HOW TO DESIGN, AND TUNE, A COMPUTED TORQUE CONTROLLER:
AN INTRODUCTION, AND A MATLAB EXAMPLE

LLUíS ROS*

Abstract. This note briefly introduces the computed torque control method for trajectory tracking. The method is applicable to fully actuated robots, i.e, those whose inverse dynamics can be solved for any feasible acceleration. This includes many systems, like robot arms or hands, or any tree-like mechanism with all its joints actuated. Using simple explanations, we see how such a controller can be obtained using feedback linearization, and how its gains can be tuned to satisfy a desired settling time for the error signal. We end up discussing the advantages and shortcomings of the controller. A companion Matlab script can be downloaded from https://bit.ly/3QShxYi that implements and tests the controller on a simple actuated pendulum.

Key words. Computed torque control, inverse dynamics, feedback linearization, trajectory tracking.

1. Goal. We wish to design a control law that is able to stabilize a robot along a desired trajectory. We assume the robot is fully actuated, and unconstrained, so its configuration is given by a vector $\mathbf{q} = (q_1, \ldots, q_n)$ of n independent coordinates, each corresponding to an actuated joint angle or displacement[1]. We also assume that the trajectory is expressed as a function $\mathbf{q}_d(t)$ that gives, for each time t, the desired value of $\mathbf{q}$ (Fig. 1). The first and second derivatives of this function, $\dot{\mathbf{q}}_d(t)$ and $\ddot{\mathbf{q}}_d(t)$, provide the desired velocity and acceleration at t.
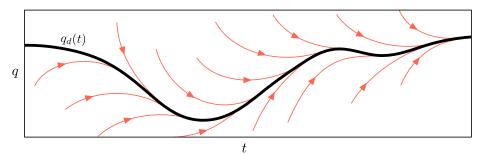

Fig. 1: Idea of trajectory stabilization: Given a desired trajectory $\mathbf{q}_d(t)$ (in black), we wish to design a control law that makes the possible real trajectories (in red) asymptotically convergent to $\mathbf{q}_d(t)$ as t increases. Each red trajectory shown here corresponds to a different set of initial conditions $(t, \mathbf{q})$ of the robot. Only the behaviour of one component of $\mathbf{q}$ is depicted.

2. The control law. To design the controller, we depart from the Euler-Lagrange equation of the robot,

$$\mathbf{M}\,\ddot{\mathbf{q}} + \mathbf{C}\,\dot{\mathbf{q}} + \mathbf{G} - \mathbf{u}_f = \mathbf{u}, \tag{2.1}$$

where $\mathbf{M}$ is the $n \times n$ positive-definite mass matrix, $\mathbf{C}$ is the $n \times n$ Coriolis matrix, $\mathbf{G} \in \mathbb{R}^n$ is the gravity term, $\mathbf{u}_f \in \mathbb{R}^n$ is the generalized force of friction, and $\mathbf{u} \in \mathbb{R}^n$ is the vector of motor forces and torques (u_i is the force or torque applied by the motor at q_i).

Our first step will be to change the dynamics of the system into a simpler one that is easier to control. This can be done using the feedback law

$$\mathbf{u} = \mathbf{M}\,\mathbf{v} + \mathbf{C}\,\dot{\mathbf{q}} + \mathbf{G} - \mathbf{u}_f, \tag{2.2}$$

---

*Institut de Robòtica i Informàtica Industrial (CSIC/UPC), C. Llorens Artigas 4-6, 08028 Barcelona, Catalonia (ros@iri.upc.edu, http://www.iri.upc.edu/people/ros/).

[1]Computed-torque controllers can also be designed for constrained systems [1], as long as they are fully actuated, but these are left out of the scope of this note.

where $\mathbf{v} \in \mathbb{R}^n$ is a new control input that is yet to be chosen. Note that if we substitute Eq. (2.2) into (2.1) we obtain

$$\mathbf{M}\,\ddot{\mathbf{q}} = \mathbf{M}\,\mathbf{v}$$

and since $\mathbf{M}$ is nonsingular, this implies that

$$\ddot{\mathbf{q}} = \mathbf{v}. \tag{2.3}$$

Therefore, the use of the feedback in (2.2) converts our robot into a linear system— the double integrator in (2.3)—so we can now stabilize this system along $\mathbf{q}_d(t)$ using linear control theory. The process of transforming (2.1) into (2.3) is known as feedback linearization.

Note that (2.3) contains n scalar ODEs of the form $\ddot{q}_i = v_i$, so, if we choose v_i to depend only on q_i and $\dot{q}_i$, the evolution of q_i will be fully determined by the ith ODE $\ddot{q}_i = v_i$. To stabilize (2.3) along $\mathbf{q}_d(t)$, thus, it suffices to design a control law for each scalar subsystem $\ddot{q}_i = v_i$ independently. For ease of notation, let us drop the subindex i and write $\ddot{q}_i = v_i$ simply as

$$\ddot{q} = v. \tag{2.4}$$

A simple law that stabilizes the system along $q_d(t)$ is

$$v = \ddot{q}_d - k_p(q - q_d) - k_v(\dot{q} - \dot{q}_d) \tag{2.5}$$

where $k_p > 0$ and $k_v > 0$. Certainly, if we substitute (2.5) into (2.4) we obtain

$$\ddot{q} - \ddot{q}_d + k_p(q - q_d) + k_v(\dot{q} - \dot{q}_d) = 0, \tag{2.6}$$

or, equivalently,

$$\ddot{\varepsilon} + k_v\dot{\varepsilon} + k_p\varepsilon = 0, \tag{2.7}$$

where

$$\varepsilon(t) = q(t) - q_d(t)$$

is the trajectory error. Equation (2.7) is a 2nd order linear homogeneous ODE that determines the evolution of $\varepsilon(t)$, and it is well known that, if we choose $k_p > 0$ and $k_v > 0$, then

$$\lim_{t \to \infty} \varepsilon(t) = 0$$

independently of the initial conditions $q(0)$ and $\dot{q}(0)$. Such a controller, thus, achieves global stability.

A common choice is to set

$$k_p = \omega_0^2,$$
$$k_v = 2\omega_0,$$

so (2.7) describes a critically damped harmonic oscillator for the ε coordinate, with natural frequency ω_0 [2, 9]. A proper value for ω_0 can be chosen by specifying the desired settling time T_s for $\varepsilon(t)$. This is the time needed to ensure $\varepsilon(t) \leq 0.02 \cdot \varepsilon(0)$ for $t > T_s$, assuming $\dot{\varepsilon}(0) = 0$. Appendix A shows that, to achieve such a time, one must set

$$\omega_0 = 5.8339/T_s. \tag{2.8}$$

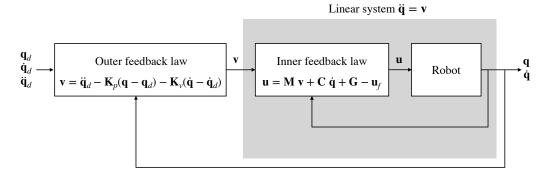Linear system $\ddot{\mathbf{q}} = \mathbf{v}$

Fig. 2: Block diagram of the computed torque control law.

By particularizing Eq. (2.5) for each one of the coordinates in $\mathbf{q} = (q_1, \ldots, q_n)$, we see that the n-dimensional control law that stabilizes (2.3) along $\mathbf{q}_d(t)$ is

$$
\mathbf{v} = \underbrace{\begin{bmatrix} \ddot{q}_{1,d} \\ \vdots \\ \ddot{q}_{n,d} \end{bmatrix}}_{\ddot{\mathbf{q}}_d} - \underbrace{\begin{bmatrix} k_{p,1} & & \\ & \ddots & \\ & & k_{p,n} \end{bmatrix}}_{\mathbf{K}_p} \underbrace{\begin{bmatrix} q_1 - q_{1,d} \\ \vdots \\ q_n - q_{n,d} \end{bmatrix}}_{\mathbf{q} - \mathbf{q}_d} -
$$
$$
- \underbrace{\begin{bmatrix} k_{v,1} & & \\ & \ddots & \\ & & k_{v,n} \end{bmatrix}}_{\mathbf{K}_v} \underbrace{\begin{bmatrix} \dot{q}_1 - \dot{q}_{1,d} \\ \vdots \\ \dot{q}_n - \dot{q}_{n,d} \end{bmatrix}}_{\dot{\mathbf{q}} - \dot{\mathbf{q}}_d}
$$

(2.9)

where $k_{p,i}$ and $k_{v,i}$ are the position and velocity gains for q_i, and $q_{i,d}$ is the ith coordinate of $\mathbf{q}_d$. If we write Eq. (2.9) as

$$
\mathbf{v} = \ddot{\mathbf{q}}_d - \mathbf{K}_p\left(\mathbf{q} - \mathbf{q}_d\right) - \mathbf{K}_v(\dot{\mathbf{q}} - \dot{\mathbf{q}}_d)
$$

and substitute it back into Eq. (2.2), we finally obtain

$$
\mathbf{u} = \mathbf{M} \underbrace{\left[\, \ddot{\mathbf{q}}_d - \mathbf{K}_p(\mathbf{q} - \mathbf{q}_d) - \mathbf{K}_v(\dot{\mathbf{q}} - \dot{\mathbf{q}}_d)\,\right]}_{\mathbf{v}} + \mathbf{C}\,\dot{\mathbf{q}} + \mathbf{G} - \mathbf{u}_f
$$

(2.10)

which is the usual computed-torque control law for a fully actuated robot (Fig. 2).

Observe that, to control the robot using (2.10), we need sensory feedback of the $\mathbf{q}$ and $\dot{\mathbf{q}}$ values at each iteration of the control loop, not only to obtain the configuration and velocity errors $\mathbf{q} - \mathbf{q}_d$ and $\dot{\mathbf{q}} - \dot{\mathbf{q}}_d$, but also the values of $\mathbf{M}$, $\mathbf{C}$, $\mathbf{G}$, and $\mathbf{u}_f$, whose dependencies on $\mathbf{q}$ and $\dot{\mathbf{q}}$ are

$$
\mathbf{M}(\mathbf{q}), \quad \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}), \quad \mathbf{G}(\mathbf{q}), \quad \mathbf{u}_f(\mathbf{q}, \dot{\mathbf{q}}).
$$

Moreover, the calculation of $\mathbf{u}$ using (2.10) has to be fast to achieve a high-enough control frequency. Since this calculation is equivalent to solving the inverse dynamics for the $\mathbf{v}$ value in (2.10), achieving fast algorithms for this purpose is of utmost importance in this context.

3. Discussion. The computed torque control law in Eq. (2.10) ensures the convergence of the robot to $\mathbf{q}_d(t)$ independently from the initial conditions $\mathbf{q}(0)$. This property

is remarkable, and it is sometimes referred to as "global stability". The main drawback of the law is that it requires a very good model of the robot, otherwise the feedback in Eq. (2.2) will not transform the system into a double integrator as required. Another drawback is that limits in the control forces or torques are not considered in this method. If such limits are somehow surpassed while operating the robot, the convergence towards $\mathbf{q}_d(t)$ may take longer, or even be impossible. This is why robots being controlled by such laws typically employ powerful motors. Also, choosing a suitable settling time T_s is crucial. Small T_s values are always preferrable, but yield a more aggressive controller, which increases the risk of violating the force limits. Thus, T_s cannot be chosen too small to ensure a proper functioning of the robot.

4. An example. A companion Matlab script implements and tests the controller on a simple actuated pendulum [5]. In its default run, the script starts showing the pendulum at rest in its bottom configuration, and then simulates the tracking of a step trajectory $\mathbf{q}_d(t)$ that requires the pendulum to go to its upright configuration, and then back to the initial one. A perturbation force is applied while the pendulum is in inverted balance to see how the controller is able to counteract disturbances. The script is fully commented for ease of comprehension.

5. Further reading. Many books cover the topic of this note. For further details we refer to [7, 4, 6, 3]. The book in [7] covers the advanced topic of designing a computed torque controller that is more robust to various sources of uncertainty, such as modeling errors, unknown loads, or computational errors. Note also that, like [7], some references use the term "inverse dynamics control" as a synonym of "computed torque control".

Acknowledgements. I am grateful to Enric Celaya, Ricard Bordalba, Pere Giró, Adriano del Río, and Siro Moreno for fruitful discussions around the topic of this note.

REFERENCES

[1] R. BORDALBA, *Kinodynamic planning and control of closed-chain robotic systems*, https://www.iri.upc.edu/thesis/show/111. Ph.D. thesis, Universitat Politècnica de Catalunya, 2021.

[2] R. FEYNMAN, R. B. LEIGHTON, AND M. SANDS, *The Feynman lectures on physics*, https://www.feynmanlectures.caltech.edu/I_21.html. Lecture 21: The harmonic oscillator.

[3] O. KHATIB AND B. SICILIANO, *Springer Handbook of Robotics*, Springer, 2016. Chapter 8.

[4] K. M. LYNCH AND F. C. PARK, *Modern Robotics*, Cambridge University Press, 2017.

[5] L. ROS, *Companion Matlab script of this note*, https://bit.ly/3QShxYi.

[6] J.-J. SLOTINE AND W. LI, *Applied Nonlinear Control*, Prentice Hall, 1991.

[7] M. W. SPONG, S. HUTCHINSON, AND M. VIDYASAGAR, *Robot Modeling and Control*, Wiley, 2020.

[8] M. TENENBAUM AND H. POLLARD, *Ordinary Differential Equations: An Elementary Textbook for Students of Mathematics, Engineering, and the Sciences*, Courier Corporation, 1985.

[9] S. THORNTON AND J. MARION, *Classical Dynamics of Particles and Systems*, Brooks, 5th ed., 2003.

Appendix A. Natural frequency for a desired settling time. Consider a damped harmonic oscillator described by the ODE

$$m\ddot{x} = -kx - c\dot{x},\tag{A.1}$$

where m is the attached mass, x is the mass position, k is the spring constant, and c is the viscous friction coefficient of the mass-ground contact (Fig. 3). By defining

$$\omega_0 = \sqrt{\frac{k}{m}}\tag{A.2}$$

$$\xi = \frac{c}{2\sqrt{mk}}\tag{A.3}$$

Eq. (A.1) can be written as

$$\ddot{x} + 2\xi\omega_0\dot{x} + \omega_0^2 = 0,\tag{A.4}$$

where ω_0 is the natural frequency of the oscillator and ξ is its damping ratio. It is well known that, for a fixed value of k and m, the fastest way in which x converges to zero, without oscillations, occurs when $\xi = 1$. In such a case, [9, 8] show that the solution of (A.4) is

$$x(t) = (c_1 + c_2 t)e^{-\omega_0 t},\tag{A.5}$$

where c_1 and c_2 are to be determined using the initial conditions

$$x(0) = x_0,\tag{A.6}$$
$$\dot{x}(0) = v_0.\tag{A.7}$$

Evaluating (A.6) and (A.7) using (A.5), and solving for c_1 and c_2, we obtain

$$c_1 = x_0,\tag{A.8}$$
$$c_2 = v_0 + x_0\omega_0,\tag{A.9}$$

so (A.5) can be written as

$$x(t) = [\,x_0 + (v_0 + x_0\omega_0)\,t\,]\cdot e^{-\omega_0 t}.\tag{A.10}$$

We wish to find the value ω_0 for which $x(t)$ achieves a desired settling time T_s. This is the time needed for $x(t)$ to be below $0.02 \cdot x_0$, assuming $v_0 = 0$ and $x_0 \neq 0$. Thus, we must solve

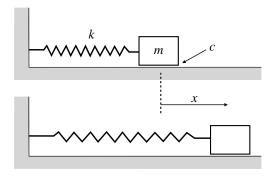$$(x_0 + x_0\omega_0 T_s)e^{-\omega_0 T_s} = 0.02\,x_0,\tag{A.11}$$

Fig. 3: A harmonic oscillator with viscous friction at the ground contact. Top: the equilibrium configuration. Bottom: a generic configuration at a distance x from the equilibrium one.

or equivalently, dividing both sides by x_0,

$$(1 + \omega_0 T_s)e^{-\omega_0 T_s} = 0.02. \tag{A.12}$$

If we define $P = \omega_0 T_s$, and rearrange the terms, (A.12) becomes

$$(1 + P) = 0.02 \, e^P \tag{A.13}$$

which has the single solution

$$P \approx 5.8339 \tag{A.14}$$

for $P > 0$ (P must be positive because $\omega_0 > 0$ and $T_s > 0$). Therefore, the value ω_0 that achieves the desired settling time is

$$\omega_0 \approx \frac{5.8339}{T_s}. \tag{A.15}$$