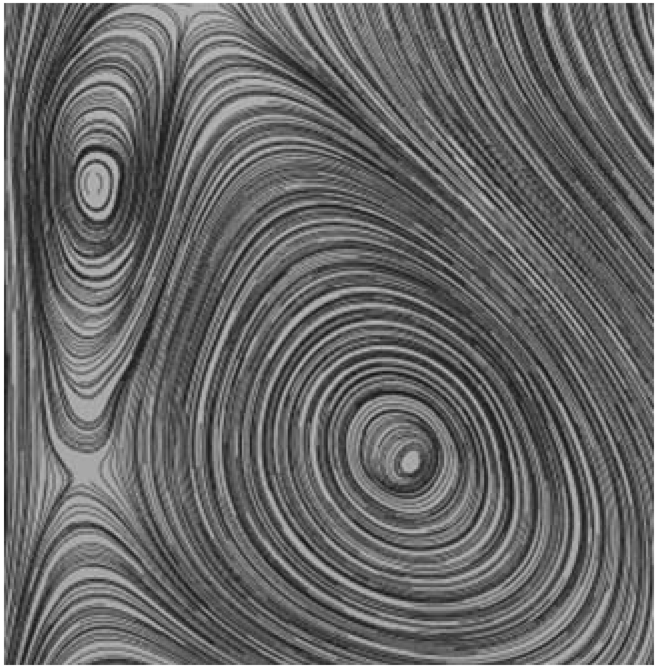
Part 1: diffeomorphic matching for imitation learning

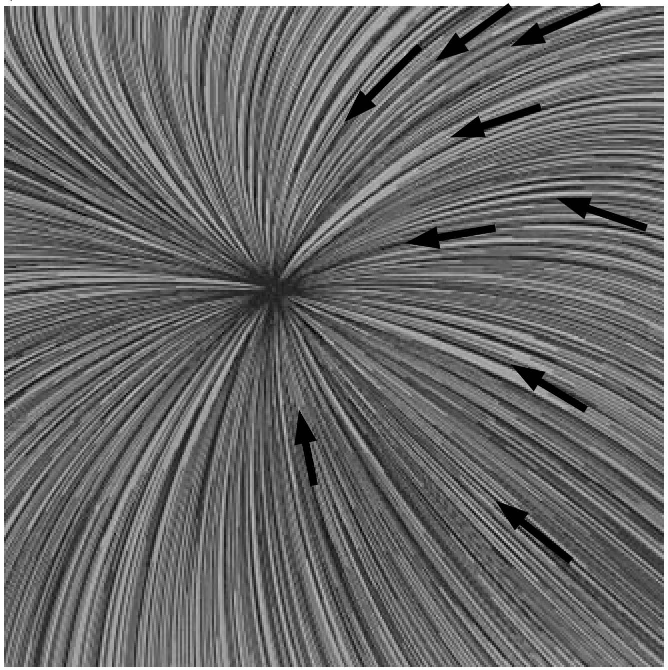
Nicolas Perrin perrin@isir.upmc.fr
CNRS / Sorbonne Université - ISIR

Problem: learn a vector field with asymptotic guarantees.

Specifically: global asymptotic stability.



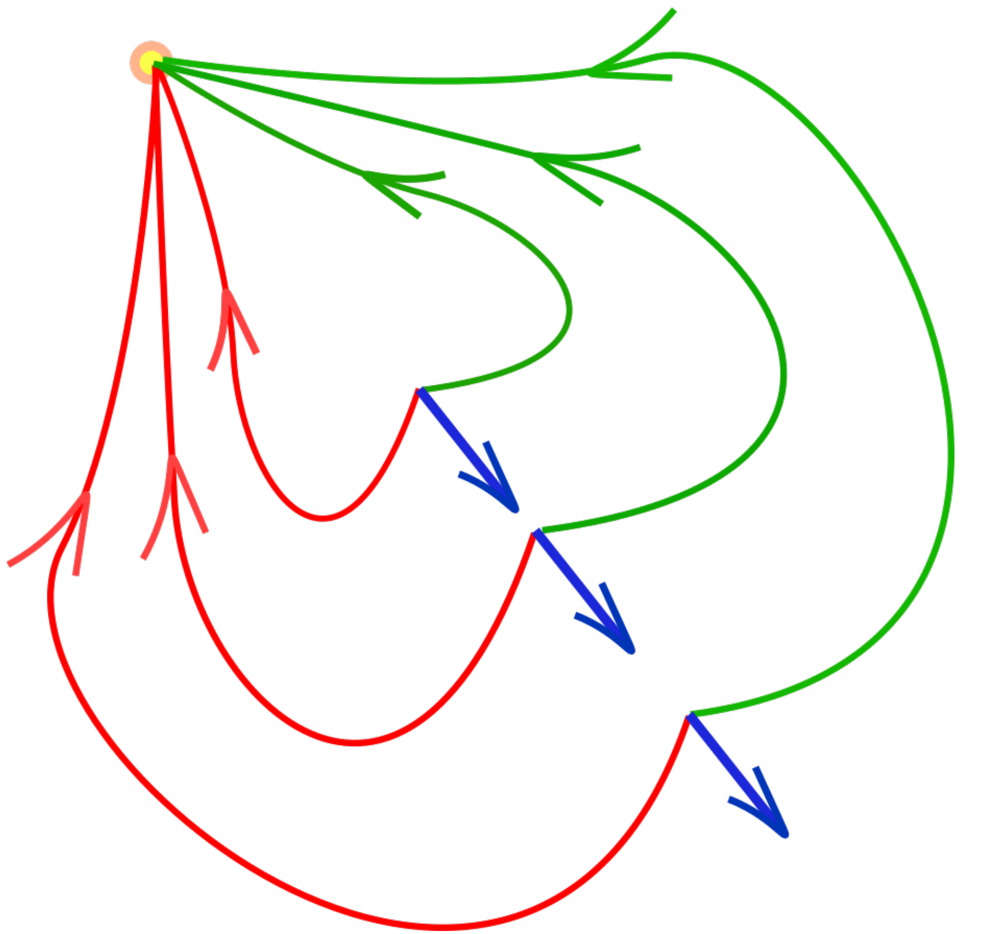
$\mathbf{v} = f_\theta(\mathbf{x})$



Often, f_θ has the form:

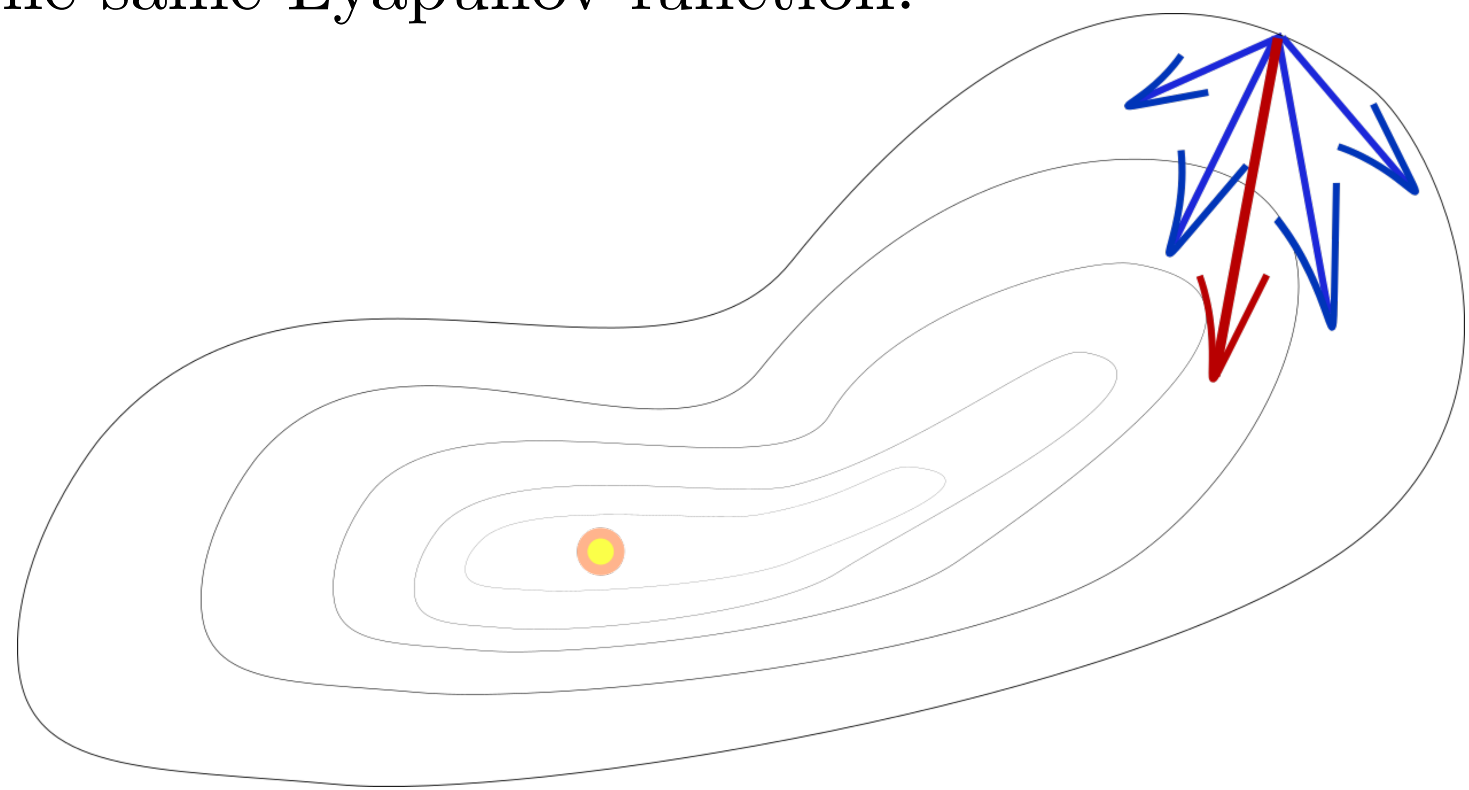
$$f_\theta = \sum_i \alpha_i g_{\theta_i}(\mathbf{x})$$

Issue: the sum of asymptotically stable vector fields is not asymptotically stable.

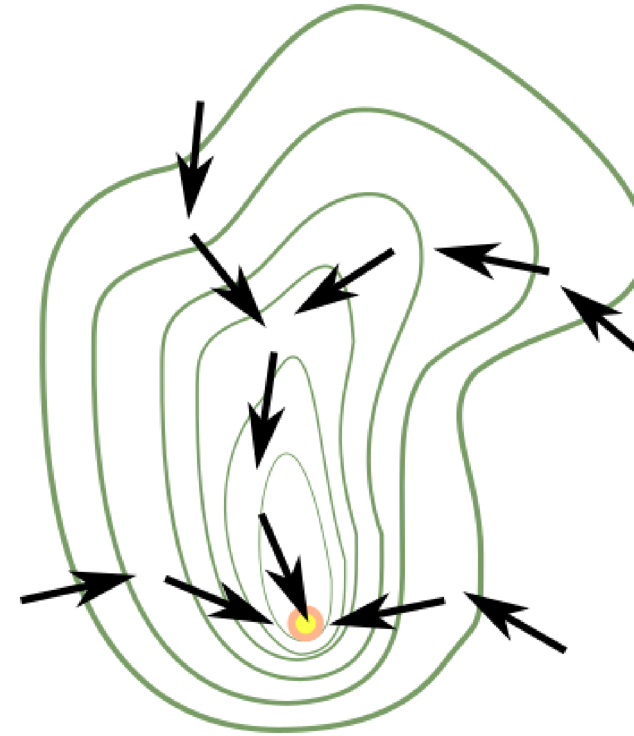


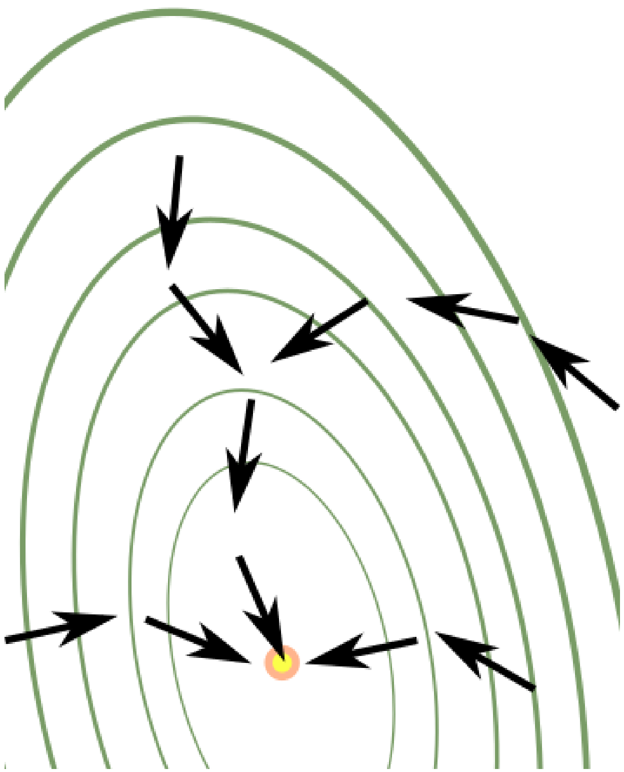
$$f_\theta = \sum_i \alpha_i g_{\theta_i}(\mathbf{x})$$

But: the sum is asymptotically stable if all the g_{θ_i}'s are compatible with the same Lyapunov function.



It is difficult to construct complex Lyapuov functions...





...so the seminal approach SEDS [*] was based on quadratic Lyapunov functions, which is quite restrictive.

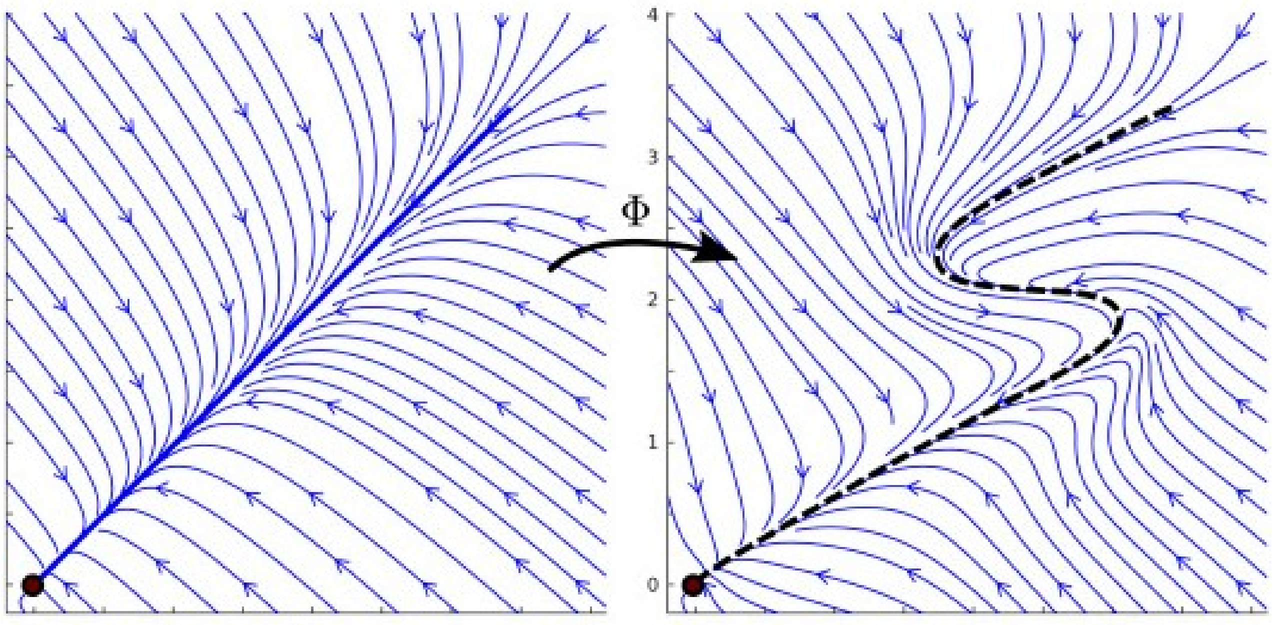
[*] Khansari-Zadeh et al., "Learning stable nonlinear dynamical systems with Gaussian mixture models", IEEE Trans. on Robotics 2011

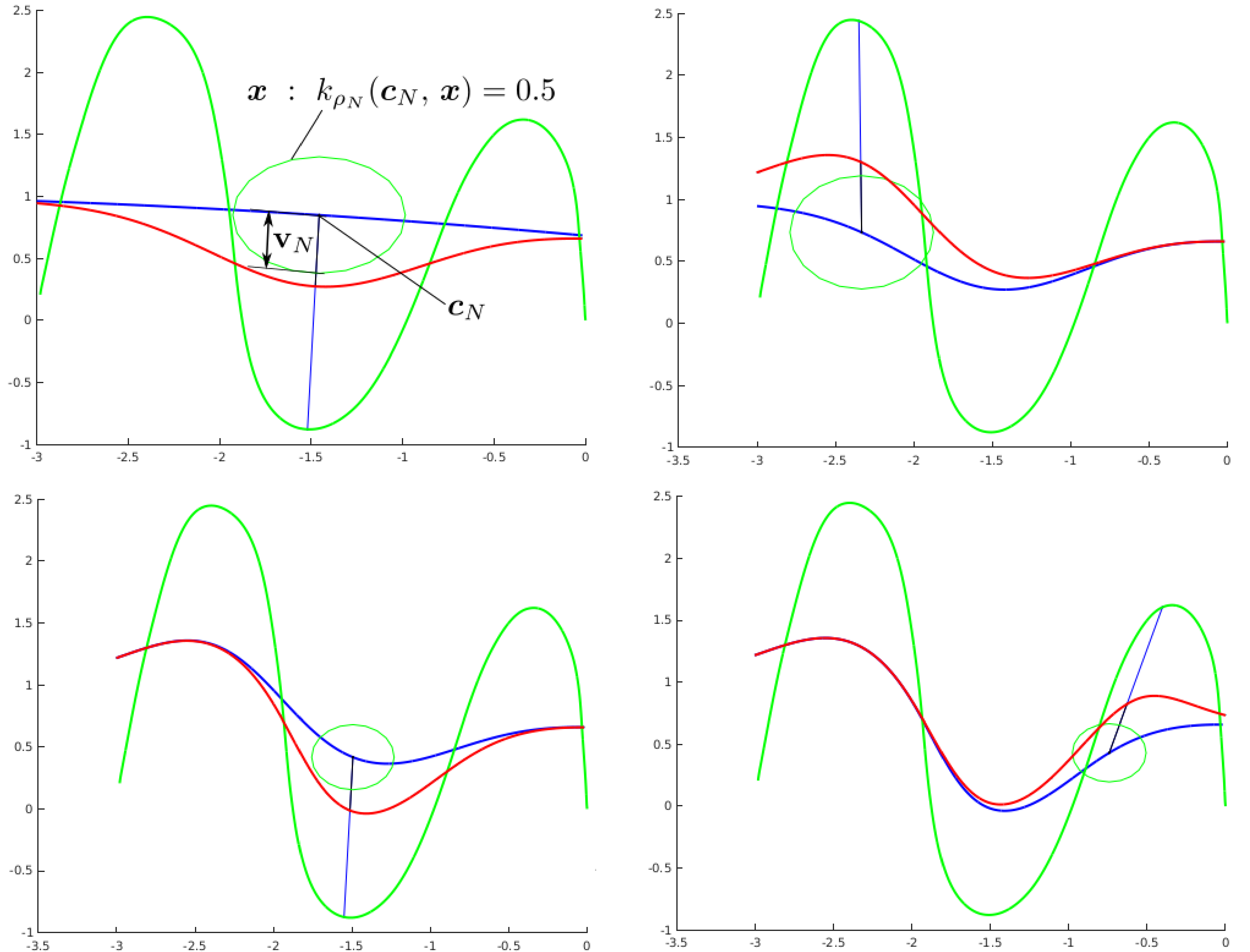
More expressive convex sets of Lyapunov candidates have been used (e.g. WSAQF [*]) to turn the search of a Lyapunov function to an optimization problem, but they are still quite restrictive.

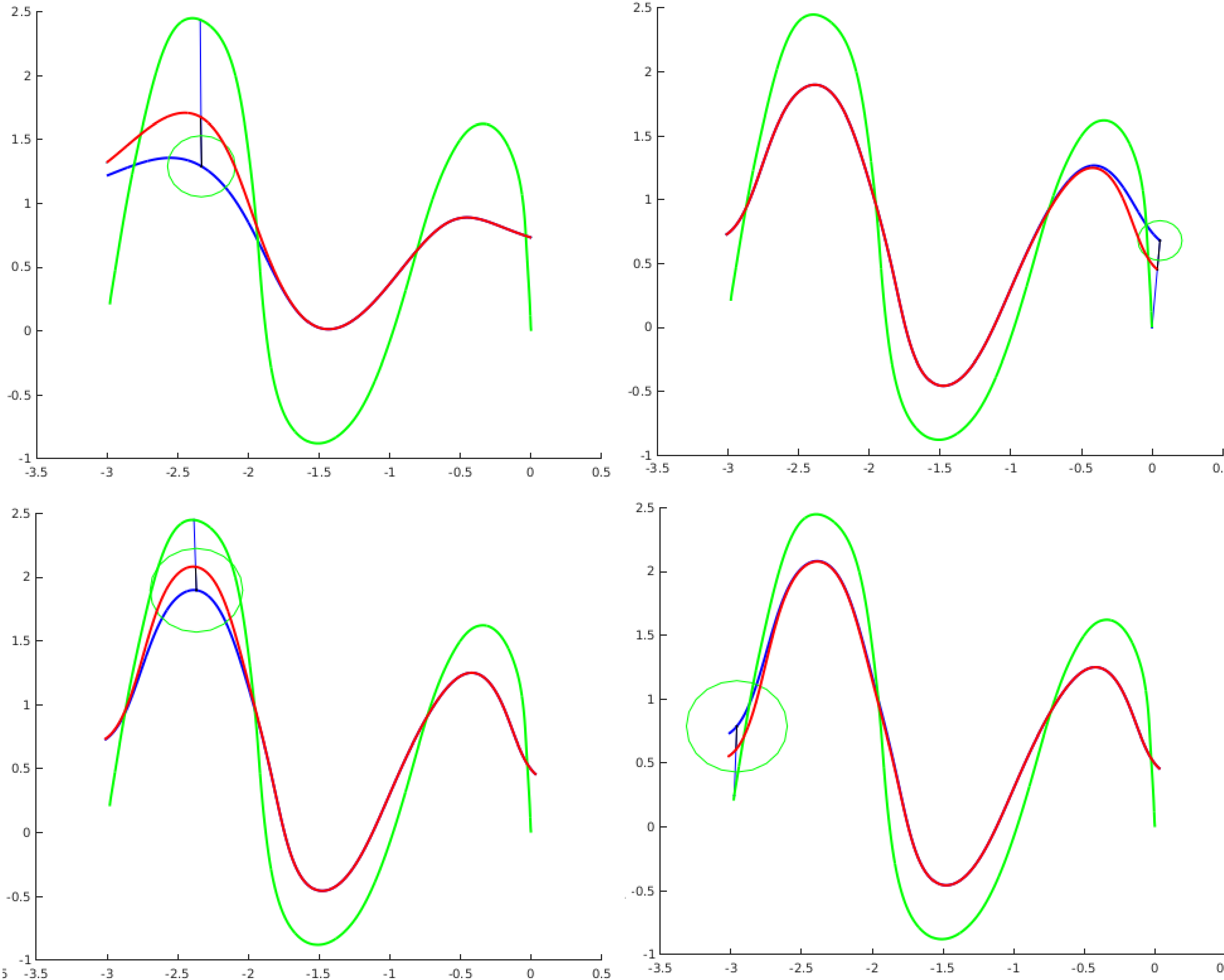
[*] Khansari-Zadeh et al., "Learning Control Lyapunov Function to Ensure Stability of Dynamical System-based Robot Reaching Motions", Robotics and Autonomous Systems 2014

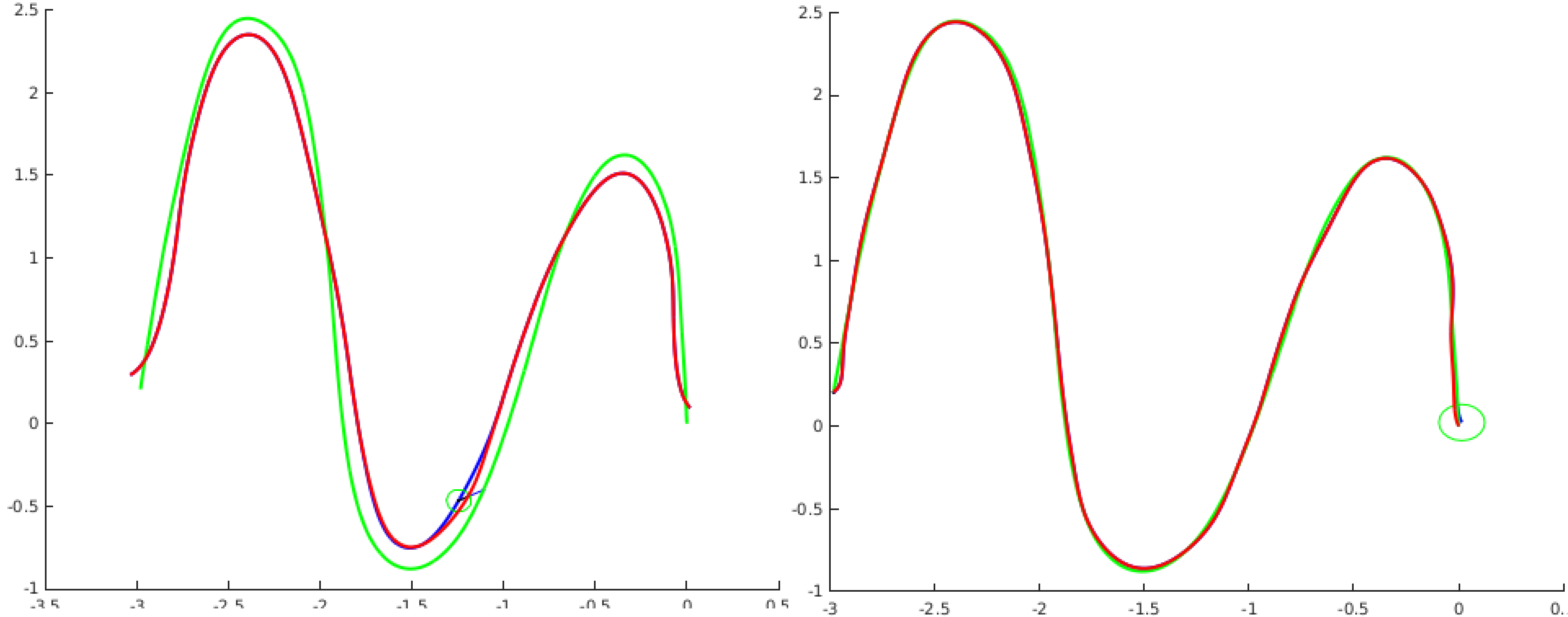
We proposed a method based on another property:

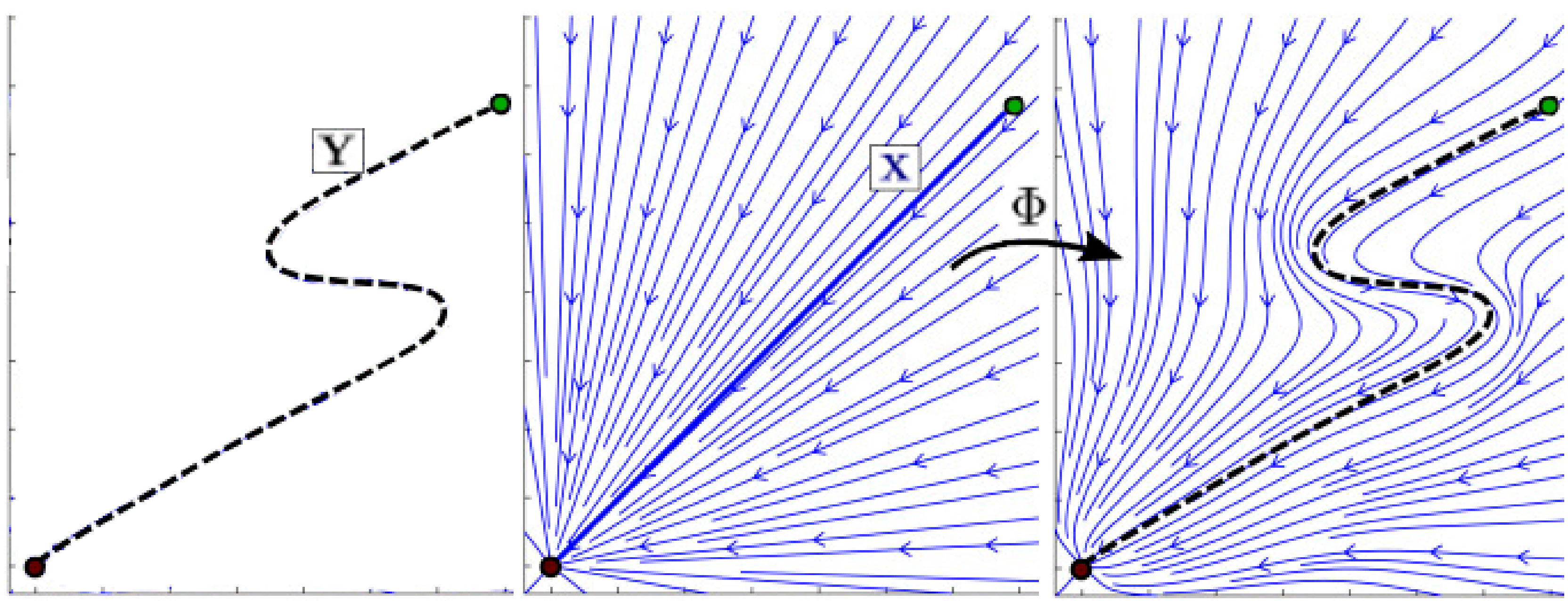
A stable vector field remains stable after application of a diffeomorphism.

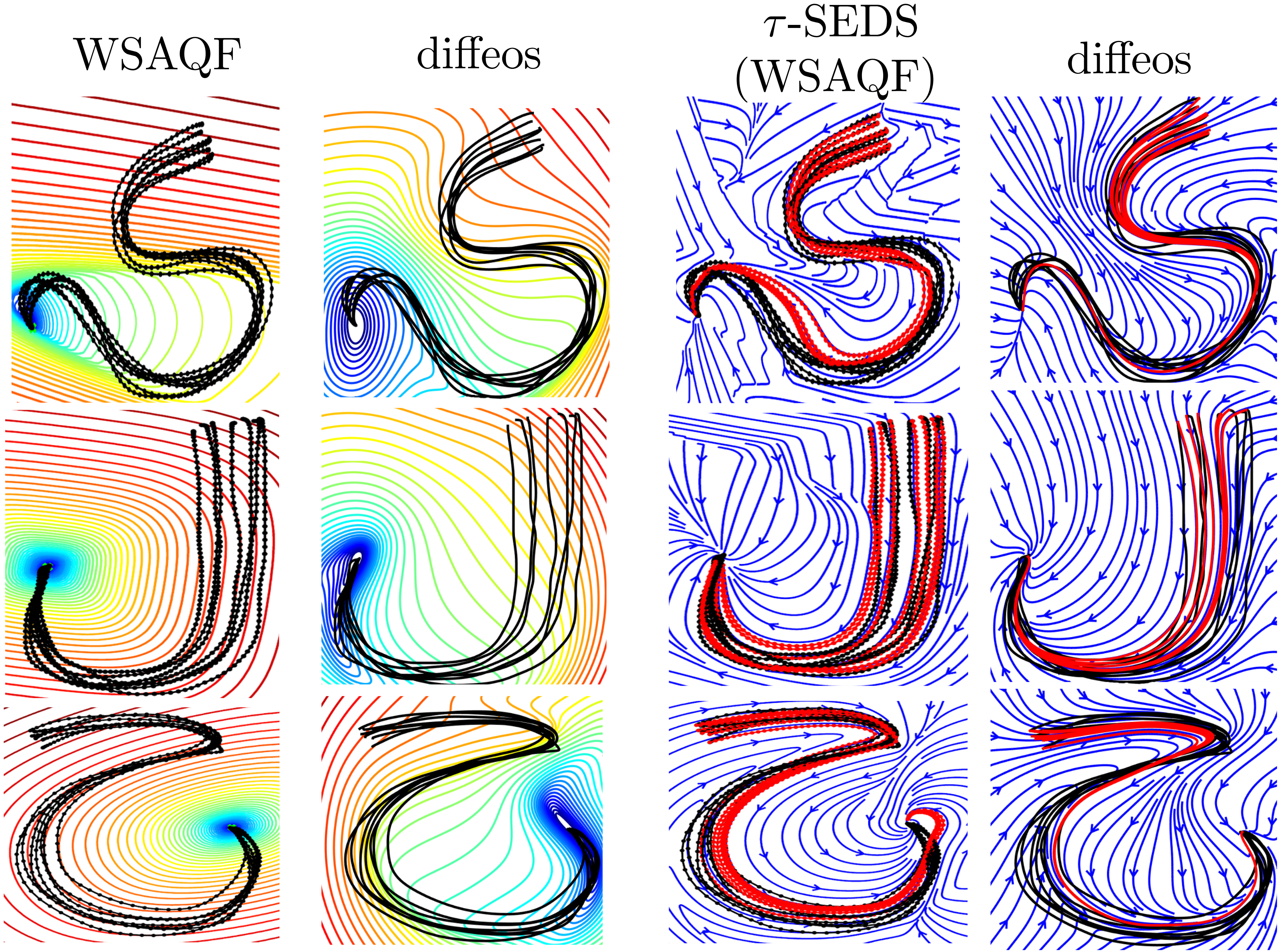



$\boldsymbol{x} : k_{\rho_N}(\boldsymbol{c}_N, \boldsymbol{x}) = 0.5$
$\mathbf{v}_N$
$\boldsymbol{c}_N$


2.5 2 1.5 1 0.5 0 -0.5 -1
-3.5 -3 -2.5 -2 -1.5 -1 -0.5 0 0.5
2.5 2 1.5 1 0.5 0 -0.5 -1
-3.5 -3 -2.5 -2 -1.5 -1 -0.5 0 0.5
2.5 2 1.5 1 0.5 0 -0.5 -1
-3.5 -3 -2.5 -2 -1.5 -1 -0.5 0 0.5
2.5 2 1.5 1 0.5 0 -0.5 -1
-3.5 -3 -2.5 -2 -1.5 -1 -0.5 0 0.5


2.5
2
1.5
1
0.5
0
-0.5
-1
-3.5
-3
-2.5
-2
-1.5
-1
-0.5
0
0.5

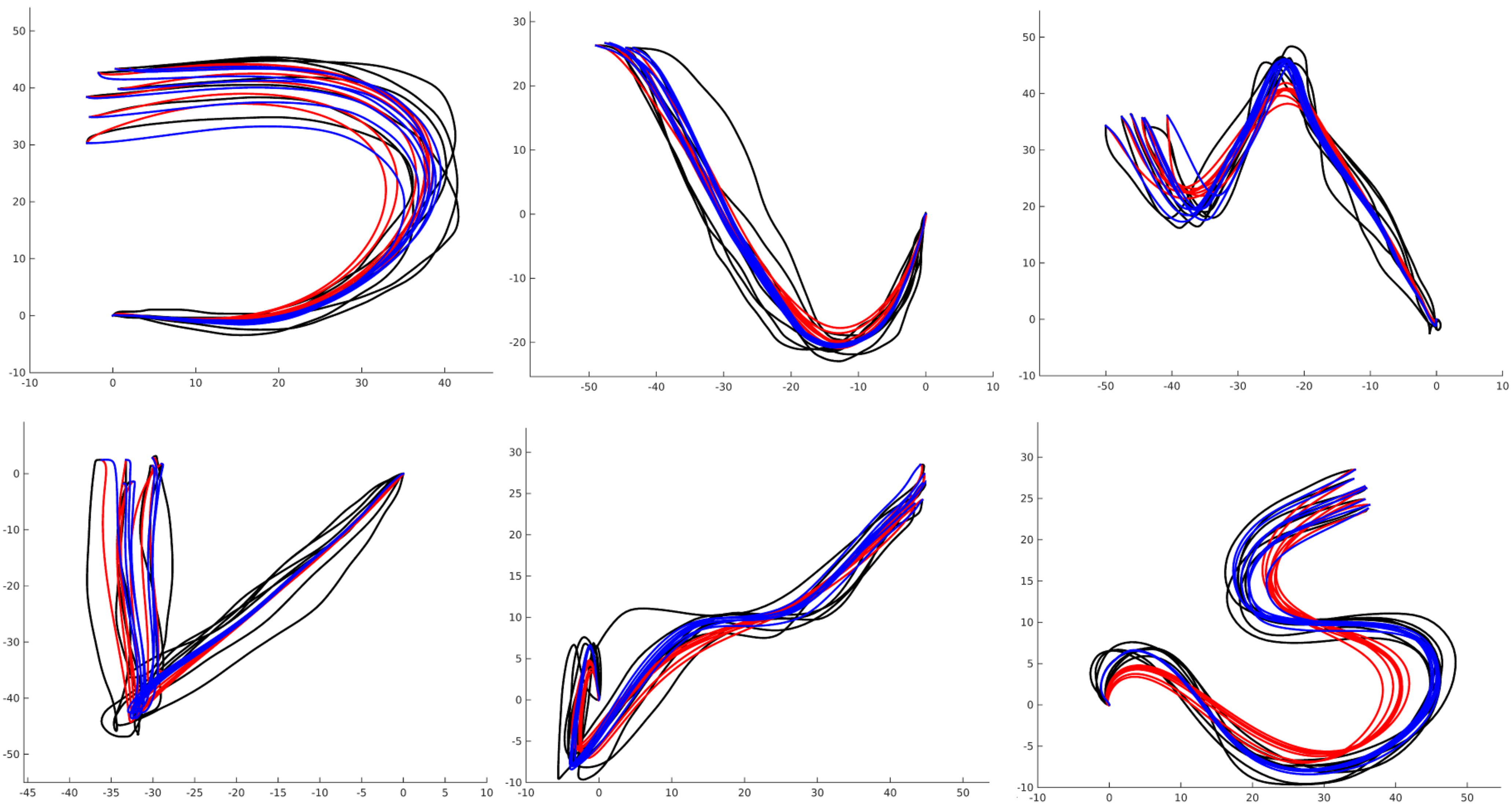

Y
X
Φ


WSAQF
diffeos
τ-SEDS (WSAQF)
diffeos

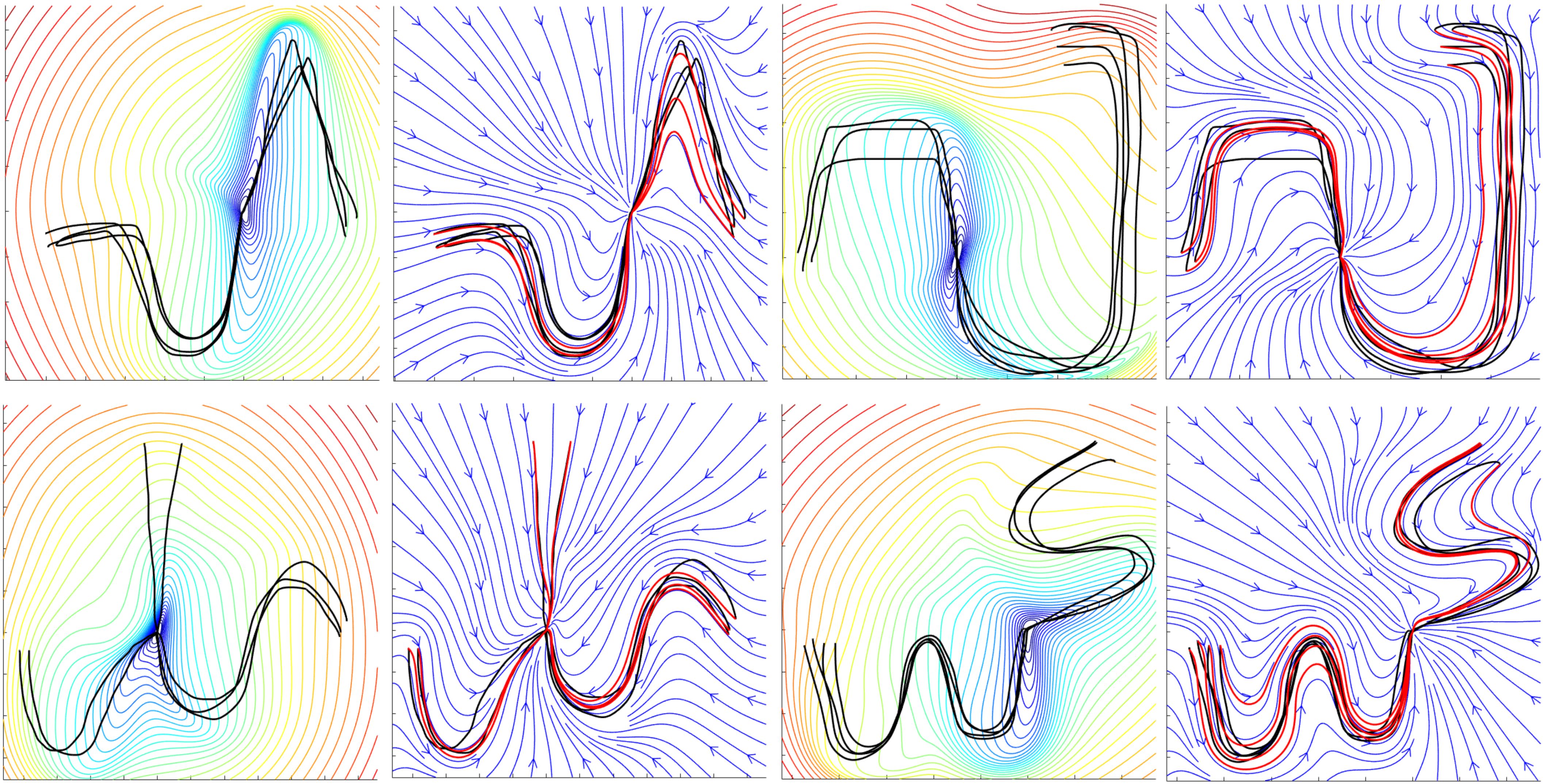
Speed and scalability:

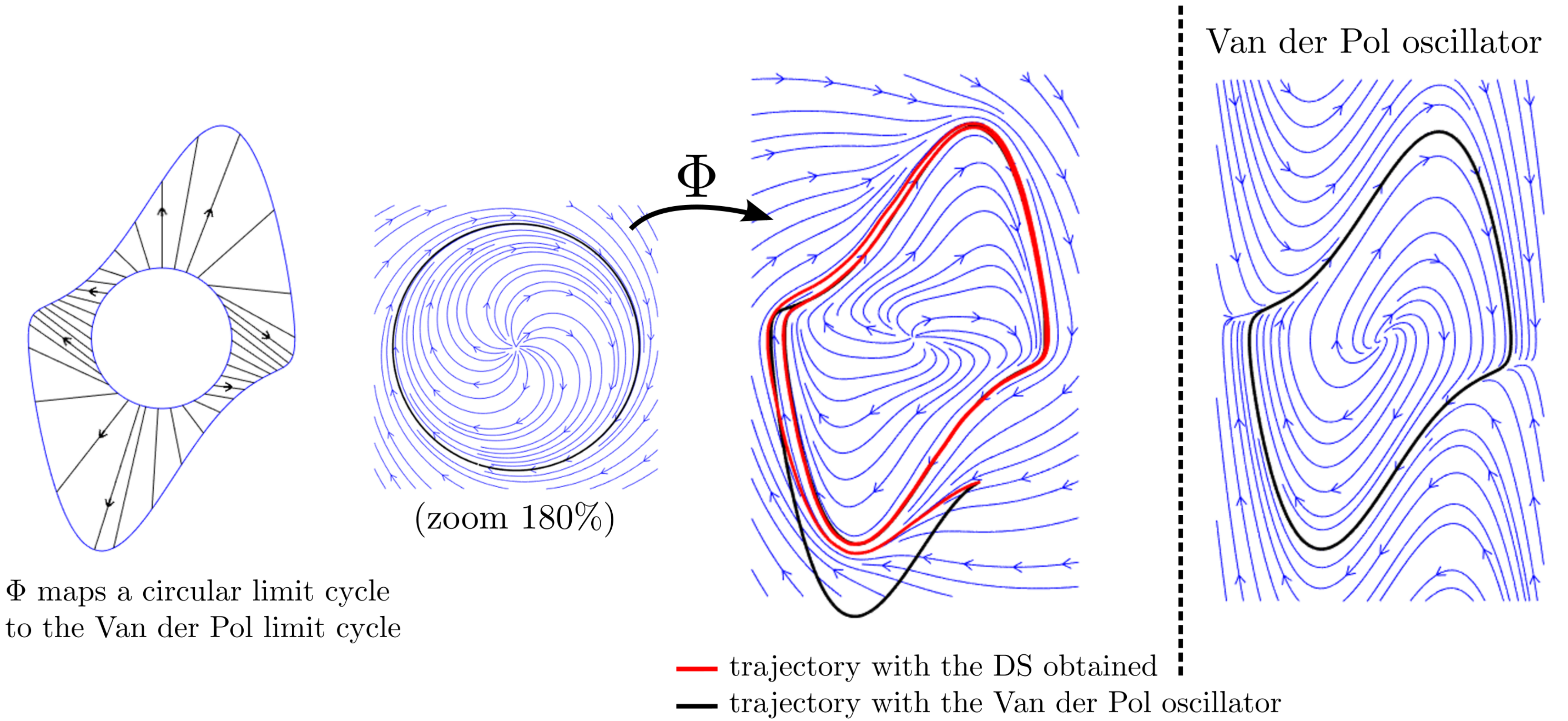
	N	our algorithm	LDDMM
Learning: average duration of the construction of Φ	20	0.25 s	2.78 s
	50	0.25 s	14.5 s
	100	0.26 s	53.3 s
Forward evaluation: average duration of the computation of $\Phi(\mathbf{X})$	20	3.05 ms	157 ms
	50	3.35 ms	804 ms
	100	3.72 ms	3130 ms
Backward evaluation: average duration of the computation of $\Phi^{-1}(\mathbf{Y})$	20	29.8 ms	145 ms
	50	35 ms	798 ms
	100	38.5 ms	3110 ms
Accuracy: average value of $\mathsf{dist}(\Phi(\mathbf{X}), \mathbf{Y})$	20	3.49×10^{-3}	18.2×10^{-3}
	50	8.32×10^{-3}	22.2×10^{-3}
	100	9.51×10^{-3}	22×10^{-3}

Dimensionality	2	4	6	8	10	12	14
Training time with SEDS (s)	10.74	26.96	81.14	110.0	251.1	473.3	875.3
Training time with our method (s)	0.482	0.456	0.498	0.522	0.575	0.576	0.604



Possible extensions:





Φ
(zoom 180%)
Van der Pol oscillator
Φ maps a circular limit cycle
to the Van der Pol limit cycle
trajectory with the DS obtained
trajectory with the Van der Pol oscillator

Part 2: a specific approach to reinforcement learning for robotics?

Nicolas Perrin perrin@isir.upmc.fr
CNRS / Sorbonne Université - ISIR

1. Exploration
2. Feature/structure engineering
3. Composition

Preamble: a few words on RL

Objective: optimize a policy $\pi_\rho(s)$ to collect reward.

A value function $V_\theta(s)$ or $Q_\mu(s, a)$ is also trained to approximate $E_{\pi^*, s_0=s}\left\{\sum_i \gamma^i r(s_i)\right\}$. It is used as a substitute to the rewards to get gradients for ρ updates.

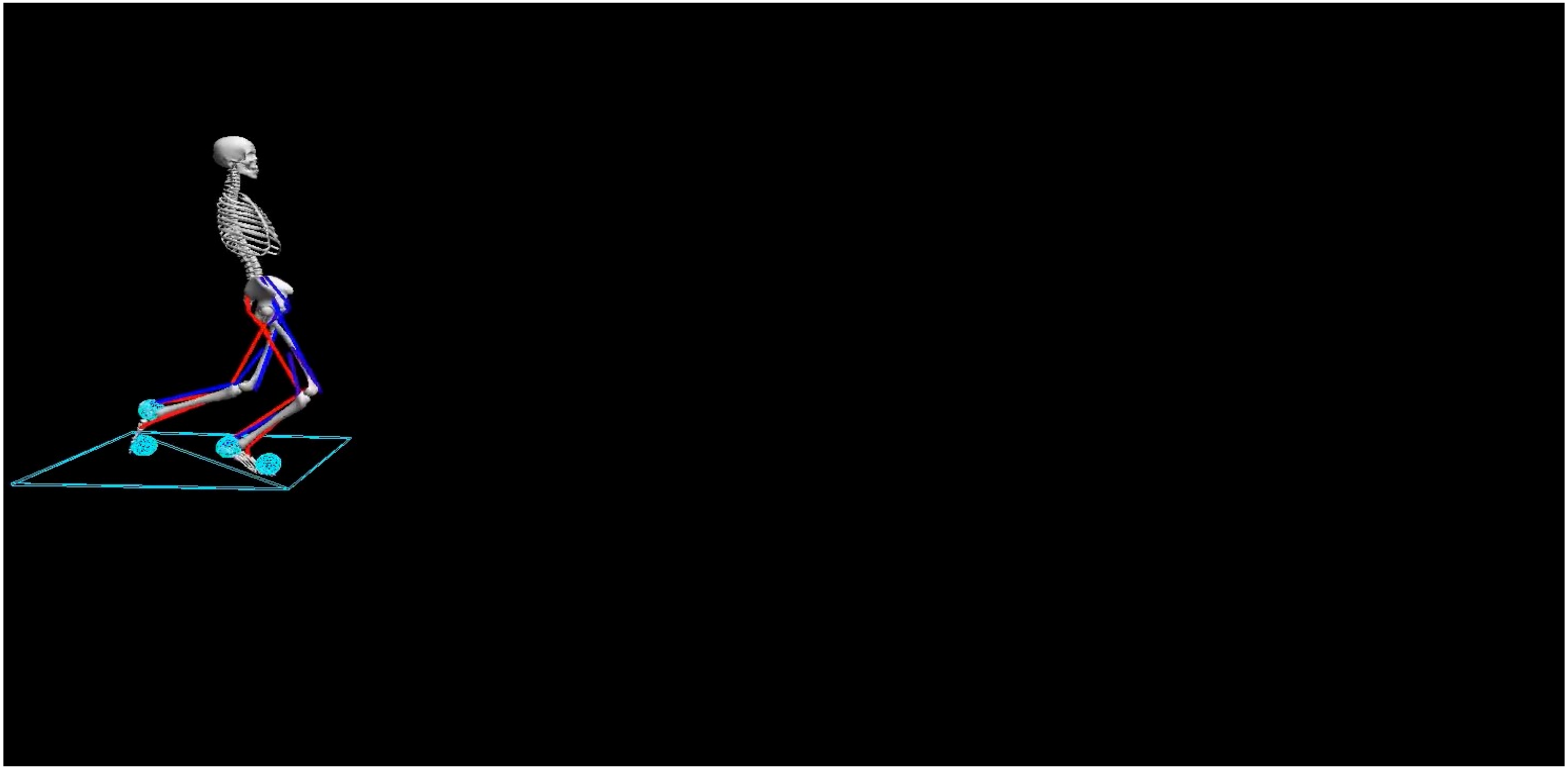
On-policy or Off-policy methods.

Exploration is interesting both to find new solutions and to get good gradients for V_θ.

1. Exploration

Better exploration may be obtained via motion planning, exploiting *resets* and *determinism* in simulation (and maybe the fact that outputs are torques).

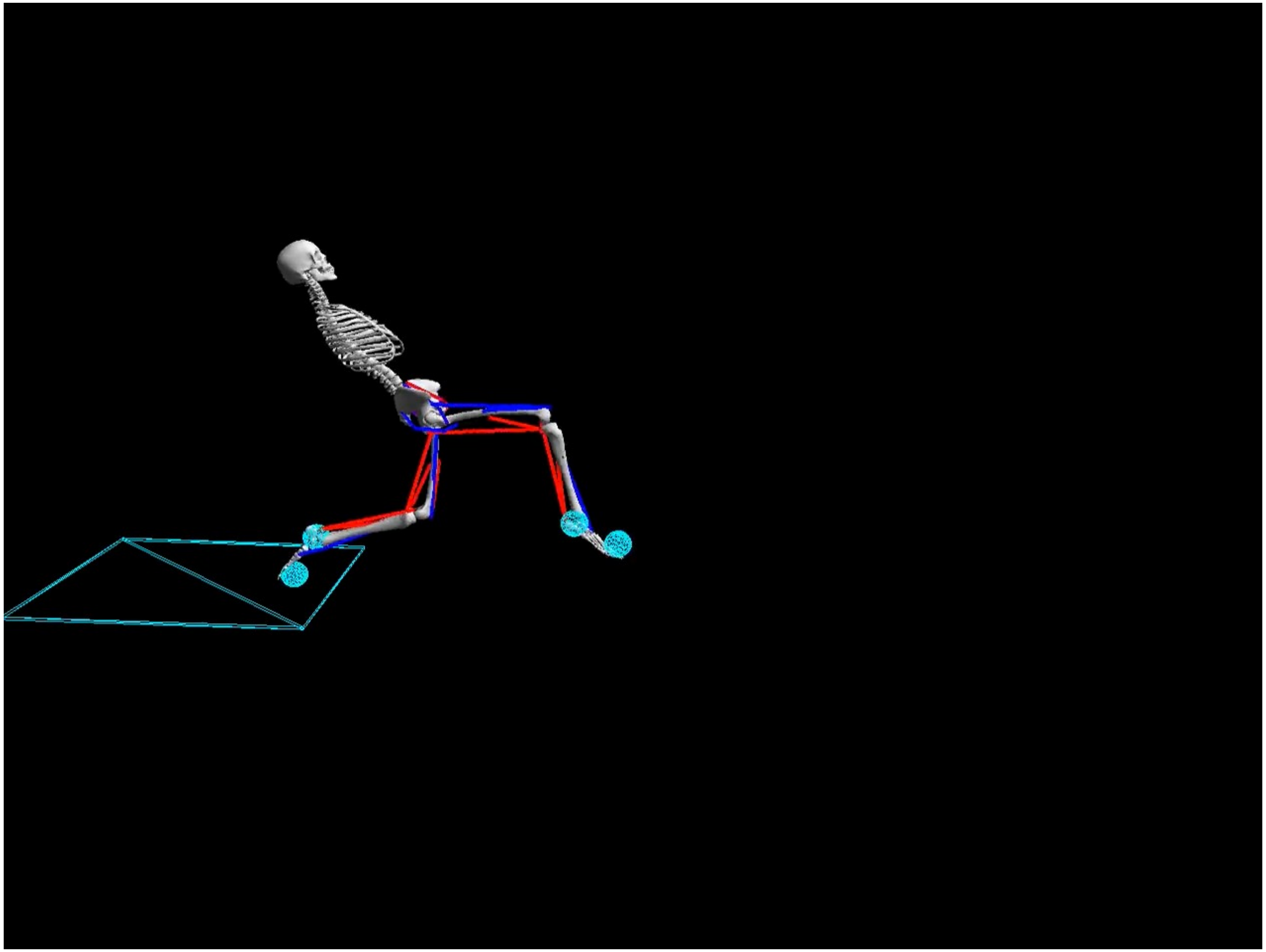
Attempts on the learning to run challenge:



Attempts on the learning to run challenge:

Attempts on the learning to run challenge:

Attempts on the learning to run challenge:

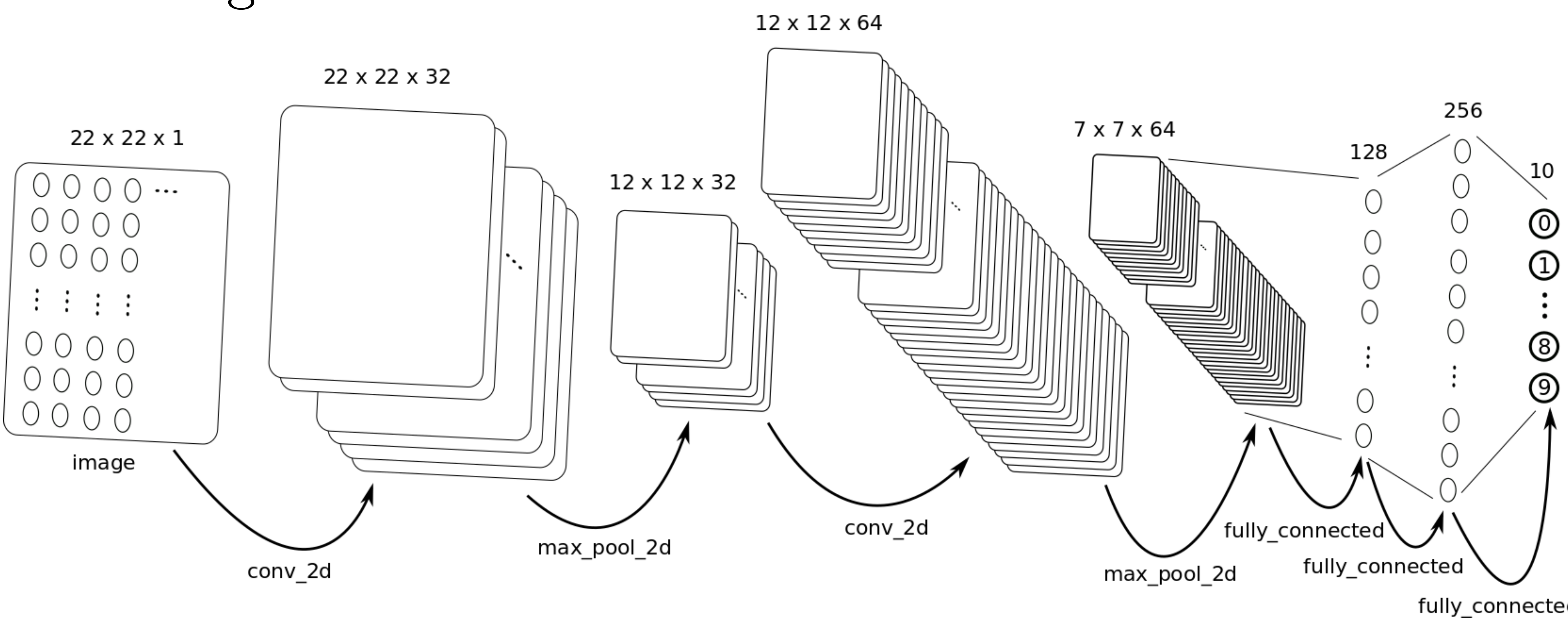


2. Feature/structure engineering

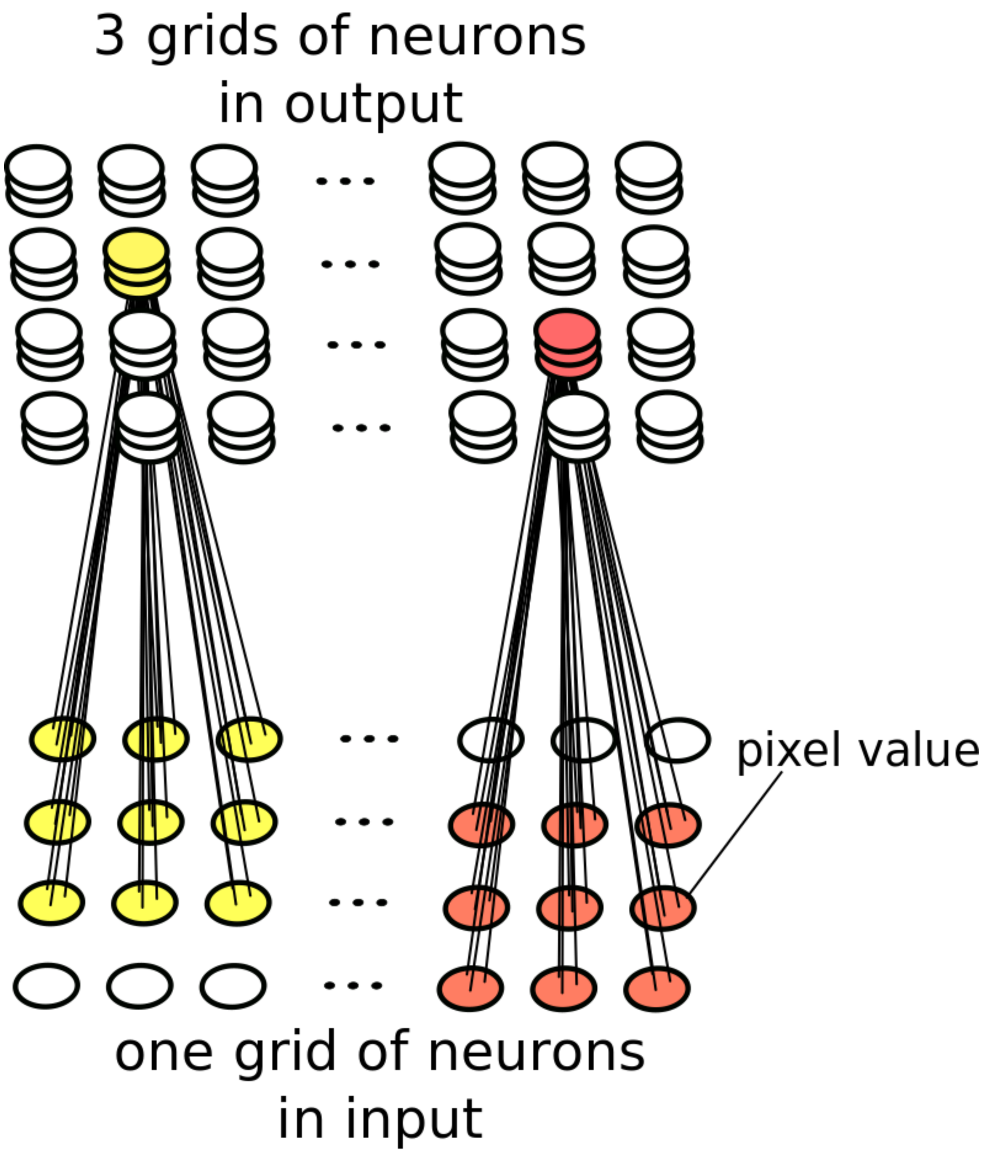
A function transforms inputs into outputs: $\boxed{a = f_\theta(s)}$

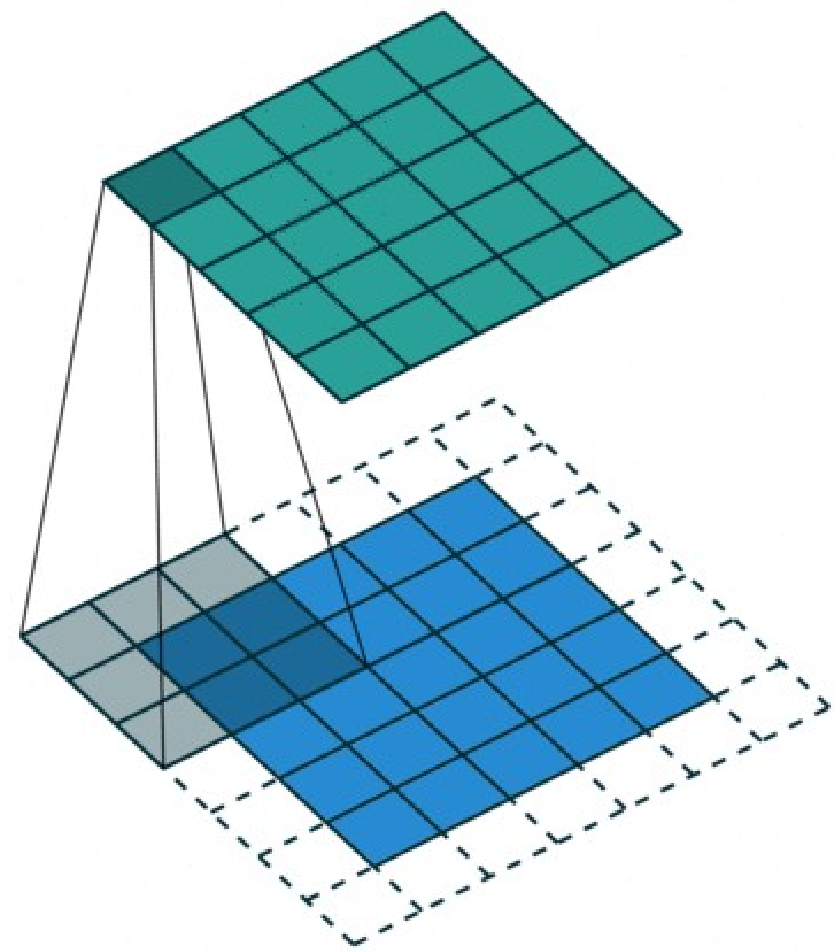
Learning objective: optimize θ.

A typical structure when f is a deep neural network for image classification:



Convolutions:





Example of convolution filter (Sobel):

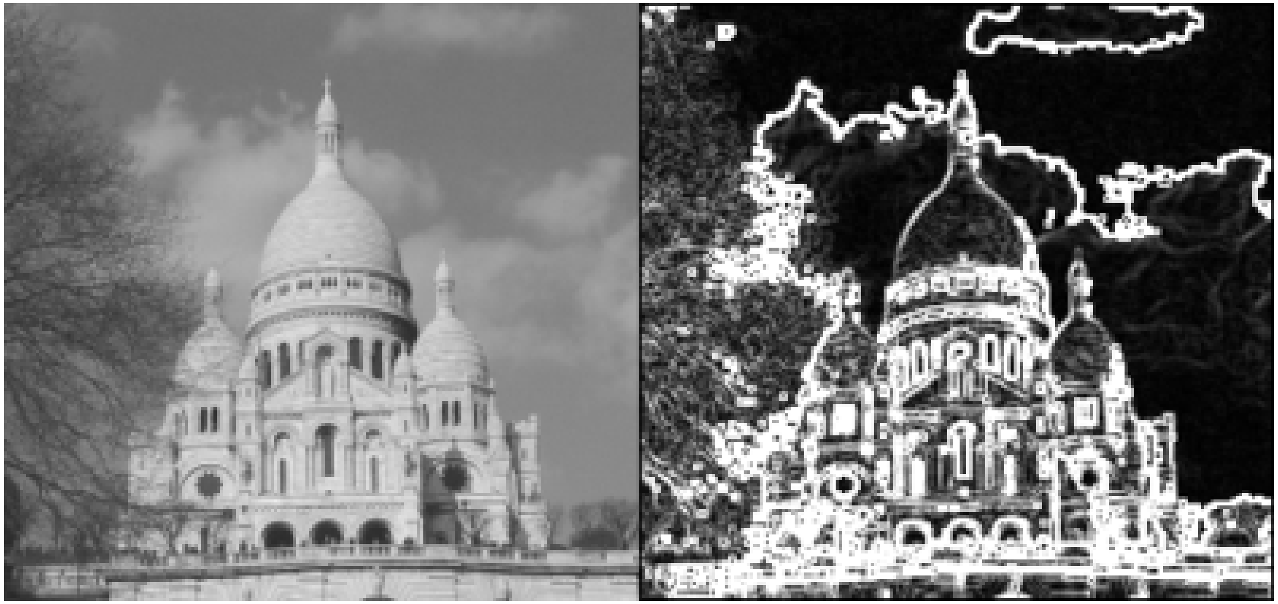
-1	0	+1
-2	0	+2
-1	0	+1

x filter

+1	+2	+1
0	0	0
-1	-2	-1

y filter

Two Sobel filters (x and y) applied to an image:



Convolutions are great feature extractors for images.

Can we find something similar when the inputs are either:

- joint angles and velocities + free-floating base position and velocity,
- or position and velocities for the floating base and every rigid body of the robot?

(do we also add torques, some contact points and contact forces?)

In other words: are there robotic control equivalents of convolutions?

Some “canonical” operations or tools in robotic control:

- Going from joint space to Euclidean space, and vice versa.
- PIDs.
- Solving OCPs.
- Euler-Lagrange equation.
- Using simplified dynamical models (=convexifying optimization problems)?
- Relying on important quantities, such as kinetic energy, center of mass, centroidal angular momentum, center of pressure, end-effector positions
- Composing tasks, . . .

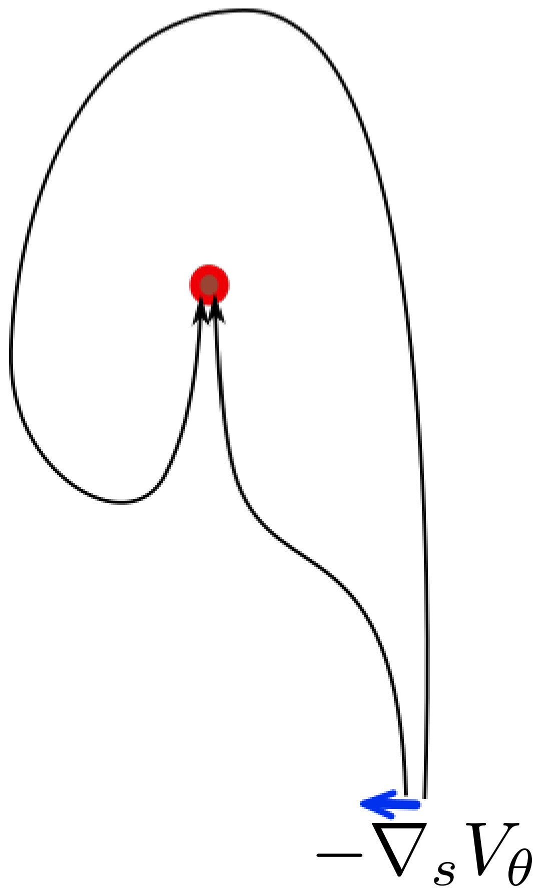
Actor-critic RL: train both $\pi_\rho(s)$ and $V_\theta(s) = E_{\pi^*, s_0=s}\left\{\sum_i \gamma^i r(s_i)\right\}$.

π_ρ is trained just because it is hard to compute $\arg\max_a V_\theta(s')$ (or $\arg\max_a (Q_\mu(s,a)$ in the case of a state-action value function).

In robotics, training π_ρ may be a bad idea:

- the actions are complex and need to be precise;
- they could probably be obtained analytically because a (relatively) good model of the dynamics is known;
- focusing on learning costs (the value function) may be better because typically costs do not have to be as precise as actions.

$$\mathrm{a}^* = \arg\max_a \{\mathrm{V}_\theta(s') | s \to_a s'\}$$

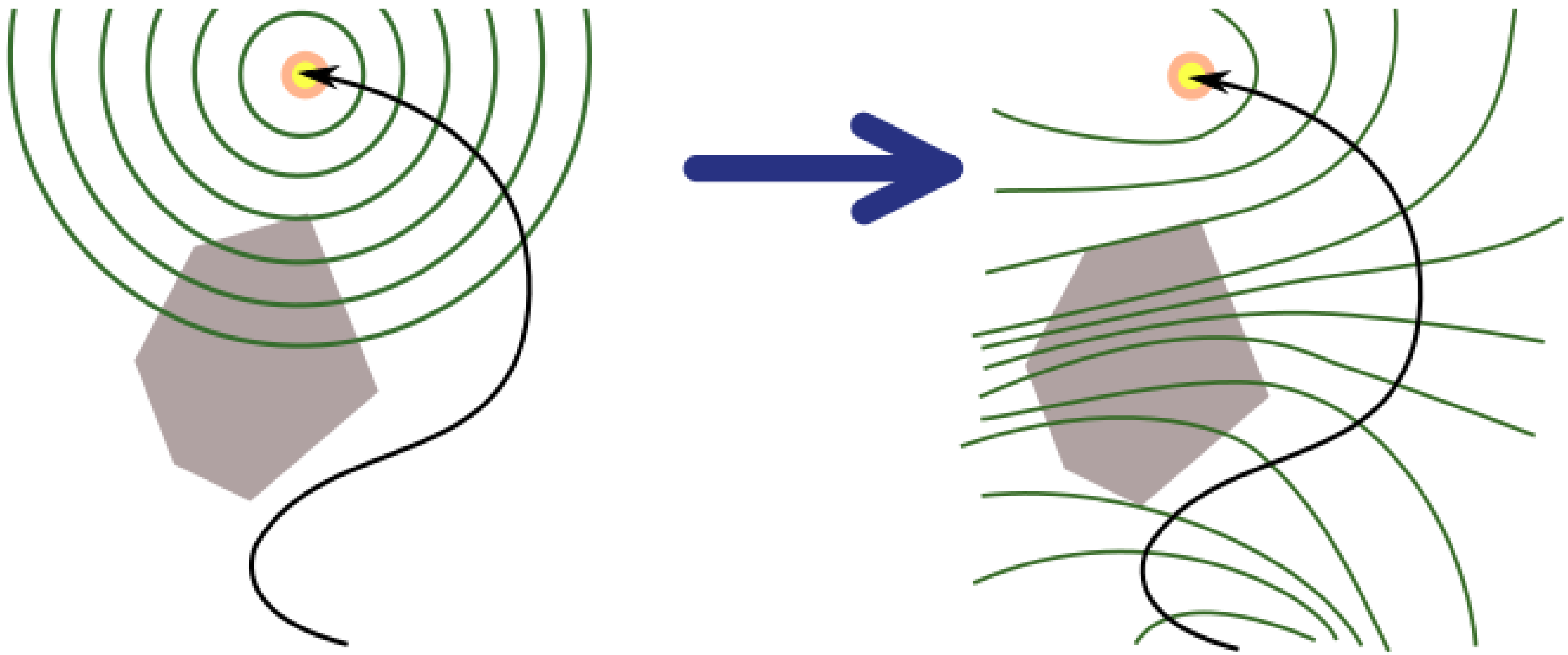


$s' = s + \delta s$:

$$\max_{\delta s} V_\theta(s + \delta s)$$
$$\|\delta s\| \leq K$$

$s' = s(t + \delta t) = s + b(s) + M(s)a$:

$$\max_a V_\theta(s + b(s) + M(s)a)$$
$$\|a\| \leq K_a$$



Other potential idea: when learning $V_\theta(s)$ or $Q_\mu(s,a)$, put redundancies inside the functions to facilitate the discovery of good correlations (and thus the training of V_θ or Q_μ):

$$V_\theta(s) = W_\theta(q, \dot{q}, x(q), \dot{x}(q, \dot{q}), f_c)$$

3. Composition

?

SAC-X (Google Deepmind)