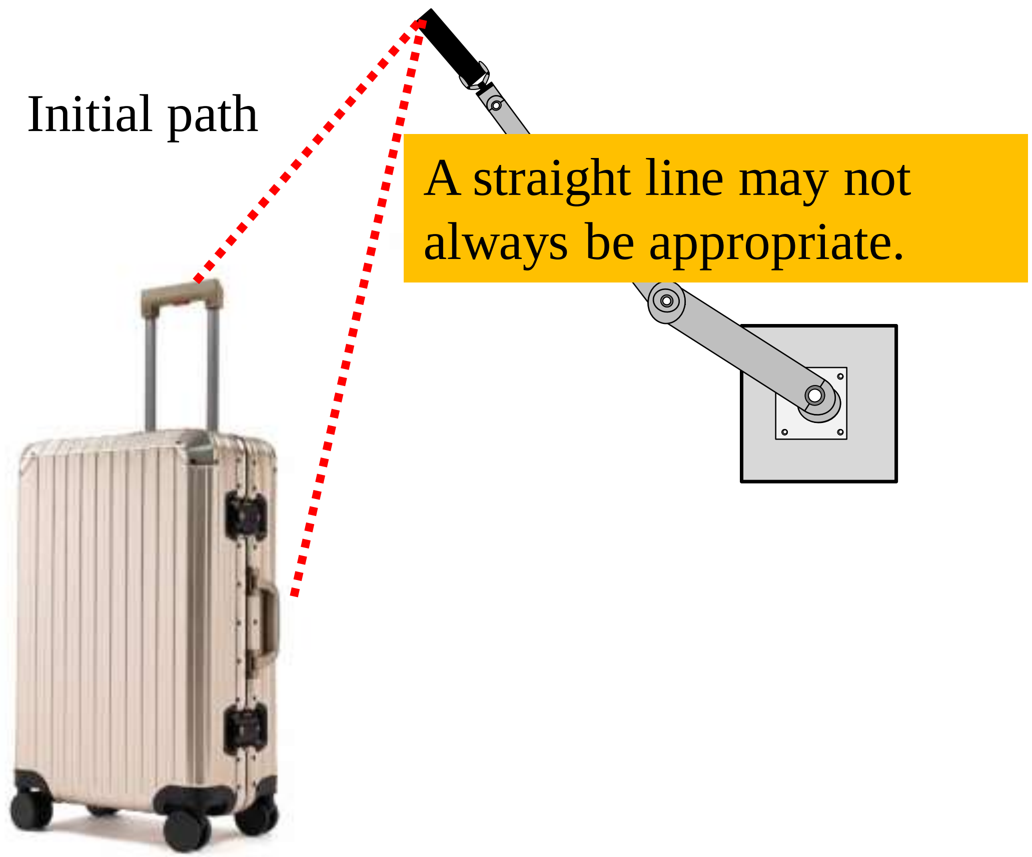
PART II

COUPLING AND MODULATING CONTROLLERS

Chapter 8
Adapting and Modulating an Existing Control Law

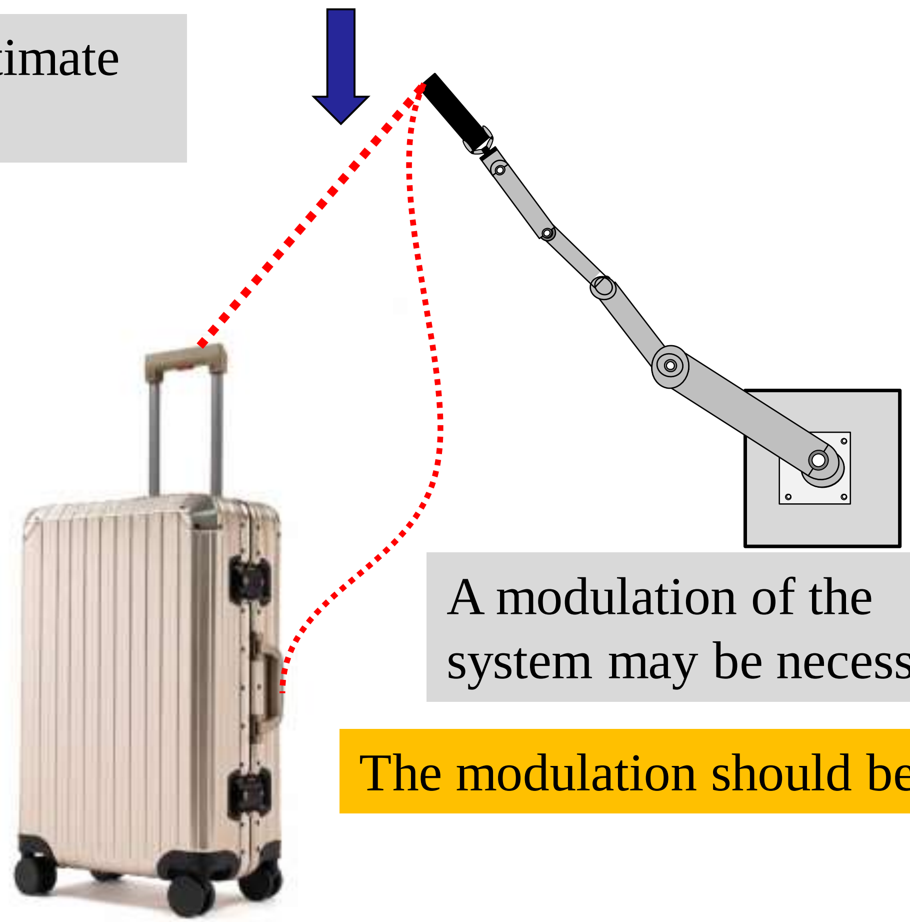
Adapting a controller



Adapting a controller

untouched

One should not re-estimate the entire system.

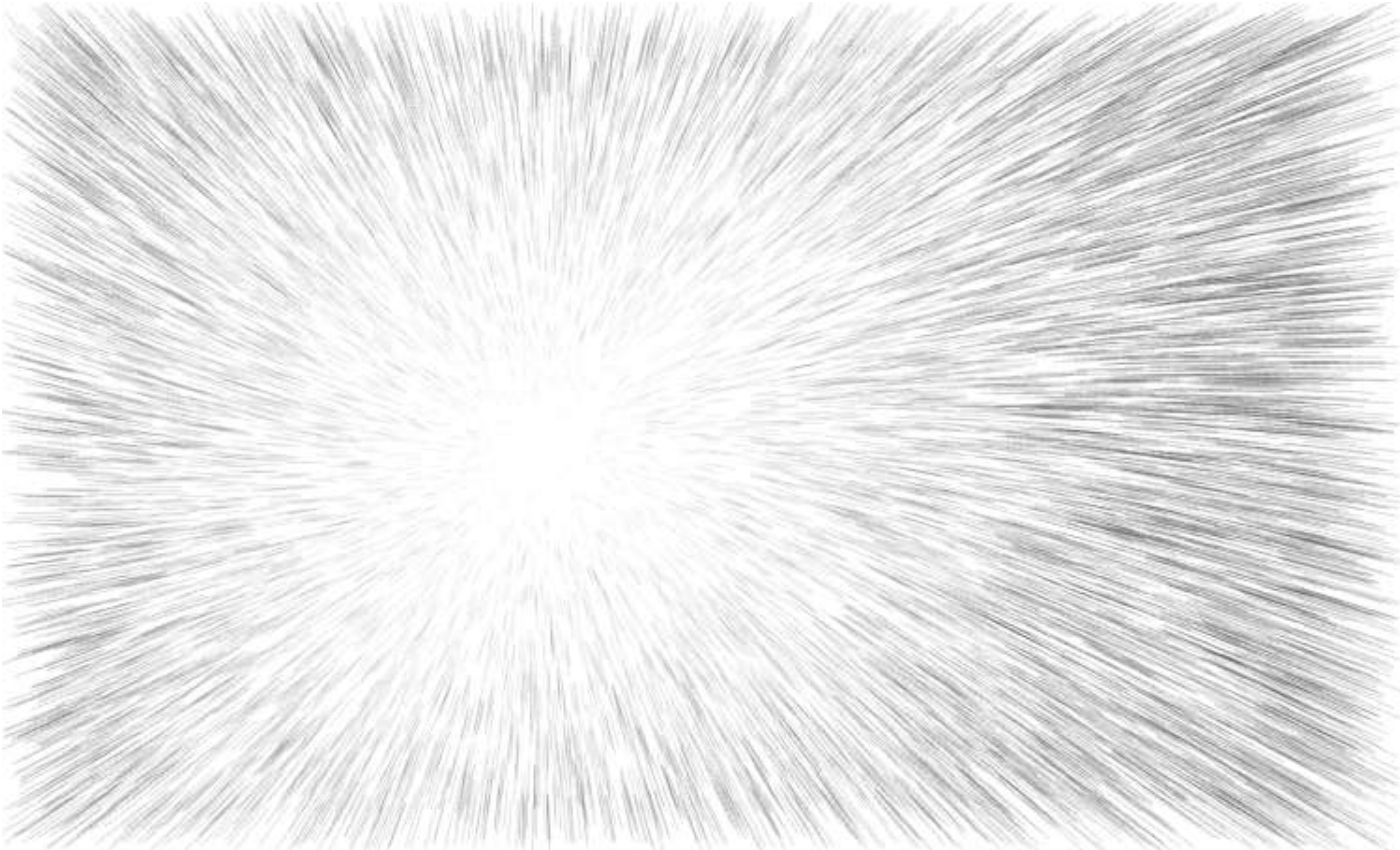


A modulation of the system may be necessary.

The modulation should be ***local***.

Initial – Nominal DS

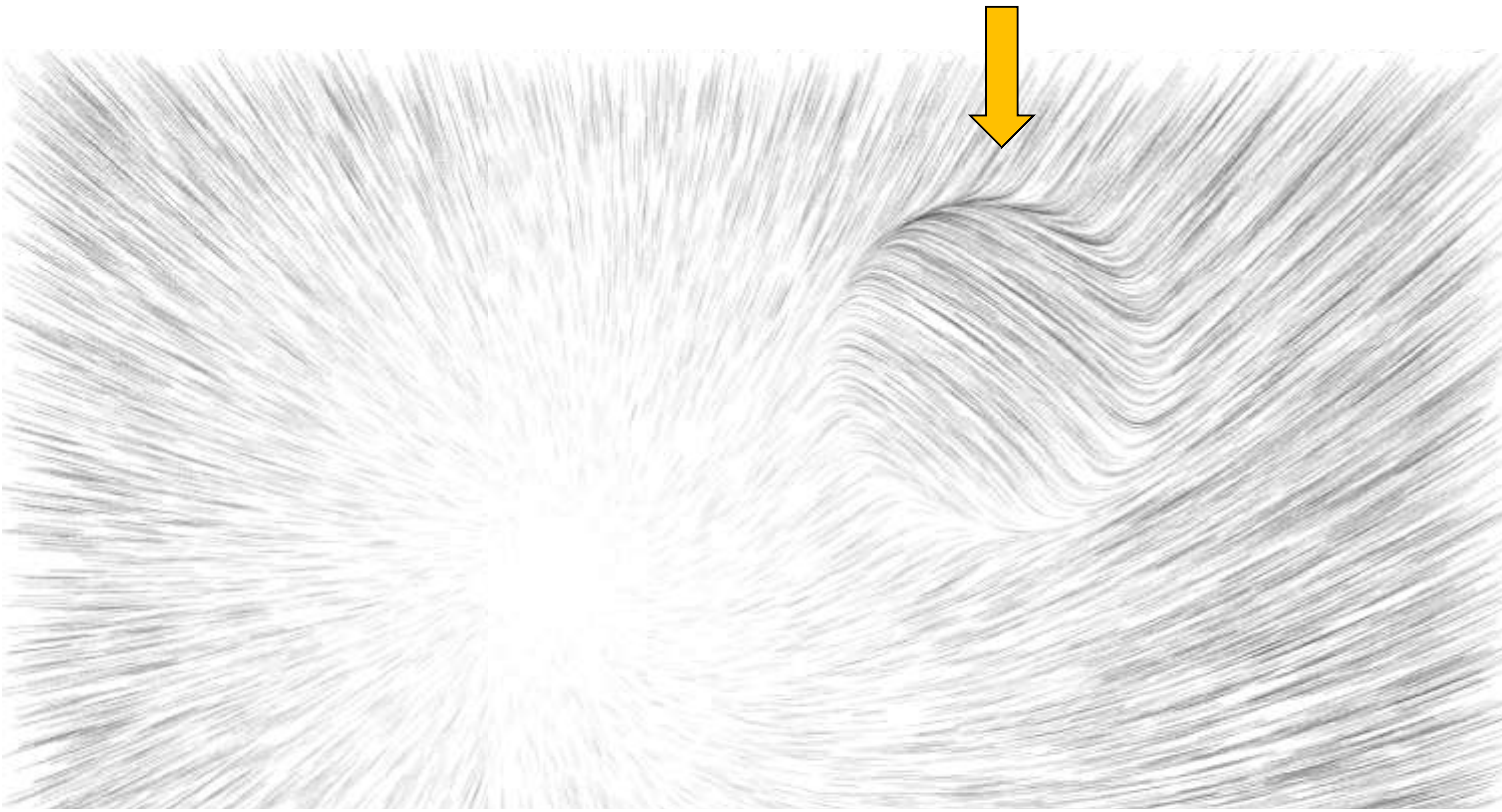
Start with an initial DS $\dot{x} = f(x)$ with attractor $\dot{x}^* = f(x^*) = 0.$



We will call this the *nominal* DS

DS after Local Modulation

Generate a ***local*** modulation



How can we modulate the DS while preserving the stability properties?

Modulation – Theoretical Approach

Idea:

Modulate the original DS in such a way that the new DS:

- remains a first order DS, ✓
- preserves asymptotic stability at its attractor x^*, ?
- has still a single attractor. ?

Which properties do we need for *M* to satisfy the other two requirements?

The modulated dynamics is given by:

$$\dot{x} = M(x) f(x) \,, \quad M(x) \in \mathbb{R}^{N \times N}$$

Does this satisfy our three desiderata?

The matrix M above should not be confused with the metric M of contraction theory introduced in Lecture 3.

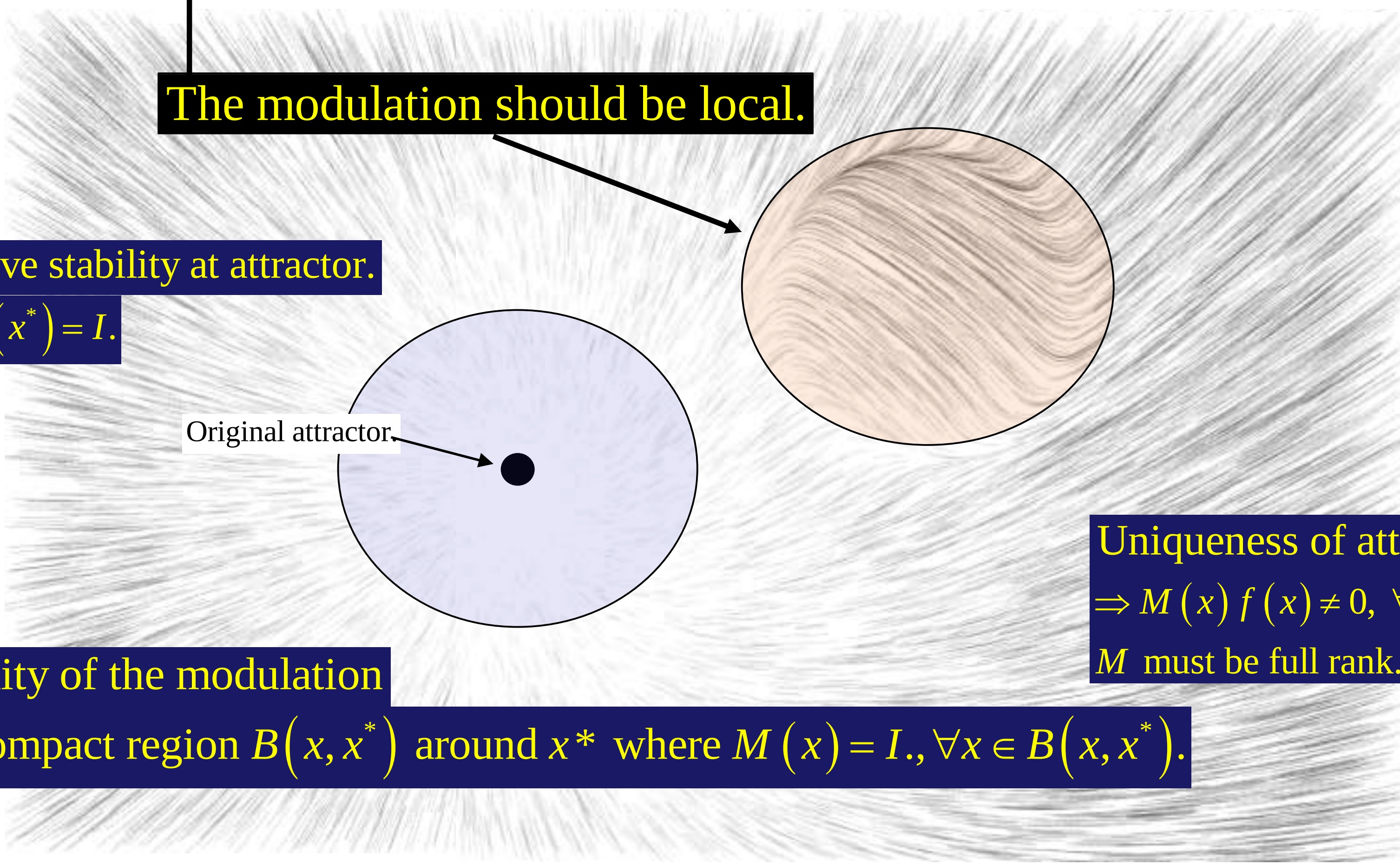
DS after Local Modulation

$$\dot{x} = M(x)\, f(x)$$

The modulation should be local.

Preserve stability at attractor.

$\Rightarrow M(x^*) = I.$



Uniqueness of attractor

$\Rightarrow M(x)\, f(x) \neq 0, \; \forall x \neq x^*$

M must be full rank.

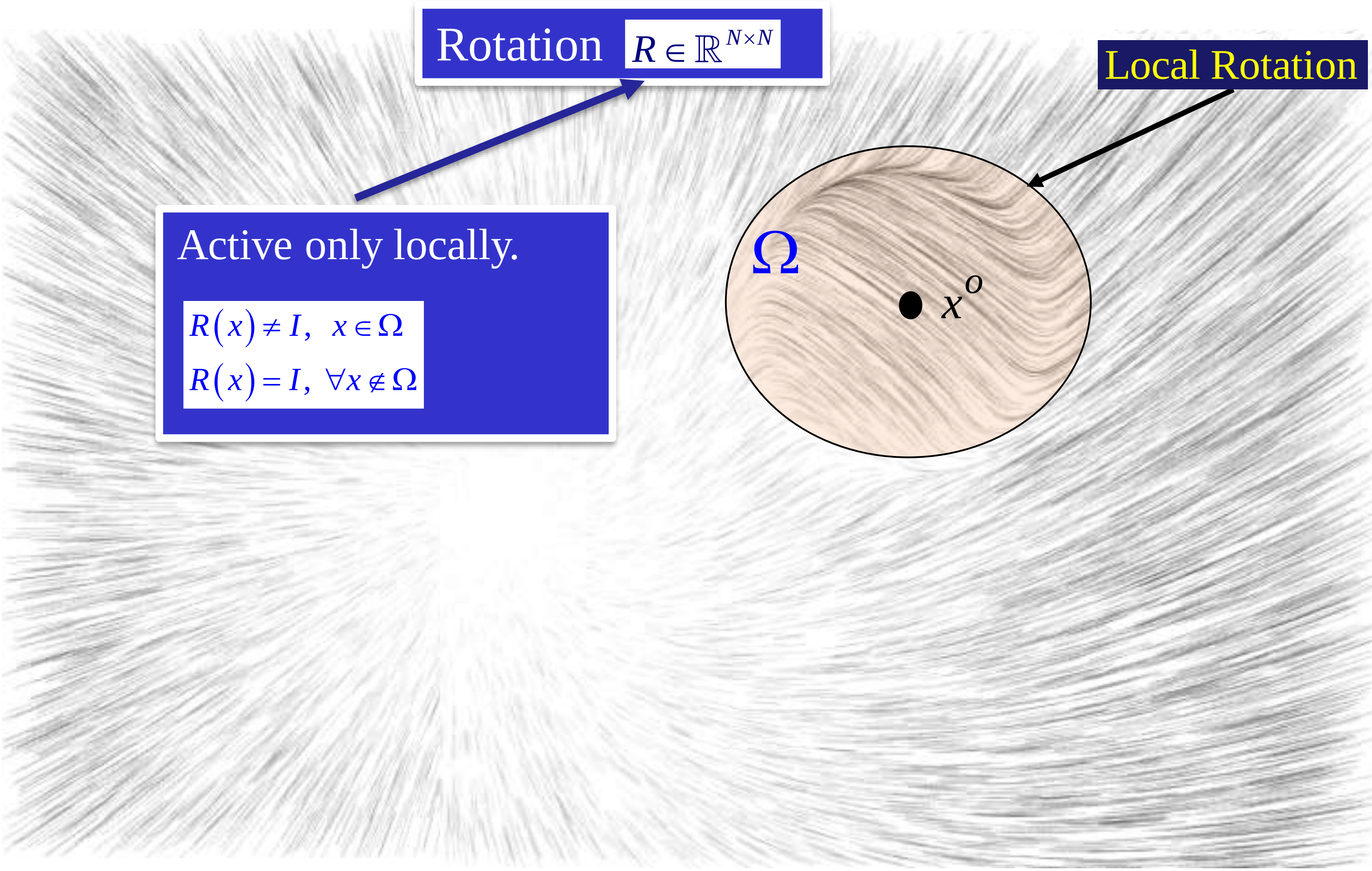
Locality of the modulation

$\exists$ a compact region $B(x, x^*)$ around x^* where $M(x) = I., \forall x \in B(x, x^*)$.

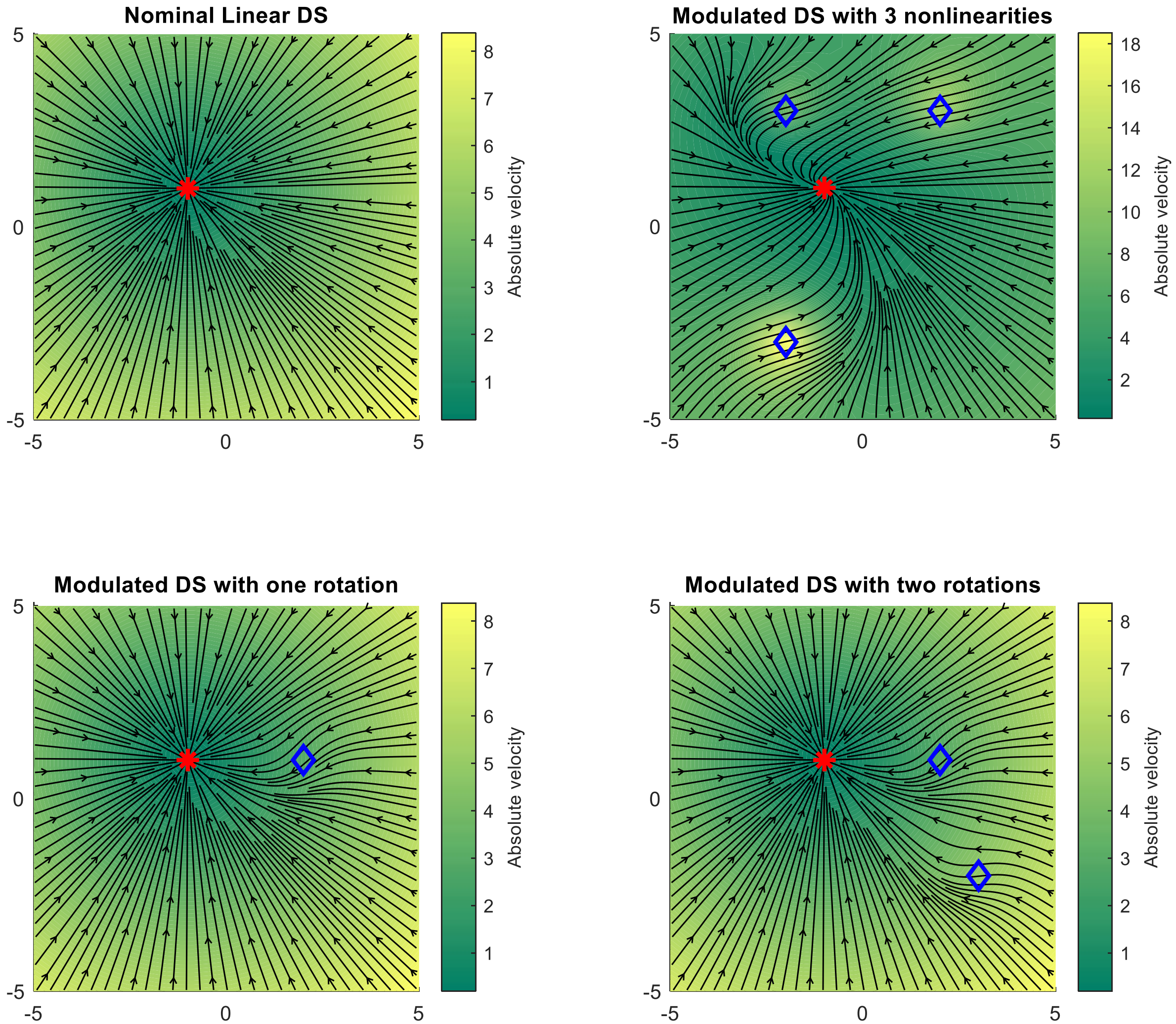
DS after Local Modulation

$$\dot{x} = M(x) f(x)$$

$$M(x) = R(\phi(x)), \quad \phi(x) = e^{-\|x-x^o\|}\phi_o.$$



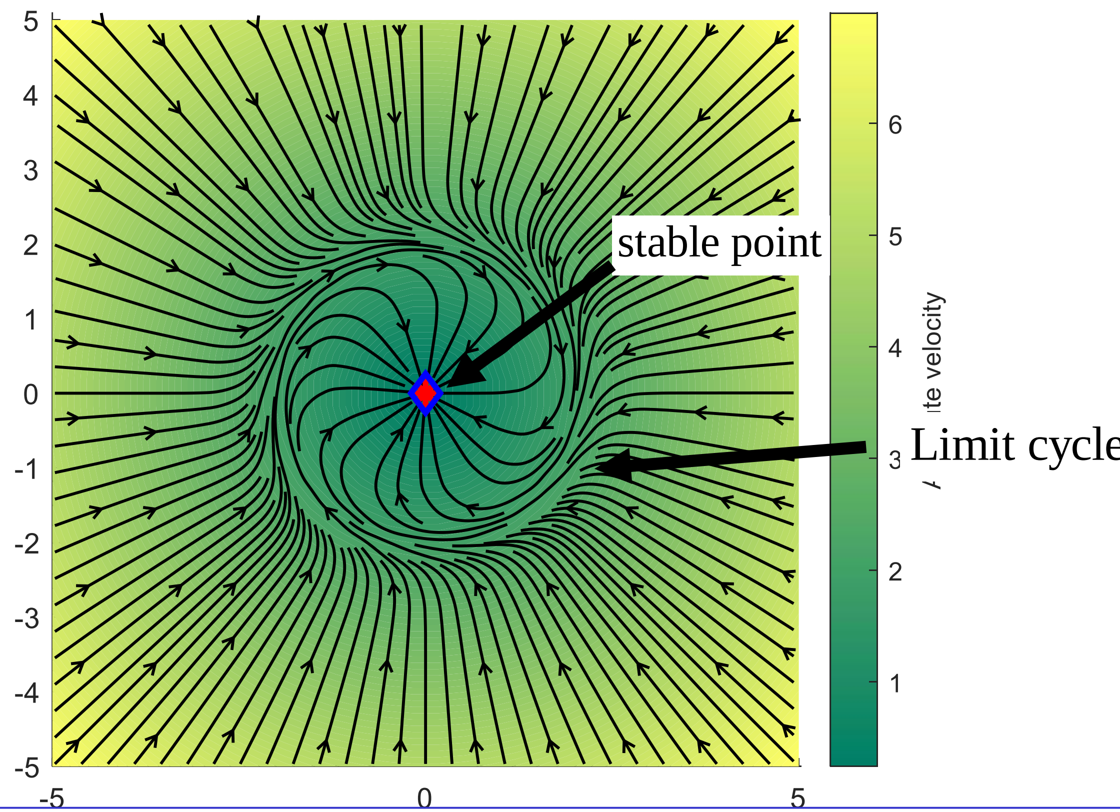
Modulation – Local Rotations - Examples



Modulation – Rotation Leading to a Limit Cycle

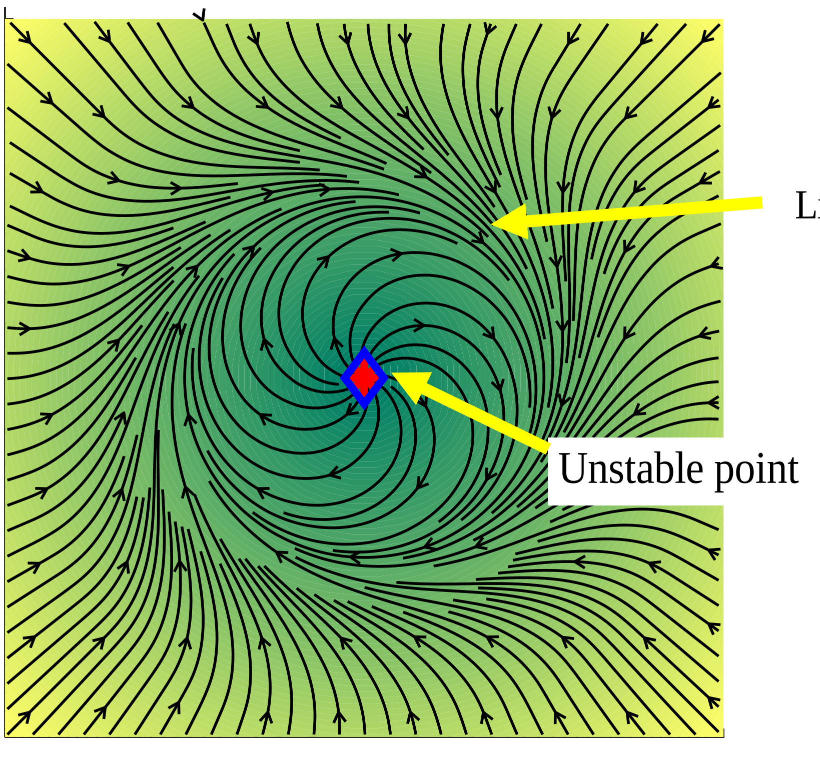
Modulations can be used to generate **new properties** for the DS

$$M=\begin{bmatrix}\cos(\gamma\theta) & -\sin(\gamma\theta)\\ \sin(\gamma\theta) & \cos(\gamma\theta)\end{bmatrix},\qquad \gamma(x,x^{o})=e^{-\frac{1}{\sigma^{2}}\left(\|x\|-r\right)^{2}},\quad \sigma>0.$$

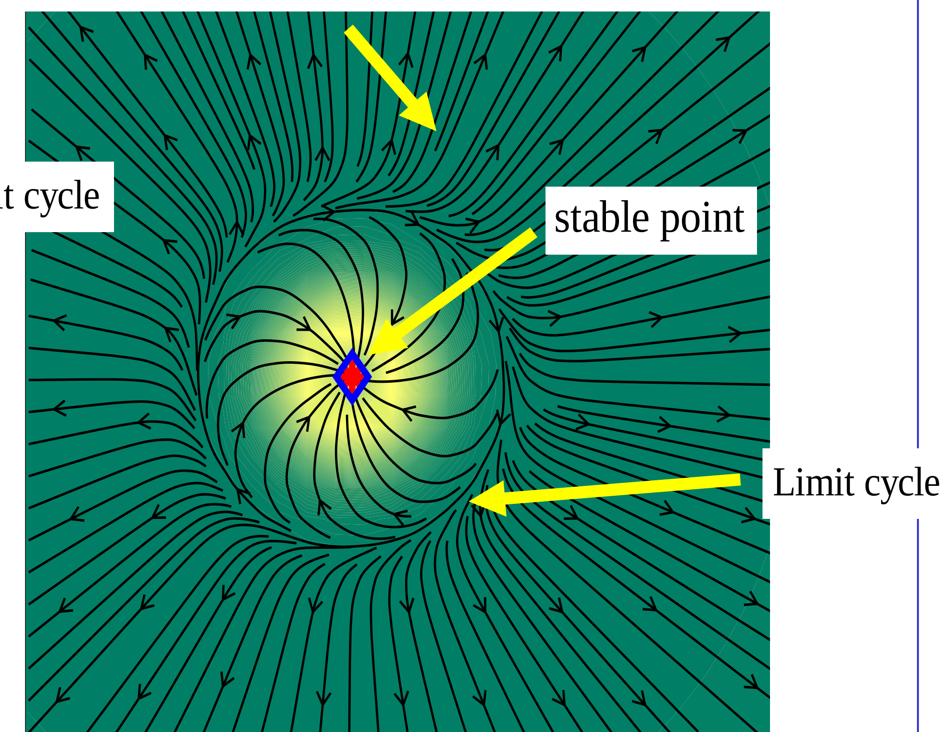


Diverging Systems

Invert flow and change stability of the nominal system (see Today's exercises).

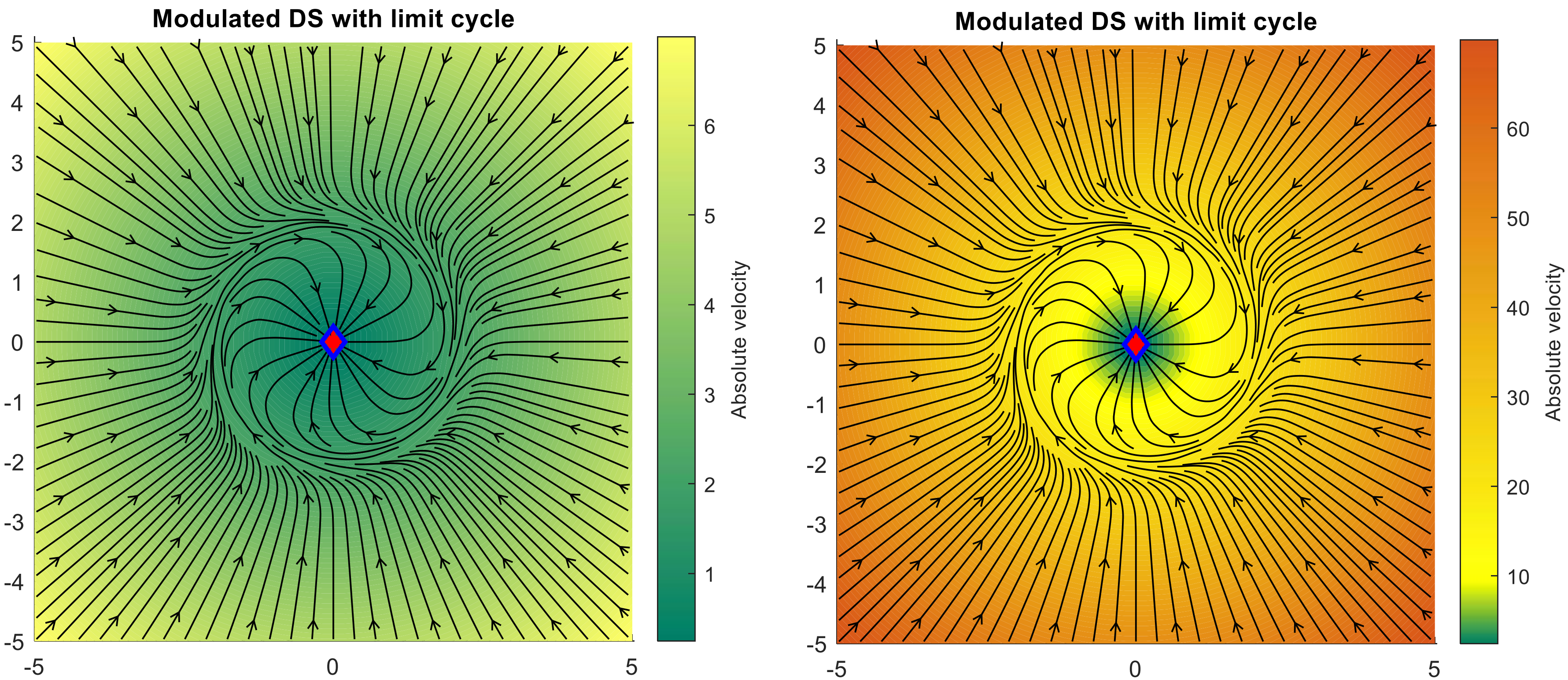


Flow diverging from attractor and limit cycle



Modulation - Scaling

Multiplicative term in front of the modulation will increase/decrease the speed of the DS but *will not modify the type of dynamics*. $\dot{x} = \lambda M(x) f(x), \quad \lambda \in \mathbb{R}.$



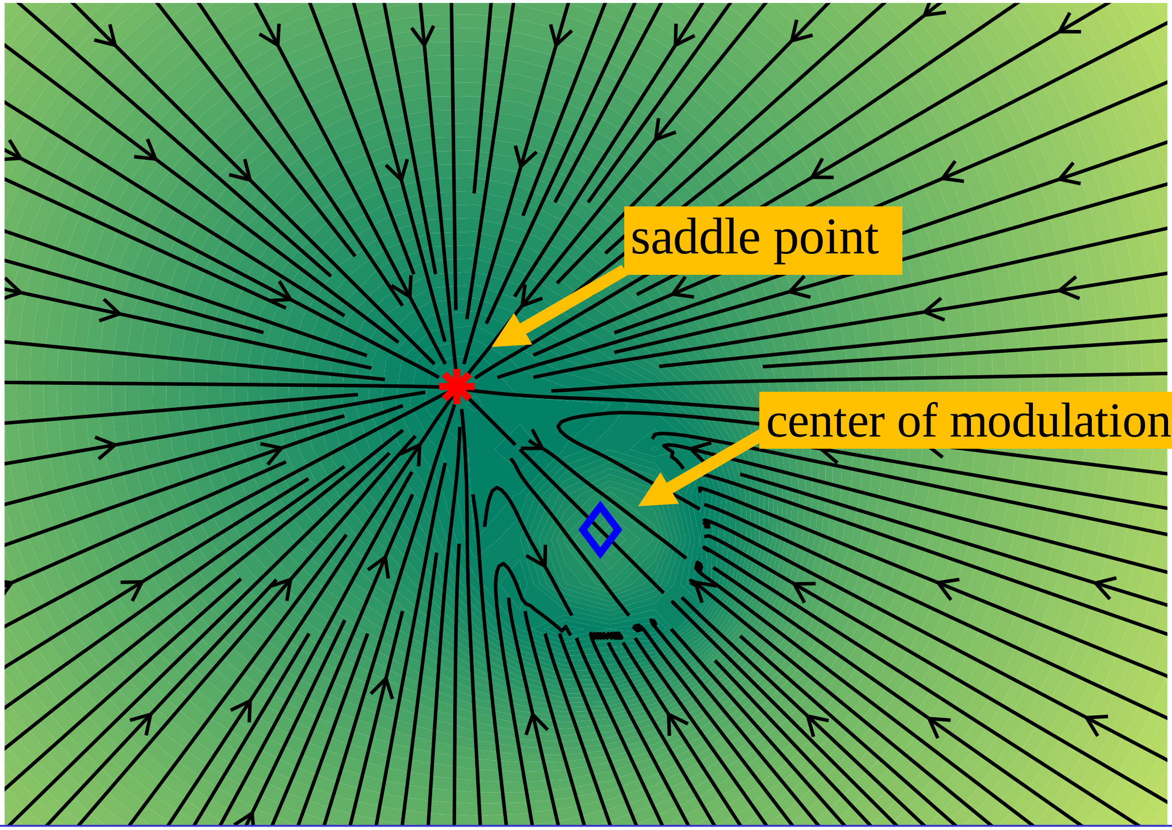
$\lambda = 1$ $\qquad$ $\lambda = 10$

Modulation – Other Properties

Modulations can be used to generate **new properties** for the DS

$$M = \begin{bmatrix} 1-2\gamma\left(x,x^o\right) & 0 \\ 0 & 1\text{-}2\gamma\left(x,x^o\right) \end{bmatrix}, \qquad \gamma\left(x,x^o\right) = e^{-\frac{1}{\sigma^2}\left\|x-x^o\right\|}, \quad \sigma > 0.$$

ERRATUM: book, section 8.1.1



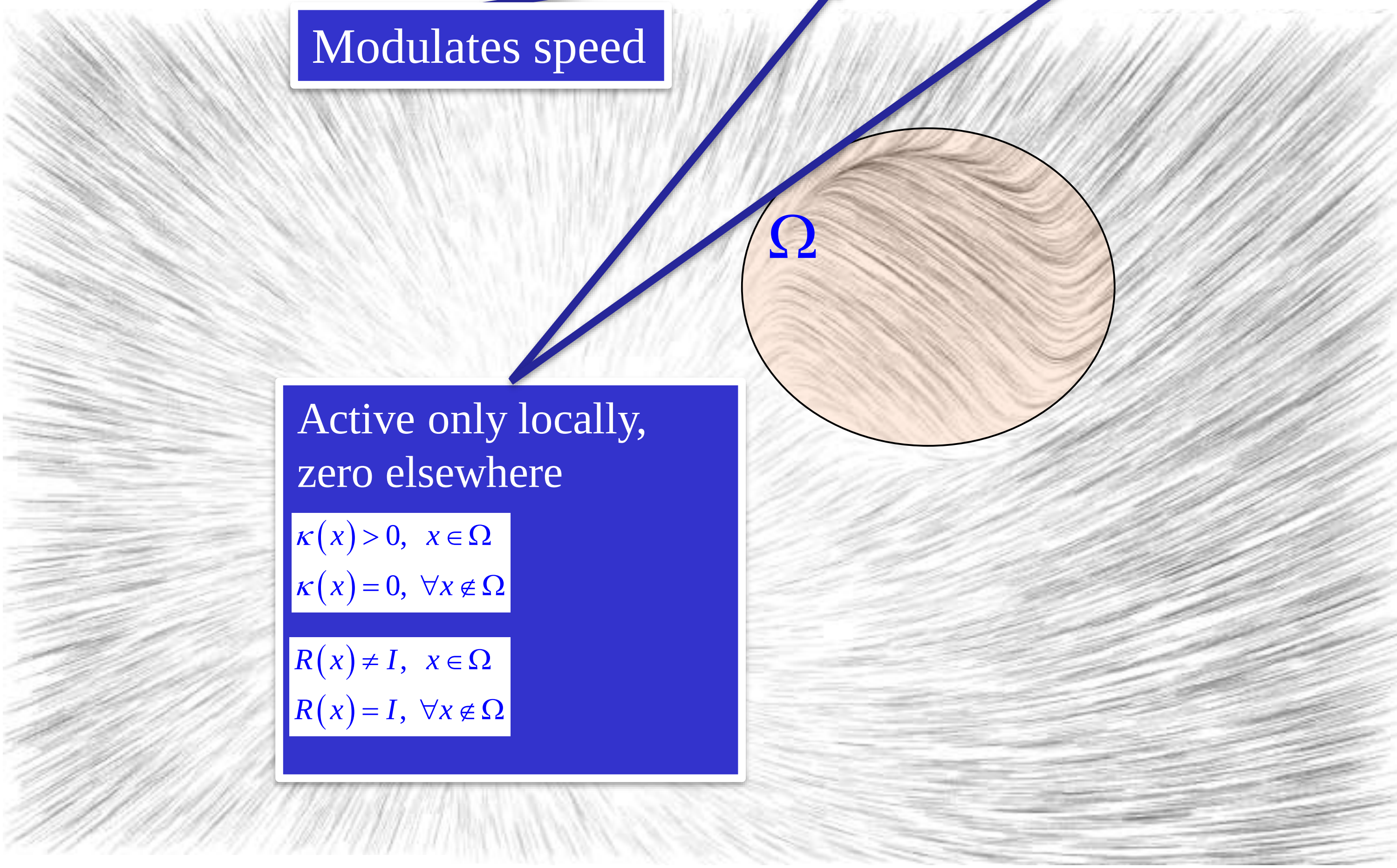
Parameterization of the modulation

$$\dot{x} = M(x)\, f(x)$$

$$M(x) = \left(1 + \kappa(x)\right) R(x)$$

Rotation $R \in \mathbb{R}^{N \times N}$

Modulates speed



Active only locally, zero elsewhere

$$\kappa(x) > 0, \quad x \in \Omega$$
$$\kappa(x) = 0, \quad \forall x \notin \Omega$$

$$R(x) \neq I, \quad x \in \Omega$$
$$R(x) = I, \quad \forall x \notin \Omega$$

Parameterization of the modulation

The modulation function can rotate and speed up the dynamics locally.

$$M(x) = (1+\kappa(x))R(x) \longleftarrow$$

Rotation

Direction / axes can be learned from data

Modulates the speed

Speed and region of activation can be learned from data.

In 2D, a rotation is defined solely by an angle ϕ, $R(\phi) = \begin{bmatrix} \cos(\phi) & -\sin(\phi) \\ \sin(\phi) & \cos(\phi) \end{bmatrix}$.

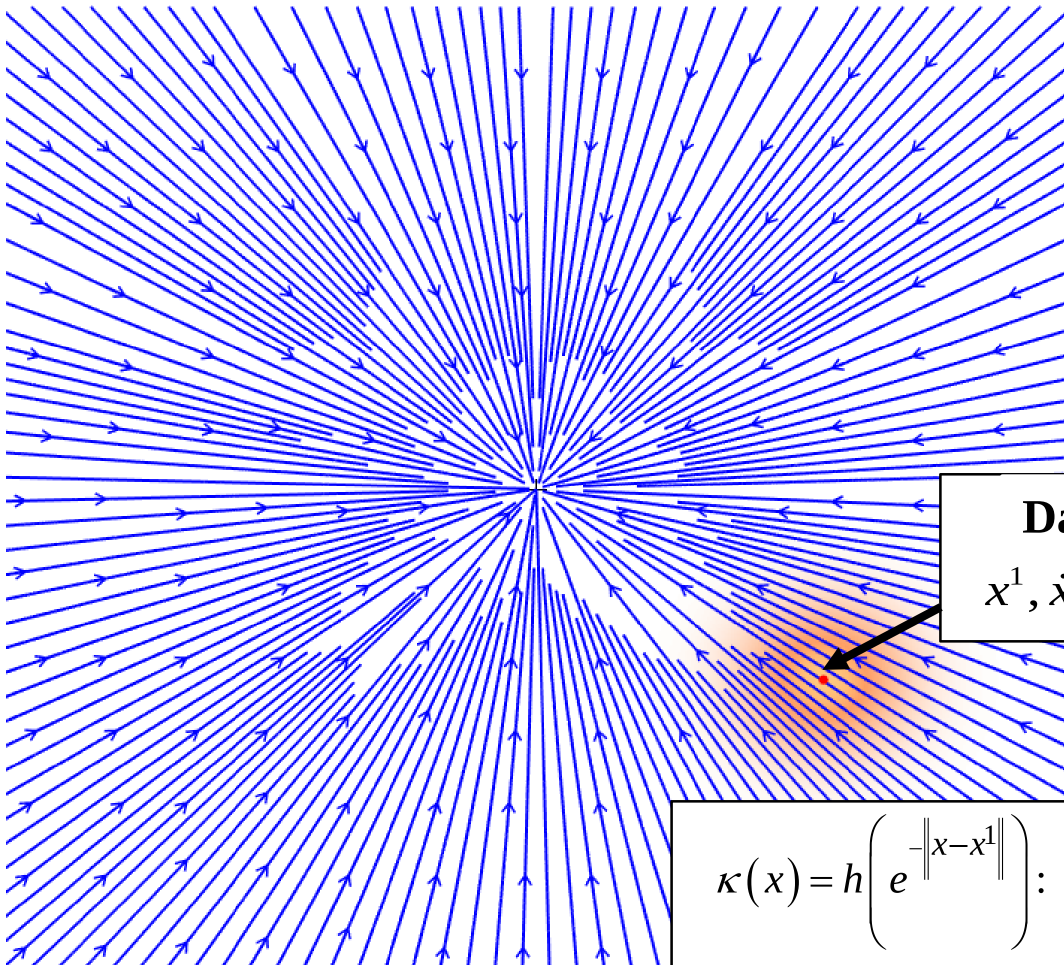
In 3D, a rotation $R(\phi, \mu_R)$ is defined by an angle ϕ around a vector μ_R.

The modulation function is hence parameterized as:

$$M\left(x; \Theta = \{\phi, \mu_R, \kappa\}\right) = \left(1+\kappa\left(x; X, \dot{X}\right)\right) R\left(\phi\left(x; X\right), \mu_R\left(x; X\right)\right), \quad X, \dot{X} : \text{data for learning}$$

Learning the modulation

Example – generated a rotation around a single demonstrated point



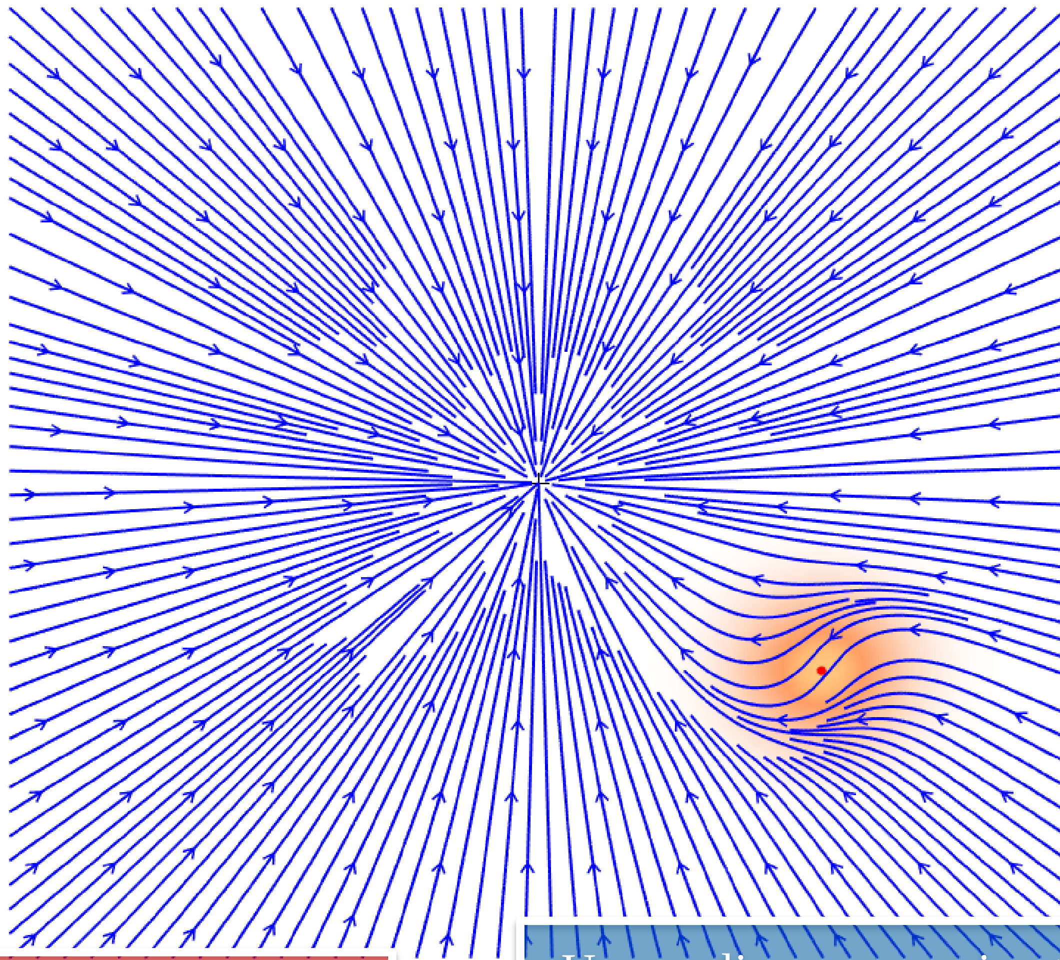
Data

$x^1, \dot{x}^1 = 0$

$\kappa(x) = h\left(e^{-\left\|x - x^1\right\|}\right)$: truncated exponential

$\phi_1 = \phi \sim random([0, \pi])$

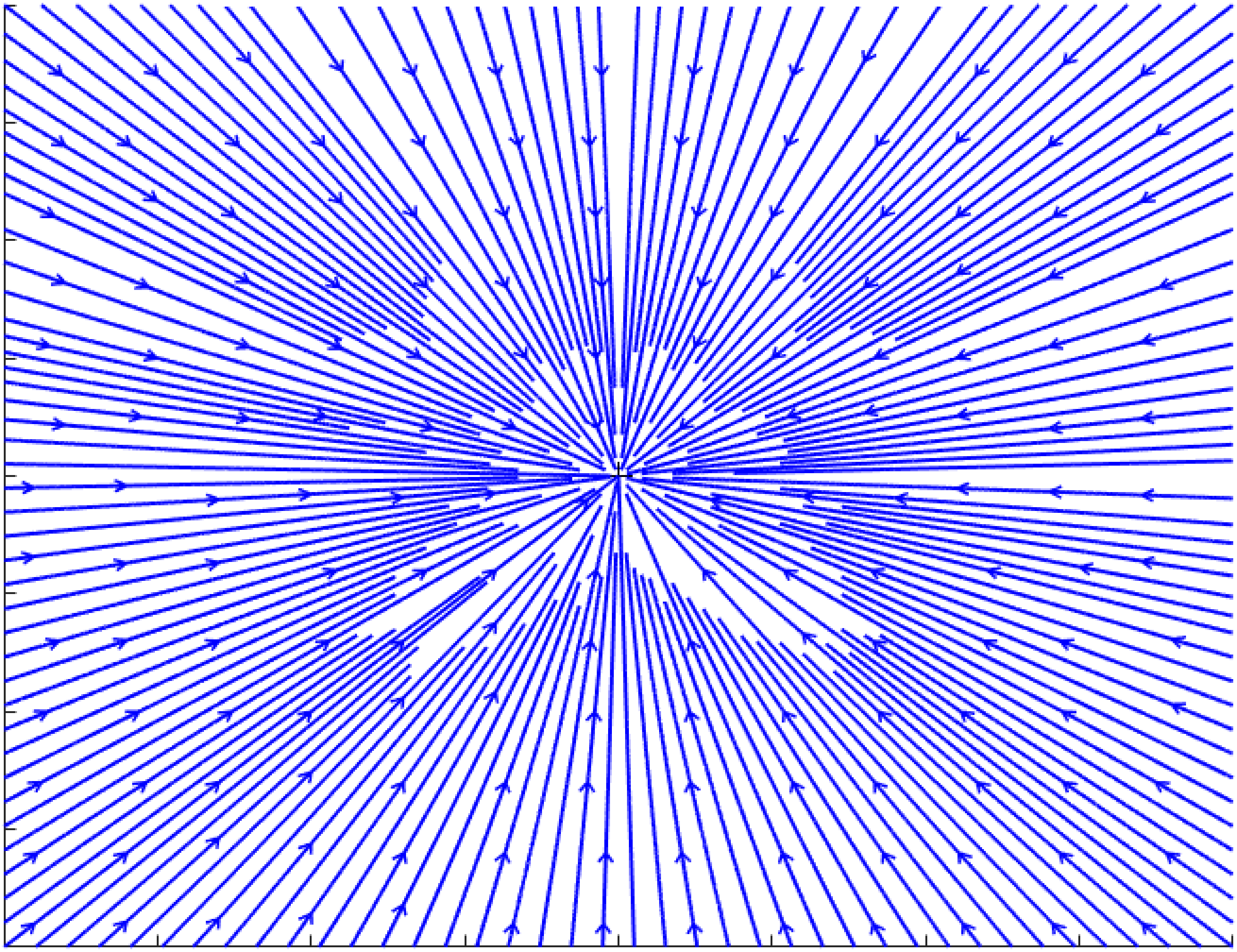
Example – learning a rotation around a single point



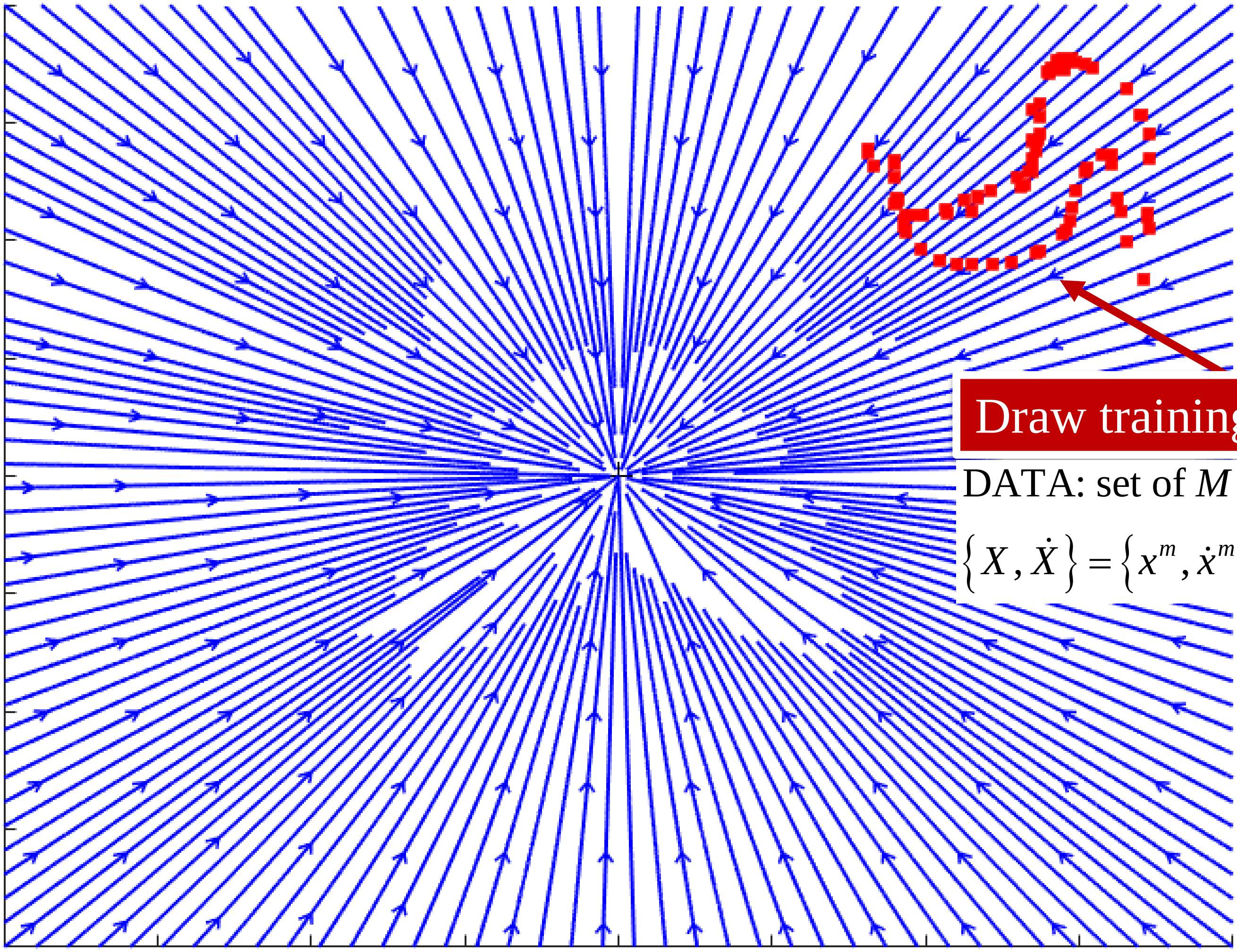
This can model only a single rotation

Use nonlinear regression techniques to model more complex modulations from full demonstrated trajectories.

Original linear dynamics: $\dot{x} = f\left(x\right)$



Original linear dynamics: $\dot{x} = f(x)$

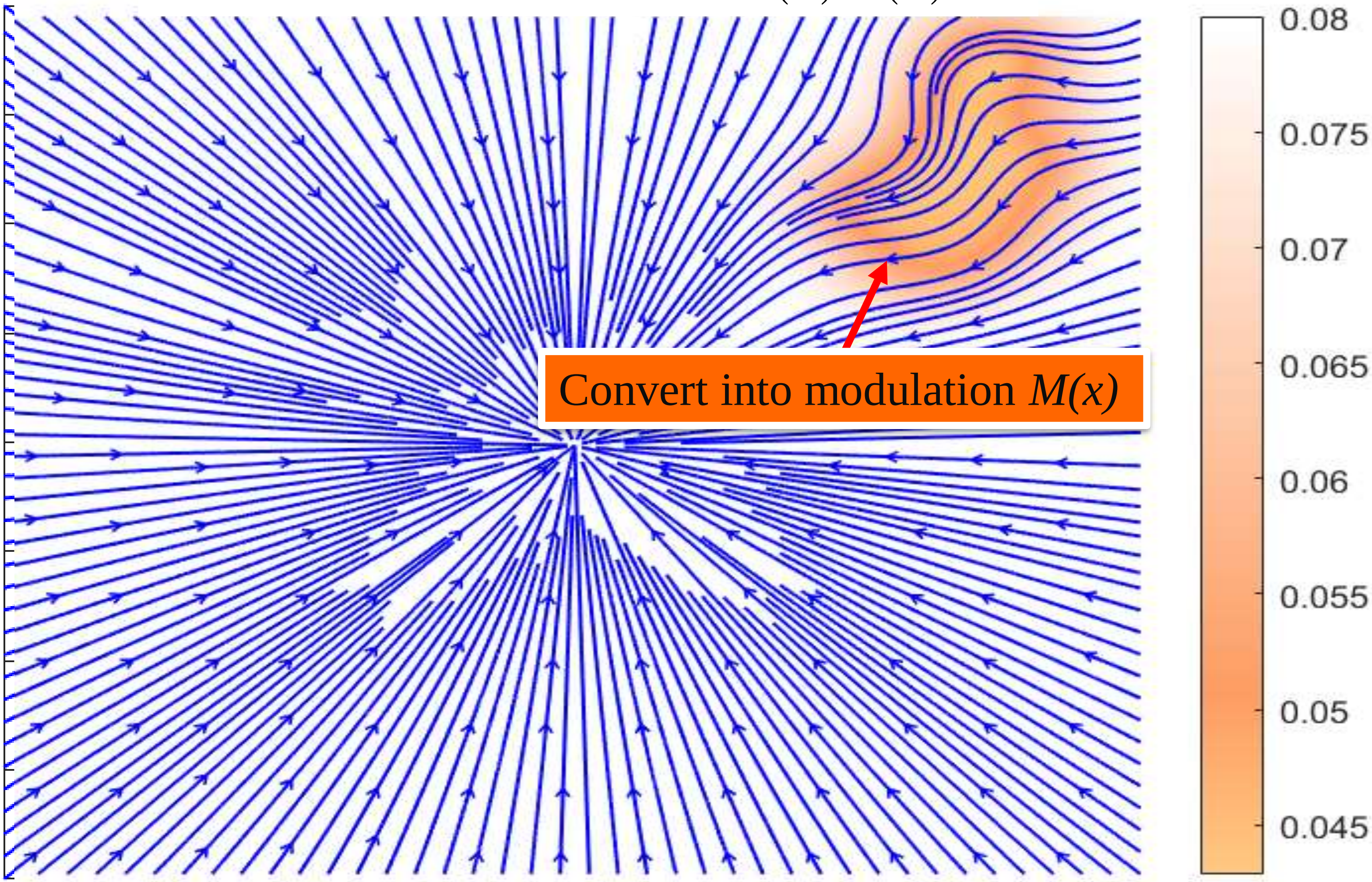


Draw training datapoints

DATA: set of M trajectories

$$\left\{X, \dot{X}\right\} = \left\{x^m, \dot{x}^m\right\}_{m=1}^{M}$$

Reshaped dynamics: $\dot{x} = M(x)\, f(x)$



Learning a Modulation (2D example)

General formulation

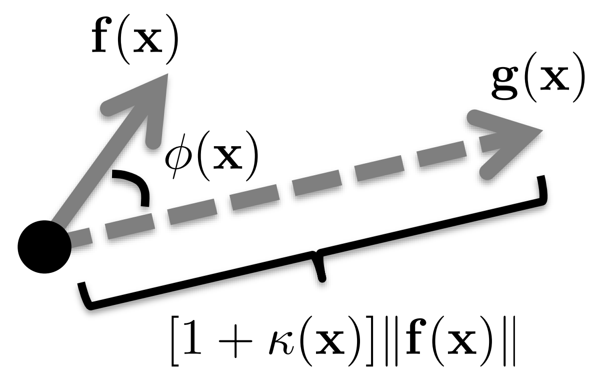
$$\dot{\mathbf{x}} = \mathbf{g}(\mathbf{x}) = \mathbf{M}(\mathbf{x})\mathbf{f}(\mathbf{x})$$

x	**Robot's state**
R(x)	**Rotation Matrix**
$[\kappa(x)]$	**Scaling factor**

Modulation composed of a scaling and rotation:

$$\mathbf{M}(\mathbf{x}) = [1 + \kappa(\mathbf{x})]\mathbf{R}(\mathbf{x})$$

$$\mathbf{R}(\mathbf{x}) = \begin{bmatrix} \cos(\phi(\mathbf{x})) & -\sin(\phi(\mathbf{x})) \\ \sin(\phi(\mathbf{x})) & \cos(\phi(\mathbf{x})) \end{bmatrix}$$



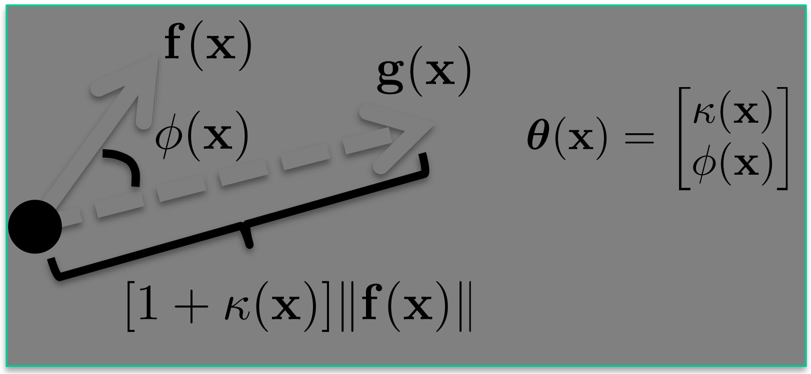
Define the parameter vector:

$$\boldsymbol{\theta}(\mathbf{x}) = \begin{bmatrix} \kappa(\mathbf{x}) \\ \phi(\mathbf{x}) \end{bmatrix}$$

For dimension >2, the parameter vector is expanded by parameters describing the rotation set.

From Trajectory Data to Reshaping Parameters

x	Robot's state
$\mathbf{R(x)}$	Rotation Matrix
$[\kappa(x)]$	Scaling factor
x_m	Orignal data-points
x_m^o	New data-points



Trajectory data

$$\left\{x^m, \dot{x}^m\right\}_{m=1}^{M}$$

Nominal vel.

$${}^{o}\dot{x}^m = f\left(x^m\right)$$

Nominal pos.

$$\left\{{}^{o}x^m\right\}_{m=1}^{M}$$

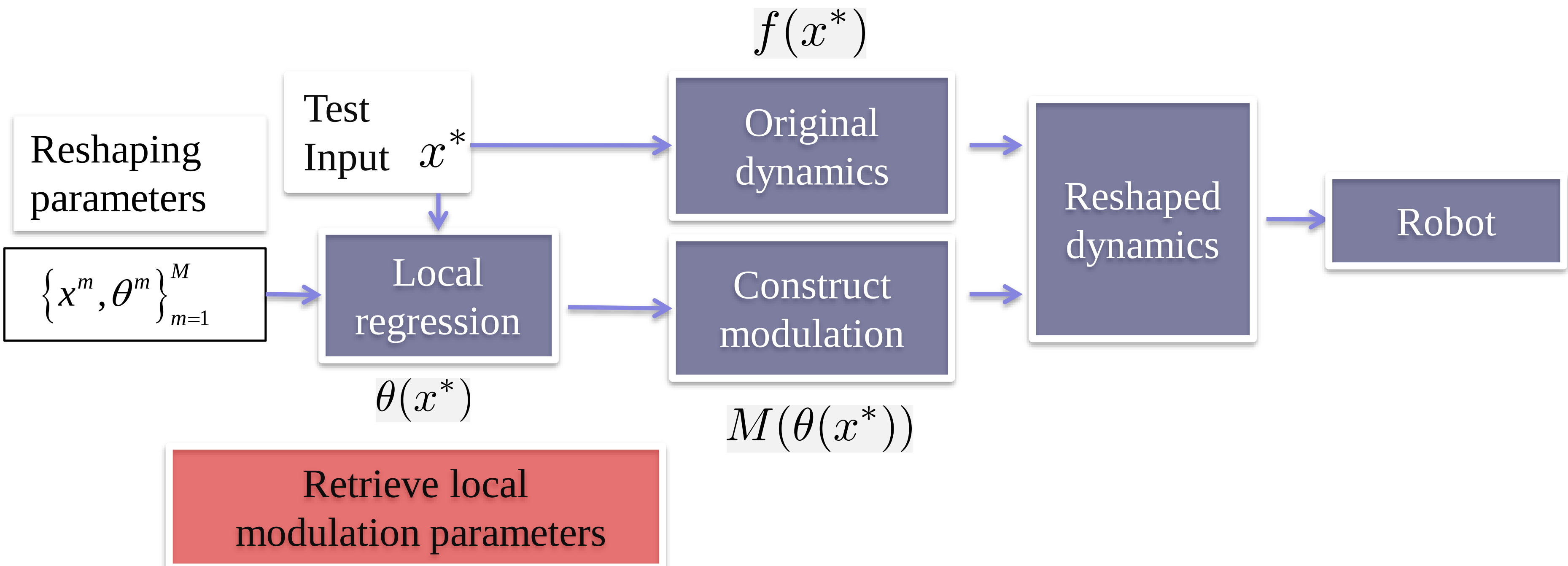
$$\kappa_m = \frac{\left\|\dot{x}^m\right\|}{\left\|{}^{o}\dot{x}^m\right\|} - 1$$

$$\phi_m = \frac{\operatorname{acos}\left(\left\langle \dot{x}^m, {}^{o}\dot{x}^m \right\rangle\right)}{\left\|\dot{x}^m\right\|\left\|{}^{o}\dot{x}^m\right\|}$$

Reshaping parameters

$$\left\{x^m, \theta^m\right\}_{m=1}^{M}$$

Learning and Using the Reshaped Dynamics



Pseudo-Code for Converting Data into Modulation Parameters

Algorithm 8.1 Procedure for converting 2D or 3D trajectory data to modulation data.

Require: Trajectory data $\{x^m, \dot{x}^m\}_{m=1}^M$

1: **for** $m = 1 \rightarrow M$ **do**

2: Compute nominal velocity: ${}^o\dot{x}^m = f(x^m)$

3: Compute rotation vector (3D only): $\mu^m = \frac{\dot{x}^m \times {}^o\dot{x}^m}{\|\dot{x}^m\| \|{}^o\dot{x}^m\|}$

4: Compute rotation angle: $\phi^m = \arccos \frac{\dot{x}^{m^T}\, {}^o\dot{x}^m}{\|\dot{x}^m\| \|{}^o\dot{x}^m\|}$

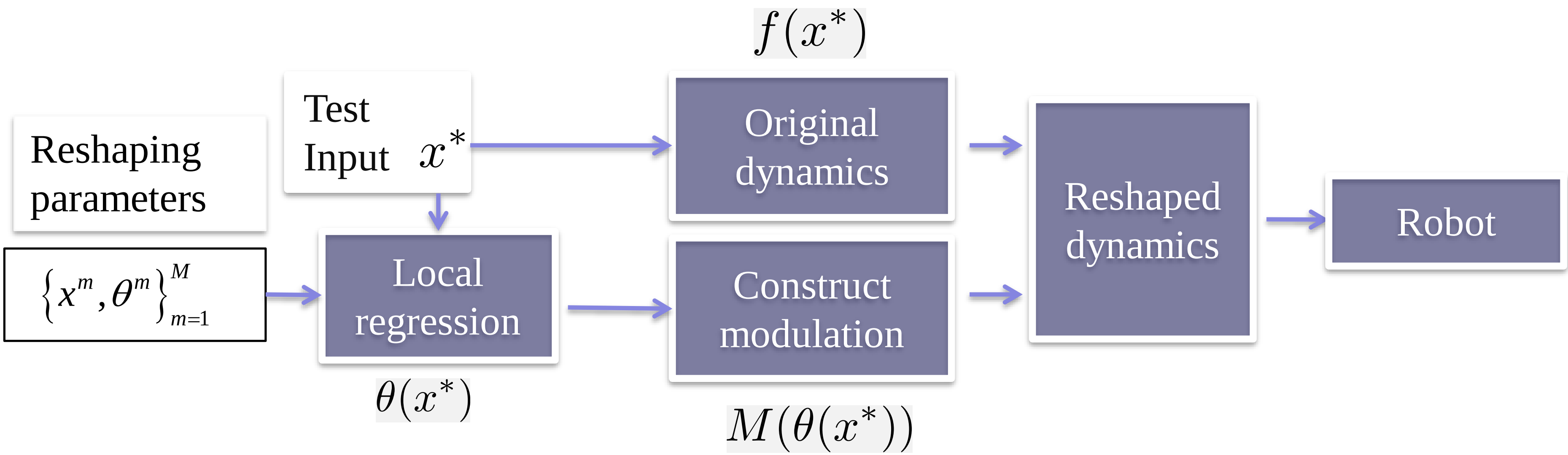
5: Compute scaling: $\kappa_m = \frac{\|\dot{x}^m\|}{\|{}^o\dot{x}^m\|} - 1$

6: $3\mathrm{D} : \theta^m = [\phi^m \mu^m, \kappa^m], \quad 2\mathrm{D} : \theta^m = [\phi^m, \kappa^m]$

7: **end for**

8: **return** Modulation data $\{x^m, \theta^m\}_{m=1}^M$

Learning and Using the Reshaped Dynamics



Use Gaussian Process Regression (GPR)

$$M\left(x; \Theta = \{\phi, \mu_R, \kappa\}\right) = \left(1 + \kappa\left(x; X, \dot{X}\right)\right) R\left(\phi\left(x; X\right), \mu_R\left(x; X\right)\right)$$

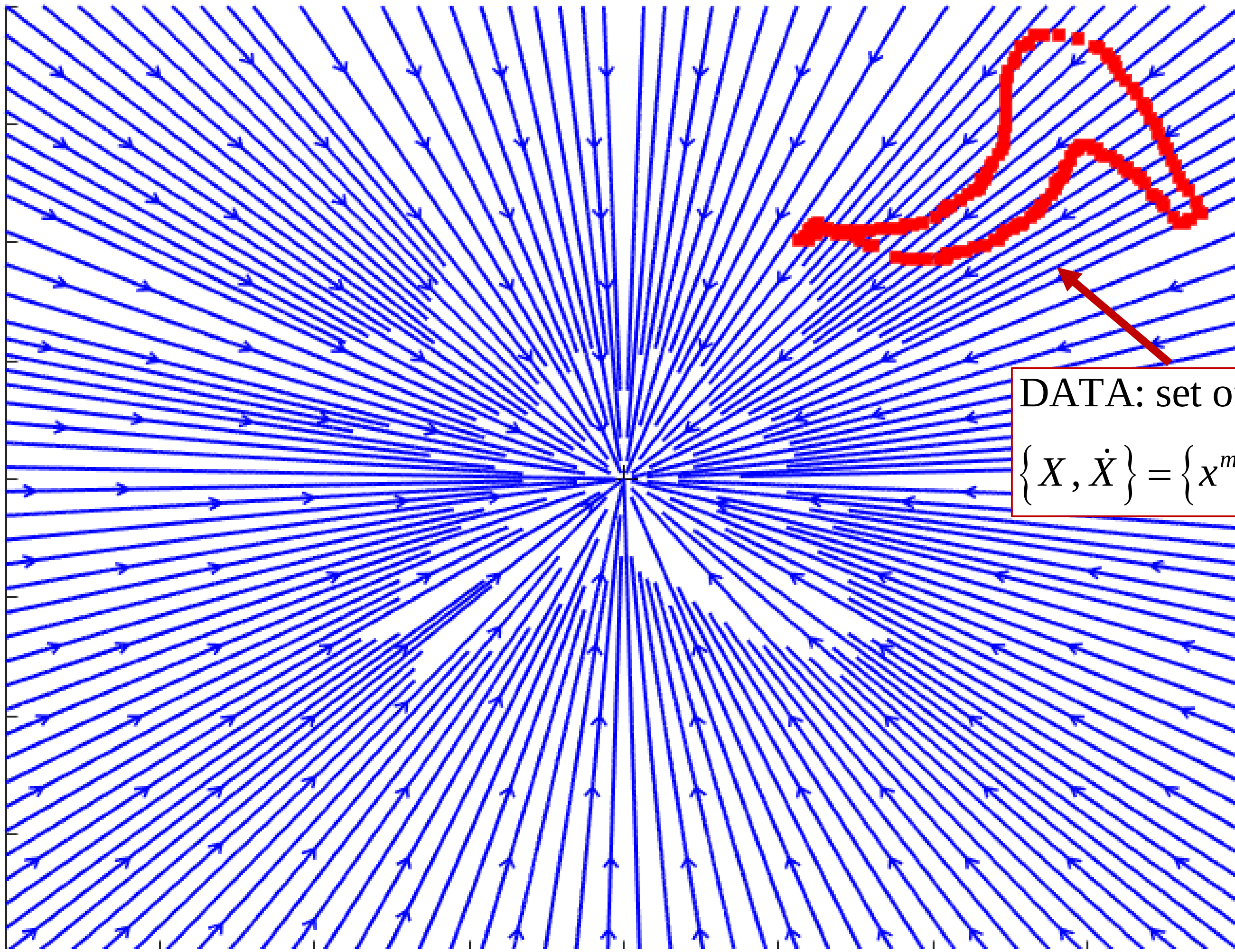
Estimate the vector through more GPRs

Estimate the speed scaling through 1 GPR
Scalar function changing in space.

1 GPR function to estimate state-dependent angle

Gaussian Process Regression (GPR) for Learning the Modulation

Original linear dynamics: $\dot{x} = f\left(x\right)$



DATA: set of M datapoints

$\left\{X, \dot{X}\right\} = \left\{x^m, \dot{x}^m\right\}_{m=1}^{M}$

Gaussian Process Regression (GPR) for Learning the Modulation

GPR model:

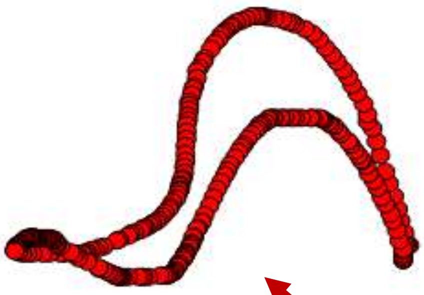
$$\dot{x} = \sum_{i=1}^{M} \alpha_i k\left(x, x^i\right)$$

Kernel:

$$k\left(x, x'\right) = \sigma_f e^{-\frac{\|x-x'\|}{2l}}$$

$$\text{with } \alpha = \left[K\left(X, X\right) + \sigma_n^{\ 2} I\right]^{-1} \begin{bmatrix} \dot{x}^1 \\ \ldots \\ \dot{x}^M \end{bmatrix}$$

$$\text{Gram matrix: } K\left(X, X\right) = \begin{bmatrix} k\left(x^1, x^1\right) & k\left(x^1, x^2\right) & \ldots\ldots\ldots k\left(x^1, x^M\right) \\ . & & \\ . & & \\ k\left(x^M, x^1\right) & k\left(x^M, x^2\right) & \ldots\ldots \end{bmatrix}$$



DATA: set of M datapoints

$$\left\{X, \dot{X}\right\} = \left\{x^m, \dot{x}^m\right\}_{m=1}^{M}$$

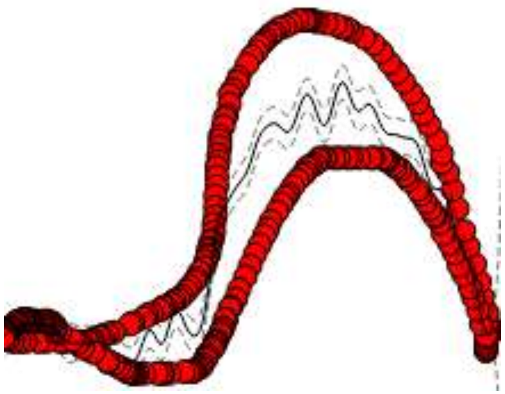
Hyperparameters

$l > 0$: lengthscale parameter ~ kernel width - controls tightness of the fit.

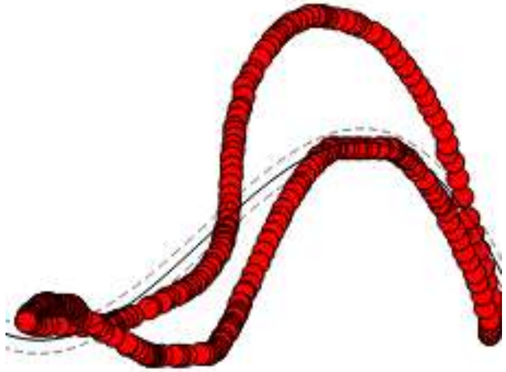
$\sigma_f > 0$: scales the kernel to the range of values of the output.

$\sigma_n > 0$: noise parameter - controls tolerance on fit imprecision.

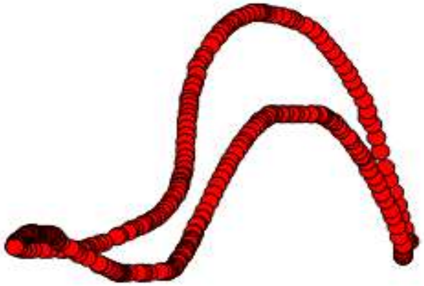
Lengthscale parameter

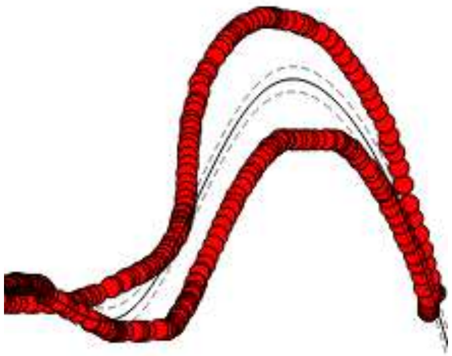


Too small



Too large





Good fit

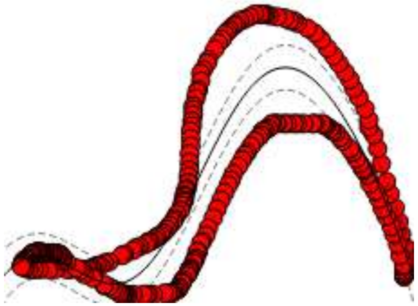
Kernel:

$$k\left(x,x'\right)=\sigma_f e^{-\frac{\|x-x'\|}{2l}}$$

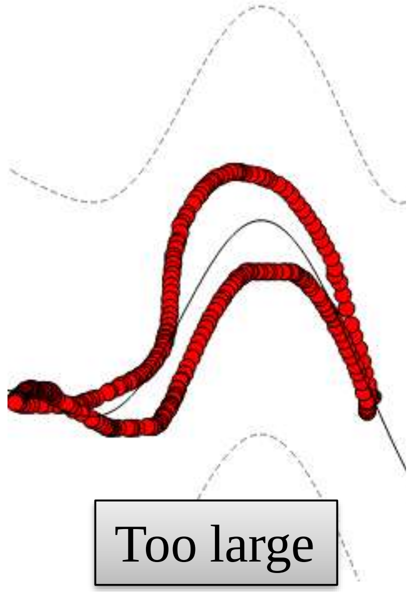
$l>0$: controls tightness of the fit.

Careful: this is an illustration, as to learn the modulation we regress in velocity and not on x_2.

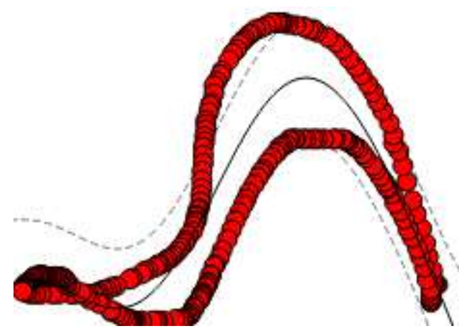
Noise parameter



Too small



Too large

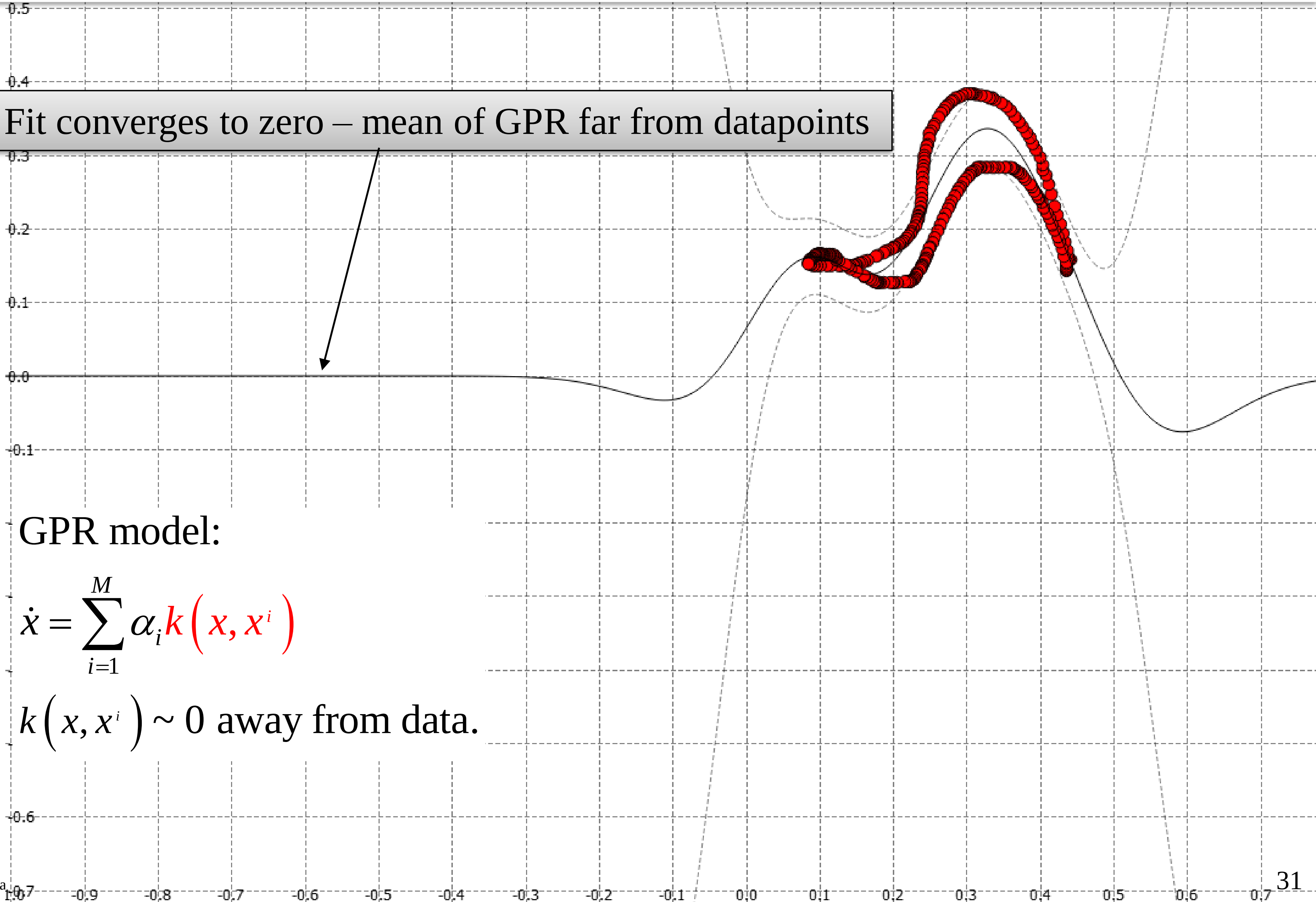


Good fit

$$\dot{x} = \sum_{i=1}^{M} \alpha_i k\left(x, x^i\right), \text{with } \alpha = \left[K\left(X, X\right) + \sigma_n{}^2 I\right]^{-1} \begin{bmatrix} \dot{x}^1 \\ \dots \\ \dot{x}^M \end{bmatrix}$$

$\sigma_n > 0$: noise parameter - controls tolerance on fit imprecision

Convergence away from data



Fit converges to zero – mean of GPR far from datapoints

GPR model:

$$\dot{x} = \sum_{i=1}^{M} \alpha_i k\left(x, x^i\right)$$

$k\left(x, x^i\right) \sim 0$ away from data.

Convergence away from data

Modulated dynamics:

$$\dot{x} = \left(1+\kappa\left(x\right)\right) R\left(\phi\left(x\right)\right) f\left(x\right)$$

$\rightarrow 0$

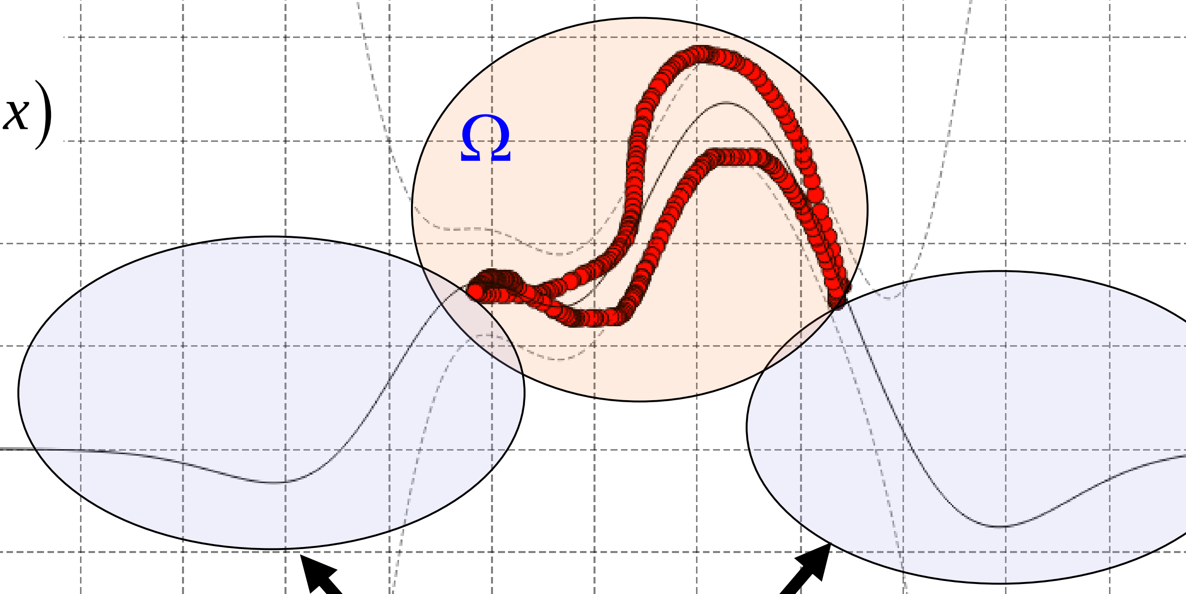
$\rightarrow 0$

$\rightarrow I$

Active only locally, zero elsewhere

$\kappa(x) > 0, \quad x \in \Omega$
$\kappa(x) = 0, \quad \forall x \notin \Omega$

$R(x) \neq I, \quad x \in \Omega$
$R(x) = I, \quad \forall x \notin \Omega$



Hard to determine where the GPR function vanishes exactly.

Forces stability away from modulation

GPR model:

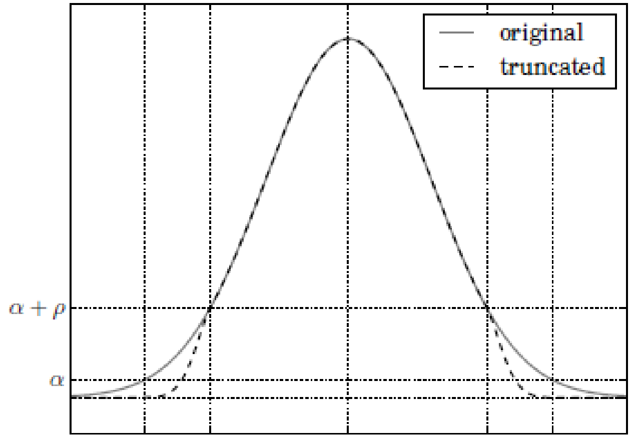
$$\dot{x} = \sum_{i=1}^{M} \alpha_i k\left(x^*, x^i\right), \qquad x^* : \text{query point}$$

$$\text{with } \alpha = \left[K\left(X, X\right) + \sigma_n{}^2 I\right]^{-1} \begin{bmatrix} \dot{x}^1 \\ \dots \\ \dot{x}^M \end{bmatrix}$$

Rewrite as a function of query point

$$\dot{x} = \underbrace{\left[K\left(X, X\right) + \sigma_n{}^2 I\right]^{-1}}_{\alpha\left(x^*\right)} K_{Xx^*}$$

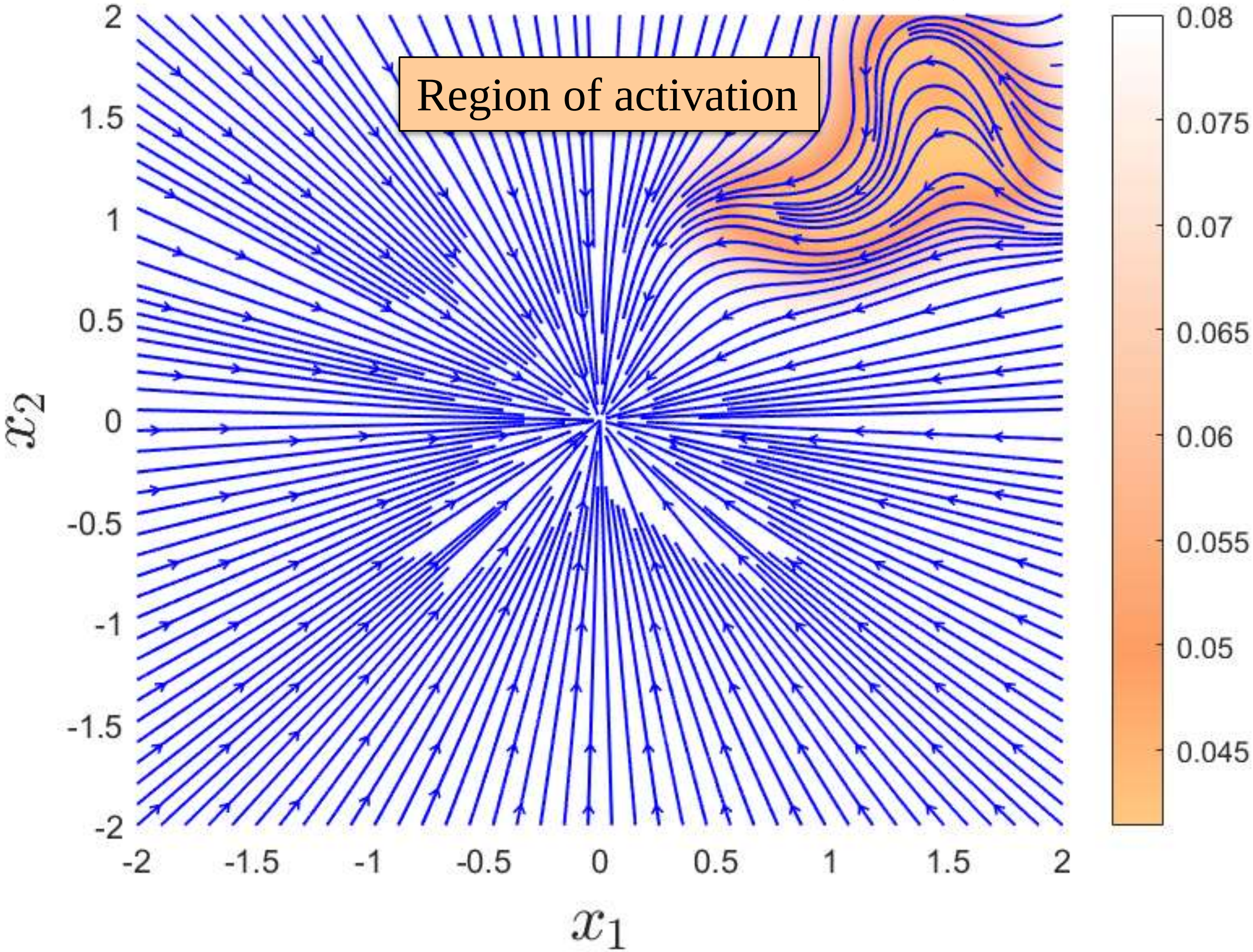
Truncate the GPR function at fixed bounds.



$$\boxed{\alpha'\left(x^*\right) = \begin{cases} 0 & \alpha\left(x^*\right) < \underline{\alpha} \\ \frac{1}{2}\left(1 + \sin\left(\frac{2\pi\left(\alpha\left(x^*\right) - \underline{\alpha}\right)}{2\rho} - \frac{\pi}{2}\right)\right)\alpha\left(x^*\right) & \underline{\alpha} \le \alpha\left(x^*\right) \le \underline{\alpha} + \rho. \\ \alpha\left(x^*\right) & \underline{\alpha} + \rho < \alpha\left(x^*\right) \end{cases}}$$

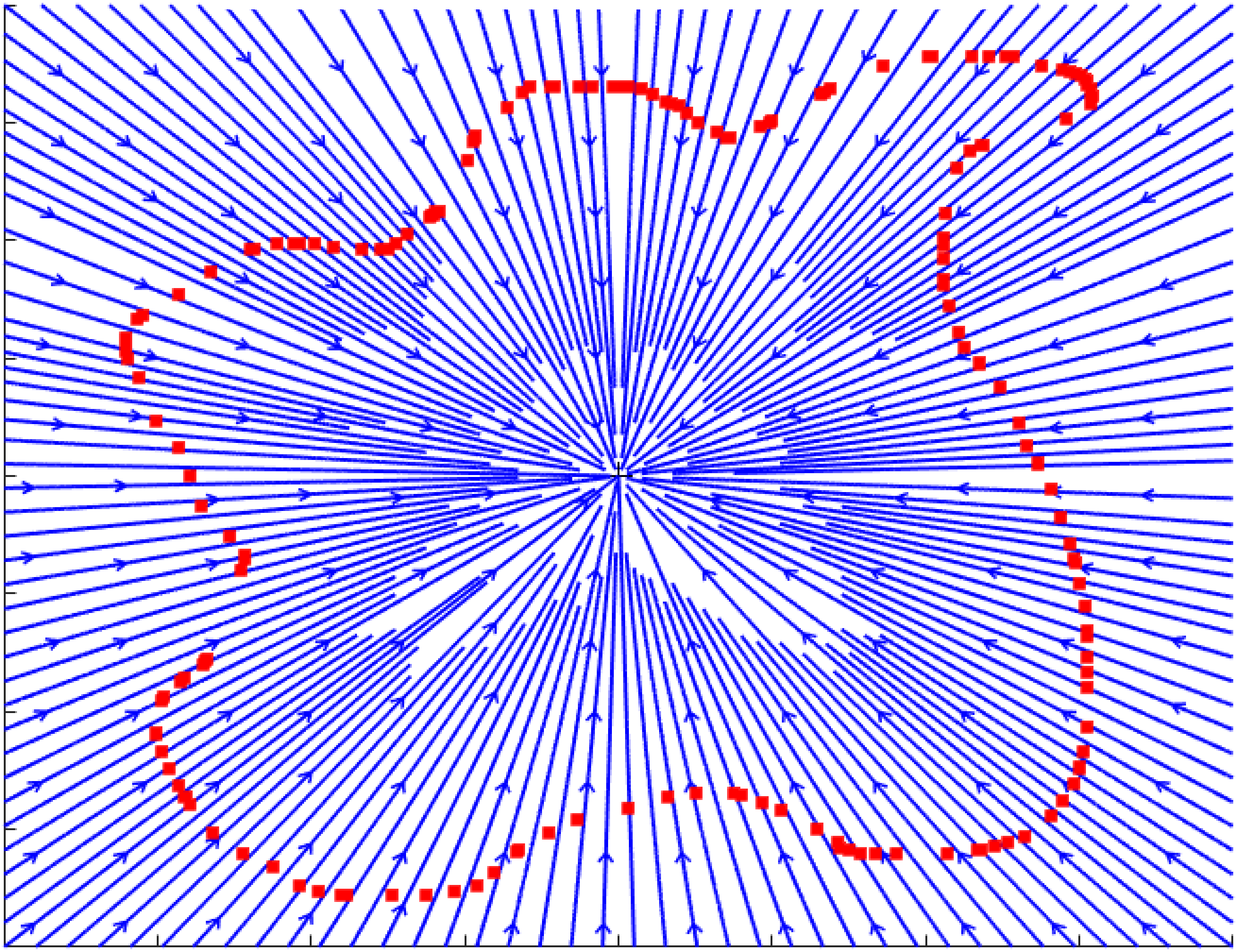
Example: Modulation Learned by GPR

Reshaped dynamics: $\dot{x} = M(x) f(x)$



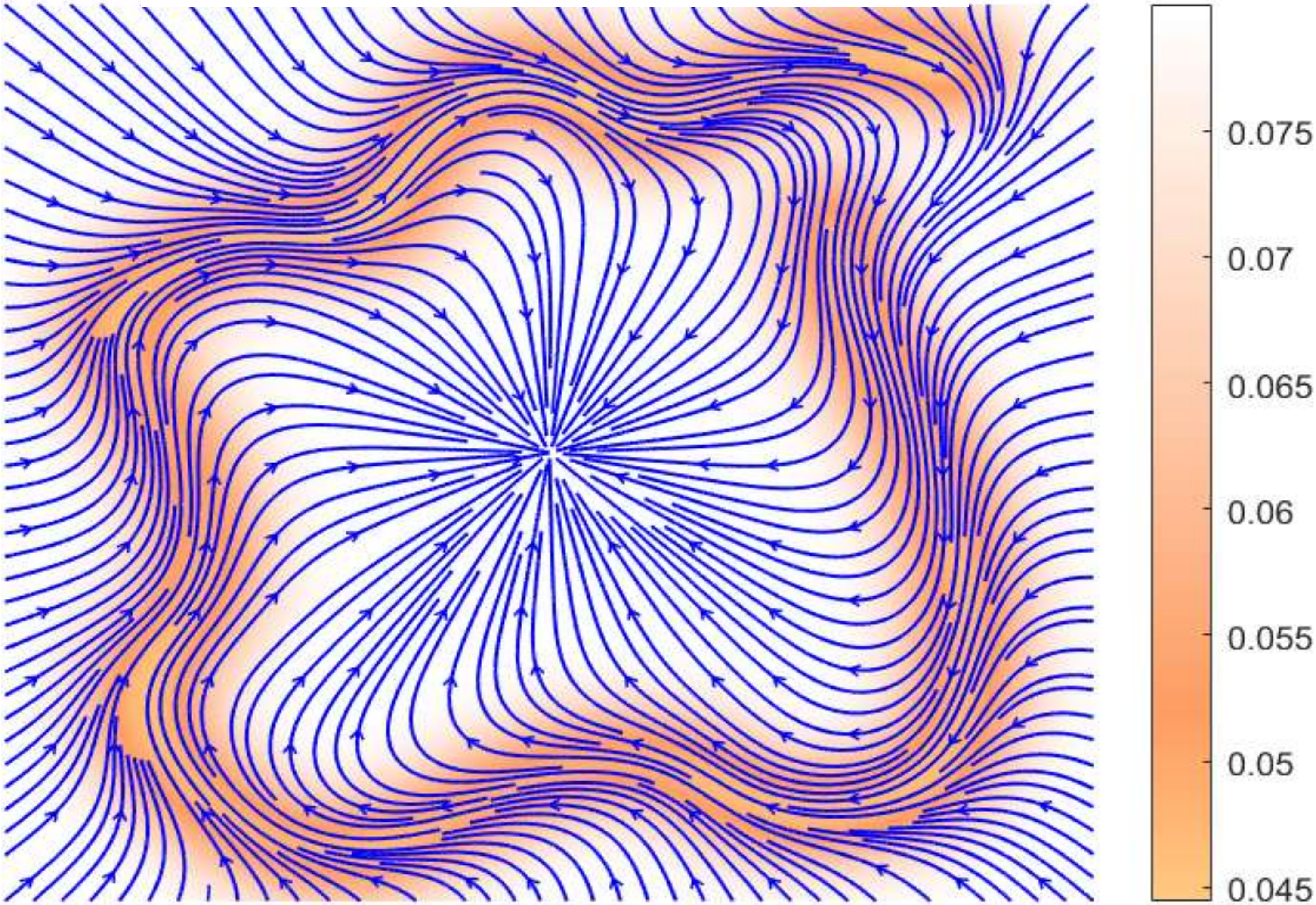
Example: Modulation Learned by GPR

Original dynamics: $\dot{x} = f\left(x\right)$

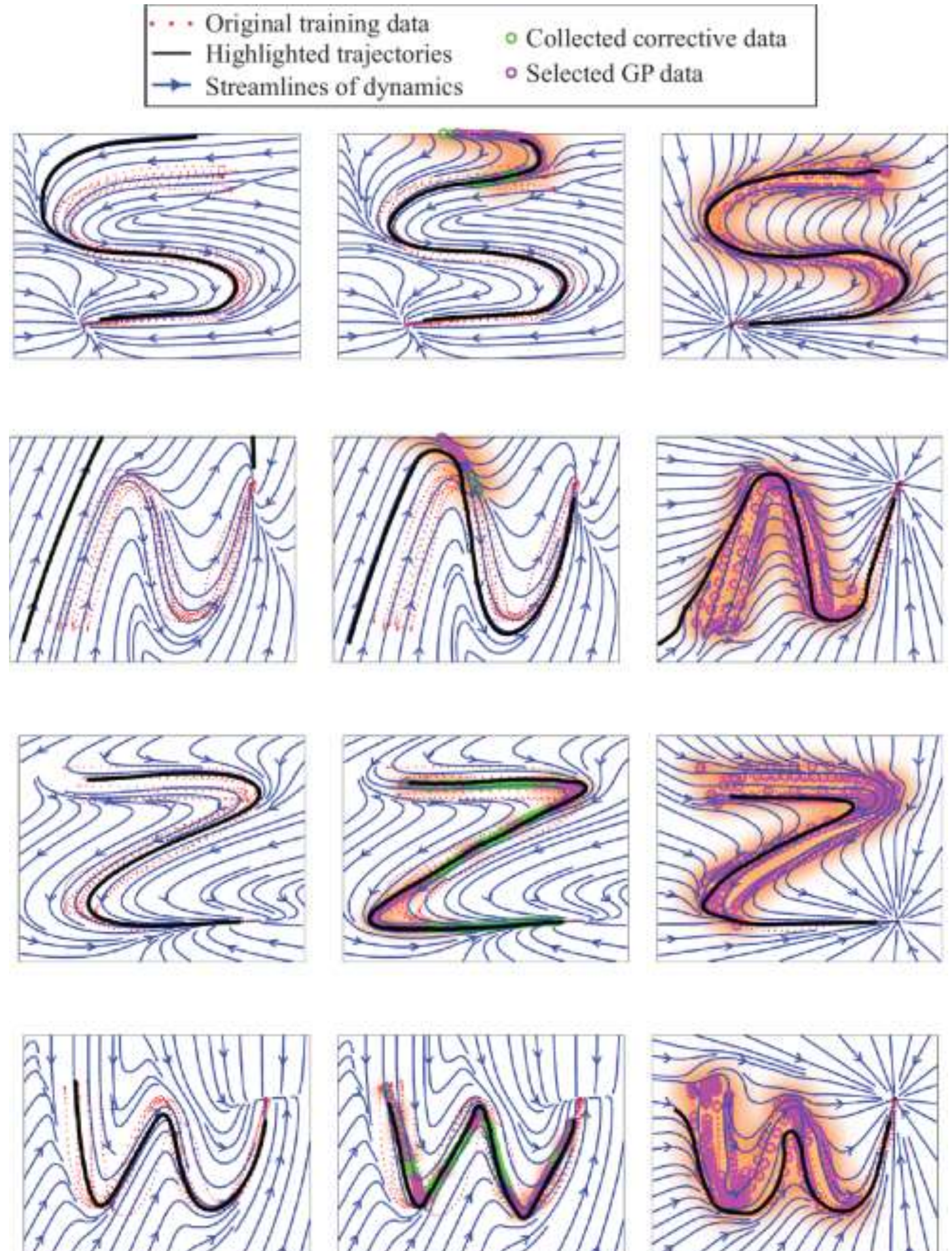


Example: Modulation Learned by GPR

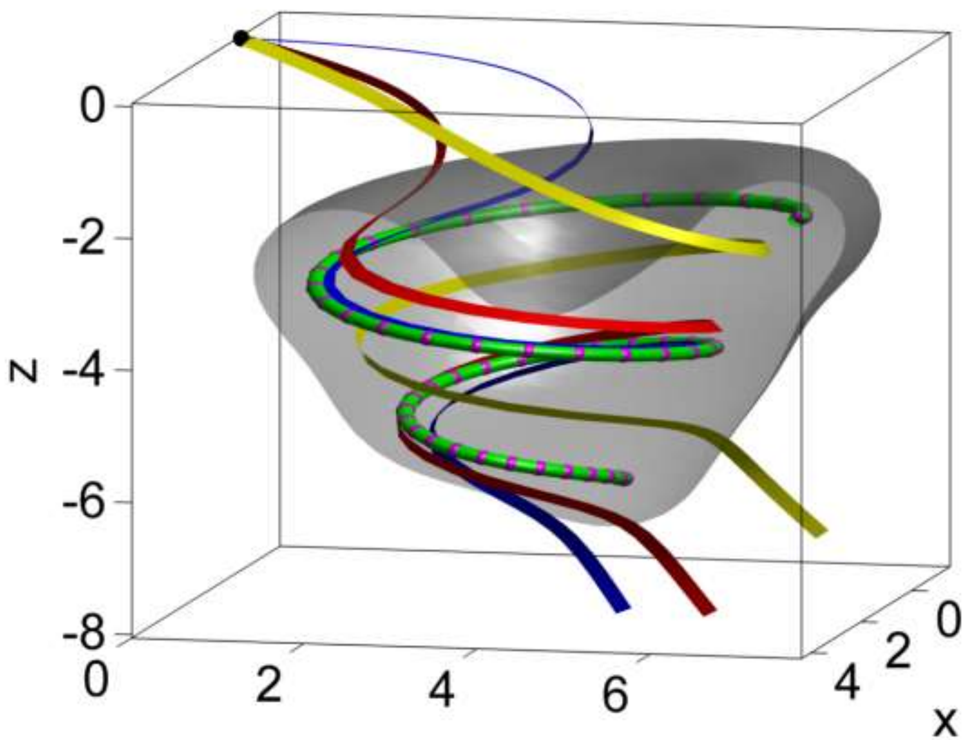
Reshaped dynamics: $\dot{x} = M(x) f(x)$



Can be used to improve SEDS fit on nonlinear trajectories.



Example: 3D Modulation Learned by GPR



Training data

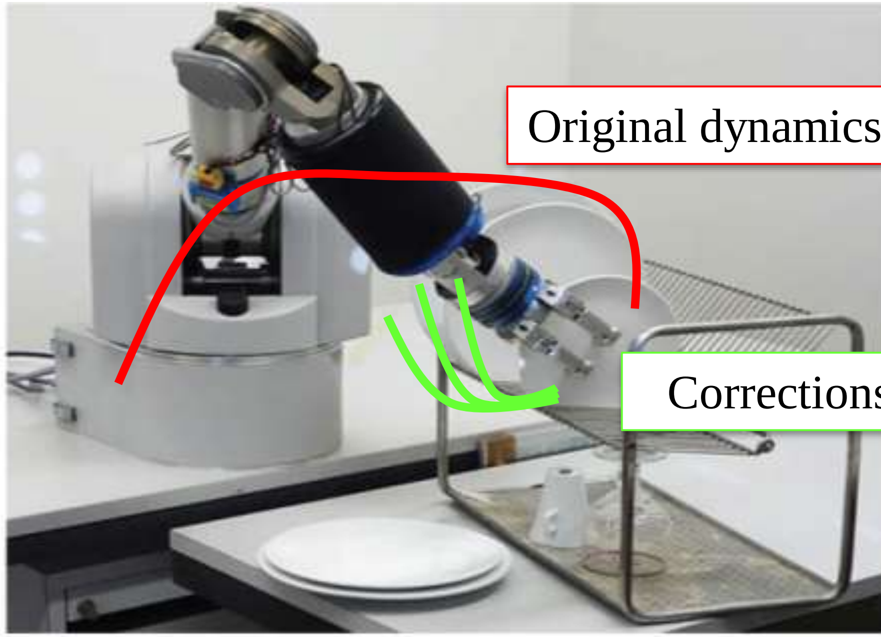
Training points used by GP-MDS

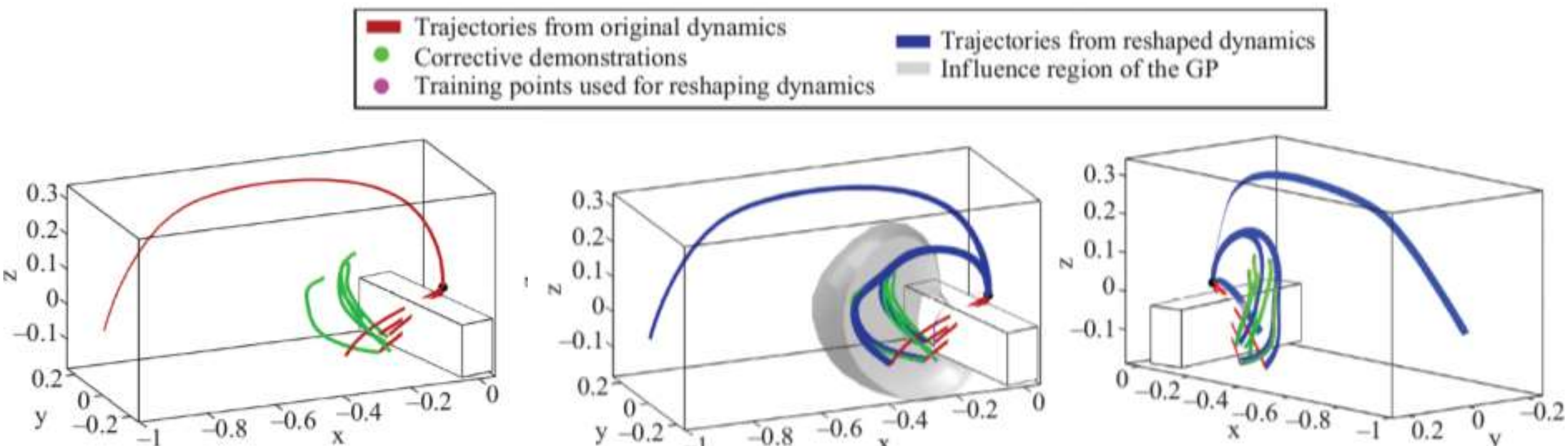
Influence region of the GP

Example trajectories from original dynamical system

Example trajectories from reshaped dynamical system

Example: 3D Modulation For Robot Control





Learning an external modulation

Sommer, N., Kronander, K. and Billard, A., 2017, Learning externally modulated dynamical systems, IROS

Learning an external modulation

Until now, the modulation depends solely on the current state of the system.

$$\dot{x} = M(x)\, f(x).$$

To provide more flexibility and allow the user to control for the activation of the modulation, we make it dependent on an external signal.

$$\dot{x} = M(x, s)\, f(x), \quad s \in \mathbb{R}^M : \text{ external input}$$

s : could be measured by any external sensor

<u>Stability conditions</u>

- *M(x,s)* locally active
- *M(x,s)* is full rank
- *M(x,s)f(x)* is stable at *x**

How should we design $M(x, s)$ to preserve properties of nominal DS?

Design of the modulation function

The modulation combines a state and input–dependent scaling and rotation.

$$M(x,s)=\left(1+\kappa(x,s)\right)R(x,s)$$

Parameterized by $\theta(x,s) \;=h_s(s)\left[\phi(x)\mu_R,\kappa(x)\right]$

Rotation parameters $\phi(x):\mathbb{R}^N \rightarrow [-\pi;\pi]$

Design an state-dependent angle function: $\phi(x)=h_x(x)\phi_c,$

$\phi_c \in [-\pi,\pi]$

Speed scaling

$\kappa(x):\mathbb{R}^N \rightarrow \mathbb{R}^+$

Design of the modulation function

The modulation combines a state and input–dependent scaling and rotation.

$$M(x,s) = (1+\kappa(x,s))R(x,s)$$

Parameterized by $\theta(x,s) \ = h_s(s)\left[\phi(x)\mu_R, \kappa(x)\right]$

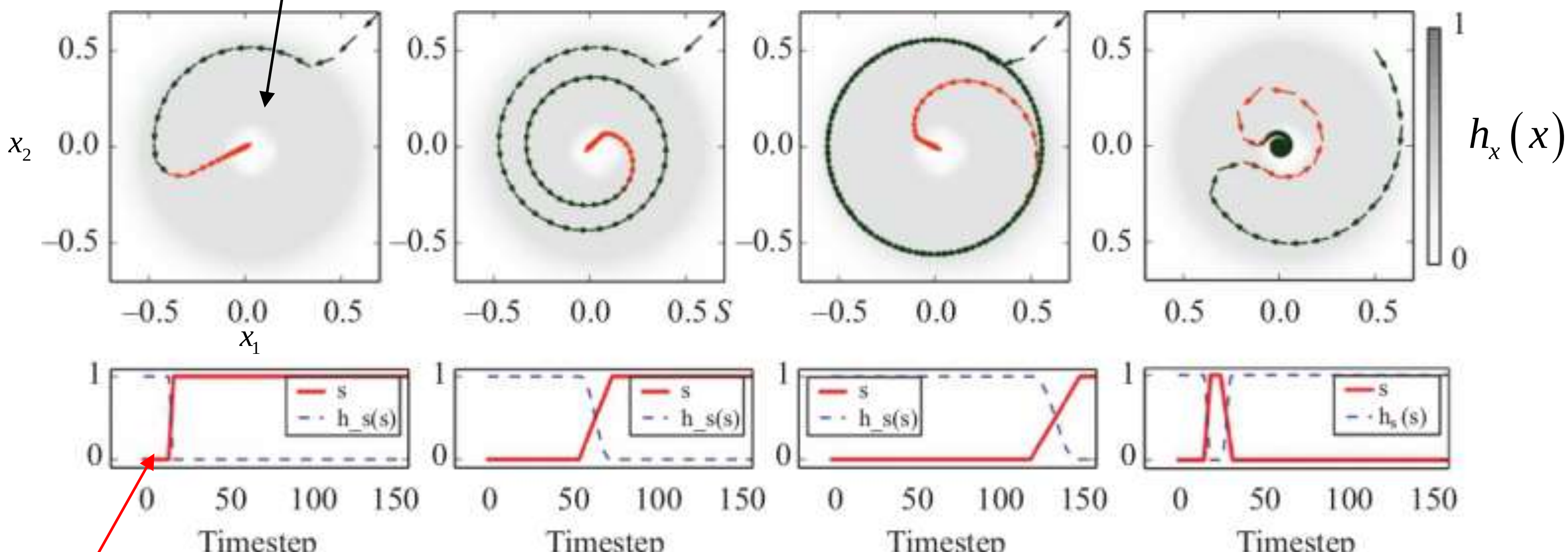
$h_s(s) : \mathbb{R}^M \rightarrow [0,1]$ - determines when the modulation is active depending on external input

h_x determines the shape of the nonlinearity: $\phi(x) = h_x(x)\phi_c$.

Example of a modulation function

$$h_x(x) = \begin{cases} 0 & \text{if } \|x\| < 0.08 \\ 50 \cdot \|x\| - 4 & \text{if } 0.08 \leq \|x\| < 0.1 \\ 1 & \text{if } 0.1 < \|x\| < 0.7 \\ -20 \cdot \|x\| + 15 & \text{if } 0.7 \leq \|x\| < 0.85 \\ 0 & \text{otherwise.} \end{cases}$$

Grey scale shows value of $h_x(x)$.



(a) Early S chg. (b) Late S chg. (c) Limit cycle (d) Different OD

Activation of input s over time.

$$h_s(s) = \begin{cases} 1 & \text{if } s < 0.0 \\ 1 - 10s^3 + 15s^4 - 6s^5 & \text{if } 0.0 \leq s \leq 1.0 \\ 0 & \text{otherwise.} \end{cases}$$

Application for tactile exploration

1. Original dynamics

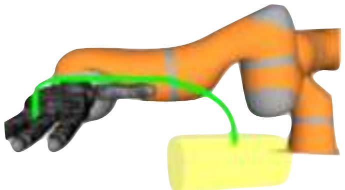
2. Learn parameterization of modulation

$$\theta(x,s)=h_s(s)\left[\phi(x)\mu_R,\kappa(x)\right]$$

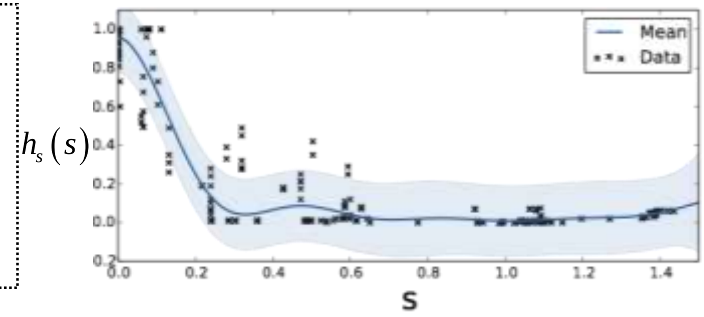
3. Learn activation function

$$h_s(s):\mathbb{R}^M\rightarrow[0,1]$$

Original dynamics



Modulated dynamics



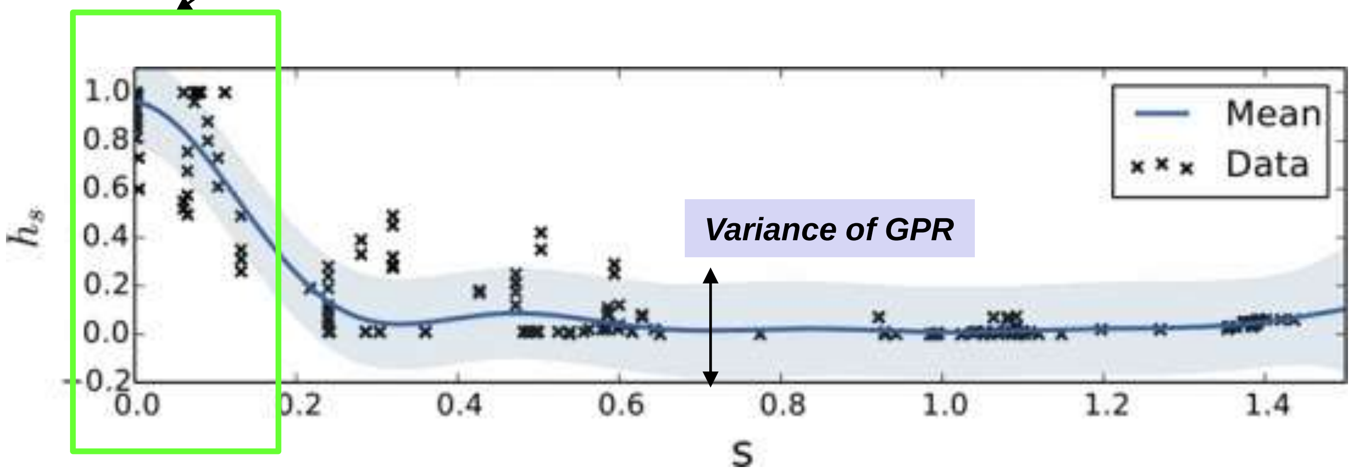
Application to tactile exploration

Scenario:

- "Blind" exploration of the space
- Each time make contact with object, localize the object and estimate its orientation
- Position the hand in the right orientation to grasp the object once certain of its location

Learn modulation function of the input

The more certain the algorithm is on position of object, the larger h_s.

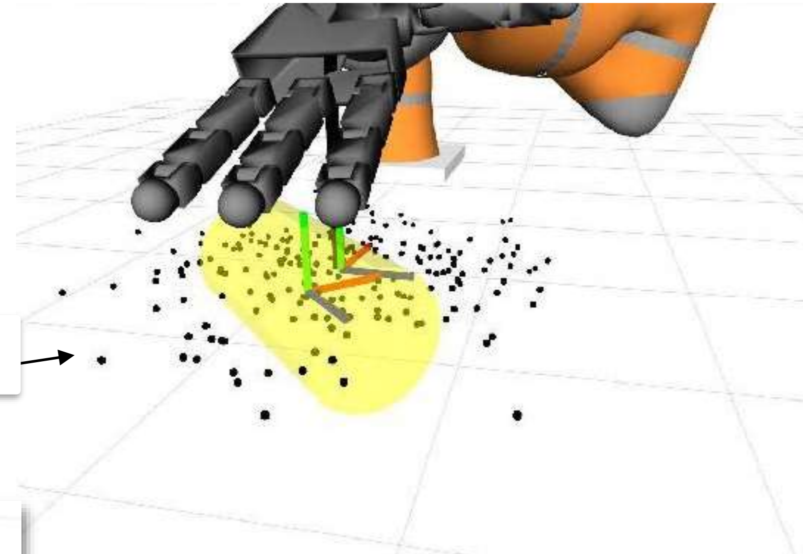


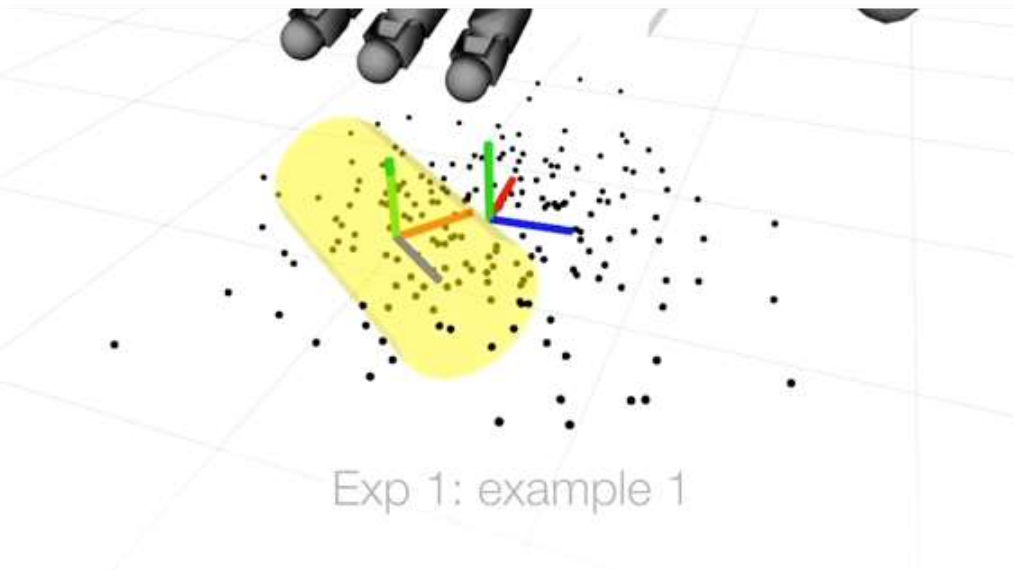
s: variance of object's position estimate

Learn modulation function of the input

Use particle filter to determine the likelihood of the object's location

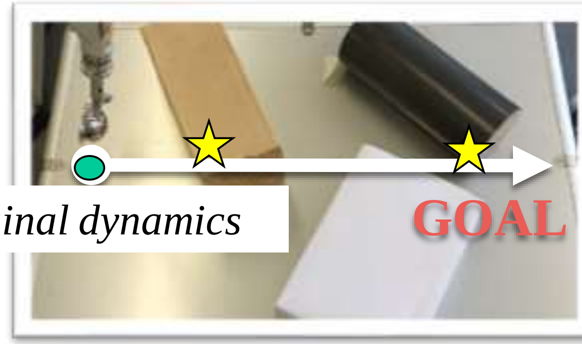
Spread of particles at initialization

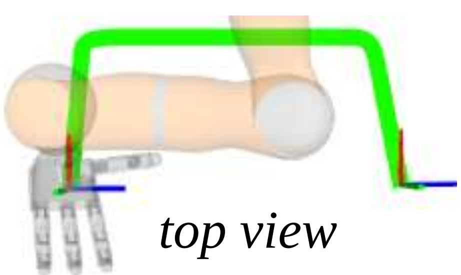




Application to obstacle avoidance

- External input: collision detection
- Learn a modulation to adapt trajectory depending on collision information





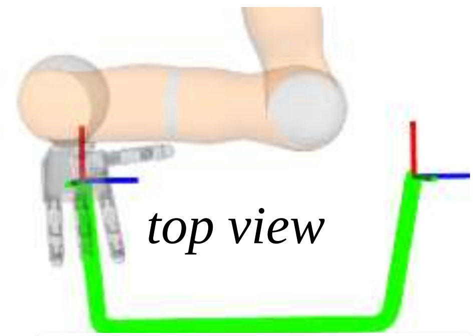
Original dynamics

External signal: *s*

Time since last contact

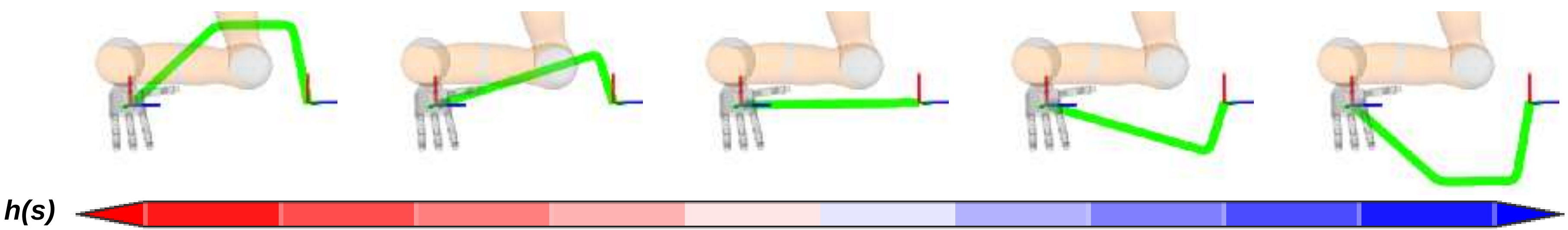
+

Angle of last collision



Modulated dynamics

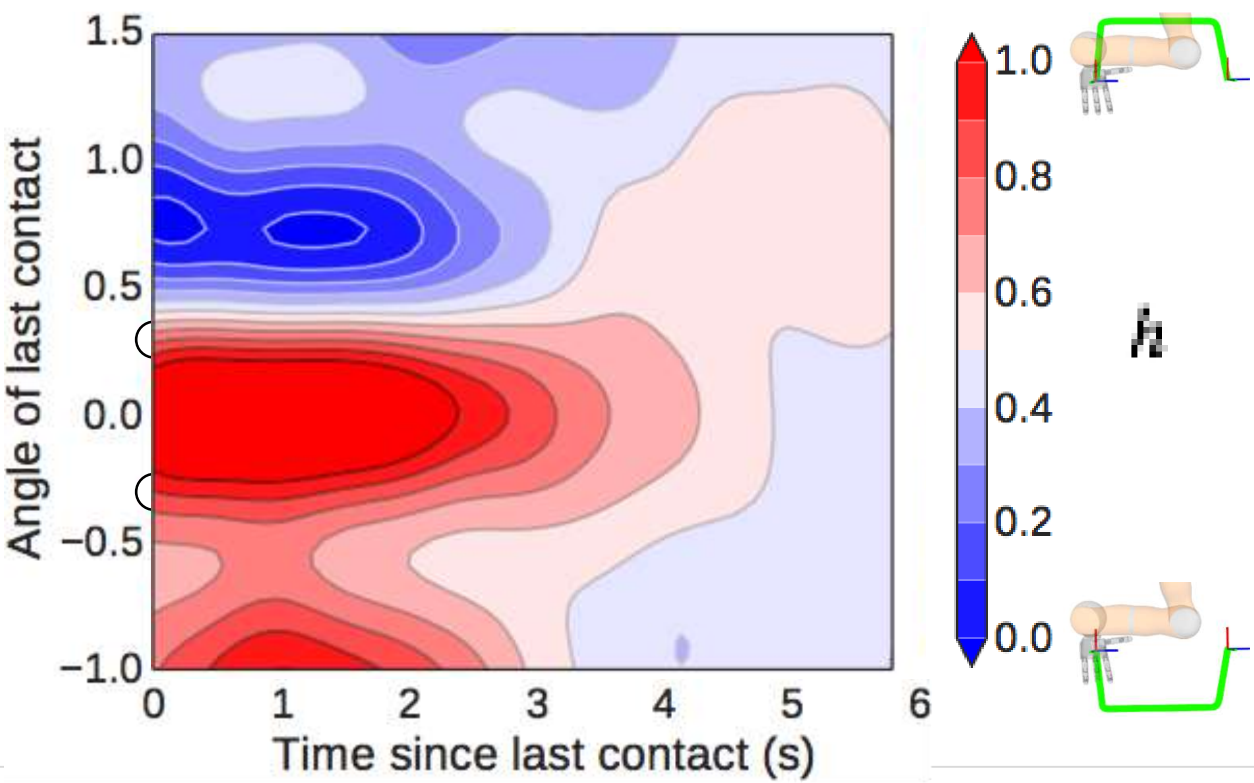
Resulting dynamics with different values of activation



0.75 0.6 0.5 0.4 0.25

Application to obstacle avoidance

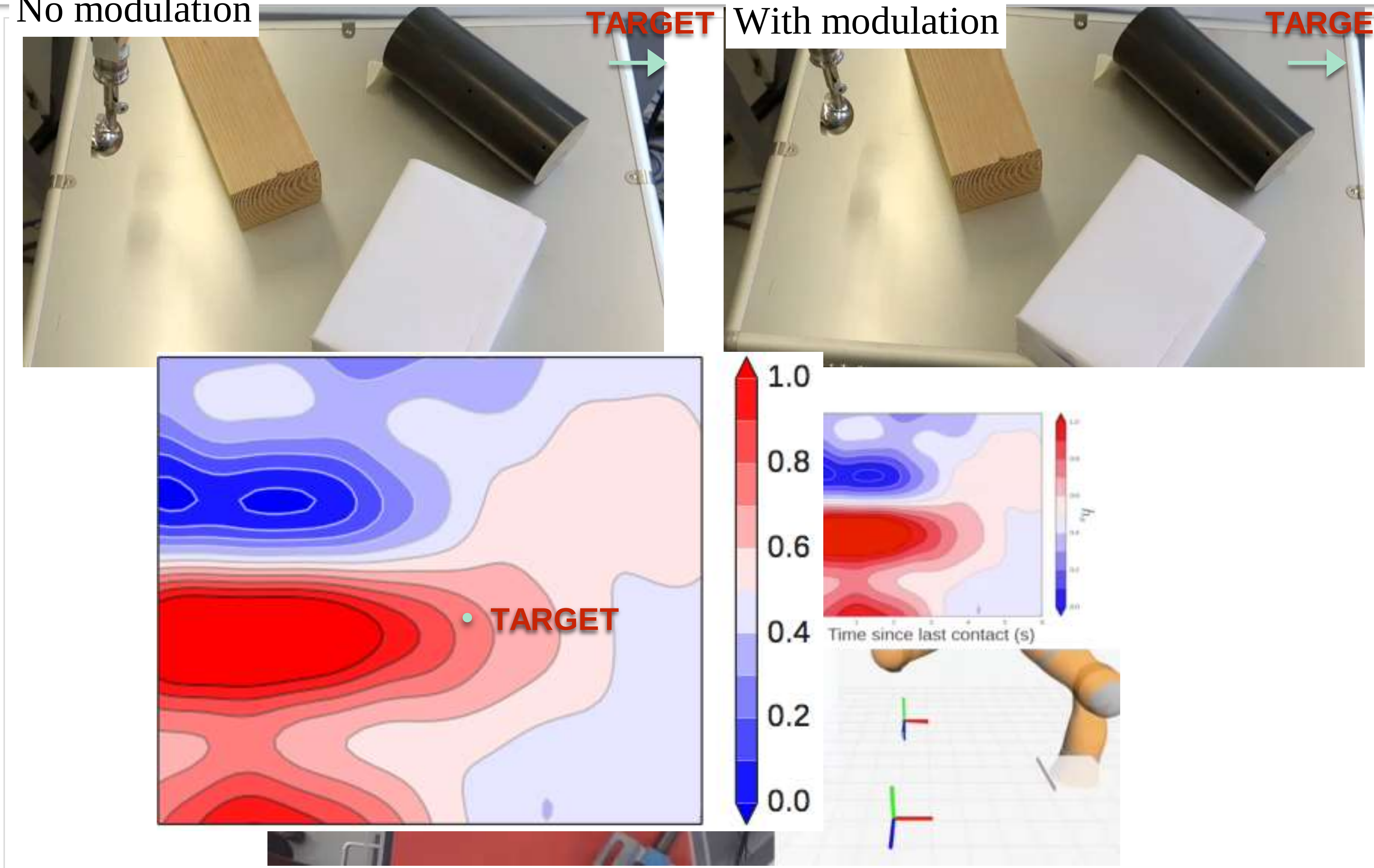
Teaching the activation function



8 demonstrations

Learn activation function (using Gaussian Process Regression)

Application to obstacle avoidance



Summary

- Introduced a closed-form modulation of DS
  - Active locally only
  - Can be activated through external input
  - Can be designed to preserve stability properties of original DS

- The modulation can be learned from data

- The modulated system inherits DS properties:
  - enables highly reactive motion

- Modulating a DS is advantageous:
  - Enables to shape a simple nominal (linear DS) to make it nonlinear
  - Well suited for learning from demonstration