



HAL
open science

Efficient Lexicographic Optimization for Prioritized Robot Control and Planning

Kai Pfeiffer, Abderrahmane Kheddar

► **To cite this version:**

Kai Pfeiffer, Abderrahmane Kheddar. Efficient Lexicographic Optimization for Prioritized Robot Control and Planning. 2024. lirmm-04645551

HAL Id: lirmm-04645551

<https://hal-lirmm.ccsd.cnrs.fr/lirmm-04645551>

Preprint submitted on 11 Jul 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Efficient Lexicographic Optimization for Prioritized Robot Control and Planning

Kai Pfeiffer¹ | Abderrahmane Kheddar^{2,3}

¹Schaeffler Hub for Advanced Research, School of Mechanical and Aerospace Engineering, Nanyang Technological University, Singapore

²Joint Robotics Laboratory (JRL) UMI3218/RL, CNRS/AIST, Tsukuba, Japan

³Interactive Digital Human, University of Montpellier, CNRS, LIRMM, UMR5506, Montpellier, France

Correspondence

Kai Pfeiffer.

Email: kaipfeifferrobotics@gmail.com

Abstract

In this work, we present several tools for efficient sequential hierarchical least-squares programming (S-HLSP) for lexicographical optimization tailored to robot control and planning. As its main step, S-HLSP relies on approximations of the original non-linear hierarchical least-squares programming (NL-HLSP) to a hierarchical least-squares programming (HLSP) by the hierarchical Newton's method or the hierarchical Gauss-Newton algorithm. We present a threshold adaptation strategy for appropriate switches between the two. This ensures optimality of infeasible constraints, promotes numerical stability when solving the HLSP's and enhances optimality of lower priority levels by avoiding regularized local minima. We introduce the solver $\mathcal{N}ADM_2$, an alternating direction method of multipliers for HLSP based on nullspace projections of active constraints. The required basis of nullspace of the active constraints is provided by a computationally efficient turnback algorithm for system dynamics discretized by the Euler method. It is based on an upper bound on the bandwidth of linearly independent column subsets within the linearized constraint matrices. Importantly, an expensive initial rank-revealing matrix factorization is unnecessary. We show how the high sparsity of the basis in the fully-actuated case can be preserved in the under-actuated case. $\mathcal{N}ADM_2$ consistently shows faster computations times than competing off-the-shelf solvers on NL-HLSP composed of test-functions and whole-body trajectory optimization for fully-actuated and under-actuated robotic systems. We demonstrate how the inherently lower accuracy solutions of the alternating direction method of multipliers can be used to warm-start the non-linear solver for efficient computation of high accuracy solutions to non-linear hierarchical least-squares programs.

KEYWORDS

optimisation, robots, nonlinear programming, hierarchical systems, discrete time systems, optimal control

1 | INTRODUCTION

1.1 | Context and contribution

Lexicographic multi-objective optimization (LMOO) is the hierarchical stacking of p optimization problems¹ (lexmin.: lexicographically minimize)

$$\begin{aligned} \text{lexmin.}_{x,v} \quad & \|v_{\mathbb{C}_1}\|_g, \dots, \|v_{\mathbb{C}_p}\|_g & (\text{LMOO}) \\ \text{s.t} \quad & f_{\mathbb{C}_{\cup p}}(x) \leq v_{\mathbb{C}_{\cup p}} \end{aligned}$$

The symbol \leq summarily describes equality and inequality constraints in the constraint set \mathbb{C} . The symbol \cup represents the union of constraint sets from levels 1 to p as $\mathbb{C}_{\cup p} := \mathbb{C}_1 \cup \dots \cup \mathbb{C}_p$. The function $f_{\mathbb{C}}(x) \in \mathbb{R}^{|\mathbb{C}|}$ in dependence of the variable vector $x \in \mathbb{R}^n$ represents the constraint set \mathbb{C} . Such problems are characterized by the optimal infeasibility (slacks v) $\|v_{\mathbb{C}_{\cup l-1}}^*\|_g > 0$ or optimality $v_{\mathbb{C}_{\cup l-1}}^* = 0$ of higher priority levels 1 to $l-1$, which must be preserved by the lower priority levels l to p . The infeasibility

Abbreviations: HLSP, hierarchical least-squares programming; NL-HLSP, non-linear hierarchical least-squares programming; S-HLSP, sequential hierarchical least-squares programming.

is thereby optimal / minimal with respect to some norm $g \geq 1$. Our above formulation of LMOO is a modification of the classical one as described in² to include (feasible and infeasible) inequality constraints. A specific form of LMOO is non-linear hierarchical least-squares programming (NL-HLSP) with $g = 2$. NL-HLSP's have been commonly utilized in instantaneous robot feed-forward³ and feed-back⁴ control. This enables an intuitive control formulation as no weights between different constraints need to be tuned. Furthermore, robot safety and physical stability is enhanced as critical constraints of different importance are strictly separated from control objectives like reaching tasks. Instantaneous prioritized robot control can be solved by a real-time and anytime algorithm modification of sequential hierarchical least-squares programming (S-HLSP) for NL-HLSP⁵. It utilizes the current hierarchical least-squares programming (HLSP) approximation of the original NL-HLSP to deduce a new robot control step. By utilizing trust-region constraints or regularization, the validity of the approximation is maintained at the current robot state. In the recent work⁶, S-HLSP has been leveraged for the resolution of NL-HLSP representing optimal control or trajectory optimization problems. Here, not only one instance, but a longer horizon of the robot control and state is considered. This enables robots to achieve a wide range of motions by reasoning in anticipating fashion about its physical and mechanical limits⁷. In this work, we propose a fast hierarchical least-squares programming (HLSP) solver based on the alternating direction method of multipliers (ADMM). The proposed HLSP solver is computationally more efficient than other solver methods when the number of iterations is limited. Sparse nullspace projections are leveraged to eliminate structured constraints like dynamics equations in optimal control scenarios. The nullspace basis is based on an efficient implementation of the turnback algorithm⁸ with an upper bound on the bandwidth in case of multiple-shooting transcription by Euler integration. At the same time, we are able to handle multi-stage constraints like regularization of momentum evolution for safe robot control. This is a distinguishing factor to recursive methods like differential dynamic programming (DDP,⁹). The efficiency of the proposed methods is evaluated on non-linear test-functions and robot trajectory planning.

1.2 | Non-linear programming

Non-linear programming (NLP) is a broad classification of optimization problems and includes non-linear and smooth convex and non-convex constraints and objectives¹⁰. It is a special form of LMOO with $p = 2$, typically feasible constraints $v_1 = 0$ and $|\mathbb{C}_2| = 1$ such that the norm notation can be omitted. Solution approaches typically involve the repeated approximation of the original non-linear program to a simpler one at the current working point. Inequality constraints are typically recast by including penalty terms in the cost function. The primal-dual interior-point method is characterized by a barrier functions with primal penalization towards the boundary of the feasible region¹¹. Exact penalty functions like non-smooth indicator functions have been investigated for example in the context of augmented Lagrangian methods. Here, infeasibilities are infinitely penalized outside of the primal feasible region, and not penalized otherwise¹². The augmented Lagrangian can for example be used within the ADMM. It consists of alternating updates of the primal and the dual¹³. Another solution approach can be found in sequential quadratic programming (SQP)¹⁴. The original non-linear optimality conditions are approximated to second order by Newton's method. The resulting quadratic program (QP) sub-problem is then iteratively solved for a primal and dual sub-step. In all methods above, convergence relies on globalization methods to direct the approximate sub-steps in terms of infeasibility reduction and optimality. Filter methods with trust region constraints are popular in SQP¹⁵. The trust-region constraint maintains validity of the QP sub-problems by limiting the step-size. On the other hand, line search methods directly curtail the resulting step in order to fulfill for example Armijo's condition¹⁶. Line search in combination with a filter method has been proposed for the interior-point method¹⁷.

1.3 | Prioritized robot control and planning

NL-HLSP's can exhibit sparsity patterns, for example resulting from discrete optimal control problems. We consider discretization by direct transcription methods, namely numerical integration by the Euler method. Due to stage-wise variable dependency of the constraints, the resulting optimality conditions exhibit block-diagonal structure. Exploiting this sparsity is critical in order to maintain linear complexity in the control horizon length¹⁸. Differential dynamic programming methods⁹ are a popular tool to leverage such sparse problem formulations. They are very efficient due to low bandwidth only dependent on the number of the input controls. Originally developed for unconstrained systems, in recent years many developments have been proposed for constrained ones. These include interior-point method¹⁹ and augmented Lagrangian²⁰ based approaches. Prioritized trajectory optimization has been treated for example in²¹ which solves a hierarchy of quadratic programs each projected into the nullspace

of the previous level. Sparsity of the constraints is exploited by leveraging DDP. The approach in²² explicitly considers the active constraints in order to preserve the hierarchical ordering. Using principles from time-delay systems, linear and robust controls result from quadratic program solutions at each instance of a model-predictive controller. Both approaches can only handle input and state limit constraints. This is in contrast to LMOO as proposed in²³, which is based on prioritized Pareto efficiency. However, sparsity is not exploited. In this work, we propose a sparse solver for prioritized trajectory optimization cast as NL-HLSP under multi-stage equality and inequality constraints. Constraints can involve variables from several stages, for example regularization of the momentum evolution for safe robot control. Such problems can be solved by off-the-shelf sparse non-linear solvers¹⁷. However, higher efficiency can be achieved with more dedicated solvers like the aforementioned prioritized solvers based on the reduced Hessian formulation⁶. Here, the right choice of nullspace basis reduces the number of variables (dense programming) or non-zeros (sparse programming). This offsets the computational burden of computing a basis of the nullspace. Several sparsity preserving bases of nullspace have been proposed both in the communities of structural mechanics and control theory. The authors in²⁴ described a sparsity preserving basis based on identifying linearly independent sub-sets. These arise due to the limited bandwidth of the blocks. Several improvements have been proposed. The work in²⁵ proposes sparsity enhancing improvements. A more efficient computation leveraging columns updates of the linearly independent sub-matrices is described in⁶. A control theoretic approach has been developed for example in²⁶.

1.4 | Overview

This article is structured as follows. First, we describe the current state-of-the-art and our contributions in non-linear hierarchical least-squares programming and prioritized non-linear optimal control (Sec. 2). We then introduce a heuristic for adjusting the threshold for second-order information (Sec. 3). This promotes numerical stability when solving the HLSP sub-problems and avoids local regularized minima of lower priority levels. We develop the HLSP solver $\mathcal{N}ADM_2$ based on the ADMM, see Sec. 4. We present how to tune parameters of the ADMM for algorithmic efficiency (Sec. 4.2) and make some considerations towards warm-starting (Sec. 4.3) and the computation of dual variables (Sec. 4.4). Next, we design an efficient implementation of the turnback algorithm for the computation of nullspace basis of banded matrices. Specifically, we consider dynamics discretized by Euler integration and derive an upper bound on the bandwidth of the resulting nullspace basis (Sec. 5). We show that the algorithm can be highly parallelized, which is a distinguishing factor in comparison to recursive methods like the DDP (Sec. 5.7).

NOMENCLATURE

l	Current priority level
p	Overall number of priority levels, excluding the trust region constraint on $l = 0$
n	Number of variables
r	Rank of matrix
n_r	Number of remaining variables after nullspace projections
m	Number of constraints
$x \in \mathbb{R}^n$	Primal of NL-HLSP
$\Delta x \in \mathbb{R}^n$	Primal of HLSP
$\Delta z \in \mathbb{R}^{n_r}$	Primal of projected HLSP
$\Delta \hat{z} \in \mathbb{R}^{n_r}$	Auxiliary primal of projected HLSP
$f(x) \in \mathbb{R}^m$	Non-linear constraint function of variable vector $x \in \mathbb{R}^n$
\mathbb{E}_l	Set of $m_{\mathbb{E}}$ equality constraints (eq.) of level l
\mathbb{I}_l	Set of $m_{\mathbb{I}}$ inequality constraints (ineq.) of level l
\mathcal{I}_l	Set of $m_{\mathcal{I}}$ inactive inequality constraints (ineq.) of level l
\mathcal{A}_l	Set of $m_{\mathcal{A}}$ active equality and inequality constraints of level l
$\mathbb{E}_{\cup l}$ (or $\mathcal{E}_{\cup l}$)	Set union $\mathbb{E}_{\cup l} := \bigcup_{i=1}^l \mathbb{E}_i = \mathbb{E}_1 \cup \dots \cup \mathbb{E}_l$ with $m_{\mathbb{E}_{\cup l}}$ constraints
$A_{\mathbb{E}} \in \mathbb{R}^{m_{\mathbb{E}} \times n}$	Matrix representing a set \mathbb{E} of $m_{\mathbb{E}}$ linear constraints
$b_{\mathbb{E}} \in \mathbb{R}^{m_{\mathbb{E}}}$	Vector representing a set \mathbb{E} of $m_{\mathbb{E}}$ linear constraints
$\mathcal{N}(A_{\mathcal{A}_l})$	Operator to compute the nullspace basis $Z_{\mathcal{A}_l}$ and the rank r of a matrix $A_{\mathcal{A}_l}$
$Z_{\mathcal{A}_l} \in \mathbb{R}^{n \times n_r}$	Nullspace basis of matrix $A_{\mathcal{A}_l} \in \mathbb{R}^{m_{\mathcal{A}_l} \times n}$ with rank r , $n_r = n - r$ and $A_{\mathcal{A}_l} Z_{\mathcal{A}_l} = 0$

$N_{l-1} \in \mathbb{R}^{n \times n_r}$	Accumulated nullspace basis $N_{\mathcal{A}_{\cup l}} = Z_{\mathcal{A}_1} \dots Z_{\mathcal{A}_l}$
$\tilde{M} \in \mathbb{R}^{m \times n_r}$	Matrix $\tilde{M} = MN$ projected into the nullspace basis $N \in \mathbb{R}^{n \times n_r}$ of a matrix $A \in \mathbb{R}^{m \times n}$ of rank r ; $n_r = n - r$
	(variable elimination)
$v \in \mathbb{R}^m$	Slack variable of NL-HLSP
$\hat{v} \in \mathbb{R}^m$	HLSP equivalent of slack variable
$v^* \in \mathbb{R}^m$	Optimal slack variable
\mathcal{L}	Lagrangian
K	Gradient of Lagrangian $K := \nabla \mathcal{L}$
H_l, \hat{H}_l	Hierarchical Lagrangian Hessian, positive definite equivalent
ρ, σ	ADMM step-size parameter
$\lambda \in \mathbb{R}^m$	Lagrange multiplier
v	Scaled Lagrange multiplier
k	Outer iteration of NL-HLSP solver
ι	Inner iteration of HLSP solver
ρ	Trust region radius
ν	Activation threshold of inequality constraints
χ	Convergence threshold of S-HLSP
$\epsilon_{adaptive,l}$	Adaptive second-order information (SOI) threshold of level l

2 | PROBLEM DEFINITION AND CONTRIBUTIONS

2.1 | Non-linear Hierarchical Least-Squares Programming

In this article, we consider non-linear hierarchical least-squares problems (NL-HLSP) as preemptive transcription²⁸ of LMOO with $g = 2$:

$$\begin{aligned}
 \min_{x, v_{\mathbb{E}_l}, v_{\mathbb{I}_l}} \quad & \frac{1}{2} \|v_{\mathbb{E}_l}\|_2^2 + \frac{1}{2} \|v_{\mathbb{I}_l}\|_2^2 & l = 1, \dots, p \\
 \text{s.t.} \quad & f_{\mathbb{E}_l}(x) = v_{\mathbb{E}_l} \\
 & f_{\mathbb{I}_l}(x) \leq v_{\mathbb{I}_l} \\
 & f_{\mathcal{A}_{\cup l-1}}(x) = v_{\mathcal{A}_{\cup l-1}}^* \\
 & f_{\mathcal{I}_{\cup l-1}}(x) \leq 0
 \end{aligned} \tag{NL-HLSP}$$

Each problem corresponding to the levels $l = 1, \dots, p$ is solved in order. At convergence of each level l at the primal $x_l^* \in \mathbb{R}^n$, the feasible $v_l^* \in \mathbb{R}^{m_l} = 0$ or optimally infeasible points $v_l^* \neq 0$ of the sets of equality and inequality constraints $\mathbb{E}_l = m_{\mathbb{E}_l}$ and $\mathbb{I}_l = m_{\mathbb{I}_l}$ is identified. The slack variables $v_{\mathcal{A}_{\cup l-1}}^*$ are the optimal ones identified for the higher priority levels 1 to $l-1$. The *active set* $\mathcal{A}_{\cup l-1}$ contains all constraints that are active at convergence of levels 1 to $l-1$. The active set includes all equality constraints $\mathbb{E}_{\cup l-1}$, and furthermore all violated / infeasible ($v^* > 0$) or saturated ($v^* = 0$) inequality constraints of \mathbb{I}_l . In a similar vein, the *inactive set* $\mathcal{I}_{\cup l-1}$ contains all the remaining feasible inequality constraints ($v^* = 0$) of the sets $\mathbb{I}_{\cup l-1}$.

Contribution: we make some considerations towards resolving limitations of NL-HLSP's. Specifically, we demonstrate how we can identify local minima associated with negative function values, see Sec. 6.2.

2.2 | Sequential Hierarchical Least-Squares Programming

Sequential hierarchical least-squares programming (S-HLSP) is a method to resolve NL-HLSP. Here, the NL-HLSP is linearized around the current working point x_k at every *outer* iteration k to a HLSP by virtue of the hierarchical Newton's method⁵.

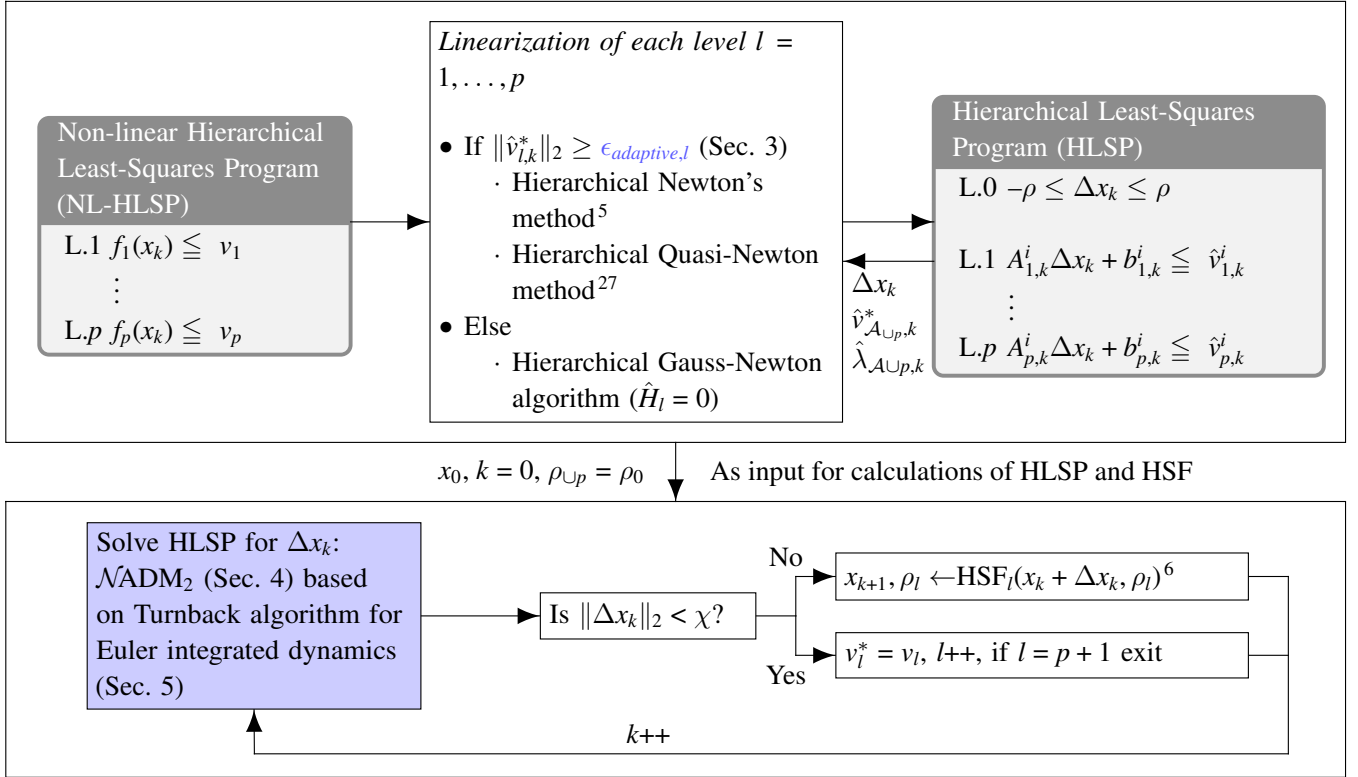


FIGURE 1 A symbolic overview of the sequential hierarchical least-squares programming (S-HLSP) with trust region and hierarchical step-filter (HSF) based on the SQP step-filter¹⁵ to solve non-linear hierarchical least-squares programming (NL-HLSP) with p levels. Our contributions, an adaptive threshold for second-order information and the HLSP sub-problem solver \mathcal{NADM}_2 in combination with an efficient turnback algorithm for Euler integrated dynamics, are marked in blue.

Hierarchical least-squares programs (HLSP) are problems of the form

$$\begin{aligned}
 \min_{\Delta x, \hat{v}_{\mathbb{E}_l}, \hat{v}_{\mathbb{I}_l}} \quad & \frac{1}{2} \|\hat{v}_{\mathbb{E}_l}\|_2^2 + \frac{1}{2} \|\hat{v}_{\mathbb{I}_l}\|_2^2 \quad l = 1, \dots, p \\
 \text{s.t.} \quad & A_{\mathbb{E}_l} \Delta x - b_{\mathbb{E}_l} = \hat{v}_{\mathbb{E}_l} \\
 & A_{\mathbb{I}_l} \Delta x - b_{\mathbb{I}_l} \leq \hat{v}_{\mathbb{I}_l} \\
 & A_{\mathcal{A}_{l-1}} \Delta x - b_{\mathcal{A}_{l-1}} = \hat{v}_{\mathcal{A}_{l-1}}^* \\
 & A_{\mathcal{I}_{l-1}} \Delta x - b_{\mathcal{I}_{l-1}} \leq 0
 \end{aligned} \tag{HLSP}$$

Variables $\hat{\cdot}$ are the linear equivalents to the non-linear ones of the NL-HLSP. Notably, the problem constraints are linear. The constraint matrices and vectors A and b represent this linearization (Jacobians and Hessians) of non-linear constraints f .

Utilizing Fletcher's filter method, the resulting primal steps from the HLSP sub-problems are accepted or rejected, depending on sufficient progress in terms of constraint infeasibility reduction and optimality. The HLSP sub-problems are subject to a trust-region constraint in order to maintain the validity of the approximation. The trust-region radius is increased or decreased depending on step acceptance and rejection, respectively. Linearization methods of the NL-HLSP to HLSP include the hierarchical Newton's method (using second order information (SOI) in form of the hierarchical Hessian) or the hierarchical Gauss-Newton algorithm (no SOI)⁵. A switch between the two is decided upon the residual of the HLSP sub-problem.

Contribution: An overview of S-HLSP is given in Fig. 1. In this work, we propose an adaptive thresholding strategy for SOI augmentation in the hierarchical Newton's method (Sec. 3). This promotes numerical stability when solving the HLSP sub-problems and enhances solution optimality of lower priority levels. Furthermore, an efficient solver for HLSP based on the ADMM is presented (Sec. 4). As a first-order method, the solver primarily relies on matrix-vector multiplications instead of

matrix factorizations as for the IPM. This solver is efficient in approximating a solution of moderate accuracy with a limited number of iterations with respect to its IPM equivalent. The approximate primal guess then can be used to warm-start a high accuracy solver as we demonstrate in Sec. 6.2.

2.3 | Prioritized trajectory optimization

A specific form of NL-HLSP's are prioritized non-linear trajectory optimization problems of the form

$$\begin{aligned} \min_{x_T, v_{\cup l, t}} \quad & \frac{1}{2} \|v_{\cup l, t}\|_2^2 \quad l = 1, \dots, p \\ \text{s.t.} \quad & f_l(x_{s_t:e_t}) \leq v_{l,t} \quad t = 0, \dots, T \\ & f_{\cup l-1}(x_{s_t:e_t}) \leq v_{\cup l-1, t}^* \end{aligned} \quad (\text{PTO})$$

Here, $f_{\cup l-1}$ represents equality and inequality constraints of lower priority levels 1 to $l-1$, which is indicated by the symbol \leq (note that in the case of inactive inequality constraints $\mathcal{I}_{\cup l-1}$, we have $v_{\cup l-1, t}^* = 0$). T is the length of the control horizon. The individual time steps $t = 0, \dots, T$ are also referred to as stages. Constraints only depend on specific variable segments / intervals $x_{s_t:e_t} := x[s_{t_0} : e_{t_1}]$. The indices $s_{t_0} \leq e_{t_1}$ with $t_0 \leq t_1$ are the start and end indices of the segments in x corresponding to time steps t_0 and t_1 . The constraint Jacobians J therefore exhibit a banded structure as follows

$$J = \begin{bmatrix} \nabla_{x_{[0]}} f(x_{s_0:e_1}) & \nabla_{x_{[1]}} f(x_{s_0:e_1}) & \cdots & 0 \\ 0 & \nabla_{x_{[1]}} f(x_{s_1:e_2}) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \nabla_{x_{[T]}} f(x_{s_{T-1}:e_T}) \end{bmatrix} \quad (1)$$

$[t]$ indicates the interval $[s_t : e_t]$. This optimal control problem structure handles multi-stage constraints (excluding the dynamics), unlike DDP. Typically, we have some initial condition on parts of the variable vector $x: = x_{:,0}$ with constant $x_{:,0}$, which can be seamlessly integrated as a high priority constraint.

Contribution: In this work, we present a sparse nullspace basis based on the turnback algorithm for Euler integrated dynamics (Sec. 5). We provide an upper bound on its bandwidth. This enables us to design an efficient turnback algorithm which does not rely on a costly initial rank-revealing matrix factorization. We demonstrate how the high degree of sparsity in the case of full actuation can be transferred to the case of under-actuation

3 | HIERARCHICAL STEP-FILTER WITH ADAPTIVE THRESHOLD FOR SECOND ORDER INFORMATION

S-HLSP utilizes the hierarchical Newton's method⁵ (or the Quasi-Newton equivalent²⁷) or the hierarchical Gauss-Newton algorithm to linearize NL-HLSP. Switching between the two can be based on the principle that at convergence, variables corresponding to infeasible constraints need to be 'locked' in the HLSP by a full rank Hessian in order to not disturb the optimal infeasibility of the non-linear constraints. Furthermore, in robotics constraint Jacobians in the HLSP are typically rank deficient at infeasible points due to kinematic and algorithmic singularities²⁹. The Newton's method and its second-order information (SOI) then acts as a regularization and enables a global solution to the HLSP. At the same time, deactivating SOI promotes solution optimality. SOI is typically full-rank on the variables that the corresponding constraints occupy. Therefore, these variables can not be used any more for the resolution of lower priority levels. If SOI is unnecessarily activated for feasible constraints, this results in less optimal local minima for constraints on lower priority levels.

The switching method proposed in²⁷ adheres to the following strategy

$$\begin{aligned} H_l &= J^T J + \text{SOI}_l \quad \text{if } \|\hat{v}_l\|_2 \geq \epsilon \text{ (Newton's method)} \\ H_l &= J_l^T J_l \quad \text{otherwise (Gauss-Newton algorithm)} \end{aligned} \quad (2)$$

SOI is defined as

$$\text{SOI}_l := \hat{H}_l = R_l^T R_l \quad (3)$$

\hat{H}_l represents some positive definite regularization (for example Higham³⁰ or symmetric Schur regularization³¹, Broyden-Fletcher-Goldfarb-Shanno algorithm (BFGS)³², weighted identity matrices³³, ...) of the hierarchical Lagrangian Hessian of a level l ⁵. It involves SOI and approximate Lagrange multipliers of the levels 1 to l . ϵ is a constant threshold on the linear slacks \hat{v} as an indicator for constraint infeasibility. The linear slacks \hat{v} capture well-posed / compatible HLSP sub-problems and enable SOI deactivation even if an iterate $x^k \neq x^*$. In contrast, the non-linear slack v_l of feasible constraints only vanishes at convergence x^* . Here, we propose an adaptive strategy for the SOI augmentation thresholds $\epsilon_{adaptive,l}$ of each level l in (2), see Sec. 3.2. This avoids manual tuning of the SOI activation threshold which is oftentimes necessary for HLSP sub-problem solvers of different accuracy and in dependency of the problem configurations. The method is based on the HSF for S-HLSP globalization, which is recalled in Sec. 3.1.

3.1 | The hierarchical step-filter

The HSF⁶ based on the SQP step-filter¹⁵ measures the progress in the approximated HLSP sub-problems with respect to the original NL-HLSP. Each filter \mathcal{F}_l of the levels $l = 1, \dots, p$ of the NL-HLSP consists of pairs $(h_{\cup l-1}, \|f_l^+\|_2^2)$ with

$$h_{\cup l-1}(x_k + \Delta x_k) = \|f_{\mathbb{I}_{\cup l-1}}^+ - v_{\mathbb{I}_{\cup l-1}}^*\|_1 + \|f_{\mathbb{E}_{\cup l-1}}^+ - v_{\mathbb{E}_{\cup l-1}}^*\|_1 \quad (4)$$

Here

$$f_l^+ := \begin{bmatrix} f_{\mathbb{E}_l} \\ \max(0, f_{\mathbb{I}_l}) \end{bmatrix} \quad (5)$$

$h_{\cup l-1}$ reflects feasibility of the constraints while $\|f_l^+(x_k + \Delta x_k)\|_2^2$ indicates objective optimality. As can be seen, we use the non-linear slacks $v_l = f_l^+(x_k + \Delta x_k)$ instead of the linear ones \hat{v}_l from the HLSP.

A new point $h_{\cup l-1}(x_k + \Delta x_k)$ and $\|f_l^+(x_k + \Delta x_k)\|_2^2$ resulting from a new primal step Δx_k of the HLSP sub-problem is acceptable to all filter points $j \in \mathcal{F}_l$ if sufficient progress in feasibility or optimality has been achieved:

$$h_{\cup l-1} \leq \beta h_{\cup l-1}^j \quad \text{or} \quad \|f_l^+\|_2^2 + \gamma h_{\cup l-1} \leq \|f_l^{*j}\|_2^2 \quad (6)$$

β is a value close to 1 and γ is a value close to zero and adhere to the condition $0 < \gamma < \beta < 1$. Since the model reliably represents the non-linear problem, the trust region radius is increased. Otherwise, the step is rejected and the trust region radius is reduced. The HSF of level l converges once $\|\Delta x_k\|_2 < \chi$ falls below the threshold χ . This process is repeated for each priority level $l = 1, \dots, p$.

3.2 | Adaptive SOI thresholding

The threshold adaptation strategy for $\epsilon_{adaptive,l}$ is outlined in Alg. 1. On each level $l = 1, \dots, p$ and at every outer iteration, the filter front $(h_{\cup l-1,front}, \|f_l^+\|_{2,front}^2)$ is updated by $\text{adaptEps}(\epsilon_{adaptive,l}, (h_{\cup l-1,front}, \|f_l^+\|_{2,front}^2), (h_{\cup l-1}, \|f_l^+\|_2^2), c_l, \text{accepted}_l, \kappa, \underline{\epsilon}, \bar{\epsilon})$. It represents the most optimal point by the margin δ that has been encountered so far on a level l .

Definition 1 (Filter front). A point $(h_{\cup l-1,front}, \|f_l^+\|_{2,front}^2)$ is the front of a filter \mathcal{F}_l to degree $\delta \leq 1$, if it dominates all points of previous iterates x_j with $j = 1, \dots, k$ according to

$$h_{\cup l-1,front} \leq h_{\cup l-1}^j \quad \text{and} \quad \|f_l^+\|_{2,front}^2 < \delta \|f_l^{*j}\|_2^2 \quad (7)$$

An exemplary visualization is given in Fig. 2. The choice $j = 1, \dots, k$ (and not considering the filter elements $j \in \mathcal{F}_l$) is motivated by the fact that a filter \mathcal{F}_l does not include all iterates x_j with $j = 0, \dots, k$ due to a concept referred to as f-type iteration where the focus is put on optimality of the HSF level l , see¹⁵. Since this comes possibly at the cost of increase in constraint violation $h_{\cup l-1}$, the threshold adaptation takes place on all levels including the current HSF level l . We reinitialize the filter front of each level $i = 1, \dots, p$ at the start of the step filter of a level l by $(h_{\cup l-1}(x_{l-1}^*), \|f_l^+(x_{l-1}^*)\|_2^2)$. x_{l-1}^* is the primal obtained at the KKT point of the previous level $l-1$. This handles cases where x_0 is a feasible ($f_l^+ = 0$), but x_{l-1}^* is an infeasible point ($f_l^+ \neq 0$) to constraints, since otherwise no other filter front can be identified (since the condition $\|f_l^+\|_2^2 < \|f_l^+\|_{2,front}^2 = 0$ would need to be fulfilled).

Algorithm 1 adaptEps

Input: $\epsilon_{adaptive}$, $(h_{front}, \|f^+\|_{2,front}^2)$, $(h, \|f^+\|_2^2)$, c , *accepted*, κ , $\underline{\epsilon}$, $\bar{\epsilon}$

Output: $\epsilon_{adaptive}$, $(h_{front}, \|f^+\|_{2,front}^2)$, c

- 1: **if** *accepted* **then**
- 2: **if** $h \leq h_{front}$ & $\|f^+\|_2^2 < \delta \|f^+\|_{2,front}^2$ **then**
- 3: $\epsilon_{adaptive} \leftarrow \min(\epsilon_{adaptive} \cdot \kappa, \bar{\epsilon})$
- 4: $h_{front} = h$
- 5: $\|f^+\|_{2,front}^2 = \|f^+\|_2^2$
- 6: $c = 0$
- 7: **end if**
- 8: **else if** $c > \zeta$ **then**
- 9: $\epsilon_{adaptive} \leftarrow \max(\epsilon_{adaptive}/\kappa, \underline{\epsilon})$
- 10: **end if**
- 11: $c \leftarrow c + 1$
- 12: **return** $\epsilon_{adaptive}$, $(h_{front}, \|f^+\|_{2,front}^2)$, c

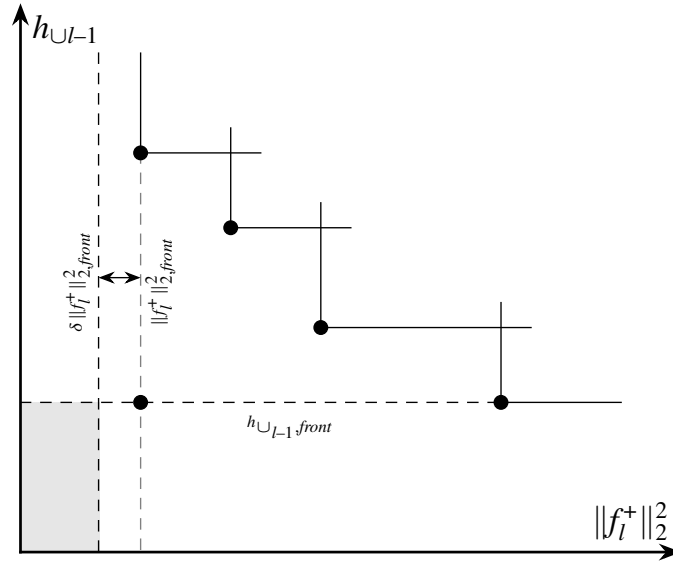


FIGURE 2 Iterates $(h_{\cup l-1}(x_k), \|f_l^+(x_k)\|_2^2)$ of level l . A new filter front needs to lie within the shaded area.

The SOI augmentation threshold is relaxed / increased by a factor $\kappa > 1$ if a sub-step leads to a new filter front. For one, progress towards feasibility $\|f_l^+\|_2^2 = 0$ is required. This is ensured by the degree $\delta \leq 1$. At the same time, the condition $h_{\cup l-1} \leq h_{\cup l-1, front}$ ensures that the current SOI sufficiently encapsulates SOI from constraints of previous levels $f_{\cup l-1}^+$ (since otherwise the ill-posed HLSP sub-step may increase constraint violation $h_{\cup l-1}$).

On the other hand, the adaptation strategy tightens / decreases the threshold if a step is rejected according to (6). Furthermore, ζ steps must have been accepted with the current SOI threshold. This promotes trust-region reductions without SOI augmentation in order to escape local regularized minima. This is motivated by the analogies between trust-region methods and the Levenberg-Marquardt method³³ (smaller trust-region radius relates to higher regularization, whereby we prefer smaller trust-region radii without regularization / SOI).

The described procedure leads to a more moderate and slower adaptation strategy than for example directly coupling SOI activations to step acceptances and rejections. At the same time, the proposed heuristic for adapting the SOI threshold gives no guarantee that $0 < \|\hat{v}_l\|_2^2 = \|v_l\|_2^2 < \epsilon_{adaptive, l}$ holds at an optimally infeasible KKT point x^* . This can be explained by the fact that by virtue of the trust-region constraint, such a first-order point x^* can always be obtained (the SQP filter convergence proof in¹⁵ only requires the norm of the Hessian $\|\hat{H}_l\|_2$ to be bounded above; this is given for example for physically consistent systems

like robots with bounded Jacobians; no rank requirements are made). Nonetheless, we show on test functions (Sec. 6.2) that SOI is reliably activated for infeasible constraints even if the initial threshold is chosen far away.

4 | ALTERNATING DIRECTION METHOD OF MULTIPLIERS FOR HLSP

HLSP solvers both based on the active-set method³ and the interior-point method³⁴ have been proposed. While the former is very efficient with little changes of the active-set due to warm-starting capabilities, the latter one exhibits numerical stability and constant computation times even in the case of ill-posed problem formulations. However, both methods rely on expensive matrix factorizations in every inner iteration. In recent years, the ADMM for solving constrained optimization problems has seen a sharp rise in popularity, for example in distribute optimization³⁵. Here, we outline an ADMM for HLSP which mostly relies on matrix-vector operations. Such first order methods typically approach a solution of moderate accuracy in few iterations¹³. The proposed solver $\mathcal{N}ADM_2$ is based on nullspace projections of active constraints as described in Sec. 4. We detail our choice of the step-size parameter (Sec. 4.2) and our warm-starting strategy (Sec. 4.3). Our derivations are finally concluded with some considerations regarding the computation of the Lagrange multipliers of the active constraints $\mathcal{A}_{\cup l-1}$ (Sec. 4.4).

4.1 | Reduced Hessian based ADMM for HLSP

Following the approach in³⁶, which first proposed an ADMM based on the reduced Hessian, we introduce the change of variables

$$\Delta x_l = \Delta x_{l-1}^* + N_{l-1} \Delta z_l \quad (8)$$

with the overall primal solution

$$\Delta x^* = \sum_{l=1}^p N_{l-1} \Delta z_l^* \quad (9)$$

N_{l-1} is a basis of the nullspace of the active constraints $\mathcal{A}_{\cup l-1}$ such that

$$A_{\mathcal{A}_{\cup l-1}} N_{l-1} = 0 \quad (10)$$

and $N_0 = I_{n \times n}$. The particular solution Δx_{l-1}^* (with $\Delta x_0 = 0$) fulfills the condition $A_{\mathcal{A}_{\cup l-1}} \Delta x_{l-1}^* - b_{\mathcal{A}_{\cup l-1}} - v_{\mathcal{A}_{\cup l-1}}^* = 0$ (and which is obtained during the resolution of the higher priority levels). With appropriate choice of the nullspace basis N , this leads to either a decrease in variables (dense programming) or non-zeros (sparse programming). In this work, we rely on the turnback algorithm for the computation of sparse nullspace basis of banded matrices, see Sec. 5.

The change of variables leads to the following projected optimization problem, where \tilde{A} is the projected variable $\tilde{A} = AN$

$$\begin{aligned} \min. & \quad \frac{1}{2} \|v_{\mathbb{E}_l}\|_2^2 + \frac{1}{2} v_{\mathbb{I}_l}^T \|_2^2 \\ & \Delta z_l, \Delta \check{z}_l, v_{\mathbb{E}_l}, \\ & v_{\mathbb{I}_l}, w_{\mathbb{I}_l}, w_{\mathcal{I}_{\cup l-1}} \\ \text{s.t.} & \quad \tilde{A}_{\mathbb{E}_l} \Delta z_l - \check{b}_{\mathbb{E}_l} = v_{\mathbb{E}_l} \quad l = 1, \dots, p \\ & \quad \tilde{A}_{\mathbb{I}_l} \Delta z_l - \check{b}_{\mathbb{I}_l} \leq v_{\mathbb{I}_l} \\ & \quad \tilde{A}_{\mathcal{I}_{\cup l-1}} \Delta z_l - \check{b}_{\mathcal{I}_{\cup l-1}} \leq 0 \end{aligned} \quad (11)$$

The vector \check{b}_{Ξ} represents the expression

$$\check{b}_{\Xi_l} := b_{\Xi_l} - A_{\Xi_l} \Delta x_{l-1}^* \quad (12)$$

with the corresponding indices $\Xi_l = \{\mathbb{E}_l, \mathbb{I}_l, \mathcal{I}_{\cup l-1}\}$.

We introduce the slack variables $w_{\mathcal{I}_{U,l-1}}$ and $w_{\mathbb{I}_l}$, similarly to³⁴. Furthermore, the auxiliary variable $\Delta\hat{z}_l$ is added to the problem as in³⁷. The HLSP then writes as

$$\begin{aligned} \min_{\substack{\Delta z_l, \Delta\hat{z}_l, v_{\mathbb{E}_l}, \\ v_{\mathbb{I}_l}, w_{\mathbb{I}_l}, w_{\mathcal{I}_{U,l-1}}}} & \frac{1}{2} \| [v_{\mathbb{E}_l}^T \ v_{\mathbb{I}_l}^T]^T \|^2 + \mathbf{I}_+(w_{\mathbb{I}_l}) + \mathbf{I}_+(w_{\mathcal{I}_{U,l-1}}) \\ \text{s.t.} & \tilde{A}_{\mathbb{E}_l} \Delta z_l - \check{b}_{\mathbb{E}_l} = v_{\mathbb{E}_l} \quad l = 1, \dots, p \\ & \tilde{A}_{\mathbb{I}_l} \Delta z_l - \check{b}_{\mathbb{I}_l} = v_{\mathbb{I}_l} + w_{\mathbb{I}_l} \\ & \tilde{A}_{\mathcal{I}_{U,l-1}} \Delta z_l - \check{b}_{\mathcal{I}_{U,l-1}} = w_{\mathcal{I}_{U,l-1}} \\ & \Delta\hat{z}_l = \Delta z_l \end{aligned} \quad (13)$$

The slacks are penalized for negative values by the non-smooth indicator function \mathbf{I}_+

$$\mathbf{I}_+(w_{\Psi_l}) = \begin{cases} 0 & w_{\Psi_l} \geq 0 \\ +\infty & \text{otherwise} \end{cases} \quad (14)$$

where

$$\Psi_l = \{\mathcal{I}_{U,l-1}, \mathbb{I}_l\} \quad (15)$$

The augmented Lagrangian of level l of (11) writes as

$$\begin{aligned} \tilde{\mathcal{L}}_l(\Delta z_l, \Delta\hat{z}_l, v_{\mathbb{E}_l}, v_{\mathbb{I}_l}, w_{\mathbb{I}_l}, w_{\mathcal{I}_{U,l-1}}) &= \frac{1}{2} \|v_{\mathbb{E}_l}\|_2^2 + \frac{1}{2} \|v_{\mathbb{I}_l}\|_2^2 + \mathbf{I}_+(w_{\mathbb{I}_l}) + \mathbf{I}_+(w_{\mathcal{I}_{U,l-1}}) + \frac{\rho_{\mathbb{E}_l}}{2} \|\tilde{A}_{\mathbb{E}_l} \Delta z_l - \check{b}_{\mathbb{E}_l} - v_{\mathbb{E}_l} + v_{\mathbb{E}_l}\|_2^2 \\ &+ \frac{\rho_{\mathbb{I}_l}}{2} \|\tilde{A}_{\mathbb{I}_l} \Delta z_l - \check{b}_{\mathbb{I}_l} - v_{\mathbb{I}_l} - w_{\mathbb{I}_l} + v_{\mathbb{I}_l}\|_2^2 + \frac{\rho_l}{2} \|\tilde{A}_{\mathcal{I}_{U,l-1}} \Delta z_l - \check{b}_{\mathcal{I}_{U,l-1}} - w_{\mathcal{I}_{U,l-1}} + v_{\mathcal{I}_{U,l-1}}\|_2^2 + \frac{\sigma}{2} \|\Delta\hat{z}_l - \Delta z_l + \sigma^{-1} \lambda_{\Delta z_l}\|_2^2 \end{aligned} \quad (16)$$

where

$$v := \frac{1}{\rho} \lambda \quad (17)$$

The step-size parameters $\sigma > 0$ and ρ , the distinctions and choices $\rho_{\mathbb{E}_l} \rightarrow \infty$ and $\rho_{\mathbb{I}_l} = \rho_l$ are further explained in Sec. 4.2. λ are the Lagrange multipliers associated with the corresponding problem constraints Ξ . Resulting from the Karush-Kuhn-Tucker (KKT) first order optimality conditions $\tilde{K}_{v_{\mathbb{E}_l}} = 0$ and $\tilde{K}_{v_{\mathbb{I}_l}} = 0$ (with $\tilde{K} := \nabla \tilde{\mathcal{L}}$), we obtain the primal substitutions

$$v_{\mathbb{E}_l} = \tilde{A}_{\mathbb{E}_l} \Delta z_l - \check{b}_{\mathbb{E}_l} + v_{\mathbb{E}_l} \quad (18)$$

$$v_{\mathbb{I}_l} = \frac{\rho_l}{1 + \rho_l} (\tilde{A}_{\mathbb{I}_l} \Delta z_l - \check{b}_{\mathbb{I}_l} - w_{\mathbb{I}_l} + v_{\mathbb{I}_l}) \quad (19)$$

We then successively compute the alternating steps

$$\Delta\hat{z}_l^{k+1} \leftarrow \arg \min_{\hat{z}_l} \tilde{\mathcal{L}}_l(\Delta z_l, \Delta\hat{z}_l, v_{\mathbb{E}_l}, v_{\mathbb{I}_l}, w_{\mathbb{I}_l}, w_{\mathcal{I}_{U,l-1}}) \quad (20)$$

$$\Delta z_l^{k+1} \leftarrow \alpha \Delta\hat{z}_l^{k+1} + (1 - \alpha) \Delta z_l^k \quad (21)$$

$$v_{\mathbb{E}_l}^{k+1} \leftarrow (18) \quad (22)$$

$$v_{\mathbb{I}_l}^{k+1} \leftarrow (19) \quad (23)$$

$$\hat{w}_{\mathbb{I}_l}^{k+1} \leftarrow \tilde{A}_{\mathbb{I}_l} \Delta\hat{z}_l^{k+1} - v_{\mathbb{I}_l}^{k+1} \quad (24)$$

$$\hat{w}_{\mathcal{I}_{U,l-1}}^{k+1} \leftarrow \tilde{A}_{\mathcal{I}_{U,l-1}} \Delta\hat{z}_l^{k+1} \quad (25)$$

$$w_{\Psi}^{k+1} \leftarrow \max(\check{b}_{\Psi}, \alpha \hat{w}_{\Psi}^{k+1} + (1 - \alpha) w_{\Psi}^k + v_{\Psi}^k) \quad (26)$$

$$v_{\Psi}^{k+1} \leftarrow u_{\Psi}^k + \alpha \hat{w}_{\Psi}^{k+1} + (1 - \alpha) w_{\Psi}^k - w_{\Psi}^{k+1} \quad (27)$$

$$v_{\mathbb{E}_l}^{k+1} \leftarrow u_{\mathbb{E}_l}^k + \alpha (\tilde{A}_{\mathbb{E}_l} \Delta\hat{z}_l^{k+1} - v_{\mathbb{E}_l}^{k+1}) + (1 - \alpha) \check{b}_{\mathbb{E}_l} - \check{b}_{\mathbb{E}_l} \quad (28)$$

The parameter $\alpha \in (0, 2)$ is the over-relaxation parameter (typically $\alpha = 1.6$)³⁸. For the computation of the primal \hat{z}^{k+1} , we consider the optimality condition $\tilde{K}_{\hat{z}_l} = 0$ which leads to the expression

$$C_l \Delta\hat{z}_l^{k+1} = r_l \quad (29)$$

The positive definite matrix C_l is defined as

$$C_l = \tilde{A}_{\mathbb{E}_l}^T \tilde{A}_{\mathbb{E}_l} + \frac{\rho_l}{1 + \rho_l} \tilde{A}_{\mathbb{I}_l}^T \tilde{A}_{\mathbb{I}_l} + \rho_l \tilde{A}_{\mathcal{I}_{U-l}}^T \tilde{A}_{\mathcal{I}_{U-l}} + \sigma I \quad (30)$$

I is an identity matrix. The right hand side writes as

$$r_l = \tilde{A}_{\mathbb{E}_l}^T (\check{b}_{\mathbb{E}_l} - v_{\mathbb{E}_l}) + \frac{\rho_l}{1 + \rho_l} \tilde{A}_{\mathbb{I}_l}^T (\check{b}_{\mathbb{I}_l} + w_{\mathbb{I}_l} - v_{\mathbb{I}_l}) + \rho_l \tilde{A}_{\mathcal{I}_{U-l}}^T (\check{b}_{\mathcal{I}_{U-l}} + w_{\mathcal{I}_{U-l}} - v_{\mathcal{I}_{U-l}}) + \sigma z_l^k \quad (31)$$

Once the alternating steps of level l have converged with $\|\tilde{K}_l\|_2 < \eta$, the active constraint sets \mathcal{A}_l^* and \mathcal{A}_l corresponding to \mathcal{I}_{U-l} and \mathbb{I}_l need to be composed. η is a positive numerical threshold. The level l^* is referred to as ‘virtual’ priority level and maintains the prioritization between active sets of \mathcal{I}_{U-l} and \mathbb{I}_l ³⁴. We use the following decision criteria to determine active constraints

$$w_{\mathcal{I}_{U-l}} < \nu \quad \text{and} \quad \lambda_{\mathcal{I}_{U-l}} > \nu \quad (32)$$

$$w_{\mathbb{I}_l} < \nu \quad \text{and} \quad v_{\mathbb{I}_l} < -\nu \quad (33)$$

The resolution of the HLSP is then continued with the ADMM of the next level $l + 1$ projected into the nullspace N_l of the new active set $\mathcal{A}_{U-l} = \mathcal{A}_{U-l} \cup \mathcal{A}_l$. The remaining inactive constraints are contained in the updated inactive set \mathcal{I}_{U-l} .

4.2 | Choice of the step-size parameters ρ

Since equality constraints \mathbb{E}_l are necessarily active at convergence, we choose $\rho_{\mathbb{E}_l} \rightarrow \infty$ ³⁹. It can be seen that this leads to a more efficient algorithm since the dual update $v_{\mathbb{E}_l}$ (28) is zero and therefore does not need to be computed.

Similarly, the choice $\rho_{\mathbb{I}_l} \rightarrow \infty$ for the inequality constraints \mathbb{I}_l would render its corresponding equation in (27) obsolete. In this case, the inequality constraints are treated as equalities. Consequently, at ADMM convergence, feasible inequality constraints are saturated (with $\tilde{A}_{\mathbb{I}_l} z_l - \check{b}_{\mathbb{I}_l} = 0$) and infeasible constraints are active ($v_{\mathbb{I}_l} < 0$). However, we noticed that this leads to increased and unnecessary constraint activations due to the limited convergence accuracy of the ADMM (see also Sec. 4.3). Instead, we set the step-size parameter $\rho_{\mathbb{I}_l} = \rho_l$ according to³⁷.

4.3 | Warm-starting HLSP’s

Oftentimes, a slowly evolving sequence of programs (parametric program) needs to be resolved, for example in the context of S-HLSP (see Sec. 2.2). In this case, and in contrast to interior-point methods, the ADMM can be easily warm-started, i.e., a good initial guess for the primal and dual variables reduces the number of alternating iterations until convergence. We store the optimal primal and dual values z_l^* , $w_{\Psi_l}^*$ and $v_{\Psi_l}^*$ and the step-size parameter ρ_l^* after convergence of each level $l = 1, \dots, p$. In the next problem instance, the primal and dual variables are then warm-started with these values. If exactly the same HLSP is solved, our algorithm therefore converges as expected with zero iterations with $\Delta x_{k+1}^* = \Delta x_k^* = \sum_{l=1}^p N_{l-1} \Delta z_{l,k}^*$ (9).

Nonetheless, we observed that by warm-starting the primal and dual variables, constraints previously activated tend to be activated again in the next iteration. Potentially, this is caused by the inherently moderate accuracy of ADMM. This can artificially delay convergence of outer methods like S-HLSP if these constraints are not actually active in the corresponding non-linear program. We therefore reset the primal and dual sub-steps to zero in every new HLSP instance. One argumentation for this procedure is that at S-HLSP convergence, the primal sub-step $\|\Delta x\|_2 \leq \chi$ vanishes and therefore poses a good initial guess when a non-linear parametric program is solved.

4.4 | Lagrange multipliers of active constraints

Considering the dual ascent step $v_{\mathcal{A}_{U-l}} = v_{\mathcal{A}_{U-l}} - \nabla_{v_{\mathcal{A}_{U-l}}} \mathcal{L}$ for the update of the Lagrange multipliers $v_{\mathcal{A}_{U-l}}$, we can see that the gradient $\nabla_{v_{\mathcal{A}_{U-l}}} \mathcal{L} = 0$ since $A_{\mathcal{A}_{U-l}} \Delta x_0 - b_{\mathcal{A}_{U-l}} - v_{\mathcal{A}_{U-l}}^* = 0$ and $A_{\mathcal{A}_{U-l}} (\Delta x_0 + N_{l-1} \Delta z) - b_{\mathcal{A}_{U-l}} - v_{\mathcal{A}_{U-l}}^* = A_{\mathcal{A}_{U-l}} N_{l-1} \Delta z = 0$ as well. Therefore, the Lagrange multipliers associated with the active constraints \mathcal{A}_{U-l} (and whose nullspace the problem of level l is projected into) are not updated.

As noted in³⁴, the Lagrange multipliers of the active constraints are not necessary as none of the other primal or dual variables depend on it. However, the Lagrange multipliers may be needed within a non-linear solver based on Newton's method. Here, the Lagrange multipliers are used for the hierarchical Hessian. We use a fast conjugate gradient method to compute the Lagrange multipliers if required by the non-linear solver. In case that we use the turnback nullspace bases (see Sec. 5), we use the L factor of the LU decomposition of $A_{\mathcal{A}_{U,t-1}}$ for preconditioning the CG algorithm for accelerated convergence. Note that with the choice of other nullspace basis (for example based on the QR decomposition), matrix factorizations can be re-used for efficient computation of the Lagrange multipliers⁵.

5 | TURNBACK ALGORITHM FOR EULER INTEGRATED DYNAMICS

One critical element of the above nullspace method based HLSP solver is to efficiently compute a basis of the nullspace of the active constraints. The appropriate choice of the nullspace basis N leads to either a decrease in variables (dense programming) or non-zeros (sparse programming). In this work, we rely on the turnback algorithm for the computation of sparse nullspace basis for banded matrices, which arise in discrete optimal control problems⁴⁰. The main computational step of the turnback algorithm is to determine linearly independent subsets in the matrix A , to which a certain number of columns of A is linearly dependent. These columns are then used to compute a basis of the nullspace. Additionally, in our desired context of PTO, it is important to preserve resulting banded structures of the constraints as much as possible. The turnback algorithm is able to do so by considering nullspace vectors which are computed with respect to subsets of the block diagonal matrix instead of the whole one. In this work, we introduce some computational shortcuts to the turnback algorithm tailored to dynamics discretized by Euler integration. Importantly, we avoid a costly initial rank-revealing matrix factorization.

First, we formulate our system dynamics discretized by Euler integration (Sec. 5.1). We then outline the algorithmic details of the original turnback algorithm (Sec. 5.2). It is based on identifying linearly independent column subsets in the matrix A . In Sec. 5.3, we show how to identify these subsets in the case of Euler integrated dynamics and derive an upper bound on the number of columns in the subsets. This enables us in Sec. 5.4 to design an efficient turnback algorithm without the need of an expensive initial rank-revealing matrix factorization. Finally, we address the full-rank property of the resulting basis of nullspace (Sec. 5.5), demonstrate how the high degree of sparsity in the case of full actuation can be transferred to the case of under-actuation (Sec. 5.6) and comment on the parallelization of our algorithm (Sec. 5.7).

5.1 | Euler integrated dynamics

The dynamics of a rigid-body system are described by the inverse dynamics Newton-Euler equations⁴¹

$$\mathcal{ID}(q, \dot{q}, \tau, \gamma) := M\ddot{q} = S^T \tau - V(q, \dot{q}) + J^T \gamma \quad (35)$$

The joint torques $\tau \in \mathbb{R}^{n_\tau}$ and contact forces $\gamma \in \mathbb{R}^{n_\gamma}$ are considered the input variables of the system. $S \in \mathbb{R}^{n_\tau \times n_q}$ is a full-rank selection matrix describing under-actuation of the system $n_\tau < n_q$. The joint angles $q \in \mathbb{R}^{n_q}$, velocities $\dot{q} \in \mathbb{R}^{n_q}$ and accelerations $\ddot{q} \in \mathbb{R}^{n_q}$ describe the system state. $M(q) \in \mathbb{R}^{n_q \times n_q}$ is the whole-body inertia matrix. $V(q, \dot{q}) \in \mathbb{R}^{n_q}$ describes linear and non-linear force effects like Coriolis, centrifugal, gravitational and frictional forces. The Jacobian $J(q) \in \mathbb{R}^{n_\gamma \times n_q}$ is associated with the contact points. It has been noted in⁴² that the inverse dynamics form (explicit joint torques) is computationally advantageous compared to the forward dynamics equations (in contrast to explicit joint accelerations).

In the following, for visualization purposes, we introduce the change of variables

$$\tilde{q} = \frac{1}{\Delta t} q \quad \text{and} \quad \tilde{\tau} = \Delta t \tau \quad (36)$$

The states $s \in \mathbb{R}^{T n_s}$ (with $n_s = n_q + n_{\dot{q}}$) and controls $u \in \mathbb{R}^{T n_u}$ (with $n_u = n_\tau + n_\gamma$) are defined as

$$s = [\tilde{q}_1^T \ \dot{\tilde{q}}_1^T \ \cdots \ \tilde{q}_T^T \ \dot{\tilde{q}}_T^T]^T \quad \text{and} \quad u = [\tilde{\tau}_0^T \ \gamma_0^T \ \cdots \ \tilde{\tau}_{T-1}^T \ \gamma_{T-1}^T]^T \quad (37)$$

We assume known constant \tilde{q}_0 and $\dot{\tilde{q}}_0$.

quaternions) to facilitate the linear integration scheme above. For the remainder of this work, we therefore assume $n_q = n_{\dot{q}}$. Gimbal lock can be avoided for example as described in⁵.

The derivatives of $\mathcal{I}\mathcal{D}$ with respect to q and \dot{q} can be computed according to⁴⁴. Similarly, the first and second order derivatives of a function $f(q)$ with respect to \tilde{q} writes as

$$\frac{\partial f(q)}{\partial \tilde{q}} = \Delta t \frac{\partial f(q)}{\partial q} \quad (38)$$

$$\frac{\partial^2 f(q)}{\partial \tilde{q}^2} = \Delta t^2 \frac{\partial^2 f(q)}{\partial q^2} \quad (39)$$

This results in the partial derivatives

$$E_{1,t} := \frac{\partial f_{dyn,1}(t)}{\partial q_t} = -I, \quad E_{2,t} := \frac{\partial f_{dyn,1}(t)}{\partial \dot{q}_t} = -I, \quad E_{3,t} := \frac{\partial f_{dyn,1}(t)}{\partial q_{t+1}} = I, \quad E_{4,t} := \frac{\partial f_{dyn,1}(t)}{\partial \dot{q}_{t+1}} = -I \quad (40)$$

$$B_t := \frac{\partial f_{dyn,2}}{\partial \tilde{\tau}_t} = \frac{\partial f_{dyn,2}}{\partial \tau_t} \frac{\partial \tau_t}{\partial \tilde{\tau}_t} = -G_t S^T, \quad F_t := \frac{\partial f_{dyn,2}(t)}{\partial \gamma_t} = G_t J_t^T \quad (41)$$

$$D_{1,t} := \frac{\partial f_{dyn,2}(t)}{\partial \tilde{q}_t} = \frac{\partial f_{dyn,2}(t)}{\partial q_t} \frac{\partial q_t}{\partial \tilde{q}_t} = \frac{\partial f_{dyn,2}(t)}{\partial q_t} \Delta t, \quad D_{2,t} := \frac{\partial f_{dyn,2}(t)}{\partial \dot{q}_t} = L_t(-\Delta t \dots) \quad (42)$$

$$D_{3,t} := \frac{\partial f_{dyn,2}(t)}{\partial \tilde{q}_{t+1}}, \quad D_{4,t} := \frac{\partial f_{dyn,2}(t)}{\partial \dot{q}_{t+1}} = L_t(-\Delta t \dots) \quad (43)$$

It can be observed that due to the substitutions (37), Δt does not appear as denominator. This is numerically advantageous for small time steps $\Delta t \ll 1$ s due to better matrix conditioning. Ruiz equilibration $\hat{A} = S_t A S_r$ ⁴⁵ can equally be employed but comes at a higher computational cost. The nullspace basis of the original matrix A becomes $Z = S_r \hat{Z}$ with $\hat{A} \hat{Z} = 0$.

5.2 | Turnback algorithm

The turnback algorithm based on the LU decomposition to compute a nullspace basis for a banded matrix $A \in \mathbb{R}^{m \times n}$ consists of the following steps⁸:

1. Compute rank revealing $P^T L U Q^T$ decomposition of A (rank r_A). Then, $r_Z = n - r_A$.
2. Determine the index vector $b \in \mathbb{R}^{r_Z}$, which indicates the first non-zero entry of each column of

$$Z_{LU} = Q \begin{bmatrix} -U_1^{-1} U_2 \\ I \end{bmatrix} \quad (44)$$

Z is upper block triangular due to the block-diagonal structure of A .

3. Determine the turnback pivot columns $\pi \in \mathbb{R}^{r_Z}$. They are the row indices of the permuted identity matrix in (44).
4. For each index $i = 1, \dots, r_Z$ in b , add columns to the sub-matrix $G_i \in \mathbb{R}^{n \times r_Z}$ to the right of column b_i of A until linear dependency is detected. The turnback pivot column $\pi(i)$ is not added to the sub-matrix.
5. Compute the null-vector

$$z_i = Q \begin{bmatrix} U_1^{-1} u_2 \\ 0_{\pi(i)-r_A-1} \\ 1 \\ 0_{n-\pi(i)} \end{bmatrix} \quad (45)$$

Q , U_1 and u_2 result from the LU decomposition of the sub-matrix G_i . u_2 corresponds to the column $\pi(i)$ of A .

The resulting turnback nullspace basis is full-rank since each pivot-column is chosen only once during the submatrix augmentation and therefore has a similar structure to (44) with a permuted identity matrix ensuring full column rank.

5.3 | Subset determination for turnback algorithm

In the following, we derive a conservative bound for the number of columns which are needed for linearly independent sub-sets of the Euler integrated dynamics. We structure the permuted matrices (PDXED) and (PDIED) as

$$G_t := \left[\begin{array}{c|c} G_{1,t} & G_{2,t} \\ \hline G_{1,t}^{ua} & G_{2,t}^{ua} \end{array} \right] \quad (46)$$

The operator $\lceil a \rceil$ rounds the scalar a to its nearest upper integer.

Theorem 1. *If B_t and E_t (or namely, M_t and S_t^T) with $t = 0, \dots, T$ are of full column rank $r_{B_t} = n_\tau$ and $r_{E_t} = n_q$, the basis of nullspace of $A := \nabla_x f_{\text{dyn}}$ (DED) is of rank $r_Z = T(n_\tau + n_\gamma)$. The linear independent sub-sets of A are banded within width of $\beta \leq (2 + \mu)Tn_s + (3 + \mu)(n_\tau + n_\gamma)$. The subset augmentation factor μ is given by*

$$\mu = \left(\left\lceil \frac{2n_{ua}}{n_q - n_{ua}} \right\rceil \right) \quad \text{for } 0 \leq n_{ua} < n_q \quad (47)$$

Proof. Full actuation First, we consider the case of computing a nullspace basis of DED in the case of full actuation (empty matrices $G_{1,t}^{ua}, G_{2,t}^{ua}$ with $n_{ua} = 0$). The rank of DED is $r_A = Tn_s$ (number of rows, with full row rank). The dimension of the nullspace basis follows with $r_Z = T(n_s + n_\tau + n_\gamma) - Tn_s = T(n_\tau + n_\gamma)$ (number of columns minus rank of matrix A). The bandwidth can be identified by finding row and column permutations P_t and Q_t , such that the column subset corresponding to time step t is permuted to the upper left, see PDXED for explicit and PDIED for implicit Euler integrated dynamics, respectively. The $3(n_\tau + n_q)$ leftmost columns $G_{1,t}$ are clearly full rank due to the block-diagonal consisting of full-rank elements B and E . The rightmost $n_q + 3n_\gamma$ columns $G_{2,t}$ are linearly dependent of them. The linearly independent subsets of DED are maximally of length $\beta = 2n_s + 3(n_\tau + n_\gamma)$. The above is successively applied to all time steps $t = 0, \dots, T$.

Under-actuation We now consider the case of under-actuation of degree $n_{ua} > 0$, such that $n_\tau + n_{ua} = n_q$. In the following, we do not consider turnback pivot columns corresponding to contact forces γ . These columns are already used in the nullspace basis corresponding to the contact forces itself, while repeated use would violate the full-rank property, see Sec. 5.5.

Considering the permutations PDXED or PDIED, we see that G_t has at most rank (number of rows of the subset)

$$r_{G_t} = (3 + \mu)(n_\tau + n_q + n_{ua}) \quad (48)$$

The subset augmentation factor μ adds additional time steps to the sub-set. The number of columns is (columns of subset minus the pivot columns that need to be in the linear subset)

$$c_{G_t} = (3 + \mu)n_\tau + (4 + 2\mu)n_q \quad (49)$$

The maximum dimension of the nullspace of G_t is then

$$n_{G_t} = c_{G_t} - r_{G_t} = (1 + \mu)n_q - (3 + \mu)n_{ua} \quad (50)$$

The number of linearly dependent columns within the given subset n_{G_t} needs to be larger than the number of pivot columns (as these are used to form the basis of the nullspace)

$$n_{G_t} \geq n_\tau \quad (51)$$

Inserting (50), the expression for μ (47) follows.

With this choice of μ , we find a subset which is linear dependent to our n_τ pivot columns. The n_γ columns corresponding to the contact forces are already linearly dependent of G_t as discussed above. The bandwidth of the linearly independent matrix sub-sets therefore becomes

$$\beta = (2 + \mu)n_s + (3 + \mu)(n_\tau + n_\gamma) \quad (52)$$

□

The augmentation factor μ is a conservative measure since r_{G_t} is an approximation of the exact rank \hat{r}_{G_t} of G_t , with $r_{G_t} \geq \hat{r}_{G_t}$ (therefore, the bandwidth is most likely smaller with $n_{G_t} \leq \hat{n}_{G_t}$). In case of full under-actuation $n_{ua} = n_q$, the nullspace basis

Algorithm 2 turnbackParam

Input: $T, n, n_\tau, n_\gamma, n_s, n_{ua}, \beta$
Output: $r_A, r_Z, b \in \mathbb{R}^T, b^+ \in \mathbb{R}^T, \pi \in \mathbb{R}^{r_z}$

```

1:  $r_A = Tn_s$ 
2:  $r_Z = T(n_\tau + n_\gamma)$ 
3:  $n_\pi = 0$ 
4:  $n_b = 0$ 
5: for  $t = 0 : T - 1$  do
6:    $b(t) = n_b$ 
7:    $b^+(t) = \min(n_b + \beta, n)$ 
8:   for  $j = 0 : n_\gamma$  do
9:      $\pi(n_\pi) = n_b + n_\tau + j$ 
10:     $n_\pi \leftarrow n_\pi + 1$ 
11:   end for
12:   for  $j = 0 : n_q - n_{ua}$  do
13:     if Explicit Euler integrated dynamics then
14:        $\pi(n_\pi) = n_b + n_\tau + n_\gamma + n_q + n_{ua} + j$ 
15:     else if Implicit Euler integrated dynamics then
16:        $\pi(n_\pi) = n_b + n_\tau + n_\gamma + n_{ua} + j$ 
17:     end if
18:      $n_\pi \leftarrow n_\pi + 1$ 
19:   end for
20:    $n_b \leftarrow n_b + n_\tau + n_\gamma + n_s$ 
21: end for
22: return  $b, b^+, \pi, r_A, r_Z$ 

```

becomes dense with $\mu \rightarrow \infty$ as expected. This means that the system response of each time t_i is fully dependent on the system state at any other given time t_j with $j \neq i$.

As we show in Sec. 5.6, the bands of the turnback nullspace basis for dynamics integrated by the Euler method exhibit internal sparsity patterns. Still, the bandwidth β (52) is in contrast to an effective bandwidth of $n_\tau + n_\gamma$ for DDP (neglecting the cost of the forward roll-out for the state calculation requiring operations in n_s^2). In future work, we desire to incorporate DDP principles into HLSP for further computational efficiency. Nonetheless, the computation of the nullspace basis can be highly parallelized, as we describe in Sec. 5.7. This is not possible for DDP due to its recursive nature. Also, the backward recursion would need to be computed for every priority level. In contrast, the projection into the nullspace of the dynamics only needs to be done once. This might be more efficient for a high number of priority levels. Furthermore, multi-stage constraints involving states and controls from several time steps (aside from dynamics constraints) are handled due to the broad optimization point of view. Sparsity of such constraints is preserved by relying on less specialized formulations of the turnback algorithm as described in⁵.

5.4 | Turnback algorithm for Euler integrated dynamics

Based on theorem 1, we can implement a computationally efficient version of the turnback algorithm. Foremost, the linearly dependent column subsets of the matrix can be chosen according to the known bandwidth of Z . This means that an initial rank revealing LU decomposition of the matrix is not necessary. Concretely, the indices b indicate the first and $b^+ = b + \beta$ the last column of the sub-matrix of A . Furthermore, the turnback pivot columns are set as the columns corresponding to $D_{1,t}$ and the last n_τ columns of M_t . The reasoning is that the under-actuated part typically describes the free-flyer dynamics of the system which are well conditioned as they represent the full linear and rotational inertia of the system. Note that in theorem 1, we assume full-rank of M . This is typically given for physically consistent systems⁴⁶. Algorithm 2 details the computation of above values. The modified turnback algorithm then consists of following steps:

1. $b, b^+, \pi \leftarrow$ Alg. 2.

2. For each index $t = 1, \dots, T$, compute the LU decomposition of the column submatrix $G_t := A(\mathcal{C}_t)$ of A . The column set \mathcal{C}_t is given by the column range from $b(t)$ to $b^+(t)$ without the turnback pivot columns contained within. This leads to the set $\mathcal{C}_t = [b(t), b^+(t)] \setminus \pi(i)$ with $i = t(n_s + n_\tau + n_\gamma) + n_s, \dots, (t+1)(n_s + n_\tau + n_\gamma)$.
3. Compute the null-vector according to (45).

In case of SOI augmentation $V := [\nabla_x f_{dyn}^T \ R^T]^T$, where R is a factor of the hierarchical Hessian $\hat{H} := \nabla_x^2 f_{dyn}^T = R^T R^5$, we apply following two-step computation of a basis of the nullspace: first, $\mathcal{N}_{ib,ed}$ computes a basis of the nullspace according to the turnback algorithm for Euler integrated dynamics described above ($N_{\nabla_x f_{dyn}} \leftarrow \mathcal{N}_{ib,ed}(\nabla_x f_{dyn})$), and secondly, \mathcal{N}_{ib} according to the turnback algorithm as described in⁶ ($N_2 \leftarrow \mathcal{N}_{ib}(RN_{\nabla_x f_{dyn}})$). We then have $VN_{\nabla_x f_{dyn}}N_2 = 0$. Note, that \mathcal{N}_{ib} does not provide any sparsity guarantees but has been observed to reliably deliver sparse bases on a wide variety of sparsity patterns³⁶. At the same time, due to the high variable occupancy of the equation of motion, lower levels typically are not resolved anymore since most variables are eliminated by the projections.

5.5 | Full-rank property of turnback nullspace basis

Due to numerical inaccuracies, it can turn out that the pivot columns of a time step t are linearly independent of the corresponding column sub-matrix $A(\mathcal{C}_t)$ to a small error $\|\hat{u}_2\|_2 \leq \delta$ with $\delta \ll 1$ such that

$$A(\mathcal{C}_t) = P^T L \begin{bmatrix} U_1 & u_2 \\ 0 & \hat{u}_2 \end{bmatrix} Q^T \quad (53)$$

Furthermore, nullspace vectors may have an error higher than a tolerance $\|Az\|_2 > \delta$.

In these cases, we further augment the sub-matrix with blocks corresponding to timesteps $t^+ > t$ and $t^- < t$ to the ‘left’ and ‘right’ of stage t . The full-rank property of the resulting nullspace basis is preserved by not adding columns of A that correspond to turnback pivot columns of lower time-steps $t^- < t$. The basis of the nullspace then exhibits the following structure (we depict the extreme case of full augmentation)

$$Z_{ib} = \begin{bmatrix} X_{1,1} & X_{1,1} & \dots & X_{1,T-1} & X_{1,T} \\ I & & & & \\ X_{2,1} & X_{2,2} & \dots & X_{2,T-1} & X_{2,T} \\ X_{3,1} & I & \dots & & \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ X_{n-2,1} & X_{n-2,2} & \dots & X_{n-2,T-1} & X_{n-2,T} \\ X_{n-1,1} & X_{n-1,2} & \dots & I & \\ X_{n,1} & X_{n,2} & \dots & X_{n,T-1} & I \\ X_{n,1} & X_{n,2} & \dots & X_{n,T-1} & X_{n,T} \end{bmatrix} \quad (54)$$

The identity matrices correspond to the turnback pivot columns of A . These ensure full-column rank of the turnback nullspace. Furthermore, it can be easily confirmed that the above is full-rank as

- columns to the left are not a linear combination of each of its columns to the right, as this would destroy the sparsity (green)
- columns to the right are not a linear combination of each of its columns to the left, as they can not eliminate the entries on the same rows as the sparse rows (green).

5.6 | Under-actuated systems

We consider the basis of nullspace of (PDXED) for fully-actuated systems ($n_{ua} = 0$) in the case of explicit Euler integrated dynamics

$$[G_{1,t} \ G_{2,t}] Z_t = 0 \quad \text{with} \quad Z_t = \begin{bmatrix} -G_{1,t}^{-1} G_{2,t} \\ I \end{bmatrix} \quad (55)$$

Using block-wise inversion⁴⁷ of the matrix $G_{1,t}$ with full-rank and invertible B and E (see theorem 1), we get

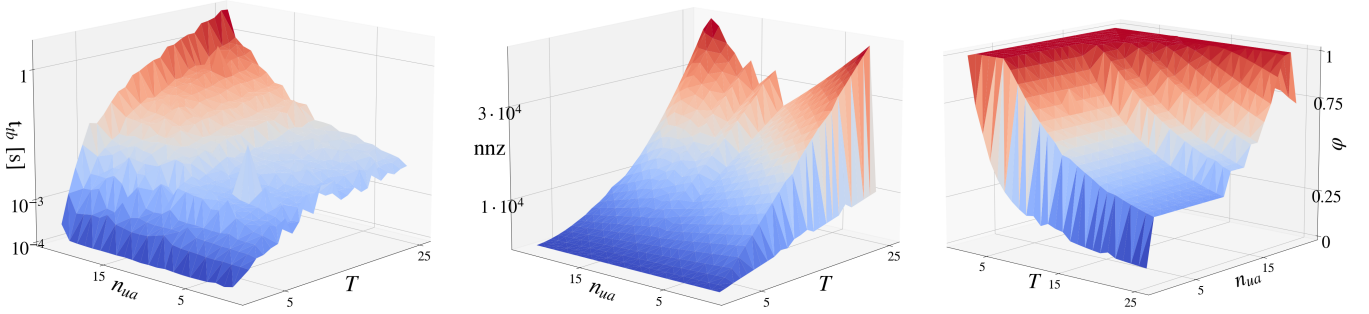


FIGURE 4 Computation time t_{ib} , number of non-zeros (nnz) and density (φ) of $Z^T Z$ of the turnback nullspace Z for Euler integrated dynamics with $n_q = n_{\dot{q}} = 22$ and $n_\gamma = 24$ in dependence of control horizon T and under-actuation n_{ua} .

The last three examples (inverted pendulum, manipulator and robot dog) are PTO where we use the turnback algorithm for dynamics integrated by Euler integration (Sec 5). The latter two simulations are thereby concerned with under-actuated systems where we follow the developments outlined in Sec. 5.6.

The simulations are run on an 11th Gen Intel Core i7-11800H 2.30GHz \times 16 with 23 GB RAM. The implementations of $\mathcal{N}ADM_2$ and the turnback algorithm are based on the Eigen library⁴⁹ and implemented in C++. The matrix C (30) is factorized ($O(n^3)$) only if the step-size parameter ρ is updated. We use a LDLT decomposition with low non-zero fill-in (for example compared to the QR decomposition). Otherwise, $\mathcal{N}ADM_2$ relies on matrix-vector operations ($O(n^2)$) as a first-order method. The HLSP solver $\mathcal{N}IPM_2$ based on the IPM (matrix factorizations in every iteration), which we proposed in our previous work³⁴, is modified by incorporating the proposed turnback nullspace basis for dynamics integrated by the Euler method. We can expect computational advantage for $\mathcal{N}ADM_2$ if

$$\iota_{\mathcal{N}ADM_2} + \iota_{\mathcal{N}ADM_2, \rho} \cdot n < \iota_{\mathcal{N}IPM} \cdot (1 + n) \quad (57)$$

ι is the number of inner iterations of the respective solvers. $\iota_{\mathcal{N}ADM_2, \rho}$ is the number of factorization updates of $\mathcal{N}ADM_2$. n is the number of problem variables. In case of an increase of the KKT norm (which is not related to a change of ρ), we increase the regularization factor σ ($\sigma_0 = 1 \cdot 10^{-6}$) and reset ρ ($\rho_0 = 0.1$). The number of inner iterations of $\mathcal{N}ADM_2$ is limited to 1500, or 2000 for the test in Sec. 6.5. We use the analytical hierarchical Hessian⁵ as needed for the Newton's method in Fig. 1. In the robotics examples, the NL-HLSP's and HLSP's are computed by the pinocchio library⁵⁰ and the automatic differentiation package CppAD⁵¹. The Lagrange multipliers are computed according to Sec. 4.4 by the Conjugate Gradient solver LSQR⁵². The turnback algorithm is based on the rank-revealing LU decomposition provided by the library LUSOL⁵³. Note that we only depict the computation times of the HLSP sub-solvers. Due to memory limitations, all simulations except for the turnback algorithm evaluation are run on a single thread. $\mathcal{N}ADM_2$ and $\mathcal{N}IPM_2$ are compared to the hierarchical versions of the off-the-shelf solvers H-MOSEK⁵⁴, H-GUROBI⁵⁵ and H-OSQP³⁷. All solvers rely on the same framework for active and inactive set composition.

6.1 | Turnback algorithm for Euler integrated dynamics

First, we evaluate the computational efficiency of the turnback algorithm adapted to discrete Euler integrated dynamics. We compose randomized matrices M , D and B in (DED). We choose $n_q = n_{\dot{q}} = 22$, $n_\gamma = 24$ and $n_\tau = n_{\dot{q}} - n_{ua}$ with variable n_{ua} (this corresponds to the dimensions of the robot dog Solo12 with four feet exerting forces and torques, such that $n_\gamma = 4 \cdot 6$, and $n_{ua} = 6$). We use 8 threads for the parallel computations of the turnback algorithm as described in Sec. 5.7.

The results are given in Fig. 4. Depicted are the computation time and the number of non-zeros and density ($\varphi = \text{nnz}(A)/(n \cdot m)$ with $A \in \mathbb{R}^{m \times n}$) of the normal form $Z^T Z$ of the turnback nullspace depending of the time horizon $T = 0, \dots, 25$ and under-actuation $n_{ua} = 0, \dots, n_q$. We first consider the fully actuated case $n_{ua} = 0$. For $T = 25$, the turnback computation time is $t_{ib} = 4.9 \cdot 10^{-3}$ s. The resulting normal form $Z^T Z$ contains 103412 non-zeros with a density of 0.078. It can be observed that with under-actuation $n_{ua} > 0$, there is a sharp incline in non-zeros and density of $Z^T Z$. For example for $n_{ua} = 1$, the density increases to 0.328 with four times as many non-zeros (415125). This can be explained by the higher coupling within the diagonal blocks as demonstrated in Sec. 5.6. Nonetheless, the computation time does not increase as dramatically to $t_{ib} = 6.9 \cdot 10^{-3}$ s. For full

under-actuation $n_{ua} = n_q$, the resulting nullspace basis is dense ($\varphi = 1$) as expected. At the same time, the increase in non-zeros is quadratic. In contrast, for low n_{ua} , the linear increase of non-zero entries in T is clearly distinguishable.

6.2 | NL-HLSP test-functions

l		$f_l(x) \leq v_l$	$\mathcal{N}ADM_2$ (0.01 s)		$\mathcal{N}ADM_2 \rightarrow$ H-MOSEK (0.01 s \rightarrow 0.11 s: 0.12 s)		H-MOSEK (0.19 s)		H-OSQP (0.23 s)	
			$\ v_l^*\ _2$	Iter.	$\ v_l^*\ _2$	Iter.	$\ v_l^*\ _2$	Iter.	$\ v_l^*\ _2$	Iter.
1	Disk ineq.	$x_1^2 + x_2^2 - 1.9 \leq v_1$	$1.0 \cdot 10^{-5}$	6	$9.8 \cdot 10^{-6}$	1	$9.8 \cdot 10^{-6}$	12	$1.0 \cdot 10^{-5}$	7
2	Ros. eq.	$(1 - x_1)^2 + 100(x_2 - x_1^2)^2 = v_2$	$2.9 \cdot 10^{-4}$	14	$2.9 \cdot 10^{-4}$	14	$2.9 \cdot 10^{-4}$	17	$2.9 \cdot 10^{-4}$	13
3	Disk eq.	$x_1^2 + x_2^2 - 0.9 = v_3$	1	2	1	1	1	1	1	2
4	Disk eq.	$x_2^2 + x_3^2 - 1 = v_4$	$1.6 \cdot 10^{-6}$	2	$1.2 \cdot 10^{-16}$	1	$1.9 \cdot 10^{-16}$	1	$1.1 \cdot 10^{-11}$	2
5	Disk ineq.	$x_4^2 + x_5^2 + 1 \leq v_5$	1	1	1	1	1	1	1	1
6	Disk eq.	$x_6^2 + x_7^2 + x_8^2 - 4 = v_6$	$7.1 \cdot 10^{-7}$	4	$1.7 \cdot 10^{-8}$	1	$1.6 \cdot 10^{-10}$	8	$1.9 \cdot 10^{-8}$	1
7	Ros. eq.	$(1 - x_6)^2 + 100(x_7 - x_6^2)^2 = v_7$	$4.2 \cdot 10^{-4}$	1	$7.6 \cdot 10^{-8}$	24	$7.4 \cdot 10^{-8}$	35	$1.8 \cdot 10^{-4}$	111
8	McC. eq.	$\sin(x_9 + x_{10}) + (x_9 - x_{10})^2 - 1.5x_9 + 2.5x_{10} + 1 + M = v_8$	18.1	1	18.1	1	18.1	1	24.4	1
9	Reg. eq.	$x_{1:10} = v_9$	2.9	0	2.9	0	2.9	0	7.8	0
Σ				31		(31 \rightarrow 44) 75		76		138

TABLE 1 Non-linear test functions: optimal slacks v^* and number of outer iterations (Iter.) per priority level for a NL-HLSP with $p = 9$ and $n = 10$. The hierarchy is composed of disk, Rosenbrock (Ros.), McCormick (McC.) and regularization (Reg.) equality (eq.) and inequality (ineq.) constraints.

We apply $\mathcal{N}ADM_2$ as HLSP sub-solver for S-HLSP to solve a NL-HLSP composed of test functions as listed in Tab. 1. This problem constellation tests feasible and infeasible equality and inequality constraints. This also includes infeasibility arising from conflict with constraints from higher priority levels. Note that the McCormick function includes a large positive offset M . This enables S-HLSP to identify negative function minima (for example the minimum $f_{\text{McC}}(-0.547, -1.547) + M = -1.9133 + M$) despite its least-squares formulation. This can be easily confirmed with the following theorem:

Definition 2. A factor $M \geq 0$ is sufficiently large on the domain $\{x \in \mathbb{R}^n : x \in \mathcal{S}\}$ if a function $\hat{f}(x, M) = f(x) + M : \mathcal{S} \rightarrow \mathbb{R}_{>0}$.

Theorem 2. If $M \geq 0$ is sufficiently large on the domain $x \in \mathcal{S}$, then first-order optimality of $\hat{f}(x, M)^2$ applies at the same points $x^* \in \mathcal{S}$ as the twice continuously differentiable function $f(x)$.

Proof. We consider the first-order derivative of $\hat{f}(x, M)^2$ which writes as

$$\frac{\partial \hat{f}^2}{\partial x} = 2\hat{f} \frac{\partial \hat{f}}{\partial x} = 2\hat{f} \frac{\partial f}{\partial x} \quad (58)$$

Clearly, $\partial \hat{f}^2 / \partial x$ has the same zeros as $\partial f / \partial x$ for $\hat{f}(x, M) > 0$ on the domain $x \in \mathcal{S}$. \square

Figure 6 shows how the squared McCormick function with offset $(f_{\text{McC}} + M)^2$ has the same minima (x_9^*, x_{10}^*) as the original function in the range $x_9, x_{10} \in [-5.5, 4]$. Here, we choose $M = 20$. For even polynomials, the global value M^* , such that $\hat{f}(x^*, M^*) = 0$, can be found by polynomial optimization⁵⁶. In contrast, the squared McCormick function without offset has additional minima at the zeros of f_{McC}^2 , while missing any minima that are associated with negative function values $f_{\text{McC}} < 0$.

S-HLSP with $\mathcal{N}ADM_2$ identifies the primal solution $x = 0.983, 0.966, 0.257, 0, 0, 1.02, 1.04, 1.37, -0.547, -1.547$. The evolution of the primal over the S-HLSP outer iterations is depicted in Fig. 7. $\mathcal{N}ADM_2$ is able to solve the NL-HLSP to moderate accuracy. For example, according to Tab. 1, the Rosenbrock equality on level 7 is solved to a residual error of $\|v_7^*\|_2 = 4.2 \cdot 10^{-4}$ while H-MOSEK solves the same level to $\|v_7^*\|_2 = 7.4 \cdot 10^{-8}$. At the same time, the previous levels are solved to comparable accuracy (note that error comparisons in hierarchies need to consider that a higher error norm on a higher priority level can lead to lower error norm on a lower priority level). While being less accurate, $\mathcal{N}ADM_2$ (0.011 s) solves the problem the fastest out of all the solvers, see Fig. 5. This is partly due to the low number of outer S-HLSP iterations (31, about half as many as for H-MOSEK

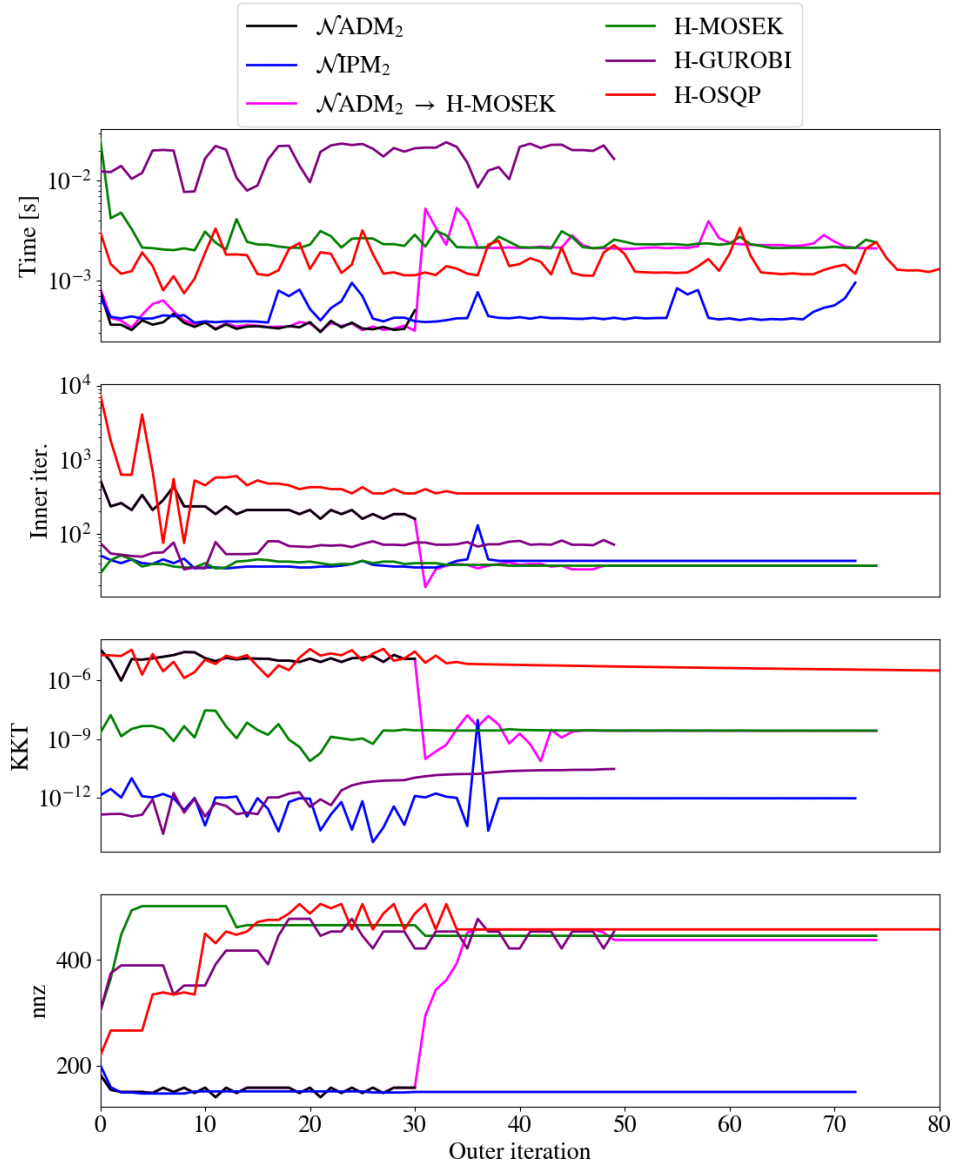


FIGURE 5 Non-linear test functions, data for the different HLSP sub-solvers over S-HLSP outer iteration: computation times per HLSP solve, number of inner iterations, KKT residuals and overall number of non-zeros handled throughout the whole hierarchy.

with 76). Still, from Fig. 5, it can be observed that the HLSP sub-problems are solved in about $4 \cdot 10^{-4}$ s. \mathcal{NIPM}_2 solves the sub-problems slightly slower in about $5 \cdot 10^{-4}$ s. This is in accordance with the number of inner iterations of $\sim 300 < \sim 50 \cdot n = \sim 500$ of \mathcal{NADM}_2 and \mathcal{NIPM}_2 , respectively. In contrast, the next fastest solver H-OSQP solves the inner iterations in about $1 \cdot 10^{-3}$ s. This clearly demonstrates the advantage of solving the KKT system projected into the nullspace of active constraints.

We furthermore consider the combination of both the low and high accuracy solvers \mathcal{NADM}_2 and H-MOSEK. It can be observed that a high accuracy solution is obtained when compared to the low accuracy solver \mathcal{NADM}_2 alone (level 7 at $7.6 \cdot 10^{-8}$ compared to $4.2 \cdot 10^{-4}$). At the same time, the computation time is lower (0.12 s) compared to the high accuracy solver H-MOSEK alone (0.19 s). Consequently, a sub-problem solver with moderate accuracy like \mathcal{NADM}_2 can be used to warm-start the S-HLSP with a lower accuracy primal guess. The reduced overall computation time follows due to the reduced number of high accuracy sub-problem solutions (44 compared to 76 for H-MOSEK alone).

Finally, we evaluate the adaptive SOI thresholding strategy developed in Sec. 3. For this, we set the initial value to $\epsilon_{adaptive,l} = 100$ (note that in all other examples, we initially set $\epsilon_{adaptive,l} = 1 \cdot 10^{-12}$) and the lower limit to $1 \cdot 10^{-12}$. The corresponding

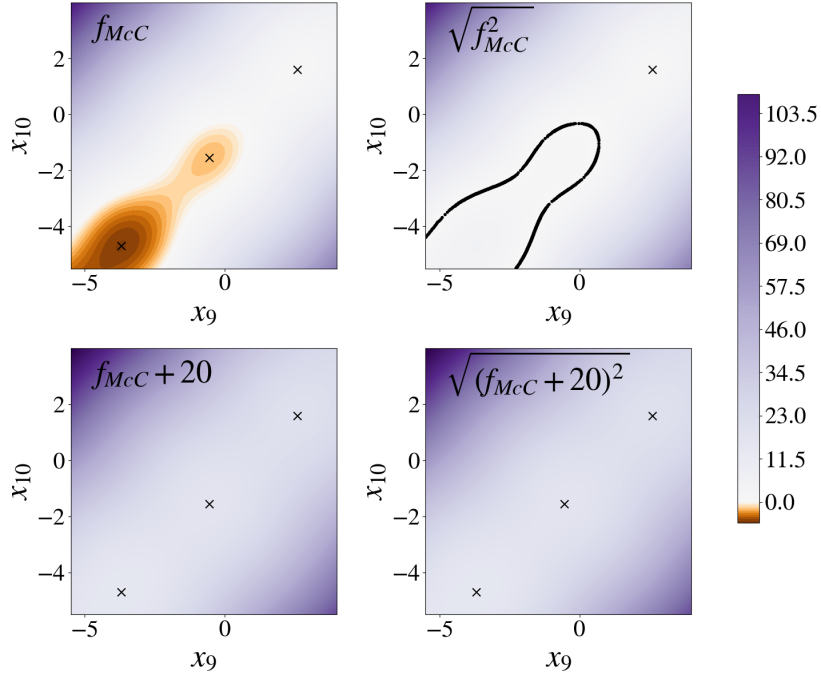


FIGURE 6 Different formulations of McCormick function: original f_{McC} , ℓ_2 -norm $\sqrt{f_{McC}^2}$, with offset $f_{McC} + M$, ℓ_2 -norm with offset $(f_{McC} + M)^2$ ($M = 20$). Negative function values are colored in orange tones. Local minima are marked in black.

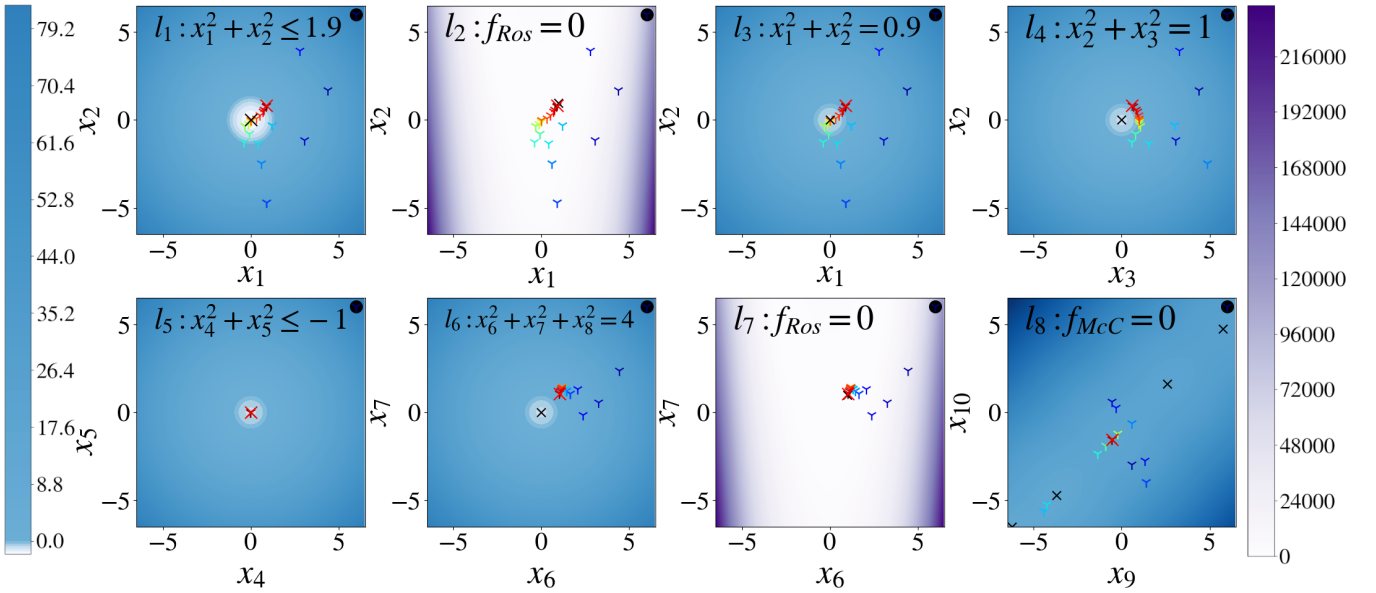


FIGURE 7 Non-linear test functions: primal x over S-HLSP outer iteration (colored triangles) from start point (black dot) to converged point (large red cross). Local minima are marked with black crosses.

parameters are chosen as $\zeta = 1$ and $\delta = 0.95$ (see Alg. 1). As can be seen from Fig. 8, a similar error reduction as in Tab. 1 is achieved. However, more iterations are necessary (92 instead of 31) as $\epsilon_{adaptive,l}$ is adjusted to the infeasibility of the constraints. Importantly, at convergence of the HSF of the infeasible levels $l = 2, 3, 5, 8$, we have $\|f_l^+\|_2^2 > \epsilon_{adaptive,l}$ and the SOI is activated. At the same time, the heuristic relaxes the threshold for example for the infeasible Rosenbrock constraint on level 2 in instances of sufficient progress in terms of optimality. Nonetheless, the SOI is erroneously activated for the feasible level 7. This motivates

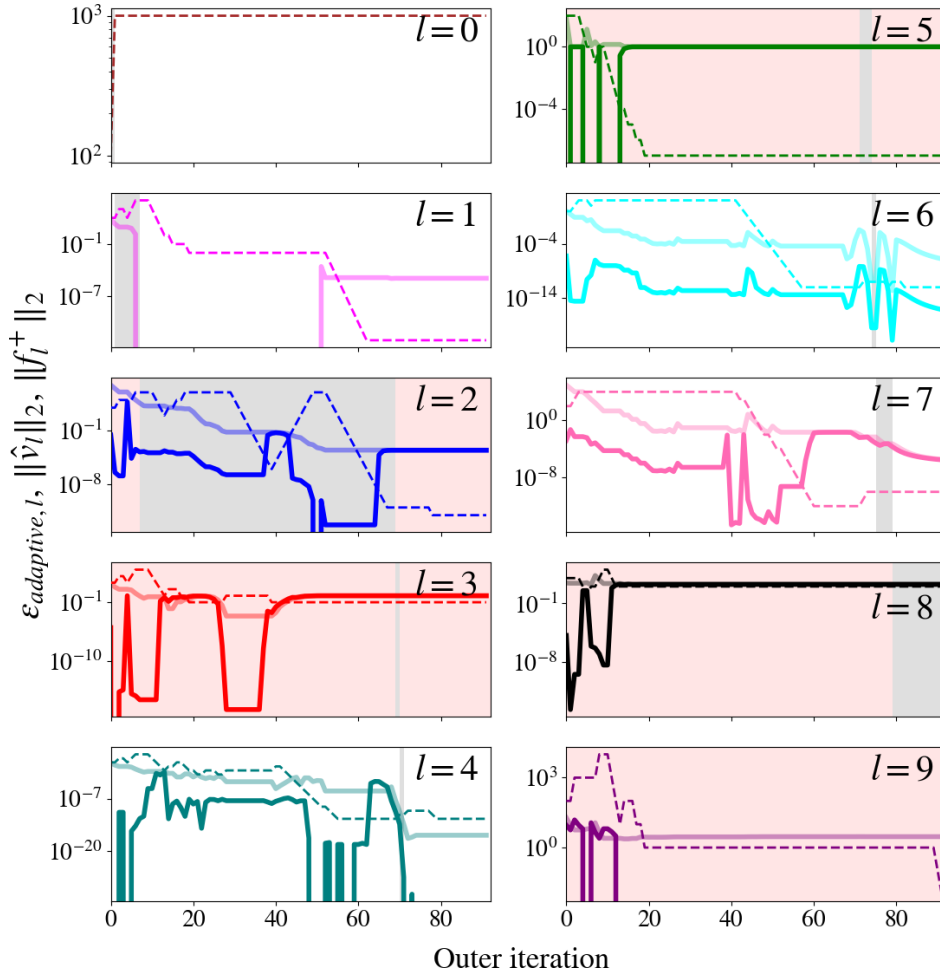


FIGURE 8 Non-linear test functions, $\mathcal{N}ADM_2$: linear slacks $\|\hat{v}_l\|_2$, non-linear error $\|f_l^+\|_2$ (light color) and $\epsilon_{adaptive,l}$ (dashed) for the levels $l = 1, \dots, p$ over outer iteration. The initial value is chosen as $\epsilon_{adaptive,l} = 100$. Gray background color indicates the outer iterations where the respective level has been resolved by the HSF. Infeasible levels are indicated by light red background color.

further investigation with respect to constraint optimality by avoiding regularized minima, for example based on machine learning methods for feasibility detection.

6.3 | Inverse kinematics of humanoid robot HRP-2

This simulation is concerned with solving an inverse kinematics problem for the humanoid robot HRP-2 with $n = 38$ degrees of freedom. The corresponding hierarchy is given in Tab. 2 ($p = 5$). The first level limits the joint angles. The second level positions the left and right feet and the left hand. The third level limits the CoM position to a bounding box. The fifth level positions the right hand towards an out-of-reach target $[-0.5 \ -0.5 \ -1]$ m below its feet. The right foot is positioned at $[0.015 \ -0.1 \ 0.1]$ m. The z component is approximately at ground level 0.1 m. Lastly, all variables are regularized to zero.

The results are given in Tab 2. The converged robot posture for $\mathcal{N}ADM_2$ is depicted in Fig. 10. Our proposed solver $\mathcal{N}ADM_2$ solves the HLSP sub-problems the fastest at about $6 \cdot 10^{-2}$ s. Fluctuations in computation time and non-zeros (Fig. 9) is due to activation and deactivations of SOI on the second level. At the same time, moderate accuracy is achieved. For example, the end-effector positioning of the left and right foot and the left hand on level 2 is resolved to an error of $3.1 \cdot 10^{-6}$ m while the error is reduced to less than $4.8 \cdot 10^{-8}$ m (H-OSQP) for the other solvers. The right hand task on level 3 is resolved to an error of 1.04 m. While this is worse than for example $\mathcal{M}IPM_2$ (0.98 m), the adaptive SOI threshold strategy enables SOI deactivation on the higher priority level 2. Without it, only manual tuning for each individual solver prevented the SOI activation which causes

l	$f_l(x) \leq v_l$	$\mathcal{N}ADM_2$ (0.06 s)		$\mathcal{N}IPM_2$ (0.40 s)		$\mathcal{N}A. \rightarrow \text{H-M.}$ (0.06 s \rightarrow 0.39 s: 0.46 s)		H-MOSEK (0.59 s)		H-GUROBI (1.06 s)		H-OSQP (0.089 s)	
		$\ v_l^*\ _2$	Iter.	$\ v_l^*\ _2$	Iter.	$\ v_l^*\ _2$	Iter.	$\ v_l^*\ _2$	Iter.	$\ v_l^*\ _2$	Iter.	$\ v_l^*\ _2$	Iter.
1	J. lim. ineq.	$5.1 \cdot 10^{-5}$	2	$3.4 \cdot 10^{-8}$	1	$3.4 \cdot 10^{-5}$	0	0	0	0	0	0	0
2	LF, RF, LH eq.	$3.1 \cdot 10^{-6}$	14	$2.6 \cdot 10^{-10}$	18	$7.8 \cdot 10^{-9}$	10	$9.9 \cdot 10^{-9}$	9	$7.4 \cdot 10^{-8}$	27	$4.8 \cdot 10^{-8}$	6
3	CoM ineq.	$4.1 \cdot 10^{-6}$	1	$1.0 \cdot 10^{-8}$	1	$1.0 \cdot 10^{-5}$	65	$1.0 \cdot 10^{-5}$	30	0	28	$2.6 \cdot 10^{-6}$	1
4	Right hand eq.	1.04	9	0.98	128	1.02	2	1.02	56	1.13	19	0.98	12
5	Reg. eq.	5.5	10	4.40	7	4.1	0	4.98	0	4.70	0	4.39	0
Σ			37		156		(48 \rightarrow 77) 125		95		74		19

TABLE 2 HRP-2 inverse kinematics: optimal slacks v^* and number of outer iterations (Iter.) per priority level for a NL-HLSP with $p = 5$ and $n = 38$. J. lim.: Joint limits, LF: left foot, RF: right foot, LH: left hand. $\mathcal{N}A.$: $\mathcal{N}ADM_2$; H-M: H-MOSEK.

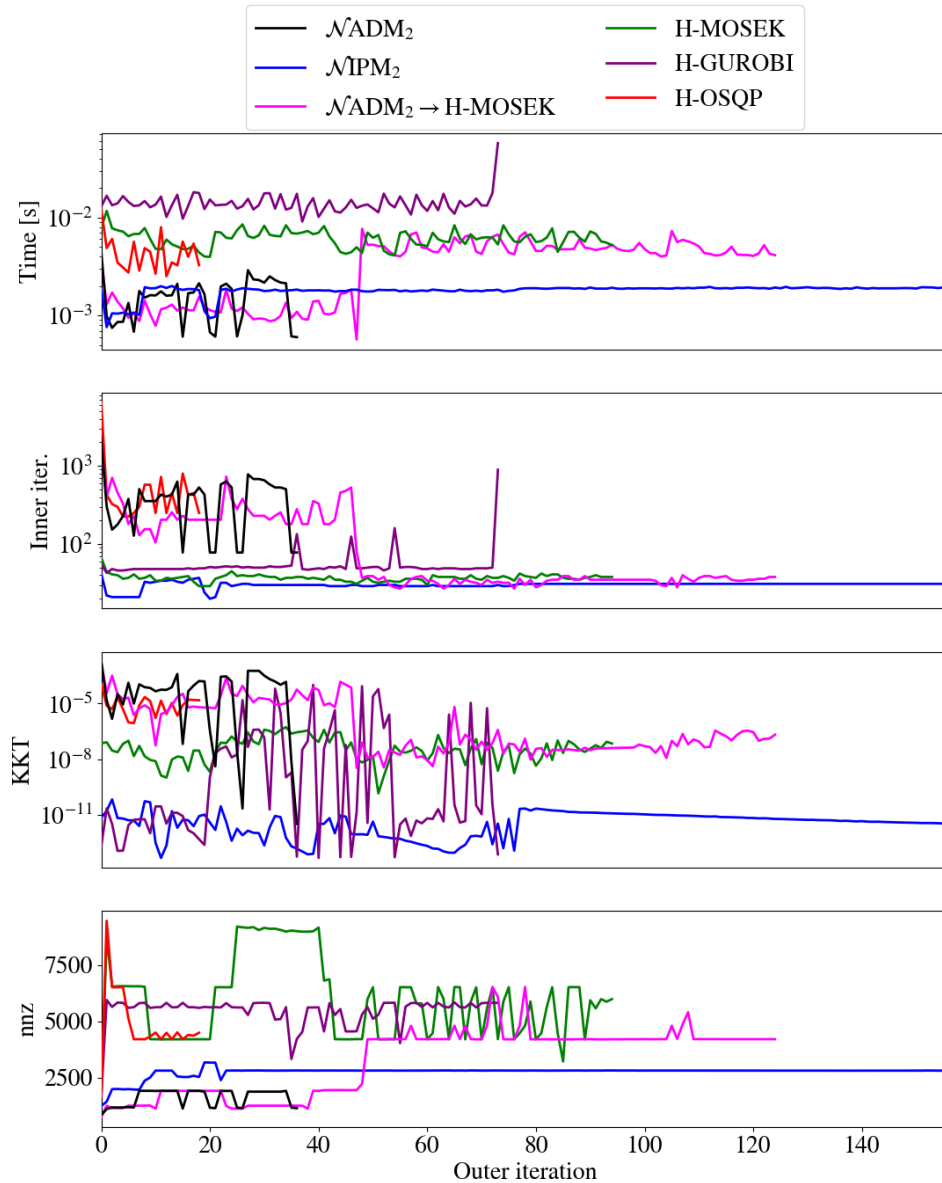


FIGURE 9 HRP-2 inverse kinematics, data for the different HLSP sub-solvers over S-HLSP outer iteration: computation times per HLSP solve, number of inner iterations, KKT residuals and overall number of non-zeros handled throughout the whole hierarchy.

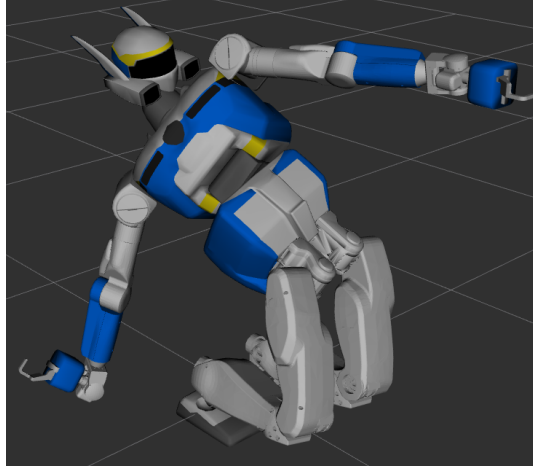


FIGURE 10 HRP-2 inverse kinematics, $\mathcal{N}ADM_2$: converged robot posture.

worse error convergence on lower priority levels due to high variable occupation. H-OSQP achieves a solution in the lowest number of outer iterations (19 compared to 48 for $\mathcal{N}ADM_2$) but is slower than $\mathcal{N}ADM_2$ due to the slow resolution of the HLSP sub-problems at about $4 \cdot 10^{-2}$ s.

6.4 | Time-optimal control of manipulator

l	$f_l(x) \leq v_l$	$\mathcal{N}ADM_2$ (4.8 s)		$\mathcal{N}IPM_2$ (4.8 s)		H-MOSEK (72.1 s)		H-GUROBI (3.2 s)	
		$\ v_l^*\ _2$	Iter.	$\ v_l^*\ _2$	Iter.	$\ v_l^*\ _2$	Iter.	$\ v_l^*\ _2$	Iter.
1	q, τ lim. ineq.	$1.7 \cdot 10^{-5}$	1	$5.4 \cdot 10^{-8}$	1	0	1	0	1
2	$f_{\text{dyn}}(x) = v_2$	$4.9 \cdot 10^{-8}$	17	$1.6 \cdot 10^{-4}$	29	$2.3 \cdot 10^{-9}$	48	$8.5 \cdot 10^{-8}$	10
3	$f_{\text{ef,adtoe}}(x) = v_3$	0.11	87	$8.1 \cdot 10^{-2}$	320	$7.8 \cdot 10^{-2}$	317	1.8	53
4	$\dot{h}(x) = v_4$	1509	1	1319	1	1363	1	211	1
5	$[q \ \dot{q}]^T = v_5$	231	0	245	1	274	1	91	7
6	$\tau = v_6$	306	1	274	1	276	1	18	13
Σ			110		353		370		86

TABLE 3 UR3e time-optimal control: optimal slacks v^* and number of outer iterations (Iter.) per priority level for a NL-HLSP with $p = 6$ and $n = 361$.

In this simulation setting, we aim to identify a time-optimal control under actuation limits for a kinematic reaching task of the fully actuated manipulator UR3e. Furthermore, we impose a regularization task of the momentum evolution for a safe robot movement. Such a constraint includes variables from several stages, which is a form of constraint typically not handled by recursive methods like DDP. Time-optimal control in least-squares programming can be achieved by a continuous approximation of the discrete optimal time t^* ⁵. Note that this requires a feasible ‘resting’ goal point, i.e. the robot can physically remain at this point until the end of the control horizon T .

The PTO hierarchy with $p = 6$ is given in Tab. 3. It is composed of state and control limits, explicit inverse Euler integrated dynamics, time-optimal control $f_{\text{dyn,adtoe}}$ and finally momentum time evolution \dot{h} , joint velocity, angle and torque regularization. The control horizon is chosen as $T = 20$ such that the number of variables is $n = 361$. The control time step is $\Delta t = 0.01$ s.

The results in Tab. 3 show S-HLSP convergence in 4.8 s and within 110 and 353 iterations for $\mathcal{N}ADM_2$ and $\mathcal{N}IPM_2$, respectively. This is in contrast to H-MOSEK which delivers a time-optimal solution in only 72.1 s. H-GUROBI fails to resolve the time-optimal control $f_{\text{dyn,adtoe}}$ and converges quickly due to SOI activation of the dynamics constraints (which occupies most of the variables such that lower levels show little variable activity).

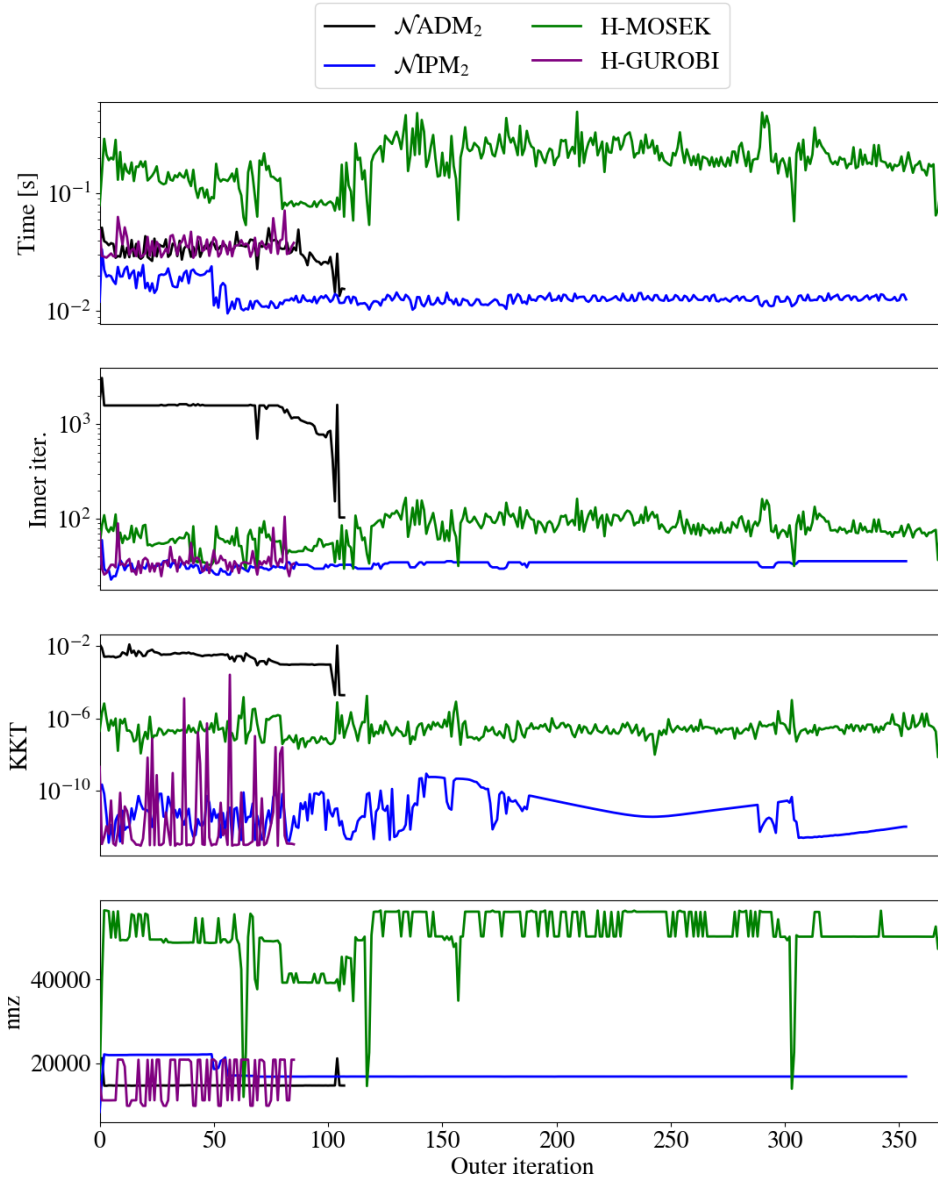


FIGURE 11 UR3e time-optimal control, data for the different HLSP sub-solvers over S-HLSP outer iteration: computation times per HLSP solve, number of inner iterations, KKT residuals and overall number of non-zeros handled throughout the whole hierarchy.

Fig. 11 shows how the projector based solvers $\mathcal{N}ADM_2$ and $\mathcal{N}IPM_2$ both solve the single S-HLSP iterations the fastest with run-times of around 0.05 s and 0.02 s (note that H-GUROBI only resolves the hierarchy up to the dynamics constraints due to SOI activation). In comparison, H-MOSEK solves the HLSP sub-problems in around 0.2 s. This is due to the significantly lower number of non-zeros throughout the hierarchy for $\mathcal{N}ADM_2$ and $\mathcal{N}IPM_2$ (20000, about half as many as for H-MOSEK).

Figure 12 shows the resulting joint torques normalized by their limits (lower graph) for $\mathcal{N}ADM_2$. A time-optimal bang-bang control profile (controls at their limits) can clearly be distinguished. This leads to a sharp drop-off of the task error $f_{ef,adtoe}$ at around control iteration 12 (upper graph).

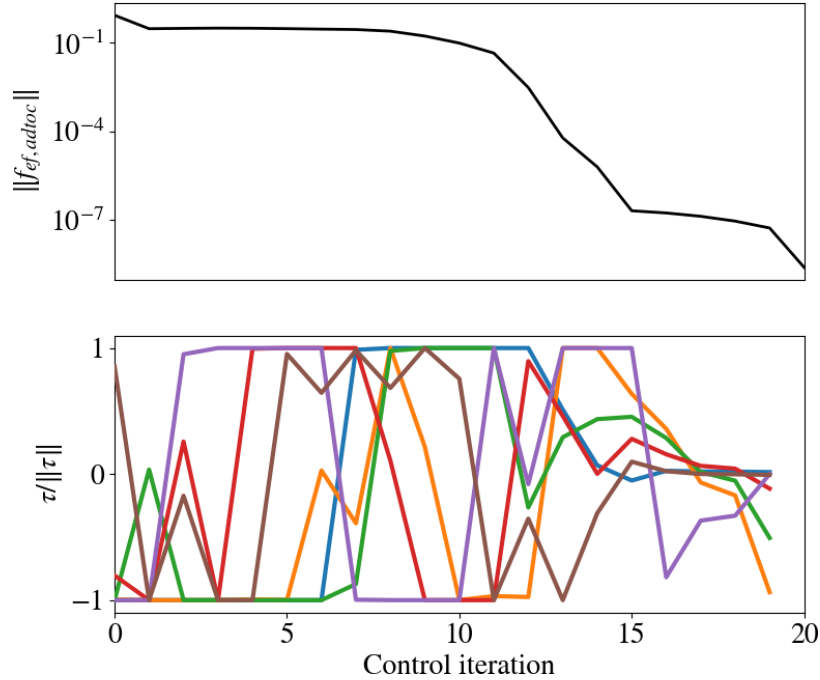


FIGURE 12 UR3e time-optimal control, $\mathcal{N}ADM_2$. Error reduction of the time-optimal reaching task (top) and joint torques normalized by their limits (bottom).

l	$f_l(x) \leq v_l$	$\mathcal{N}ADM_2$ (19.7 s)		$\mathcal{N}IPM_2$ (4.1 s)		H-MOSEK (11.8 s)		H-GUROBI (20.3 s)	
		$\ v_l^*\ _2$	Iter.	$\ v_l^*\ _2$	Iter.	$\ v_l^*\ _2$	Iter.	$\ v_l^*\ _2$	Iter.
1	$ q \leq \bar{q}, F_{cart} \leq \bar{F}, v^* \leq 0$ (only $\mathcal{N}ADM_2$ and $\mathcal{N}IPM_2$)	$6.6 \cdot 10^{-2}$	1	0	1	$1.7 \cdot 10^{-6}$	33	$5.0 \cdot 10^{-5}$	1
2	$f_{dyn}(x) = v_2$	$2.9 \cdot 10^{-5}$	2	$4.8 \cdot 10^{-7}$	81	$2.8 \cdot 10^{-8}$	129	$2.0 \cdot 10^{-8}$	152
3	$f_{ef}(q) = v_3$	3.90	371	3.19	78	6.0	2	3.2	1
4	$[q \ \dot{q}]^T = v_5$	268.6	1	263.0	22	175.2	2	274.2	18
5	$F_{cart} = v_6$	683.5	1	600.1	1	416.7	1	577.9	74
Σ			378		184		168		247

TABLE 4 Inverted pendulum swing-up: optimal slacks v^* and number of outer iterations (Iter.) per priority level for a NL-HLSP with $p = 5$ and $n = 375$ ($n^* = 450$ for $\mathcal{N}ADM_2$ and $\mathcal{N}IPM_2$).

6.5 | Swing-up of inverted pendulum

In this two-dimensional simulation setting, we aim to compute a swing up motion of a freely rotating pendulum mounted to a horizontally moving cart⁵⁷. The control input is given by the force F_{cart} (limit $\bar{F} = 100$ N) applied horizontally to the cart of weight 0.1 kg. The pendulum is of length 0.25 m and of mass 0.1 kg. The coordinates $q = [q_1 \ q_2]^T = [0 \ \pi]^T$ describe the horizontal cart position q_1 and the pendulum angle q_2 . $q_2 = 0$ corresponds to the upright pendulum position. The planning horizon is $T = 75$ ($\Delta t = 0.0025$ s) with $n = 375$ ($n^* = 450$ according to Sec. 5.6 with under-actuation $n_{ua} = 1$, for $\mathcal{N}ADM_2$ and $\mathcal{N}IPM_2$).

The hierarchy of this trajectory optimization problem is given in Tab 4. On the first two levels, joint angle and torque limits and the dynamics equations integrated by the explicit Euler method are defined. The third level contains a positioning task where the desired Cartesian position of the pendulum tip is set to 0 m and 0.5 m for the horizontal and vertical axis, respectively.

The solver data is given in Tab. 4 and Fig. 13. $\mathcal{N}IPM_2$ is able to compute a swing-up motion in 4.1 s. The corresponding robot values are depicted in Fig. 14. It can be seen from the bottom graph that F_{cart} exhibits slight jittering at the upright position from control iteration 45 onwards. This is due to SOI approximations (linear Lagrange multipliers, regularization, numerical errors, ...). A slightly worse solution is delivered by H-GUROBI (position task error of $\|v_3\|_2^2 = 3.2$ instead of $\|v_3\|_2^2 = 3.19$ for $\mathcal{N}IPM_2$). The computation time is significantly longer at 20.3 s and 247 outer iterations. This is due to the higher number of non-zeros

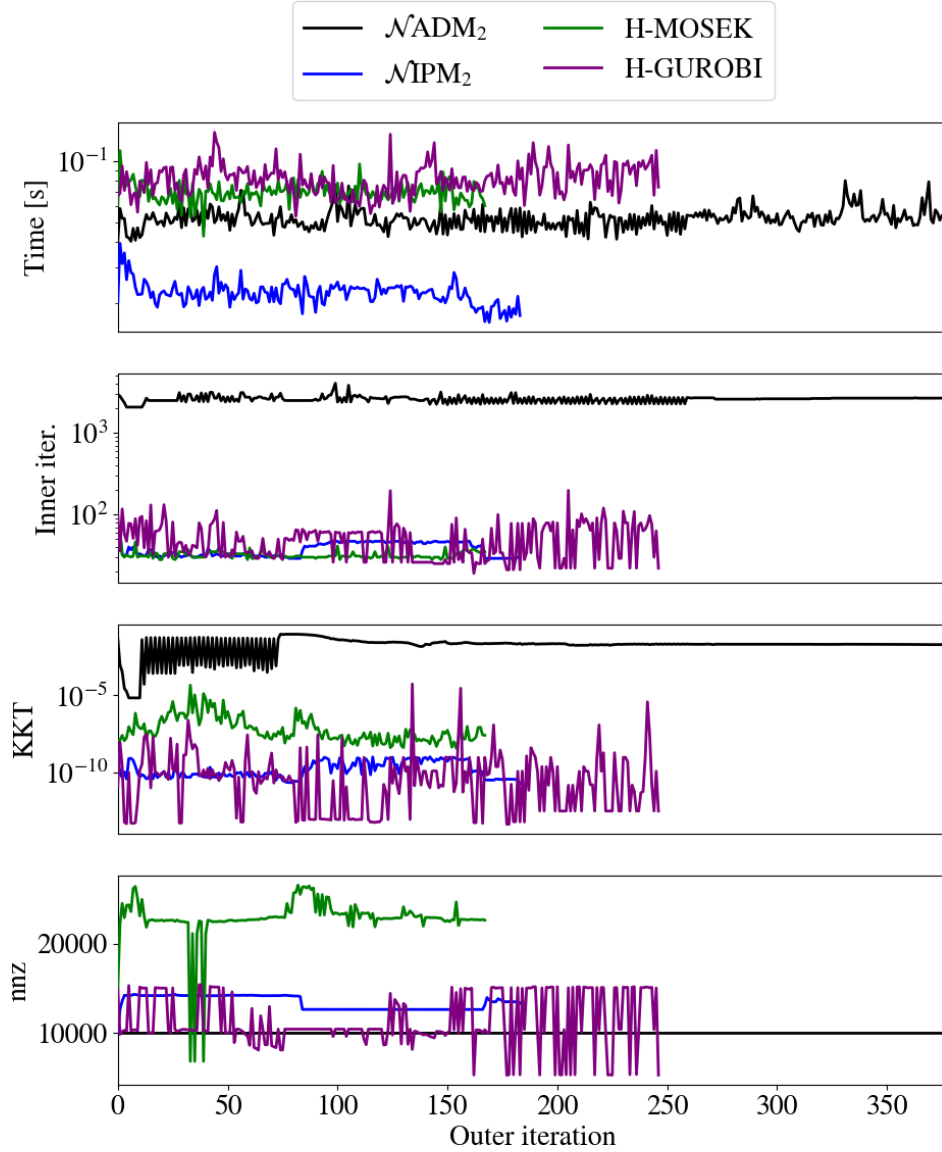


FIGURE 13 Inverted pendulum swing-up, data for the different HLSP sub-solvers over S-HLSP outer iteration: computation times per HLSP solve, number of inner iterations, KKT residuals and overall number of non-zeros handled throughout the whole hierarchy.

handled throughout the hierarchy and confirms the efficiency of the turnback algorithm in the under-actuated case. Both solvers $\mathcal{N}ADM_2$ and H-MOSEK struggle to resolve level 3 of the HLSP sub-problem (see graph of KKT norm in Fig. 13), giving rise to worse convergence in the corresponding NL-HLSP ($\|v_3^*\|_2^2 = 3.9$ for $\mathcal{N}ADM_2$ and $\|v_3^*\|_2^2 = 6.0$ for H-MOSEK; note that we increased the maximum number of inner iterations to 2000 for $\mathcal{N}ADM_2$).

6.6 | Jump of robot dog Solo12

In this example, we compute a whole-body jumping motion of the robot dog Solo12 over a horizon of $T = 15$ with $\Delta t = 0.01$ s. The base of the robot is not actuated ($n_{ua} = 6$). We use the turnback algorithm for under-actuated systems as described in Sec. 5.6. This effectively increases the number of variables from $n = 900$ to $n^* = 990$.

The control hierarchy is given in Tab. 5. Joint, torque and contact friction cone ($\mu = 1$) constraints enforce robot safety and physicality. The jumping motion is enforced by removing contacts from the dynamics equation f_{dyn} in the time interval from

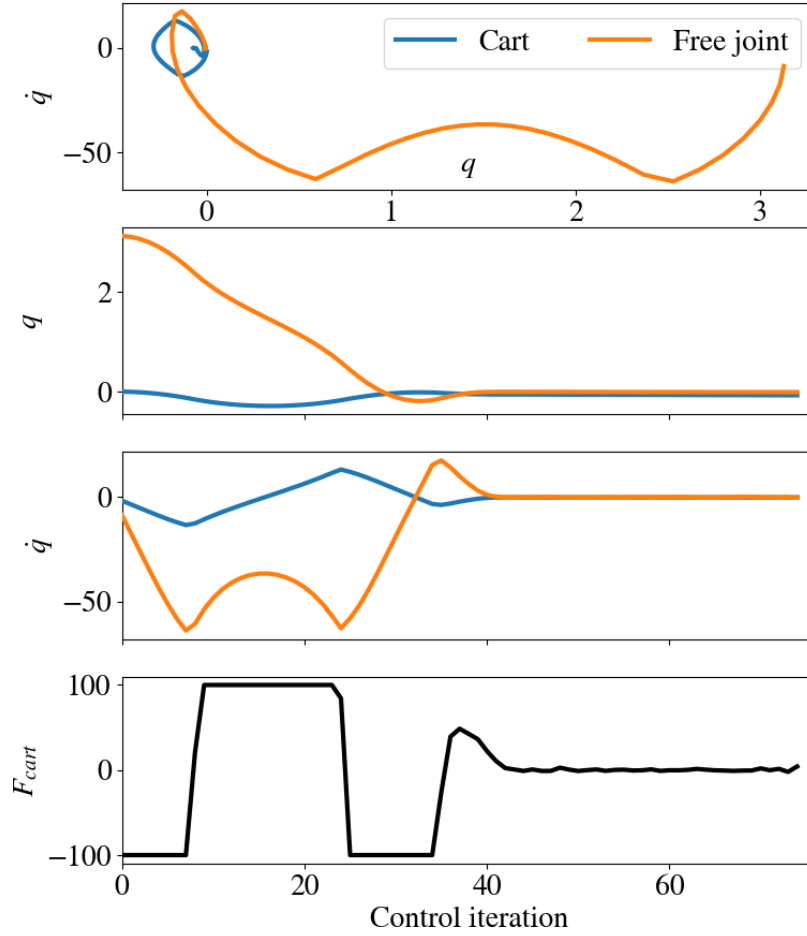


FIGURE 14 Inverted pendulum swing-up, $\hat{\mathcal{M}}\text{IPM}_2$: position and velocity q and \dot{q} of the cart and the freely swinging pendulum, force F_{cart} applied to the cart.

l	$f_l(x) \leq v_l$	$\hat{\mathcal{N}}\text{ADM}_2$ (7.0 s)		$\hat{\mathcal{M}}\text{IPM}_2$ (12.2 s)		H-MOSEK (80.8 s)		H-GUROBI (7.4 s)	
		$\ v_l^*\ _2$	Iter.	$\ v_l^*\ _2$	Iter.	$\ v_l^*\ _2$	Iter.	$\ v_l^*\ _2$	Iter.
1	$ q \leq \bar{q}, \tau \leq \bar{\tau}, \tau^* \leq 0$ (only $\hat{\mathcal{N}}\text{ADM}_2$ and $\hat{\mathcal{M}}\text{IPM}_2$), $\gamma_z \geq 0$	$4.8 \cdot 10^{-4}$	1	$1.4 \cdot 10^{-7}$	1	0	1	$4.7 \cdot 10^{-5}$	33
2	$f_{\text{dyn}}(x) = v_2$	$1.2 \cdot 10^{-8}$	5	$2.8 \cdot 10^{-6}$	2	$4.8 \cdot 10^{-9}$	55	$3.1 \cdot 10^{-7}$	1
3	$\sqrt{\gamma_x^2 + \gamma_y^2} \leq \mu\gamma_z$	$1.0 \cdot 10^{-5}$	4	0	28	0	30	$1.4 \cdot 10^{-8}$	1
4	$f_{\text{ef}}(q) = v_{3,1}$ $10^{-3} \cdot q_{\text{act}} = v_{3,2}$	$4.9 \cdot 10^{-3}$ $2.1 \cdot 10^{-3}$	6	$5.2 \cdot 10^{-4}$ $4.4 \cdot 10^{-4}$	12	$2.5 \cdot 10^{-4}$ $1.7 \cdot 10^{-3}$	30	$1.8 \cdot 10^{-2}$ $2.4 \cdot 10^{-3}$	3
5	$\dot{h}(x) = v_4$	83.0	35	130.5	14	176.7	51	89.5	3
6	$\begin{bmatrix} q^T & \dot{q}^T \end{bmatrix}^T = v_5$	49.0	1	36.8	4	87.3	5	38.6	1
7	$\begin{bmatrix} \tau^T & \gamma^T \end{bmatrix} = v_6$	36.1	1	108.1	12	315.5	7	19.0	1
Σ			54		74		180		44

TABLE 5 Solo12 jump: optimal slacks v^* and number of outer iterations (Iter.) per priority level for a NL-HLSP with $p = 6$ and $n = 900$ ($n^* = 990$ for $\hat{\mathcal{N}}\text{ADM}_2$ and $\hat{\mathcal{M}}\text{IPM}_2$).

$t = 4$ to $t = 13$. The contacts f_{ef} after landing are shifted by $[0.05 \ 0.05 \ 0]$ m compared to the initial stance. On the same level, we add a posture regularization task with low weight on the actuated robot joints. Furthermore, the evolution of the angular momentum \dot{h} is regularized to zero. This promotes a stable flight phase.

The results are given in Fig. 15. Our solver $\hat{\mathcal{N}}\text{ADM}_2$ (7.0 s, 54 outer iterations) solves the NL-HLSP faster than $\hat{\mathcal{M}}\text{IPM}_2$ (12.2 s, 74 outer iterations), H-MOSEK (80.8 s, 180 outer iterations) and H-GUROBI (7.4 s, 44 outer iterations). However, the HLSP

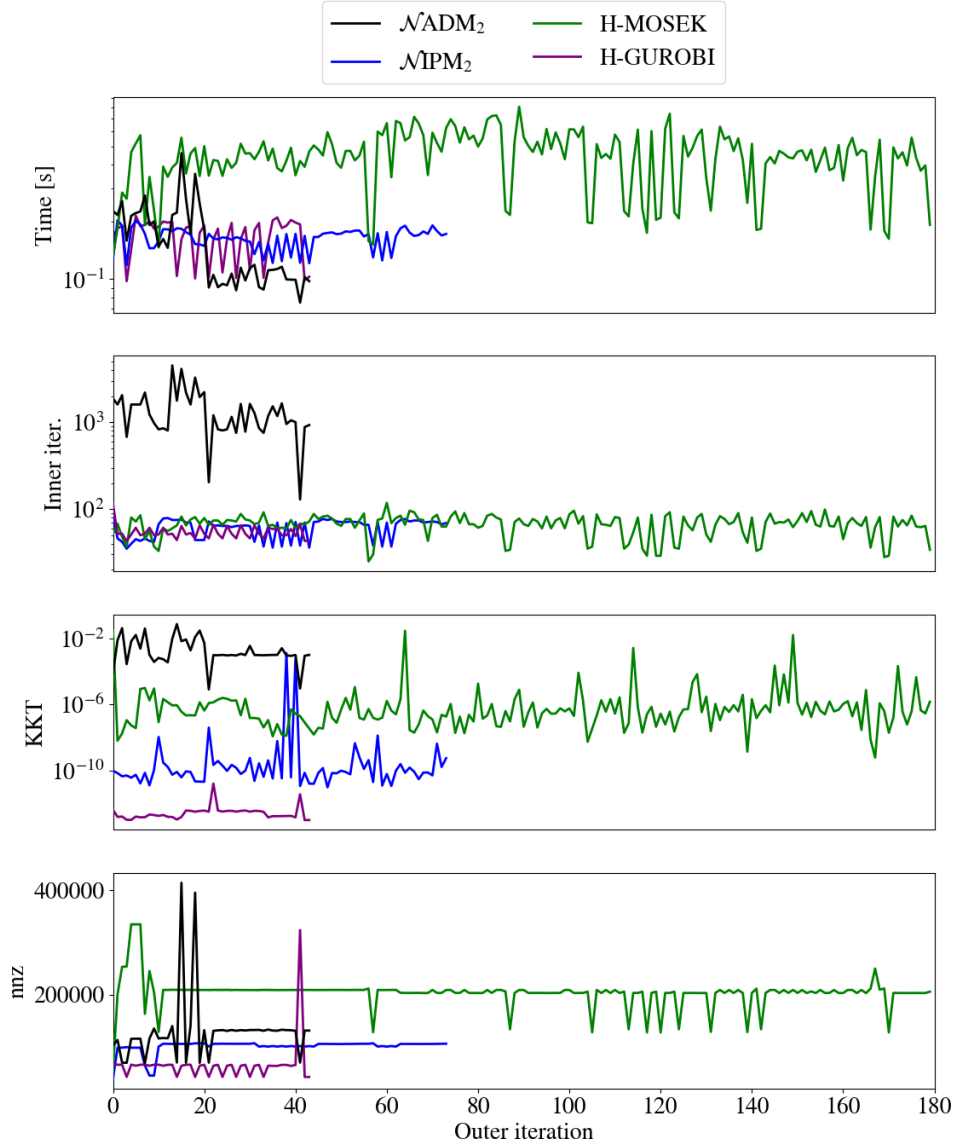


FIGURE 15 Solo12 jump, data for the different HLSP sub-solvers over S-HLSP outer iteration: computation times per HLSP solve, number of inner iterations, KKT residuals and overall number of non-zeros handled throughout the whole hierarchy.

solve times fluctuate with SOI activations which introduce a large number of non-zeros due to the SOI of the dynamics equations (see nnz peaks in bottom graph of Fig. 15). This could for example be avoided by generating strictly dynamically feasible outer iterates, which has been proposed in a SQP trust-region method⁵⁸. In contrast, SOI activations of the dynamics constraints do not occur for $\mathcal{M}IPM_2$, which can be contributed to the high-accuracy nature of the solver. Combined with the lower number of iterations (~ 200 times lower than the ones of $\mathcal{N}ADM_2$), the HLSP sub-problem solve times are limited to under 0.2 s. The HLSP times for H-GUROBI are competitive. The solver is efficiently warm-started by the primal of the full-rank level 5, such that levels below converge with zero iterations. In comparison, both the solvers $\mathcal{N}ADM_2$ and $\mathcal{M}IPM_2$ exhibit significantly faster HLSP solve times than H-MOSEK, which resolves every level of the hierarchy. This emphasizes the advantage of projection based methods for active constraints elimination in order to resolve sparse problems with lower number of non-zeros. This also indicates the high sparsity introduced by the turnback nullspace for under-actuated systems with virtual controls as described above.

The high accuracy solver $\mathcal{M}IPM_2$ achieves the greatest error reduction consistently throughout the priority levels, see Tab. 5. The corresponding robot torso trajectory and contact forces are depicted in Fig. 16 and Fig. 17, respectively. It can be observed that for a shorter time horizon of 0.14 s, the posture task is infeasible and the torso is only moved around 0.025 m into the desired direction. The robot feet however are translated to their desired position, as can be seen from the corresponding error reduction

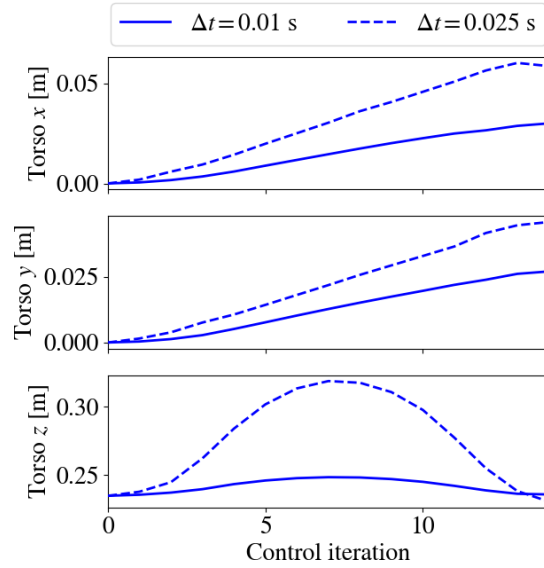


FIGURE 16 Solo12 jump, $\mathcal{M}IPM_2$, torso trajectory, dashed for $\Delta t = 0.025$ s.

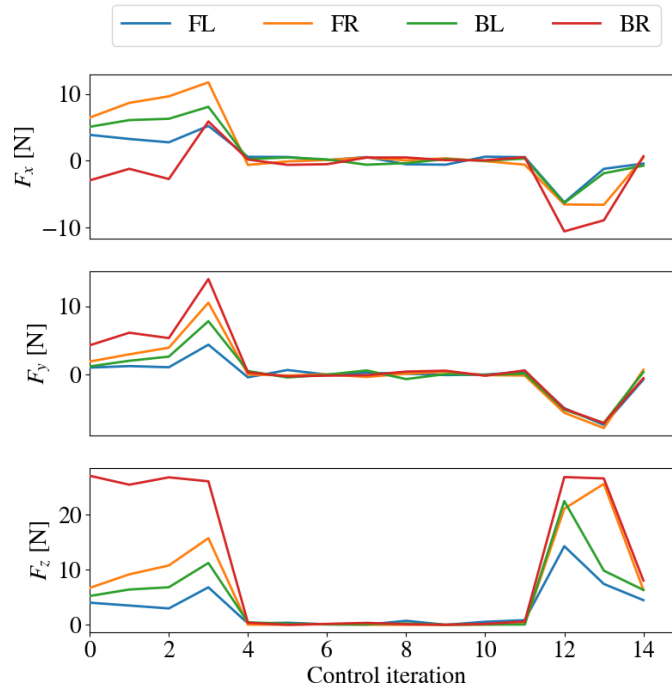


FIGURE 17 Solo12 jump, $\mathcal{M}IPM_2$, contact forces.

of f_{ef} in Tab. 5. With a longer time horizon of 0.35 s with $\Delta t = 0.025$ s, the robot manages a larger torso transfer and shifts its body by the desired amount of 0.05 m.

7 | CONCLUSION

In this article, we proposed several tools for efficiently solving prioritized trajectory optimization problems in robot motion planning. We designed a threshold adaptation strategy in order to appropriately activate or deactivate SOI. This promotes optimality of the NL-HLSP and numerical stability when solving its HLSP approximation. We proposed the ADMM solver

$\mathcal{N}^{\text{ADM}}_2$ for efficiently solving HLSP's. It is based on a reduced Hessian formulation with nullspace basis projections of active constraints. We directed our attention to problems of block-diagonal structure arising from optimal control formulations. Accordingly, we designed a sparsity leveraging turnback nullspace basis of upper bounded bandwidth for dynamics discretized by Euler integration.

The proposed HLSP solver's efficiency was demonstrated within the S-HLSP framework to solve NL-HLSP's. The reduced Hessian formulation significantly reduces the number of non-zeros handled throughout the hierarchy with a sufficient number of priority levels. With a limited number of inner iterations, this enables a fast search for an optimal point of lower accuracy. We showed how such a point can be used to warm-start S-HLSP to find a high accuracy solution. Applicability to PTO for robot scenarios with multi-stage constraints was demonstrated for a fully-actuated and under-actuated robots. In the latter case, we showed how the high sparsity of the turnback nullspace for fully-actuated robots can be maintained in the case of under-actuation. The SOI threshold adaptation strategy was shown to adjust to constraint infeasibility even from a far off starting point.

Our method sparsely and once and for all eliminates the dynamics constraints from the HLSP sub-problems. In future work, we aim to extend the turnback algorithm to higher-order integration methods. Furthermore, as a non-recursive method, our method is associated with a high memory footprint as all control time-steps need to be handled in one big solution system. We therefore aim to implement recursive formulations of our method to achieve higher solver maturity, for example based on DDP principles. While this is associated with low bandwidth and high degree of sparsity, special attention needs to be paid with respect to multi-stage constraints and high number of priority levels, as the recursions need to be computed for every level.

FUNDING

This work is partly supported by the Schaeffler Hub for Advanced Research at Nanyang Technological University, under the ASTAR IAF-ICP Programme ICP1900093. This work is partly supported by the Research Project I.AM. through the European Union H2020 program (GA 871899).

CONFLICT OF INTEREST DISCLOSURE

The authors have no relevant financial or non-financial interests to disclose.

DATA AVAILABILITY

Data generated by our algorithms S-HLSP and the HLSP sub-problem solvers are available from the corresponding author on request.

References

1. Serali HD, Soyster AL. Preemptive and nonpreemptive multi-objective programming: Relationship and counterexamples. *Journal of Optimization Theory and Applications*. 1983;39:173-186.
2. Lai L, Fiaschi L, Cococcioni M, Deb K. Pure and Mixed Lexicographic-Paretian Many-Objective Optimization: State of the Art. *Natural Computing*. 2022.
3. Escande A, Mansard N, Wieber PB. Hierarchical quadratic programming: Fast online humanoid-robot motion generation. *The International Journal of Robotics Research*. 2014;33(7):1006–1028.
4. Djeha M, Gergondet P, Kheddar A. Robust Task-Space Quadratic Programming for Kinematic-Controlled Robots. *IEEE Transactions on Robotics*. 2023;39(5):3857-3874.
5. Pfeiffer K, Escande A, Gergondet P, Kheddar A. The Hierarchical Newton's Method for Numerically Stable Prioritized Dynamic Control. *IEEE Transactions on Control Systems Technology*. 2023:1-14.
6. Pfeiffer K, Kheddar A. Sequential Hierarchical Least-Squares Programming for Prioritized Non-Linear Optimal Control. 2024.

7. Meduri A, Shah P, Viereck J, Khadir M, Havoutis I, Righetti L. BiConMP: A Nonlinear Model Predictive Control Framework for Whole Body Motion Planning. *IEEE Transactions on Robotics*. 2023;1-18.
8. Kaneko I, Lawo M, Thierauf G. On computational procedures for the force method. *International Journal for Numerical Methods in Engineering*. 1982;18:1469-1495.
9. Mayne D. A Second-order Gradient Method for Determining Optimal Trajectories of Non-linear Discrete-time Systems. *International Journal of Control*. 1966;3(1):85-95.
10. Nocedal J, Wright SJ. *Numerical Optimization*. New York, NY, USA: Springer. second ed., 2006.
11. Forsgren A, Gill PE, Wright MH. Interior Methods for Nonlinear Optimization. *SIAM Review*. 2002;44(4):525–597.
12. Hestenes MR. Multiplier and gradient methods. *Journal of Optimization Theory and Applications*. 1969;4:303-320.
13. Boyd S, Parikh N, Chu E, Peleato B, Eckstein J. Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers. *Foundations and Trends in Machine Learning*. 2011;3:1-122.
14. Boggs P, W. Tolle J. Sequential Quadratic Programming. *Acta Numerica*. 1995;4:1-51.
15. Fletcher R, Leyffer S, Toint PL. On the Global Convergence of a Filter–SQP Algorithm. *SIAM Journal on Optimization*. 2002;13(1):44-59.
16. Armijo L. Minimization of functions having Lipschitz continuous first partial derivatives.. *Pacific Journal of Mathematics*. 1966;16(1):1 – 3.
17. Wächter A, Biegler LT. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*. 2006;106:25-57.
18. Wang Y, Boyd S. Fast Model Predictive Control Using Online Optimization. *IEEE Transactions on Control Systems Technology*. 2010;18(2):267-278.
19. Pavlov A, Shames I, Manzie C. Interior Point Differential Dynamic Programming. *IEEE Transactions on Control Systems Technology*. 2021;PP:1-8.
20. Jallet W, Bambade A, Arlaud E, El-Kazdadi S, Mansard N, Carpentier J. PROXDDP: Proximal Constrained Trajectory Optimization. working paper or preprintnot; 2023.
21. Geisert M, Del Prete A, Mansard N, Romano F, Nori F. Regularized Hierarchical Differential Dynamic Programming. *IEEE Transactions on Robotics*. 2017;33(4):819-833.
22. Wang Y, Liu Y, Leibold M, Buss M, Lee J. Hierarchical Incremental MPC for Redundant Robots: a Robust and Singularity-Free Approach. *IEEE Transactions on Robotics*. 2024:1-20.
23. Tazaki Y, Suzuki T. Constraint-Based Prioritized Trajectory Planning for Multibody Systems. *IEEE Transactions on Robotics*. 2014;30(5):1227-1234.
24. Topcu A. A contribution to the systematic analysis of finite element structures using the force method. *Ph.D. thesis, University of Essen, Germany*. 1979.
25. Gilbert JR, Heath MT. Computing a Sparse Basis for the Null Space. *SIAM Journal on Algebraic Discrete Methods*. 1987;8(3):446-459.
26. Yang J, Meijer T, Dolk V, Jager Bd, Heemels W. A System-Theoretic Approach to Construct a Banded Null Basis to Efficiently Solve MPC-Based QP Problems. In: 2019:1410-1415.
27. Pfeiffer K, Escande A, Kheddar A. Singularity Resolution in Equality and Inequality Constrained Hierarchical Task-Space Control by Adaptive Nonlinear Least Squares. *IEEE Robotics and Automation Letters*. 2018;3(4):3630-3637.
28. Cococcioni M, Pappalardo M, Sergeyev YD. Lexicographic multi-objective linear programming using grossone methodology: Theory and algorithm. *Applied Mathematics and Computation*. 2018;318:298-311. Recent Trends in Numerical Computations: Theory and Algorithms.
29. Chiaverini S. Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators. *IEEE Transactions on Robotics and Automation*. 1997;13(3):398–410.
30. Higham N. Computing the Polar Decomposition with Applications. *SIAM Journal on Scientific and Statistical Computing*. 1986;7(4):1160-1174.
31. Golub GH, Van Loan CF. *Matrix Computations (3rd Ed.)*. Baltimore, MD, USA: Johns Hopkins University Press, 1996.
32. Broyden CG. The Convergence of a Class of Double-rank Minimization Algorithms. *Journal of the Mathematics and its Applications*. 1970;6:76–90.
33. Moré JJ. The Levenberg-Marquardt algorithm: Implementation and theory. In: Watson GA., ed. *Numerical Analysis* Springer Berlin Heidelberg 1978; Berlin, Heidelberg:105-116.
34. Pfeiffer K, Escande A, Righetti L. \mathcal{N} IPM-HLSP: an efficient interior-point method for hierarchical least-squares programs. *Optimization and Engineering*. 2023:1573-2924.

35. Liu Z, Guo F, Wang W, Wu X. A distributed parallel optimization algorithm via alternating direction method of multipliers. *IET Control Theory & Applications*. 2023;17(7):896-905. doi: <https://doi.org/10.1049/cth2.12421>
36. Dang TV, Ling KV, Maciejowski J. Banded Null Basis and ADMM for Embedded MPC. *IFAC-PapersOnLine*. 2017;50(1):13170-13175. 20th IFAC World Congress.
37. Stellato B, Banjac G, Goulart P, Bemporad A, Boyd S. OSQP: an operator splitting solver for quadratic programs. *Mathematical Programming Computation*. 2020;12(4):637–672.
38. Eckstein J, Bertsekas D. On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators. *Mathematical Programming*. 1992;55:293-318.
39. Ghadimi E, Teixeira A, Shames I, Johansson M. Optimal Parameter Selection for the Alternating Direction Method of Multipliers (ADMM): Quadratic Problems. *IEEE Transactions on Automatic Control*. 2014.
40. Rao AV, Benson DA, Darby C, et al. Algorithm 902: GPOPS, A MATLAB Software for Solving Multiple-Phase Optimal Control Problems Using the Gauss Pseudospectral Method. *ACM Trans. Math. Softw.*. 2010;37(2).
41. Li Z, Ge SS. Adaptive robust controls of biped robots. *IET Control Theory & Applications*. 2013;7(2):161-175. doi: <https://doi.org/10.1049/iet-cta.2012.0066>
42. Carpentier J, Mansard N. Analytical Derivatives of Rigid Body Dynamics Algorithms. *Robotics: Science and Systems XIV*. 2018.
43. Gifftthaler M, Neunert M, Stäuble M, Buchli J, Diehl M. A Family of Iterative Gauss-Newton Shooting Methods for Nonlinear Optimal Control. 2017.
44. Singh S, Russell RP, Wensing PM. Efficient Analytical Derivatives of Rigid-Body Dynamics Using Spatial Vector Algebra. *IEEE Robotics and Automation Letters*. 2021;7:1776-1783.
45. Knight PA, Ruiz D, Uçar B. A Symmetry Preserving Algorithm for Matrix Scaling. *SIAM Journal on Matrix Analysis and Applications*. 2014;35(3):931-955.
46. Udvardia F, Schutte A. Equations of motion for general constrained systems in Lagrangian mechanics. *Acta Mech*. 2010;213.
47. Lu TT, Shiou SH. Inverses of 2×2 block matrices. *Computers & Mathematics with Applications*. 2002;43(1):119-129.
48. Pfeiffer K, Righetti L. \mathcal{N} /IPM-MPC: An Efficient Null-Space Method Based Interior-Point Method for Model Predictive Control. 2021.
49. Guennebaud G, Jacob B, others . Eigen v3. <http://eigen.tuxfamily.org>; 2010.
50. Carpentier J, Saurel G, Buondonno G, et al. The Pinocchio C++ library – A fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives. In: 2019.
51. Bell M. CppAD: A package for differentiation of C++ algorithms (2024/02/01). <http://www.coin-or.org/CppAD>; 2024.
52. Paige CC, Saunders Ma. LSQR: An Algorithm for Sparse Linear Equations and Sparse Least Squares. *ACM Transactions on Mathematical Software*. 1982;8(1):43–71.
53. Gill PE, Murray W, Saunders MA, Wright MH. Maintaining LU factors of a general sparse matrix. *Linear Algebra and its Applications*. 1987;88-89:239-270.
54. ApS M. *MOSEK Fusion API for C++ 10.1.12*. 2019.
55. Gurobi Optimization, LLC . Gurobi Optimizer Reference Manual. 2023.
56. Chen T, Lasserre JB, Magron V, Pauwels E. A sublevel moment-SOS hierarchy for polynomial optimization. *Computational Optimization and Applications*. 2021;81:31 - 66.
57. Cavdaroglu M, Olgac N. Trajectory tracking of cart-pendulum dynamics using multiple time-delayed feedback. *Control Theory & Applications, IET*. 2008;2:458 - 466. doi: [10.1049/iet-cta:20070242](https://doi.org/10.1049/iet-cta:20070242)
58. Tenny MJ, Wright SJ, Rawlings JB. Nonlinear Model Predictive Control via Feasibility-Perturbed Sequential Quadratic Programming. *Computational Optimization and Applications*. 2004;28(1):87-121.