Adaptive Model Predictive Control based on Gaussian Process

Statistical Learning and Stochastic Control

Lucas Rath Luzia Knödler Dimitrios Gkoutzos

Institute for Systems Theory and Automatic Control
University of Stuttgart

January 24, 2020

Motivation

MPC popular $\rightarrow$ dynamics model needed for MPC $\rightarrow$ complicated $\rightarrow$ GP

Table of Contents

Overview Gaussian Process-based Model Predictive Control

Model Predictive Control (MPC)

Model of Plant Dynamics
(Identification is challenging due to complex dynamics, unknown parameters)

Learning the **full plant dynamics** using Gaussian process (GP) regression

Generation of **local GPs**:

for each subspace of the GP input space a different GP is identified

simple and fixed **nominal model**
+
Learning **disturbance model** using GP regression

The disturbance model represents the error between the true behaviour of the plant and the nominal model

Overview Gaussian Process-based Model Predictive Control

Model Predictive Control (MPC)

Model of Plant Dynamics
(Identification is challenging due to complex dynamics, unknown parameters)

Learning the **full plant dynamics** using Gaussian process (GP) regression

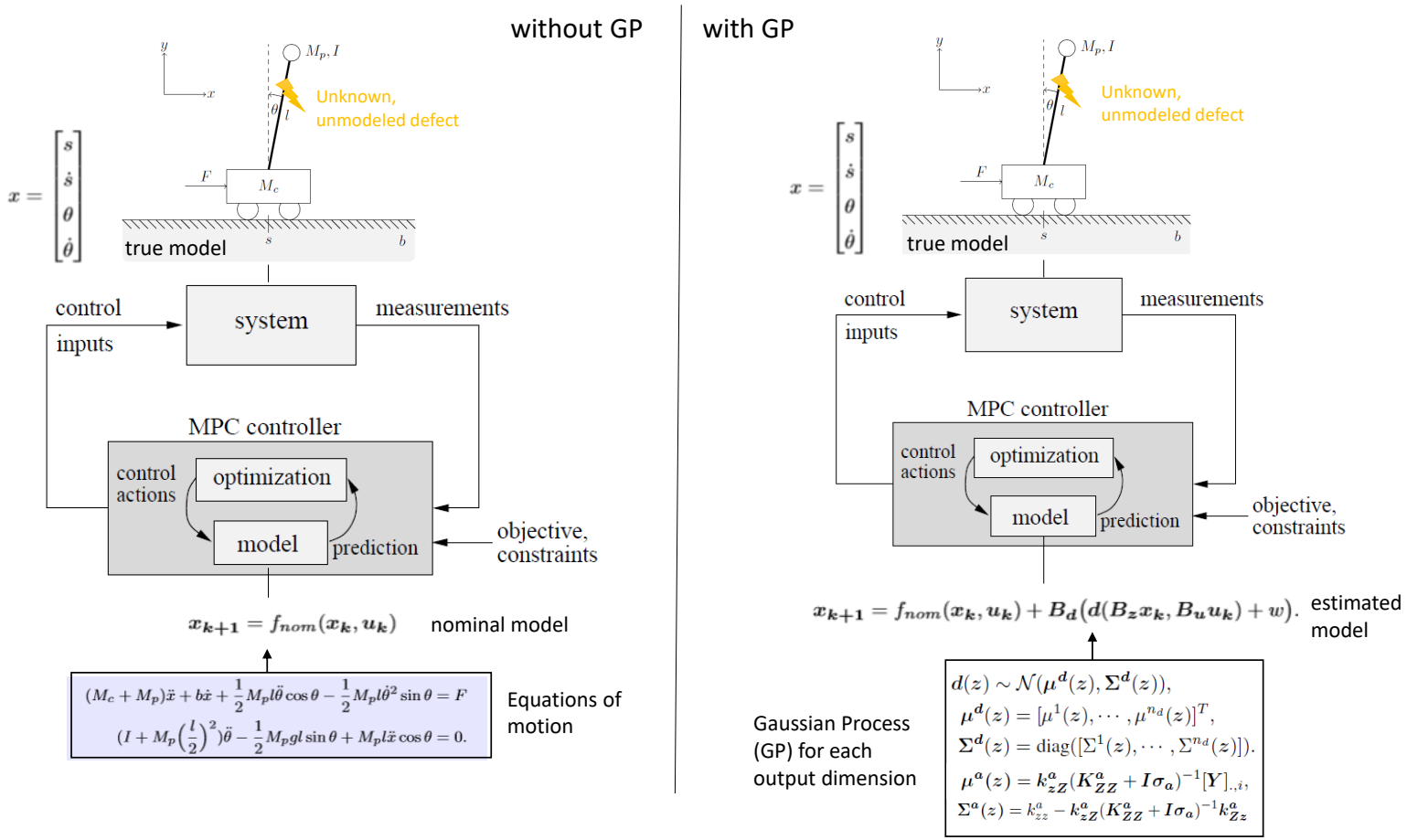
Generation of **local GPs**:

for each subspace of the GP input space a different GP is identified

simple and fixed **nominal model**
+
Learning **disturbance model** using GP regression

The disturbance model represents the error between the true behaviour of the plant and the nominal model

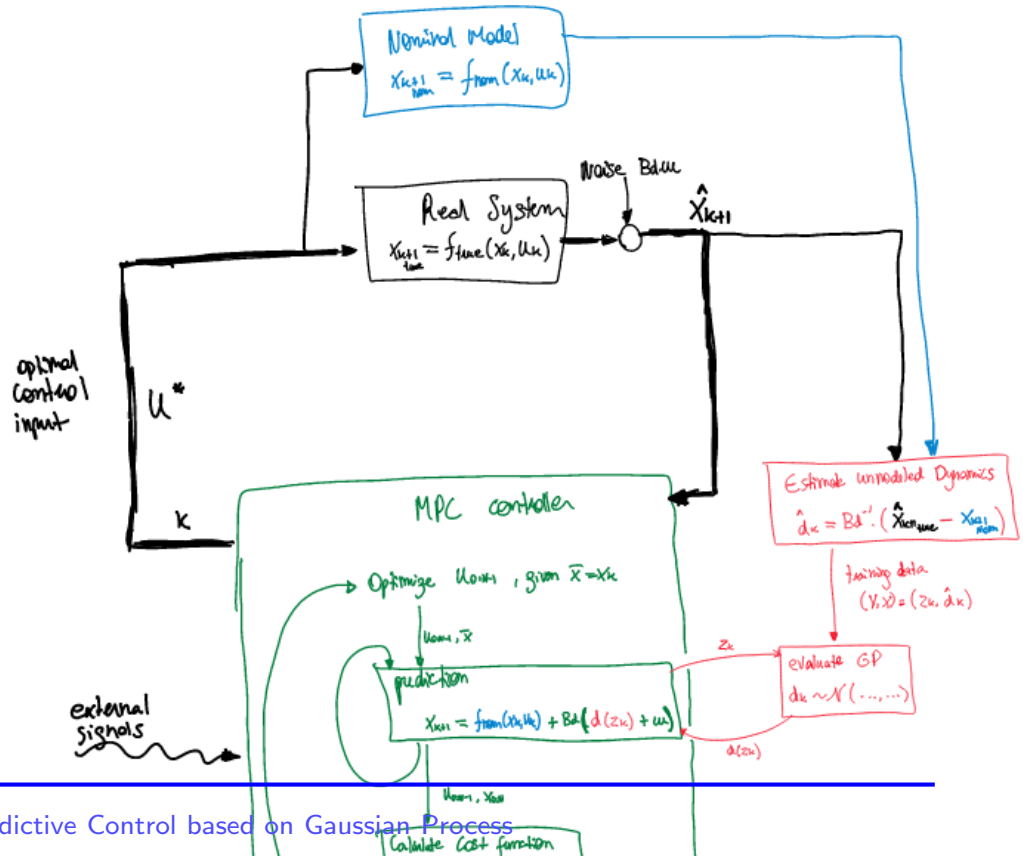
Structure GP-based MPC



Based on Arnold, M. , Negenborn, R.R. , Andersson, Göran , De Schutter, Bart. (2015). Multi-Area Predictive Control for Combined Electricity and Natural Gas Systems. 2009 European Control Conference, ECC 2009.

Introduction

diagram of MPC combined with GP (add GP to the diagram below)
different applications of GPs with MPC (nominal model + learn disturbance model, learn full dynamics, learn local GPs)
Do you mean something like this Luzia? My drawing is bit confusing maybe. I tried to copy what you did, but adding GP. Maybe we can have one diagram in the Introduction, where we just have the general structure and then an additional one after we explained MPC and GP. Because we shouldn't start with a to complicated diagram. I just realised, that you already put it after MPC and GP. perfect.



Gaussian Process

Definition (Gaussian Process (GP))

Let $X = [x_1, \ldots, x_N] \in \mathbb{R}^{n \times N}$ be a set of points, $f : \mathbb{X} \to \mathbb{R}$, $\mathbb{X} \subset \mathbb{R}^n$ a random function, $m : \mathbb{X} \to \mathbb{R}$ be any function and $k : \mathbb{X} \times \mathbb{X} \to \mathbb{R}$ be a valid covariance function.

Then a Gaussian Process $f(x) \sim \mathcal{GP}(m(x), k(x, x'))$ is defined as a probability distribution over the function space f, such that the joint distribution of any finite collection of function evaluations Y:

$$Y = [y_1, \ldots, y_N]^{\mathsf{T}} = [f(x_1), \ldots, f(x_N)]^{\mathsf{T}} = f(X) \tag{1}$$

is Gaussian, i.e.:

$$Y \sim \mathcal{N}(\underbrace{\mathbb{E}(f(X))}_{m(X)}, \underbrace{\mathbb{V}ar(f(X))}_{K(X,X)}) \tag{2}$$

where $m(X) = [m(x_1), \ldots, m(x_N)]^T \in \mathbb{R}^n$ is any function that evaluates the joint mean and $K(X, X) \in \mathbb{R}^{n \times n}$ is a joint covariance matrix defined element-wise by any valid covariance function $K(X, X)_{i,j} = k(x_i, x_j)$.

Gaussian Process Regression

Assume now an unknown function $f(x) : \mathbb{R}^n \to \mathbb{R}$ to be learned that defines a Gaussian Process $f(x) \sim \mathcal{GP}(m(x), k(x, x))$.

Further, assume that the observation outputs y are corrupted by an additive i.i.d. Gaussian noise $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$. such that:

$$y = f(x) + \epsilon \tag{3}$$

Consider now a training Dataset $\mathcal{D} = \{(x_i, y_i)|i = 1, \ldots, N\}$ with N observations pairs (x_i, y_i), such that $X = [x_1, \ldots, x_N]$ and $Y = [y_1, \ldots, y_N]$.

Then, clearly, the Likelihood $p(Y|X)$ is given by:

$$Y|X \sim \mathcal{N}\left(m(X), K(X, X) + \sigma_n^2 I\right) \tag{4}$$

Gaussian Process Regression

Given a new set of test points X^*, one may now wish to calculate the posterior distribution $p(f^*|X^*, Y, X)$, where we denote $f^* = f(X^*)$.

Since we know that the prior is distributed as $p(f^*|X^*) = \mathcal{N}(m(X^*), K(X^*))$, we can calculate the posterior distribution $p(f^*|X^*, Y, X)$ by means of Bayesian inference:

$$\underbrace{p(f^*|X^*, Y, X)}_{posterior} \propto \underbrace{p(f^*|X^*)}_{prior} \underbrace{p(Y|X)}_{likelihood} = \underbrace{p(f^*, Y|X^*, X)}_{joint\ probability} \tag{5}$$

where

$$\begin{bmatrix} f^* \\ Y \end{bmatrix} = \begin{bmatrix} f(X^*) \\ f(X) \end{bmatrix} + \begin{bmatrix} 0 \\ I \end{bmatrix} \Sigma, \qquad \Sigma \sim \mathcal{N}(0, \sigma_n^2 I) \tag{6}$$

leads to the joint distribution

$$\begin{bmatrix} f^* \\ Y \end{bmatrix} \Big/ X^*, X \sim \mathcal{N}\left(\begin{bmatrix} m(X^*) \\ m(X) \end{bmatrix}, \begin{bmatrix} K(X^*, X^*) & K(X, X^*) \\ K(X^*, X) & K(X, X) + \sigma_n^2 I \end{bmatrix} \right), \tag{7}$$

and, as before, $m(X) = \mathbb{E}[f(X)]$ and $K(X, X^*) = \mathbb{V}ar(f(X), f(X^*))$ are functions that define the mean and covariance of the function f at different locations.

Gaussian Process Regression

Finally, using the Update Lemma

Lemma (von Misses Update Lemma)

$$\begin{bmatrix} x \\ y \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu_x \\ \mu_y \end{bmatrix}, \begin{bmatrix} P_{xx} & P_{xy} \\ P_{yx} & P_{yy} \end{bmatrix} \right) \tag{8}$$

$$\Rightarrow \quad x \,|\, y \quad \sim \mathcal{N}\left(\mu_x + P_{xy}\, P_{yy}^{-1}(y - \mu_y) \;,\; P_{xx} - P_{xy}\, P_{yy}^{-1}\, P_{yx}\right)$$

with equation (7) leads to the posterior (predictive distribution of the function values at X^*):

Posterior / Predictive Distribution

$$f^*|X^*, Y, X \sim \mathcal{N}\left(\begin{array}{l} m(X^*) + K(X^*, X)\left(K(X, X) + I\sigma_n^2\right)^{-1}(Y - m(X)) \;, \\ K(X^*, X^*) - K(X^*, X)\left(K(X, X) + I\sigma_n^2\right)^{-1} K(X, X^*) \end{array} \right) \tag{9}$$

which is also a Gaussian Process.

Kernel Trick and Function-Space View

Making use of the *Kernel Trick*, leads to the Squared Exponential Kernel (SE):

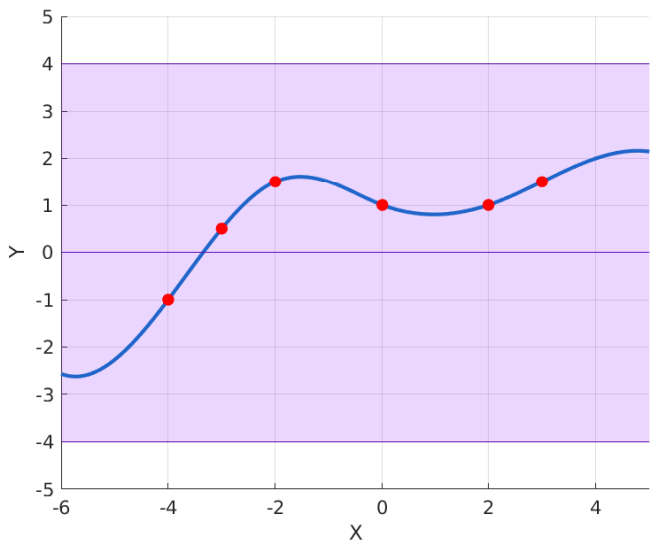
$$k(x, x') = \sigma_f^2 \, exp\left(0.5 \left\| x - x' \right\|_{M-1}^2\right) \tag{10}$$

And as usually is done, the zero mean prior is chosen:

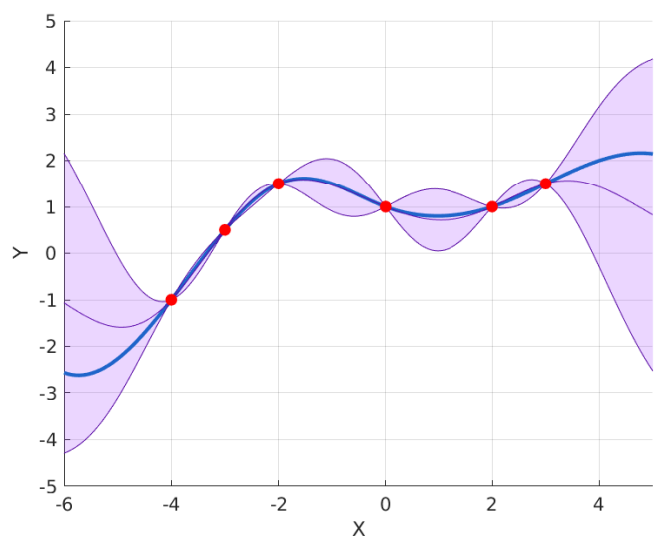
$$m(x) = \mathbf{0} \tag{11}$$

Lets visualize now how the prior and the posterior might look like for a simple 1D example:

Prior distribution: $p(f^*|X^*)$



Posterior distribution: $p(f^*|X^*, Y, X)$



Obs: in blue the true function and scattered in red the training data

Efficient Implementation

Problem: The computational complexity of a GP regression strongly depends on the number of data points N. The most computational demanding task is the matrix inversion: $(K(X,X) + I\sigma_n^2)^{-1}$.

$$f^*|X^*, Y, X \sim \mathcal{N}\left(\begin{array}{l} m(X^*) + K(X^*,X)\left(K(X,X) + I\sigma_n^2\right)^{-1}(Y - m(X)) \ , \\ K(X^*,X^*) - K(X^*,X)\left(K(X,X) + I\sigma_n^2\right)^{-1}K(X,X^*) \end{array}\right) \tag{12}$$

Solution: Use *Cholesky* decomposition: $(x = A^{-1}b \Leftrightarrow x = L^{\mathsf{T}}\backslash(L\backslash b), A = LL^{\mathsf{T}})$

$$\begin{aligned} L &= cholesky(K(X,X) + \sigma_n^2 I) \\ \alpha &= L^{\mathsf{T}}\backslash(L\backslash(Y - m(X))) \\ v &= L\backslash K(X,X^*) \end{aligned} \tag{13}$$

$$\begin{aligned} \mathbb{E}[f^*|X^*, Y, X] &= m(X^*) + K(X^*,X)\alpha \\ \mathbb{V}[f^*|X^*, Y, X] &= K(X,X) - v^{\mathsf{T}}v \end{aligned}$$

Obs: each output dimension is treated as a different GP (with possibly different hyper-parameters as well).

Hyper-parameter Optimization

The GP Likelihood is Gaussian and in the case of the zero mean function, is given by:

$$Y|X,\theta \sim \mathcal{N}(0, \underbrace{K(X,X) + \sigma_n^2 I}_{K_y}) \tag{14}$$

where $\theta = [\{M\}, \sigma_f^2, \sigma_n^2]$ is a vector containing all hyper-parameters.

Among many possible choices, we chose to parameterize the length-scale covariance matrix M as diagonal positive-semidefinite:

$$M = \begin{bmatrix} l_1 & & 0 \\ & \ddots & \\ 0 & & l_n \end{bmatrix} \tag{15}$$

with $l_i \geq 0, \forall i \in 0, \ldots, n$

such that the hyperparameter vector becomes $\theta = [l_1, \ldots, l_n, \sigma_f^2, \sigma_n^2]$

Hyper-parameter Optimization

One may optimize the GP hyper-parameters by maximizing the Log Likehood (LL):

$$log\, p(Y|X,\theta) = -\frac{1}{2} y^T K_y^{-1} y - \frac{1}{2}\log|K_y| - \frac{n}{2}\log(2\pi)$$

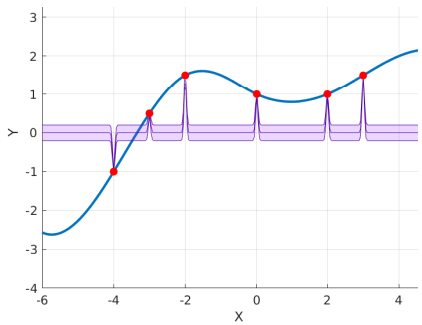
$$\theta = \arg\max_{\theta}\ log\, p(Y|X,\theta) \tag{16}$$

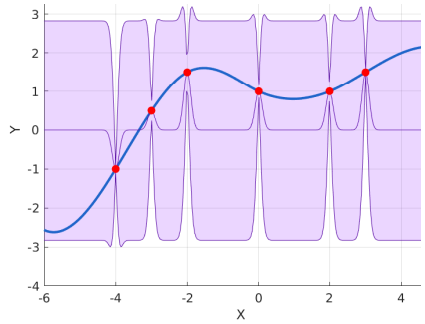
which allows local optimization of the hyper-parameters, since (16) is nonconvex (one optimization problem for each output dimension).

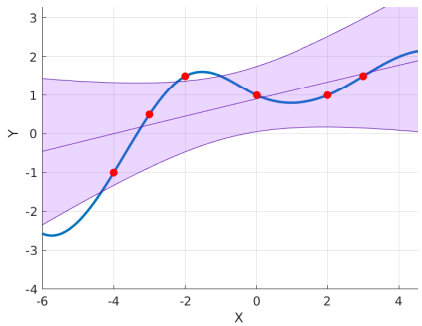
Obs: Even gradient-free tools like `fmincon` from Matlab, showed to be efficient. Nevertheless, the gradient of (16) can be easily derived, as shown in [RW06]

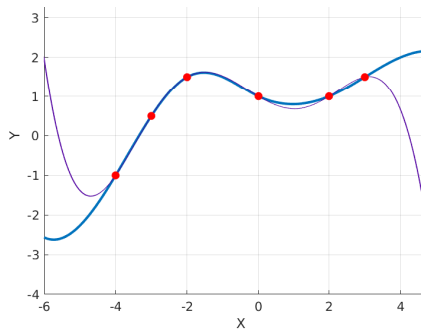
Hyper-parameter Optimization

Different hyper-parameters might lead to completely different Gaussian processes

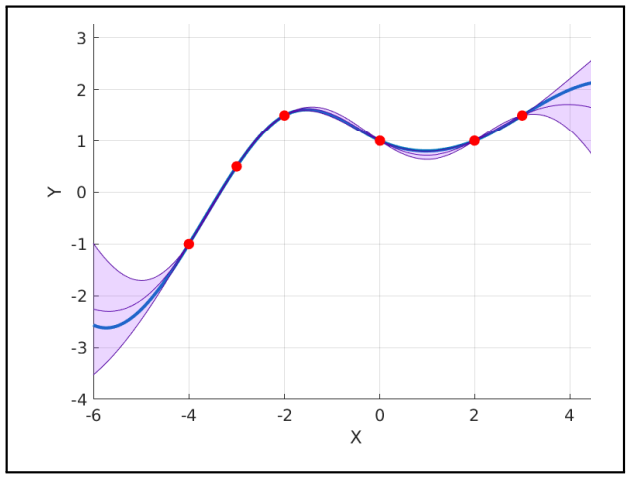








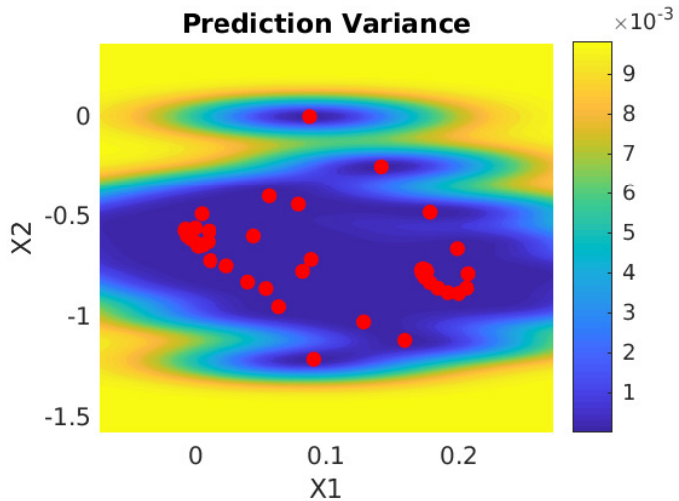
After hyper-parameter optimization



Sparse Gaussian Process Regression

Problem: The computational complexity of a GP regression strongly depends on the number of data points N. How to keep N low, while ensuring good prediction performance?

Solution: (i) Set a size limit for the dictionary. (ii) If dictionary is full, make a selection of the so called, *inducing points*.



Strategy A) Choose the closest point in the Euclidean space to the new point to be replaced.

Strategy B) Add new points, update model and remove the points with the lowest covariances.

Nonlinear MPC (NMPC)

NMPC

$$
\begin{aligned}
x^*_{0:N}, u^*_{0:N-1} = \underset{x_{0:N}, u_{0:N-1}}{\arg\min} \quad & \sum_{k=0}^{N-1} f_o(x_k, u_k) + \phi(X_N) \\
s.t. \quad & x_{k+1} = f_d(x_k, u_k) \\
& x_0 = \bar{x} \\
& x_k \in \mathcal{X}(x_k) \\
& x_N \in \mathcal{X}_f \\
& u_k \in \mathcal{U}(x_k)
\end{aligned}
\tag{17}
$$

NMPC Algorithm:

1) Measure (estimate) current state $\bar{x}$
2) Solve the open-loop discrete-time infinite-horizon optimal control problem (17)
3) Implement the first portion of the optimal input $u_{NMPC}(\bar{x}) = u^*_0$
4) Go to 1)

Adaptive Process Model

One of the challenges of MPC is that unmodeled dynamics might deteriorate its performance. To this end, an adaptive process model [KVR+19] is proposed as follows:

Adaptive Process Model

$$x_{k+1} = f_d(x_k, u_k) + B_d\ (d(z_k) + w) \tag{18}$$

where

$z_k = [Bz_x.x_k; Bz_u.u_k] \in \mathbb{R}^{n_z}$	collection of relevant features
$w \sim \mathcal{N}(0, \sigma_n^2)$	process noise
$f_d(z_k) : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$	nominal discrete-time process model
$d(x_k, u_k) \sim \mathcal{GP}(m(z_k), K(z_k))$	Gaussian Process function

The GP function $d(z_k)$ will be responsible for capturing the unmodeled dynamics

The matrix $B_d \in \mathbb{R}^{n \times n_d}$ is used to select a subspace of states that are affected by both GP process and noise. In a similar way, the matrices Bz_x and Bz_u select a subset of states that are relevant for regression.

Estimating Unmodeled Dynamics - Generating GP Training Data

Given the adaptive process model

$$x_{k+1} = f_d(x_k, u_k) + B_d\ (d(z_k) + w) \tag{19}$$

Assume that the state observation $\hat{x}_{k+1}$ at time k+1 can be perfectly measured.

This means that the training data (X, Y) can be generated to be the difference between the true measured and the nominal simulated states:

$$\begin{aligned} X &= z_k \\ Y &= B_d^{\dagger}\left(x_{k+1} - f_d(x_k, u_k)\right) = d_{true} + w \end{aligned} \tag{20}$$

Propagation of uncertainty

The adaptive process model allows us to

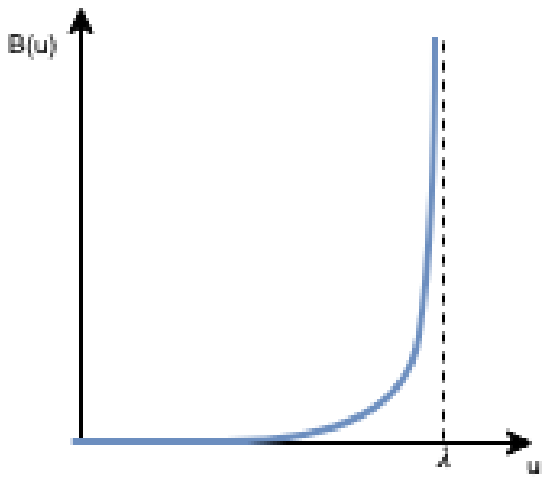
Efficient MPC Formulation

Idea: Can we somehow get rid of the input constraints to improve optimizer performance?

$$u_k \in \mathcal{U} = \{u : g_u(u) \leq \lambda_u\}$$

Solution: Input constraints can be removed and incorporated as a barrier function $B(u)$ into the cost function as:

$$B(u) = q_u \left(-log\left(\lambda_u - g_u(u)\right)\right)$$



Remark: The barrier function is not defined for $g_u(u) > \lambda_u$. This so called "hard-constraints" are usually necessary (actuation limit) and should not be the responsible for infeasibility.

Efficient MPC Formulation

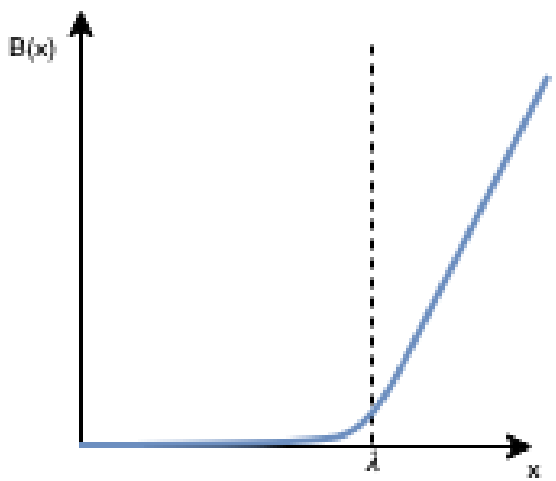
Idea: Can we somehow get rid of the state constraints to improve optimizer performance?

$$x_k \in \mathcal{X} = \{x : g_x(x) \leq \lambda_x\}$$
$$x_N \in \mathcal{X} = \{x : g_{x_N}(x) \leq \lambda_{x_N}\}$$

Solution: State constraints can also be removed and incorporated as relaxed barrier functions $B_r(x)$ into the cost function as:

$$B_r(x) = \frac{q_x}{2}\left(\sqrt{\frac{(4+\gamma(\lambda-x)^2)}{\gamma}} - (\lambda - x)\right) \approx q_x \frac{(|\lambda - x| - (\lambda - x))}{2}$$

which makes optimization more efficient while always ensuring feasibility.



Efficient MPC Formulation

Second, the equality constraints, usually corresponding only to the state space variables $x_{0:N}$, can be easily removed from the set of variables to be optimized.

This is usually possible because the state variables at any time step are a function of the given initial state and the sequence of inputs to be optimized.

To this end, we define the unconstrained nonlinear MPC as follows:

Efficient Unconstrained NMPC

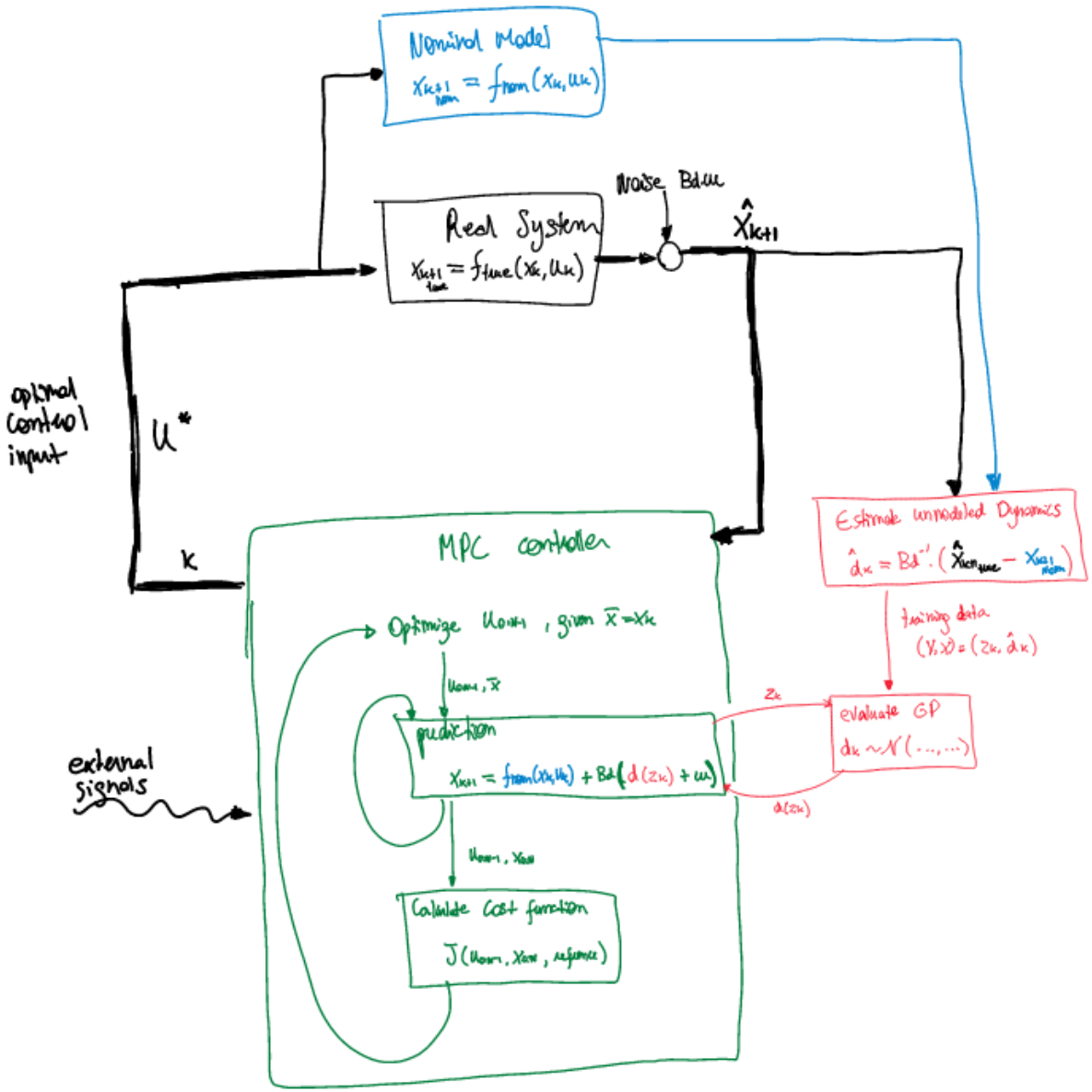
$$\min_{u_{0:N-1}} \quad \sum_{k=0}^{N-1} f_o(x_k, u_k) + q_x\, B_r(x_k) + q_u\, B(u_k) + \phi(X_N)$$

$$\text{where} \quad x_k(\bar{x}, u_{0:k-1}) = f(f(\cdots f(f(\bar{x}, u_0), u_1)\dots), u_{k-1}) \tag{21}$$

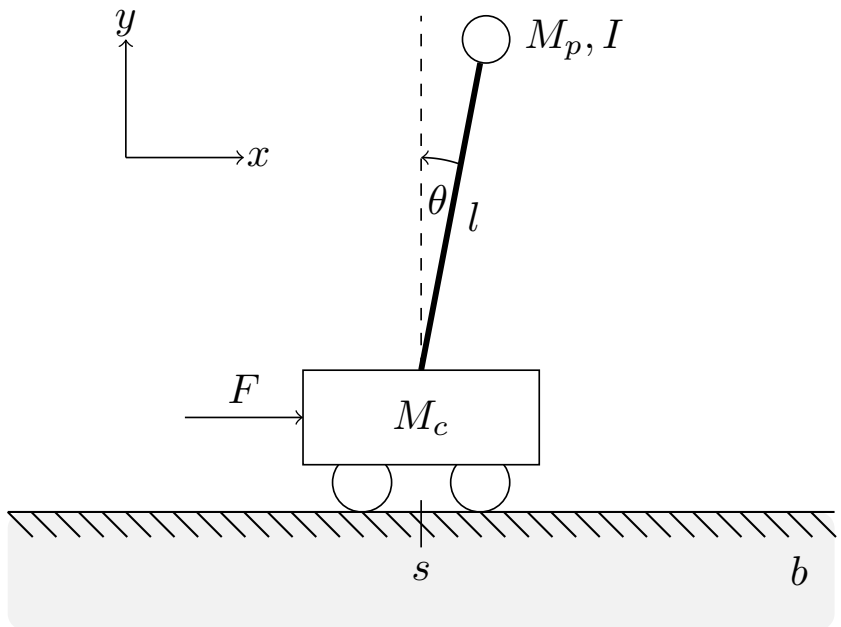
$$B_r(x) = \frac{q_\beta}{2}\left(\sqrt{\frac{(4+\gamma(\lambda-x)^2)}{\gamma}} - (\lambda - x)\right)$$

This unconstrained optimization problem, can be solved very efficiently with nonlinear optimization solvers and is always feasible.

Adaptive MPC overview



The Inverted Pendulum Problem



Continuous time dynamics

$$(M_c + M_p)\ddot{x} + b\dot{x} + \frac{1}{2}M_p l\ddot{\theta}\cos\theta - \frac{1}{2}M_p l\dot{\theta}^2\sin\theta = F \tag{22}$$

$$(I + M_p\left(\frac{l}{2}\right)^2)\ddot{\theta} - \frac{1}{2}M_p gl\sin\theta + M_p l\ddot{x}\cos\theta = 0. \tag{23}$$

Nominal and True Dynamics

Inverted Pendulum: Dynamic Models

$$
\begin{aligned}
(M_c + M_p)\ddot{x} + b\dot{x} + \frac{1}{2}M_p l\ddot{\theta}\cos\theta - \frac{1}{2}M_p l\dot{\theta}^2\sin\theta &= F \\
(I + M_p\left(\frac{l}{2}\right)^2)\ddot{\theta} - \frac{1}{2}M_p gl\sin\theta + M_p l\ddot{x}\cos\theta &= 0.
\end{aligned}
\tag{24}
$$

Nominal Model:
$$x_{k+1} = f_{d_{nom}} \tag{25}$$

True Model/ Plant Dynamics:
$$x_{k+1} = f_{d_{nom}} + B_d\big(d(B_z x_k) + w\big) \tag{26}$$

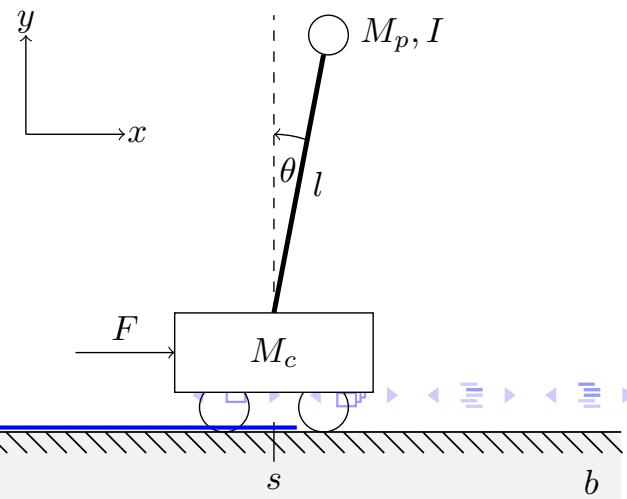
$$d(B_z x_k) = 0.1\theta - 0.01\dot{\theta} + 0.0524 \tag{27}$$

$$x = \begin{bmatrix} s & \dot{s} & \theta & \dot{\theta} \end{bmatrix}^T \tag{28}$$

$$z = B_z x = \begin{bmatrix} \theta & \dot{\theta} \end{bmatrix}^T \tag{29}$$

$$B_z = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{30}$$

$$B_d = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix}^T \tag{31}$$



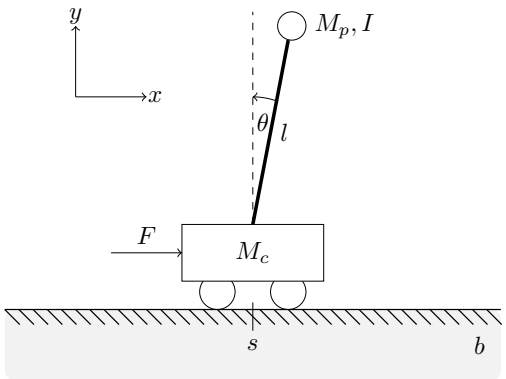
Cost function and Constraints

Model

$$\mu^x_{k+1} = f_{est}(\mu^x_k, u_k) \tag{32}$$
$$= f_{nom}(\mu^x_k, u_k) + B_d(d(\mu^z_k) + w) \tag{33}$$

with

$$z_k = \begin{bmatrix} \theta & \dot{\theta} \end{bmatrix}^T \tag{34}$$



Cost Function

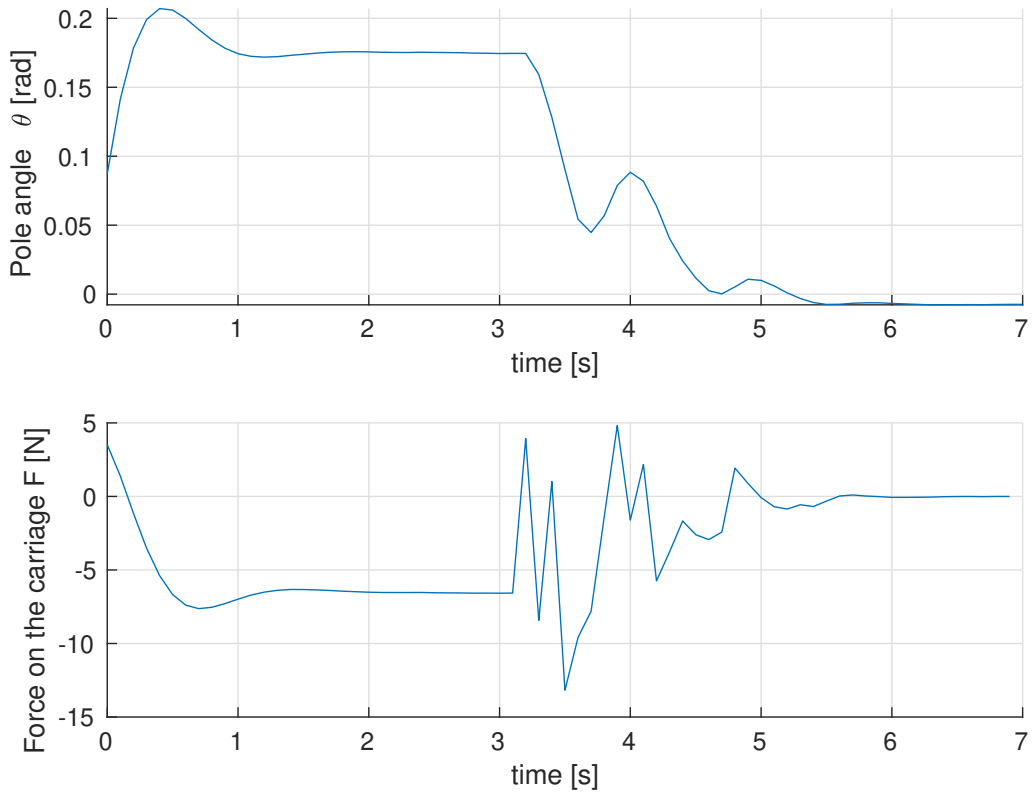
$$f_o(\mu^x_k, u_k) = (C\mu^x_k - r(t))^T Q(C\mu^x_k - r(t)) + Ru^2 \tag{35}$$
$$\phi(x_N) = (C\mu^x_N - r(t))^T Q(C\mu^x_N - r(t)) \tag{36}$$

with

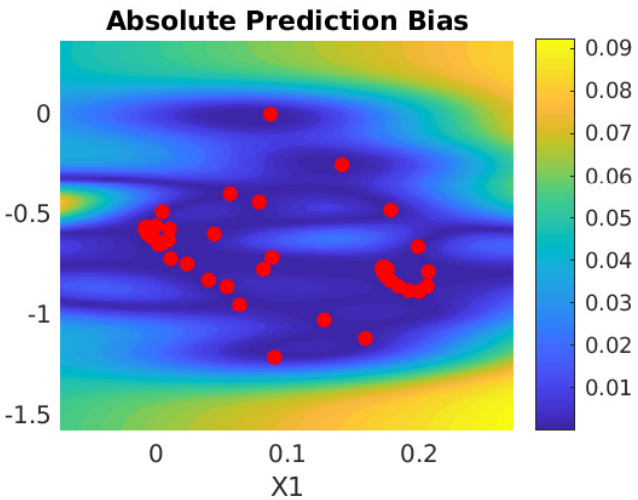
$$r(t) = [\dot{s} \;\; \theta \;\; \dot{\theta}]^{\mathsf{T}} = [0 \;\; 0 \;\; 0] \tag{37}$$

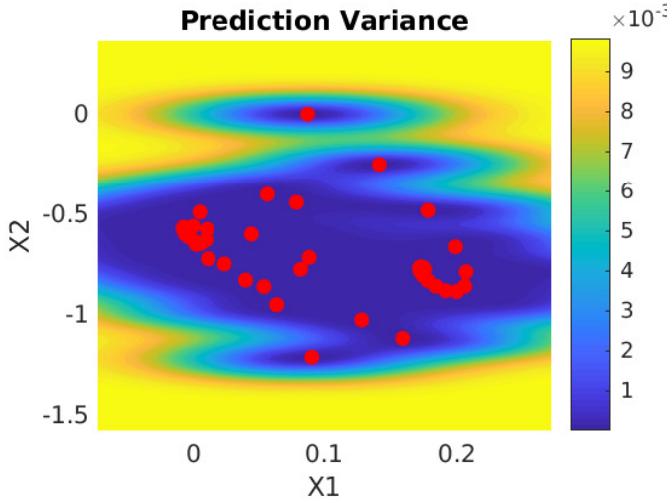
Simulation results

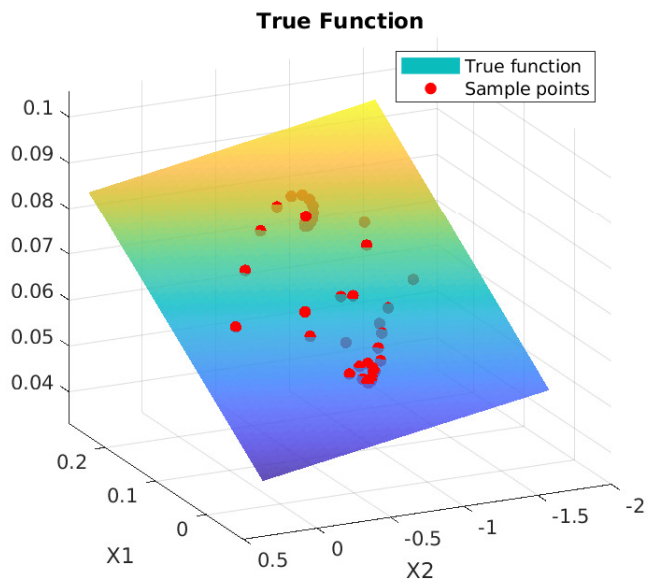
Starting positon at XXX degress. GP accumulates data to the dictionary. At t=3s, we activate predictions with GP

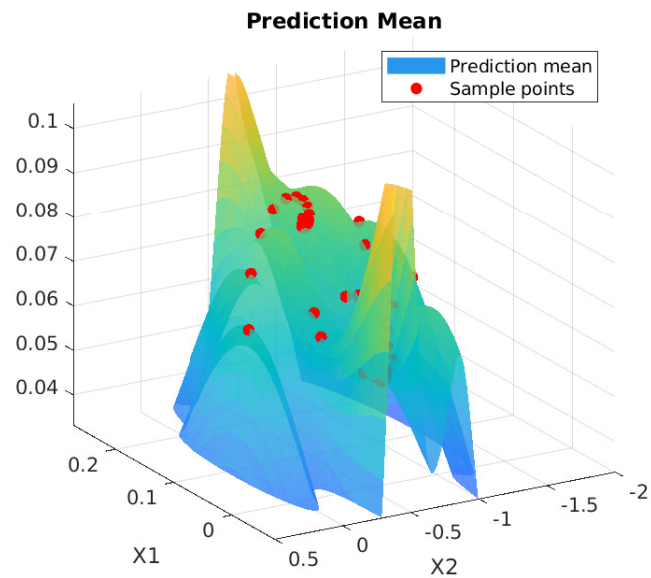


Learning analysis









Training

Discuss the posterior function before and after Hyper-parameter optimization

Show image of d before and after hyper-parameter optimization

Racing Car Problem

Vehicle Dynamics

Show basic vehicle dynamics equations

True and Nominal Dynamics

Show difference between true and nominal model

Cost function

Inspired by [KHLZ19]

Try to formulate the problem as a $\min \max$ problem

min distance from predictions (colored points) - which depend on the inputs: steering angle, and gas pedal

maximize centerline projections (grey points) - which depend on the input: centerline velocity

State and input constraints

Inspired by [KHLZ19]

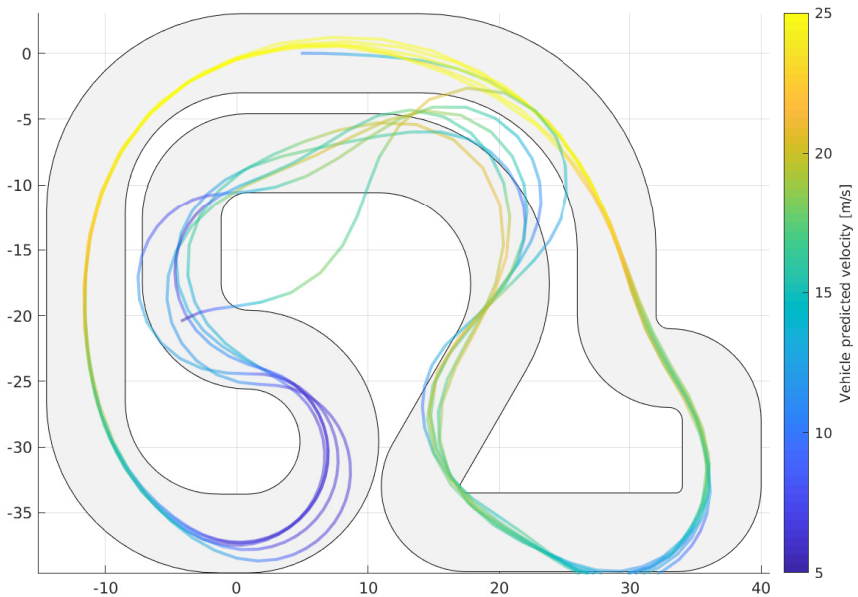
Try to formulate the problem as a $\min \max$ problem

min distance from predictions (colored points) - which depend on the inputs: steering angle, and gas pedal

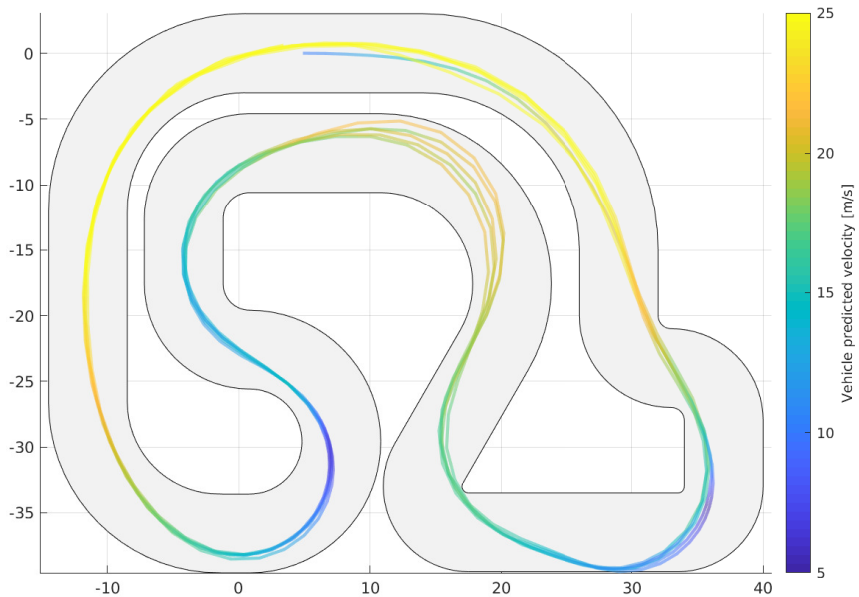
maximize centerline projections (grey points) - which depend on the input: centerline velocity

Results

MPC with unkown dynamics



Adaptive GP MPC



Outlook and Conclusion

- GP can introduce high nonlinearities in the prediction, making more difficult for the optimizer to a find a good local optimum.
- Replacing the inequality constraints by relaxed barrier functions increase significantly the computational performance, while always ensuring feasibility.
- Hyper-parameter optimization (GP training) plays an important role in the final controller performance

Bibliography I

Juraj Kabzan, Lukas Hewing, Alexander Liniger, and Melanie N. Zeilinger, *Learning-Based Model Predictive Control for Autonomous Racing*, IEEE Robotics and Automation Letters **4** (2019), no. 4, 3363–3370 (en).

Juraj Kabzan, Miguel de la Iglesia Valls, Victor Reijgwart, Hubertus Franciscus Cornelis Hendrikx, Claas Ehmke, Manish Prajapat, Andreas BÃ¼hler, Nikhil Gosala, Mehak Gupta, Ramya Sivanesan, Ankit Dhall, Eugenio Chisari, Napat Karnchanachari, Sonja Brits, Manuel Dangel, Inkyu Sa, Renaud DubÃ©, Abel Gawel, Mark Pfeiffer, Alexander Liniger, John Lygeros, and Roland Siegwart, *AMZ Driverless: The Full Autonomous Racing System*, arXiv:1905.05150 [cs] (2019) (en), arXiv: 1905.05150.

Carl Edward Rasmussen and Christopher K. I. Williams, *Gaussian processes for machine learning*, Adaptive computation and machine learning, MIT Press, Cambridge, Mass, 2006 (en), OCLC: ocm61285753.