Pose Optimization of Task-Redundant Robots in Second-Order Rest-to-Rest Motion with Cascaded Dynamic Programming and Nullspace Projection

Moritz Schappler[1][0000−0001−7952−7363]

Leibniz University Hannover, Institute of Mechatronic Systems,
An der Universität 1, 30823 Garbsen, Germany
moritz.schappler@imes.uni-hannover.de

Abstract. An optimal trajectory for the redundant coordinate for robots in tasks with rotational symmetry such as machining has to be found to ensure good performance and overall feasibility. Due to high nonlinearity of performance criteria especially for parallel robots a sole local optimization may lead to infeasible solutions for large-scale motion. A pointing task consisting of multiple rest-to-rest trajectories with given dense sample times is regarded as given. Constraints regarding system limits on position, velocity and acceleration have to be met. The proposed algorithm combines nullspace projection for local optimization between the rest poses with dynamic programming at the rest poses in a cascaded scheme to optimize the rotation around the tool axis. Applications to other types of redundancy are also possible. The proposed local/global optimization scheme only needs wide discretization of the redundant coordinate and therefore has acceptable computational performance for offline optimization of robot motion. It is able to find feasible and near-optimal trajectories for a six-degree-of-freedom (DoF) parallel robot in several exemplary five-DoF tractories with many constraints.

Keywords: Robot manipulator · Task redundancy · Inverse kinematics · Trajectory optimization · Dynamic programming · Nullspace projection

1 Introduction and State of the Art

Resolution of kinematic redundancy is a persistent topic in robotics research. Task redundancy or functional redundancy is a special case if the task requires less DoF than are controlled by the end effector in it's operational space regardless of the dimension of the joint space. It is explicitly relevant for (fully) parallel robots where not the joint space as for serial robots, but the operational space of the moving platform is the essential structural kinematic characteristic. The focus on the following discussion of related work and of the paper's examples is therefore put on parallel robots. However, the proposed approach of this paper is not restricted to task redundancy or parallel robots and can be directly transferred to serial robots with one DoF of redundancy.

1.1 State of the Art and Related Work

Kinematic redundancy can be distinct in intrinsic redundancy, (e.g. 7-DoF serial robots in 6-DoF tasks), and task redundancy, as e.g. for 6-DoF robots in 5-DoF tasks. The degree of intrinsic [task] redundancy is the excess of dimension of the joint space [operational space] over the task space. A formal definition of redundancy is given in [29, 11, 15] for serial robots and in [8] for parallel robots.

Applications for task redundancy are mainly tasks with rotational symmetry of the tool or the process in general. Machining tasks using serial robots are discussed for milling in [34, 18, 19] or for drilling in [35]. Other examples using serial robots are arc welding [11] and fiber placement for composite materials [5]. The main difference of these tasks from a kinematic point of view is that milling tasks (and arc welding) usually require redundancy optimization for a continuous trajectory (which is also the focus of this paper) and drilling (or spot welding) only requires an optimization for approaching the pointing pose. Examples for parallel robots' task redundancy with more than one redundant DoF are end milling [30] or milling with a spherical cutter [32], where all three rotational coordinates can be subject to optimization.

The *objective of exploiting the redundancy* is for once to improve feasibility by avoiding joint limits [11, 35] or singularities. Obstacle collision avoidance can be implemented by minimizing corresponding potential functions [21, 10, 4]. Singularity avoidance can be implemented as an optimization objective e.g. by
- the joint space distance to the first joint configuration that violates a parameter of singularity [11],
- the squared condition number of the robot Jacobian [35, 15, 2],
- the condition number of the parallel robot forward kinematics Jacobian [7],
- a fraction containing all singular values of the Jacobian [26],
- the Jacobian's determinant [1]
- directly using the Jacobian's condition number [28].

The homogenized pose error can be minimized as a measure for accuracy as well as singularities [12], Other optimization criteria may be rather specific such as milling process stability [19]. In general, any optimization criterion dependent on the robot configuration can be used, with certain restrictions to continuous differentiability depending on the optimization method.

The potential function of the optimization objective can be visualized well over a trajectory for the case of *redundancy of degree one*. This *performance (criterion) map* was introduced by Wenger for serial robots and parallel robots, as summarized in [25], there termed "feasibility map". This map is especially useful for visual inspection in the case of large-scale motion relative to the robot workspace size. An exemplary map is given Fig. 1 for a hexapod robot with redundant orientation coordinate φ_z, which may be the irrelevant rotation around a tool axis. The map was e.g. used in [25] to create cycloidic trajectories directly in the performance map to plan changes between the 2^3 working modes (elbow configurations) of a planar parallel robot. A "robot transmission ratio map" was used in [34] to optimize the trajectory for a six-axis serial robot in a milling task. Maps called "state space grids" were used in [4] for a seven-DoF serial robot in

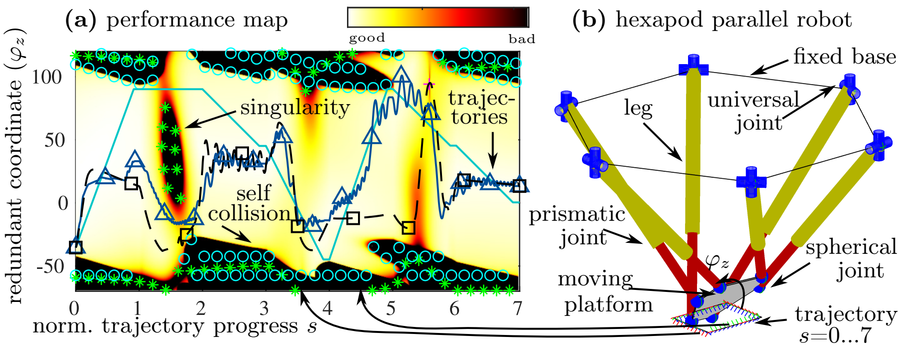

Fig. 1. Exemplary surface map of a performance criterion **(a)** for a parallel robot **(b)**

six-DoF tasks. The maps for the 2^3 extended aspects (e.g. base, elbow and hand configurations) were created for one joint angle as the redundancy parameter.

For *two degrees of redundancy* the heatmap visualisation for the trajectory is not possible any more and a two-dimensional map can be given for single poses of the robot. This has been shown for seven-DoF robots in five-DoF tasks such as medical laser operations [27] or by using an additional rotary table for an industrial robot in five-DoF milling tasks [18]. The performance map is then build by discretization of one joint axis and the end effector rotation.

Different geometric methods for redundancy resolution are available, mainly with the goal to separate the kinematic formulation into task and redundant coordinates for the subsequent optimization. A non-exhaustive list comprises

- task space orthogonal decomposition [15, 2] and twist decomposition [11],
- separation of joint coordinates in redundant and non-redundant on position level for serial [23] and parallel [16] robots or on velocity level [24],
- giving a parametric form of the dynamics equation with the redundant end effector coordinate [22],
- expressing the end effector angular velocity in the local frame and removing the last component corresponding to the redundant coordinate [36, 24] (for serial robots), trivial for the planar case [1] (e.g. for a planar parallel robot),
- formulating the inverse kinematics on position level and exploiting nonlinear orientation to eliminate the redundant task space coordinate by using a Z-Y-X Tait-Bryan-angle residual [28] (for position level and higher order).

For online optimization and fast offline planning nullspace projection is the most efficient *optimization algorithm* for the differential inverse kinematics. It is the basis for many works on serial robots using locally optimal redundancy resolution [21, 3, 11, 36, 24, 14]. Due to more sophisticated modeling, *nullspace projection* is more scarce in parallel robot inverse kinematics literature. The method is used mainly for kinematic redundancy [26] and a review on "redundancy in parallel mechanisms" [8] does not mention the case of task redundancy at all. It is introduced e.g. in [1] for the trajectory of a planar parallel robot and in the author's previous work [28] generally for spatial (fully) parallel robots.

Most formulations presented above are on *velocity level (first order)* which hinders trajectory optimization using acceleration and actuator force constraints like in [26], which can be avoided by an *acceleration-level (second-order)* formulation. Using only proportional feedback of the optimization objective potential there as in [21, 1] may lead to unwanted oscillations, which might be encountered by a PD feedback [3, 28] or an analytic derivative of the nullspace projection [24].

Other approaches than nullspace projection for trajectory redundancy optimization can be chosen from various *existing optimization algorithms* such as

- SQP [15] (for an industrial robot, without input or state constraints),
- the Simplex algorithm and quadratic programming [22] (hexapod example),
- discrete optimization of a redundant coordinate at rest poses of a kinematically redundant planar parallel robot [12],
- interval analysis [16] (at the example of a hexapod robot),
- (binary) small-angle perturbation [6] of the redundant coordinate of a task-redundant planar parallel robot.
- bracketing and bisection, Brent-Dekker method [35],
- genetic algorithms (GA) [30] (for three redundant coordinates of a hexapod).

While apart from the heuristic GA, the aforementioned algorithms also are only locally optimal like nullspace projection, *dynamic programming* (DP) is a deterministic *global optimization* algorithm. It is based on discretization of the optimization variable (redundant coordinate) on discrete stages (time steps). Examples for using *dynamic programming in robotics* are

- optimizing the time evolution of a given trajectory [31] as an early work,
- using Pontryagin's maximum principle for kinematic redundancy [20],
- investigation of Pareto optimality for multiple objectives for pose optimization of a seven-DoF serial robot [9],
- position-level optimization of two redundant DoF of a seven-DoF robot [27],
- wide and fine discretization of redundant joint coordinates in a static pose optimization of an industrial robot with an additional positioner [5].

For continuous trajectory optimization the method of *differential dynamic programming* (DDP) is advantageous. An overview and an elaboration on using the method on problems consisting of multiple phases is given in [13]. Each rest-to-rest motion can be considered as such a phase and each time step as a stage. Several works have used DDP in robotics, such as

- motion planning of a humanoid robot and a contribution regarding implementing box constraints in DDP [33],
- a constraint approach by an augmented Lagrangian validated at an example of reaching a target position with a seven-DoF arm [10],
- an application of DDP to task-redundant parallel robots [26] in combination with nullspace projection on velocity level.

1.2 Summary of the State of the Art and Scenario of the Paper

The main requirements on the trajectory optimization algorithm developed in this paper are motivated by machining tasks and other processes, where task redundancy of degree one exists and the desired task space trajectory of the end effector is given, including the time profile. The task is considered to consist of connected second-order rest-to-rest trajectories e.g. from G-code in CNC machining, which ensures a continuous velocity profile. Extending the resting condition also to the nullspace motion has the potential to reduce oscillations and to decouple the optimization. A feasible and desirably optimal trajectory for the redundant coordinate has to be found. A kinematic formulation is sufficient but can be extended by dynamics constraints regarding actuator force limits.

For the sake of simplicity, the *reconfiguration of the robot* between working modes is out of scope of this paper assuming a trajectory that can be performed only with one working mode. One reason is that distinct reconfiguration motion [12] reduces productive times of the machine. Reconfiguration during robot motion as discussed in [25] for parallel robots requires flipping the elbow configuration of a passive joint, which might not be possible to detect by sensors. Further, this motion leads through kinematic singularities, which can have unwanted side-effects. Additionally, the complete discretization of a performance map for all working modes of the robot is necessary for each task space trajectory. Ideally, the computation of the performance map is no requirement for obtaining the full trajectory and is only used for optional visualization.

As stated in [13], usually DDP approaches consider only one phase, i.e. one robot rest-to-rest motion. Combining rest-to-rest trajectories presents a special case regarding optimization, which has high practical relevance. DDP requires computing the variational formulation of the optimization problem with Jacobian and eventually Hessian w.r.t. input and state variables. This can be omitted by using just local optimization which requires only robot Jacobians and gradients of their performance criteria which are easy to obtain. Performing an unknown number of iterations of forward and backward passes in DDP can be avoided by using DP. In summary a trajectory optimization algorithm below the complexity of DDP for the envisioned use case scenario would increase robot performance in the given task.

1.3 Contribution of the Paper

The optimization of multiple rest-to-rest (i.e. multi-phase) trajectories can be solved by connecting the *local optimality of nullspace projection* with the *global optimality of dynamic programming.* This allows to use the local nullspace projection scheme for the optimization of the redundant coordinate between rest poses in a fine sampling time with continuous states. The approximation of global optimality of the trajectory is achieved with dynamic programming of the same optimization variable at rest poses based on the actual obtained coordinates from the local optimization. Thereby an *extension to dynamic programming* is proposed such that state values of the optimization variable are not given as discrete values, but as an *interval* determined by the final value of the nullspace projection. The performance of this approximation depends on the quality of the local optimization and the underlying assumption of limited changes of the feasibility map during one phase.

The contributions of the paper therefore are in summary

- a cascaded scheme of dynamic programming and nullspace optimization,
- an extension to dynamic programming regarding interval optimization,
- the application of the scheme on second-order rest-to-rest robot motion,
- the extension of the second-order nullspace projection inverse kinematics approach presented in [28] to support the cascaded dynamic programming.

The problem from the author's preceding conference paper [28] is solved globally using the new dynamic programming scheme. The remainder of the paper is structured as follows. The problem description in the sense of optimization is presented in Sect. 2. The nullspace optimization scheme for the robot from [28] is revised and extended in Sect. 3. Finally, the methods are combined for trajectory optimization in Sect. 4 with simulation results shown in Sect. 5.

2 Problem Description: Task Redundancy Optimization

A robot manipulator, either serial or parallel, is able to create motion with it's end effector in the operational space. The corresponding end effector pose variable is denoted as $\boldsymbol{x}_E{=}(r_x, r_y, r_z, \varphi_x, \varphi_y, \varphi_z)^{\mathrm{T}}$ and contains position $\boldsymbol{r}$ and orientation $\boldsymbol{\varphi}$. The latter is expressed with the X-Y'-Z'' Cardan angles such that the end effector's rotation matrix is formed as ${}^0\boldsymbol{R}_E(\boldsymbol{x}){=}\boldsymbol{R}_x(\varphi_x)\boldsymbol{R}_y(\varphi_y)\boldsymbol{R}_z(\varphi_z)$. The benefit of this selection is, as elaborated in [28], that rotation around the z_E axis – by definition the tool axis – of the end effector corresponds to the last coordinate φ_z. This coordinate does not influence tasks with five DoF and can therefore be eliminated in their task space coordinates $\boldsymbol{y}_E{=}(r_x, r_y, r_z, \varphi_x, \varphi_y)^{\mathrm{T}}$.

The desired task trajectory is given as a time series of partial poses, i.e. $\boldsymbol{y}_E(t)$, which is sampled discretely as $\boldsymbol{y}_E(t_k)$ with $N+1$ samples. An at least two times continuous differentiability of the position profile (preferably an S-curve) is required, therefore also $\dot{\boldsymbol{y}}_E(t)$ and $\ddot{\boldsymbol{y}}_E(t)$ exist. Since the trajectory consists of multiple (N_R) rest-to-rest phases, at rest times t_{R_i} the condition

$$\dot{\boldsymbol{y}}_E(t_{\mathrm{R}_k}) = 0 \quad \text{for} \quad k = 0, ..., N_\mathrm{R} \tag{1}$$

has to hold. Under the assumption of task redundancy, the robot is controlled in it's operational space $\boldsymbol{x}_E$ and only a continuous one-DoF trajectory $\varphi_z(t)$ has to be generated by optimization, since $\boldsymbol{y}_E(t)$ is already given.

2.1 Static Formulation of the Optimization Problem

A static formulation of the optimization problem only for the rest poses at t_{R_k} does not include system dynamics. Therefore the state can be chosen as $x_k = \varphi_z(t_{\mathrm{R}_k})$ as the redundant coordinate at rest times. The running cost $l(x_k)$ only depends on the current state. The total cost is then defined as $J = \sum_{k=0}^{N_\mathrm{R}} l(x_k)$ and an optimal sequence of states minimizes the total cost by $\boldsymbol{X}^* = \mathrm{argmin}_{\boldsymbol{X}}\, J(\boldsymbol{X}) = \{x_0^*, x_1^*, ..., x_{N_\mathrm{R}}^*\}$. This formulation still ignores the transition between the states, which can be included as shown next or by the complete DP formulation of Sect. 4.1.

2.2 Differential Formulation of the Optimization Problem

To incorporate also states between the rest poses, a dynamic formulation can be used, such as differential dynamic programming [33]. The problem can be described by a dynamic system $\boldsymbol{x}_{i+1}{=}\boldsymbol{f}(\boldsymbol{x}_i, u_i)$, where the state is denoted by $\boldsymbol{x}_i{=}[\varphi_z(t_i), \dot{\varphi}_z(t_i)]^{\mathrm{T}}$ and the input by $u_i{=}\ddot{\varphi}_z(t_i)$. The objective is to obtain a trajectory $\{\boldsymbol{X}, \boldsymbol{U}\}$ of all N samples consisting of the control sequences $\boldsymbol{U}{=}\{u_0, u_1, ..., u_{N-1}\}$ and state sequences $\boldsymbol{X}{=}\{\boldsymbol{x}_0, \boldsymbol{x}_1, ..., \boldsymbol{x}_N\}$. The final cost is defined as $l_{\mathrm{f}}(\boldsymbol{x}_N)$, the running cost as $l(\boldsymbol{x}_i, u_i)$ and the total cost is

$$J(\boldsymbol{x}_0, \boldsymbol{U}) = \sum_{i=0}^{N} l(\boldsymbol{x}_i, u_i) + l_{\mathrm{f}}(\boldsymbol{x}_N). \tag{2}$$

The optimal sequence $\boldsymbol{U}^* = \mathrm{argmin}_{\boldsymbol{U}}\, J(\boldsymbol{x}_0, \boldsymbol{U})$ minimizes the total cost. For a rest-to-rest motion additionally the condition

$$\dot{\varphi}_z(t_{\mathrm{R}_k}) = 0 \quad \text{for} \quad k = 0, ..., N_{\mathrm{R}} \tag{3}$$

is regarded to reduce oscillations and allow pausing the complete trajectory motion at rest times. This presents a *phase constraint* between multiple phases according to [13] and complicates using DDP with classical schemes.

2.3 Hypothesis of the Paper: Decoupling of Problem Phases

It is desirable to *decouple* the problem of the single rest-to-rest phases to simplify the overall optimization problem to solve. The decoupling makes it necessary to avoid using a fine discretization of the state x_k at the coupling points (rest times). Otherwise, the coupled problem of Sect. 2.2 arises. The main requirement for the decoupling is the coincidence of global cost of the static problem of Sect. 2.1 and local costs of the rest-to-rest-problem of Sect. 2.2 solved differentially. In other words, the time interdependence of current states and future costs should be low. By this, locally optimizing the cost also leads to global optimization in general. To avoid local optima, the static optimization of Sect. 2.1 performs an adequate exploration at the rest times.

If in the robot example of this paper e.g. a feasible trajectory is the overall goal (regarding singularity, joint limits and collision avoidance), then this assumption is likely to hold. If the objective is rather a complex relation of the robot state over a long time horizon, the proposed approach will less likely converge to a global optimum. Exemplarily this can be the case for the minimization of the energy consumption over the trajectory, as local increases of kinetic or potential energy may be beneficial depending on later states.

3 Local Optimization: Second-Order Inverse Kinematics

It is assumed that the optimal sequence $\boldsymbol{U}_k^*$ from Sect. 2.2 for rest-to-rest motion k with $t_{\mathrm{R}_k} \leq t \leq t_{\mathrm{R}_{k+1}}$ can be approximated by local optimization using the locally optimal nullspace projection ("NP") scheme, yielding $\boldsymbol{U}_k^* \approx \boldsymbol{U}_{k,\mathrm{NP}}$. This assumption holds for short trajectories in terms of distance in the performance map. The claim is only qualitative and has to be evaluated for the specific task at hand. In the following, the nullspace projection scheme for the second-order inverse kinematics is wrapped up for serial and parallel robots and extensions are presented which improve using it for global optimization in the next section 4.

3.1 Kinematics Model for Serial and Parallel Robots

Due to their differences in modeling, the kinematics of serial and parallel robots are introduced separately in the following. However, the general relations can be handled interchangeably in the section thereafter on nullspace trajectory motion.

Kinematics of Serial Robots are handled in robotics textbooks like [29]. The position-level forward kinematics present a mapping of joint coordinates $\boldsymbol{q}$ and end effector coordinates $\boldsymbol{x}_E(\boldsymbol{q})$. The end effector pose[1] $\boldsymbol{x}$ is chosen as presented in Sect. 2. By analytic differentiation the linear velocity and acceleration relation

$$\dot{\boldsymbol{x}} = \boldsymbol{J}_{\boldsymbol{x}}\dot{\boldsymbol{q}} \quad \text{and} \quad \ddot{\boldsymbol{x}} = \dot{\boldsymbol{J}}_{\boldsymbol{x}}\dot{\boldsymbol{q}} + \boldsymbol{J}_{\boldsymbol{x}}\ddot{\boldsymbol{q}} \tag{4}$$

can be obtained, where $\boldsymbol{J}_{\boldsymbol{x}}$ denotes the analytic manipulator Jacobian, as opposed to the geometric one. The solution to the (non-redundant) inverse differential kinematics is obtained via

$$\ddot{\boldsymbol{q}} = \boldsymbol{J}_{\boldsymbol{x}}^{-1}(\ddot{\boldsymbol{x}} - \dot{\boldsymbol{J}}_{\boldsymbol{x}}\dot{\boldsymbol{q}}). \tag{5}$$

Kinematics of Parallel Robots are constructed differently than those for serial robots, since they consists of several kinematic chains, called legs, which connect at a moving platform. The approach to kinematics is the definition of constraints equations $\boldsymbol{\Phi}(\boldsymbol{q}, \boldsymbol{x}) = \boldsymbol{0}$, see e.g. the textbook [17]. Parallel robots' joint coordinates $\boldsymbol{q}$ also contain those of passive joints in the general case, which is regarded in the following. Time differentiation of the constraints leads to

$$\boldsymbol{\Phi}_{\partial\boldsymbol{q}}\dot{\boldsymbol{q}} + \boldsymbol{\Phi}_{\partial\boldsymbol{x}}\dot{\boldsymbol{x}} = \boldsymbol{0} \quad \text{and} \quad \dot{\boldsymbol{q}}_{\mathrm{T}} = -\boldsymbol{\Phi}_{\partial\boldsymbol{q}}^{-1}\boldsymbol{\Phi}_{\partial\boldsymbol{x}}\dot{\boldsymbol{x}} = \boldsymbol{J}_{\boldsymbol{q},\boldsymbol{x}}^{-1}\dot{\boldsymbol{x}}, \tag{6}$$

where the index "T" denotes the task solution of the differential inverse kinematics (6) as opposed to the nullspace solution "N" discussed later. The index "$\partial\boldsymbol{x}$" denotes the gradient w.r.t. the coordinate $\boldsymbol{x}$ and $\boldsymbol{J}_{\boldsymbol{q},\boldsymbol{x}}^{-1}$ is the inverse manipulator Jacobian[2] referring to all coordinates. The second time derivative is obtained by differential calculus as

$$\ddot{\boldsymbol{q}}_{\mathrm{T}} = \boldsymbol{J}_{\boldsymbol{q},\boldsymbol{x}}^{-1}\ddot{\boldsymbol{x}} + \dot{\boldsymbol{J}}_{\boldsymbol{q},\boldsymbol{x}}^{-1}\dot{\boldsymbol{x}}. \quad \text{with} \quad \dot{\boldsymbol{J}}_{\boldsymbol{q},\boldsymbol{x}}^{-1} = \boldsymbol{\Phi}_{\partial\boldsymbol{q}}^{-1}(\dot{\boldsymbol{\Phi}}_{\partial\boldsymbol{q}}\boldsymbol{\Phi}_{\partial\boldsymbol{q}}^{-1}\boldsymbol{\Phi}_{\partial\boldsymbol{x}} - \dot{\boldsymbol{\Phi}}_{\partial\boldsymbol{x}}). \tag{7}$$

[1] For the sake of readability, $\boldsymbol{x}_E$ and $\boldsymbol{y}_E$ will be written as $\boldsymbol{x}$ and $\boldsymbol{y}$ in Sect. 3.
[2] The expression is not a regular inverse, since it is not square and the non-inverse does not exist. The notation is used to unify the symbols with those of serial robots.

The manipulator Jacobian $\boldsymbol{J}_{\boldsymbol{x}}$ can be obtained by selecting the rows in $\boldsymbol{J}_{\boldsymbol{q},\boldsymbol{x}}^{-1}$ associated to the active joint's coordinates $\boldsymbol{\theta} = \boldsymbol{P}_{\boldsymbol{\theta}}\boldsymbol{q}$ and matrix inversion by

$$\boldsymbol{J}_{\boldsymbol{x}} = \left(\boldsymbol{J}_{\boldsymbol{x}}^{-1}\right)^{-1} = \left(\boldsymbol{P}_{\boldsymbol{\theta}}\boldsymbol{J}_{\boldsymbol{q},\boldsymbol{x}}^{-1}\right)^{-1}. \tag{8}$$

The forward differential equations from actuator to end effector velocities and accelerations can then be expressed like in (4) as

$$\dot{\boldsymbol{x}} = \boldsymbol{J}_{\boldsymbol{x}}\dot{\boldsymbol{\theta}} \quad \text{and} \quad \ddot{\boldsymbol{x}} = \dot{\boldsymbol{J}}_{\boldsymbol{x}}\dot{\boldsymbol{\theta}} + \boldsymbol{J}_{\boldsymbol{x}}\ddot{\boldsymbol{\theta}}. \tag{9}$$

If actuator entities are given, they can be transferred to the full joint space by

$$\dot{\boldsymbol{q}} = \boldsymbol{J}_{\boldsymbol{q},\boldsymbol{x}}^{-1}\boldsymbol{J}_{\boldsymbol{x}}\dot{\boldsymbol{\theta}} \quad \text{and} \quad \ddot{\boldsymbol{q}} = \boldsymbol{J}_{\boldsymbol{q},\boldsymbol{x}}^{-1}\boldsymbol{J}_{\boldsymbol{x}}\ddot{\boldsymbol{\theta}} + \boldsymbol{J}_{\boldsymbol{q},\boldsymbol{x}}^{-1}\dot{\boldsymbol{J}}_{\boldsymbol{x}}\dot{\boldsymbol{\theta}} + \dot{\boldsymbol{J}}_{\boldsymbol{q},\boldsymbol{x}}^{-1}\boldsymbol{J}_{\boldsymbol{x}}\dot{\boldsymbol{\theta}}. \tag{10}$$

3.2 Second-Order Nullspace Inverse Kinematics Controller Scheme

In the following, all relations hold for serial and parallel robots simultaneously. Since all joints of serial robots are assumed as active, $\boldsymbol{\theta} := \boldsymbol{q}$ holds in that case. The inversion of the kinematics equations provides a nullspace solution if a task redundancy exists. The transfer from operational space $\boldsymbol{x}$ to task space $\boldsymbol{y}$ is simplified by the chosen X-Y'-Z''-angle representation such that the last row of (4) and (9) can be removed with $\boldsymbol{y} = \boldsymbol{P}_{\boldsymbol{y}}\boldsymbol{x}$ and $\boldsymbol{J}_{\boldsymbol{y}} = \boldsymbol{P}_{\boldsymbol{y}}\boldsymbol{J}_{\boldsymbol{x}}$ (for serial and for parallel robots). If no specific value for the redundant coordinate is given, the minimum-norm solution for the task motion can be obtained using the pseudo inverse † as

$$\ddot{\boldsymbol{\theta}}_{\mathrm{T,minnorm}} = \boldsymbol{J}_{\boldsymbol{y}}^{\dagger}(\ddot{\boldsymbol{y}} - \dot{\boldsymbol{J}}_{\boldsymbol{y}}\dot{\boldsymbol{\theta}}). \tag{11}$$

The nullspace solution of the inverse kinematics on acceleration level is obtained as the homogenous solution of (11), neglecting the term $\dot{\boldsymbol{N}}_{\boldsymbol{\theta}}$, as

$$\ddot{\boldsymbol{\theta}}_{\mathrm{N}} = (\boldsymbol{I} - \boldsymbol{J}_{\boldsymbol{y}}^{\dagger}\boldsymbol{J}_{\boldsymbol{y}})\boldsymbol{v}_{\boldsymbol{\theta}} = \boldsymbol{N}_{\boldsymbol{\theta}}\boldsymbol{v}_{\boldsymbol{\theta}}, \tag{12}$$

which projects arbitrary vectors $\boldsymbol{v}_{\boldsymbol{\theta}}$ into the nullspace. The full expression for the actuator accelerations can thus be combined from $\ddot{\boldsymbol{\theta}}_{\mathrm{T}}$ from (5)/(7) (serial/parallel) or (11) in combination with $\ddot{\boldsymbol{\theta}}_{\mathrm{N}}$ from (12) as

$$\ddot{\boldsymbol{\theta}} = \begin{cases} \ddot{\boldsymbol{\theta}}_{\mathrm{T}} + \ddot{\boldsymbol{\theta}}_{\mathrm{N}} & \text{for a given } \varphi_z = \varphi_{z,\mathrm{ff}} \\ \ddot{\boldsymbol{\theta}}_{\mathrm{T,minnorm}} + \ddot{\boldsymbol{\theta}}_{\mathrm{N}} & \text{otherwise.} \end{cases} \tag{13}$$

In the following, the vector $\boldsymbol{v}_{\boldsymbol{\theta}}$ for the second-order nullspace projection (12) is created by a PD controller scheme [3, 28] such that a stable nullspace optimization is performed. This substitutes the analytic scheme of [24] numerically. By using $h_{\partial\boldsymbol{\theta}}$ as a feedback for $\boldsymbol{v}_{\boldsymbol{\theta}}$ the performance criterion h is optimized, as discussed in the next subsection. The overall nullspace controller is shown in the right part of Fig. 2 and contains the controller with PD gains K_{P}, K_{D} and additional damping K_{v}. The trajectory input from (13) is placed on the left part of the figure and allows switching between the cases whether a feedforward

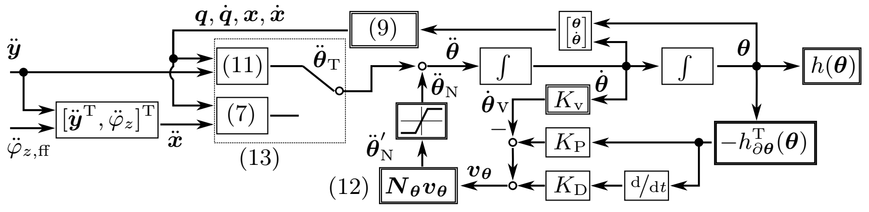
$\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{x}, \dot{\boldsymbol{x}}$
(9)
$\begin{bmatrix}\boldsymbol{\theta}\\ \dot{\boldsymbol{\theta}}\end{bmatrix}$
$\ddot{\boldsymbol{y}}$
(11)
$\ddot{\boldsymbol{\theta}}_\mathrm{T}$
$\ddot{\boldsymbol{\theta}}$
$\int$
$\int$
$\boldsymbol{\theta}$
$h(\boldsymbol{\theta})$
$\ddot{\boldsymbol{\theta}}_\mathrm{N}$
$\dot{\boldsymbol{\theta}}$
K_v
$\dot{\boldsymbol{\theta}}_\mathrm{V}$
$[\ddot{\boldsymbol{y}}^\mathrm{T}, \ddot{\varphi}_z]^\mathrm{T}$
(7)
$-$
K_P
$-h_{\partial\boldsymbol{\theta}}^\mathrm{T}(\boldsymbol{\theta})$
$\ddot{\varphi}_{z,\mathrm{ff}}$
$\ddot{\boldsymbol{x}}$
(13)
$\ddot{\boldsymbol{\theta}}_\mathrm{N}'$
(12)
$\boldsymbol{N}_{\boldsymbol{\theta}}\boldsymbol{v}_{\boldsymbol{\theta}}$
$\boldsymbol{v}_{\boldsymbol{\theta}}$
K_D
$\mathrm{d}/\mathrm{d}t$

Fig. 2. Nullspace motion controller scheme with nullspace trajectory feedforward

term $\ddot{\varphi}_{z,\mathrm{ff}}$ for the redundant coordinate is given or not. In the first case, the nullspace motion develops around the feedforward trajectory $\varphi_{z,\mathrm{ff}}(t)$ and in case of $h = \mathrm{const}$ the redundant coordinate will move according to it.

The model requires computing matrices $\boldsymbol{J}_{\boldsymbol{q},\boldsymbol{x}}^{-1}$ and $\dot{\boldsymbol{J}}_{\boldsymbol{q},\boldsymbol{x}}$ which depend on the full robot state $\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{x}, \dot{\boldsymbol{x}}$, shown by the upper feedback branch. The actual value φ_z of the redundant coordinate has to be computed via the position-level inverse kinematics and the velocity $\dot{\varphi}_z$ using the manipulator Jacobian with (4) or (9).

3.3 Performance Criteria for Gradient Projection

For a feasible motion strict constraints regarding self collision, joint limits and singularities have to be met, which are implemented as objective within the nullspace optimization. Optimizing multiple criteria h_i with only one degree of freedom does not allow using task priority schemes like [21, 14]. The drawback of a weighted sum $h{=}\sum w_i h_i$ as nullspace objective is that prioritization is performed by the weights w_i, which lack of a physical meaning and often have to be tuned manually. To encounter the weights-tuning problem, optimization criteria to achieve feasibility constraints were chosen that lead to infinite penalty, following e.g. [35]. The multi-objective problem is encountered by choosing objective functions with an activation threshold to avoid permanent activity of multiple criteria. This can be seen as a special case of set-based task-priority frameworks like [14].

Joint Limits are regarded with the hyperbolic joint limit criterion

$$h_\mathrm{lim}(\boldsymbol{q}) = \frac{1}{\dim(\boldsymbol{q})} \sum_{i=1}^{\dim(\boldsymbol{q})} h_\mathrm{lim}(q_i) \quad \text{with} \quad h_\mathrm{lim}(q_i) = h_\mathrm{lim,hyp}(q_i) \quad \text{and} \tag{14}$$

$$h_\mathrm{lim,hyp}(q_i) = \frac{(q_{i,\max}{-}q_{i,\min})^2}{8} \left(\frac{1}{(q_i{-}q_{i,\min})^2} + \frac{1}{(q_i{-}q_{i,\max})^2} \right) \geq 1 \tag{15}$$

from [35] with the modification of an activation threshold. The criterion is only active if the limit is approached passing a lower or upper threshold $q_{i,\mathrm{thr,min}}$ or $q_{i,\mathrm{thr,max}}$. The continuous differentiability is achieved via cubic spline transitions and switching points $q_{i,\mathrm{sw,min}}$ and $q_{i,\mathrm{sw,max}}$, resulting in the criterion

$$h_{\mathrm{lim}}(q_i) = \begin{cases} \infty & \text{for} \quad q_i < q_{i,\mathrm{min}} \quad \text{(lower limit)} \\ h_{\mathrm{lim,hyp}}(q_i) & \text{for} \quad q_{i,\mathrm{min}} \leq q_i < q_{i,\mathrm{sw,min}} \\ h_{\mathrm{spline,ll}}(q_i) & \text{for} \quad q_{i,\mathrm{sw,min}} \leq q_i < q_{i,\mathrm{thr,min}} \\ 0 & \text{for} \quad q_{i,\mathrm{thr,min}} < q_i < q_{i,\mathrm{thr,max}} \quad \text{(inactive)} \\ h_{\mathrm{spline,ul}}(q_i) & \text{for} \quad q_{i,\mathrm{thr,max}} \leq q_i < q_{i,\mathrm{sw,max}} \\ h_{\mathrm{lim,hyp}}(q_i) & \text{for} \quad q_{i,\mathrm{sw,max}} \leq q_i \leq q_{i,\mathrm{max}} \\ \infty & \text{for} \quad q_{i,\mathrm{max}} < q_i \quad \text{(upper limit).} \end{cases} \tag{16}$$

The gradient $h_{\partial \boldsymbol{q}} = \partial h / \partial \boldsymbol{q}$ can be obtained analytically. In the paper's examples, $q_{i,\mathrm{thr}}$ is set to be 90% of the limit range and $q_{i,\mathrm{sw,max}} = (q_{i,\mathrm{thr,max}} + q_{i,\mathrm{max}})/2$.

Platform coordinate limits have to be restricted for the global optimization algorithm presented in Sect. 4. The redundant coordinate's range is $\varphi_{z,\mathrm{min}} \leq \varphi_z \leq \varphi_{z,\mathrm{max}}$, also ensured by nullspace optimization and a criterion $h_{\varphi_z,\mathrm{lim}}$ analogue to (16) with φ_z instead of q_i. The limit $\varphi_{z,\mathrm{min}}$ can be adjusted over time around a given feedforward reference $\varphi_{z,\mathrm{ff}}(t)$ to enlarge or diminish the allowed range of the redundant coordinate, as shown in Sect. 4 by spline interpolation.

Singularities are avoided by means of nullspace optimization, where the condition number $h_{\mathrm{cond,II}} = \mathrm{cond}(\boldsymbol{J_x})$ of the manipulator Jacobian is used directly as objective. Inconsistent units of the condition number and questionable physical meaning [17, 12] are ignored since an infinite numerical value represents a singularity in any case. To allow optimization of other criteria as well, an activation threshold $h_{\mathrm{cond,act}}$ for the singularity criterion is defined and a cubic spline transition h_{spline} leads to direct use of the condition number after the threshold $h_{\mathrm{cond,thr}}$, resulting in

$$h_{\mathrm{sing}} = \begin{cases} 0 & \text{for} \quad h_{\mathrm{cond}} < h_{\mathrm{cond,act}} \quad \text{(inactive)} \\ h_{\mathrm{spline}}(h_{\mathrm{cond}}) & \text{for} \quad h_{\mathrm{cond,act}} \leq h_{\mathrm{cond}} < h_{\mathrm{cond,thr}} \\ h_{\mathrm{cond}} & \text{otherwise} \quad \text{(reaches } \infty \text{ in singularity).} \end{cases} \tag{17}$$

For obtaining the gradient $h_{\partial \boldsymbol{q}}{=}\partial h / \partial \boldsymbol{q}$, a numeric implementation via difference quotients is used, as detailed in [28]. In the further examples of this paper, $h_{\mathrm{cond,II,act}}{=}h_{\mathrm{cond,II,thr}}{=}1$ leads to a permanent optimization of the condition number, which has a weak (but questionable) correlation with other performance criteria [17].

Additionally, singularities of the inverse kinematics of $\boldsymbol{\Phi}_{\partial \boldsymbol{q}}$ for parallel robots and $\boldsymbol{J_y}$ for serial robots are handled in the same manner. This allows to avoid serial (type I, $\boldsymbol{\Phi}_{\partial \boldsymbol{q}}$) and parallel (type II, $\boldsymbol{J_x}$) singularities for parallel robots. The respective criteria are termed $h_{\mathrm{sing,I}}$ and $h_{\mathrm{sing,II}}$.

Self-Collisions of the robot structure are considered by a simplified convex geometric model of the robot. The six links are represented by capsules (cylinders ending in half-spheres) to allow a fast geometric check for body intersection. The platform is modeled as a ring of six capsules, resulting in 39 elementary collision checks with a minimal distance $d_{\mathrm{coll,min}} = \min d_i$ of any of the objects. An intersection leads to a negative value. The collision criterion

$$h_{\mathrm{coll}}(d_{\mathrm{coll,min}}) = \begin{cases} 0 & \text{for} \quad d_{\mathrm{coll,min}} \geq d_{\mathrm{coll,thr}} \quad \text{(safe distance)} \\ h_{\mathrm{spline}}(d_{\mathrm{coll,min}}) & \text{for} \quad d_{\mathrm{thr,hyp}} \leq d_{\mathrm{coll,min}} < d_{\mathrm{coll,thr}} \\ h_{\mathrm{coll,hyp}}(d_{\mathrm{coll,min}}) & \text{for} \quad 0 < d_{\mathrm{coll,min}} < d_{\mathrm{thr,hyp}} \\ \infty & \text{for} \quad d_{\mathrm{coll,min}} \leq 0 \quad \text{(collision)} \end{cases} \tag{18}$$

is activated if a safety threshold $d_{\mathrm{coll,thr}}$ is exceeded. A cubic spline sets the transition from zero to one branch of a hyperbola similar to (15), beginning at distance $d_{\mathrm{thr,hyp}}$, which reaches ∞ in case of collision. The gradient is computed numerically via difference quotients.

The Combined Performance Criterion is obtained by a weighted sum

$$h = w_{\mathrm{sing,I}} h_{\mathrm{sing,I}} + w_{\mathrm{sing,II}} h_{\mathrm{sing,II}} + w_{\mathrm{lim}} h_{\mathrm{lim}} + w_{\mathrm{coll}} h_{\mathrm{coll}} + w_{\varphi_z,\mathrm{lim}} h_{\varphi_z,\mathrm{lim}}, \tag{19}$$

where $w_{\mathrm{sing,I/II}}$=1, w_{lim}=1, w_{coll}=1 and $w_{\varphi_z,\mathrm{lim}}$=100 are chosen by manual tuning to obtain a sufficiently strong repulsion from the coordinate limits for the example in Sect. 4. The gains K_{P}=1, K_{D}=0.7 and K_{v}=0.8 were chosen to reduce oscillations of the second-order system resulting from the PD feedback of a double integrator plant. For visualization of h within a performance map, high values above a threshold of e.g. $h_{\mathrm{thr}} = 1000$ are saturated to increase the number of heatmap colors for distinction of good values. The component of h leading to infinity can then be highlighted by a separate marker.

3.4 Extensions for Rest-to-Rest Motion

To achieve a rest-to-rest motion of the robot trajectory $\boldsymbol{x}(t)$, an additional braking has to be implemented within the nullspace motion $\varphi_z(t)$. Otherwise, a nullspace velocity $\dot{\varphi}_z$ is likely to still persist at the task's resting points. As elaborated before in Sect. 2 and (1), rest-to-rest motion for the task trajectory $\boldsymbol{y}(t)$ is assumed as given. A time-varying technically feasible limit $\dot{\varphi}_{z,\mathrm{min}}(t)$ and $\dot{\varphi}_{z,\mathrm{max}}(t)$ for the redundant coordinate's velocity is set. When the resting time t_{R} is approached with $t > t_{\mathrm{R}} - T_{\mathrm{dec}}$, the limit is linearly decreasing to zero in the deceleration time T_{dec}, which is obtained from acceleration and velocity limits. The nullspace braking is implemented within the saturation block in Fig. 2. A predicted velocity from the task and nullspace acceleration for the next discrete time step is computed by $\dot{\boldsymbol{\theta}}_{\mathrm{pre}}(k+1) = \dot{\boldsymbol{\theta}}(k) + (\ddot{\boldsymbol{\theta}}_{\mathrm{T}} + \ddot{\boldsymbol{\theta}}'_{\mathrm{N}})\Delta t$. The redundant coordinate's velocity is obtained by the Jacobian relation $\dot{\varphi}_{z,\mathrm{pre}} = \boldsymbol{P}_{\varphi_z} \boldsymbol{J}_{\boldsymbol{x}} \dot{\boldsymbol{\theta}}_{\mathrm{pre}}(k+1)$, where $\boldsymbol{P}_{\varphi_z} = [\boldsymbol{0}^{\mathrm{T}}, 1]$ selects the last row. If $\dot{\varphi}_{z,\mathrm{pre}}$ exceeds the limit $\dot{\varphi}_{z,\mathrm{max}}$, an additional nullspace acceleration $\ddot{\varphi}_{z,\mathrm{add}} = (\dot{\varphi}_{z,\mathrm{max}} - \dot{\varphi}_{z,\mathrm{pre}})/\Delta t$ is created to truncate the velocity. This acceleration is then added to the robot's joint space by $\ddot{\boldsymbol{\theta}}_\mathrm{N} = \ddot{\boldsymbol{\theta}}'_\mathrm{N} + \boldsymbol{J}_{\boldsymbol{x}}^{-1}[\boldsymbol{0}^\mathrm{T}, \ddot{\varphi}_{z,\mathrm{add}}]^\mathrm{T}$.

Additionally, a damping is added in the nullspace, which promotes following the redundant coordinate's feedforward $\dot{\varphi}_{z,\mathrm{ff}}$. The block K_v in Fig. 2 is implemented by a feedback law $\dot{\boldsymbol{\theta}}_\mathrm{v} = K_\mathrm{v}\boldsymbol{P}_{\varphi_z}\boldsymbol{J}_{\boldsymbol{x}}(\dot{\varphi}_z - \dot{\varphi}_{z,\mathrm{ff}})$ of the redundant coordinate's velocity.

3.5 Trajectory Performance Criterion

To obtain a criterion for evaluation of the complete rest-to-rest trajectory k, an RMS criterion w.r.t. the normalized path coordinate s is defined as

$$h_\mathrm{int} = \frac{1}{s_{\mathrm{R}_k} - s_{\mathrm{R}_{k-1}}} \int_{s_{\mathrm{R}_{k-1}}}^{s_{\mathrm{R}_k}} h(\boldsymbol{q}(s), \boldsymbol{x}_E(s))\mathrm{d}s. \tag{20}$$

The path coordinate s is used instead of time t to reduce the weight of sections with low velocity. It has integer values at the rest times t_{R_k} and interpolates linearly according to the distance of the task coordinate $\boldsymbol{y}$. The use of the root mean square (RMS) in (20) and the definition of h in (19) make it susceptible to the nearly infinite penalty values when a singularity or a limit is approached. This desirable behavior allows the simultaneous incorporation of the constraints and the objective in the optimization of rest poses via dynamic programming, which is the focus of the next section.

4 Global Optimization: Dynamic Programming

The local optimization using nullspace projection from Sect. 3 is now used for the global trajectory optimization problem introduced in Sect. 2. First, the problem is solved with the well-known (discrete) dynamic programming method in Sect. 4.1 which is then extended to a novel state-interval-based dynamic programming method in Sect. 4.2.

4.1 Discrete Dynamic Programming

The stage-wise definition of the rest-to-rest trajectory optimization problem is

$$J^* = \min_{\boldsymbol{U}} J(u_1, ..., u_n) = \sum_{k=1}^{n} l_k(x_{k-1}, u_k) \tag{21}$$

$$\text{s.t.} \quad x_k = f_k(x_{k-1}, u_k) \tag{22}$$

$$x_k \in X_k \tag{23}$$

$$u_k \in U_k(x_{k-1}). \tag{24}$$

The total cost J is a sum of costs l_k on each of the n stages, which are only influenced by decisions u_k on that stage and the stage's initial state x_{k-1}. The initial value x_0 is given. The goal is to find a trajectory of decisions $\boldsymbol{U} = \{u_0, u_1, u_{n-1}\}$ as argmin in (21). Optimal values are marked with an asterisk, like J^*. Only a discrete set of states X_k is possible (23). The transfer between states is denoted as function f_k in (22) which produces the transfer costs l_k. Decisions are restricted to a set U_k leading to these exact states (24). The set of states is predetermined as a discretization of the continuous decision variable x for n_ref reference values as

$$X_\mathrm{ref} = \{x_{\mathrm{ref},1}, x_{\mathrm{ref},2}, ..., x_{\mathrm{ref},n_\mathrm{ref}}\}. \tag{25}$$

In a forward iteration, the running costs are obtained for in total $|X_{k-1}||U_k|$ different transfers from stage $k-1$ to stage k, termed by

$$l_k(x_{k-1}, u_k) \quad \forall \quad x_{k-1} \in X_{k-1} \wedge u_k \in U_k(x_{k-1}). \tag{26}$$

The cumulated cost J_k for each of the states x_k of stage k is then obtained via

$$J_k(x_{k-1}, u_k) = J^*_{k-1}(x_{k-1}) + l_k(x_{k-1}, u_k), \tag{27}$$

where $J^*_{k-1}(x_{k-1})$ is the optimal total cost from x_0 to x_{k-1}, initialized with $J^*_0(x_0)$=0. The optimal series of decisions for each state x_k on stage k is obtained via considerations based on Bellman's principle of optimality as

$$J^*_k(x_k) = \min\{J_k(x_{k-1}, u_k) \mid x_k{=}f(x_{k-1}, u_k) \wedge u_k \in U_k(x_{k-1})\}. \tag{28}$$

All predecessor states x_{k-1} are considered. The set $U_k(x_{k-1})$ of decision variables is selected such that each of the reference states $X_k := X_\mathrm{ref}$ is reached once. If the state x_{k-1} can only be reached by violation of a constraint, this is marked by the previous iteration with infinite cost $J^*_k(x_{k-1})$ for that state, resulting to

$$U_k(x_{k-1}) = \begin{cases} \emptyset & \text{for } J^*_k(x_{k-1}) = \infty \\ \{u_k \mid x_k{=}f(x_{k-1}, u_k) \ \forall \ x_k{\in}X_k\} & \text{otherwise.} \end{cases} \tag{29}$$

The optimal series of decisions and states leading to $J^*_k(x_k)$ is written as

$$\begin{aligned} U^*(x_k) &= \{u^*_1, ..., u^*_k\} \quad \text{and} \\ X^*(x_k) &= \{x^*_0, x^*_1, ..., x^*_{k-1}, x_k\}. \end{aligned} \tag{30}$$

The dynamic programming algorithm now consists of alternating forward passes (26) and backward passes (28) to select optimal stage decisions. The algorithm is performed for all stages from $k = 1, ..., n$. The best final state then contains the optimal solution as

$$J^* = J^*_k(x^*_n) = \min J^*_k(x_n) \tag{31}$$
$$U^* = U^*(x^*_n) = \{u^*_1, ..., u^*_n\} \tag{32}$$
$$X^* = X^*(x^*_n) = \{x^*_0, x^*_1, ..., x^*_n\} \tag{33}$$

Robot Trajectory Example: The general form of the algorithm is transferred to the robot example in the following by assigning the DP variables from above with the physical variables in Sect. 2 and 3. The stages correspond to the rest positions of the trajectory. In the example of [28] depicted in Fig. 1 this means $n = N_\mathrm{R} = 7$. Each transition $x_k{=}f(x_{k-1}, u_k)$ is a rest-to-rest trajectory $\varphi_z(t)$ for the redundant coordinate computed by a trapezoidal velocity profile for the given trajectory time base $t_{\mathrm{R}_{k-1}} \leq t \leq t_{\mathrm{R}_k}$ between given states in $x_{k-1} = \varphi_z(t_{\mathrm{R}_{k-1}})$ and $x_k = \varphi_z(t_{\mathrm{R}_k})$. The decision u_k is therefore just the selection of a target position x_k. The discretization is chosen as $x_\mathrm{min} = \varphi_{z,\mathrm{min}} = -180°$, $x_\mathrm{max} = \varphi_{z,\mathrm{max}} = 180°$ and $n_\mathrm{ref} = 9$, leading to $\mathit{\Delta} x = \mathit{\Delta}\varphi_z = 45°$. The cost is determined by the RMS of the performance criterion (20), according to Sect. 3.3, i.e.

$$l_k(x_{k-1}, u_k) = \frac{1}{s_{\mathrm{R}_k} - s_{\mathrm{R}_{k-1}}} \int_{s_{\mathrm{R}_{k-1}}}^{s_{\mathrm{R}_k}} h(\boldsymbol{q}(s), \boldsymbol{x}_E(s))\mathrm{d}t. \tag{34}$$

The initial value $x_0 = \varphi_{z,0} \approx -35°$ and $\boldsymbol{q}(t{=}0)$ is selected by a gradient-descent approach for the optimal joint limit criterion. The joint configuration $\boldsymbol{q}(t)$ is obtained by the inverse kinematic scheme from Sect. 3.

A step-by-step solution with the dynamic programming approach above is shown in Fig. 3,a for the first stage $k = 1$. As only one previous stage $X_0 = \{x_0\}$ exists, the only one transfer to each x_k from (30) is the optimal transfer. The trajectory evaluation is immediately aborted if a robot constraint is violated. This speeds up the algorithm and is marked by black lines in Fig. 3. The corresponding cost l_k is set to a high penalty (in this case ∞) to discard this state automatically in the backward recursion (28). If all decisions towards a state x_{k-1} are invalid, the first condition in (29) becomes active and from this state no actions will be taken on further. The forward iteration for the next stage $k = 2$ therefore only continues for valid states. Only one trajectory for the first state and the last state give feasible transfers, as visible in Fig. 3,b. The recursion therefore remains trivial. In iteration $k = 3$ both remaining states from X_2 lead to feasible transfers to the three states in X_3, as can be seen in Fig. 3,c. Further, each state in X_3 can be reached by both states of X_2, from which the upper one has lower cost and is therefore marked as optimal (magenta). Lines originating from the lower state are marked as valid (cyan), but not optimal.

The final result of the algorithm after performing the remaining stages $k = 4...7$ is shown in Fig. 4 with the highlighted optimal trajectory that has been found. The continuous optimization variable can only very roughly be approximated by the chosen discretization of $\varphi_z = 45°$. Choosing a finer discretization is able to improve the result in this case, but using the discrete DP is not applicable to the general case, if a feasible solution can not be found via straight lines in the performance map. Further, the DP algorithm has complexity $\mathcal{O}(n n_\mathrm{ref}^2)$, making the presented implementation inefficient for such continuous problems.

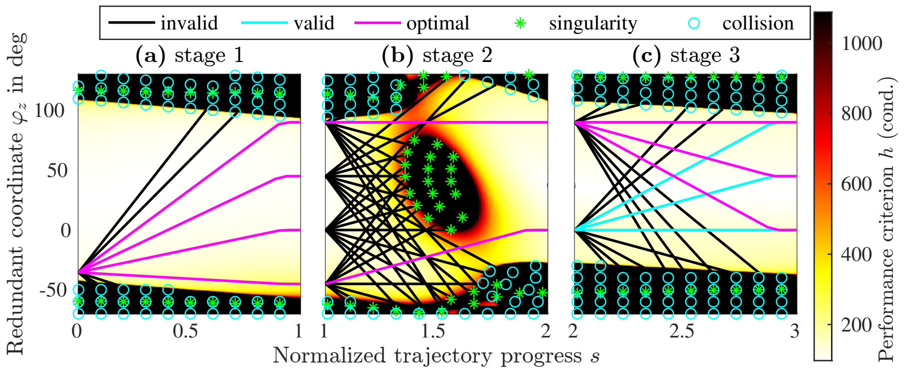

Fig. 3. Decisions on the first three stages using discrete DP for the reference problem

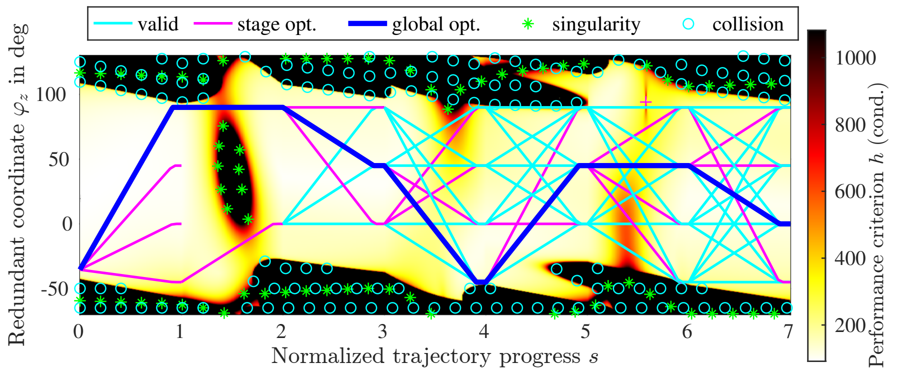

Fig. 4. Result of discrete dynamic programming for the reference problem

4.2 State-Interval Dynamic Programming

The drawbacks of the classical discrete dynamic programming can be solved by defining a target interval for the states rather than a fixed value. A state interval

$$[x] = [\bar{x} - {}^{\Delta x}/_{2}, \bar{x} + {}^{\Delta x}/_{2}] \tag{35}$$

is defined by a center $\bar{x}$ and the interval width Δx. Instead of a fixed set of discretized reference states X_{ref} in (25), the states are now[3]

$$X_{\mathrm{ref}} = \{[x_{\mathrm{ref},1}], [x_{\mathrm{ref},2}], ..., [x_{\mathrm{ref},n_{\mathrm{ref}}}]\}. \tag{36}$$

The DP formulation from the previous section is adapted in such a way that previous states remain specific values x_{k-1}. Future states are considered as an

[3] Intervals are assumed non-overlapping (i.e. half-open) for the sake of mathematical proof, however the symbol $[\cdot]$ for closed intervals is used to enhance readability.

interval, allowing a set of solutions $x_k \in [x_k]$ or $\bar{x}_k - {}^{\Delta x}/_2 \leq x_k < \bar{x}_k + {}^{\Delta x}/_2$. The set of decision variables from (29) now becomes

$$U_k(x_{k-1}) = \begin{cases} \emptyset & \text{for } J_k^*(x_{k-1}) = \infty \\ \{u_k \mid f(x_{k-1}, u_k) \in [x_k] \;\forall\; [x_k] \in X_k\} & \text{otherwise.} \end{cases} \tag{37}$$

Since the intervals do not overlap, Bellman's principle of optimality still holds under the assumption of an optimal state transfer regarding the subproblem in $f(x_{k-1}, u_k)$. This means that the transfer to the actual reached state x_k within the interval $[x_k]$ is assumed to be optimal among all possible local transfer strategies $\tilde{u}_k$, i.e.

$$l_k(x_{k-1}, u_k) = \min_{\tilde{u}_k} \{l_k(x_{k-1}, \tilde{u}_k) \;\forall\; \tilde{u}_k \mid f(x_{k-1}, \tilde{u}_k) \in [x_k]\}. \tag{38}$$

The transition strategy u_k in this case is not only correspondence to a target state x_k, but also a specific control strategy for the subproblem $f(x_{k-1}, u_k)$. In the trajectory optimization example this includes the optimal decision for each of the dense trajectory samples, as mentioned in Sect. 2.2. The assumption (38) is unlikely to strictly hold in the presented highly nonlinear robot application. However, approximating this by a near-optimal or at least feasible u_k may already be enough to obtain acceptable results regarding difficulties in finding valid solutions at all, facing the constraints.

The decision u_k now only has to assure that the next state lies in it's allowed interval. This improves the possibility to perform a local optimization in the state transfer function, which corresponds to the trajectory optimization via the nullspace projection. Reaching the interval has to be implemented with extensions to the local optimization or with constraints, as discussed in Sect. 3.3.

The backward recursion from (28) using the interval approach now is

$$J_k^*([x_k]) = \min \{J_k(x_{k-1}, u_k) \mid f(x_{k-1}, u_k) \in [x_k] \wedge u_k \in U_k(x_{k-1})\}. \tag{39}$$

The cumulated cost $J_k(x_{k-1}, u_k)$ does not change compared to (27), since looking retrospectively, the previous state x_{k-1} is the actual obtained value, no interval.

The proposed approach is illustrated at the previous robot trajectory example. Interval limits are set as before as $x_{\mathrm{ref},1} = \varphi_{z,\min} = -180°$, $x_{\mathrm{ref},n_{\mathrm{ref}}} = \varphi_{z,\max} = 180°$, $n_{\mathrm{ref}} = 9$, $\Delta x = \Delta\varphi_z = 45°$ and $x_0 = \varphi_{z,0} \approx -35°$. The term $x_{\min}$ and $x_{\max}$ is not used since the limits for the optimization variable are extended by the interval half-span $\Delta x/2$. A decision u_k consists of a reference trajectory as in the previous discrete DP example. This trajectory is only tracked via a feed-forward gain and a nullspace controller, as explained in Sect. 3. A tolerance band around the trajectory ensures reaching the respective target interval $[x_k]$, as shown in Fig. 5 for three exemplary decisions on the first stage.

As in the previous example, not all target states lead to valid results, especially since the global distribution of the performance map is not known in advance and therefore a wide span is chosen. All transitions of the first stage

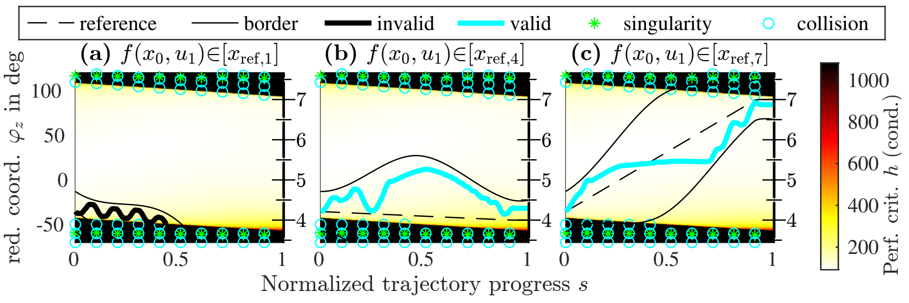

Fig. 5. Decisions u_1 on the first stage using state-interval dynamic programming to exemplary state intervals 1 **(a)**, 4 **(b)** and 7 **(c)** on stage 2. The numbers i on the axis to the right of each performance map correspond to the state intervals $[x_{\mathrm{ref},i}]$

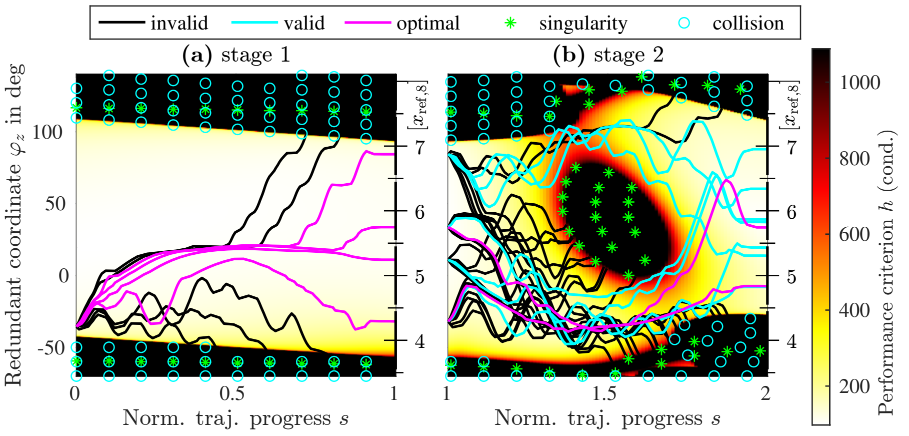

Fig. 6. All decisions on the first two stages using state-interval DP

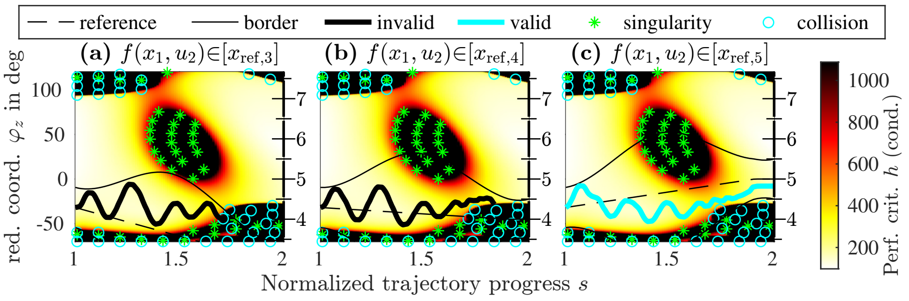

Fig. 7. Decisions u_2 on the second stage from $x_1 \in [x_{\mathrm{ref},4}]$ using state-interval DP

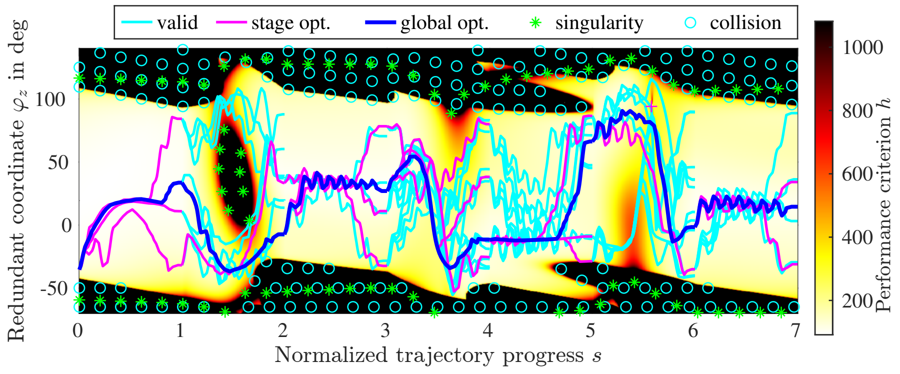

Fig. 8. Result of state-interval dynamic programming for the reference problem

are shown in Fig. 6,a. The valid lines are continued in the next stage. Since only one starting state exists, all valid lines are also optimal, which is highlighted by the color magenta. The continuation for stage 2 is shown first exemplarily together with the reference trajectory and spline-based tolerance intervals in Fig. 7. Figure. 6,b contains all transfers. Multiple decisions u_1 lead to the same state intervals of $[x_2]$, as visible for the cyan lines ending in $[x_{\mathrm{ref},5}]$ and $[x_{\mathrm{ref},6}]$ (noted by the numbers at the right side of Fig. 6,b). Therefore the best decisions according to (39) are selected for continuation, marked magenta. The result of the algorithm is depicted in Fig. 8.

4.3 Overlapping Intervals

The approach presented above enforces the redundant coordinate in a tolerance band by using a repulsing potential $h_{\varphi_z,\mathrm{lim}}$. This has the effect that a local optimum on this border between intervals can not be reached. A solution to achieve this is the use of overlapping intervals. It still has to be assured, that at each stage only the prescribed number of state intervals are continued. Otherwise the number of states will grow from state to state and the underlying optimality principle of dynamic programming does not hold any more. The extension leads to additional possible decisions on each set, extending the set X_k in (37) to

$$X'_{\mathrm{k}} = \{[x_{\mathrm{ref},1}], [x_{\mathrm{ref},2}], ..., [x_{\mathrm{ref},n_{\mathrm{ref}}}], [x_{\mathrm{add},1}], ..., [x_{\mathrm{add},n_{\mathrm{add}}}]\}. \tag{40}$$

The additional intervals are set to be overlapping with the existing ones, i.e.

$$[x_{\mathrm{add},k}] = [x_{\mathrm{ref},k}] + {}^{\Delta x}\!/_{2} = x_{\mathrm{ref},k} \leq x < x_{\mathrm{ref},k+1} \tag{41}$$

and $n_{\mathrm{add}} = n_{\mathrm{ref}} - 1$. To prevent the number of states from increasing, the cumulated cost in the backward recursion (39) is not evaluated for the new, overlapping intervals from (40).

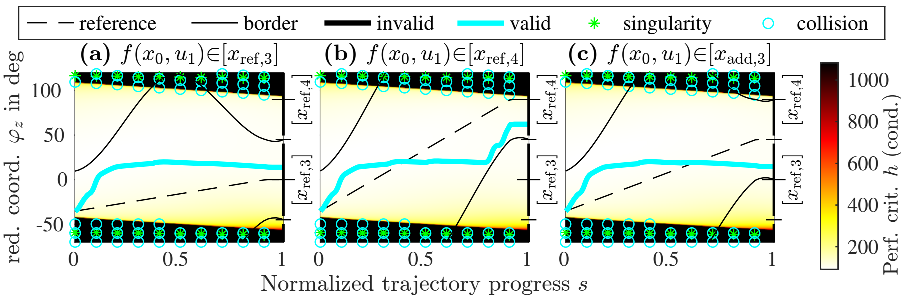

Fig. 9. Decisions u_1 on the first stage using overlapping state-intervals in dynamic programming to exemplary state intervals 3 **(a)**, 4 **(b)** and the overlapping interval in-between **(c)** on stage 2

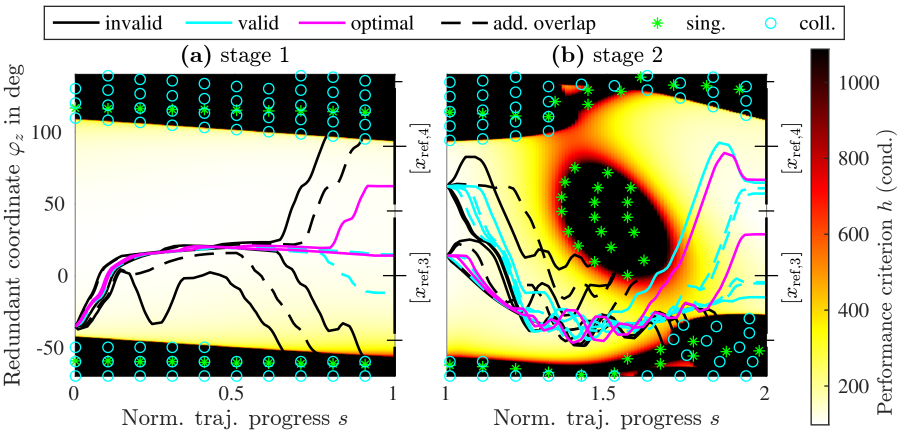

Fig. 10. Decisions on stage 1 and 2 for the reference problem using overlapping intervals

Robot Trajectory Example: The modification of overlapping intervals is again demonstrated at the robot example. The chosen parameters are now $n_{\mathrm{ref}} = 5$ with $\Delta x = \Delta\varphi_z = 90°$. Due to the overlapping intervals, the number of forward iterations stays the same, but fewer states are considered for continuation. Similar to Fig. 5, the first stage transfer is investigated in Fig. 9. The transfer in Fig. 9,c corresponds to the additional interval $[x_{\mathrm{add},3}]$ between $[x_{\mathrm{ref},3}]$ in Fig. 9,a and $[x_{\mathrm{ref},4}]$ in Fig. 9,b.

Similar to Fig. 6 all transfers on the first two stages are shown in Fig. 10. Transfers from additional overlapping intervals are marked with dashed lines. Now already in the first stage in Fig. 10,a multiple actions lead to a transfer to the interval $[x_{\mathrm{ref},3}]$, which seemingly contains the local optimum. The best of these is selected in stage 2 (Fig. 10,b) for continuation.

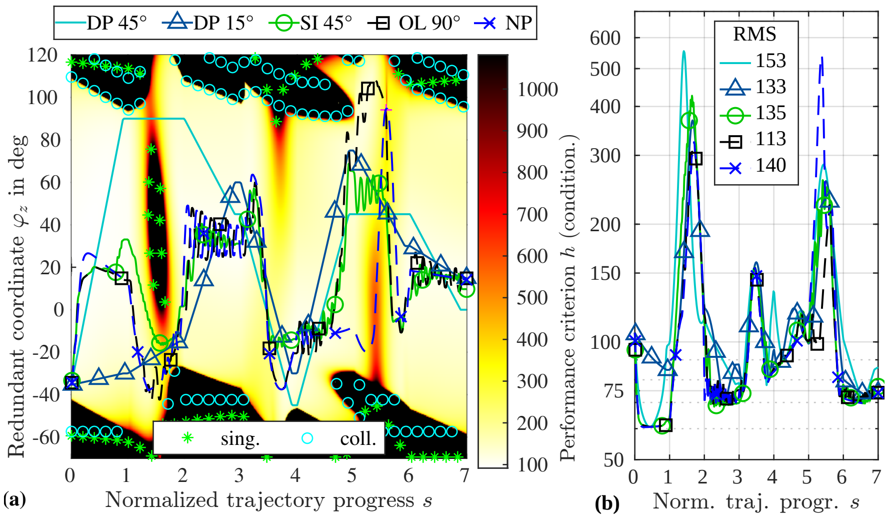

Fig. 11. Nullspace motion in performance map **(a)** and evolution of criterion **(b)**

5 Simulative Validation for a Parallel Robot

In the following a quantitative evaluation of the proposed new method of state-interval dynamic programming (SI-DP) from Sect. 4.2 is performed with a comparison against classical dynamic programming (DP) from Sect. 4.1 and the previous approach of local optimization using nullspace projection (NP) from Sect. 3 (and [28]). An overview of robot and task can be obtained from Fig. 1. The same benchmark task as in the previous section is used. The robot has the following dimensions: base diameter 1200 mm, platform diameter 300 mm and distance of hexagonally aligned platform coupling joint pairs of 100 mm. Collision bodies for the leg chains have a diameter of 40 mm. The task is beginning at $[r_x, r_y, r_z] = [-50, 40, 700]$ mm and $[\varphi_x, \varphi_y] = [45°, 0°]$ in the robot base frame and has a length of 900 mm and duration of 31.6 s with seven rest poses. The trajectory has 31647 samples with a sample time of 1 ms.

The comparison in Fig. 11 shows that the SI-DP with a discretization of 45° outperforms the classical discrete DP with the same discretization in the critical phases of the trajectory. Further, DP presents a different trajectory with φ_z=90° at s=2 due to the narrow passage. The reason can be found in the only very rough discretization, which is beneficial for the SI-DP and reduces computation time. A further improvement is achieved by using the overlapping (OL) intervals from Sect. 4.3 with discretization of 90°. The evaluation of one trajectory sample needs 0.4 ms for the DP and 1.2 ms for the SI-DP and OL-SI-DP, due to more complex nullspace equations. A Linux desktop computer with Intel i5-7500 CPU and MATLAB implementation using MEX-compiled functions was used. The total optimization for DP with 45° took 3.4 min with 535k trajectory samples, corresponding to 16.9 times the full trajectory length. The SI-DP with

45° took 13.2 min for 664k samples, i.e. 21 full trajectory equivalents and the OL-SI-DP took 10 min for 421k samples. The RMS value according to (20) is given in Fig. 11,b and shows the improved performance by the SI-DP by a value of 135 and OL by 113. The DP results can be improved by finer discretization of 15°, which leads to a similar RDP value of 134 like SI-DP, instead of 153, but takes 25.7 min for 4M samples, i.e. 128 full trajectory equivalents.

The NP method performs better in the first two phases since the optimal solution is between intervals of the SI-DP, which is a repulsing potential. Using the OL method solves this problem. The only local optimality of NP becomes visible at s=5 where a local optimum is followed by a region of high cost terms, which has to be traversed. This leads to a moderately worse RMS value of 140, but needs the least computational effort with 39 s for one full trajectory.

The example is modified in the following to pose higher restrictions on the robot. The trajectory length is increased to 1400 mm and the pointing direction now is 45° to the outside, i.e. $[r_x, r_y, r_z] = [-200, 150, 700]$ mm and $[\varphi_x, \varphi_y] = [-45°, 0°]$ for the starting point. The robot is shown in Fig. 12 for three different points of this second trajectory. The cases of singularity and collision are highlighted for illustration. The trajectory optimization is performed with the same methods and settings as before with results presented in Fig. 13. Due to the very narrow passages of possible rotation angle φ_z, the 45°-DP does not find a solution, as well as the local optimization. The fine-discretization 15°-DP and the proposed 45°-SI-DP find a solution, where SI-DP outperforms the DP by means of RMS of the condition number and OL performs best due to less constraints for enforcing the interval.

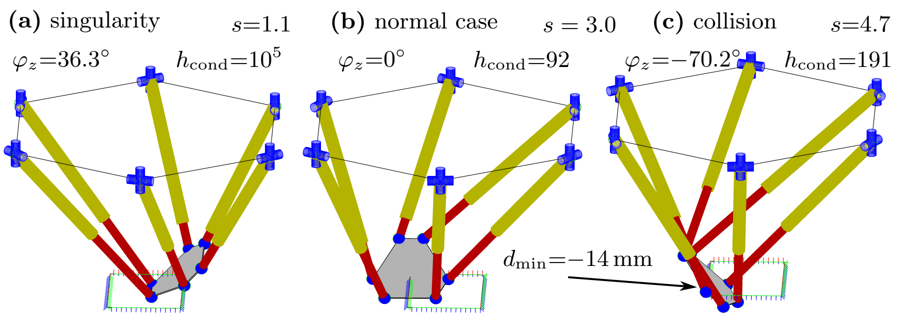

Fig. 12. Robot in three poses of the second performance map example from Fig. 13

For both examples oscillations can be observed in Fig. 11 and Fig. 13 for the nullspace optimization methods SI-DP, OL-SI-DP and NP. This could be omitted by further tuning of PD gains and damping. However at least for SI-DP the oscillation is partially inherent to the method since a repulsing potential from the coordinate limits is used.

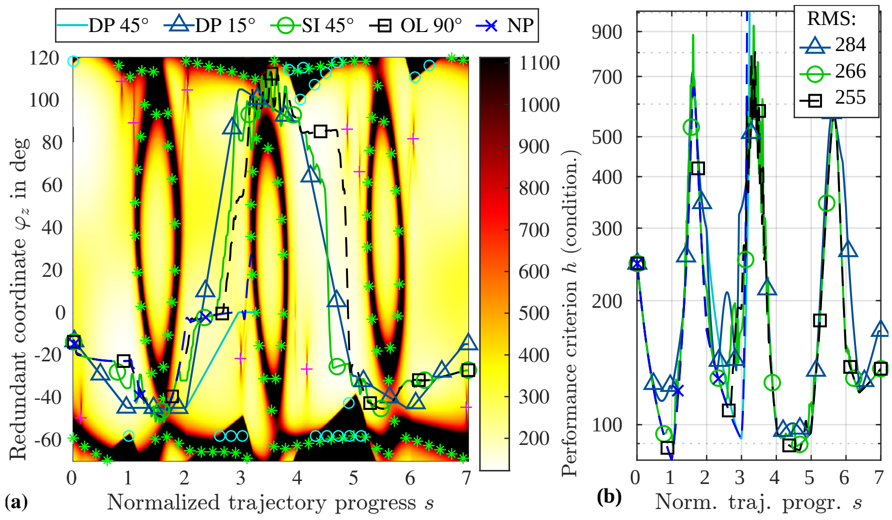

Fig. 13. Results for the second example: Performance map **(a)** and criterion **(b)**

6 Conclusion

The presented algorithm corresponds to a combination of differential and classical dynamic programming and is applicable to trajectory optimization problems of dynamic systems with continuous state and stage variables and rest-to-rest state transitions. The performance measure in intermediate steps should have a correlation with the global optimization objective. This can be the case by globally optimizing the average values of local objectives and in the presence of high penalties for constraints. An example of a robot trajectory optimization is given, however similar problems may arise in other disciplines as well. The results show the improved performance compared to discrete DP or local optimization at acceptable computation times for offline trajectory planning. The approach is not restricted to second-order trajectories or task redundancy within robotics. Optimizing multiple redundant degrees of freedom is also possible, but without the performance map visualization used throughout the paper. Instead of nullspace optimization, also other local optimization techniques could be used, which leads to a still open future investigation on conditions for the analogy of the proposed method to existing ones like DDP.

Acknowledgements The author acknowledges the support by the Deutsche Forschungsgemeinschaft (DFG) under grant number 341489206. MATLAB Code to reproduce the results is available at GitHub under free license at https://github.com/SchapplM/robotics-paper_icinco2021.

References

1. Agarwal, A., Nasa, C., Bandyopadhyay, S.: Dynamic singularity avoidance for parallel manipulators using a task-priority based control scheme. Mechanism and Machine Theory **96**, 107–126 (2016). https://doi.org/10.1016/j.mechmachtheory.2015.07.013
2. Corinaldi, D., Angeles, J., Callegari, M.: Posture optimization of a functionally redundant parallel robot. In: Advances in Robot Kinematics 2016, pp. 101–108. Springer (2016). https://doi.org/10.1007/978-3-319-56802-7_11
3. De Luca, A., Oriolo, G., Siciliano, B.: Robot redundancy resolution at the acceleration level. Laboratory Robotics and Automation **4**, 97–97 (1992)
4. Ferrentino, E., Salvioli, F., Chiacchio, P.: Globally optimal redundancy resolution with dynamic programming for robot planning: a ros implementation. MDPI Robotics **10**(1), 42 (2021). https://doi.org/10.3390/robotics10010042
5. Gao, J., Pashkevich, A., Caro, S.: Optimization of the robot and positioner motion in a redundant fiber placement workcell. Mechanism and Machine Theory **114**, 170–189 (2017). https://doi.org/10.1016/j.mechmachtheory.2017.04.009
6. Gao, Y., Chen, K., Gao, H., Xiao, P., Wang, L.: Small-angle perturbation method for moving platform orientation to avoid singularity of asymmetrical 3-RRR planner (sic) parallel manipulator. Journal of The Brazilian Society of Mechanical Sciences and Engineering **41**, 1–18 (2019). https://doi.org/10.1007/s40430-019-2012-4
7. Gosselin, C., Schreiber, L.T.: Kinematically redundant spatial parallel mechanisms for singularity avoidance and large orientational workspace. IEEE Transactions on Robotics **32**(2), 286–300 (2016). https://doi.org/10.1109/tro.2016.2516025
8. Gosselin, C., Schreiber, L.T.: Redundancy in Parallel Mechanisms: A Review. Applied Mechanics Reviews **70**(1) (01 2018). https://doi.org/10.1115/1.4038931
9. Guigue, A., Ahmadi, M., Hayes, M., Langlois, R., Tang, F.: A dynamic programming approach to redundancy resolution with multiple criteria. In: Proceedings 2007 IEEE International Conference on Robotics and Automation. pp. 1375–1380. IEEE (2007). https://doi.org/10.1109/ROBOT.2007.363176
10. Howell, T.A., Jackson, B.E., Manchester, Z.: Altro: A fast solver for constrained trajectory optimization. In: 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 7674–7679 (2019). https://doi.org/10.1109/iros40897.2019.8967788
11. Huo, L., Baron, L.: The joint-limits and singularity avoidance in robotic welding. Industrial Robot: An International Journal **35**(5), 456–464 (2008). https://doi.org/10.1108/01439910810893626
12. Kotlarski, J., Do Thanh, T., Heimann, B., Ortmaier, T.: Optimization strategies for additional actuators of kinematically redundant parallel kinematic machines. In: Robotics and Automation (ICRA), 2010 IEEE International Conference on. pp. 656–661. IEEE (2010)
13. Lantoine, G., Russell, R.P.: A hybrid differential dynamic programming algorithm for constrained optimal control problems. part 1: Theory. Journal of Optimization Theory and Applications **154**(2), 382–417 (2012). https://doi.org/10.1007/s10957-012-0039-0
14. Lillo, P.D., Chiaverini, S., Antonelli, G.: Handling robot constraints within a set-based multi-task priority inverse kinematics framework. In: 2019 International Conference on Robotics and Automation (ICRA). pp. 7477–7483 (2019). https://doi.org/10.1109/ICRA.2019.8793625

15. Léger, J., Angeles, J.: Off-line programming of six-axis robots for optimum five-dimensional tasks. Mechanism and Machine Theory **100**, 155–169 (2016). https://doi.org/10.1016/j.mechmachtheory.2016.01.015
16. Merlet, J.P., Perng, M.W., Daney, D.: Optimal trajectory planning of a 5-axis machine-tool based on a 6-axis parallel manipulator. In: Advances in Robot Kinematics, pp. 315–322. Springer (2000). https://doi.org/10.1007/978-94-011-4120-8_33
17. Merlet, J.P.: Parallel Robots, Solid Mechanics and Its Applications, vol. 128. Springer Science & Business Media, 2nd edn. (2006). https://doi.org/10.1007/1-4020-4133-0
18. Mousavi, S., Gagnol, V., Bouzgarrou, B.C., Ray, P.: Control of a multi degrees functional redundancies robotic cell for optimization of the machining stability. Procedia CIRP **58**, 269–274 (2017). https://doi.org/10.1016/J.PROCIR.2017.04.004
19. Mousavi, S., Gagnol, V., Bouzgarrou, B.C., Ray, P.: Stability optimization in robotic milling through the control of functional redundancies. Robotics and Computer-Integrated Manufacturing **50**, 181–192 (2018). https://doi.org/10.1016/j.rcim.2017.09.004
20. Nakamura, Y., Hanafusa, H.: Optimal redundancy control of robot manipulators. The International Journal of Robotics Research **6**(1), 32–42 (1987). https://doi.org/10.1177/027836498700600103
21. Nakamura, Y., Hanafusa, H., Yoshikawa, T.: Task-priority based redundancy control of robot manipulators. The International Journal of Robotics Research **6**(2), 3–15 (1987)
22. Oen, K.T., Wang, L.C.T.: Optimal dynamic trajectory planning for linearly actuated platform type parallel manipulators having task space redundant degree of freedom. Mechanism and Machine Theory **42**(6), 727–750 (2007). https://doi.org/10.1016/j.mechmachtheory.2006.05.006
23. Ozgoren, M.K.: Optimal inverse kinematic solutions for redundant manipulators by using analytical methods to minimize position and velocity measures. Journal of Mechanisms and Robotics **5**(3) (06 2013). https://doi.org/10.1115/1.4024294
24. Reiter, A., Müller, A., Gattringer, H.: On higher order inverse kinematics methods in time-optimal trajectory planning for kinematically redundant manipulators. IEEE Transactions on Industrial Informatics **14**(4), 1681–1690 (2018). https://doi.org/10.1109/TII.2018.2792002
25. Reveles R., D., Pamanes G., J.A., Wenger, P.: Trajectory planning of kinematically redundant parallel manipulators by using multiple working modes. Mechanism and Machine Theory **98**, 216–230 (2016). https://doi.org/10.1016/j.mechmachtheory.2015.09.011
26. Santos, J.C., da Silva, M.M.: Redundancy resolution of kinematically redundant parallel manipulators via differential dynamic programing. Journal of Mechanisms and Robotics **9**(4) (2017). https://doi.org/10.1115/1.4036739
27. Schappler, M.: Simulative Optimierung der Bahnplanung mit mehrfacher Redundanz bei der roboterassistierten Laserosteotomie. Bachelor's thesis, Leibniz Universität Hannover, Institut für Mechatronische Systeme (2013). https://doi.org/10.15488/10214
28. Schappler, M., Ortmaier, T.: Singularity avoidance of task-redundant robots in pointing tasks: On nullspace projection and cardan angles as orientation coordinates. In: Proceedings of the 18th International Conference on Informatics in Control, Automation and Robotics (ICINCO 2021) (2021). https://doi.org/10.5220/0010621103380349

29. Sciavicco, L., Siciliano, B.: Modelling and control of robot manipulators. Springer Science & Business Media (2012). https://doi.org/10.1007/978-1-4471-0449-0
30. Shaw, D., Chen, Y.S.: Cutting path generation of the Stewart-platform-based milling machine using an end-mill. International Journal of Production Research **39**(7), 1367–1383 (2001). https://doi.org/10.1080/00207540010023529
31. Shin, K., McKay, N.: A dynamic programming approach to trajectory planning of robotic manipulators. IEEE Transactions on Automatic Control **31**(6), 491–500 (1986). https://doi.org/10.1109/TAC.1986.1104317
32. Smirnov, V., Plyusnin, V., Mirzaeva, G.: Energy efficient trajectories of industrial machine tools with parallel kinematics. In: 2013 IEEE International Conference on Industrial Technology (ICIT). pp. 1267–1272 (2013). https://doi.org/10.1109/icit.2013.6505855
33. Tassa, Y., Mansard, N., Todorov, E.: Control-limited differential dynamic programming. In: 2014 IEEE International Conference on Robotics and Automation (ICRA). pp. 1168–1175. IEEE (2014). https://doi.org/10.1109/ICRA.2014.6907001
34. Zargarbashi, S., Khan, W., Angeles, J.: Posture optimization in robot-assisted machining operations. Mechanism and Machine Theory **51**, 74–86 (2012). https://doi.org/10.1016/j.mechmachtheory.2011.11.017
35. Zhu, W., Qu, W., Cao, L., Yang, D., Ke, Y.: An off-line programming system for robotic drilling in aerospace manufacturing. The International Journal of Advanced Manufacturing Technology **68**(9-12), 2535–2545 (2013). https://doi.org/10.1007/s00170-013-4873-5
36. Žlajpah, L.: On orientation control of functional redundant robots. In: Robotics and Automation (ICRA), 2017 IEEE International Conference on. pp. 2475–2482. IEEE (2017). https://doi.org/10.1109/ICRA.2017.7989288