An Overview of Multi-task Control for Redundant Robot Based on Quadratic Programming

Qingkai Li[1], Yanbo Pang[1], Wenhan Cai[1], Yushi Wang[1], Qing Li[1], and Mingguo Zhao[1,2(✉)]

[1] Department of Automation, Tsinghua University, Beijing, China
mgzhao@mail.tsinghua.edu.cn

[2] Beijing Innovation Center for Future Chips, Tsinghua University, Beijing, China

Abstract. Redundant degrees of freedom provide a feasible solution for robots to better interact with environments, but it also leads to an increase in the complexity of kinematic and dynamic control. This paper presents an overview of methods that realize the whole-body control (WBC) of a redundant robot with a quadratic programming (QP) approach. We first induce the general QP form from the single-task control method for redundant robots and describe WBC as a multi-task control problem in the form of multi-objective optimization (MOO). Then different QP algorithms for multi-task control with fixed and transitional priority are introduced. Weighted quadratic programming (WQP) and hierarchical quadratic programming (HQP) for fixed priority, and generalized hierarchical control (GHC) and recursive hierarchical projection (RHP) for transitional priority will be explicated, respectively. Finally, a toolkit implementing the above algorithms is provided, based on which simulation results are presented and remarks on different algorithms are given.

Keywords: quadratic programming · whole-body control · redundant robot · multi-task control · multi-objective optimization

1 Introduction

With increasing degrees-of-freedom (DoFs) in robot actuation, robots are capable of human-like versatility, achieving multiple task objectives simultaneously and adapting to changing environments. Consider the task of transferring a cup of water, a humanoid robot must consider the attitude of the water cup while grabbing and moving the cup, and it will also need to adjust to new tasks in time if the environment is disturbed by other people or objects. Although redundant DoFs of robots bring the possibility to achieve these tasks, how to coordinate the many tasks so as to achieve a reasonable control is an important research problem.

A main approach to the multi-task control problem is whole-body control (WBC), whose basic idea is to assign different priority to multiple tasks. Dating back to the 1980 s, nullspace projection based methods were already used

Z. Deng (Ed.): CIAC 2023, LNEE 1082, pp. 641–666, 2023.
https://doi.org/10.1007/978-981-99-6187-0_64

to deal with the coupling between tasks [7,16,26,29,31]. Tasks with lower priority were projected into the nullspace of tasks with higher priority and then the pseudo-inverse was calculated to solve the problem. However, as the control scenario evolves, inequality constraints such as friction cone constraint, maximum torque constraint, etc., have also become control problems needed to be considered. Because the nullspace and pseudo-inverse based methods can only deal with equality constraints, a more popular solution dealing with inequality constraints is to transform the control problem into a quadratic programming (QP) problem to implement WBC [3,8,20,21,27]. In QP problems, the control tasks are abstracted as optimization objectives. One task corresponds to one objective, while multiple tasks can correspond to a single objective or to multiple objectives. For example, the horizontal position x and vertical position y of the end effector as two control tasks can be abstracted into two optimization objectives, e.g., minimizing the tracking error in each position, but they can also be abstracted as one optimization objective, e.g., minimizing the total error of x and y. In robot control, what often needs to be considered is the scenario of abstracting multi-tasks into multiple objectives, i.e., the multi-objective optimization (MOO) problem of QP.

One of the most straightforward strategies to tackle multiple tasks is using the commands from the nullspace projection method as part of the references and dealing with other inequality constraints in one QP [17,18], which here named projected quadratic programming(PQP). Another method is to assign different weights to the optimization objectives of different tasks, thus aggregating the optimization problems of multiple tasks into a weighted quadratic programming (WQP) [1,23]. However, task priorities are not strictly hierarchical in both the above two methods, which means tasks with lower priority will influence the performance of higher ones. In the contrast, the hierarchical quadratic programming (HQP) method solves QP problems with different priorities in turn, using the results from higher priority tasks as constraints for lower priority tasks to ensure a strict hierarchy of priorities [4,9,15].

In order to adapt to more complex and flexible scenarios, it is also necessary to adjust the priority between different tasks, to which the key is to ensure the continuity of the adjusting process. A simple approach would be to continuously change the weights based on WQP to alter the priority of tasks [28], but limited by the vague meaning of weights, it is difficult to design the change rules of different weights under the same standard. Another method is to achieve the transition between different priorities based on the intermediate variables of task space, where strict hierarchical priorities before and after the transition are maintained [22]. Based on this method, [19] raised a continuous transition HQP(CT-HQP) to include inequality constraints, but HQP has to be done for the optimization problem before and after the transition in order to obtain the intermediate variables, which greatly increases the computational complexity. [6] introduced the continuous nullspace projections method. On this basis, [24] proposed the generalized projection matrix, and a generalized hierarchical control (GHC) framework was established in [25] to implement hierarchical control and

priority transition. Recently, [12] combined the continuous nullspace projection matrix and HQP to propose an iterative form of projection matrix transition, and established the recursive hierarchical control (RHP) framework for priority transition similar to HQP. Main QP algorithms for multi-task control of robots are summarized in Fig. 1.

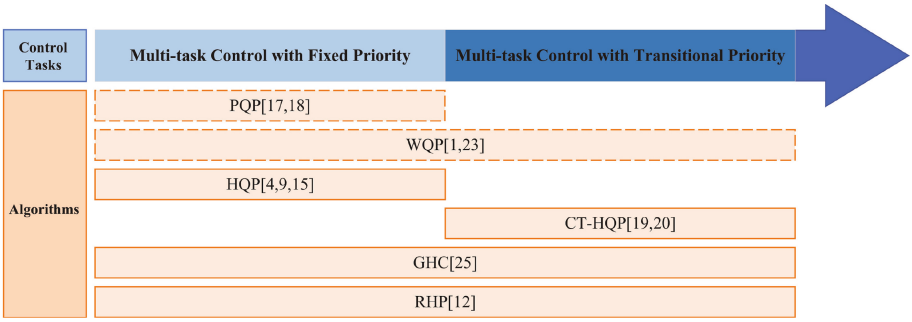


Fig. 1. QP algorithms for robot multi-task control. Tasks are strictly hierarchical in HQP, CT-HQP, GHC and RHP, but not in PQP and WQP, which are depicted in dotted box.

This paper provides an overview of algorithms related to WBC of redundant robots using QP, then compares and discusses the relationship between different algorithms. The single-task redundant robot control problem is first introduced in Sect. 2 based on a specific scenario, and is then described as a single-objective quadratic programming problem. The multi-task control problem is later described as an MOO problem similarly. In Sect. 3, algorithms for multi-task control problem, which mainly focus on weighting strategy and hierarchical optimization strategy, will be given. Algorithms for fixed priorities are presented in Sect. 3.1, whereas algorithms for transition priorities are presented in Sect. 3.2, and both will be discussed in Sect. 3.3. Different algorithms will be compared through simulation results in Sect. 4, and concluding remarks will be given in Sect. 5.

2 System Model and Preliminary Analysis

Consider a robot with n DoFs and r task vectors, the forward kinematics can be written as

$$\begin{cases} \boldsymbol{x}_i = \boldsymbol{f}_i(\boldsymbol{q}) \\ \dot{\boldsymbol{x}}_i = \boldsymbol{J}_i \dot{\boldsymbol{q}} \\ \ddot{\boldsymbol{x}}_i = \boldsymbol{J}_i \ddot{\boldsymbol{q}} + \dot{\boldsymbol{J}}_i \dot{\boldsymbol{q}}, \end{cases} \tag{1}$$

where $\boldsymbol{x}_i \in \mathbb{R}^{m_i}(i = 1, ..., r)$ is the vector of task i, for example, the position or orientation of manipulator, $\boldsymbol{J}_i(\boldsymbol{q}) = \dfrac{\partial \boldsymbol{f}_i(\boldsymbol{q})}{\partial \boldsymbol{q}}$ is the Jacobian matrix and $\boldsymbol{q} \in \mathbb{R}^n$ is the general vector. For a redundant robot, we have $m_i \leqslant n, \forall i = 1, \cdots, r$.

2.1 Single-task Control Method

For the ith task, the objective can be achieved by setting an expected velocity $\dot{\boldsymbol{x}}_i^*$ or acceleration $\ddot{\boldsymbol{x}}_i^*$. Take the feedback control problem

$$\begin{aligned} \dot{\boldsymbol{x}}_i^* &= \dot{\boldsymbol{x}}_i^{ref} + \boldsymbol{k}_p(\boldsymbol{x}_i^{ref} - \boldsymbol{x}_i^{fb}) \\ \ddot{\boldsymbol{x}}_i^* &= \ddot{\boldsymbol{x}}_i^{ref} + \boldsymbol{k}_d(\dot{\boldsymbol{x}}_i^{ref} - \dot{\boldsymbol{x}}_i^{fb}) + \boldsymbol{k}_p(\boldsymbol{x}_i^{ref} - \boldsymbol{x}_i^{fb}) \end{aligned} \tag{2}$$

for example, where $\boldsymbol{k}_p, \boldsymbol{k}_d$ are the PD gains of the PD controller, $\boldsymbol{x}_i^{fb}, \dot{\boldsymbol{x}}_i^{fb}$ are the feedback positions and velocities of task i, meanwhile $\boldsymbol{x}_i^{ref}, \dot{\boldsymbol{x}}_i^{ref}, \ddot{\boldsymbol{x}}_i^{ref}$ are the reference positions, velocities and accelerations of task i. The expected angular velocity of joints $\dot{\boldsymbol{q}}^*$ can be obtained by solving a QP problem

$$\dot{\boldsymbol{q}}^* \in \arg\min_{\dot{\boldsymbol{q}}} \|\boldsymbol{J}_i\dot{\boldsymbol{q}} - \dot{\boldsymbol{x}}_i^*\|^2 + w_r\|\dot{\boldsymbol{q}}\|^2, \tag{3}$$

where $w_r \geqslant 0$ is the weight of the regular term. By selecting $w_r > 0$, the quadratic matrix of the QP problem is positive definite to avoid algorithm singularity, and thus (3) has a solution. The above QP problem at the velocity level is called the damped least-squares methods [5].

For a torque-controlled robot, the expected acceleration $\ddot{\boldsymbol{x}}_i^*$ must be taken into consideration. Considering a robot with n_a actuation joints, the dynamical equation for the generalized coordinates $\boldsymbol{q}$ is given as

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{S}_a^T\boldsymbol{\tau} + \boldsymbol{J}_c^T\boldsymbol{\omega}_c, \tag{4}$$

where $\boldsymbol{M}(\boldsymbol{q}) \in \mathbb{R}^{n\times n}$ is the generalized inertia matrix, $\boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \in \mathbb{R}^n$ is the non-linear term of Coriolis force, centrifugal force and gravity, $\boldsymbol{S}_a^T \in \mathbb{R}^{n\times n_a}$ is the transpose of the torque selection matrix, $\boldsymbol{\tau} \in \mathbb{R}^{n_a}$ is the actuation torque of joints, $\boldsymbol{J}_c^T$ is the transpose of Jacobian matrix of contact points with the environment and $\boldsymbol{\omega}_c$ is the external wrench applied to the robot at the contact points. The inverse dynamical, i.e. obtaining the actuation torque from the expected acceleration, can be described as the following QP problem

$$\begin{aligned} \arg\min_{\ddot{\boldsymbol{q}}, \boldsymbol{\tau}, \boldsymbol{\omega}_c} \ & \|\boldsymbol{J}_i\ddot{\boldsymbol{q}} - (\ddot{\boldsymbol{x}}_i^* - \dot{\boldsymbol{J}}_i\dot{\boldsymbol{q}})\|^2 + \|\boldsymbol{W}_r[\ddot{\boldsymbol{q}}^T, \boldsymbol{\tau}^T, \boldsymbol{\omega}_c^T]^T\|^2 \\ \text{s.t. } & \boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{S}_a^T\boldsymbol{\tau} + \boldsymbol{J}_c^T\boldsymbol{\omega}_c. \end{aligned} \tag{5}$$

$\boldsymbol{\omega}_c$ can be considered as constraints instead of optimization variables when it is known or can be denoted as a linear combination of other optimization variables, based on specific scenarios [14].

The QP problem given in (3) and (5) can be written in the following general form

$$\begin{aligned} \underset{\boldsymbol{x},\boldsymbol{w}_i,\boldsymbol{v}_i}{\arg\min} \ & \|\boldsymbol{w}_i\|^2 + \|\boldsymbol{v}_i\|^2 && \text{(6a)} \\ \text{s.t. } & \boldsymbol{A}_i\boldsymbol{x} + \boldsymbol{w}_i = \boldsymbol{b}_i && \text{(6b)} \\ & \boldsymbol{ld}_{si} \leqslant \boldsymbol{C}_{si}\boldsymbol{x} + \boldsymbol{v}_i \leqslant \boldsymbol{ud}_{si} && \text{(6c)} \\ & \boldsymbol{ld}_{hi} \leqslant \boldsymbol{C}_{hi}\boldsymbol{x} \leqslant \boldsymbol{ud}_{hi} \quad , && \text{(6d)} \end{aligned}$$

where equality tasks are described as equality constraints in (6b) and inequality tasks are described as inequality constraints in (6c). (6d) is the hard constraint, which must be satisfied in a task, such as the dynamical equation in (5). $\boldsymbol{w}_i, \boldsymbol{v}_i$ are the slack variables for equality and inequality constraints, respectively. $\boldsymbol{A}_i, \boldsymbol{b}_i$ are the task matrix and vector for equality constraints, and $\boldsymbol{C}_{si}$,$\boldsymbol{ld}_{si}$,$\boldsymbol{ud}_{si}$ are the constraint matrix, lower and upper bound for inequality constraints, respectively. In fact, (6b) can be written in the form of (6c) mathematically, but it is retained because equality tasks are more common in robots, and algorithms can be simplified considering equality tasks. For example, when only (6a–6b) are included in an optimization problem, it can be solved by taking the pseudo-inverse [9]. The result is given by $\boldsymbol{x}^* = \boldsymbol{A}_i^{\dagger}\boldsymbol{b}_i$, where $\boldsymbol{A}_i^{\dagger}$ is the M-P inverse matrix of $\boldsymbol{A}_i$. Although calculating pseudo-inverse is more computationally efficient than solving an optimization problem, and multi-task control based on projection matrix has been studied thoroughly [7], friction cone constraint, joint torque limitation, etc. need to be considered in actual robot control problems, so inequality constraints in (6) must be included in the optimization problem for a more accurate control. Such tasks cannot be solved by pseudo-inverse so further discussion has to be made on multi-objective optimization.

2.2 Multi-task Control Method

The general QP form for a single task has been derived as (6), which is a single-objective optimization problem with constraints. Similarly, solution for multi-task control can be described in the form of an MOO problem with constraints, as

$$\begin{aligned} \underset{\boldsymbol{x},\boldsymbol{w}_i,\boldsymbol{v}_i}{\arg\min} \ & \boldsymbol{F} = [f_1, \cdots, f_r] \\ \text{s.t. } & f_i = \|\boldsymbol{w}_i\|^2 + \|\boldsymbol{v}_i\|^2 \\ & \boldsymbol{A}_i\boldsymbol{x} + \boldsymbol{w}_i = \boldsymbol{b}_i \\ & \boldsymbol{ld}_{si} \leqslant \boldsymbol{C}_{si}\boldsymbol{x} + \boldsymbol{v}_i \leqslant \boldsymbol{ud}_{si} \\ & \boldsymbol{ld}_{hi} \leqslant \boldsymbol{C}_{hi}\boldsymbol{x} \leqslant \boldsymbol{ud}_{hi} \\ & i = 1, 2, \cdots, r. \end{aligned} \tag{7}$$

In multi-task control, tasks are often coupled with each other so that they can not be satisfied simultaneously because of interference. As a result, an optimal solution $\boldsymbol{x}^*$ that satisfies $f_i(\boldsymbol{x}^*) \leqslant f_i(\boldsymbol{x}), \forall i \in \mathcal{I}$ often does not exist, where

for simplicity, we define $\mathcal{I} = \{1, 2, \cdots, r\}$. Naturally, we would like to assign different priority to different tasks according to their importance, so that an extra constraint concerning the priority of tasks must be defined to solve the MOO problem.

3 QP Algorithms for Multi-task Control

In this section, existing algorithms for multi-task control problem will be introduced. Algorithms for multi-task control with fixed task priority will be discussed in Sect. 3.1 and those for multi-task control with transition task priority will be presented in Sect. 3.2.

3.1 Multi-task Control with Fixed Priority

MOO problem itself is difficult to solve directly, and a common approach is to transform it into a single-objective optimization problem. In robot multi-task control, this is achieved by considering the trade-off between different tasks to give a single evaluation. How multi-task control is realized in different algorithms are introduced in this section.

Weighted Quadratic Programming. One of the most commonly used methods in robotic MOO is the linear weighting method, where each task objective $f_i (i \in \mathcal{I})$ in (7) is given a weight λ_i and then added together to form a single-objective optimization problem of $\sum_{i=1}^{r} \lambda_i f_i$. The sum of multiple quadratic functions, where equality constraints and inequality constraints are given different weights, is still a quadratic function. Therefore the WQP problem can be described as

$$\arg\min_{\boldsymbol{x}, \overline{\boldsymbol{w}}, \overline{\boldsymbol{v}}} \ \|\overline{\boldsymbol{w}}\|_{\boldsymbol{Q}_e} + \|\overline{\boldsymbol{v}}\|_{\boldsymbol{Q}_i} \tag{8a}$$

$$\text{s.t. } \overline{\boldsymbol{A}}\boldsymbol{x} + \overline{\boldsymbol{w}} = \overline{\boldsymbol{b}} \tag{8b}$$

$$\overline{\boldsymbol{l}\boldsymbol{d}}_s \leqslant \overline{\boldsymbol{C}}_s \boldsymbol{x} + \overline{\boldsymbol{v}} \leqslant \overline{\boldsymbol{u}\boldsymbol{d}}_s \tag{8c}$$

$$\overline{\boldsymbol{l}\boldsymbol{d}}_h \leqslant \overline{\boldsymbol{C}}_h \boldsymbol{x} \leqslant \overline{\boldsymbol{u}\boldsymbol{d}}_h \tag{8d}$$

The new task matrix and task vector are obtained by stitching together all the task matrices, i.e.

$$\overline{\boldsymbol{A}} = \begin{bmatrix} \boldsymbol{A}_1 \\ \vdots \\ \boldsymbol{A}_r \end{bmatrix}, \overline{\boldsymbol{b}} = \begin{bmatrix} \boldsymbol{b}_1 \\ \vdots \\ \boldsymbol{b}_r \end{bmatrix} \tag{9}$$

$\|\overline{\boldsymbol{w}}\|_{\boldsymbol{Q}_e} = \overline{\boldsymbol{w}}^T \boldsymbol{Q}_e \overline{\boldsymbol{w}}$ is the weighted norm and the weight matrix $\boldsymbol{Q}_e$ is a diagonal matrix, whose diagonal elements are the weights of different equality tasks. $\boldsymbol{Q}_i$ is the weight matrix for inequality tasks, similarly. (8d) represents all of the hard

constraints. Obviously, the weight matrix $\boldsymbol{Q}_e$ and $\boldsymbol{Q}_i$ reflect the importance of different tasks, and the larger the diagonal element of the matrix, the more important the corresponding task is. This algorithm has a high computational efficiency, especially in [1,23] where the soft constraints (8c) are not included. The slack variable $\boldsymbol{w}_i$ can be eliminated by substituting (8b) into (8a) so $\boldsymbol{x}$ is the only optimization variable. In this case, the computational efficiency is close to that of the single-task optimization. However, the biggest drawback of this algorithm is the vague meaning of weights. It is difficult to compare the importance of tasks under different dimensions. In addition, even if the weights can be compared, the importance between tasks is only a relative concept and different tasks are still coupled. Thus tasks are not strictly hierarchical in WQP.

Hierarchical Quadratic Programming. Another way to handle the multi-task control problem is to choose one of the sub-objectives as the optimization objective and the other sub-objectives as the constraints. The HQP proposed by [15] describes the optimization problem on the k-th priority level as

$$\arg\min_{\boldsymbol{x}_k, \overline{\boldsymbol{v}}_k} \left\|\overline{\boldsymbol{A}}_k \boldsymbol{x}_k - \overline{\boldsymbol{b}}_k\right\|_{\boldsymbol{Q}_e} + \|\overline{\boldsymbol{v}}_k\|_{\boldsymbol{Q}_i} \tag{10a}$$

$$\text{s.t.} \overline{\boldsymbol{ld}}_{s,k} \leqslant \overline{\boldsymbol{C}}_{s,k} \boldsymbol{x}_k + \overline{\boldsymbol{v}}_k \leqslant \overline{\boldsymbol{ud}}_{s,k} \tag{10b}$$

$$\overline{\boldsymbol{ld}}_h \leqslant \overline{\boldsymbol{C}}_h \boldsymbol{x}_k \leqslant \overline{\boldsymbol{ud}}_h \tag{10c}$$

$$\overline{\boldsymbol{A}}_{k-1}^{aug} \boldsymbol{x}_k = \overline{\boldsymbol{A}}_{k-1}^{aug} \boldsymbol{x}_{k-1}^* \tag{10d}$$

$$\overline{\boldsymbol{ld}}_{k-1}^{aug} \leqslant \overline{\boldsymbol{C}}_{k-1}^{aug} \boldsymbol{x}_k + \overline{\boldsymbol{v}}_{k-1}^{aug\,*} \leqslant \overline{\boldsymbol{ud}}_{k-1}^{aug} \tag{10e}$$

where $\overline{\boldsymbol{A}}_k, \overline{\boldsymbol{b}}_k$ are the task matrix and vector of all the equality tasks on the k-th priority level, $\overline{\boldsymbol{C}}_{s,k}, \overline{\boldsymbol{ld}}_{s,k}, \overline{\boldsymbol{ud}}_{s,k}, \overline{\boldsymbol{v}}_k$ are the constraint matrix, lower and upper bound and the slack variables for inequality constraints on the k-th priority level, respectively. Constraints in (10d) take the optimization results of equality task from all of the higher priority levels into consideration. $\boldsymbol{x}_{k-1}^*$ is the optimization result from the upper level and $\overline{\boldsymbol{A}}_{k-1}^{aug}$ is the task matrix of all the tasks on the upper $k-1$ levels, which is defined as

$$\overline{\boldsymbol{A}}_{k-1}^{aug} = \begin{bmatrix} \overline{\boldsymbol{A}}_1 \\ \vdots \\ \overline{\boldsymbol{A}}_{k-1} \end{bmatrix} \tag{11}$$

Constraints in (10e) take the optimization results of inequality tasks from all of the higher priority levels into consideration. $\overline{\boldsymbol{C}}_{k-1}^{aug}, \overline{\boldsymbol{ld}}_{k-1}^{aug}, \overline{\boldsymbol{ud}}_{k-1}^{aug}$ are the constraint matrix, lower and upper bound for inequality constraints on the upper $k-1$ priority levels, and $\overline{\boldsymbol{v}}_{k-1}^{aug\,*}$ is the optimal solution for the upper $k-1$ priority

levels respectively. They are defined as

$$\overline{\boldsymbol{C}}_{k-1}^{aug} = \begin{bmatrix} \overline{\boldsymbol{C}}_{s,1} \\ \vdots \\ \overline{\boldsymbol{C}}_{s,k-1} \end{bmatrix}, \overline{\boldsymbol{v}}_{k-1}^{aug\,*} = \begin{bmatrix} \overline{\boldsymbol{v}}_1^* \\ \vdots \\ \overline{\boldsymbol{v}}_{k-1}^* \end{bmatrix}, \overline{\boldsymbol{ld}}_{k-1}^{aug} = \begin{bmatrix} \overline{\boldsymbol{ld}}_{s,1} \\ \vdots \\ \overline{\boldsymbol{ld}}_{s,k-1} \end{bmatrix}, \overline{\boldsymbol{ud}}_{k-1}^{aug} = \begin{bmatrix} \overline{\boldsymbol{ud}}_{s,1} \\ \vdots \\ \overline{\boldsymbol{ud}}_{s,k-1} \end{bmatrix} \tag{12}$$

respectively.

Setting the optimization results of tasks from all of the higher levels as constraints in (10d–10e) will ensure that results from the current level will not have an impact on results from higher level, so as to guarantee that tasks are strictly hierarchical.

Notice that (10c) means that hard constraints are considered on every level, so that all the hard constraints have the highest priority level. This is because hard constraints like dynamical equation and maximum torque limitation must be satisfied in robotic control tasks. If they are not considered at the beginning, the constraints from higher level may conflict with the hard constraints and no feasible solution can be found, as depicted in Fig. 2.

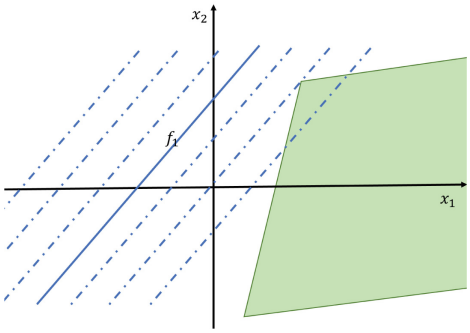


Fig. 2. No feasible solution can be found when hard constraints are not arranged to the highest level. Dotted lines are contours of $f_1(\boldsymbol{x})$, the solid line is the task constraint and the green area is the solution space of the hard constraint. (Color figure online)

In (10), all optimization results from the upper $k-1$ levels need to be added as constraints when solving the QP problem of the k-th level, so as k increases, constraints in (10d,10e) increases and computational efficiency deteriorates significantly. To solve this problem, the constraints in (10d) are equivalently expressed as

$$\boldsymbol{x}_k = \boldsymbol{N}_{k-1}\boldsymbol{u}_k + \boldsymbol{x}_{k-1}^* \tag{13}$$

according to [4], where $\boldsymbol{N}_{k-1} \in \mathbb{R}^{n\times(n-m_{k-1})}$ is the matrix composed of the basis of the nullspace of $\overline{\boldsymbol{A}}_{k-1}^{aug}$, $m_{k-1} = \text{Rank}(\overline{\boldsymbol{A}}_{k-1}^{aug})$. $\boldsymbol{N}_{k-1}$ can be obtained by doing

QR decomposition or singular value decomposition(SVD) to $\overline{\boldsymbol{A}}_{k-1}^{aug}$. For example,

$$\overline{\boldsymbol{A}}_{k-1}^{aug} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T, \tag{14}$$

then $\boldsymbol{N}_{k-1}$ consists of the last $n - m_{k-1}$ columns of $\boldsymbol{V}$. $\boldsymbol{x}_k$ derived from(13) satisfies

$$\overline{\boldsymbol{A}}_{k-1}^{aug}\boldsymbol{x}_k = \underbrace{\overline{\boldsymbol{A}}_{k-1}^{aug}\boldsymbol{N}_{k-1}}_{\boldsymbol{O}}\boldsymbol{u}_k + \overline{\boldsymbol{A}}_{k-1}^{aug}\boldsymbol{x}_{k-1}^* = \overline{\boldsymbol{A}}_{k-1}^{aug}\boldsymbol{x}_{k-1}^*, \forall \boldsymbol{u}_k \in \mathbb{R}^{n-m_{k-1}}. \tag{15}$$

[4] presented a nullspace-based HQP that eliminates the constraint of (10d) and uses the parameter vector $\boldsymbol{u}_k$ as the optimization variable of the k-th priority level. However, the case with inequality constraints are not discussed, a more general form containing inequality constraints is presented here as

$$\arg\min_{\boldsymbol{u}_k, \overline{\boldsymbol{v}}_k} \ \left\|\overline{\boldsymbol{A}}_k(\boldsymbol{N}_{k-1}\boldsymbol{u}_k + \boldsymbol{x}_{k-1}^*) - \overline{\boldsymbol{b}}_k\right\|_{\boldsymbol{Q}_e} + \|\overline{\boldsymbol{v}}_k\|_{\boldsymbol{Q}_i} \tag{16a}$$

$$\text{s.t. } \overline{\boldsymbol{ld}}_{s,k} \leqslant \overline{\boldsymbol{C}}_{s,k}(\boldsymbol{N}_{k-1}\boldsymbol{u}_k + \boldsymbol{x}_{k-1}^*) + \overline{\boldsymbol{v}}_k \leqslant \overline{\boldsymbol{ud}}_{s,k} \tag{16b}$$

$$\overline{\boldsymbol{ld}}_h \leqslant \overline{\boldsymbol{C}}_h(\boldsymbol{N}_{k-1}\boldsymbol{u}_k + \boldsymbol{x}_{k-1}^*) \leqslant \overline{\boldsymbol{ud}}_h \tag{16c}$$

$$\overline{\boldsymbol{ld}}_{k-1}^{aug} \leqslant \overline{\boldsymbol{C}}_{k-1}^{aug}(\boldsymbol{N}_{k-1}\boldsymbol{u}_k + \boldsymbol{x}_{k-1}^*) + \overline{\boldsymbol{v}}_{k-1}^{aug\,*} \leqslant \overline{\boldsymbol{ud}}_{k-1}^{aug}, \tag{16d}$$

for which the optimal solution is $\boldsymbol{u}_k^*$. By substituting $\boldsymbol{u}_k^*$ into (13), the optimal solution $\boldsymbol{x}_k^*$ for (10) can be derived as

$$\boldsymbol{x}_k^* = \boldsymbol{N}_{k-1}\boldsymbol{u}_k^* + \boldsymbol{x}_{k-1}^*, \tag{17}$$

In addition, as the solution proceeds, the dimension of $\overline{\boldsymbol{A}}_k^{aug}$ increases, and solving $\boldsymbol{N}_k$ directly through $\overline{\boldsymbol{A}}_k^{aug}$ will consume a lot of time. Because the previous task space does not change when a new task matrix is added, the HQP algorithm with recursive form shown in Fig. 3 is often used [4,13,14], where $\text{null}(\widehat{\boldsymbol{A}}_k)$ is the matrix composed of the basis of the nullspace of $\widehat{\boldsymbol{A}}_k$.

Obviously, when $\hat{\boldsymbol{A}}_k$ is a full rank matrix, $\boldsymbol{N}_k = \boldsymbol{O}$ and subsequent optimal solutions will not change according to (17), so the algorithm can terminate at this step. In terms of robot control, $\hat{\boldsymbol{A}}_k$ has full rank means that at this point the robot's actuation joints have no more redundant DoFs for subsequent tasks while ensuring the higher priority tasks, therefore the subsequent tasks are not reachable.

Compared with WQP, multiple QP problems need to be solved in HQP, so that it is a more time consuming approach for WBC. However, with the help of algorithm based on parametrization of the optimization variables and the recursive form of the nullspace basis matrix, computational efficiency significantly improves. In addition, [9] proposed a numerical algorithm for the HQP problem using the active set method, which ensures that (16) with up to 40-dimensional optimization variables can be solved in less than 1 ms. The real-time performance of WBC can thus be guaranteed.

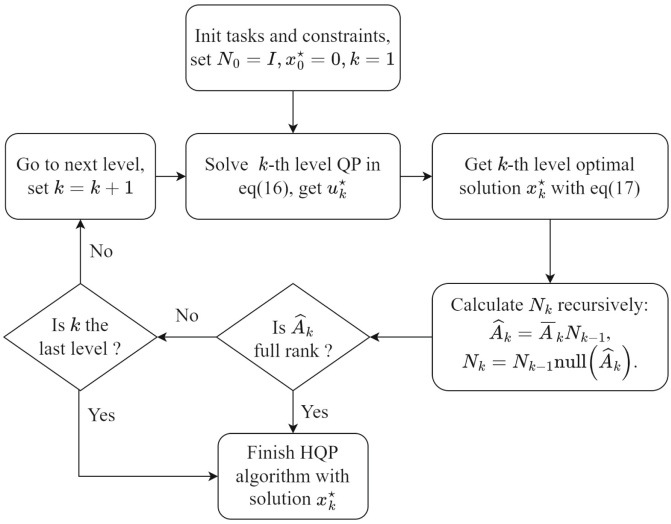


Fig. 3. Diagram of HQP algorithm with recursive form.

3.2 Multi-task Control with Transitional Priority

Changing the weights continuously is a straightforward approach to priority transition in WQP, but this method relies greatly on the setting of tasks to determine the form of transition, so it would not be discussed here. The transition algorithm based on intermediate variables in [19,20] can simultaneously realize the transition of inequality constraints, but it is often computationally demanding and does not have obvious advantages over the transition method based on projection matrix. As a result, this section mainly focuses on priority transition algorithms based on continuous projection matrix.

Generalized Hierarchical Control. The generalized hierarchical control (GHC) proposed in [25] was extended from (8) by using the generalized projection matrix. In GHC, different equality tasks $f_i = \|\boldsymbol{A}_i\boldsymbol{x} - \boldsymbol{b}_i\|^2$ no longer sum with weights. Each task corresponds to a new optimization variable $\boldsymbol{x}_i$, and is solved independently. The optimal solution $\boldsymbol{x}_i^*$ for each task is then summed by the generalized projection matrix. The form of GHC is presented in the following as

$$\arg\min_{\boldsymbol{x}',\overline{\boldsymbol{v}}} \sum_{i=1}^{r} \|\boldsymbol{A}_i\boldsymbol{x}_i - \boldsymbol{b}_i\|^2 + \|\overline{\boldsymbol{v}}\|_{\boldsymbol{Q}_i} \tag{18a}$$

$$\text{s.t. } \overline{\boldsymbol{ld}}_s \leqslant \overline{\boldsymbol{C}}_s\boldsymbol{x} + \overline{\boldsymbol{v}} \leqslant \overline{\boldsymbol{ud}}_s \tag{18b}$$

$$\overline{\boldsymbol{ld}}_h \leqslant \overline{\boldsymbol{C}}_h\boldsymbol{x} \leqslant \overline{\boldsymbol{ud}}_h \tag{18c}$$

$$\boldsymbol{x} = \sum_{i=1}^{r} \boldsymbol{P}_i^{gp}\boldsymbol{x}_i = \boldsymbol{P}\boldsymbol{x}', \tag{18d}$$

where $\boldsymbol{P} = [\boldsymbol{P}_1^{gp} \cdots \boldsymbol{P}_r^{gp}]$, $\boldsymbol{x}' = \left[\boldsymbol{x}_1^T \cdots \boldsymbol{x}_r^T\right]^T$.

The generalized projection matrix $\boldsymbol{P}_i^{gp}$ is similar to the continuous projection matrix in [6], but tasks are not projected into the nullspace of a specific task, rather projected into the nullspace of all the tasks by $\boldsymbol{P}_i^{gp}$. $\boldsymbol{P}_i^{gp}$ is a function of $\alpha_{ij} \in [0,1]$, which portrays the extent of projection from task i to the nullspace of task j. When $\alpha_{ij} = 0$, task i will not be projected to the nullspace of task j, meaning that task i will not be affected by task j, i.e., task i has a higher priority level. When $\alpha_{ij} = 1$, task i will be fully projected to the nullspace of task j, meaning that task i will not affect task j, i.e., task i has a lower priority level. When $0 < \alpha_{ij} < 1$, the priority is not strictly hierarchical and the larger α_{ij} is, the higher the importance of task j is compared to task i. A priority matrix can therefore be defined as

$$\Psi_{ghc} = \begin{bmatrix} \alpha_{11} & \cdots & \alpha_{1r} \\ \vdots & \ddots & \vdots \\ \alpha_{r1} & \cdots & \alpha_{rr} \end{bmatrix} \in \mathbb{R}^{r \times r}, \tag{19}$$

where $\alpha_{ij} + \alpha_{ji} = 1, \forall i \neq j$ for the consistency of priority between a pair of tasks. The generalized projection matrix $\boldsymbol{P}_i^{gp}(\Psi_{ghc})$ for task i can be calculated in the following three steps:

Step 1: Arrange the task matrix $\boldsymbol{A}_j$ in descending order of $\alpha_{ij}, j = 1, \cdots, r$ into an augmented matrix $\overline{\boldsymbol{A}}_i^s = \left[\boldsymbol{A}_{s1}^T \cdots \boldsymbol{A}_{sr}^T\right]^T$;

Step 2: Perform Gaussian elimination to the row vector of $\overline{\boldsymbol{A}}_i^s$ and get a set of orthogonal bases $\boldsymbol{B}_i \in \mathbb{R}^{m_i \times n}$ of its row space, where m_i is the rank of $\overline{\boldsymbol{A}}_i^s$. Construct an m_i-dimensional diagonal matrix $\boldsymbol{\Lambda}_i^s = \text{diag}(\alpha_{is_1}, \cdots, \alpha_{is_1}, \alpha_{is_2}, \cdots)$ according to the dimension of each task matrix in $\overline{\boldsymbol{A}}_i^s$;

Step 3: Calculate the generalized projection matrix

$$\boldsymbol{P}_i^{gp} = \boldsymbol{I}_n - \boldsymbol{B}_i^T \boldsymbol{\Lambda}_i^s \boldsymbol{B}_i. \tag{20}$$

The diagonal elements of (19) have different meaning from other elements. When $\alpha_{ii} = 1$, task i will be projected into its own nullspace, which is equivalent to deleting this task. When $\alpha_{ii} = 0$, task i is fully activated. In conclusion, the priority between different tasks is defined by Ψ_{ghc}, and by tuning the parameters of Ψ_{ghc}, priority transition can be realized.

Recursive Hierarchical Projection. The algorithm proposed in [12] realize priority transition of equality tasks with the continuous transition of the nullspace projection matrix based on HQP. This algorithm is called RHP-HQP and is similar to (16). All of the r tasks are divided into n_l levels, and the k-th level QP problem has the following form:

$$\arg\min_{\boldsymbol{u}_k, \boldsymbol{v}_k} \left\| \boldsymbol{A}_k^{A,s}(\boldsymbol{P}_{k-1}^{rhp}\boldsymbol{u}_k + \boldsymbol{x}_{k-1}^*) - \hat{\boldsymbol{b}}_k^{A,s} \right\|_{\boldsymbol{Q}_e} + \|\overline{\boldsymbol{v}}_k\|_{\boldsymbol{Q}_i} \tag{21a}$$

$$\text{s.t. } \overline{\boldsymbol{ld}}_{s,k} \leqslant \overline{\boldsymbol{C}}_{s,k}(\boldsymbol{P}_{k-1}^{rhp}\boldsymbol{u}_k + \boldsymbol{x}_{k-1}^*) + \overline{\boldsymbol{v}}_k \leqslant \overline{\boldsymbol{ud}}_{s,k} \tag{21b}$$

$$\overline{\boldsymbol{ld}}_h \leqslant \overline{\boldsymbol{C}}_h(\boldsymbol{P}_{k-1}^{rhp}\boldsymbol{u}_k + \boldsymbol{x}_{k-1}^*) \leqslant \overline{\boldsymbol{ud}}_h \tag{21c}$$

$$\overline{\boldsymbol{ld}}_{k-1}^{aug} \leqslant \overline{\boldsymbol{C}}_{k-1}^{aug}(\boldsymbol{P}_{k-1}^{rhp}\boldsymbol{u}_k + \boldsymbol{x}_{k-1}^*) + \overline{\boldsymbol{v}}_{k-1}^{aug\,*} \leqslant \overline{\boldsymbol{ud}}_{k-1}^{aug} \tag{21d}$$

where the inequality constraint matrix, upper and lower bound in (21b–21d) are defined in the same way as (16), the optimal solution $\boldsymbol{x}_i^*$ is defined as

$$\boldsymbol{x}_k^* = \boldsymbol{x}_{k-1}^* + \boldsymbol{P}_{k-1}^{rhp}\boldsymbol{u}_k^*, \tag{22}$$

which is similar to (17). A priority matrix

$$\Psi_{rhp} = \begin{bmatrix} \alpha_{11} & \cdots & \alpha_{1r} \\ \vdots & \ddots & \vdots \\ \alpha_{n_l 1} & \cdots & \alpha_{n_l r} \end{bmatrix} \in \mathbb{R}^{n_l \times r} \tag{23}$$

is also defined in RHP-HQP to indicate the priority between tasks. In this matrix, variables on the k-th row $\alpha_{kj} \in [0,1], j = 1, \cdots, r$ represents the extent of freedom task j has on the k-th priority level. When $\alpha_{kj} = 0$, task j has no freedom on this priority level, i.e., the priority of task j is lower than k; When $\alpha_{kj} = 1$, task j has a priority level no lower than k; When $0 < \alpha_{kj} < 1$, task j is in a transitional state on this priority level, and the larger α_{kj} is, the more important it is on this level. For the consistency of priority, we have $\forall m < n, \alpha_{mj} \leqslant \alpha_{nj}$. If the priority level of task j is k, then $\alpha_{mj} = 0, (m < k)$ and $\alpha_{mj} = 1, (m \geqslant k)$.

By selecting n_k tasks that satisfy $\alpha_{kj} \neq \alpha_{k-1,j}, j = 1, \cdots, r$ according to (23), the task matrices need to be solved on the k-th level $\boldsymbol{A}_k^{A,s}$ can be determined. Arrange the task matrices and objective vectors of these n_k tasks in descending order of α_{kj} into an augmented matrix and vector in the form of

$$\boldsymbol{A}_k^{A,s} = \begin{bmatrix} \boldsymbol{A}_{s_1} \\ \vdots \\ \boldsymbol{A}_{s_{n_k}} \end{bmatrix}, \boldsymbol{b}_k^{A,s} = \begin{bmatrix} \boldsymbol{b}_{s_1} \\ \vdots \\ \boldsymbol{b}_{s_{n_k}} \end{bmatrix}. \tag{24}$$

Construct a diagonal matrix with the variable relating to the selected tasks as

$$\boldsymbol{\Lambda}_k^s = \begin{bmatrix} \alpha_{k,s_1}\boldsymbol{I}_{d_{s_1}} & & \\ & \ddots & \\ & & \alpha_{k,s_{n_k}}\boldsymbol{I}_{d_{s_{n_k}}} \end{bmatrix}, \tag{25}$$

where $\boldsymbol{I}_{d_s} \in \mathbb{R}^{d_s \times d_s}$ is an identity matrix, d_s is the row number of $\boldsymbol{A}_s$.

To activate or delete a task, the objective vector in (21a) is given by

$$\hat{\boldsymbol{b}}_k^{A,s} = \boldsymbol{\Lambda}_k^s \boldsymbol{b}_k^{A,s} + (\boldsymbol{I} - \boldsymbol{\Lambda}_k^s)\, \boldsymbol{A}_k^{A,s} \boldsymbol{x}_{k-1}^* \tag{26}$$

with an approach similar to intermediate value [19,22].

After getting $\boldsymbol{A}_k^{A,s}$ and $\boldsymbol{\Lambda}_k^s$, the recursive hierarchical projection matrix $\boldsymbol{P}_k^{rhp}$ on the k-th level can be calculated in the following three steps:

Step 1: Calculate the task matrix after projection $\tilde{\boldsymbol{A}}_k^{A,s} = \boldsymbol{A}_k^{A,s}\boldsymbol{P}_{k-1}$;

Step 2: Perform Gaussian elimination to the row vector of $\tilde{\boldsymbol{A}}_k^{A,s}$ and get a set of

orthogonal bases $\boldsymbol{B}_k \in \mathbb{R}^{m_k \times n}$ of its row space, where m_k is the rank of $\tilde{\boldsymbol{A}}_k^{A,s}$. Construct an m_k-dimensional diagonal matrix $\boldsymbol{\Lambda}_k^{s,m_k}$ according to the dimension of each task matrix in $\tilde{\boldsymbol{A}}_k^{A,s}$;

Step 3: Calculate the recursive hierarchical projection matrix $\boldsymbol{P}_k^{rhp}$ on the k-th level

$$\boldsymbol{P}_0^{rhp} = \boldsymbol{I} \tag{27}$$

$$\boldsymbol{P}_k^{rhp} = \boldsymbol{P}_{k-1}^{rhp}(\boldsymbol{I}_n - \boldsymbol{B}_k^T \boldsymbol{\Lambda}_k^{s,m_k} \boldsymbol{B}_k). \tag{28}$$

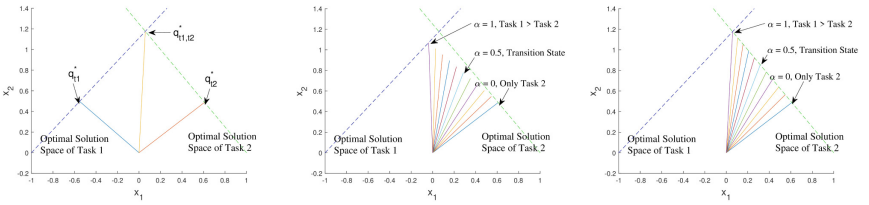
When all the variables in Ψ_{rhp} are 0 or 1, $\boldsymbol{\Lambda}_k^s$ is an identity matrix and therefore $\hat{\boldsymbol{b}}_k^{A,s} = \boldsymbol{b}_k^{A,s}$. $\boldsymbol{P}_k^{rhp}$ in this case, which projects the task into the nullspace of all the tasks prior to level k, has a similar meaning to the definition in [2,30].

3.3 Comparison Between Different Algorithms

First, let us consider the two fixed priority algorithms. HQP-original with form (10) and HQP-nullspace with form (16) solve the first level of task in the same way. Especially when there is only one level of task, HQP is exactly the same with WQP, which is defined in (8). Therefore WQP can be viewed as the special case of HQP with only one level of task. When the priority variables of RHP in (23) are set as 0 or 1, all the tasks at each priority level will be fully solved and the hierarchy will be maintained. In this case, the solution given by (21) is the same as that of HQP, hence HQP can be seen as a special case of RHP where all the tasks are in strict priority hierarchy. Considering numerical solution, although HQP and RHP are more computationally efficient than the original constraint form, under some cases, e.g. $\overline{\boldsymbol{A}}_k \boldsymbol{N}_{k-1} \simeq \boldsymbol{O}$ or $\boldsymbol{A}_k^{A,s} \boldsymbol{P}_{k-1}^{rhp} \simeq \boldsymbol{O}$ these two algorithms may become ill conditioned. As a result, regular terms may need to be introduced to avoid singularities, but notice that the optimal results may be affected. In general, algorithms from (8) to (16) then to (21) is a transition from particular form to a more general form.

Then, the two priority transition algorithms GHC and RHP will be discussed. For ease of description, tasks are defined as in priority transition if variables $\alpha_{ij} \in (0, 1)$ in (19) and (23), and are defined as in strict priority hierarchy if $\alpha_{ij} = 0, 1$. On one hand, the solutions given by GHC and RHP are not necessarily the same in strict priority hierarchy. The former will solve each task separately and then trim them in the order of priority, while the latter will solve each task in turn according to their priority order. On the other hand, GHC and RHP are not equivalent in priority transition. Although the two are similar in the idea of projection matrix transition, but the meaning of the two projection matrices are different: In GHC, it gradually cuts the optimal solution of the lower priority tasks off the optimal solution space of the higher priority task; While in RHP, priority transition relies on intermediate value transition for task objectives and transition in the corresponding feasible solution space during the transformation of the nullspace projection matrix.

Take two single objective tasks in the two dimensional space for example. Each task has an optimal solution $\boldsymbol{q}_{t1}^*, \boldsymbol{q}_{t2}^*$ and a one dimensional nullspace, and



(a) Optimal solution space (b) GHC priority transition (c) RHP priority transition

Fig. 4. Two single objective tasks in the two dimensional space with transition priorities.

both can be satisfied simultaneously with an optimal solution $\boldsymbol{q}^*_{t1,t2}$, as depicted in Fig. 4(a). The task priority transition is set as from solving only task 2 to solving both task 1 and task 2, with task 1 having a higher priority than task 2. A variable α can be defined to represent the process, and the priority matrix for GHC is defined as

$$\boldsymbol{\Psi}_{ghc} = \begin{bmatrix} 1-\alpha & 1-\alpha \\ \alpha & 0 \end{bmatrix}, \ \alpha : 0 \rightarrow 1 \tag{29}$$

while the priority matrix for RHP is

$$\boldsymbol{\Psi}_{rhp} = \begin{bmatrix} \alpha \& 0 \\ \alpha \& 1 \end{bmatrix}, \ \alpha : 0 \rightarrow 1. \tag{30}$$

When $\alpha = 0$, only task 2 needs to be solved and the solution from GHC and RHP are the same. When $\alpha = 1$, task 1 and task 2 must be finished simultaneously and task 2 has a lower priority level than task 1. It can be observed from Fig. 4(b) that the solution from GHC is located in the optimal solution space of task 1 and task 2 has no effect on task 1, thus the order of the priority is satisfied. However, this solution is not the optimal solution $\boldsymbol{q}^*_{t1,t2}$ for both tasks. From the perspective of multi-objective optimization, the solution given by GHC is not Pareto optimum under strict priority hierarchy. This is because tasks in GHC can be considered as solved separately and projection matrix can only ensure the priority between tasks. In RHP, as illustrated in Fig. 4(c), its solution is Pareto optimum under strict priority hierarchy. In priority transition state when $\alpha \in (0, 1)$, both task 1 and task 2 are affected in GHC and the solution is not optimal for either one. In RHP however, $\boldsymbol{\Psi}_{rhp}(2, 2) = 1$ so that task 2 is always fully solved, therefore, the priority transition results in a gradual transition to the optimal solution of task 1 while maintaining task 2 as the optimal solution.

4 Simulation Results

This section demonstrates the nature of different multi-task control algorithms with simulation on a robotic arm with 4 DoFs working in the vertical plane. As Fig. 5 illustrates, the position and orientation of the end-effector is defined as (x_e, y_e) and θ respectively, and the angular positions of the four actuated joints are defined as $\boldsymbol{q} = [q_1, q_2, q_3, q_4]^T$.

Additionally, all the algorithms introduced in Sect. 3 have been implemented in one toolkit called WBCKits, which provides convenient methods to add tasks, constraints and set task weights and priorities. After selecting specific algorithm and task setting, WBCKits can automatically construct QP(s) using the general linear algebra library Eigen [11] and get the optimal results with QP solver qpOASES [10]. WBCKits is freely available and the C++ source code is published on https://github.com/yueqing-li/WBCKits. A Python wrapper is also provided for exploiting further employment like combining with learning algorithm.

In Sect. 4.1 the main comparison is made between the control effects and computational efficiency of different algorithms in multi-task control under fixed priority. In Sect. 4.2, control effects of algorithms will be compared under transitional priority, and the preservation of priority hierarchy in different transition methods will also be compared.

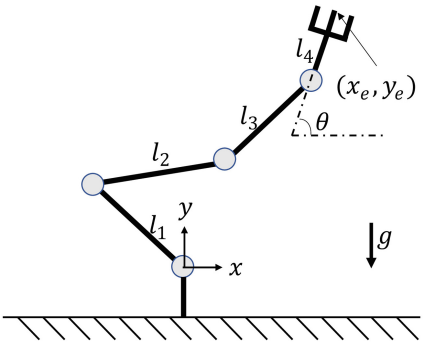


Fig. 5. Model of a planar robotic arm with 4 DoFs. $l_1 = l_2 = l_3 = 0.5m$, $l_4 = 0.2m$.

Table 1. Task PD gains and corresponding priority or weights.

Sub-tasks	$K_p(\mathrm{s}^{-2})$	$K_d(\mathrm{s}^{-1})$	Priority Level	WQP-1 Weights	WQP-2 Weights
Horizontal position x	100	30	1	1000	15
Vertical position y	100	30	2	100	10
Orientation θ	35	10	3	10	5
Angular position of joints	20	8	4	1	1

4.1 Tasks with Fixed Priority

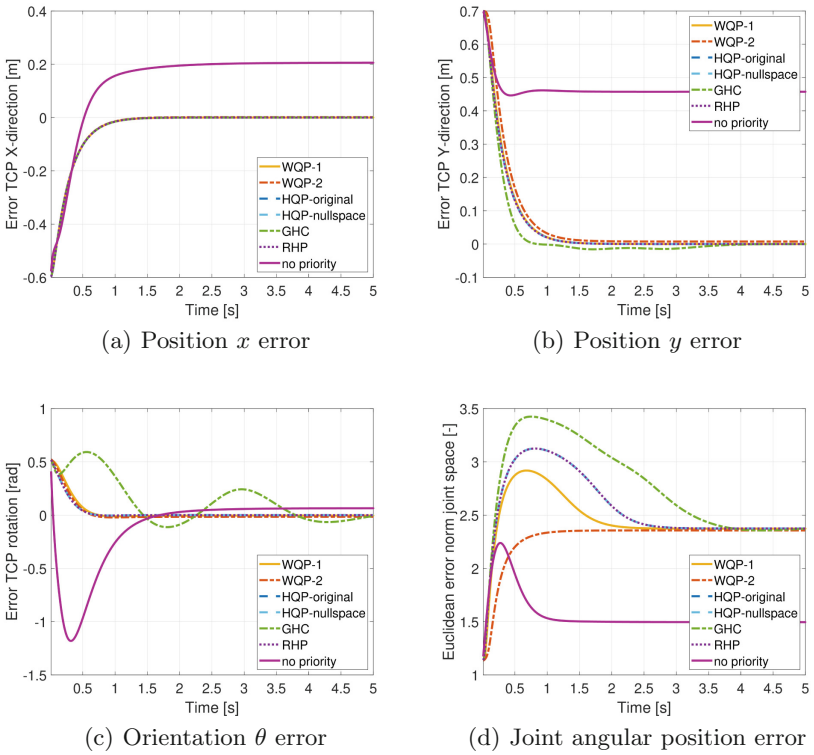
In this section, the tracking error of tasks with different priority and the computation time of different algorithms will be demonstrated. Simulation results are based on the 4-DoF planar robotic arm model with a maximum joint torque constraint. Four sub-tasks, including tracking the horizontal position x_e, vertical position y_e and the orientation θ of the tool center point (TCP) and the angular position of joints, are imposed.

The four tasks are controlled in task space using PD feedback to obtain task acceleration, and the PD gains and task priority for each task are shown in Table 1, where smaller number of priority level means higher priority. Figure 6 depicts the step response curves for each task with the same feedback factor for different algorithms. To compare the characteristics of WQP, two sets of weighting parameters are used, as shown in Table 1. The "no priority" curves in Fig. 6 are obtained without using the multi-task control algorithms. Torque is generated by calculating the inverse dynamics after solving each task separately and summing their joint acceleration results. Table 2 presents the static error of different algorithms for all tasks at the 10th second of the simulation. The acceleration loss in Fig. 7 depicts the difference between the expected task acceleration of PD feedback control and the actual task acceleration calculated from optimal joint acceleration.

As can be seen in Fig. 6, RHP performs almost identically to HQP with fixed task priority and both converge rapidly to zero static error state. GHC performs similarly to the other two hierarchical control algorithms (HQP, RHP) on the highest priority task (horizontal position x) but has a slower converging speed on tasks with lower priority. In WQP, tasks with larger weights converge slower because of the influence of tasks with smaller weights, but the smaller weighted tasks may converge faster than the hierarchical control algorithms accordingly.

When there exists a joint configuration $\boldsymbol{q}$ and $\dot{\boldsymbol{q}} = \mathbf{0}$ that enables multiple task objectives to be achieved simultaneously, these tasks are said to be feasible simultaneously. In this simulation, the first three tasks are feasible simultaneously, and zero static error of the first three tasks is achieved in every hierarchical control algorithm (HQP, GHC and RHP), as shown in Table 2. The fourth task, on the other hand, is not feasible simultaneously because it conflicts with the previous tasks, and therefore has a static error.

WQP, on the other hand, does not guarantee zero static error for tasks with larger weights. It can be seen from Fig. 7 that when the weights are rela-



(a) Position x error (b) Position y error

(c) Orientation θ error (d) Joint angular position error

Fig. 6. Control effect of the 4-DoF planar robotic arm with different algorithms.

Table 2. Static error of the 4-DoF planar robotic arm with different algorithms for all tasks at the 10th second of the simulation.

Algorithm	Horizontal error Δx(m)	Vertical error Δy(m)	Orientation error$\Delta\theta$(rad)	Norm of joint angular position error $\|\Delta \boldsymbol{q}\|_2$
WQP-1	0.000	0.000	–0.004	2.373
WQP-2	0.001	0.008	–0.016	2.357
HQP-original	0.000	0.000	0.000	2.375
HQP-nullspace	0.000	0.000	0.000	2.375
GHC	0.000	0.000	0.000	2.375
RHP	0.000	0.000	0.000	2.375

tively small, an obvious static error exists between the optimal acceleration and expected acceleration, and increasing the weights can reduce the error. However, the impact of different dimensions and other relative tasks makes it difficult to determine task weight settings in a quantitative way to guarantee errors of more important tasks.

Some comments should be made on the convergence of GHC. To deal with multi-task problem, GHC will directly cut off the components of lower priority tasks that have an impact on higher priority tasks. This is possible to result in the situation where the optimization results of lower priority tasks do not keep

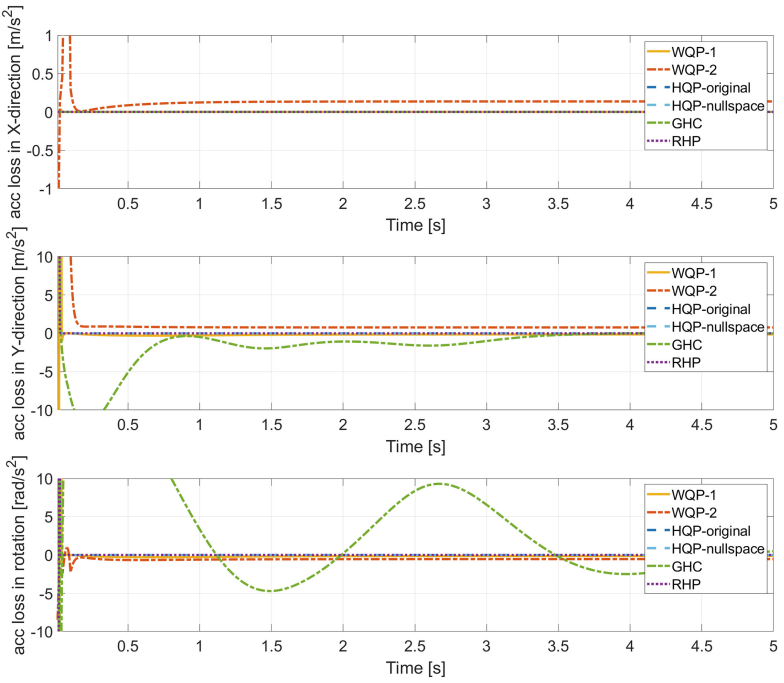


Fig. 7. Acceleration loss in the space of each task of optimal solutions from different algorithms.

up with the expected value even if the problem itself satisfies the multi-objective problem, such as the non Pareto minimum mentioned in Fig. 4(b) when $\alpha = 1$. From Fig. 6(b) we can see that GHC converges faster in position y than any other algorithm, but this is not because GHC has a better convergence when dealing with lower priority tasks. After cutting off the influence on the x direction from the y direction, the acceleration mapped to the y direction from the optimal solution at the current moment is smaller than the expected acceleration of PD feedback, as can be seen in Fig. 7. As a result, GHC converges faster in the early stage. The fluctuation in position y error after about 1.2 s is also caused by this factor. This feature of GHC will greatly affect its convergence. Take the orientation θ in 6(c) for example, the optimized acceleration is the exact opposite of the expected acceleration, thus at 0.2–0.5 s the task is feasible but the error increases. In order not to affect the higher priority tasks of position x and y, the orientation task does not reach static stability until about 8 s later (not shown in the figure).

Figure 8 depicts the histogram of time consumed by different algorithms solving the WBC problem in each control cycle during the whole simulation process, counted by interval. It can be seen that the calculation time of WQP is much lower than other algorithms because its structure is simple and it only needs

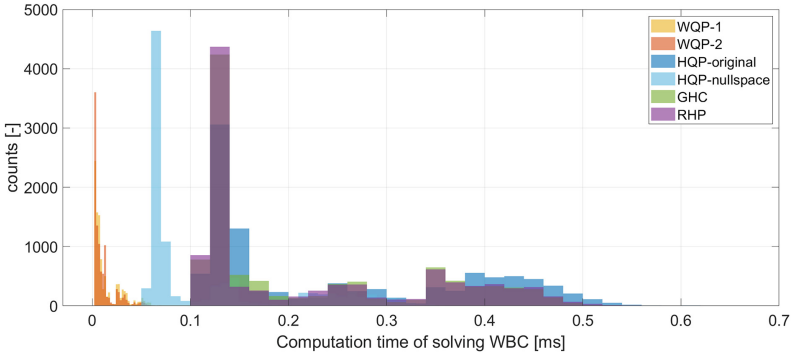


Fig. 8. Counts of computation time of 4-DOF planar robotic arm with different algorithms.

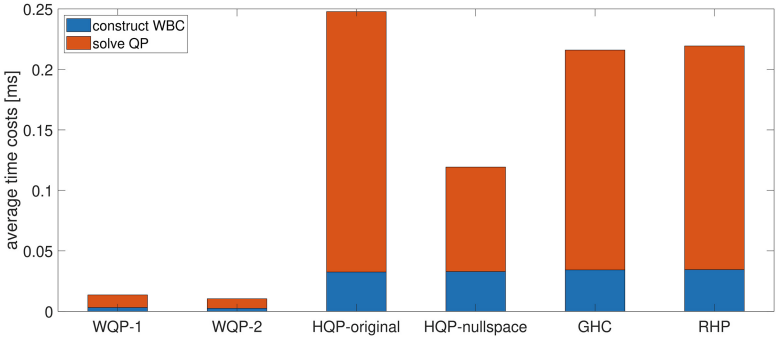


Fig. 9. Average computation time of 4-DOF planar robotic arm with different algorithms.

to solve one QP problem. HQP-nullspace is next, while the time consumed by HQP-original, GHC and RHP are the most and are almost the same.

Figure 9 is a bar chart of the average time consumed by different algorithms in two parts: constructing WBC and solving QP. It can be seen that the average time spent on solving the QP problem by HQP-nullspace is less than that of HQP-original, which is in accordance with the previous theoretical analysis. The time consumption of HQP-nullspace, GHC and RHP in construction WBC should be higher than HQP-original theoretically because of additional matrix decomposition for calculating nullspace. While here we find that their time consumption is about the same. Because in this problem, the 4-DOF robotic model is relatively simple and the complexity of matrix decomposition is not high.

4.2 Tasks with Transitional Priority

This section uses the dynamic addition and deletion of tasks and the dynamic change of priority to reflect the characteristics of the priority transition algorithms, and the impact of introducing the transition process on priority changes.

On the basis of tasks in Sect. 4.1, a limit on the angular position of the first joint is imposed. Let the lower and upper bounds of the first joint position q_1 be $[\underline{q}_1, \overline{q}_1]$ and set the buffer width as $\beta_1 \in \mathbb{R}^+$, then this task can be defined as

$$\min_{\boldsymbol{q}} \|\boldsymbol{J}_1 \boldsymbol{q} - \ddot{x}_{j1}\|, \tag{31}$$

where $\boldsymbol{J}_1 \in \mathbb{R}^{1\times 4}$ is a zero matrix with $\boldsymbol{J}_1(1,1) = 1$. $\ddot{x}_{j1}$ is defined as

$$\ddot{x}_{j1} = \begin{cases} k_p((\underline{q}_1 + \beta_1) - q_1) - k_v \dot{q}_1, \text{ if } q_1 < \underline{q}_1 + \beta_1 \\ k_p((\overline{q}_1 - \beta_1) - q_1) - k_v \dot{q}_1, \text{ if } q_1 > \overline{q}_1 - \beta_1 \\ 0, \qquad\qquad \text{otherwise} \end{cases}. \tag{32}$$

When $q_1 \in [\underline{q}_1 + \beta_1, \overline{q}_1 - \beta_1]$, this task is deleted and the priority set of other tasks are the same with Sect. 4.1. For WQP, the corresponding weight will be set as 0, while the priority matrix of GHC and RHP are set as

$$\boldsymbol{\Psi}_{ghc,0} = \begin{bmatrix} 0\,0\,0\,0\,0 \\ 1\,0\,0\,0\,0 \\ 1\,1\,0\,0\,0 \\ 1\,1\,1\,0\,0 \\ 1\,1\,1\,1\,1 \end{bmatrix}, \quad \boldsymbol{\Psi}_{rhp,0} = \begin{bmatrix} 0\,0\,0\,0\,0 \\ 1\,0\,0\,0\,0 \\ 1\,1\,0\,0\,0 \\ 1\,1\,1\,0\,0 \\ 1\,1\,1\,1\,0 \end{bmatrix}, \tag{33}$$

respectively. When $q_1 \leqslant \underline{q}_1$ or $q_1 \geqslant \overline{q}_1$, this task is activated and has the highest priority to other tasks. For WQP, the corresponding weight will be set to 10 times the previous maximum task weight as $10w_{\max}$, while the priority matrix of GHC and RHP are set as

$$\boldsymbol{\Psi}_{ghc,1} = \begin{bmatrix} 0\,0\,0\,0\,1 \\ 1\,0\,0\,0\,1 \\ 1\,1\,0\,0\,1 \\ 1\,1\,1\,0\,1 \\ 0\,0\,0\,0\,0 \end{bmatrix}, \quad \boldsymbol{\Psi}_{rhp,1} = \begin{bmatrix} 0\,0\,0\,0\,1 \\ 1\,0\,0\,0\,1 \\ 1\,1\,0\,0\,1 \\ 1\,1\,1\,0\,1 \\ 1\,1\,1\,1\,1 \end{bmatrix}, \tag{34}$$

respectively. An activation parameter can be defined by the angular position of joint q_1 as

$$\alpha_{j1} = \begin{cases} 0, & \text{if } \underline{q}_1 + \beta_1 \leqslant q_1 < \overline{q}_1 - \beta_1 \\ 1, & \text{if } q_1 < \underline{q}_1 \text{ or } q_1 > \overline{q}_1 \\ (\underline{q}_1 + \beta_1 - q_1)/\beta_1, & \text{if } \underline{q}_1 \leqslant q_1 < \underline{q}_1 + \beta_1 \\ (q_1 - \overline{q}_1 + \beta_1)/\beta_1, & \text{if } \overline{q}_1 - \beta_1 \leqslant q_1 < \overline{q}_1 \end{cases}. \tag{35}$$

With this parameter, the priority of joint limit task can change continuously. In WQP, the weight for this task is set to be $10\alpha_{j1}w_{\max}$, while in GHC and RHP the priority matrix are set as

$$\boldsymbol{\Psi}_{ghc} = (1 - \alpha_{j1})\boldsymbol{\Psi}_{ghc,0} + \alpha_{j1}\boldsymbol{\Psi}_{ghc,1} \tag{36}$$

$$\boldsymbol{\Psi}_{rhp} = (1 - \alpha_{j1})\boldsymbol{\Psi}_{rhp,0} + \alpha_{j1}\boldsymbol{\Psi}_{rhp,1}, \tag{37}$$

respectively.

In this simulation, the setting of other tasks are again shown in Table 1. The lower and upper bounds of the first joint position q_1 is set as $[-15°, 15°]$ and the buffer width is set as $\beta_1 = 5°$. In order to reflect the influence of joint position limit, the reference position of the TCP is planned so that when it is close to the final target position, the first joint will enter the buffer zone and the remaining joints cannot complete the planar position task, ending up approaching a singular pose. Two transition processes are designed for RHP, and the difference is that the same priority is expressed with different $\boldsymbol{\Psi}_{rhp,0}$. Label RHP-1 corresponds to (34) and label RHP-2 corresponds to (38). Figure 10 depicts the curves of activation parameter α_{j1} and the angular position of the first joint over time, and Fig. 11 illustrates the control effects of the other tasks under the joint limit with different algorithms.

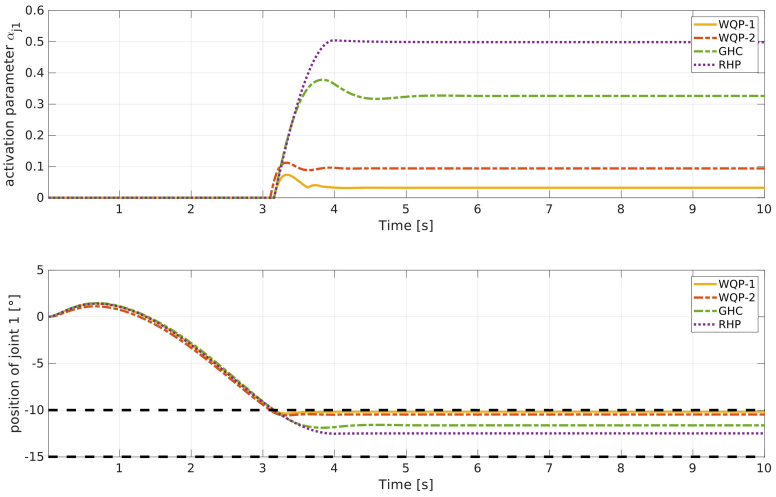
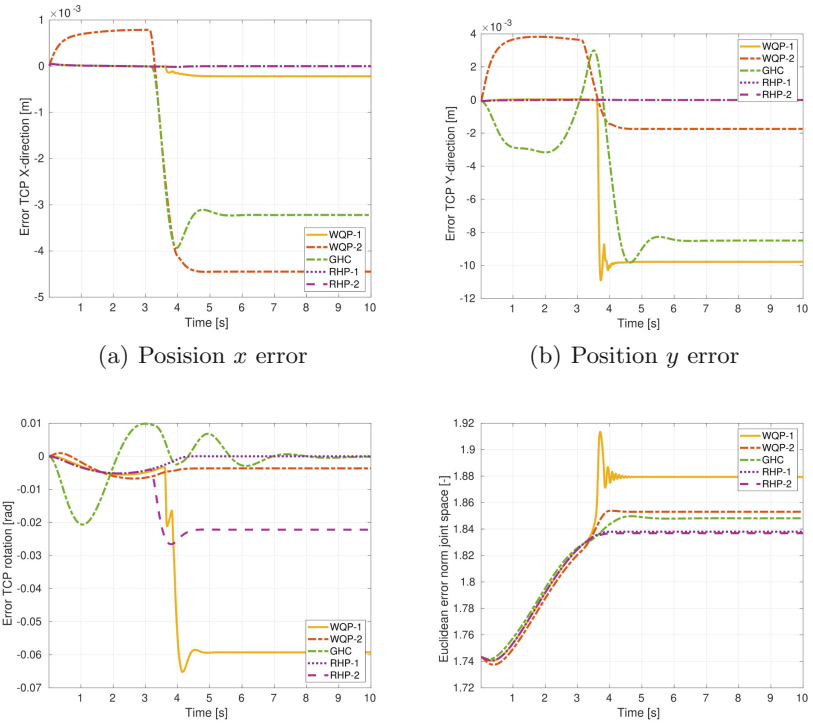


Fig. 10. Activation parameter and angular position of joint 1 of the 4-DOF planar robotic arm with different algorithms.

It can be seen that both GHC and RHP can use the transition state to achieve an effect similar to the partial completion of tasks in the WQP algorithm. When the first joint reaches the buffer zone as the task progresses, the joint position limit task is gradually activated to avoid the joint position exceeding the lower bound. Combined with the results in Sect. 4.1 we can see that different weight ratios have a huge impact on the task results in WQP, and the difficulty of quantitatively analyzing the weights is once again reflected. The weight itself can only reflect a relative relationship between tasks, and when tasks that are coped with each other increases, the weight change of a single task may have a great impact on the overall solution result, and this impact is difficult to describe by model. In contrast, GHC and RHP in transition are easier to analyze. For GHC, after it reaches a steady state, the activation parameter α_{j1} is about 0.326,

meaning that the optimization results for subsequent tasks will be clipped by about one-third on the first joint. This results in an error between the optimal solution of GHC and the expected acceleration of the PD feedback controller on the subsequent tasks, and will lead to the error shown in Fig. 11 after entering the transition process. RHP will complete the lower priority tasks as much as possible after partially completing the joint limit (or choosing a joint space configuration that does not affect the joint limit task), so it can be seen that the lower priority tasks can still be tracked well.



(a) Posision x error (b) Position y error

(c) Orientation θ error (d) Joint angular position error

Fig. 11. Control effect of the 4-DOF planar robotic arm under the joint limit with different algorithms.

It should be noted that the setting of the RHP priority matrix will affect the completion of tasks during the transition process. If the priority matrix at the beginning is set as

$$\boldsymbol{\Psi}_{rhp,0} = \begin{bmatrix} 1\,0\,0\,0\,0 \\ 1\,1\,0\,0\,0 \\ 1\,1\,1\,0\,0 \\ 1\,1\,1\,1\,0 \\ 1\,1\,1\,1\,0 \end{bmatrix}, \tag{38}$$

Table 3. Performance of different QP algorithms for redundant robot control.

	WQP	HQP-original	HQP-nullspace	GHC	RHP
Strict hierarchy	no	yes	yes	yes	yes
Pareto optimum	yes	yes	yes	no	yes
Numerical singularity	no	no	yes	yes	yes
Continuous priority transition	yes	no	no	yes	yes
Hierarchy in transition	no	–	–	partial	partial
Computation efficiency	very fast	slow	fast	slow	slow

although it also means that there is no joint limit task and the priority order of other tasks is the same as Sect. 4.1, but in the transition process, it also has the meaning of gradually transitioning the priority of other tasks to a lower level in addition to gradually activating the joint limit task as the highest priority. This will result in conflict on the fourth level of the orientation task and the joint angular position task, i.e., the available DoFs for the orientation task become less. So as Fig. 11(c) illustrates, tracking error occurs on RHP-2 in stability state, but RHP-2 has a slightly smaller angular position error than RHP-1 as can be seen in Fig. 11(d). Therefore for RHP, the design of the transition process is also a factor that needs special consideration.

5 Conclusion

In this paper, we hope to present an overview on related algorithms for WBC of redundant robots using QP, and meanwhile offer inspiration on how to choose the appropriate algorithm according to task setting. Theoretical analysis and simulation results demonstrate the advantages and disadvantages of multi-objective optimization algorithms in different aspects. Table 3 summarizes the performance of the algorithms discussed in this paper. After the task control problem is abstracted as general quadratic programming, the conflict between tasks in the multi-task control problem is explained from the perspective of multi-objective optimization. The essential idea in dealing with these conflicts is to assign different priority to each task so as to obtain the optimal solution with trade-off.

Our analysis first compare the effect of WQP and HQP on task hierarchy with fixed priority. The former can only reduce the impact of low priority tasks on high priority tasks to a certain extent through the weighting strategy, but this impact can never be completely eliminated; while the latter strictly guarantees that low priority tasks will not affect high priority tasks during the control process, and the optimization result is Pareto optimum. The original HQP needs to solve multiple optimization problems comparable to WQP, so that its computation efficiency is much lower than WQP. By parameterizing the nullspace basis matrix, the optimization variable dimension of subsequent QP problems are reduced and the computation efficiency can be improved. However, the introduction of the nullspace basis matrix brings instability to the numerical solution

of the optimization problem, which needs to be avoided by adding a regular term.

Then the control effect of GHC and RHP are discussed under transitional priority. The optimal solution of GHC is usually not the Pareto optimum of multi-objective optimization problem and the convergence performance of GHC is not as good as RHP. Both of them can realize the continuous change of priority and the effect that some tasks are affected during the transition process while other tasks are still strictly hierarchical. Although WQP can also achieve similar transition effects, the defect that its weight setting is not quantifiable will be more obvious and the priority of other tasks is not strictly hierarchical. It should be noted that there may be many ways to design the RHP priority matrix, which will eventually affect the priority of other tasks during the transition process, while the priority matrix of GHC is relatively unique.

Finally, the above algorithms are implemented in a toolkit, which can easily realize various multi-task control algorithms with the help of a unified interface. With numerous simulations, the characteristics of each algorithm are compared and summarized.

Despite the progress in algorithms, there are still drawbacks such as lack of considering the time dimension and time consuming of inequality tasks transition. One consideration of this issue is expanding the algorithm to combining model prediction or probability. Besides, the trade-off between algorithm effects and computational efficiency should also be considered. Therefore, another consideration is the efficiency of the algorithm. To validate this, benchmark data of the WBC problem for the different algorithms, which is scarce and elusive, could be explored.

A last issue of robot WBC is how to assign appropriate priority to tasks, which relies heavily on the experience of the expert. And in the transition priority algorithm, the appropriate priority should vary with the environment. In reinforcement learning (RL), an optimal policy can be learned to maximize the user-defined reward from the environment. Therefore, a possible avenue to solve this issue is combining RL with robot WBC to train an agent to learn the policy of adjusting the task priorities.

Acknowlegement. This work was supported by STI 2030-Major Projects 2021ZD0201402.

References

1. Abe, Y., da Silva, M., Popović, J.: Multiobjective control with frictional contacts. In: Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, Eurographics Association, Goslar, DEU, SCA '07, pp. 249–258 (2007)
2. Baerlocher, P., Boulic, R.: Task-priority formulations for the kinematic control of highly redundant articulated structures. In: Proceedings. 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems. Innovations in Theory, Practice and Applications (Cat. No.98CH36190), vol. 1, pp. 323–329 (1998). https://doi.org/10.1109/IROS.1998.724639

3. Bellicoso, D., Gehring, C., Hwangbo, J., Fankhauser, P., Hutter, M.: Perception-less terrain adaptation through whole body control and hierarchical optimization. In: 2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids), pp. 558–564 (2016)
4. De Lasa, M., Mordatch, I., Hertzmann, A.: Feature-based locomotion controllers. ACM Trans. Graph. (TOG) **29**(4), 1–10 (2010)
5. Deo, A.S., Walker, I.D.: Overview of damped least-squares methods for inverse kinematics of robot manipulators. J. Intell. Robot. Syst. **14**(1), 43–68 (1995)
6. Dietrich, A., Albu-Schäffer, A., Hirzinger, G.: On continuous null space projections for torque-based, hierarchical, multi-objective manipulation. In: 2012 IEEE International Conference on Robotics and Automation, pp. 2978–2985 (2012). https://doi.org/10.1109/ICRA.2012.6224571
7. Dietrich, A., Ott, C., Albu-Schäffer, A.: An overview of null space projections for redundant, torque-controlled robots. Int. J. Robot. Res. **34**(11), 1385–1400 (2015)
8. Du, W., Fnadi, M., Benamar, F.: Whole-body motion tracking for a quadruped-on-wheel robot via a compact-form controller with improved prioritized optimization. IEEE Robot. Autom. Lett. **5**, 516–523 (2020)
9. Escande, A., Mansard, N., Wieber, P.B.: Hierarchical quadratic programming: Fast online humanoid-robot motion generation. Int. J. Robot. Res. **33**(7), 1006–1028 (2014)
10. Ferreau, H., Kirches, C., Potschka, A., Bock, H., Diehl, M.: qpOASES: a parametric active-set algorithm for quadratic programming. Math. Program. Comput. **6**(4), 327–363 (2014)
11. Guennebaud, G., Jacob, B., et al.: Eigen v3 (2010) . https://eigen.tuxfamily.org
12. Han, G., Wang, J., Ju, X., Zhao, M.: Recursive hierarchical projection for whole-body control with task priority transition. In: 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 11312–11319 (2021)
13. Herzog, A., Rotella, N., Mason, S., Grimminger, F., Schaal, S., Righetti, L.: Momentum control with hierarchical inverse dynamics on a torque-controlled humanoid. Auton. Robot. **40**(3), 473–491 (2016)
14. Hutter, M., Sommer, H., Gehring, C., Hoepflinger, M., Bloesch, M., Siegwart, R.: Quadrupedal locomotion using hierarchical operational space control. Int. J. Robot. Res. **33**(8), 1047–1062 (2014)
15. Kanoun, O., Lamiraux, F., Wieber, P.B., Kanehiro, F., Yoshida, E., Laumond, J.P.: Prioritizing linear equality and inequality systems: application to local motion planning for redundant robots. In: 2009 IEEE International Conference on Robotics and Automation, pp. 2939–2944 (2009) . https://doi.org/10.1109/ROBOT.2009.5152293
16. Khatib, O.: A unified approach for motion and force control of robot manipulators: the operational space formulation. IEEE J Robot. Autom. **3**, 43–53 (1987)
17. Kim, D., Carlo, J.D., Katz, B., Bledt, G., Kim, S.: Highly dynamic quadruped locomotion via whole-body impulse control and model predictive control. ArXiv abs/1909.06586 (2019)
18. Kim, D., Jorgensen, S.J., Lee, J., Ahn, J., Luo, J., Sentis, L.: Dynamic locomotion for passive-ankle biped robots and humanoids using whole-body locomotion control. Int. J. Robot. Res. **39**, 936–956 (2019)
19. Kim, S., Jang, K., Park, S., Lee, Y., Lee, S.Y., Park, J.: Continuous task transition approach for robot controller based on hierarchical quadratic programming. IEEE Robot. Autom. Lett. **4**(2), 1603–1610 (2019). https://doi.org/10.1109/LRA.2019.2896769

20. Kim, S., Jang, K., Park, S., Lee, Y., Lee, S.Y., Park, J.: Whole-body control of non-holonomic mobile manipulator based on hierarchical quadratic programming and continuous task transition. In: 2019 IEEE 4th International Conference on Advanced Robotics and Mechatronics (ICARM), pp. 414–419 (2019)
21. Klemm, V., Morra, A., Gulich, L., Mannhart, D., Rohr, D., Kamel, M., de Viragh, Y., Siegwart, R.Y.: LQR-assisted whole-body control of a wheeled bipedal robot with kinematic loops. IEEE Robot. Autom. Lett. **5**, 3745–3752 (2020)
22. Lee, J., Mansard, N., Park, J.: Intermediate desired value approach for task transition of robots in kinematic control. IEEE Trans. Robot. **28**(6), 1260–1277 (2012). https://doi.org/10.1109/TRO.2012.2210293
23. Liu, M., Micaelli, A., Evrard, P., Escande, A., Andriot, C.: Interactive dynamics and balance of a virtual character during manipulation tasks. In: 2011 IEEE International Conference on Robotics and Automation, pp. 1676–1682. IEEE (2011)
24. Liu, M., Hak, S., Padois, V.: Generalized projector for task priority transitions during hierarchical control. In: 2015 IEEE International Conference on Robotics and Automation (ICRA), pp. 768–773 (2015). https://doi.org/10.1109/ICRA.2015.7139265
25. Liu, M., Tan, Y., Padois, V.: Generalized hierarchical control. Autonom. Robot. **40**(1), 17–31 (2016)
26. Nakamura, Y., Hanafusa, H., Yoshikawa, T.: Task-priority based redundancy control of robot manipulators. Int. J. Robot. Res. **6**, 3–15 (1987)
27. Saab, L., Ramos, O.E., Keith, F., Mansard, N., Souéres, P., Fourquet, J.Y.: Dynamic whole-body motion generation under rigid contacts and other unilateral constraints. IEEE Trans. Robot. **29**, 346–362 (2013)
28. Salini, J., Padois, V., Bidaud, P.: Synthesis of complex humanoid whole-body behavior: a focus on sequencing and tasks transitions. In: 2011 IEEE International Conference on Robotics and Automation, pp. 1283–1290 (2011). https://doi.org/10.1109/ICRA.2011.5980202
29. Sentis, L., Khatib, O.: A whole-body control framework for humanoids operating in human environments. In: Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006 ICRA 2006, pp. 2641–2648 (2006)
30. Siciliano, B., Slotine, J.J.: A general framework for managing multiple tasks in highly redundant robotic systems. In: Fifth International Conference on Advanced Robotics 'Robots in Unstructured Environments, vol. 2, pp 1211–1216 (1991). https://doi.org/10.1109/ICAR.1991.240390
31. Siciliano, B., Slotine, J.J.E.: A general framework for managing multiple tasks in highly redundant robotic systems. In: Fifth International Conference on Advanced Robotics 'Robots in Unstructured Environments, vol. 2, pp. 1211–1216 (1991)