LEARNING-BASED AND PARAMETRIC OPTIMIZATION METHODS FOR SOLVING COMPETITIVE MULTI-AGENT DECISION-MAKING AND GAME-THEORETIC MPC PROBLEMS

Alberto Bemporad

imt.lu/ab

Funded by the European Union

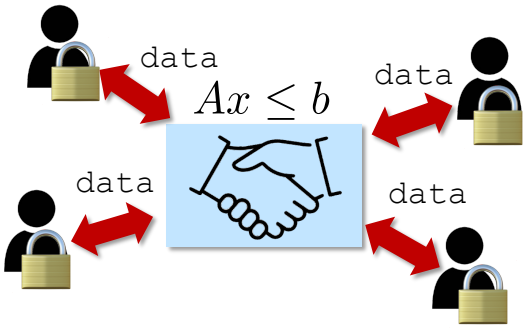
European Research Council
Established by the European Commission

IMT SCHOOL FOR ADVANCED STUDIES LUCCA

Workshop 13: Game Theory Meets MPC - Rio de Janeiro, Brazil, December 9, 2025

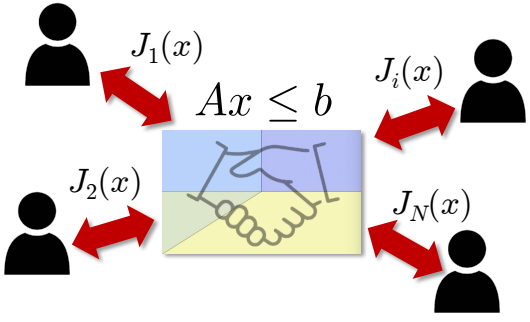
CONTENTS OF MY TALK

1. An **active learning** method to find generalized Nash equilibria from **best-response data**



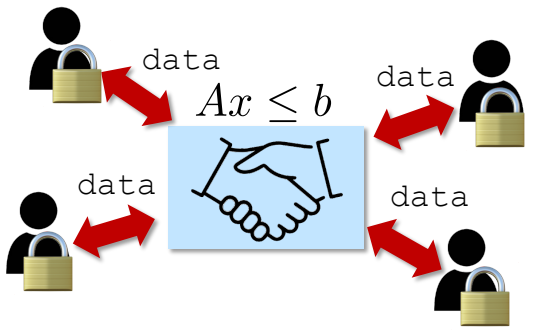
```
pip install gnep-learn
```

2. A **multiparametric solver** for **linear-quadratic** generalized Nash equilibrium problems



```
pip install nash-mpqp
```

LEARNING NASH EQUILIBRIA FROM DATA



Contents based on the following paper:

[1] F. Fabiani and A. Bemporad, "*An active learning method for solving competitive multi-agent decision-making and control problems*," IEEE Transactions on Automatic Control, vol. 70, no. 4, pp. 2374–2389, 2025

GNEP PROBLEM

- N (**non-cooperative**) agents, mutually interacting, each one solving

$$f_i(x_{-i}) = \arg\min_{x_i \in \mathcal{X}_i} \quad J_i(x_i, x_{-i}) \qquad x_i \in \mathbb{R}^{n_i}$$
$$\text{s.t.} \qquad (x_i, x_{-i}) \in \Omega \qquad x_{-i} \in \mathbb{R}^{n-n_i}$$

- **Objectives** J_i are **private**, **global constraint set** Ω is **shared**.
  Possible local constraints $x_i \in \mathcal{X}_i$ are either private or handled in Ω

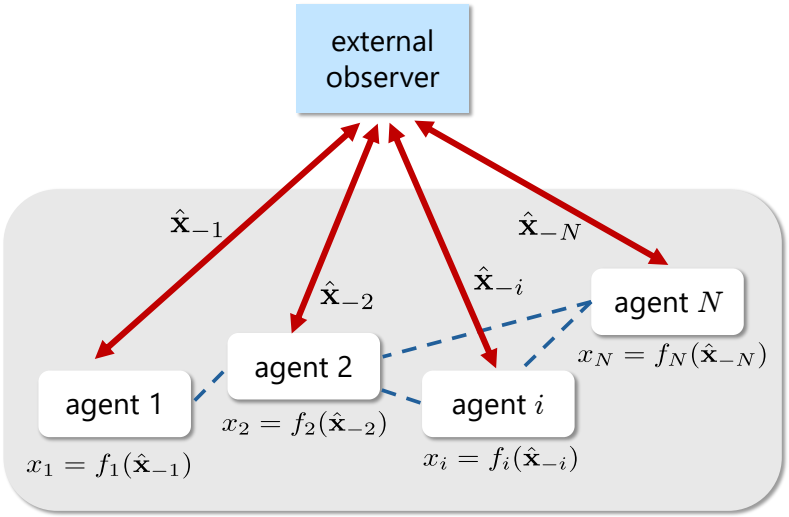
- **Best responses** $x_i = f_i(x_{-i})$ are single valued and can be **queried** at given x_{-i}

- **Goal**: learn a **stationary collective action profile** $x^* = (x_1^*, \dots, x_N^*) \in \Omega$:

$$x_i^* = f_i(x_{-i}^*) \qquad \forall i = 1, \dots, N$$

ACTIVE-LEARNING APPROACH

- Have an **external observer query** each agent i at selected $\hat{x}_{-i}$ and collect best responses $x_i = f_i(\hat{x}_{-i})$



- Based on such data, **update** an **affine surrogate model** $\hat{f}_i$ of best-response f_i

$$\hat{f}_i(x_{-i}, \theta_i) = \Gamma_i \begin{bmatrix} x_{-i} \\ 1 \end{bmatrix} \qquad \theta_i = \mathrm{vec}(\Gamma_i)$$

- **compute** an **approximate GNE**: **(constrained least-squares)**

$$\hat{x}^* = \underset{x \in \Omega}{\mathrm{argmin}} \sum_{i=1}^{N} \left\| x_i - \hat{f}_i(x_{-i}, \theta_i^{k+1}) \right\|_2^2$$

(minimum norm solution taken if multiple solutions exist)

- **Repeat** by querying each agent at $\hat{x}^*_{-i}$ until convergence (if any occurs)

ACTIVE-LEARNING APPROACH: KEY QUESTIONS

- How to **initialize** the affine models (θ_i)?
- How to **update** the affine models?
- If **convergence** occurs, do we get a GNEP solution to the **original** problem?

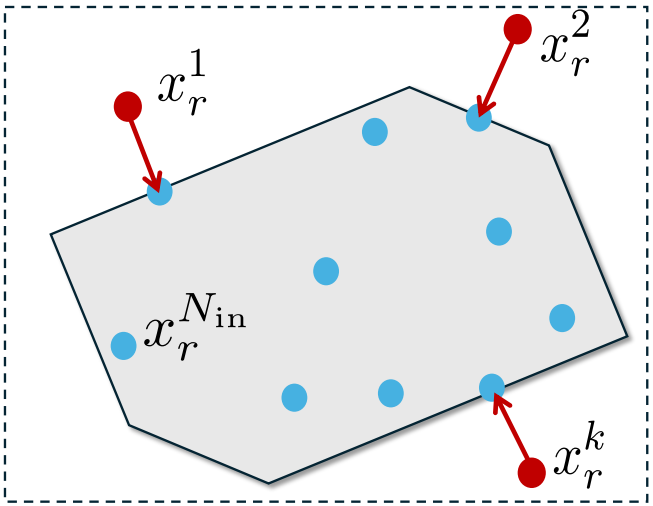
Note: we only assume that best-response functions f_i are **continuous**, $\Omega \neq \emptyset$

No assumption on existence and uniqueness of equilibria, convexity of objectives, differentiability, etc.

We also assume **noise-free** best-response queries.
For misleading responses, see (Franci, Fabiani, Bemporad, CDC 2025) → **ThC13.7**

INITIALIZATION

- Select initial points x_r^k **randomly**, $k = 1, \ldots, N_{\text{in}}$ in a hyper-box of interest (e.g., random or Latin-hypercube sampling)



- **Project** them onto Ω:

$$\hat{x}^k = \underset{x \in \Omega}{\text{argmin}} \; \|x - x_r^k\|_2^2$$

- For each point $\hat{x}^k$, get each agent's **best response** $x_i^k = f_i(\hat{x}^k_{-i})$

- Collect **initial dataset** $\{(\hat{x}^k_{-i}, x_i^k), k = 1, \ldots, N_{\text{in}}\}$

RECURSIVE BR-MODEL LEARNING

- Use bank of **linear Kalman filters** to **estimate** model parameters θ_i **recursively**:

$$\phi_i^k = \begin{bmatrix} \hat{x}_{-i}^k \\ 1 \end{bmatrix}, \; a_i^k = P_i^k \phi_i^k$$

$$P_i^{k+1/2} = P_i^k - \frac{a_i^k (a_i^k)'}{1 + (\phi_i^k)' a_i^k}$$

$$\theta_i^{k+1} = \theta_i^k + P_i^{k+1/2} \phi_i^k (\mathbf{x_i^k} - (\phi_i^k)' \theta_i^k)$$

$$P_i^{k+1} = P_i^{k+1/2} + \beta I$$

$$\theta_i = \mathrm{vec}(\Gamma_i)$$

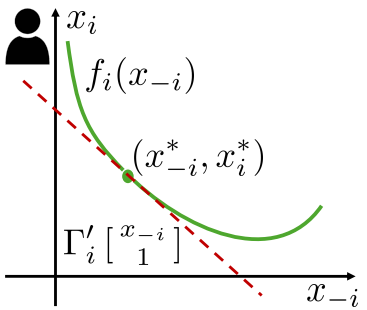
- β = **learning rate**. The larger β, the faster the estimator converges

- $P_0^i = \alpha I$. The smaller α, the larger the ℓ_2**-regularization** on θ^i

CONVERGENCE PROPERTIES

- **Main result**: Assume $\lim_{k\to\infty} \theta_i^k = \tilde{\theta}_i$ for all $i = 1, \dots, N$. Then, $\lim_{k\to\infty} \hat{x}^k = x^*$ is an equilibrium of the **original** GNEP, i.e., $x_i^* = f_i(x_{-i}^*)$

- Each affine surrogate $\hat{f}_i$ coincides with f_i **pointwise** at x_{-i}^*



☹ **Weakness**: we cannot guarantee if $\lim_{k\to\infty} \theta_i^k$ exist for all i

☺ Strength: very **minimal assumptions** about the problem definition!
(best responses f_i continuous, Ω nonempty)

- **Generalization**: $x_i = g_i(w_i)$. Each w_i is a **private** local vector subject to local constraints, and only x_i is shared

SIMPLE NUMERICAL EXAMPLE

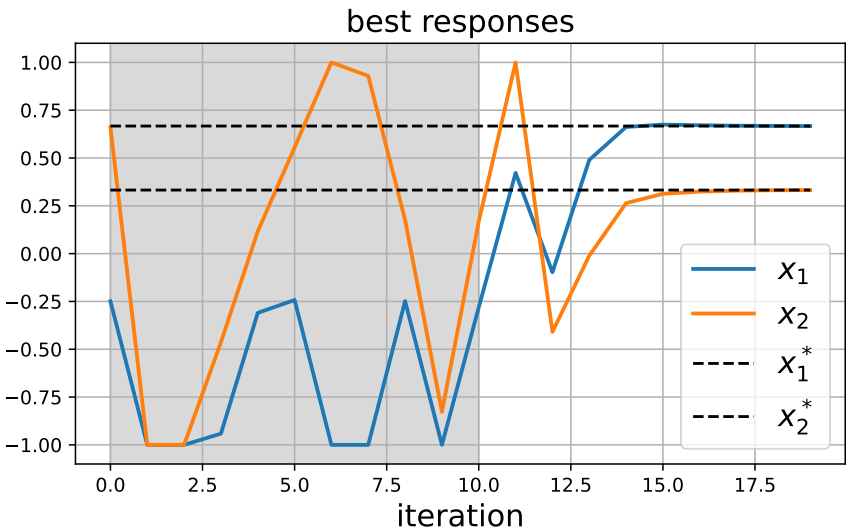
- Nash equilibrium problem:

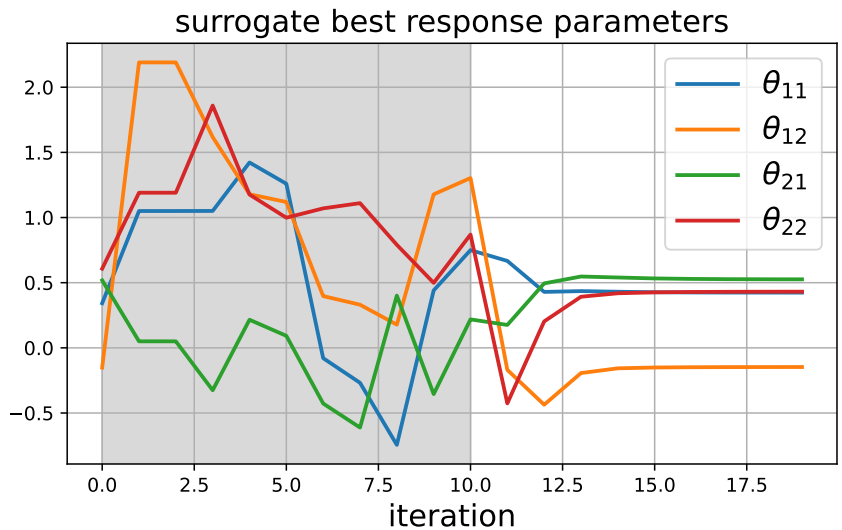
$$f_1(x_2) = \underset{-1\le x_1\le 1}{\operatorname{argmin}}\ x_1^4 - x_1x_2 - x_1$$

$$f_2(x_1) = \underset{-1\le x_2\le 1}{\operatorname{argmin}}\ x_2^4 + x_1x_2 - x_2$$

- Generate $N_{\text{in}} = 10$ initial random samples in $[-1,1]^2$ and run active-learning algorithm for another 10 iterations

- Result: $x_1^* \approx 0.7077, x_2^* = 0.4181$





PYTHON PACKAGE

gnep-learn

```
from gnep_learn import GNEP

n = 2  # number of agents
dim = [1, 1]  # each agent optimizes one variable
N_init = 10 # number of initial random sampling iterations
N = 20 # total number of iterations

def J1(x1,x2):
    return x1**4 -x1*x2 - x1 # function minimized by agent #1
def J2(x2,x1):
    return x2**4 +x1*x2 - x2 # function minimized by agent #2

J = [lambda x1, x2: J1(x1,x2)[0],
     lambda x2, x1: J2(x2,x1)[0]]  # agents' objectives

lbx = -1. * np.ones(2) # lower bound on x
ubx = 1. * np.ones(2) # upper bound on x

gnep = GNEP(J, dim, lbx=lbx, ubx=ubx) # create GNEP object

xeq, XX, XXeq, _ = gnep.learn(N=N, N_init=N_init) # learn equilibrium
```

```
pip install gnep-learn
```

github.com/bemporad/gnep-learn

NUMERICAL EXAMPLE

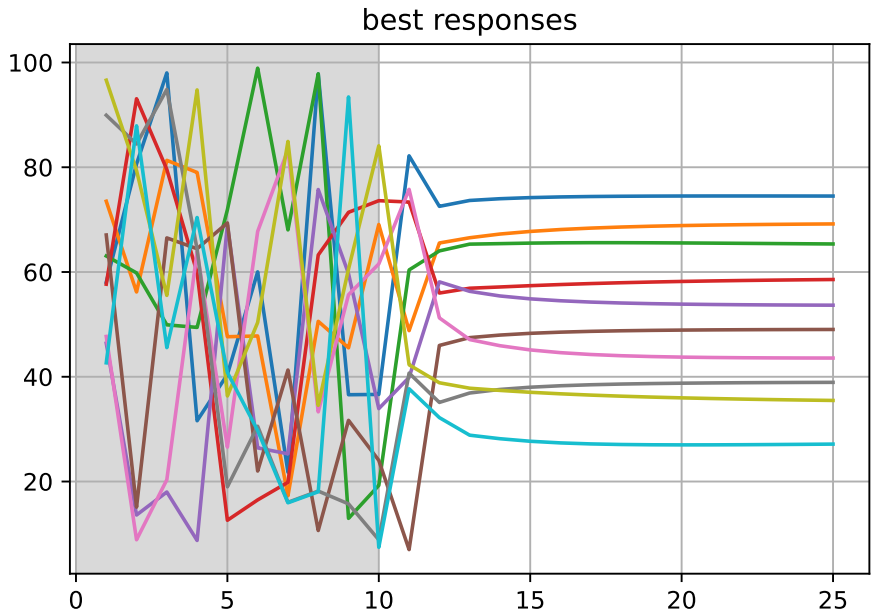
(Salehisadaghiani, Shi, Pavel, 2017 - Example 1)

- Nash equilibrium problem with $N = 10$ agents:

$$J_i(x_i, x_{-i}) = N\left(1 + \frac{i-1}{2}\right)x_i - x_i(60N - \mathbf{1}'x) \qquad \Omega = [7, 100]^N$$

- Generate $N_{\text{in}} = 10$ initial random samples in Ω and run active-learning algorithm for another 15 iterations

- Learning rate $\beta = 1$



NUMERICAL EXAMPLE

(Facchinei, Kanzow, 2009 - Example 1)

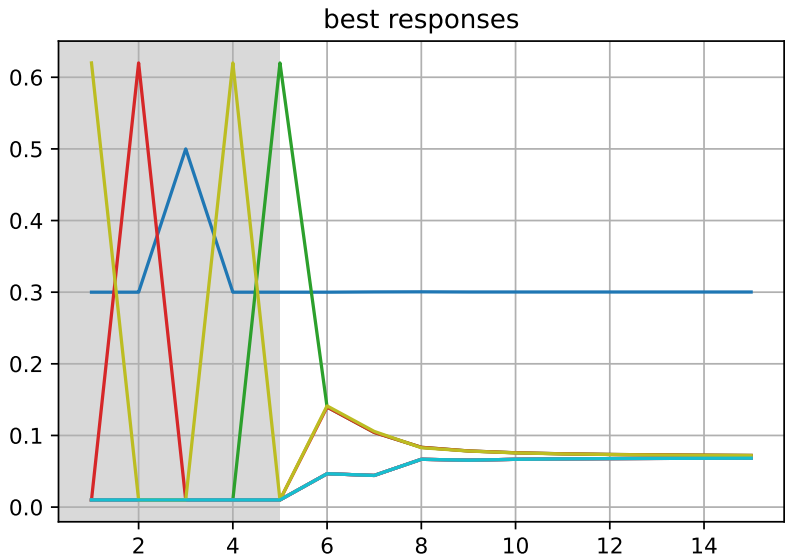
- GNEP, $N = 10$ agents (internet switching behavior induced by selfish/ users)

$$J_i(x_i, x_{-i}) = -\frac{x_1}{\mathbf{1}'x}(1 - \mathbf{1}'x) \qquad x_1 \in [0.3, 0.5],\ x_j \in [0.01, 100], \forall j > 1$$

- Coupling constraint: $\mathbf{1}'x \leq 1$

- $N_{\text{in}} = 5$ initial random samples, active-learning algorithm run for 10 iterations

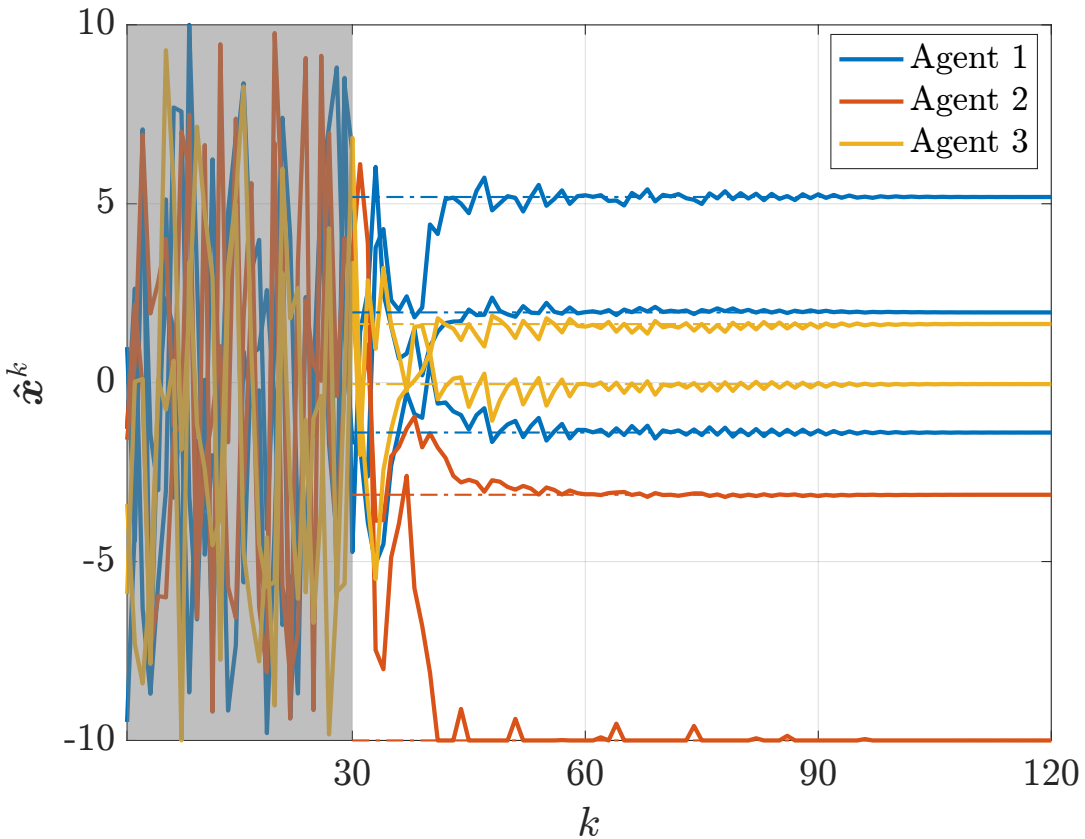
- Learning rate $\beta = 1$



NUMERICAL EXAMPLE

(Facchinei, Kanzow, 2009 - Example 3)

- GNEP with $N = 3$ agents of dimensions $3, 2, 2$
- Coupling constraint: $\mathbf{1}'x \leq 1$
- $N_{\text{in}} = 30$ initial random samples, active-learning algorithm run for 90 iterations
- Learning rate $\beta = 1$



NUMERICAL EXAMPLE - NO NASH EQUILIBRIUM EXISTS

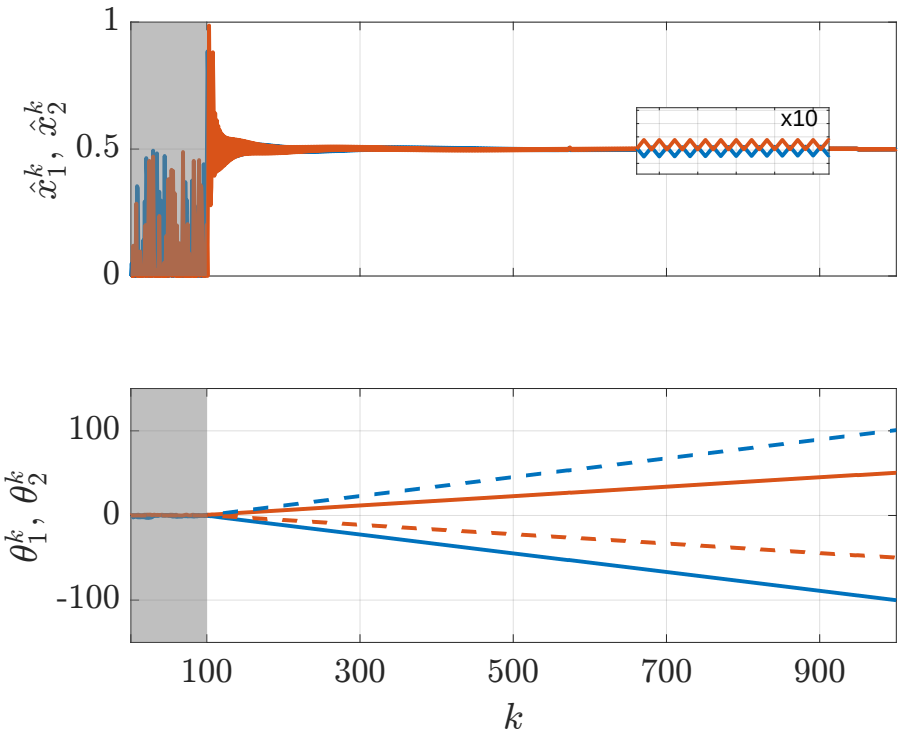
(Fabiani, Bemporad, 2025)

- Nash equilibrium problem with $N = 2$ agents, no equilibrium exists

$$f_1(x_2) = \underset{0 \le x_1 \le 1}{\operatorname{argmin}} \; -x_1^2 + 2x_1x_2$$

$$f_2(x_1) = \underset{0 \le x_2 \le 1}{\operatorname{argmin}} \; x_2 - 2x_1x_2$$

- Learning rate $\beta = 1$
- Best response **oscillates** around $\frac{1}{2}$
- Parameter vector θ **diverges**



NUMERICAL EXAMPLE: COMPETING LQR CONTROLLERS

- **Square linear system** controlled by N agents

$$\begin{cases} z(k+1) = Az(k) + \sum_{i=1}^{N} B_i u_i(k) \\ y_i(k) = C_i z(k) \end{cases}$$

A **unstable**

- Each agent i controls input u_i with a **policy** $u_i = K_i z$ to influence output y_i, given the other agents' policies K_{-i}

$$z(k+1) = A_i z(k) + B_i u_i(k), \qquad A_i \triangleq A + \sum_{j \neq i} B_j K_j$$

- Each agent i minimizes the infinite-horizon **LQR cost**

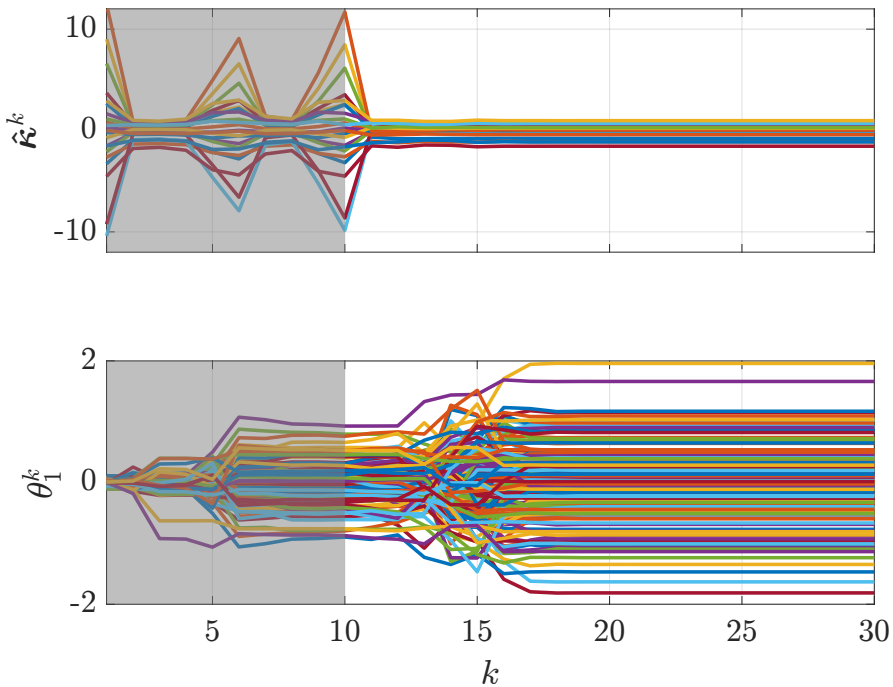
$$J_i(K_i, K_{-i}) = \sum_{k=0}^{\infty} y_i(k)' Q_i y_i(k) + u_i'(k) R_i u_i(k)$$

i.e., $K_i = f_i(K_{-i})$ = LQR gain for (A_i, B_i, Q_i, R_i) ($\Rightarrow$ coupled Alg. Riccati eqns)

NUMERICAL EXAMPLE: COMPETING LQR CONTROLLERS (CONT'D)

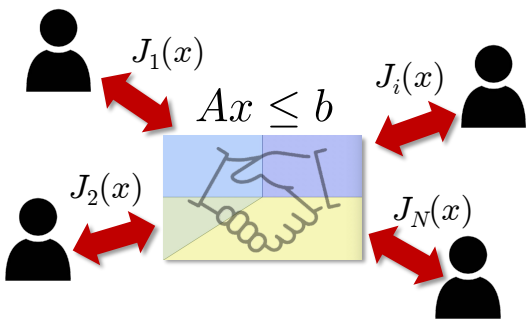
- **Init**: centralized LQR gains with $Q = I, R = I$, randomly perturbed system
- $N = 3$ agents. Agent #i weights n_{y_i} outputs out of n_y
- Highly nonlinear problem due to Riccati equations. Run algorithm 1000 times

Ex.	n_z	n_y	$n_{y_1}, n_{y_2}, n_{y_3}$	convergence
A	8	6	$\{5, 2, 5\}$	$600/1000$
B	7	4	$\{3, 2, 4\}$	$493/1000$
C	4	6	$\{2, 4, 2\}$	$876/1000$
D	5	6	$\{4, 5, 5\}$	$825/1000$
E	6	5	$\{2, 5, 3\}$	$981/1000$
F	4	6	$\{2, 3, 3\}$	$895/1000$
G	8	6	$\{2, 5, 4\}$	$949/1000$
H	4	3	$\{3, 3, 2\}$	$986/1000$
I	5	6	$\{2, 5, 2\}$	$629/1000$
J	8	5	$\{4, 5, 2\}$	$947/1000$



Note: if convergence occurs, K_i are always the same!

MULTIPARAMETRIC LINEAR QUADRATIC GNEP



Contents based on the following paper:

[1] S. Hall and A. Bemporad, "*Solving Multiparametric Generalized Nash Equilibrium Problems and Explicit Game-Theoretic Model Predictive Control*, arXiv 2512.05505, 2025

MULTIPARAMETRIC LINEAR QUADRATIC GAMES

- N agents with best responses

$$\begin{cases} x_i^*(x_{-i}, p) = \arg\min\limits_{x_i} \dfrac{1}{2} x' Q_i x + (c_i + F_i p)' x \\ \text{s.t. } Ax \leq b + Sp \end{cases}$$

parameter $p \in \mathbb{R}^{n_p}$

- Best response = solution to a **multiparametric Quadratic Programming** (mpQP) problem with parameters (x_{-i}, p)
- Assume each QP #i is **strictly convex** wrt $x_i \Rightarrow$ best response is **unique**
- Possible **local constraints** on x_i embedded in global constraints $Ax \leq b + Sp$

MULTIPARAMETRIC LINEAR QUADRATIC GAMES

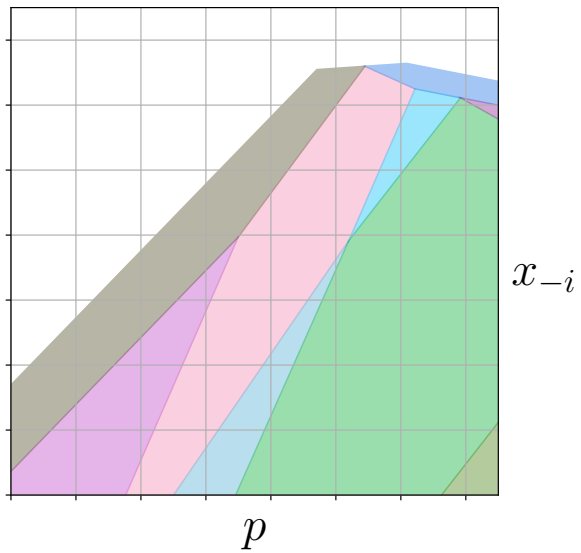
- The best response $x_i^*(x_{-i}, p)$ is a **continuous piecewise affine** function of (x_{-i}, p) over a **polyhedral partition** of the (x_{-i}, p)-space

(Bemporad, Morari, Dua, Pistikopoulos, 2002)

$$x_i^*(x_{-i}, p) = \begin{cases} H_{-i}^1 x_{-i} + G_i^1 p + h_i^1 & \text{if } (x_{-i}, p) \in CR_i^1 \\ \vdots & \vdots \\ H_{-i}^{N_i} x_{-i} + G_i^{N_i} p + h_i^{N_i} & \text{if } (x_{-i}, p) \in CR_i^{N_i} \end{cases}$$

$$CR_i^j = \left\{ (x_{-i}, p) : C_{-i}^j x_{-i} + D_i^j p \le e_i^j \right\}$$

$$j = 1, \ldots, N_i$$



- Each critical region CR_i^j corresponds to an optimal active set of QP #i
- We have N partitions, one per agent i, with N_i critical regions (CRs)

EQUILIBRIUM CONDITIONS

- Consider a **combination** $\mathcal{C}_k = (j_1, \ldots, j_N)$ of agents' CRs, $k = 1, \ldots, \prod_{i=1}^{N} N_i$

- A combination $\mathcal{C}_k$ is **valid** if, for each constraint involving agents i and j, either both have it active or both inactive

- **Equilibrium condition** for $\mathcal{C}_k = (j_1, \ldots, j_N)$:

$$\begin{cases} x_1 = H_{-1}^{j_1} x_{-1} + G_1^{j_1} p + h_1^{j_1} \\ \vdots \quad\quad \vdots \\ x_N = H_{-N}^{j_N} x_{-N} + G_N^{j_N} p + h_N^{j_N} \end{cases} \quad \text{and} \quad \begin{cases} C_{-1}^{j_1} x_{-1} + D_1^{j_1} p \le e_1^{j_1} \\ \vdots \\ C_{-N}^{j_N} x_{-N} + D_N^{j_N} p \le e_N^{j_N} \end{cases}$$

- A **critical region** for the **mp-GNEP** associated with $\mathcal{C}_k$ is the projection onto the p-space of the polyhedron defined by the above constraints

CRITICAL REGIONS WITH UNIQUE EQUILIBRIUM

- For each valid combination $\mathcal{C}_k$, rewrite the equilibrium condition as

$$M_x^k x = M_p^k p + M_1^k \qquad M_x^k \in \mathbb{R}^{n\times n} \qquad \text{(parametric linear system)}$$

- If M_x^k is invertible, we get a **unique equilibrium** for all p:

$$x^*(p) = (M_x^k)^{-1} M_p^k p + (M_x^k)^{-1} M_1^k$$

with associated **polyhedral** critical region

$$CR_k = \left\{ p : C_{-i}^{j_i} x^*(p) + D_i^{j_i} p \le e_i^{j_i}, \quad i = 1, \dots, N \right\}$$

- Only nonempty full-dimensional CR's are stored (based on Chebyshev radius)

- **Note**: CR_k's may **overlap** ⇒ **multiple equilibria** exist for the same p

NO EQUILIBRIUM AND MULTIPLE EQUILIBRIA

- If M_x^k is singular, compute **SVD decomposition** $M_x^k = U\Sigma V'$

- Change variables $y_1 = V_1'x, y_2 = V_2'x$ to rewrite the equilibrium condition as

$$\begin{aligned} y_1 &= \Sigma_1^{-1}(U_1'M_p^k p + U_1'M_1^k) \\ 0 &= U_2'M_p^k p + U_2'M_1^k \end{aligned} \qquad U = [U_1\, U_2], \quad V = [V_1\, V_2]$$

- Potential (**infinitely-many**) Nash equilibria:

$$x^*(p) = V_1\Sigma^{-1}(U_1'M_p^k p + U_1'M_1^k) + V_2 y_2 \qquad y_2 = \text{free vector}$$

- Conditions for existence of **full-dimensional** critical regions and equilibria:

$$U_2'M_p^k = 0,\ U_2'M_1^k = 0$$

and the **projection** $CR_k = \left\{p : \exists y_2 : C_{-i}^{j_i} x^*(p) + D_i^{j_i} p \le e_i^{j_i},\ i = 1, \dots, N\right\}$ is nonempty and full-dimensional

CHOOSING AMONG MULTIPLE EQUILIBRIA

- In CR_k's with ∞-many equilibria, we can select the **minimum-norm solution**:

$$\|x^*(p, y_2)\|_2^2 = \|\Sigma^{-1}U_1'(M_p^k p + U_1' M_1^k)\|_2^2 + \|y_2\|_2^2$$

- We can split CR_k into **sub-regions** by solving the following mpQP for $p \in CR_k$:

$$\begin{aligned} y_2^*(p) = \arg\min\ & \|y_2\|_2^2 \\ \text{s.t. } & C_{-i}^{j_i} x^*(p, y_2) + D_i^{j_i} p \le e_i^{j_i}, \quad i = 1, \dots, N \end{aligned}$$

- **Alternative**: given a **social welfare cost** $f(x)$, solve instead the mpQP

$$\begin{aligned} y_2^*(p) = \arg\min\ & f(x^*(p, y_2)) \\ \text{s.t. } & C_{-i}^{j_i} x^*(p, y_2) + D_i^{j_i} p \le e_i^{j_i} \\ & i = 1, \dots, N \end{aligned}$$

e.g.: $f(x)$ **utilitarian sum** of agents' objectives

$$f(x) = \sum_{i=1}^{N} \frac{1}{2} x' Q_i x + (c_i + F_i p)' x$$

- In both cases, set the unique equilibrium $x^*(p) = x^*(p, y_2^*(p))$ (PWA function)

VARIATIONAL GNES

- A **variational GNE** is an equilibrium where all the agents involved in a **shared constraint** have equal **dual variables** (=cost sensitivity for that constraint)

- From the **KKT conditions** of each agent's mpQP, we retrieve the **dual variables** $\lambda_i^*(x_{-i}, p) = \Lambda_x x_{-i} + \Lambda_p p + \lambda_1$ associated with coupling constraints

- To compute a variational GNE:
  - for each considered constraint c involving at least two agents, **add**

$$\Lambda_x^j x_{-j} + \Lambda_p^j p + \lambda_1^j = \Lambda_x^i x_{-i} + \Lambda_p^i p + \lambda_1^i, \quad \forall j \neq i$$

    to the equilibrium conditions $M_x^k x = M_p^k p + M_1^k$

  - recompute SVD and check if vGNE solutions $x^*(p)$ exist

  - compute (sub)region $\overline{CR}_k \subseteq CR_k$ as before

VARIATIONAL GNES

- A way to partition $CR_k \setminus \overline{CR}_k$ is to solve the following mpQP for $p \in CR_k$:

$$\min_{y_2} \sum_{c} \sum_{i \in \mathcal{I}_c, i \neq i_1} \|\lambda^*_{c,i}(x^*_{-i}(p, y_2), p) - \lambda^*_{c,i_1}(x^*_{-i_1}(p, y_2), p)\|_2^2$$
$$\text{s.t. } C^{j_i}_{-i} x^*_{-i}(p, y_2) + D^{j_i}_i p \leq e^{j_i}_i,\ i = 1, \ldots, N$$

- $\overline{CR}_k \equiv$ critical region of the mpQP above corresponding to **unconstrained solution**, where the optimal cost is zero ($\lambda^*_{c,i} = \lambda^*_{c,i_1}, \forall i \neq i_1,\ \forall c \in I_c$)

- **Note**: other vGNE solutions may exist among the infinitely many, corresponding to different combinations of active local constraints

PYTHON PACKAGE

```
from nash_mpqp import NashMPQP

dim = [2,3,2]  # number of variables for each agent
split = "variational" # or None, "min-norm", "welfare", "variational-split"

nash_mpqp = NashMPQP(dim, pmin, pmax, xmin, xmax,
                     Q, c, F, A, b, S, lb, ub, split = split)

nash_mpqp.solve()
```

`lb`, `ub`: bounds on variables (finite or $\pm\infty$)

`pmin`, `pmax`: range of parameters

`xmin`, `xmax`: range of variables

`pip install nash-mpqp`

`github.com/bemporad/nash_mpqp`

EXAMPLE: SIMPLE 2-AGENT GNEP

- 2 agents, each controlling one variable x_1, x_2 (Rosen, 1965)

$$\min_{x_1 \geq 0} \frac{1}{2}(x_1)^2 - x_1 x_2 + p_2 x_1$$
$$\text{s.t. } -x_1 - x_2 \leq p_1$$

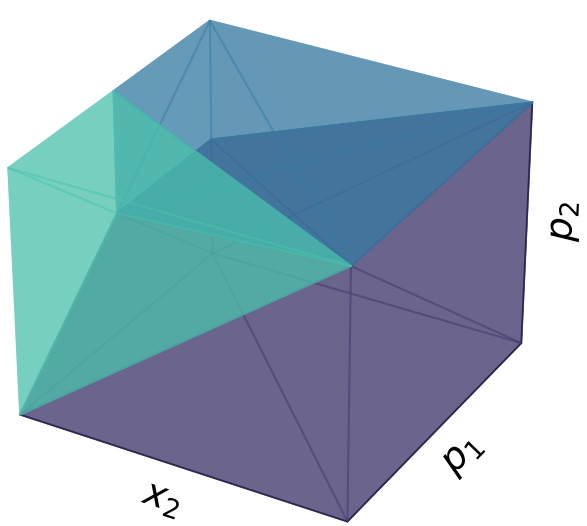
$$Q_1 = \begin{bmatrix} 1 & -1 \\ -1 & 0 \end{bmatrix}, \; F_1 = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$$

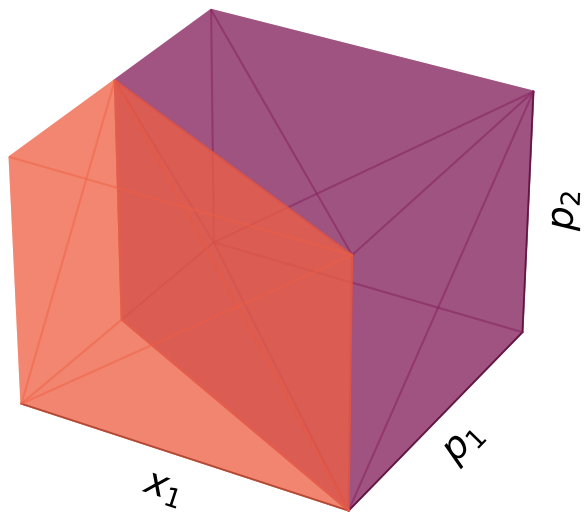
$$\min_{x_2 \geq 0} (x_2)^2 + x_1 x_2$$
$$\text{s.t. } -x_1 - x_2 \leq p_1$$

$$Q_2 = \begin{bmatrix} 0 & 1 \\ 1 & 2 \end{bmatrix}$$

- Parameter vector: $p = [p_1 \; p_2]'$

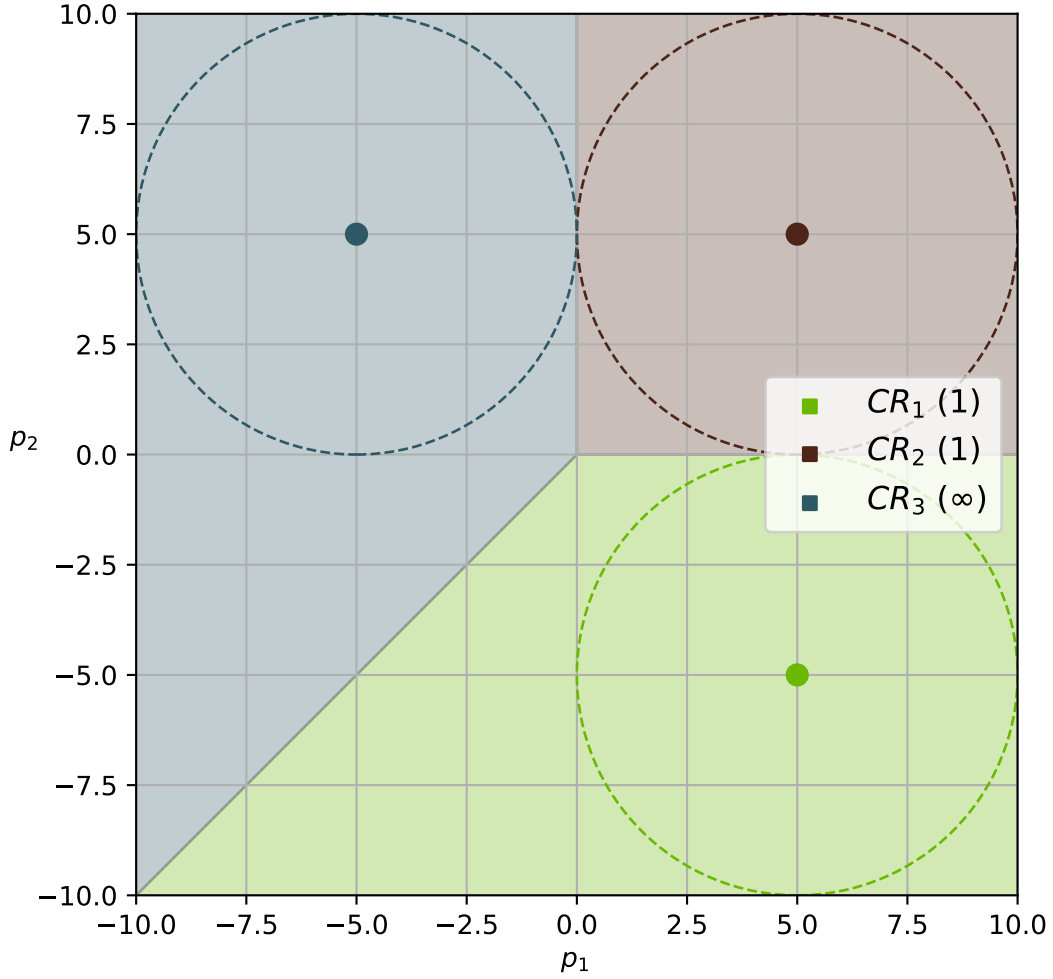
- CRs of each agent's best-response mpQP:





EXAMPLE: SIMPLE 2-AGENT GNEP

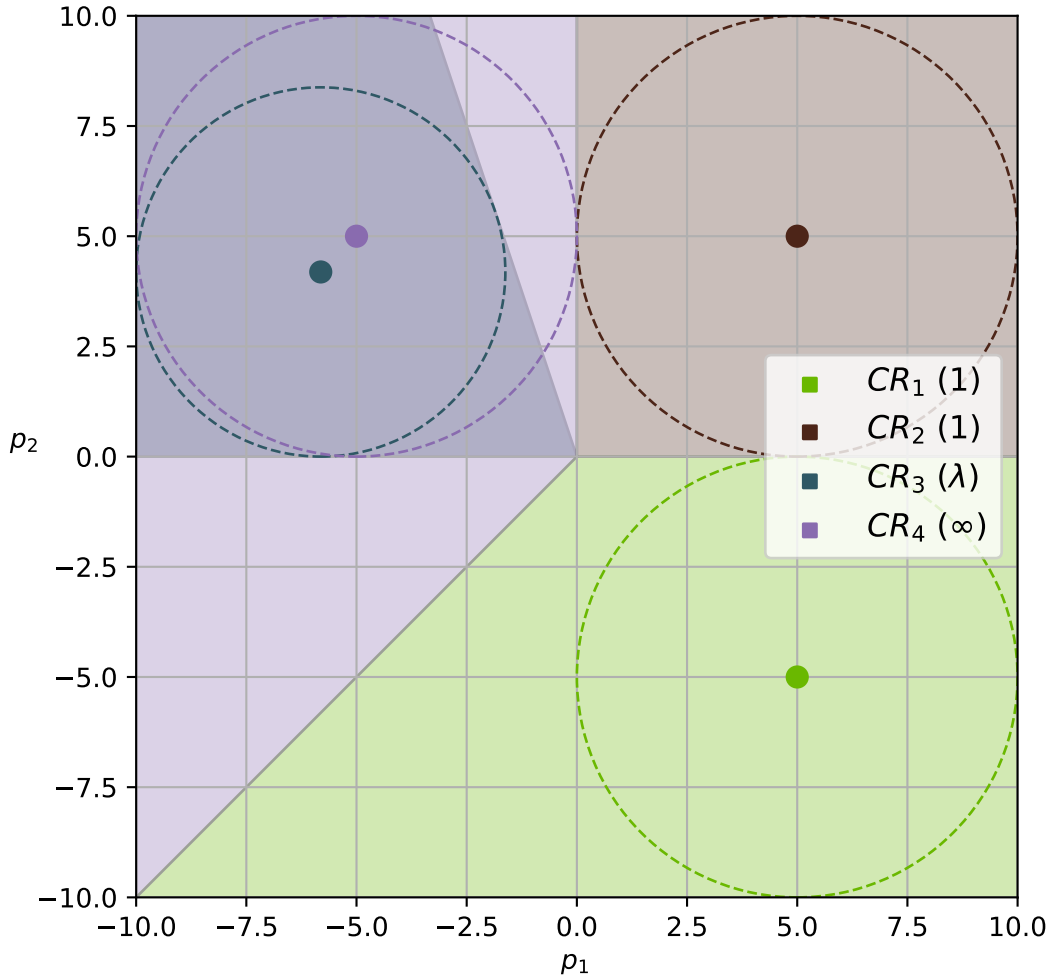
- **Critical regions:** no splitting in the case of infinitely-many equilibria



(1=single solution, ∞ = infinitely-many solutions)

EXAMPLE: SIMPLE 2-AGENT GNEP

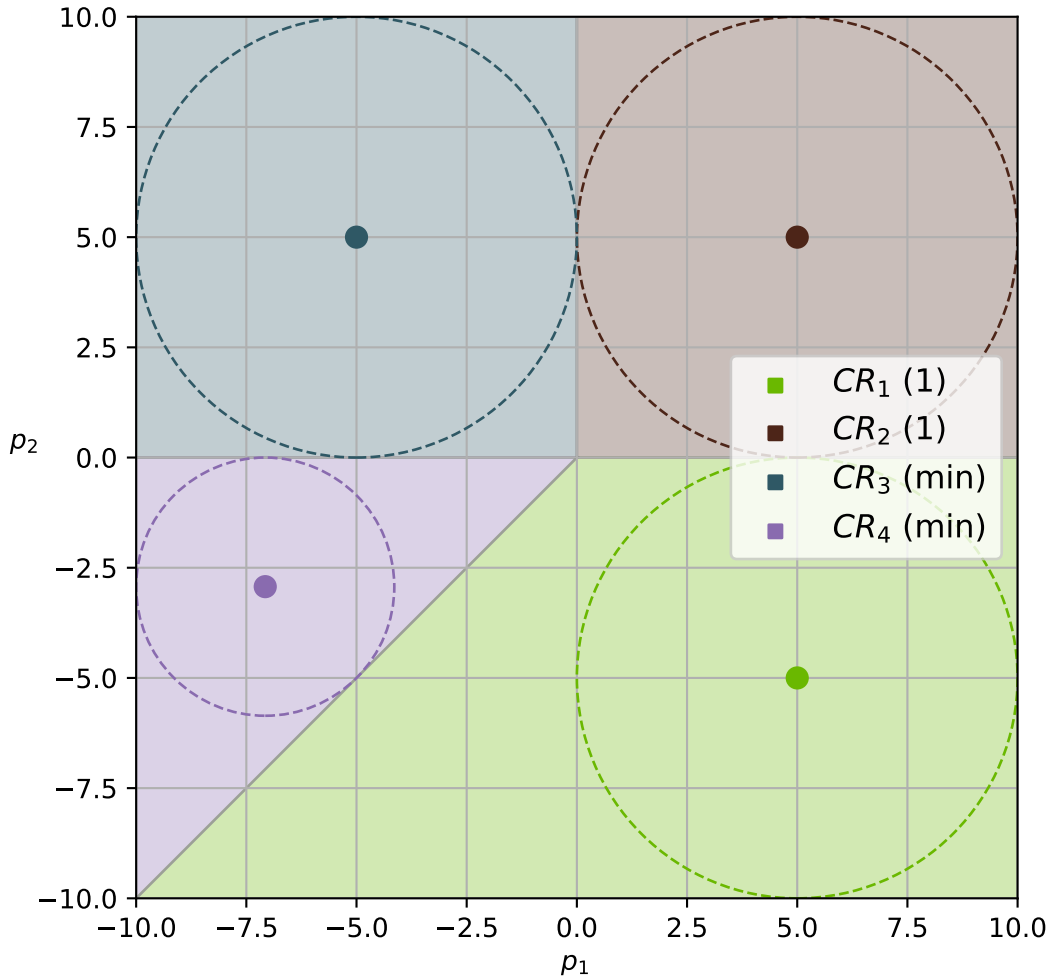
- **Critical regions**: adding **variational-GNE** regions



(1=single solution, ∞ = infinitely-many solutions, λ = variational GNE)

EXAMPLE: SIMPLE 2-AGENT GNEP

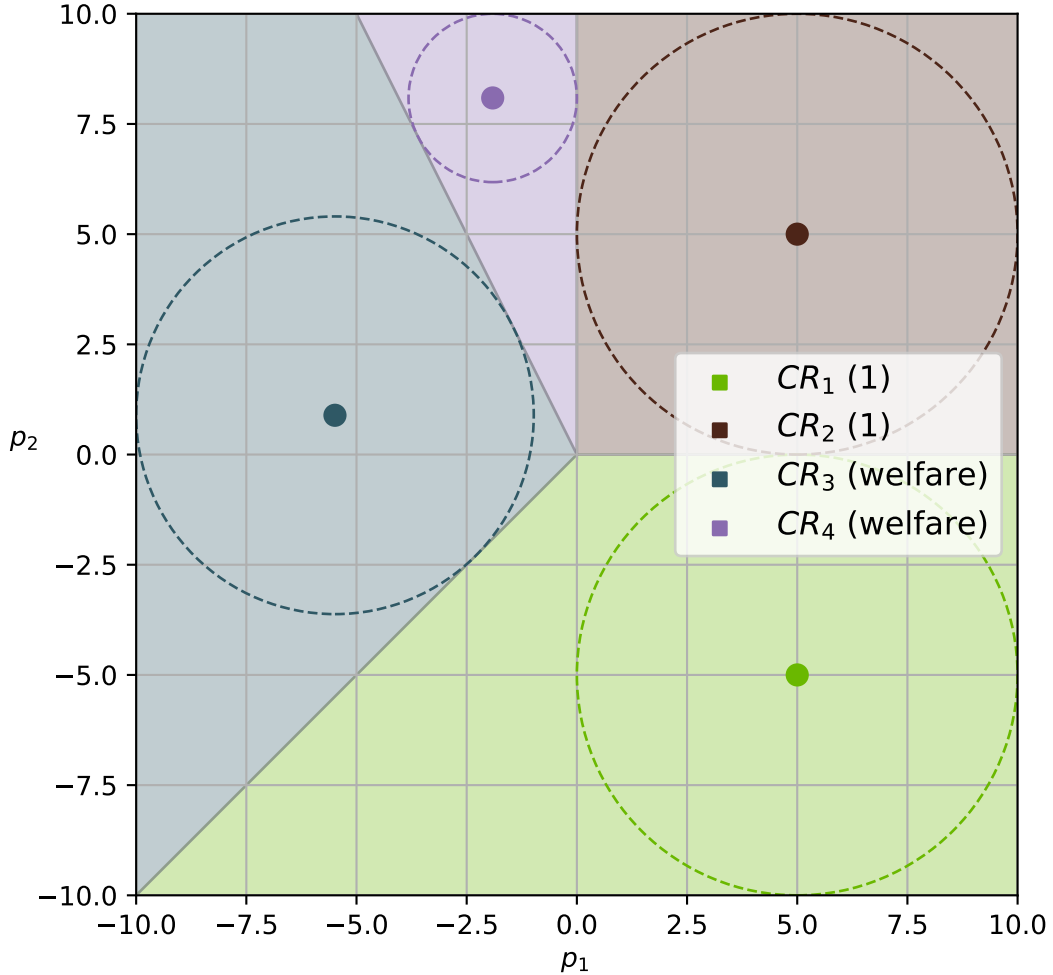
- **Critical regions**: use **min-norm GNE** solution criterion to split



(1=single solution, ∞ = infinitely-many solutions, min = min-norm GNE solution)

EXAMPLE: SIMPLE 2-AGENT GNEP

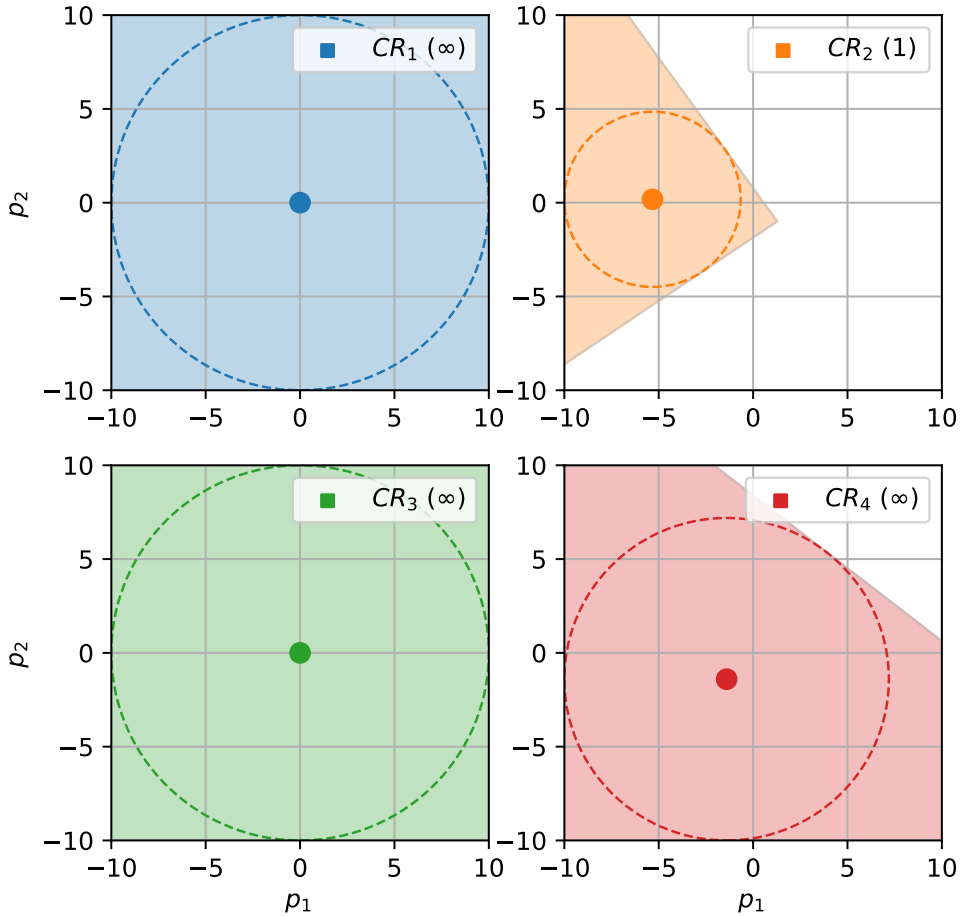
- **Critical regions**: use **welfare GNE** solution criterion to split



(1=single solution, ∞ = infinitely-many solutions, welfare = welfare GNE solution)

EXAMPLE: SIMPLE 3-AGENT GNEP

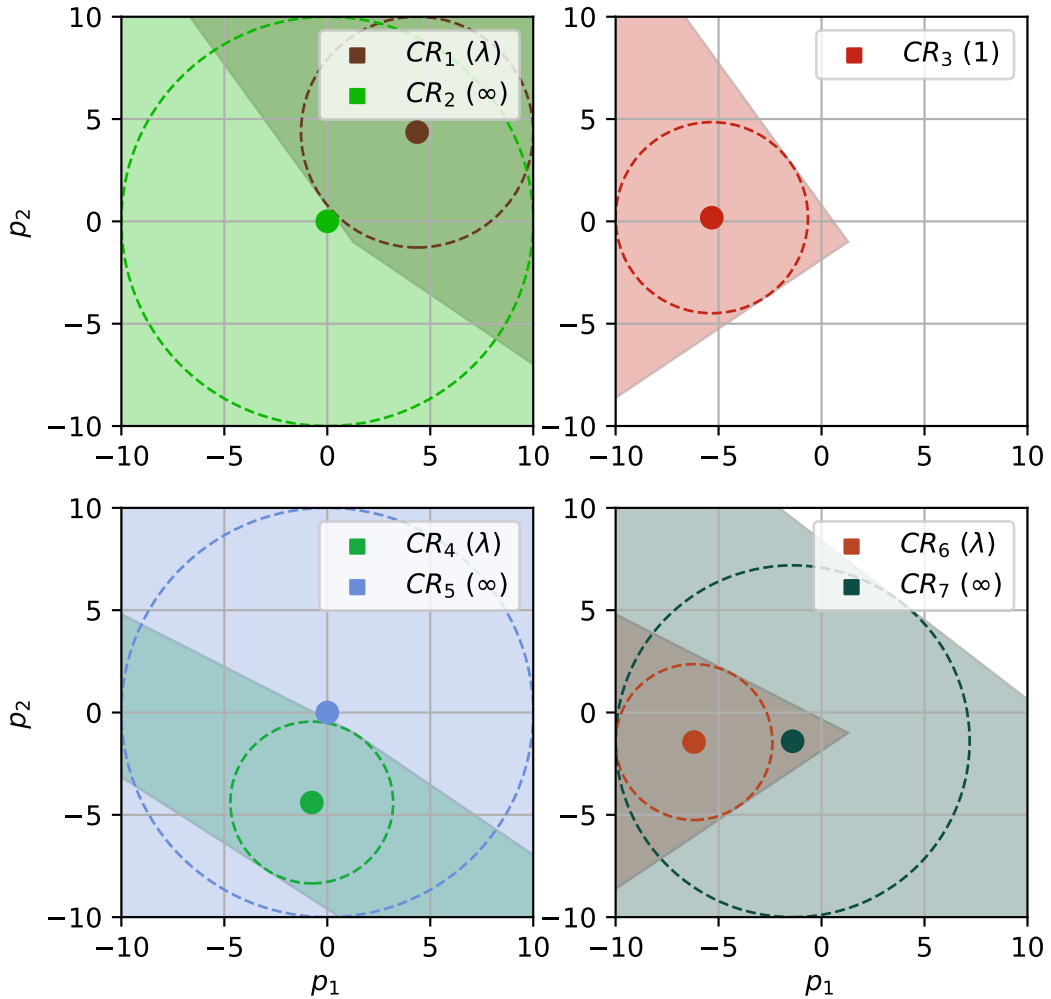
- **3 agents**, controlling variables $x_1 \in \mathbb{R}^2, x_2 \in \mathbb{R}^3, x_3 \in \mathbb{R}^2$
- **2 parameters** $p \in \mathbb{R}^2$
- randomly generated costs with random $Q_i \succ 0, c_i, F_i$
- **2 constraints**: random inequalities involving all agents, no local constraints



Note: critical regions are overlapping!

EXAMPLE: SIMPLE 3-AGENT GNEP

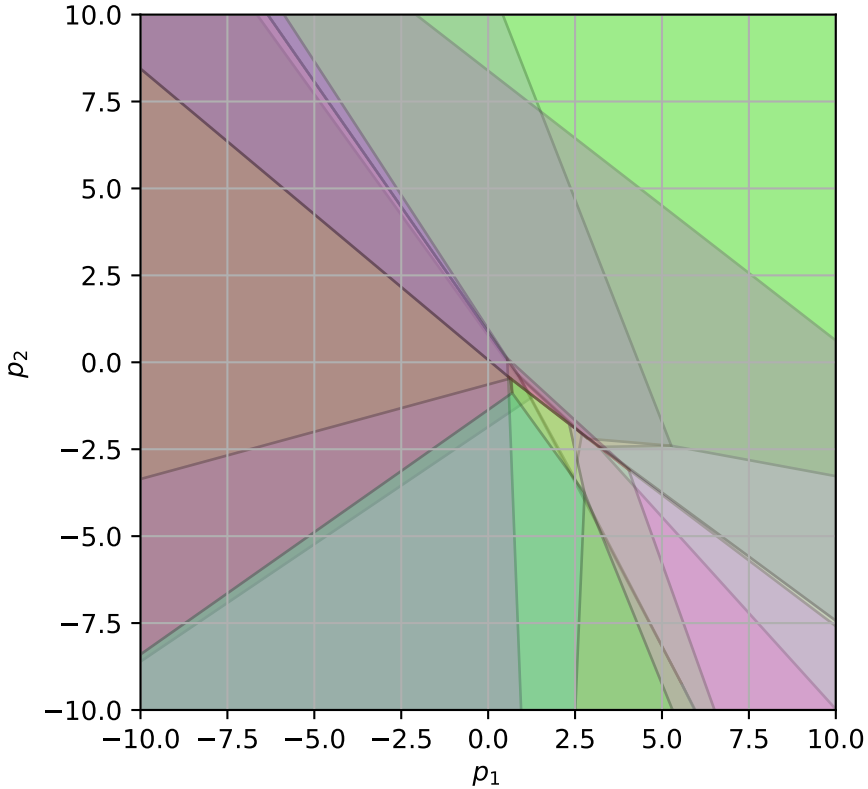
- **Critical regions**: adding **variational-GNE** regions



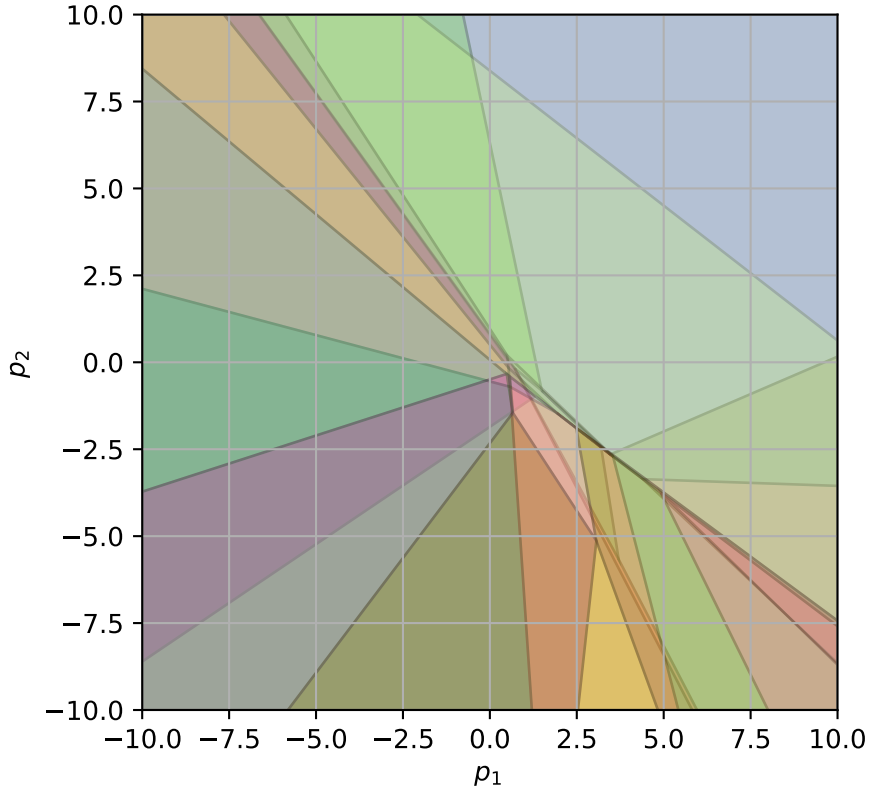
(1=single solution, ∞ = infinitely-many solutions, λ = variational GNE)

EXAMPLE: SIMPLE 3-AGENT GNEP

- **Critical regions**: use either **min-norm** or **welfare** GNE criterion to split CRs with infinitely-many equilibria



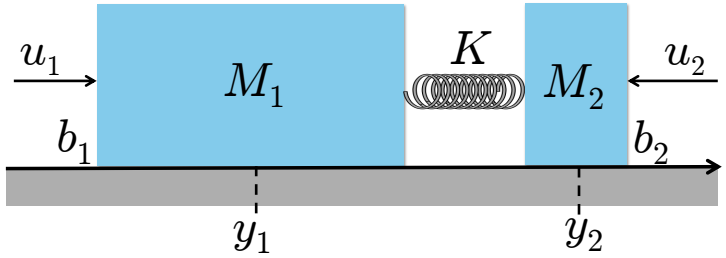
$$\min \|x^*(p)\|_2$$



$$\min \sum_{i=1}^{3} J_i(x_i, x_{-i})$$

EXAMPLE: GAME-THEORETIC MPC

- **2-mass-spring-damper system** with $N = 2$ agents, each controlling one force (u_1 or u_2)



- $M_1 = 3, M_2 = 1, K = 0.5, \beta_1 = 1.5, \beta_2 = 1$ (MKS units)

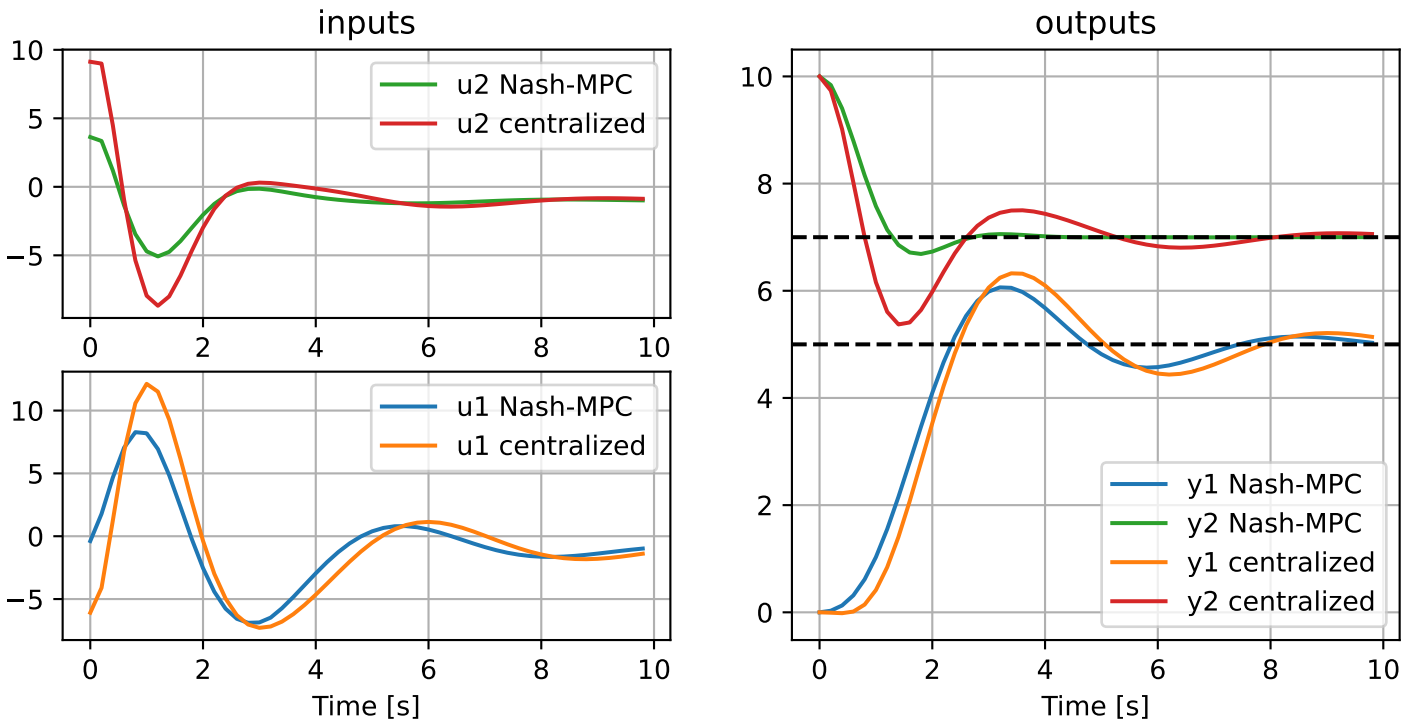
- Each agent minimizes its own MPC cost over a horizon of $N = 10$ steps:
  - **Agent 1**: $\min \sum (e_1 - e_2)^2 + (\Delta u_1 + \Delta u_2)^2$ (same tracking error, opposite Δu's)
  - **Agent 2**: $\min \sum e_2^2 + 0.1(\Delta u_2)^2$ (only on its own output and input increment)

- **State constraint**: $y_2 - y_1 \geq 1$ imposed over the first $N_c = 3$ steps

- **Parameters**: $p = (z(t), u(t-1), r(t)) \in \mathbb{R}^{4+2+2}$

EXAMPLE: GAME-THEORETIC MPC

- Solution: 4 critical regions with unique GNE, 3 with infinitely-many GNEs
- Split the latter using **minimum-norm criterion** -> **19 critical regions**
- Compare with **centralized MPC** minimizing sum of costs + state constraints



- Other criteria (v-GNE, minimum-norm, welfare) give similar closed-loop results

CONCLUSIONS

CONCLUSIONS

- **Active-learning** method can compute GNEs of a very broad class of problems, where **only best responses are measurable** and global constraints are known

- For the case of **parametric** quadratic costs and linear constraints, we can compute GNEs / vGNEs solutions **explicitly**

- **Current research**: **learn** approximate parametric **nonlinear** GNEP solutions

ERC Advanced Grant "COMPACT" (2024-2029)

Funded by the European Union

European Research Council
Established by the European Commission