2.82% / 5.67% maximum relative error on the two states)

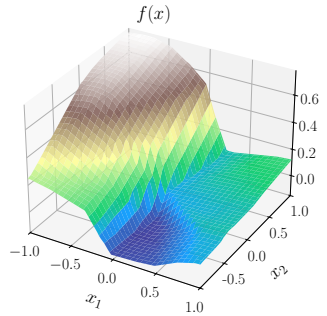
EXAMPLE: APPROXIMATE EXPLICIT MPC

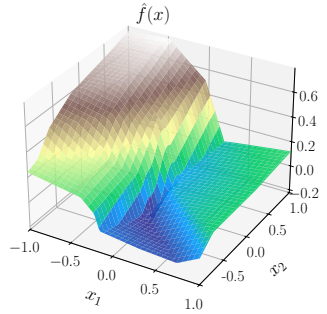
- Exact **explicit MPC** laws $f(x)$ for linear systems are PWA, but the number of regions **grows fast** with the number of constraints (**multiparametric-QP** solution)
  (2 params, 10 vars, 30 inequalities $\rightarrow$ f has 162 regions)

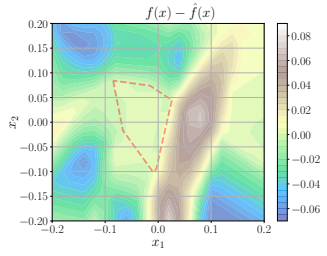
- Learn **NN ReLU (5,5)** $\hat{f}$ + saturation that also matches the **unconstrained QP solution** in unconstrained region $H_0 x \leq K_0$,
  $\hat{\delta}(x;\beta) = \max\left(1 - \beta \max\left(H_0 x - K_0, 0\right), 0\right)$ (β trainable)

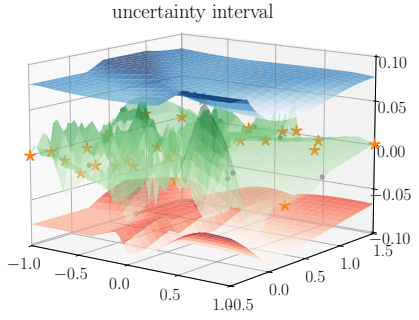
- The resulting WCE bound is a **guaranteed additive disturbance** on the input: $u(x) = \hat{f}(x) + e(x)$ with $e^{\ell}(x) \leq e(x) \leq e^{u}(x)$

- **Active learning** vs **standard regression**:
  10^5 random samples $\rightarrow$ WCE ≈ 0.400
  active learning $\rightarrow$ WCE $= 0.065$, with comparable MSE



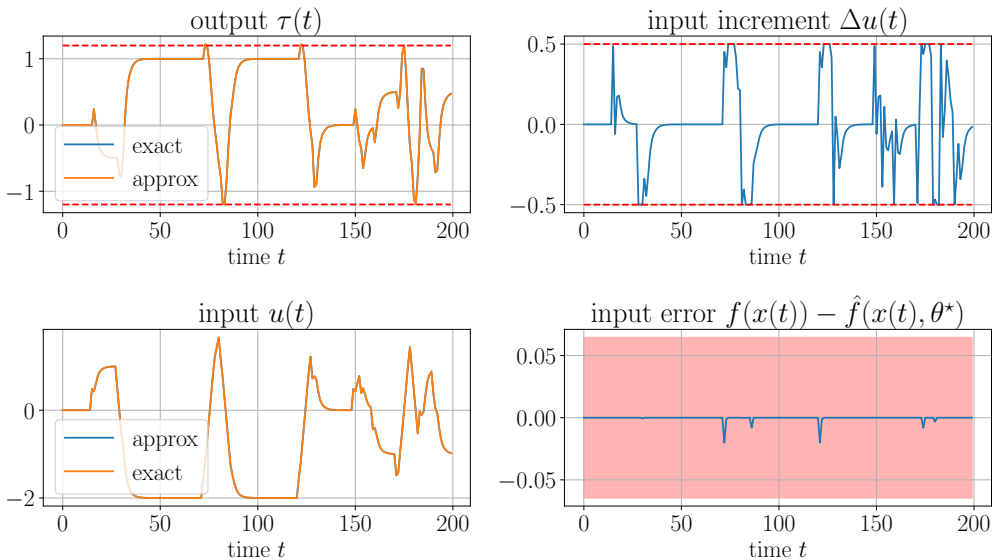






EXAMPLE: APPROXIMATE LINEAR MPC

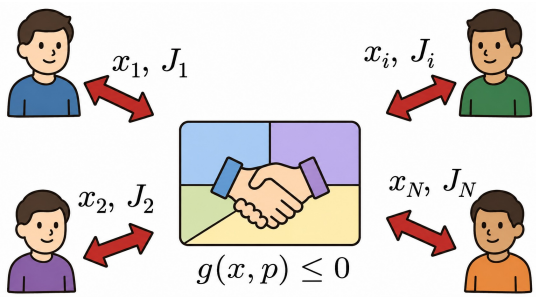
- **Linear MPC** of non-minimum-phase SISO system
  ($N = N_u = 20$, 80 inequalities, 4 parameters (2 states, reference, previous input)
- Same NN surrogate as before + saturation to enforce Δu bounds
- $M = 1000$ active learning steps from $N_0 = 1000$ initial random samples



WCE $\bar{e}^\star \approx 0.065$

- **Without active learning** (10^5 random samples): WCE $\bar{e}^\star \approx 0.398$

LEARNING MULTIPARAMETRIC (GNE) SOLUTIONS



Contents based on the following paper:

A. Bemporad, T. Tatarenko, "*Learning approximate solutions to multiparametric generalized Nash equilibrium problems*," arXiv preprint 2605.28757, 2026

MULTIPARAMETRIC GENERALIZED NASH EQUILIBRIUM PROBLEM

- N agents, agent #i decides $x_i \in \mathbb{R}^{n_i}$ given x_{-i} and a **parameter** $p \in \mathcal{P}$ (**best response**, measurable)

$$\bar{x}_i(x_{-i}, p) \in \arg\min_{x_i} \; J_i(x_i, x_{-i}, p) \;\; \text{s.t. } g(x,p) \le 0,\; h(x,p) = 0$$

- Constraints g, h are **shared** by all agents $x = (x_i, x_{-i})$ (assume local constraints on x_i are embedded in g, h)

- **Value function**: $\bar{J}_i(x_{-i}, p) = J_i(\bar{x}_i(x_{-i}, p), x_{-i}, p)$ (measurable)

- **Goal**: for every $p \in \mathcal{P}$, find a **generalized Nash equilibrium** (GNE) $x^\star(p) = (x_1^\star(p), \dots, x_N^\star(p))$

$$x_i^\star(p) = \bar{x}_i(x_{-i}^\star(p), p) \qquad i = 1, \dots, N$$

- Neither **convexity** nor **monotonicity** assumed on the game!

LEARNING WITHOUT SOLVING GNES

- **Standard regression**: train a NN $\hat{x}(p;\theta)$ on a dataset of **solved GNEs** $\{(p_k, x^\star(p_k))\}$
  - Computing many GNEs for different p_k is **expensive**
  - Multiple (or no) GNEs may exist for the same p_k: **which one** to put in the dataset?

- **Key fact**: $x^\star(p)$ solves the GNEP $\iff$ the Nikaido-Isoda (NI) gap function $= 0$

$$NI(x,p) = \sum_{i=1}^{N} \Big(J_i(x_i, x_{-i}, p) - J_i(\bar{x}_i(x_{-i}, p), x_{-i}, p) \Big) \geq 0$$

over the feasible set $\{x:\ g(x,p) \leq 0,\ h(x,p) = 0\}$

- NI only requires (cheaper) **best-response** data $\bar{x}_i$, not full GNE solutions

$\Rightarrow$ use $NI(x,p)$ as a training loss!

TWO PARAMETRIC MODELS

- **GNE (solution) model**: calibrate model params θ_2 such that $\hat{x}(p,\theta_2) \approx x^\star(p)$

- Training $\hat{x}$ directly against NI would require **bilevel optimization** to evaluate the best response $\bar{J}_i(\hat{x}_{-i}(p_k,\theta_2),p_k)$ for the NI terms during training

$$\nu_i(p_k,\theta_2) = J_i(\hat{x}_i(p_k,\theta_2),\hat{x}_{-i}(p_k,\theta_2),p_k) - \bar{J}_i(\hat{x}_{-i}(p_k,\theta_2),p_k)$$

- **Value-function model**: first, fit $\hat{J}_i(x_{-i},p,\theta_{1i}) \approx \bar{J}_i(x_{-i},p), i=1,\dots,N$ on **best-response** data $\{(x_{-i,k},p_k,\bar{J}_i(x_{-i,k},p_k))\}$

$$\theta_{1i}^\star = \arg\min \frac{1}{M}\sum_{k=1}^{M} \left\| \hat{J}_i(x_{-i,k},p_k,\theta_{1i}) - \bar{J}_{k,i} \right\|_2^2 + r_{1i}(\theta_{1i})$$

then train $\hat{x}$ against the **approximate** NI function $\hat{\nu}_i(p_k,\theta_2)$, where $\bar{J}_i$ is approximated by $\hat{J}_i \Rightarrow$ **single-level optimization** wrt θ_2

SINGLE-LEVEL LEARNING PROBLEM

- **GNE model training**: minimize the (approximate) NI gap over the training set subject to feasibility:

$$\min_{\theta_2} \frac{1}{M}\sum_{k=1}^{M} \ell_{NI}(\hat{\nu}(p_k,\theta_2))$$
$$\text{s.t. } g(\hat{x}(p_k,\theta_2),p_k)\le 0,\ \ h(\hat{x}(p_k,\theta_2),p_k)=0,\ \ \forall k$$

where $\ell_{NI}(\hat{\nu}) = J_i(\hat{x}_i(p_k,\theta_2),\hat{x}_{-i}(p_k,\theta_2),p_k) - \underbrace{\hat{J}_i(\hat{x}_{-i}(p_k,\theta_2),p_k,\theta^\star_{1i})}_{\text{value fcn surrogate}}$

- **Feasibility** on all M samples: rewrite as the single scalar equality

$$\max_k \left(\max_j \underbrace{\max(g_j(\hat{x}(p_k,\theta_2),p_k),0)}_{\text{inequality violation}},\ \max_t \underbrace{|h_t(\hat{x}(p_k,\theta_2),p_k)|}_{\text{equality violation}} \right) = 0$$

SMOOTH CONSTRAINT PENALTY

- Penalize violation of the max-constraint via **log-sum-exp smoothing** ($\beta, \gamma \gg 1$):

$$\ell^c_{\beta,\gamma}(\theta_2) = \frac{\beta}{\gamma} \log\left(\sum_k \left(\sum_j e^{\gamma \max(g_j(\hat{x}(p_k,\theta_2),p_k),0)} + \sum_t e^{\gamma|h_t(\hat{x}(p_k,\theta_2),p_k)|}\right)\right)$$

- Yields the **unconstrained** GNE learning problem

$$\theta_2^\star = \arg\min_{\theta_2} \ \ell^c_{\beta,\gamma}(\theta_2) + r_2(\theta_2) + \frac{1}{M}\sum_{k=1}^{M} \ell_{NI}(\hat{\nu}(p_k,\theta_2))$$

that we solve by **L-BFGS** via automatic differentiation

- β trades off **feasibility** vs deviation from best responses (e.g., $\beta = 100$)

CHOICE OF THE NI-BASED LOSS

- Nikaido-Isoda (NI) gap terms:

$$\hat{\nu}_i(p_k, \theta_2) = J_i(\hat{x}_i(p_k, \theta_2), \hat{x}_{-i}(p_k, \theta_2), p_k) - \hat{J}_i(\hat{x}_{-i}(p_k, \theta_2), p_k, \theta_1^\star)$$

- Possible choices of the NI-based loss $\ell_{NI}(\hat{\nu})$:

  (a) **Sum**: $\ell_{NI}(\hat{\nu}) = \sum_{i=1}^{N} \hat{\nu}_i$. For small β, the model may become **super-optimal** (i.e., $\hat{\nu}_i < 0$, due to exploiting the imperfect $\hat{J}_i$ to beat the true best response)

  (b) **Hinge**: $\ell_{NI}(\hat{\nu}) = \sum_{i=1}^{N} \max(\hat{\nu}_i, 0)$. Only penalize **positive** gaps

  (c) **Smooth hinge**: $\ell_{NI}(\hat{\nu}) = \frac{1}{2} \sum_{i=1}^{N} \left(\hat{\nu}_i + \sqrt{\hat{\nu}_i^2 + \epsilon} \right)$. Differentiable surrogate of hinge loss to make optimization easier (best choice in our experiments)

SPECIAL CASE: STANDARD MULTIPARAMETRIC PROGRAMMING

- For $N = 1$ (or decoupled agents with no shared constraints), the problem is

$$x^\star(p) \in \arg\min_x \; J(x,p) \;\; \text{s.t. } g(x,p) \leq 0,\; h(x,p) = 0$$

- The NI gap term reduces to $\nu(p_k, \theta_2) = J(\hat{x}(p_k,\theta_2), p_k) - J(x^\star(p_k), p_k)$
- The term $J(x^\star(p_k), p_k)$ is **constant**, so no value-function model is needed!
- Learning problem = **jointly minimize** the cost over all samples, under the constraint penalty:

$$\theta_2^\star = \arg\min_{\theta_2} \; \ell^c_{\beta,\gamma}(\theta_2) + r_2(\theta_2) + \frac{1}{M}\sum_{k=1}^{M} J(\hat{x}(p_k,\theta_2), p_k)$$

- **Note**: **no optimization at all** need to create the dataset, which is just $\{p_k\}$

WHY CONTINUOUS NN MODELS ARE JUSTIFIED

- We take $\hat{x}_i$ and $\hat{J}_i$ as feedforward **neural networks** (other choices are possible)
- True value function $\bar{J}_i(x_{-i}, p)$ is **convex** and **continuous** if J_i, g, h are jointly convex (classical parametric-optimization result)
- **GNE map** $x^\star(p)$: restrict to **variational** GNEs (v-GNE), solutions of a variational inequality $VI(F(\cdot, p), X(p))$ with pseudo-gradient $F = (\partial J_1/\partial x_1, \ldots, \partial J_N/\partial x_N)$
- **New sufficient condition**: **strong variational stability** (SVS), generalizing strong monotonicity

$$\langle F(x,p), x - x^\star(p)\rangle \geq \nu \|x - x^\star(p)\|^2, \ \forall x \in X(p),\ p \in \mathcal{P}$$

- **Proposition**: SVS + mild projection regularity $\Rightarrow$ a **continuous** v-GNE selection $x^\star(p)$ exists, even for **some non-monotone** games

TRAINING PIPELINE AND SOFTWARE

- **Dataset**: p_k by Latin hypercube sampling in $\mathcal{P}$; x_k obtained by projecting a random point onto the shared feasible set for p_k (least-distance QP); compute $\bar{J}_{k,i}$ by solving each agent's best response (BR)
- **Training**: Adam (1000 epochs) + L-BFGS (2000 iterations), 32 random restarts, model selected by lowest validation loss
- **At prediction time** (optional): a projection step or an output saturation enforces hard box constraints
- **Metrics**: BR error MSE_{BR}; average/worst-case constraint violation $\bar{v}$, $v_{\max}$

```
pip install mpfit
```

mpfit

```
pip install nashopt
```

NashOpt

EXAMPLE: LINEAR-QUADRATIC GNEPS

- 2-agent **strongly-monotone** LQ-GNEP, costs quadratic and affine in p
  5 shared inequalities, and $-1 \leq x_1, x_2 \leq 1, p \in [-1, 1]^2$

- $\hat{J}_i$: NN $(30, 20)$ swish; $\hat{x}$: NN $(10, 5)$ ReLU + linear bypass; $M = 1000$ train/val/test samples

NI loss (best β)	$\bar{v}$	$v_{\max}$	MSE_{BR}
sum, $\beta = 1000$	3.29e-06	2.11e-03	2.32e-02
hinge, $\beta = 10$	3.33e-05	1.33e-02	2.55e-03
smooth hinge, $\beta = 1000$	1.49e-06	1.49e-03	4.64e-02

v = constraint violation, BR = best response

- Training time ≈ 30 s, **prediction time** $\approx 0.08\ \mu$s

- Scaling to $N \in \{2, 3, 4\}$ agents, $n_p \in \{2, 3, 4\}$ parameters, $20N$ shared constraints: $\bar{v}$ as low as $2 \cdot 10^{-4}$, training time ≈ 250-900 s

EXAMPLE: A NON-MONOTONE LQ-GNEP

- 2-agent **not monotone** linear-quadratic (LQ) GNEP
- The GNEP satisfies the new **SVS** assumption $\Rightarrow$ a continuous v-GNE selection is guaranteed (here, piecewise affine in p)
- **Value-function** model: NN $(10, 5)$ tanh; **solution model**: NN $(5, 3)$ leaky-ReLU
- $M = 1000$ train/val/test samples

Result: $\text{MSE}_{\text{BR}} = 0.00423$, **zero** constraint violation on test data, training time 13.7 s

EXAMPLE: QUADRATICALLY-CONSTRAINED QUADRATIC GNEP

- $N = 3$ agents, 1 variable each, $n_p = 2$ parameters $-1 \leq p_1, p_2 \leq 1$

- Shared **linear** constraints plus 4 shared **convex quadratic** constraints, $-1 \leq x_i \leq 1$

- Architectures/regularization as in the LQ-GNEP example, $M = 1000$ samples

  Results: mean constraint violation $1.18 \cdot 10^{-4}$, worst-case $4.37 \cdot 10^{-2}$
  $\text{MSE}_{\text{BR}} = 0.132$

  Training time 47.6 s (33.1 s for the 3 value-function models)
  prediction time $\approx 0.114\ \mu$**s**

EXAMPLE: NONLINEAR GNEPS

- **Internet-switching game** (Facchinei, Kanzow, 2009): $J_i(x,p) = -\frac{x_i}{\sum_j x_j}\left(1 - \frac{\sum_j x_j}{p}\right)$, $\sum_i x_i \le p, x_i \ge \ell$

- J_i are convex. Analytical best response is available

- $N \in \{2,\dots,10\}$ agents; $\hat{J}_i$: NN $(10,5)$ swish; $\hat{x}$: NN $(2N,2N)$ ReLU

- Data uniformly sampled on the feasible set (from Dirichlet distribution), $1000(N-1)$ training and validation samples, $N_{\text{test}} = 1000$ test samples

- **Results**:

N	MSE_{BR}	training time (s)	$\bar{v}$, $v_{\max}$
2	6.06e-06	17.1	0, 0
5	9.52e-05	81.4	0, 0
10	1.78e-03	348.2	0, 0

EXAMPLE: MULTIPARAMETRIC PROGRAMMING ($N = 1$)

- **mpQP**: $n_x = 10$ vars, $n_p = 6$ params, 50 linear constraints
  Exact mpQP solver needs 1627 critical regions

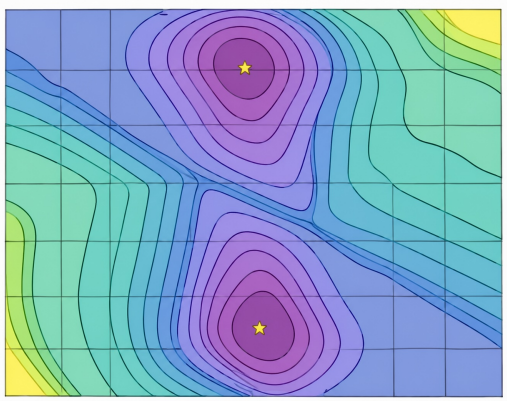
- **mpQCQP**: same problem $+20$ random convex quadratic constraints
  no exact explicit solution exists

- NN $(30, 20)$ ReLU with linear bypass, warm-started by regression on optimal solutions in the training set

	mpQP	mpQCQP
Avg. relative error	0.000807	0.00218
Avg. / worst-case violation	0.00306 / 0.171	0.00377 / 0.245
Training time (s)	130.3	269.7
CPU time $\hat{x}$ vs $x^\star$	0.16 μs / 0.45 ms	0.18 μs / 2.18 ms

$\Rightarrow$ more than $10^3\times$ **speedup** at inference time vs. solving the QP/QCQP online

LEARNING NONCONVEX SURROGATES



Contents based on the following paper:

R. Wang, P. Patrinos, A. Bemporad, "*Parametric nonconvex optimization via convex surrogates*," arXiv preprint 2604.05640, 2026

(to be presented at CDC'2026)

ONLINE NONCONVEX OPTIMIZATION

- **Goal**: solve **nonconvex** optimization problems **online** for varying parameters

$$\min_{x \in \mathcal{X}} f(x,p) \quad \text{s.t. } g(x,p) \leq 0 \qquad x \in \mathbb{R}^n,\ p \in \mathcal{P} \subset \mathbb{R}^{n_p}$$

f **nonconvex**, $g(\cdot, p)$ **convex** for fixed p, $\mathcal{X}$ convex compact

- We want to exploit parametric structure to solve new instances **faster** and possibly to **global optimality**

  $\Rightarrow$ **learning to optimize** / **amortized optimization** (Amos, 2023)

- **Example**: input-constrained nonlinear MPC, p = current state & reference

BEYOND LEARNING THE SOLUTION MAP

- **Fully-amortized** approach = learn $\hat{x}(p;\theta) \approx x^\star(p)$ (e.g., as described earlier)

  $\Rightarrow$ $\hat{x}(p;\theta)$ may be **infeasible**, unless the model is constrained by design

  $\Rightarrow$ $\hat{x}(p;\theta)$ may be **suboptimal**

- **Idea**: learn a **surrogate** $\hat{f}(x,p)$ of $f(x,p)$ that **can be minimized efficiently** at run time under $g(x,p) \leq 0$ to yield a good approximate solution $\hat{x}(p)$

  $\Rightarrow$ $\hat{f}(x,p)$ must capture the **nonconvexity** of $f(x,p)$

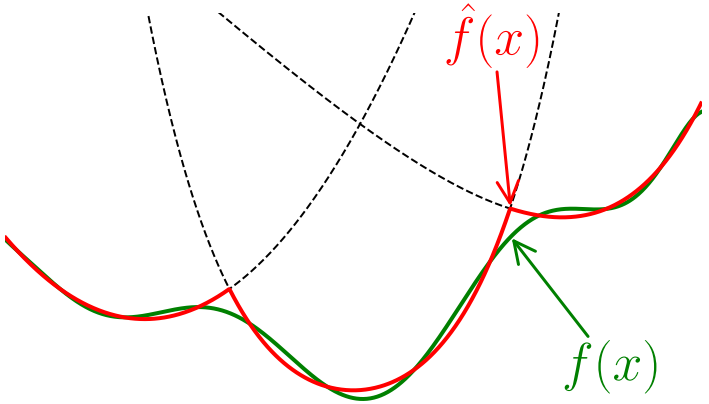
  $\Rightarrow$ $\hat{f}(x,p)$ must be **easy to minimize** under $g(x,p) \leq 0$

MIN-OF-CONVEX SURROGATE

- **Main idea**: parameterize the surrogate $\hat{f}$ as the **minimum** of **convex** functions

$$\hat{f}(x,p) = \min_{i=1,\dots,K} \hat{f}_i(x,p)$$

$\hat{f}_i(\cdot,p)$ **convex** in x



- **More generally**: $\hat{f}_i$ = composition of a **convex** function with a **monotonic** one:

$$\hat{f}(x,p) = \min_{i=1,\dots,K} h_i\big(\bar{f}_i(x,p)\big) \qquad h_i : \mathbb{R} \to \mathbb{R} \quad \text{monotonically increasing}$$

(h_i does not need to be convex, so $\hat{f}_i = h_i \circ \bar{f}_i$ are **not necessarily convex**)

- Extends the approach of fitting parametric **convex** functions

(Schaller, Bemporad, Boyd, arXiv, 2025)

```
pip install lpcf
```

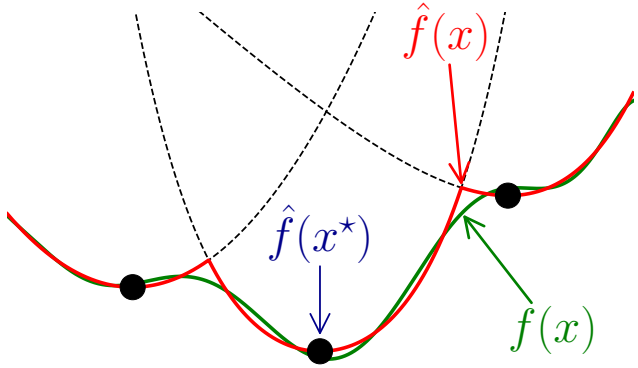
MINIMIZING THE SURROGATE

- **Step 1**: solve K **convex** problems (possibly **in parallel**)

$$x_i^\star(p) \in \arg\min_{x\in\mathcal{X}} \bar{f}_i(x,p) \quad \text{s.t. } g(x,p)\le 0 \quad = \quad \arg\min_{x\in\mathcal{X}} h_i(\bar{f}_i(x,p)) \quad \text{s.t. } g(x,p)\le 0$$

- **Step 2**: select the best among the K candidates via h_i

$$i^\star(p) = \arg\min_{i=1,\dots,K} h_i\big(\bar{f}_i(x_i^\star(p),p)\big)$$
$$x^\star = x^\star_{i^\star(p)}(p)$$



- Extension to **nonconvex** constraint functions g is also possible

LEARNABLE BUILDING BLOCKS

- **Monotonic** h_i: feedforward NN with **nonnegative weights**, monotonic activations (ReLU, softplus, sigmoid) + linear last layer

- **Convex** $\bar{f}_i(x,p)$: any parametric family, coefficients learned as NNs of p
  - $\alpha\|x\|_2^2 + \|L(p)x\|_2^2 + c(p)^\top x + d(p)$ **convex quadratic**
    $L(p)$ lower-triangular, $\alpha \geq 0$
  - $\max_{t=1,\dots,L} \left(A_t(p)x + b_t(p)\right)$ **convex piecewise affine**
  - $\sum_{t=1}^{L} \max(A_t(p)x - b_t(p), 0)^2$ **max-squared**
  - $NN(x;\varphi(p))$ parametric **input-convex NN** (Amos et al., 2017)
    (Schaller, Bemporad, Boyd, 2025)

SMOOTHED MINIMUM FOR TRAINING

- The exact $\min_i(\cdot)$ is **nonsmooth**: to match gradients of a differentiable f, replace it **during training only** by the **log-sum-exp smoothing**

$$\hat{f}(x,p) = -\gamma \log \sum_{i=1}^{K} \exp\left(-\tfrac{1}{\gamma} h_i(\bar{f}_i(x,p)) \right)$$

- Converges to the exact min-of-convex surrogate as $\gamma \to 0$
- γ trades off **approximation quality** vs **numerical conditioning**
- **At solve time** (after training): use (hard) minimum to evaluate $\hat{f}(x,p)$, so to solve K convex problems in parallel and select the best solution

LEARNING PROBLEM AND REGULARIZATION

- Dataset $\{x_k, p_k, f_k\}_{k=1}^N$, $f_k = f(x_k, p_k)$: only **function evaluations** needed, no solver required

- **Data-fitting loss**: let Θ = parameters of h_i and $\bar{f}_i$. We want to minimize the MSE

$$\frac{1}{N}\sum_{k=1}^{N}\left(f_k - \hat{f}_\Theta(x_k, p_k)\right)^2$$

- **Optionally**, if gradients $\nabla_x f_h$ are also available, use **gradient-matching** regularization:

$$\mathcal{R}_1(\Theta) = \frac{w_1}{M_1}\sum_{h=1}^{M_1}\left\|\nabla_x \hat{f}_\Theta(x_h, p_h) - \nabla_x f_h\right\|_2^2$$

LEARNING PROBLEM AND REGULARIZATION

- **Moreover**, if in addition a subset of **optimal solutions** $x_h^\star$ (with duals $\lambda_h^\star$) are also available, satisfying the KKT stationarity condition

$$\nabla_x f(x_h^\star, p_h^\star) + \nabla_x g(x_h^\star, p_h^\star)\lambda_h^\star = 0, \quad h = 1, \dots, M_2$$

  add them as **optimality regularization**

$$\mathcal{R}_2(\Theta) = \frac{w_2}{M_2} \sum_{h=1}^{M_2} \left\| \nabla_x \hat{f}_\Theta(x_h^\star, p_h^\star) + \nabla_x g(x_h^\star, p_h^\star)\lambda_h^\star \right\|_2^2$$

- **Note**: points x_k where the solver converged loosely, even if excluded from $\mathcal{R}_2$ because of excessive KKT violation, remain useful for MSE loss $\mathcal{L}$

- Finally, add standard regularization $\mathcal{R}_3(\Theta)$ (e.g., ℓ_2 or ℓ_1 penalties) to avoid overfitting the training data

DATA SAMPLING FOR TRAINING

- **Uniform** sampling of x_k in $\mathcal{X}$ under-represents its **boundary**, where f's behavior matters most for constrained optimization

- **Projected sampling**: (i) sample $\{x_k\}$ uniformly in an **enlarged** domain $\tilde{\mathcal{X}} \supseteq \mathcal{X}$; (ii) project onto $\mathcal{X}$

$$x_k^{\text{proj}} = \arg\min_{x \in \mathcal{X}} \|x - x_k\|_2^2$$

  enriching data **near the boundary**; if $\mathcal{X}$ = hyperbox $\Rightarrow$ projection = clipping

- Sampling covers **all of $\mathcal{X}$**, not only the p-dependent feasible set $\{x \in \mathcal{X} : g(x,p) \leq 0\} \Rightarrow$ surrogate $\hat{f}$ captures the **global** landscape of f in $\mathcal{X}$

- p_k: sampled uniformly in $\mathcal{P}$, or application driven (e.g., inherited from previously deployed-controller data when approximating NL-MPC)

TRAINING PIPELINE AND SOFTWARE

- **Training**: `Adam` (1000 epochs) + `L-BFGS` ($\leq$ 2000 epochs), implemented in `jax-sysid` (Bemporad, 2025) 

```
pip install jax-sysid
```

- Two alternatives of using the surrogate $\hat{f}$ for each given p:
  - **replace** the original problem completely, take $\hat{x}(p)$ as the solution
  - use $\hat{x}(p)$ from the parallel convex solve as a **warm start** for the original nonconvex solver (e.g., SQP) to get $x^\star(p)$

- Convex subproblems solved with `CVXPY` (Agrawal et al., 2018) + `Clarabel` (Goulart, Chen, 2024)

EXAMPLE: NONCONVEX FUNCTION APPROXIMATION

- **Six-hump camel-back** function on $x_1 \in [-2, 2], x_2 \in [-1, 1]$

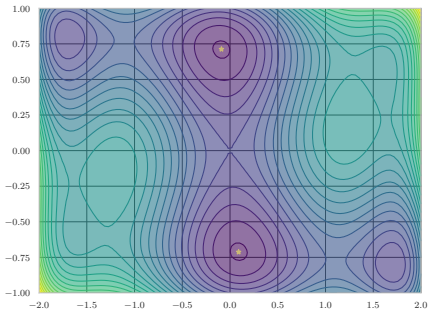
$N = 1000$ samples

$\hat{f}_i$ = **max-squared** fcn ($L = 10$)

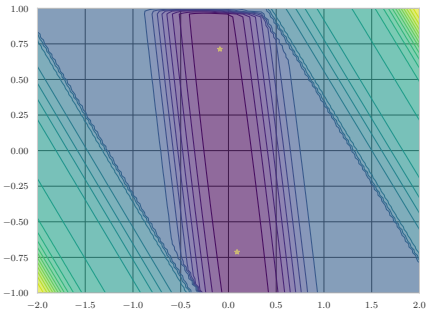
shared h_i = tiny `tanh` NN

$K = 2$: both global minima captured

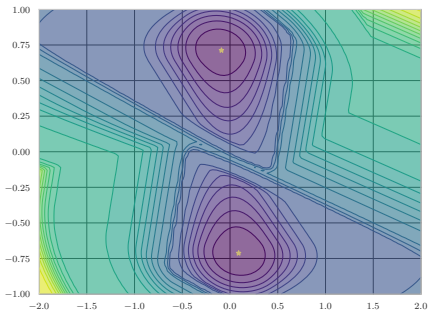
$K = 5$: $\hat{f}$ very close to f ($R^2 = 0.992$)
training time = 7.4 s [Apple M5 Max CPU]



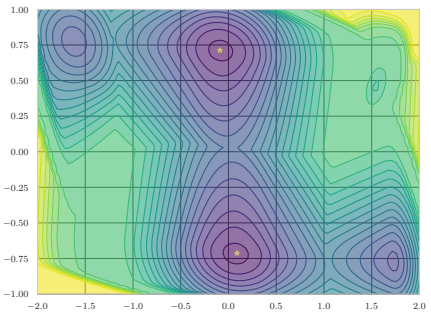
ground truth



$K = 1$



$K = 2$



$K = 5$

- **Note**: Minimizing the **surrogate** ($K = 5$) directly lands near the global min while only 64/100 random `L-BFGS` starts on the **original** function do

EXAMPLE: NONCONVEX PATH-TRACKING MPC

- **Unicycle robot** tracking a Lissajous path

MPC horizon $N_p = 10$, move blocking $N_b = 4$

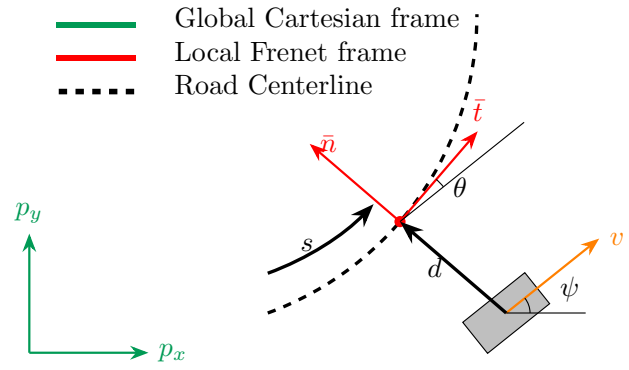
input bounds + **state bound** $A(x_0) + B(x_0)u_0 \in \mathcal{X}$

surrogate $\hat{f}$: $K = 2$ **ICNN** softplus (5,5) $\quad (\Theta \in \mathbb{R}^{28128})$

1000 training samples w/ gradient-matching regularization

CPU time: $\approx$ 3 hrs [Intel i9-13900 HX]

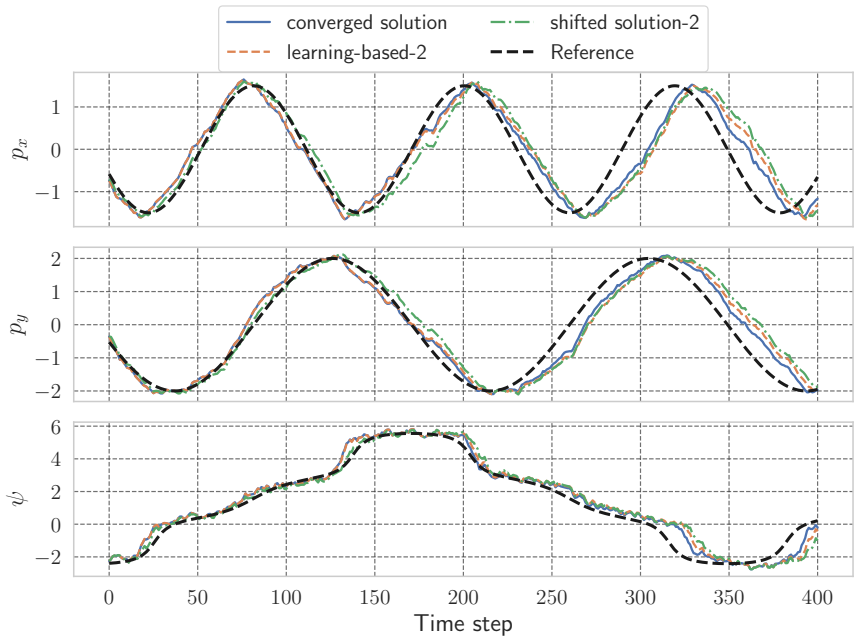
$\hat{f}$ used to **warm start** SQP (`CasADi`+`qpOASES`)



Frenet coordinates $x = (s, d, \theta)$

- Online CPU time:

	mean	max
cold-start - full SQP	4.140	15.565
prev. shifted - full SQP	**2.582**	8.686
prev. shifted - 2 QPs	**1.717**	4.426
$\hat{x}(p)$ - full SQP (incl. min $\hat{f}$)	2.739	5.472
$\hat{x}(p)$ - 2 QPs (incl. min $\hat{f}$)	2.211	3.089



CONCLUSIONS

CONCLUSIONS

- Different learning-based methods for **modeling** and **optimization**:
  - **Active-learning** for regression with **worst-case error bound** quantification
  - Learn approximate **solutions** of **multiparametric GNEPs** (and **mpNLPs**)
  - Learn **objective function** surrogates in **min-of-convex fcns** form to accelerate, and possibly globalize, online nonconvex optimization

- **Current research**:
  - Amortized optimization for NL-MPC (Pillitteri, Bemporad, CDC 2026)

Postdoc positions available!

(contact me if interested)

Funded by the European Union

European Research Council
Established by the European Commission

ERC Advanced Grant "COMPACT" (2024-2029)