MPTC – Modular Passive Tracking Controller for stack of tasks based control frameworks

I think dark mode yields a better visulization (change to dark mode in the setting)

I/ General robot dynamic and task space equations

General robot dynamic equation can be written as:

$$\begin{aligned} M(q)\ddot{q} + C(q,\dot{q})\dot{q} + \tau_g(q) &= \tau \\ &= S(\tau_j + \tau_{int}) + L_{all}^T w_{all} \end{aligned}$$

- S is a selection matrix, having a form of $S = [0_{n_{act}\times 6} \quad I_{n_{act}\times n_{act}}]^T$
- $\tau_j \in \mathbb{R}^{n_{act}}$ is the actuated joint torque vector
- w_{all} is the vector that collects all the external wrench applies to the robot's links
- $L_{all} = [L_1^T \quad \dots \quad L_{n_L}^T]^T$ collects all the Jacobian of the links that the corresponding wrenches above apply to.

Denote the task k as x_k, the velocity level is $\dot{x}_k = J_k\dot{q}$. Then the acceleration level:

$$\begin{aligned} \ddot{x}_k = \dot{J}_k\dot{q} + J_k\ddot{q} = \dot{J}_k\dot{q} + J_kM^{-1}(\tau - \tau_g - C\dot{q}) &= (\dot{J}_k - J_kM^{-1}C)\dot{q} + J_kM^{-1}(\tau - \tau_g) \\ &= -Q_k\dot{q} + J_kM^{-1}(\tau - \tau_g) \end{aligned}$$

Then the task velocity error is:

$$\dot{\tilde{x}}_k = \dot{x}_{k,ref} - \dot{x}_k = \dot{x}_{k,ref} - J_k\dot{q}$$

Task acceleration error is:

$$\ddot{\tilde{x}}_k = \ddot{x}_{k,ref} - \ddot{x}_k = \ddot{x}_{k,ref} - J_kM^{-1}(\tau - \tau_g) + Q_k\dot{q}$$

Classically, the definition of dynamic terms in the task space are: (square matrix with dimension equals to the dimension of the task k)

- Task space inertia of the task k: $M_k = (J_kM^{-1}J_k^T)^{-1}$
- Coriolis and Centrifugal matrix $C_k = M_kQ_kT_k^T$,
  with
  $T_k = M_kJ_kM^{-1}$ is the weighted pseudo inverse of the Jacobian J_k^T

II/ Modular Passive Tracking Controller (main part)

II.1/ Formulation for one task k:

The idea is picking the Lyapunov function based on energy components, then find the desired task force $f_{k,des}$ so that the dynamic of the task is passive w.r.t the task velocity error as the output and the implement task force error as the input.
I will copy the detail computation from the paper here:

$$V_k = \underbrace{\frac{1}{2}\,\dot{\tilde{\boldsymbol{x}}}_k^T\,\boldsymbol{M}_k\,\dot{\tilde{\boldsymbol{x}}}_k}_{E_{kin,k}} + \underbrace{\frac{1}{2}\,\tilde{\boldsymbol{x}}_k^T\,\boldsymbol{K}_k\,\tilde{\boldsymbol{x}}_k}_{E_{pot,k}} \quad , \tag{13}$$

where the positive definite, symmetric matrix $\boldsymbol{K}_k$ denotes the task stiffness. This Lyapunov function is positive definite in the task position error $\tilde{\boldsymbol{x}}_k$ and the task velocity error $\dot{\tilde{\boldsymbol{x}}}_k$.

Now we differentiate (13) and insert (8), which yields:

$$\begin{aligned} \dot{V}_k &= \dot{\tilde{\boldsymbol{x}}}_k^T \left(\boldsymbol{M}_k\,\ddot{\tilde{\boldsymbol{x}}}_k + \frac{\dot{\boldsymbol{M}}_k}{2}\,\dot{\tilde{\boldsymbol{x}}}_k + \boldsymbol{K}_k\,\tilde{\boldsymbol{x}}_k \right) \\ &= \dot{\tilde{\boldsymbol{x}}}_k^T \Big(\underbrace{\boldsymbol{M}_k\,\boldsymbol{J}_k\,\boldsymbol{M}^{-1}}_{\boldsymbol{T}_k} (\boldsymbol{\tau}_g - \boldsymbol{\tau}) + \boldsymbol{M}_k\,\boldsymbol{Q}_k\,\dot{\boldsymbol{q}} \\ &\qquad + \boldsymbol{M}_k\,\ddot{\boldsymbol{x}}_{k,ref} + \boldsymbol{C}_k\,\dot{\tilde{\boldsymbol{x}}}_k + \boldsymbol{K}_k\,\tilde{\boldsymbol{x}}_k \Big) \quad . \end{aligned} \tag{14}$$

Here, we made use of the equality

$$\frac{\dot{\boldsymbol{M}}_k}{2} = \frac{\boldsymbol{C}_k^T + \boldsymbol{C}_k}{2} = \underbrace{\frac{\boldsymbol{C}_k^T - \boldsymbol{C}_k}{2}}_{\text{skew-symmetric}} + \boldsymbol{C}_k \quad , \tag{15}$$

The desired task force defined as $f_{k,des} = T_k \tau_{k,des}$ is chosen to cancel out all the the terms in the bracket and make the Lyapunov function $\dot{V}_k \prec 0$. That leads to:

$$f_{k,des} = T_k \tau_g + M_k Q_k \dot{q} + M_k \ddot{x}_{k,ref} + (C_k + D_k)\dot{\tilde{x}}_k + K_k \tilde{x}_k$$

However, in the real implementation we may only achieve the actual task force f_k instead of $f_{k,des}$. f_k can be written from f_{k,des} and the task force error as:

$$f_k = T_k \tau = f_{k,des} - (f_{k,des} - f_k) = f_{k,des} - \tilde{f}_k$$

Then the derivative of the Lyapunov function is written as:

$$\dot{V}_k = -\dot{\tilde{x}}_k^T D_k \dot{\tilde{x}}_k + \dot{\tilde{x}}_k^T \tilde{f}_k = \dot{V}_{k,des} + \dot{\tilde{V}}_k < \dot{\tilde{x}}_k^T \tilde{f}_k$$

Or equivalently:

Note that the controlled system (at task level) is passive with respect to input $\tilde{\boldsymbol{f}}_k$, output $\dot{\tilde{\boldsymbol{x}}}_k$ and the storage function V_k from (13). While the desired Lyapunov rate $\dot{V}_{k,des}$ is purely dissipative for a positive definite damping matrix $\boldsymbol{D}_k$, the term $\dot{\tilde{V}}_k$ may be non-zero, depending on factors including unknown perturbations, under-actuation and other actuation limits, task inconsistencies and prioritization. While the desired Lyapunov

With this influence of the actual task force above, from the above task acceleration level formula, we can “obtain a task dynamic of the form”:

$$M_k \ddot{\tilde{x}}_k + (C_k + D_k)\dot{\tilde{x}}_k + K_k x_k = \tilde{f}_k$$

In which $D_k, K_k \succ 0$ are the tuning parameters. Noted here original terms belongs to the internal dynamics still appear which are M_k and C_k

II.2/ Develop for stack of tasks:

Stacking all desired task forces $\boldsymbol{f}_{k,des}$ from (17) for $k \in \{1,..,n_T\}$ yields

$$\boldsymbol{f}_{des} = \begin{bmatrix} \boldsymbol{f}_{1,des} \\ \vdots \\ \boldsymbol{f}_{n_T,des} \end{bmatrix} . \quad (21)$$

Similarly, we stack (16) for $k \in \{1,..,n_T\}$ to obtain

$$\underbrace{\begin{bmatrix} \boldsymbol{f}_1 \\ \vdots \\ \boldsymbol{f}_{n_T} \end{bmatrix}}_{\boldsymbol{f}} = \underbrace{\begin{bmatrix} \boldsymbol{T}_1 \\ \vdots \\ \boldsymbol{T}_{n_T} \end{bmatrix}}_{\boldsymbol{T}} \boldsymbol{\tau} . \quad (22)$$

All elements of

This is the mapping from the actual generalized forces $\boldsymbol{\tau}$ to the *stack of actual task forces* $\boldsymbol{f}$ via the corresponding collected

Introduce a new term so-called *commanded task forces,* resulted from a **optimization controller** that minimize the error between the desired task force and the commanded task force.

The formula to obtain the desired task force above can be treated as one of a equality constraint in the optimazation problem setup.

$$\boldsymbol{f}_{cmd} = \begin{bmatrix} \boldsymbol{f}_{1,cmd} \\ \vdots \\ \boldsymbol{f}_{n_T,cmd} \end{bmatrix} = \boldsymbol{T}\ \boldsymbol{\tau}_{cmd} = \underbrace{\boldsymbol{T}\,\boldsymbol{U}}_{\boldsymbol{T}_u}\ \boldsymbol{u}_{cmd}$$

sponding stack of task force command errors:

$$\boxed{\tilde{\boldsymbol{f}}_{cmd} = \boldsymbol{f}_{des} - \boldsymbol{f}_{cmd} = \boldsymbol{f}_{des} - \boldsymbol{T}\,\boldsymbol{\tau}_{cmd} = \boldsymbol{f}_{des} - \boldsymbol{T}_u\,\boldsymbol{u}_{cmd} \quad . \quad (25)}$$

Actually, u_{cmd} is the optimization variables while U is so-called actuation mapping matrix that maps the optimization variables to the generalize forces influenced to the dynamic.

SOME DEVELOPING OF OVERALL LYAPUNOV FUNCTION WITH A SET OF SCALAR WEIGHT FOR EACH TASK, WE CAN NEGLECT IT AT THIS MOMENT.

From this definition, the **cost function** is defined as:

C. Overall cost function

Based on $\tilde{\boldsymbol{f}}_{cmd}$ from (25), we formulate an overall cost function

$$\begin{aligned} \boxed{G = \frac{1}{2}\,\tilde{\boldsymbol{f}}_{cmd}^T\ \boldsymbol{W}\ \tilde{\boldsymbol{f}}_{cmd}} \quad (34) \\ = \frac{1}{2}\boldsymbol{\tau}_{cmd}^T \boldsymbol{T}^T \boldsymbol{W} \boldsymbol{T} \boldsymbol{\tau}_{cmd} - \boldsymbol{f}_{des}^T \boldsymbol{W} \boldsymbol{T} \boldsymbol{\tau}_{cmd} + \frac{1}{2}\boldsymbol{f}_{des}^T \boldsymbol{W}\,\boldsymbol{f}_{des} \\ \boxed{= \frac{1}{2}\boldsymbol{u}_{cmd}^T \boldsymbol{T}_u^T \boldsymbol{W} \boldsymbol{T}_u \boldsymbol{u}_{cmd} - \boldsymbol{f}_{des}^T \boldsymbol{W} \boldsymbol{T}_u \boldsymbol{u}_{cmd}} + \frac{1}{2}\boldsymbol{f}_{des}^T \boldsymbol{W}\,\boldsymbol{f}_{des} \ . \end{aligned}$$

III/ Controller for the underactuated case:

1/ Optimization variable

In the case of the humanoid control,
U and u_{cmd} are:

$$u_{cmd} = \begin{bmatrix} \tau_{joint,cmd} \\ w_{EE,cmd} \end{bmatrix} \qquad U = \begin{bmatrix} S & L_{EE}^T \end{bmatrix}$$

2/ Contact and joint torque constraint:

2) contact and actuation constraints: The commanded end effector wrenches $\boldsymbol{w}_{EE,cmd}$ just introduced are often subject to inequality constraints (the so-called "contact constraints"). As an example, in walking related applications (see Sec. V-B) these contact constraints are typically expressed in the form of unilaterality and friction cone constraints. Omitting such contact constraints may lead to a failure of the robot. Additionally, due to the physical limitations of the robot, it often makes sense to also constrain the commanded joint torques $\boldsymbol{\tau}_{j,cmd}$. This way, actuator saturation may be avoided.

3/ The cost function and the weight matrix

Neglecting the constant term from the above cost function:

$$\underset{u_{cmd}}{\text{minimize}} \quad G_{QP} = \frac{1}{2} \boldsymbol{u}_{cmd}^T \boldsymbol{T}_u^T \boldsymbol{W} \boldsymbol{T}_u \boldsymbol{u}_{cmd} - \boldsymbol{f}_{des}^T \boldsymbol{W} \boldsymbol{T}_u \boldsymbol{u}_{cmd} \quad ,$$
$$\text{subject to} \quad \text{contact and joint torque constraints} \quad .$$

With the weight $W = \Lambda^{-1}\Psi$

Where Γ is a block-diagonal matrix with the task space inertia matrices $\{M_1,\, M_2, \ldots, M_n\}$ as its diagonal

Ψ is a diagonal matrix collects all the scalar weigh between each tasks in the overall Lyanpunov function:

$$V = \sum_{k=1}^{n_T} \left(\psi_k V_k \right)$$