Supplementary Material
- of -
Using Physics Knowledge for Learning Rigid-body Forward Dynamics with Gaussian Process Force Priors

Lucas Rath[*,1] **A. René Geist**[*,1] **Sebastian Trimpe**[1,2]

[1]Max Planck Institute for Intelligent Systems, Stuttgart
{geist, rath}@is.mpg.de

[2]Institute for Data Science in Mechanical Engineering, RWTH Aachen University
trimpe@dsme.rwth-aachen.de

1 Implicitly Constrained Dynamics

In this section, we provide further insight on the EOM of a robot arm in implicit ODE form. The Lagrangian forward dynamics (FD) of an *unconstrained* robot arm are given by

$$\ddot{q}' = M^{-1}Q, \tag{1}$$

with the generalized unconstrained acceleration $\ddot{q}'$, inertia matrix $M(q;\theta_\mathrm{A})$, and joint-related torques $Q(x;\theta_\mathrm{A})$. The term "unconstrained" refers in the context of a robot arm to its end-effector being not in contact with a surface. If the end-effector of the robot presses onto a surface, the surface impresses a force Q_I onto the robot changing its acceleration (cf. [1, p. 181] and [2]) to

$$\ddot{q} = M^{-1}\left(Q_b + P(Q + Q_z)\right) = M^{-1}Q_b + M^{-1}P(Q + Q_z), \tag{2}$$

with $Q_b = A^\mathsf{T}(AM^{-1}A^\mathsf{T})^+ b$, and $P = I - A^\mathsf{T}(AM^{-1}A^\mathsf{T})^+ AM^{-1}$. The terms A and b are obtained by differentiation of the holonomic implicit position-level constraint $c(q) = 0$ that is enforced by Q_I, reading

$$A\ddot{q} = b, \tag{3}$$

with $A(q,\dot{q};\theta_\mathrm{A}) : \mathbb{R}^{2n_q} \to \mathbb{R}^{m\times n_q}$, $b(q,\dot{q};\theta_\mathrm{A}) : \mathbb{R}^{2n_q} \to \mathbb{R}^m$, and $m < n_q$.

Remark 1.1 *In our implementation of a robot arm multi-body dynamics library in PyTorch, we formulated the implicit constrained EOM* (2) *as*

$$\ddot{q} = L\,b + TM^{-1}(Q + Q_z), \tag{4}$$

with $L = M^{-1}A^\mathsf{T}(AM^{-1}A^\mathsf{T})^+$ *and* $T = P^\mathsf{T} = I - LA = I - M^{-1}A^\mathsf{T}(AM^{-1}A^\mathsf{T})^+A$.

Remark 1.2 *Equation* (2) *denotes the minimum-norm solution to the problem*

$$\begin{aligned} \min \quad & \|\ddot{q} - \ddot{q}'\|_M^2, \\ \textit{subject to} \quad & A\,\ddot{q} = b. \end{aligned} \tag{5}$$

Remark 1.2 has been initially observed by Gauß [3] and was subsequently formulated for Lagrangian multi-body dynamics in terms of the Moore-Penrose pseudo inverse by Udwadia and Kalaba [2].

In the following, the null space of $A \in \mathbb{R}^{m\times n}$ is defined as $\mathsf{N}(A) = \{\ddot{q} \in \mathbb{R}^{n_q} : A\ddot{q} = 0\}$, its range space as $\mathsf{R}(A) = \{b \in \mathbb{R}^m : \exists\ddot{q} \in \mathbb{R}^{n_q} \text{ such that } b = A\ddot{q}\}$, and further we have that $\mathbb{R}^{n_q} = \mathsf{R}(A^\mathsf{T}) \oplus \mathsf{N}(A)$ and $\mathbb{R}^m = \mathsf{R}(A) \oplus \mathsf{N}(A^\mathsf{T})$ [4].

As a consequence of the D'Alembert-Lagrange principle the part of all forces doing virtual work, $Q' \in Q + Q_z$, must be in $Q' \in \mathsf{N}(A)$ while the implicit constraint forces Q_I doing no virtual work lie in $Q_I \in \mathsf{R}(A^\mathsf{T})$ (cf. [5]). Moreover, $Q_z \in \mathsf{N}(A)$.

[*]: equal contribution.

5th Conference on Robot Learning (CoRL 2021), London, UK.

Remark 1.3 *T and P are oblique projectors. That is these matrices are idempotent $T = T^2$ and $P = P^2$ but not symmetric $T \neq T^\mathsf{T}$ and $P \neq P^\mathsf{T}$. First, note that it is a projector because*

$$T^2 = I + M^{-1}A^\mathsf{T}(AM^{-1}A^\mathsf{T})^+AM^{-1}A^\mathsf{T}(AM^{-1}A^\mathsf{T})^+A - 2M^{-1}A^\mathsf{T}(AM^{-1}A^\mathsf{T})^+A, \quad (6)$$

$$= I - M^{-1}A^\mathsf{T}(AM^{-1}A^\mathsf{T})^+A = T, \quad (7)$$

and analogously also $P = P^2$. Second, it is straightforward to show that these matrices are not symmetric.

In the case of a surface constraint, A^T expressed in Cartesian coordinates corresponds to a vector that is perpendicular to the surface while $\mathsf{N}(A)$ denotes all vectors that are tangential to the surface.

Table 1: Summary of symbols related to modeling the system's implicitly constrained forward dynamics equations.

Symbol	**Description**
$\ddot{q} = M^{-1}(Q + Q_z + Q_I) = M^{-1}Q_b + H\bar{Q}$	Generalized acceleration of an implicitly constrained dynamical system
$H = M^{-1}\left(I_n - A^\mathsf{T}(AM^{-1}A^\mathsf{T})^+AM^{-1}\right)$	
M	Generalized mass matrix
$c(q)$	Implicit constraint equation
A, b	Matrices of $\ddot{c} = A\ddot{q} + b$
$\lambda = (AM^{-1}A^\mathsf{T})^+\left(b - AM^{-1}Q\right)$	Lagrangian multiplier of the implicit constraint forces
F_I	Cartesian force that is applied by the surface constraint onto the end-effector. F_I is always normal to the surface constraint.
F_z	Cartesian dissipative force acting on the end-effector while being tangential to the surface constraint.
Q_C, Q_G, Q_D, Q_u	Generalized bias, gravitational, dissipative and actuation forces
$Q = Q_D + Q_u + Q_G + Q_C$	Generalized forces acting on the unconstrained system
$Q_I = A^\mathsf{T}\lambda = A^\mathsf{T}(AM^{-1}A^\mathsf{T})^+\left(b - AM^{-1}Q\right)$	F_I being transformed into the generalized coordinate space.
Q_z	F_z being transformed into the generalized coordinate space.
$Q_b = A^\mathsf{T}(AM^{-1}A^\mathsf{T})^+b$	
$\bar{Q} = Q + Q_z = Q_K + Q_U$	
Q_K, Q_U	Known und unknown generalized forces of $\bar{Q}$

2 Multi-output Gaussian Processes

A Gaussian process defines a collection of random variables such that every finite set of these random variables are being normally distributed [6]. More concretely, we can model the FD $\boldsymbol{f} = [f_1 \dots f_n]^\mathsf{T}$ as a multi-task GP, writing $\hat{\boldsymbol{f}} \sim \mathcal{GP}(\boldsymbol{m}(\boldsymbol{x}), K(\boldsymbol{x}, \boldsymbol{x}'))$, where $\boldsymbol{m}(\boldsymbol{x})$ denotes a vector-valued mean function and $K(\boldsymbol{x}, \boldsymbol{x}')$ a matrix-valued kernel (covariance function), such that

$$\boldsymbol{m}(\boldsymbol{x}) = \begin{bmatrix} m_1(\boldsymbol{x}) & \dots & m_n(\boldsymbol{x}) \end{bmatrix}^\mathsf{T}, \quad K(\boldsymbol{x}, \boldsymbol{x}') = \begin{bmatrix} k_{1,1}(\boldsymbol{x}, \boldsymbol{x}') & \cdots & k_{1,n'}(\boldsymbol{x}, \boldsymbol{x}') \\ \vdots & \ddots & \vdots \\ k_{n,1}(\boldsymbol{x}, \boldsymbol{x}') & \cdots & k_{n,n'}(\boldsymbol{x}, \boldsymbol{x}') \end{bmatrix}, \quad (8)$$

where $m_j(\boldsymbol{x})$ is the scalar mean function of $f_j(\boldsymbol{x})$ and $k_{i,j}(\boldsymbol{x}, \boldsymbol{x}')$ corresponds to the scalar kernel between $f_i(\boldsymbol{x})$ and $f_j(\boldsymbol{x}')$. A multi-task GP defines a multivariate normal distribution on a given input data set X as $\hat{\boldsymbol{f}}(X) \sim \mathcal{N}(\boldsymbol{\mu}_X, \Sigma_{X,X})$, with the mean vector and covariance matrix respectively

$$\boldsymbol{\mu}_X = \begin{bmatrix} \boldsymbol{m}(\boldsymbol{x}_1)^\mathsf{T} & \dots & \boldsymbol{m}(\boldsymbol{x}_N)^\mathsf{T} \end{bmatrix}^\mathsf{T}, \quad \Sigma_{X,X^\star} = \begin{bmatrix} K(\boldsymbol{x}_1, \boldsymbol{x}'_1) & \cdots & K(\boldsymbol{x}_1, \boldsymbol{x}'_{N'}) \\ \vdots & \ddots & \vdots \\ K(\boldsymbol{x}_N, \boldsymbol{x}'_1) & \cdots & K(\boldsymbol{x}_N, \boldsymbol{x}'_{N'}) \end{bmatrix}. \tag{9}$$

For X, (**??**) reads $Y(X) = \hat{\boldsymbol{f}}(X) + \boldsymbol{\epsilon}_Y$, with $\boldsymbol{\epsilon}_Y \sim \mathcal{N}(\mathbf{0}\,, \Sigma_Y)$, with $\Sigma_Y = I_N \otimes \Sigma_y$. In turn, with the prediction locations $X' = [x_1'^\mathsf{T} \dots x_{N_\mathrm{p}}'^\mathsf{T}]^\mathsf{T}$, the joint distribution of $Y(X)$ and $\hat{\boldsymbol{f}}(X')$ reads

$$\begin{bmatrix} \hat{\boldsymbol{f}}(X^\star) \\ Y(X) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_{X^\star} \\ \boldsymbol{\mu}_X \end{bmatrix}, \begin{bmatrix} \Sigma_{X^\star,X^\star} & \Sigma_{X^\star,X} \\ \Sigma_{X,X^\star} & \Sigma_{X,X} + \Sigma_Y \end{bmatrix} \right). \tag{10}$$

Conditioning on the observations yields the predictive posterior distribution

$$\hat{\boldsymbol{f}}(X^\star)|\mathcal{D}, \Sigma_Y \sim \mathcal{N}(\boldsymbol{\mu}^\star, \Sigma^\star), \quad \begin{aligned} \boldsymbol{\mu}^\star &= \boldsymbol{\mu}_{X^\star} + \Sigma_{X^\star,X} \left(\Sigma_{X,X} + \Sigma_Y\right)^{-1} (\boldsymbol{y} - \boldsymbol{\mu}_X), \\ \Sigma^\star &= \Sigma_{X^\star,X^\star} - \Sigma_{X^\star,X} \left(\Sigma_{X,X} + \Sigma_Y\right)^{-1} \Sigma_{X,X^\star}. \end{aligned} \tag{11}$$

3 Linearly Transformed GPs and Implicitly Constrained Dynamics

In what follows, we use two useful properties of GPs.

Remark 3.1 *A GP $f \sim \mathcal{GP}(m(x), K(x, x'))$ is closed under a linear transformation B (cf. [6]), writing*

$$Bf \sim \mathcal{GP}\left(B(\boldsymbol{x})\boldsymbol{m}(\boldsymbol{x}),\ B(\boldsymbol{x})K(\boldsymbol{x}, \boldsymbol{x}')B^\mathsf{T}(\boldsymbol{x}')\right). \tag{12}$$

Remark 3.2 *The sum of GPs $f_i \sim \mathcal{GP}(m_i(x), K_i(x, x'))$ also yields a GP (cf. [7]), as*

$$f \sim \mathcal{GP}\left(\sum m_i(x),\ \sum K_i(x, x')\right). \tag{13}$$

Now, recall that the proposed framework places a GP prior on the forces of the FD EOM as

$$\hat{\bar{Q}} \sim \mathcal{GP}(m_{\bar{Q}}(x; \theta_\mathrm{A}),\ k_{\bar{Q}}(x, x'; \theta_\mathrm{M})), \tag{14}$$

in which $m_{\bar{Q}}(x; \theta_\mathrm{A}) \hat{=} Q_\mathrm{K}$ denotes an analytical model prior for $\bar{Q}$ while $k_{\bar{Q}}(x, x'; \theta_\mathrm{M})$ denotes an appropriately chosen kernel function. Then by inserting (14) into (2), a structured GP model is obtained as

$$\hat{\ddot{q}} \sim \mathcal{GP}(m_{\ddot{q}},\ k_{\ddot{q}}), \tag{15}$$

with $m_{\ddot{q}} = M^{-1}Q_b + HQ_\mathrm{K}$ and $k_{\ddot{q}} = Hk_{Q_\mathrm{U}}H^\mathsf{T}$. In the case of (15), $B \hat{=} H$ is a priori known transformation in terms of $\{A, b, M^{-1}\}$.

A considerable practical challenge forms the fast computation of the covariance matrix $\Sigma^{\ddot{q}}_{X,X'}$ over two input data sets $X = [x_1, x_2, ..., x_N]$ with $X \in \mathbb{R}^{n_{\ddot{q}} \times N}$ and $X' = [x'_1, x'_2, ..., x'_N]$ with $X' \in \mathbb{R}^{n_{\ddot{q}} \times N'}$. In our implementation of the structured GP, the line of code that computes $\Sigma^{\ddot{q}}_{X,X'}$ is found in the "SGPKernel" class and reads

```
# Mulitask covariance linear transformation --> T1 * Cov * T2^\transp
  cov_quant1_quant2 = torch.einsum('iab,ijbc,jdc->ijad',
      linearoperator1,
      covar_F_F,
      linearoperator2
  )
  return cov_quant1_quant2
```

Here, we used torch's "Einsum" function as it resulted in a fast computation of $\Sigma^{\ddot{q}}_{X,X'}$. Figure 1 illustrates how the above code computes the matrix $\Sigma^{\ddot{q}}_{X,X'}$. Here, $\Sigma^{\ddot{q}}_{X,X'}$ is stored as a four dimensional tensor of size $N \times N' \times n_{\ddot{q}} \times n_{\ddot{q}}$ which can be interpreted as a $N \times N'$ matrix of

$n_{\ddot{q}} \times n_{\ddot{q}}$ matrices that depend on the respective i-th and j-th input points. This matrix is computed using the tensors "linearoperator1" of size $N \times \mathtt{a} \times \mathtt{b}$ and "linearoperator2" of size $N' \times \mathtt{c} \times \mathtt{d}$ as well as the covariance matrix prior "covar_F_F" of size $N \times N' \times \mathtt{b} \times \mathtt{c}$. Note that in Figure 1, "linearoperator1" denotes the three-dimensional array $[[H(x_1)], [H(x_2)], ..., [H(x_N)]]$ as in this example we compute $\Sigma^{\ddot{q}}_{X,X'}$. However, the above computation of a structured covariance matrix allows to transform a force prior covariance matrix by *any* analytical tensor of appropriate size.

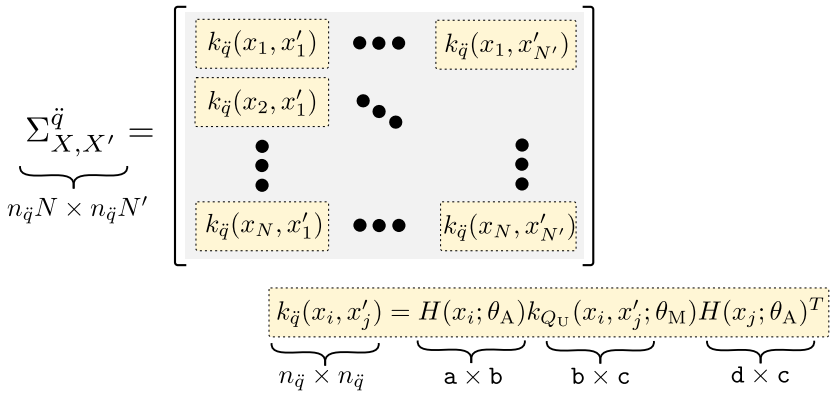


Figure 1: Illustration of the computation of the structured GP's covariance matrix $\Sigma^{\ddot{q}}_{X,X'}$.

Other unknown dynamical quantities can also be expressed as a linear transformation of known analytical functions. For example, the Lagrange multipliers are described analytically as

$$\lambda = (AM^{-1}A^{\mathsf{T}})^{+}\left(b - AM^{-1}Q\right), \tag{16}$$

and subsequently the surface normal force reads

$$Q_I = A^{\mathsf{T}}\lambda. \tag{17}$$

Also, the acceleration of the unconstrained system $\ddot{q}'$ is obtained as $\ddot{q}' = M^{-1}Q$.

Assuming Assumption 1 & 2 of the main manuscript apply to the robot, the unknown forces are potentially $Q_{\mathrm{U}} \hat{=} Q_z + Q$ which by using Remark 13 can be modelled via two separate GPs $\hat{Q} \sim \mathcal{GP}(m_Q(x), K_Q(x, x'))$ and $\hat{Q}_z \sim \mathcal{GP}(m_z(x), K_z(x, x'))$ as

$$\hat{\bar{Q}} \sim \mathcal{GP}(m_{\bar{Q}}(x), K_{\bar{Q}}(x, x')), \tag{18}$$

with $m_{\bar{Q}} = m_Q + m_z$ and $K_{\bar{Q}} = K_Q + K_z$. General nonlinear processes are rarely the result of the sum of two latent processes. Yet, for the case of rigid body mechanics it is an axiomatic truth that the sum of force functions results again in a force function.

Therefore, with (18), the joint distribution between $\ddot{q}$, $\ddot{q}'$, and λ reads

$$\begin{bmatrix} \hat{\lambda} \\ \hat{\ddot{q}}' \\ \hat{\ddot{q}} \end{bmatrix} \sim \mathcal{GP}\left(\begin{bmatrix} (AM^{-1}A^{\mathsf{T}})^{+}b - Lm_Q \\ M^{-1}m_Q \\ M^{-1}Q_b + Hm_{\bar{Q}} \end{bmatrix}, \begin{bmatrix} Lk_QL^{\mathsf{T}} & Lk_Q[M^{-1}]^{\mathsf{T}} & Lk_QH^{\mathsf{T}} \\ M^{-1}k_QL^{\mathsf{T}} & M^{-1}k_Q[M^{-1}]^{\mathsf{T}} & M^{-1}k_QH^{\mathsf{T}} \\ Hk_QL^{\mathsf{T}} & Hk_Q[M^{-1}]^{\mathsf{T}} & Hk_{\bar{Q}}H^{\mathsf{T}} \end{bmatrix}\right). \tag{19}$$

In addition, we can also denote the joint distribution between Q, Q_z, and $\ddot{q}$ as

$$\begin{bmatrix} \hat{Q} \\ \hat{Q}_z \\ \hat{\ddot{q}} \end{bmatrix} \sim \mathcal{GP}\left(\begin{bmatrix} m_Q \\ m_z \\ M^{-1}Q_b + Hm_{\bar{Q}} \end{bmatrix}, \begin{bmatrix} k_Q & 0 & k_QH^{\mathsf{T}} \\ 0 & k_z & k_zH^{\mathsf{T}} \\ Hk_Q & Hk_z & Hk_{\bar{Q}}H^{\mathsf{T}} \end{bmatrix}\right). \tag{20}$$

Albeit, the off-diagonal terms of the joint-distribution between Q and Q_z are zero and hence the processes are uncorrelated.

In the main manuscript we only resorted to the GP conditioning formula for predicting the *same* process at different input locations X and $X^\star$. In a similar fashion, if two GPs $f^{\mathrm{a}} \sim$

$\mathcal{GP}(m^{\mathrm{a}}(x), K^{\mathrm{aa}}(x,x'))$ and $f^{\mathrm{b}} \sim \mathcal{GP}(m^{\mathrm{b}}(x), K^{\mathrm{bb}}(x,x'))$ a correlated to each other by a joint distribution

$$\begin{bmatrix} f^{\mathrm{a}} \\ f^{\mathrm{b}} \end{bmatrix} \sim \mathcal{GP}\left(\begin{bmatrix} m^{\mathrm{a}}(x) \\ m^{\mathrm{b}}(x) \end{bmatrix}, \begin{bmatrix} K^{\mathrm{aa}}(x,x') & K^{\mathrm{ab}}(x,x') \\ K^{\mathrm{ba}}(x,x') & K^{\mathrm{bb}}(x,x') \end{bmatrix} \right), \tag{21}$$

one can predict $f^{\mathrm{b}}(X^\star)$ given measurements $Y = f^{\mathrm{a}}(X) + \epsilon_Y$ with $Y \sim \mathcal{N}(0, \Sigma_Y)$ as

$$f^{\mathrm{b}}(X^\star) \mid \mathcal{D}, \Sigma_Y \sim \mathcal{N}(\mu^{\mathrm{b},\star}, \Sigma^{\mathrm{b},\star}), \tag{22}$$

$$\mu^{\mathrm{b},\star} = \mu^{\mathrm{b}}_{X^\star} + \Sigma^{\mathrm{ba}}_{X^\star,X} \left(\Sigma^{\mathrm{aa}}_{X,X} + \Sigma_Y\right)^{-1} (Y - \mu^{\mathrm{a}}_X), \tag{23}$$

$$\Sigma^{\mathrm{b},\star} = \Sigma^{\mathrm{bb}}_{X^\star,X^\star} - \Sigma^{\mathrm{ba}}_{X^\star,X} \left(\Sigma^{\mathrm{aa}}_{X,X} + \Sigma_Y\right)^{-1} \Sigma^{\mathrm{ab}}_{X,X^\star}. \tag{24}$$

Remark 3.3 *From the joint distributions* (19) *and* (20) *it follows that with* (22) *one can use the proposed framework to predict* $\{\ddot{q}, \ddot{q}', \lambda, Q_I, Q_U, Q, Q_z\}$ *using measurements of* $\{\ddot{q}, \ddot{q}', \lambda, Q_I, Q_U, Q, Q_z\}$.

Also Remark 3.3 assumes that during optimization the two GPs $\hat{Q}$ and $\hat{Q}_z$ only learn the latent function Q or Q_z, respectively. If the hyper-parameter estimates of $\hat{Q}$ are strongly influenced by the errors arising from Q_z, predictions made using Remark 3.3 can be erroneous. It remains an open question how one can ensure that a force GP prior which models a specific force function is not influenced by the errors that arise from other forces. However, one can solve this problem also methodically by first estimating the hyper-parameters of $\hat{Q}$ on data of the unconstrained system, and afterwards, training the hyper-parameters of $\hat{Q}_z$ on data of the constrained system while keeping the hyper-parameters of $\hat{Q}$ fixed.

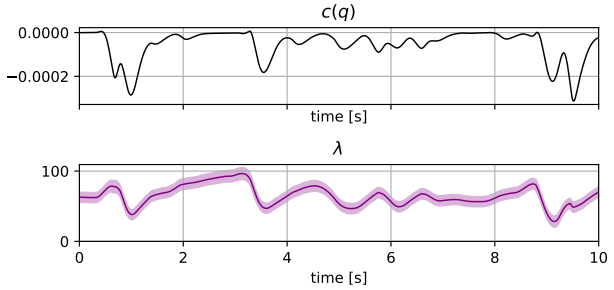


Figure 2: Top: Surface equation error in simulation. Bottom: Prediction of Lagrange multiplier λ by a structured GP model after conditioning on acceleration measurements.

For the sake of brevity, in the main manuscript, we modelled the forces inside the analytical model as a single GP such that the force prior (18) is set to $\mu_{Q_{\mathrm{U}}} \hat{=} Q_{\mathrm{K}}$ and $k_{Q_{\mathrm{U}}}$ denotes a diagonal matrix of squared-exponential kernels as in (31). With (19) and (22), one can predict the model's Lagrange multipliers as

$$\hat{\lambda}(X^\star) | \mathcal{D}, \theta = \mu^{\mathrm{b}}_{X^\star} + \Sigma^{\mathrm{ba}}_{X^\star,X} \left(\Sigma^{\mathrm{aa}}_{X,X} + \Sigma_Y\right)^{-1} (Y - \mu^{\mathrm{a}}_X), \tag{25}$$

with $\mu^{\mathrm{b}} = (AM^{-1}A^{\mathsf{T}})^{+}b - LQ_K$, $\mu^{\mathrm{a}} = M^{-1}Q_b + HQ_{\mathrm{K}}$, $\Sigma^{\mathrm{ba}} = L(X^\star)\Sigma^{\bar{Q}}_{X^\star,X}H^{\mathsf{T}}(X)$ and $\Sigma^{\mathrm{aa}}_{X,X} = H(x)k_{\bar{Q}}(X,X)H^{\mathsf{T}}(X)$. Figure 2 illustrates the prediction of λ by conditioning on $\mathcal{D}$ via (25) using the proposed structured GP model. As expected, the Lagrange multiplier remains positive, meaning that the end-effector is impressing a positive force on the surface, which is indeed what was observed during the simulation. The prediction of a loss of contact with structured GPs is left to future work. The estimation of the Lagrange multipliers as well as the surface normal force Q_I is useful for impedance control as commonly used in legged robotics.

4 Long-term Trajectory Predictions and Baumgarte Stabilization

Compared to differential-algebraic formulations of the EOM, the EOM (2) has the advantage that numerical ODE solvers can be used to compute a trajectory given initial conditions $\{q_0, \dot{q}_0\}$. Moreover, it is particularly straightforward to combine data-driven modeling with analytical equations.

Yet, (2) only respects the acceleration-level constraints explicitly. In turn, a trajectory prediction in the absence of measurement noise only lies on the position level-constraint $c(q) = 0$ if the initial state $\{q_0, \dot{q}_0\}$ lies on the position-level constraint. In practice, a numerical integration method inevitably makes errors on the system's acceleration which causes the predicted position and velocity to drift. This integration drift is additionally aggravated by the noise in the state observations.

The violation of position-level constraints by trajectory predictions forms a caveat of expressing constraint forces through implicit constraint equations as in (2) compared to the elimination of constraint forces from the EOM through a transformation to a minimal set of generalized coordinates as *e.g.*, in (1). Arguably, one could wonder why for a robotic arm it is a good choice to use implicit constraint equations combined with a set of redundant generalized coordinates instead of resorting to a minimum number of independent generalized coordinates. Such dubiety is alleviated, if we consider that through the use of implicit constraint equations the system's unconstrained dynamics (1) and constrained dynamics (2) only differ by the addition of the forces Q_I and Q_z while all other quantities such as q or M remain untouched. As pointed out in Section 3, as in both cases we use the same generalized coordinates, one can train force kernel priors both from data of the unconstrained as well as constrained system. To ensure that trajectory predictions converge to the position-level surface constraint one can resort to Baumgarte stabilization.

Baumgarte stabilization forms a common technique in multi-body dynamics to ensure that the trajectory predictions made with an explicit ODE form of EOM fulfill implicit position level-constraints. In what follows, we assume the implicit constraint equations are holonomic such that $\frac{\partial c}{\partial t} = A\dot{q} = 0$. Given a measured/predicted state-acceleration pair $\{q', \dot{q}', t, \ddot{q}'\}$, the error made in the position-, velocity-, and acceleration-level constraints amounts to

$$e_c = c(q'), \tag{26}$$

$$e_{\dot{c}} = A(q')\dot{q}', \tag{27}$$

$$e_{\ddot{c}} = A(q')\ddot{q}' - b(q', \dot{q}'). \tag{28}$$

In this case, Baumgarte [8] suggests to extend the error dynamics to yield a stable damped oscillator equation for e_c as

$$e_{\ddot{c}} + 2\omega_{n,\ddot{q}}\xi_{n,\ddot{q}}\dot{e}_c + \omega_{n,\ddot{q}}^2 e_c = 0, \tag{29}$$

with the natural frequency $\omega_{n,\ddot{q}} > 0$ and damping ratio $\xi_{n,\ddot{q}} > 0$. Usually $\xi_{n,\ddot{q}}$ is chosen to be unitary, such that the constraint error dynamics are critically damped. By inserting (26), (27), and (28) into (29) one obtains

$$A\ddot{q}' = \tilde{b}, \tag{30}$$

with $\tilde{b} = b + 2\omega_{n,\ddot{q}}\xi_{n,\ddot{q}}A\dot{q}' + \omega_{n,\ddot{q}}^2 e_{\ddot{q}} c(q')$.

Remark 4.1 *Substituting b in (2) with $\tilde{b}$ ensures that for long trajectory predictions, e_c converges to zero at the cost of introducing a small error to the predictions of $\ddot{q}$.*

5 Further Details on the Simulation

5.1 Robot Arm Simulation

For the robot arm simulation, we used PyBullet. In addition, we implemented a recursive dynamics library in PyTorch. The correctness of the PyBullet simulated dynamics and our PyTorch dynamics implementation is ensured by carefully comparing the simulated states, that is position, velocity, and acceleration. Then, instead of relying on the contact dynamics model provided by PyBullet, we compute the surface normal force Q_I using our PyTorch dynamics library which is then fed to the PyBullet simulation during data collection. The computation of Q_I is done using Baumgarté Stabilization as detailed in Section 4 to ensure that the end-effector does not leave the surface during the simulation due to small numerical errors. Moreover, we compute a viscous friction force F_z that is being applied to the end-effector, reading $F_z = -\theta_v \dot{p}_{\mathrm{E}}$ with the friction coefficient θ_v and the Cartesian end-effector velocity $\dot{p}_{\mathrm{E}}$. By use of our multibody library, Q_z is computed using F_z and subsequently integrated into the PyBullet simulation. The parameter of the friction function is set such that it significantly alters the motion of the robot arm. Without F_z, the analytical regression model matches the above discussed analytical simulation model. In turn, the effect of F_z on the robot dynamics can be seen in the error between the analytical regression model and the simulated acceleration as depicted in Figure 5a.

5.2 Data Collection

After the simulation and test environments are set up, running the experiment requires defining trajectory planning and control algorithms.

Trajectory planning While the robot is sliding its end-effector over the surface, we want to collect informative data for training different regression models. Therefore, we uniformly sample end-effector positions inside a rectangular region on the contact plane being inside the end-effector reachable space. We then coordinate the robot end-effector to sequentially connect the points following a straight-line trajectory, ensuring therefore that the end-effector remains always in contact with the surface. Between each pair of points, a task-space trapezoidal velocity profile is generated for the end-effector position. At each point, the arriving and starting velocities are zero. In addition, this profile allows setting a travel time and a maximum task-space acceleration between points which is used to avoid exceeding the actuator limits. The generated end-effector task-space position, velocity and acceleration trajectories are depicted in Figure 3. While the end-effector position-trajectory is given by the trapezoidal profile, the desired orientation is set to a constant such that the rounded tip remained perpendicular to the surface.

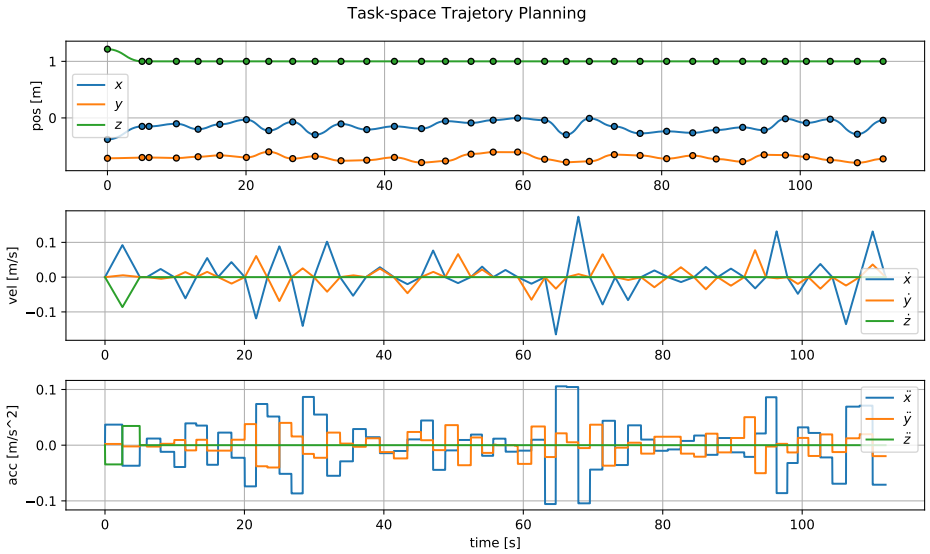


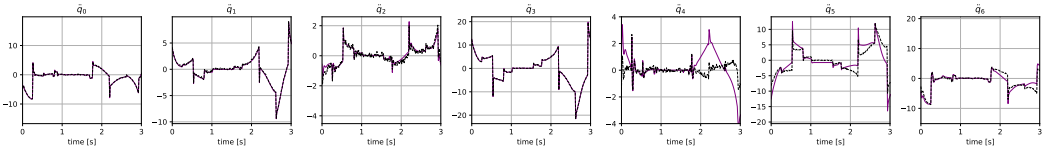
Figure 3: Task space trajectory of the robot arm's end-effector during data collection.

Task space inverse dynamics control and nullspace control To track the task-space reference trajectories with the robot arm, we use a task-space inverse dynamics controller [9, p. 347]. The task of controlling a manipulator consists of finding a time history of the generalized control forces Q_u, that are created by the actuators control input u.

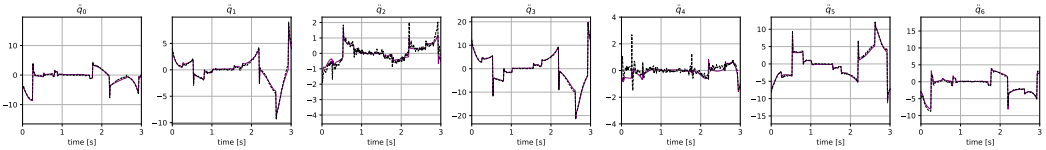
The robot arm has 7 degrees of freedom while we want to track a six dimensional end-effector trajectory (3D position and 3 Euler angles). In turn, we can achieve the same reference with more than one configuration by changing the arm's elbow elevation. The question arises of how to control this extra degree of freedom. In this work, we additionally use null-space control. The idea underlying nullspace control is to modify the desired joint-space acceleration, such that the error dynamics remains untouched, but introducing an acceleration in some subspace that achieves a secondary goal. This subspace is in our case the null-space of a Jacobian matrix that follows from the robot's differential kinematics.

5.3 Data Selection

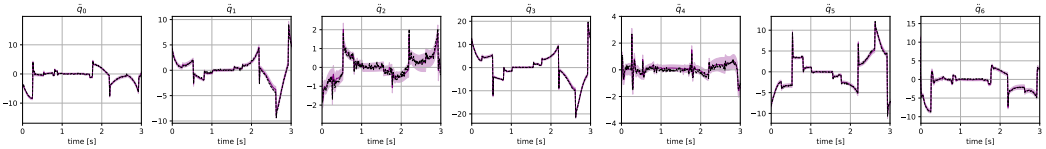
Sampling at 240 Hz generates quickly a huge amount of data points. Yet, during training, the computational complexity of GPs scales cubically with the number of data points which requires reducing the size of the initially large data set. Another problem arises when points in the training data set



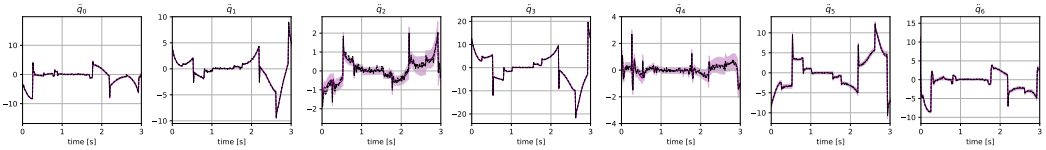
(a) Analytical Model



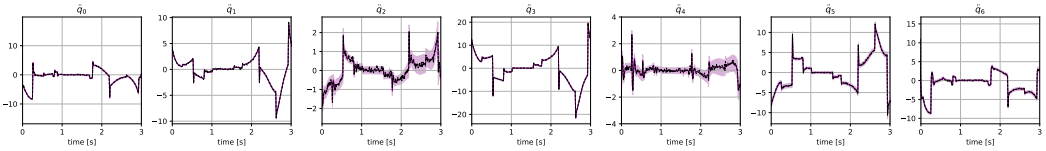
(b) Feed-forward NN



(c) GP



(d) S-GP with zero mean



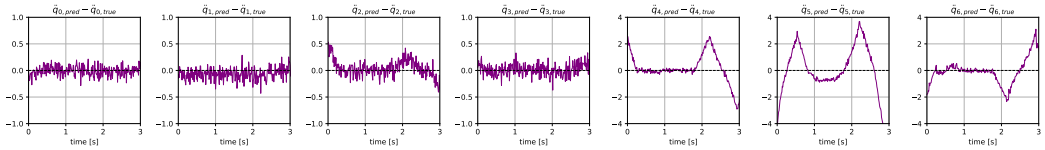
(e) S-GP with analytical mean

Figure 4: Acceleration predictions of different forward dynamics models. The black line depicts the system's noisy acceleration over time, the purple line the predicted mean and light purple – if available – the ± 2 std. confidence region. Note that the scaling of the y-axis for each model differs between the output dimensions.

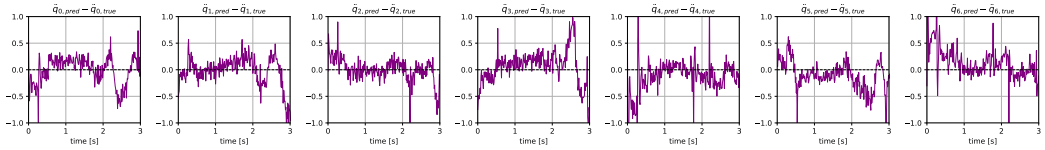
lie to close to each other inside the input space. This results in an almost singular covariance matrix potentially causing numerical problems during the computation of its Cholesky decomposition. A more detailed discussion on this issue is given in [10]. This problem can be avoided by selecting sufficiently distinct data set points in order to guarantee a numerically well-behaved covariance matrix. One practical way to select distinct data points is clustering or smart sampling techniques. In this work, we used a clustering algorithm to divide the initial large amount of training points in similar clusters and only keep a single point per cluster. However, the standard clustering techniques such as K-means clustering are computationally too slow for our data set size. Therefore, we used the Farthest Point Sampling algorithm (FPS) which has been used for Deep Learning with PointNet++ [11] for selecting a subset of relevant and informative points from a point cloud in a fast and efficient manner. The sampled points are chosen as the training dataset, while the removed points are left to the test dataset.

5.4 Assumptions on the Regression Models and Optimization Settings

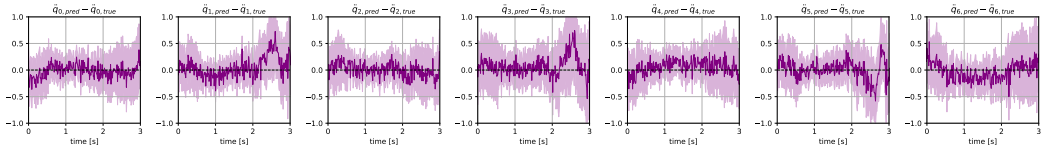
In the following, we discuss the regression models and corresponding optimization settings that are used for the experiments of the main manuscript as well as to create the additional figures being discussed in Section 6. All models use as input the full state of the robot, that is, seven joint angles, seven joint velocities, and seven control inputs in joint space coordinates and give as output the seven



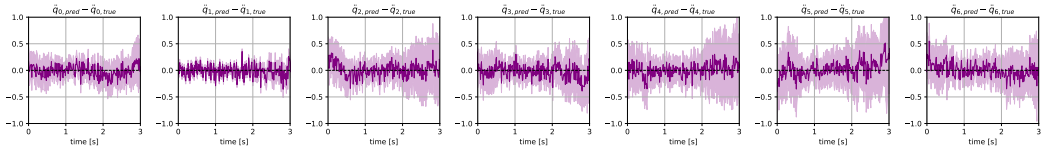
(a) Analytical Model



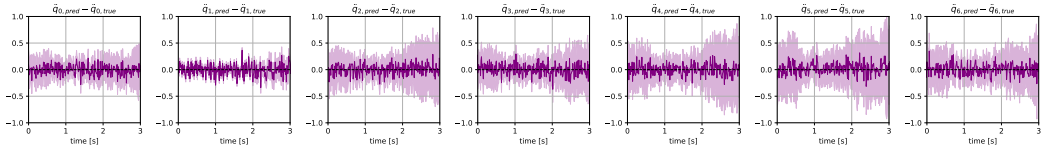
(b) Feed-forward NN



(c) GP



(d) S-GP with zero mean



(e) S-GP with analytical mean

Figure 5: Error in the acceleration prediction of different forward dynamics models. The black line depicts the system's noisy acceleration over time, the purple line the predicted mean and light purple – if available – the ± 2 std. confidence region. Note that the scaling of the vertical axis of the analytical model's plots differs from the other models' plots.

joint accelerations. Before training the regression models, Gaussian noise with a standard deviation of $\sigma_y = 0.1$ is being added to the acceleration observations.

Analytical regression model The analytical regression model is given by our PyTorch implementation of the EOM as denoted in (2). This model accurately depicts the simulated dynamics except that it does not include a model for the surface friction force Q_z. The analytical parameters are randomly initialized in a large parameter window set around the true parameter values and then trained on 10 thousand data point by minimization of a mean-squared loss function via automatic differentiation with an ADAM optimizer [12].

Neural network As a baseline, we trained a neural network on 10 thousand data point by minimization of a mean-squared loss function via automatic differentiation with an ADAM optimizer [12]. We trained three different neural networks consisting of a single hidden layer with 80 neurons, two hidden layers á 30 and 20 neurons, and four hidden layers á 200 neurons. Each hidden layer is combined with a sigmoid-activation layer. For the limited amount of data, we considered in this work we found that the single hidden layer neural network had the lowest mean absolute prediction error.

Gaussian process baseline As an additional baseline, each acceleration dimension is modeled by a one-dimensional zero-mean GP with squared exponential kernel. The input x to each kernel

consist of the robots joint angles, velocities, and control inputs such that with seven joints x is 21 dimensional. In turn, all seven GPs have in total 147 length-scale parameters (per output dimension one for each input) and 7 signal variances. In turn, θ_{M} consists of **154** hyper-parameters. The noise variance is set to $\sigma_y = 0.1$, and added to the likelihood function during optimization of the GP models.

Structured Gaussian processes In the structured GP model, the same GP as described in the above GP baseline is used as prior force model $\hat{Q}_{\text{U}}$. That is, the i-th dimension of the GP force prior $\hat{Q}_{\text{U}}$ is modeled through an independent squared exponential kernel $k_i^{\text{SE}}(x, x'; \theta_{\text{M}})$ such that

$$k_{Q_{\text{U}}} = \text{diag}(k_1^{\text{SE}}, k_2^{\text{SE}}, ..., k_{n_q}^{\text{SE}}). \tag{31}$$

In turn, $k_{Q_{\text{U}}}$ has **154** hyper-parameters. This GP force prior is then transformed by the analytical operator $H = M^{-1}P$ to yield a structured kernel on the acceleration level as

$$k_{\ddot{q}} = H k_{Q_{\text{U}}} H^{\mathsf{T}}. \tag{32}$$

In the experiments of the main manuscript as well as Section 6 we compared two different GP models. Here, "S-GP" denotes the GP $\hat{\ddot{q}} \sim \mathcal{GP}(0, k_{\ddot{q}})$ where the mean is set to zero and subsequently the GP must approximate all force functions. In comparison, "S-GP with analytical mean" denotes the GP $\hat{\ddot{q}} \sim \mathcal{GP}(M^{-1}Q_b + H(Q_C + Q_G + Q_u), k_{\ddot{q}})$ such that the difference between the data and the GP's mean is the end-effector friction force Q_z. In turn, "S-GP with analytical mean" uses additional analytical knowledge on the forces acting on the system such that the GP is only required to approximate the change in the acceleration due to Q_z.

Our framework allows optimizing either the GP hyper-parameters θ_{M}, the analytical model's parameters θ_{A}, or both simultaneously by using two independent ADAM [12] optimizer. During training, the noise variance is set to $\sigma_y = 0.1$, and added to the likelihood function. The starting learning rate of the optimizer is set to 0.2 for θ_{M} and 0.2 for θ_{A} and learning is performed until convergence.

6 Additional Simulation Results

Figure 4 shows different acceleration predictions of all seven robot arm joints over a three-second long trajectory made with the regression models discussed in Section 5.4. The same models were used as for the results in Figure 2b of the main manuscript. Figure 5 shows the error between simulated acceleration and the acceleration predicted by the models. The analytical model's predictions as shown in Figure 4a and 5a contain large errors especially in the last three joints that are close to the surface. This is not surprising as the unmodeled surface friction force Q_z affects the motion of the last joints especially. The NN with 10 thousand data points has a similar prediction accuracy as the GP baseline that has been trained on one thousand data points. The S-GP provides the smallest error as the GP directly approximates the unknown forces and is then transformed into acceleration-space using the analytical knowledge H. If in addition, an analytical mean function $\mu_{\ddot{q}=}M^{-1}Q_b + H(Q_C + Q_G + Q_u)$ is added to the S-GP the GP force prior must only approximate the unknown friction force Q_D. Subsequently, it can be observed that the prediction accuracy significantly improves through the inclusion of analytical knowledge into a data-driven regression algorithm.

References

[1] F. Udwadia and R. Kalaba. *Analytical dynamics: a new approach*. Cambridge University Press, 2007.

[2] F. E. Udwadia and R. E. Kalaba. On the foundations of analytical dynamics. *International Journal of non-linear mechanics*, 37(6):1079–1090, 2002.

[3] C. F. Gauß. Über ein neues allgemeines grundgesetz der mechanik. *Journal für die reine und angewandte Mathematik*, 4:232–235, 1829.

[4] R. W. Beard. Linear operator equations with applications in control and signal processing. *IEEE Control Systems Magazine*, 22(2):69–79, 2002.

[5] A. R. Geist and S. Trimpe. Structured learning of rigid-body dynamics: A survey and unified view from a robotics perspective. *GAMM Mitteilungen*, 44(2):34, 2021.

[6] C. K. Williams and C. E. Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

[7] D. Duvenaud. *Automatic model construction with Gaussian processes*. PhD thesis, University of Cambridge, 2014.

[8] J. Baumgarte. Stabilization of constraints and integrals of motion in dynamical systems. *Computer methods in applied mechanics and engineering*, 1(1):1–16, 1972.

[9] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo. *Robotics: modelling, planning and control*. Springer Science & Business Media, 2010.

[10] H. Mohammadi, R. L. Riche, N. Durrande, E. Touboul, and X. Bay. An analytic comparison of regularization methods for Gaussian Processes. *arXiv:1602.00853 [math, stat]*, May 2017. URL http://arxiv.org/abs/1602.00853. arXiv: 1602.00853.

[11] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 5105–5114, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

[12] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs]*, Dec. 2014. URL http://arxiv.org/abs/1412.6980. arXiv: 1412.6980.