Optimization-based Control of Robotic Systems

Gennaro Notomista

Updated: January 25, 2023

This course will introduce students to modern optimization-based methods for robot control. Robot models will be described first. Then, unconstrained and constrained optimization problems will be introduced. The special case of convex optimization will be presented and used to formulate stabilizing and safety-ensuring controllers for robotic systems. Finally, two lectures will be dedicated to the optimization-based control of manipulators and mobile robots, respectively. By the end of the course, students should be able to formulate and solve robot control problems arising in their research projects by means of optimization-based control techniques.

Contents

Introduction to Robot Control

Introduction to Feedback Control

Try to balance a stick with your eyes closed.

Solution: impossible.

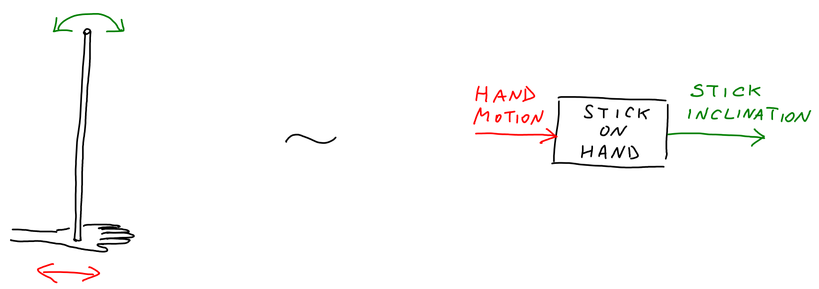


Figure 1: Balancing a stick.

Opening your eyes closes the loop.

And now balancing the stick becomes possible.

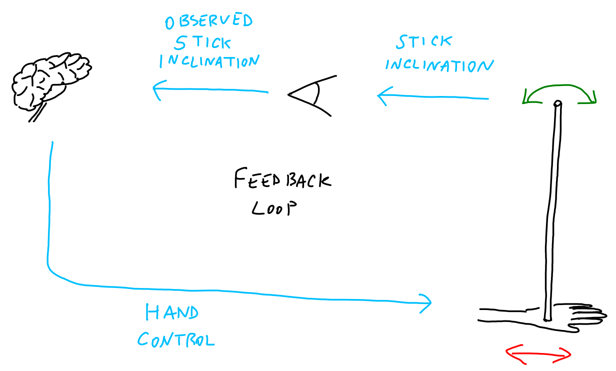


Figure 2: Closing the loop to balance a stick.

Closed-loop, or feedback, control is a powerful tool to make systems (e. g., sticks on our hand) behave as we wish (e. g., stay upright).

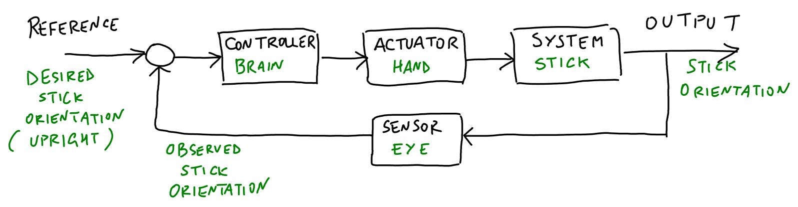


Figure 3: Block diagram of a closed-loop stick inclination control.

Feedback Control of Robotic Systems

Robotics is commonly defined as the science studying the intelligent connection between perception and action[1]—which does not sound too different from what feedback control is. With the tremendous developments that artificial intelligence and machine learning had in the last few decades, and the application of these disciplines to robotic systems, the definition given above probably does not encompass everything that nowadays we would recognize to be a robot. For the sake of controlling robots, however, the above is still an accurate definition.

[1] Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, and Giuseppe Oriolo. *Robotics: modelling, planning and control*. Springer Science & Business Media, 2010; and Richard M. Murray, Zexiang Li, and S. Shankar Sastry. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994

In these lectures, we will focus on robotic manipulators and mobile robots. The former are comprised of multiple rigid bodies interconnected to each other by different types of joints, and are typically anchored to a fix point in space. The latter can (more or less) freely move in space.

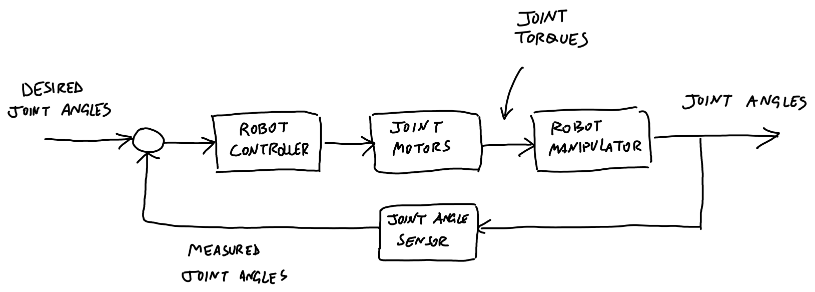


Figure 4: Feedback control of a robotic manipulator.

Figure 4 shows the feedback loop used to control a robotic manipulator, where joint torques are used to regulate joint angles to desired values.

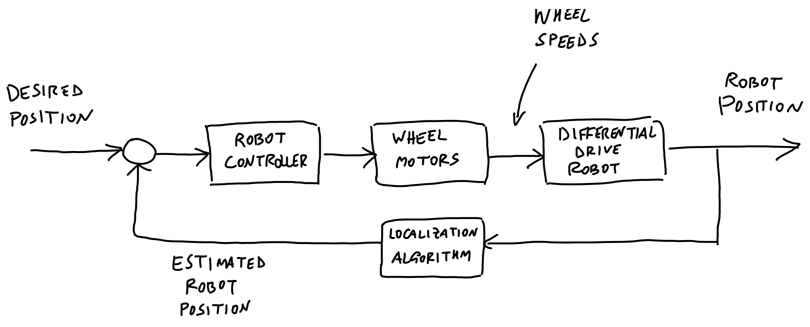


Figure 5: Feedback control of a differential drive mobile robot.

Figure 5 shows the feedback loop used to control a differential-drive mobile robot, where the speeds of its left and right wheels can be independently controlled to move forward and backward, and to turn left and right. This allows the robot to move to a desired position.

CONTROLLING A ROBOT means defining functions to evaluate the control inputs (e. g., joint torques, wheel speeds) to be supplied to the robot for it to achieve a desired behavior. In this course, we will look at optimal ways to define such functions. In particular, the controller synthesis will involve solving optimization problems in the feedback loop.

Kinematic Model of Robotic Systems

The robot kinematics are mathematical relations describing how a robot moves without considering the forces and torques that caused the motion.

THE CONFIGURATION q OF A ROBOT is a complete description of the location of every point of the robot. The set of all configurations is the configuration space and it is denoted by $\mathcal{C}$.

Example 1 *The configuration of a mobile robot translating on a plane can be described using a* 2*-dimensional vector whose components are the coordinates of the robot in a reference frame defined on the plane. Therefore,* $q = [x, y]^T \in \mathbb{R}^2 = \mathcal{C}$ *(see Fig. 6).*

Example 2 *The configuration of a mobile robot translating and rotating on a plane can be described using a* 3*-dimensional vector whose components are the coordinates of the robot in a reference frame defined on the plane and its orientation with respect to a fixed axis in the plane. Therefore,* $q = [x, y, \theta]^T \in \mathbb{R}^2 \times SO(2) = \mathcal{C}$ *(see Fig. 7).*

Example 3 *The configuration of a manipulator with n revolute joints can be described using a n-dimensional vector whose components are the angles of the n revolute joints of the robot. Therefore,* $q = [\theta_1, \ldots, \theta_n]^T \in \mathbb{T}^n = \underbrace{S^1 \times \ldots \times S^1}_{n} = \mathcal{C}$ *(see Fig. 8).*

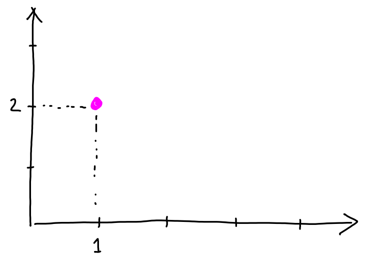


Figure 6: Planar robot at configuration $q = [1, 2] \in \mathbb{R}^2$.

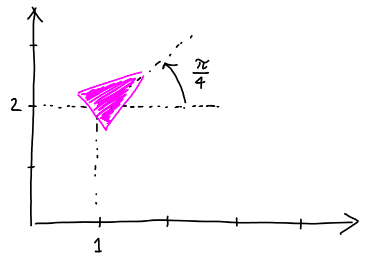


Figure 7: Planar robot at configuration $q = [1, 2, \pi/4] \in \mathbb{R}^2 \times SO(2)$.

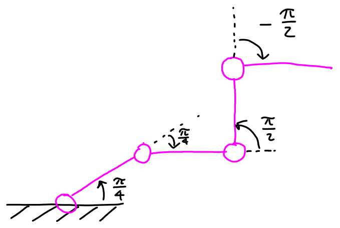


Figure 8: Manipulator robot at configuration $q = [\pi/4, -\pi/4, \pi/2, -\pi/2] \in \mathbb{T}^4$.

THE FORWARD KINEMATICS consists in determining the pose (position and orientation) of the end effector, x_e, as a function of the configuration (angles of the joints) of the robot, q:

$$x_e = f(q), \tag{1}$$

where $f : \mathcal{C} \rightarrow \mathcal{T} : q \mapsto x_e$ maps from the configuration space to the *task space* $\mathcal{T}$, to which the pose of the end effector belongs.

For robots comprised of a single rigid body, the function f is trivial, while it may be quite complicated for robotic systems comprised of multiple connected rigid bodies, such as robotic manipulators and articulated mobile robots

`Link to Google Colab for the forward kinematics of manipulators.`

THE DIFFERENTIAL (OR VELOCITY) KINEMATICS express the relation between the velocities in the task space, $\dot{x}_e$, and the velocities in the configuration space, $\dot{q}$. Since the forward kinematics map q to x_e, the mathematical expression of the differential kinematics can be determined by differentiating (1):

$$\dot{x}_e = J(q)\dot{q}, \tag{2}$$

where

$$J(q) = \frac{\partial f}{\partial q}(q) \tag{3}$$

is the Jacobian of f, which plays an important role in the analysis of the motion of robotic systems.

The Jacobian is also used in algorithms to solve the inverse kinematics problem, i. e., finding the configuration $\bar{q}$ to achieve a given pose of the end effector $\bar{x}_e$. Using optimization-based controllers we will not need to deal with this problem explicitly.

In the case of mobile robots, the kinematic model expresses the relation between velocities in the configuration space, $\dot{q}$, and control inputs, generally denoted by $u \in \mathbb{R}^m$, for some m, in the following form:

$$\dot{q} = g(q)u. \tag{4}$$

A full treatment of how to derive the kinematic model of mobile robots can be found in traditional robotics books[2]. In the following, an important example of mobile robot is reported.

[2] Mark W. Spong, Seth Hutchinson, and Mathukumalli Vidyasagar. *Robot modeling and control*. John Wiley & Sons, 2020; and Alessandro De Luca and Giuseppe Oriolo. Modelling and control of nonholonomic mechanical systems. In *Kinematics and dynamics of multi-body systems*, pages 277–342. Springer, 1995

Example 4 (Unicycle) *Unicycles are used to model a large variety of mobile robotic systems: ground, marine, and even aerial robots are very often abstracted using a rigid body that can roll without slipping on a planar surface as a coin (see Fig. 9). The kinematic model of the unicycle is given by:*

$$\begin{cases} \dot{x} = v\cos\theta \\ \dot{y} = v\sin\theta \\ \dot{\theta} = \omega, \end{cases} \tag{5}$$

where x and y are the coordinates of the position of the system in a reference frame defined on the plane where the robot moves, θ is its orientation, and v and ω are the linear and angular velocity control inputs. Therefore, defining the configuration of the robot $q = [q_1, q_2, q_3]^T = [x, y, \theta]^T$ and the control input vector $u = [u_1, u_2]^T = [v, \omega]^T$, we can write (5) as follows:

$$\dot{q} = \begin{bmatrix} \cos q_3 \\ \sin q_3 \\ 0 \end{bmatrix} u_1 + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u_2. \tag{6}$$

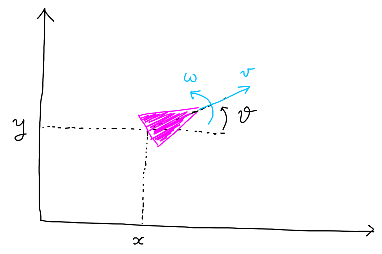


Figure 9: Unicycle.

```
Link to Google Colab for the differential kinematics of
manipulators and mobile robots.
```

Dynamic Model of Robotic Systems

While the kinematic description of a robot is purely geometric, the dynamics of a robot consist in the mathematical relation describing the effect that generalized forces (forces and torques) acting on the generalized coordinates (components of the robot configuration) of the robot have on the motion of the robot. In other words, the dynamics of a robot tell us, for instance, how joint torques, τ, of a manipulator generate joint accelerations, $\ddot{q}$. Mathematically, this can be expressed as follows:

$$D(q)\ddot{q} + C(q, \dot{q})\dot{q} + g(q) = \tau, \tag{7}$$

where the terms on the left-hand side represent the torques acting on the joints due to inertial ($D(q)\ddot{q}$), centrifugal and Coriolis ($C(q, \dot{q})\dot{q}$), and gravitational ($g(q)$) effects.

In this course, we will focus on kinematic models of robotic systems. The same formulation, however, can be applied to dynamic models as well.

Unconstrained, Constrained, and Convex Optimization Problems

In the next sections, our objective will be to design robot controllers to achieve desired robot behaviors. In other words, we will figure out what to implement inside the *Robot controller* block of Figures 4 and 5, the output of which are the control inputs for the robotic systems to control. In these lectures, we will focus on optimization-based robot controllers, i. e., control algorithms that involve solving an optimization problem to evaluate the control input.

USING OPTIMIZATION FOR ROBOT CONTROL is particularly convenient for the following reasons[3]:

(i) The language of optimization provides a general and natural way of expressing control objectives in mathematical terms

(ii) There are well-developed theoretical and algorithmic frameworks to solve optimization problems

(iii) Today, we have the computational power to deploy optimization-based controllers on most robotic platforms

[3] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004

In this section, we will recap basic concepts of optimization which will be used in the following sections to design optimization-based robot controllers.

Unconstrained Optimization

The goal in unconstrained optimization is to pick the best value for a decision variable so that a cost is minimized. Let $u = [u_1, \dots, u_m]^T \in \mathbb{R}^m$ be the decision variable, and $g : \mathbb{R}^m \to \mathbb{R}$, $g \in C^1$, be the cost function. The unconstrained optimization problem can be then stated as follows:

C^1 is the set of continuously differentiable functions.

$$\underset{u}{\text{minimize}} \;\; g(u). \tag{8}$$

$u^\star$ IS A (LOCAL) MINIMIZER of g if $\exists \delta > 0$ such that $g(u^\star) \leq g(u)$ $\forall u \in B_\delta(u^\star)$.

A necessary condition for $u^\star$ to be a (local) minimizer of g is that

$$\frac{\partial g}{\partial u}(u^\star) = 0. \tag{9}$$

To prove that is the case, let $u^\star$ be a minimizer of g and pick $\epsilon > 0$ and $v \in \mathbb{R}^m$, $\|v\| = 1$, such that $u^\star + \epsilon v \in B_\delta(u^\star)$ (see Fig. 10). Then,

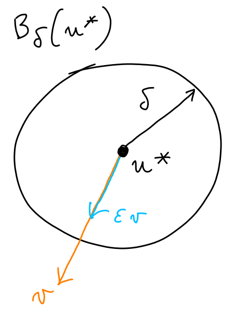


Figure 10: The set $B_\delta(u^\star) = \{u \in \mathbb{R}^m : \|u - u^\star\| < \delta\}$ is the open ball of radius δ centered at $u^\star$.

$$g(u^\star + \epsilon v) = g(u^\star) + \epsilon \frac{\partial g}{\partial u}(u^\star) v + \text{h.o.t.}. \tag{10}$$

h.o.t. stands for higher order terms and it is used to denote terms which are $o(\epsilon)$—*little-o* notation—where $\lim_{\epsilon \to 0} \frac{o(\epsilon)}{\epsilon} = 0$.

Assume $\frac{\partial g}{\partial u}(u^\star) \neq 0$. If that is the case, we could pick $v = -\frac{\partial g}{\partial u}(u^\star)^T$, which would result in

$$\begin{aligned} g(u^\star + \epsilon v) &= g(u^\star) - \epsilon \frac{\partial g}{\partial u}(u^\star) \frac{\partial g}{\partial u}(u^\star)^T + o(\epsilon) \\ &= g(u^\star) - \epsilon \left\| \frac{\partial g}{\partial u}(u^\star) \right\|^2 + o(\epsilon) \\ &< g(u^\star), \end{aligned} \tag{11}$$

for $\epsilon \to 0$. This contradicts the hypothesis that $u^\star$ is a minimizer of g. Therefore, necessarily we need $\frac{\partial g}{\partial u}(u^\star) = 0$.

Remark 1 *The condition in* (9) *is necessary but not sufficient.* $u^\star$ *could satisfy* (9) *and be a maximizer or a saddle point, rather than a minimizer.*

Remark 2 *The expression in* (11) *suggests the following numerical algorithm to solve unconstrained optimization problems:*

```
Initialize $u_0 \in \mathbb{R}^m$, $k = 0$
while $\left\| \frac{\partial g}{\partial u}(u_k)^T \right\| > \Delta$ do
    $u_{k+1} \leftarrow u_k - \alpha \frac{\partial g}{\partial u}(u_k)^T$
    $k \leftarrow k + 1$
end while
```

where $\alpha > 0$ *is the step size and* $\Delta > 0$ *is a convergence threshold. This algorithm is known as the steepest descent algorithm.*

Example 5 *Consider the following unconstrained optimization problem:*

$$\underset{u}{\text{minimize}} \;\; u^T Q u - b^T u, \tag{12}$$

where $Q = Q^T > 0$ *is a symmetric and positive definite* $m \times m$ *matrix, and* $b \in \mathbb{R}^m$*. For* $u^\star$ *to be a minimizer, we need*

$$\frac{\partial}{\partial u}(u^{\star T} Q u^\star - b^T u^\star) = 2u^{\star T} Q - b^T = 0. \tag{13}$$

The minimizer is then given by

$$u^\star = \frac{1}{2} Q^{-1} b. \tag{14}$$

We do not know, however, whether $u^\star$ *is a minimizer, a maximizer, or a saddle point.*

A sufficient condition for $u^\star$ to be a minimizer is based on the computation of the second derivative of the cost function at $u^\star$: $\frac{\partial^2 g}{\partial u^2}(u^\star) > 0 \implies u^\star$ is a minimizer.

If the function g is convex, then a local minimizer of g is the (unique) global minimizer. A function $g : \mathbb{R}^m \to \mathbb{R}$ is convex if $g(\alpha u_1 + (1-\alpha)u_2) \leq \alpha g(u_1) + (1-\alpha) g(u_2)$, $\forall \alpha \in [0,1]$ and $\forall u_1, u_2 \in \mathbb{R}^m$ (see Fig. 11). A sufficient condition for g to be convex is that $\frac{\partial^2 g}{\partial u^2}(u) \geq 0 \; \forall u \in \mathbb{R}^m$.

Example 6 (Example 5 cont.) $\frac{\partial^2 g}{\partial u^2}(u^\star) = Q > 0$*, therefore the optimization cost is convex and* $u^\star = \frac{1}{2} Q^{-1} b$ *is the unique global minimizer.*

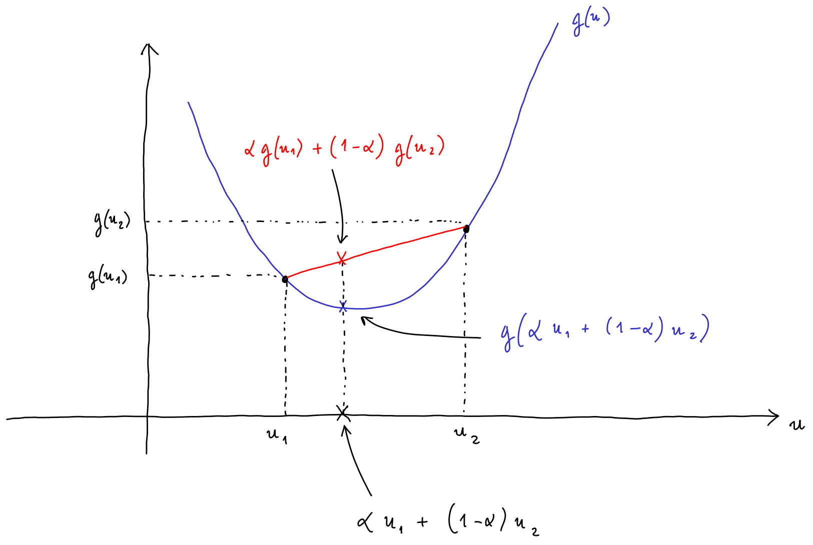

$g(u)$
$\alpha g(u_1) + (1-\alpha) g(u_2)$
$g(u_2)$
$g(u_1)$
$g(\alpha u_1 + (1-\alpha) u_2)$
u_1
u_2
u
$\alpha u_1 + (1-\alpha) u_2$

Figure 11: Definition of convex fucn-tion. g is convex if the red line is above the blue curve between u_1 and u_2.

Example 7 (Inverse kinematics) *Consider the following unconstrained optimization problem:*

$$\underset{\dot{q}}{\text{minimize}} \; \|\dot{x}_e - J\dot{q}\|^2, \tag{15}$$

where $\dot{x}_e \in \mathbb{R}^r$, $J \in \mathbb{R}^{r\times n}$, with $r \geq n$, and we dropped the dependency of J from q for ease of notation. Notice how this is the same optimization problem of Example 5, where $Q = J^T J \geq 0$, $b = 2J^T\dot{x}_e$, and the optimization variable u here is called $\dot{q}$. Then, by substituting the expressions of Q and b in (14), we can write down the solution of (15) directly:

r is the dimension of the task space $\mathcal{T}$ defined in the previous section.

There is actually an additional constant term in the optimization cost of (15) which is however irrelevant for the purpose of minimizing it.

$$\dot{q}^\star = \frac{1}{2}\underbrace{(J^T J)^{-1}}_{Q^{-1}}\underbrace{2J^T\dot{x}_e}_{b} = J_l^\dagger \dot{x}_e, \tag{16}$$

where $J_l^\dagger = (J^T J)^{-1} J^T$ is known as the left pseudo inverse of J and it exists as long as J has maximum rank ($\text{rank}(J) = n$), i.e., it is non singular.

The solution $\dot{q}^\star$ of the optimization problem considered in this example can be used to solve the inverse kinematics problem for manipulators that have less degrees of freedoms (n) that the dimension of the task space (r). The inverse kinematics problem consists in finding the configuration $\bar{q}$ that leads to a given end effector pose $\bar{x}_e$. If we set $\dot{x}_e$ in (15) equal to $\bar{x}_e - x_e$, where x_e is the actual end effector pose, the $\dot{q}^\star$ solution of (15) can be used as a control input to drive the robot to the configuration corresponding $\bar{q}$ to the end effector pose $\bar{x}_e$.

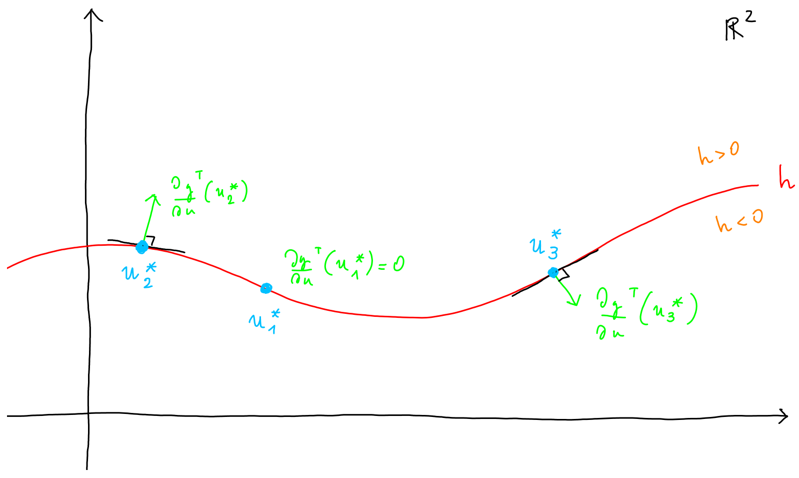
Link to Google Colab for unconstrained optimization.

Constrained Optimization: Equality Constraints

Consider the following optimization problem:

$$\begin{aligned} &\underset{u}{\text{minimize}}\ g(u) \\ &\text{subject to } h(u) = 0, \end{aligned} \tag{17}$$

where $h : \mathbb{R}^m \to \mathbb{R}^k$. In the unconstrained optimization problem in (8), we could search for the optimal u in the entire $\mathbb{R}^m$. In the constrained optimization problem (17), we restrict the search space to decision variables u such that $h(u) = 0$. Such decision variables are called *feasible*.



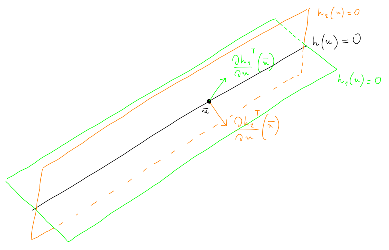
$\mathbb{R}^2$
$h>0$
$h=0$
$h<0$
$\frac{\partial g}{\partial u}^T(u_2^*)$
$\frac{\partial g}{\partial u}^T(u_1^*)=0$
u_3^*
u_2^*
u_1^*
$\frac{\partial g}{\partial u}^T(u_3^*)$

Figure 12: Optimal solutions of (17). u is optimal if the gradient of g vanishes at u (as for the case of $u_1^\star$) or if it is orthogonal to the tangent to the curve corresponding to $h = 0$ (as for the case of $u_2^\star$ and $u_3^\star$). The case depicted is for $m = 2$ and $k = 1$, so $g : \mathbb{R}^2 \to \mathbb{R}$ and $h : \mathbb{R}^2 \to \mathbb{R}$.

Looking at Fig. 12, for u to be the solution of (17), either $\frac{\partial g}{\partial u}(u)^T = 0$ or $\frac{\partial g}{\partial u}(u)^T \perp Th$, where Th denotes the tangent line/plane to the curve/surface $h = 0$. The tangent line/plane to the curve/surface $h = 0$ is orthogonal to the gradient of h, i.e., $Th \perp \frac{\partial h}{\partial u}(u)^T$. Then, we can express the optimality conditions as follows: $u^\star$ is optimal if

$$\begin{aligned} &\frac{\partial g}{\partial u}(u^\star)^T \parallel \frac{\partial h}{\partial u}(u^\star)^T \\ \text{i.e.}\quad &\frac{\partial g}{\partial u}(u^\star)^T = -\lambda \frac{\partial h}{\partial u}(u^\star)^T \quad \text{for some } \lambda \in \mathbb{R} \\ \text{i.e.}\quad &\frac{\partial g}{\partial u}(u^\star)^T + \lambda \frac{\partial h}{\partial u}(u^\star)^T = 0 \quad \text{for some } \lambda \in \mathbb{R} \\ \text{i.e.}\quad &\frac{\partial}{\partial u}(g(u^\star) + \lambda h(u^\star)) = 0 \quad \text{for some } \lambda \in \mathbb{R}. \end{aligned} \tag{18}$$

In the general case (arbitrary m and k), $h(u) = 0$ is a system of k equations, each of which ($h_i(u) = 0, i = 1, \ldots, k$) represents a $k-1$-dimensional plane (see Fig. 13). Thus, geometrically, $h(u) = 0$ represents a line. The same reasoning carried out above holds, and



$h_2(u)=0$
$h(u)=0$
$\frac{\partial h_1}{\partial u}^T(\bar{u})$
$h_1(u)=0$
$\bar{u}$
$\frac{\partial h_2}{\partial u}^T(\bar{u})$

Figure 13: $h(u) = 0$ is the line (1-dimensional variety) intersection of k $k-1$-dimensional planes, $h_1(u) = 0, \ldots, h_k(u) = 0$.

we can write the necessary condition for optimality as follows:

$$\begin{aligned} &\frac{\partial g}{\partial u}(u^\star)^T = -\sum_{i=1}^{k} \lambda_i \frac{\partial h_i}{\partial u}(u^\star)^T \quad \text{for some } \lambda \in \mathbb{R}^k \\ \text{i. e.}\quad &\frac{\partial g}{\partial u}(u^\star)^T + \lambda^T \frac{\partial h}{\partial u}(u^\star)^T = 0 \quad \text{for some } \lambda \in \mathbb{R}^k \\ \text{i. e.}\quad &\frac{\partial}{\partial u}(g(u^\star) + \lambda^T h(u^\star)) = 0 \quad \text{for some } \lambda \in \mathbb{R}^k, \end{aligned} \tag{19}$$

where $\lambda = [\lambda_1, \ldots, \lambda_k]^T$ is a k-dimensional vecetor.

THE FUNCTION $L(u, \lambda) = g(u) + \lambda^T h(u)$ which appears in (19) is known as the *Lagrangian* and λ is the vector of *Lagrange multipliers*. If $u^\star$ is a minimizer of (17), then $\exists \lambda^\star \in \mathbb{R}^k$ such that

$$\begin{cases} \frac{\partial L}{\partial u}(u^\star, \lambda^\star) = 0 \\ \frac{\partial L}{\partial \lambda}(u^\star, \lambda^\star) = 0. \end{cases} \tag{20}$$

Example 8 *Consider the following constrained optimization problem:*

$$\begin{aligned} &\underset{u}{\text{minimize}}\ \frac{1}{2}\|u\|^2 \\ &\text{subject to } Au = b, \end{aligned} \tag{21}$$

where $A \in \mathbb{R}^{k \times m}$, $k \leq m$, has linearly independent rows, and $b \in \mathbb{R}^k$, $b \in \mathcal{R}(A)$.

If A has linearly independent rows, rank $(A) = k$. This condition corresponds to enforcing linearly independent constraints.

$\mathcal{R}(A) = \{v \in \mathbb{R}^k : \exists w \in \mathbb{R}^m$ such that $Aw = v\}$ is the range of A.

To find the minimizer, we define the Lagrangian $L(u, \lambda) = \frac{1}{2}\|u\|^2 + \lambda^T(Au - b)$ and solve the following system of equations:

$$\begin{cases} \frac{\partial L}{\partial u}(u^\star, \lambda^\star) = u^{\star T} + \lambda^{\star T} A = 0 \\ \frac{\partial L}{\partial \lambda}(u^\star, \lambda^\star) = Au^\star - b = 0. \end{cases} \tag{22}$$

From the first equation we have $u^\star = -A^T\lambda^\star$, which, substituted in the second equation, gives $\lambda^\star = -(AA^T)^{-1}b$. Thus, $u^\star = A^T(AA^T)^{-1}b = A_r^\dagger b$, where $A_r^\dagger$ is known as the right pseudo inverse of A and it exists thanks to the assumption that A has linearly independent rows.

Exercise 1 (Inverse kinematics) *Solve the following constrained optimization problem:*

$$\begin{aligned} &\underset{\dot{q}}{\text{minimize}}\ \|\dot{q}\|^2 \\ &\text{subject to } J\dot{q} = \dot{x}_e, \end{aligned} \tag{23}$$

where $\dot{x}_e \in \mathbb{R}^r$, $\dot{x}_e \in \mathcal{R}(J)$, and $J \in \mathbb{R}^{r \times n}$, with $r \leq n$.

Similarly to Example 7, the solution $\dot{q}^\star$ of the optimization problem (23) can be used to solve the inverse kinematics problem, this time for redundant manipulators, i. e., manipulators that have more degrees of freedoms (n) that the dimension of the task space (r). Setting $\dot{x}_e = \bar{x}_e - x_e$ in (23), where x_e is the actual end effector pose, the $\dot{q}^\star$ can be used as a control input to drive the robot to the configuration corresponding to the desired end effector pose $\bar{x}_e$.

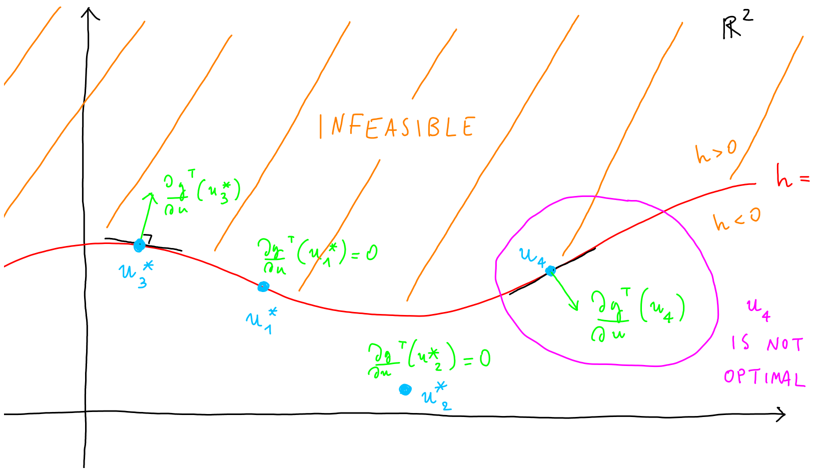
`Link to Google Colab for optimization with equality constraints.`

Constrained Optimization: Inequality Constraints

Consider the following optimization problem:

$$\begin{aligned} &\underset{u}{\text{minimize}}\ g(u) \\ &\text{subject to}\ h(u) \leq 0, \end{aligned} \tag{24}$$

where $h : \mathbb{R}^m \rightarrow \mathbb{R}^k$ and the symbol $\leq$ is used component-wise. Feasible decision variables of (8) are all u such that $h(u) \leq 0$. Necessary conditions for optimality can be derived similarly to the case of equality constraints of the previous section. The condition for u to be



$\mathbb{R}^2$
INFEASIBLE
$h>0$
$h=0$
$h<0$
$\frac{\partial g}{\partial u}^T(u_3^*)$
u_3^*
$\frac{\partial g}{\partial u}^T(u_1^*)=0$
u_1^*
u_4
$\frac{\partial g}{\partial u}^T(u_4)$
$\frac{\partial g}{\partial u}^T(u_2^*)=0$
u_2^*
u_4 IS NOT OPTIMAL

Figure 14: Optimal solutions of (24). u is optimal if the gradient of g vanishes at u (as for the case of $u_1^\star$ and $u_2^\star$) or if it is orthogonal to the tangent to the curve corresponding to $h = 0$ and pointing towards the infeasible region, where $h > 0$ (as for the case of $u_3^\star$). Notice how u_4 is not optimal. In fact, the gradient at u_4 is orthogonal to the tangent to the curve corresponding to $h = 0$ but it is pointing towards the feasible region. Therefore, there are nearby points which are feasible and yield a lower value of the cost g. The case depicted is for $m = 2$ and $k = 1$, so $g : \mathbb{R}^2 \rightarrow \mathbb{R}$ and $h : \mathbb{R}^2 \rightarrow \mathbb{R}$.

the solution of (24) are the following (see Fig. 14):

$$\begin{cases} \text{If } h(u^\star) < 0, \text{ then } \frac{\partial g}{\partial u}(u^\star) = 0 \\ \text{If } h(u^\star) = 0, \text{ then } \frac{\partial g}{\partial u}(u^\star) = 0 \ \vee\ \frac{\partial g}{\partial u}(u^\star) = -\lambda^T \frac{\partial h}{\partial u}(u^\star), \text{ for } \lambda > 0. \end{cases} \tag{25}$$

These conditions can be compactly expressed as follows:

$$\begin{cases} \frac{\partial}{\partial u}(g(u^\star) + \lambda^{\star T} h(u^\star)) = 0 \\ \lambda^{\star T} h(u^\star) = 0 \\ \lambda^\star \geq 0 \end{cases} \tag{26}$$

Therefore, if $u^\star$ is a minimizer of (24), then $\exists \lambda^\star \in \mathbb{R}^k$ such that

$$\begin{cases} \frac{\partial L}{\partial u}(u^\star, \lambda^\star) = 0 \\ h(u^\star) \leq 0 \\ \lambda^{\star T} h(u^\star) = 0 \\ \lambda^\star \geq 0. \end{cases} \tag{27}$$

Link to Google Colab for optimization with inequality constraints.

Min-norm controllers – Part I: Stability and Control Lyapunov Functions

The optimization tools developed in the previous section are used in this and the following section to develop controllers to execute high-level robotic tasks. The objective of the optimization problems will be to evaluate the best control inputs (e. g., $\dot{q}$ in the kinematic model of manipulators (2), u in the kinematic model of mobile robots (4)) with respect to some cost function and subject to a number of constraints. In particular, in the following we will focus on expressing *robotic tasks as constraints* of specific classes of optimization problems.

As we will deal with manipulators and mobile robots alike, from now on, we will express the model of such robotic systems in the following form, known as *control affine* :

Control affine structures arise in all systems whose model is derived from the Euler-Lagrange equations. The term f_0 encodes inertial effects and is nonzero only if the dynamics of the system are considered, as shown in Example 9.

$$\dot{x} = f_0(x) + f_1(x)u, \tag{28}$$

where $x \in \mathbb{R}^n$ denotes the robot state, $u \in \mathbb{R}^m$ the robot input, $f_0 : \mathbb{R}^n \to \mathbb{R}^n$ and $f_1 : \mathbb{R}^n \to \mathbb{R}^{n\times m}$ are Lipschitz continuous vector fields.

Example 9 *Equation (28) may be used to represent the kinematic and dynamic models developed in the previous sections. The following table reports the expression of the quantities in (28) for manipulators and mobile robots.*

Model	x	u	$f_0(x)$	$f_1(x)$
Kinematic model of manipulators (2) $\dot{x}_e = J(q)\dot{q}$	x_e	$\dot{q}$	0	$(J \circ f^{-1})(x_e)$
Kinematic model of mobile robots (4) $\dot{q} = g(q)u$	q	u	0	$g(q)$
Kinematic model of manipulators (7) $D\ddot{q} + C\dot{q} + g = \tau$	$\begin{bmatrix} q \\ \dot{q} \end{bmatrix}$	τ	$\begin{bmatrix} \dot{q} \\ -D^{-1}C\dot{q} - D^{-1}g \end{bmatrix}$	$\begin{bmatrix} 0 \\ D^{-1} \end{bmatrix}$

Stability-like Tasks

In these lectures, we will consider two types of task models which are able to encompass a large variety of robotic tasks. The first class of tasks are *stability-like tasks*. These tasks consist in driving the state of the system x to a desired set $S \subset \mathbb{R}^n$.

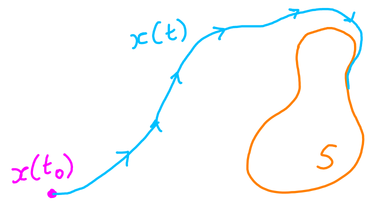


Figure 15: A stability-like task consists in driving the state of the robot, $x(t)$, from its initial condition $x(t_0)$ to a desired set S, as $t \to \infty$.

Examples of stability-like tasks include regulating the end effector of a manipulator to a desired pose, orienting the end effector towards a desired point of interest, tracking a trajectory in the task space. To formulate such tasks as constraints of an optimization problem, we will resort to an important control-theoretic technique called *control Lyapunov function*[4].

[4] Eduardo D Sontag. A lyapunov-like characterization of asymptotic controllability. *SIAM journal on control and optimization*, 21(3):462–471, 1983

Control Lyapunov Functions

Driving the state x of a system to a set S can be reworded as *making the set S asymptotically stable*, i. e.,—without going into the details of theory of stability—ensuring that, no matter what the initial value of the state is, eventually x is going to approach the set S.

Assume we would like to drive x to 0. As x follows the dynamics in (28), our goal becomes choosing u so that $x \to 0$. This, in general is a difficult problem since x is a n-dimensional vector, u is a m-dimensional vector, so we need to choose m values to drive n values to 0, and the two are related to each other by the nonlinear differential equation (28).

THE IDEA BEHIND CONTROL LYPAUNOV FUNCTIONS consists in finding a function $V : \mathbb{R}^n \to \mathbb{R}$, $V \in C^1$ and positive definite, such that $V \to 0$ as $x \to 0$. As the value of V is a scalar-valued function, we have greatly simplified the problem, since now we just need to drive one value to 0, i. e. the value $V(x)$. The dynamics of V can be found by applying chain rule as follows:

V is positive definite if $V(x) \geq 0$ for all $x \in \mathbb{R}^n$ and $V(x) = 0 \Leftrightarrow x = 0$,

$$\begin{aligned} \dot{V} &= \frac{\mathrm{d}V}{\mathrm{d}t} = \\ &= \frac{\partial V}{\partial x}\frac{\mathrm{d}x}{\mathrm{d}t} \\ &= \frac{\partial V}{\partial x}(f_0(x) + f_1(x)u) \\ &= \frac{\partial V}{\partial x}f_0(x) + \frac{\partial V}{\partial x}f_1(x)u, \end{aligned} \tag{29}$$

which is still affine in the control input u.

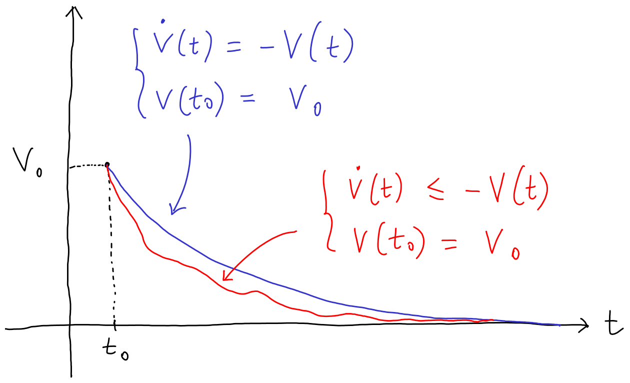


Figure 16: The comparison lemma can be used to go from the differential equation $\dot{V}(t) = -V(t)$—whose solution is the exponential function $V(t) = V_0 e^{-(t-t_0)} \to 0$ as $t \to \infty$—to the differential inequality $\dot{V}(t) \leq -V(t)$, whose solution converges to 0 as $t \to \infty$.

In order to drive V to 0, we can leverage the comparison lemma[5]

[5] Hassan K Khalil. *Nonlinear control*. Pearson New York, 2015

and enforce the following constraint (see Fig. 16):

$$\dot{V}(x,u) \leq -V(x). \tag{30}$$

If V satisfies the differential inequality (30), then $V(x(t)) \to 0$ as $t \to \infty$. Substituting the dynamics of V, (29), in (30), one obtains:

$$\frac{\partial V}{\partial x} f_0(x) + \frac{\partial V}{\partial x} f_1(x) u \leq -V(x), \tag{31}$$

which is an affine constraint of the robot control input u

V IS CALLED A CONTROL LYAPUNOV FUNCTION (CLF) for the system (28) if there exists a class $\mathcal{K}$ function α such that $\forall x \neq 0$

$$\inf_u \left\{ \frac{\partial V}{\partial x} f_0(x) + \frac{\partial V}{\partial x} f_1(x) u + \alpha(V(x)) \right\} \leq 0. \tag{32}$$

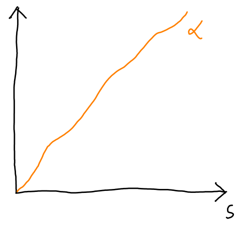


Figure 17: A class $\mathcal{K}$ function α is a continuous function such that $\alpha(0) = 0$ and α is strictly increasing.

If the condition in (32) is always satisfied, then we can always choose a u such that $\dot{V}(x,u) \leq -\alpha(V(x))$, which will result in $V \to 0$ as $t \to \infty$, i.e., it will result in the execution of the stability-like task where the set S to reach is given by $S = \{x \in \mathbb{R}^n : V(x) = 0\} = \{0\}$.

Min-norm Controller

The expression in (32) looks like a constraint that, given the value of x, we can evaluate and enforce on u to make V, and hence x, go to 0. The question that remains to answer now is: *How do we choose u?*

LET'S DO IT IN THE LAZIEST POSSIBLE WAY, i.e., by picking the u that has minimum squared norm. This idea leads to the following optimization problem to evaluate the robot controller u that drives x to 0:

$$\begin{aligned} &\underset{u}{\text{minimize}} \ \|u\|^2 \\ &\text{subject to } \frac{\partial V}{\partial x} f_0(x) + \frac{\partial V}{\partial x} f_1(x) u + \alpha(V(x)) \leq 0. \end{aligned} \tag{33}$$

Equation (33) is a convex quadratic program (QP) as the cost is quadratic in u and the constraint is affine in u. The controller $u^\star$ solution of (33) is known as the *min-norm controller*[6]. As the norm of the control input, $\|u\|$, is proportional to the energy utilized by the robot, the controller $u^\star$ is also known as the *minimum-energy controller*.

[6] Eduardo D Sontag. A 'universal' construction of Artstein's theorem on nonlinear stabilization. *Systems & control letters*, 13(2):117–123, 1989

Example 10 *Assume we have a robot with state x and dynamics $\dot{x} = u$. We would like to drive x to 0. In order to use the min-norm controller solution of (33), the first step is to define a suitable control Lyapunov function. Let us choose the following function:*

$\dot{x} = u$, known as *single integrator dynamics*, is a special case of a control affine system, where f_0 is the zero function and $f_1(x)$ is equal to an identity matrix of appropriate size for all x

$$V : \mathbb{R}^n \to \mathbb{R} : x \mapsto \|x\|^2, \tag{34}$$

which is differentiable and positive definite. The minimum-energy controller can be evaluated by solving the following optimization problem:

$$\begin{aligned} &\underset{u}{\text{minimize}}\ \|u\|^2 \\ &\text{subject to}\ \underbrace{2x^T u}_{\frac{\partial V}{\partial x} f_1(x) u} + \underbrace{\|x\|^2}_{\alpha(V(x))} \le 0. \end{aligned} \tag{35}$$

Here, we chose the class $\mathcal{K}$ function α to be the identity, i.e., $\alpha(s) = s$.

This is an inequality-constrained optimization problem like the one in (24) *and therefore can be solved by solving the system of equations and inequalities* (27). *In this simple case, the following closed form solution can be found:*

$$u^\star = \left\{ \begin{matrix} -\frac{x}{2} & \textit{if } x \neq 0 \\ 0 & \textit{otherwise} \end{matrix} \right\} = -\frac{x}{2}. \tag{36}$$

In general, when more tasks need to be executed, and therefore more constraints are enforced, one needs to resort to numerical algorithms to solve constrained optimization problems.

`Link to Google Colab for stability-like tasks of manipulators and mobile robots.`

Min-norm controllers – Part II: Invariance and Control Barrier Functions

In the previous section, we defined stability-like tasks as tasks that are executed by driving the state x of a dynamical system (28) to a desired set S. In this section, we introduce a second type of tasks which are executed by letting the state of the system *remain* within a desired set.

Invariance-like Tasks

The execution of *invariance-like tasks* consists in ensuring that the state x of a dynamical system (28) remains confined in a desired set S. In dynamical system theory, invariance—dual property of stability—can be interpreted as an example of *safety*, intended in the following sense: If a trajectory originates inside an invariant set, it will never reach the complement of the set, representing the *unsafe* region.

Examples of invariance-like tasks include avoiding joint limits of a manipulator, avoiding collisions with objects, other robots, humans in the robot environment, keeping enough energy in the battery of the robots. Analogously to the stability-like tasks, our objective is to encode invariance-like tasks as constraints of an optimization problem. In order to do so, we will leverage a control-theoretic technique known as *control barrier functions*[7].

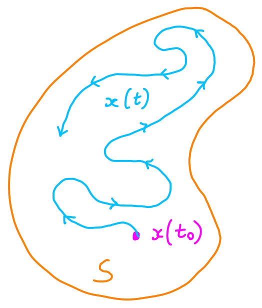


Figure 18: An invariance-like task consists in keeping the state of the robot, $x(t)$, within a desired set S, for all $t \geq t_0$.

[7] Aaron D Ames, Samuel Coogan, Magnus Egerstedt, Gennaro Notomista, Koushil Sreenath, and Paulo Tabuada. Control barrier functions: Theory and applications. In *2019 18th European control conference (ECC)*, pages 3420–3431. IEEE, 2019

Control Barrier Functions

To execute invariance-like tasks, we need to keep the state x of a system within a set S. This can be reworded as *making the set S (forward) invariant*, so that, if the state is inside the set S at the initial time t_0, it will remain inside S for all future times $t \geq t_0$, i. e.:

$$x(t_0) \in S \implies x(t) \in S \quad \forall t \geq t_0. \tag{37}$$

Forward invariant is used to specify the fact that we are interested in the invariance property for *future* times.

Assuming $x(t_0) \in S$, given the control affine dynamics (28) of x, our goal is to pick a control input u so that $x(t) \in S$ for all $t \geq t_0$. Just like stability, this is in general a difficult problem, and, just like stability, we will reduce it to a 1-dimensional problem. To do so, we make use of control barrier functions.

LET US START BY CONSIDERING a function $h : \mathbb{R}^n \to \mathbb{R}$, $h \in C^1$, whose 0-*superlevel set* is the set S to be rendered invariant, i. e., the set S can be expressed as follows:

$$S = \{x \in \mathbb{R}^n : h(x) \geq 0\}, \tag{38}$$

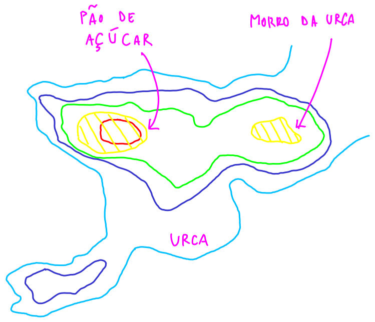


Figure 19: Crosshatched in yellow is the 200-superlevel set of the terrain elevation function (expressed in meters) in the neighborhood Urca in Rio de Janeiro.

so that

$$\begin{aligned} h > 0 &\implies x \in S^\circ \\ h = 0 &\implies x \in \partial S \\ h < 0 &\implies x \in S^C \end{aligned} \tag{39}$$

S°, ∂S and S^C denote the interior, the boundary, and the complement of the set S, respectively.

The relations in (39) suggest that, in order to keep $x \in S$, we just need to keep the (scalar) value of h to be nonnegative. In order to control the value of h, we need to know its dynamics. Proceeding similarly to (29), leveraging the dynamics of x, we express the dynamics of h using chain rule as follows:

$$\begin{aligned} \dot{h} = \frac{\mathrm{d}h}{\mathrm{d}t} &= \\ &= \frac{\partial h}{\partial x}\frac{\mathrm{d}x}{\mathrm{d}t} \\ &= \frac{\partial h}{\partial x}(f_0(x) + f_1(x)u) \\ &= \frac{\partial h}{\partial x}f_0(x) + \frac{\partial h}{\partial x}f_1(x)u. \end{aligned} \tag{40}$$

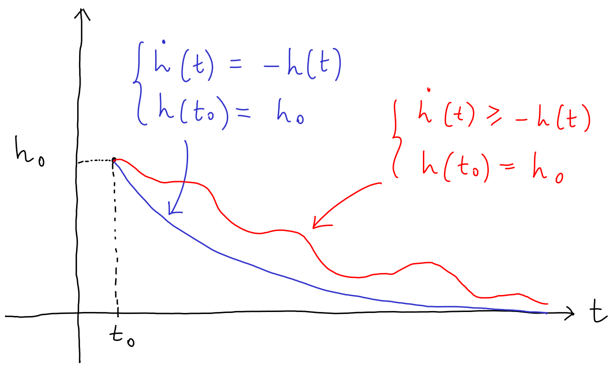


Figure 20: The comparison lemma can be used to go from the differential equation $\dot{h}(t) = -h(t)$—whose solution is the exponential function $h(t) = h_0 e^{-(t-t_0)} \to 0$ as $t \to \infty$—to the differential inequality $\dot{h}(t) \geq -h(t)$, whose solution is positive for all t provided that $h_0 > 0$.

Given these dynamics, and leveraging again the comparison lemma[8], we can enforce the following constraint (see Fig. 20):

$$\dot{h}(x, u) \geq -h(x). \tag{41}$$

[8] Hassan K Khalil. *Nonlinear control*. Pearson New York, 2015

If h satisfies the inequality constraint (41), one has that, if $h(x(t_0)) > 0$, then $h(x(t)) > 0$ for all $t \geq 0$. Substituting the dynamics of h, (40), in (41), we obtain the following affine constraint on the control input u which, if satisfied, ensures that the value of h remains positive, i. e., that the state x remains in the desired set S:

$$\frac{\partial h}{\partial x}f_0(x) + \frac{\partial h}{\partial x}f_1(x)u \geq -h(x). \tag{42}$$

h IS CALLED A CONTROL BARRIER FUNCTION (CBF) for the system (28) if there exists a class $\mathcal{K}$ function α such that $\forall x \in \mathbb{R}^n$

$$\sup_u \left\{ \frac{\partial h}{\partial x}f_0(x) + \frac{\partial h}{\partial x}f_1(x)u + \alpha(h(x)) \right\} \geq 0. \tag{43}$$

Min-norm Controller

For all values of x, the expression in (42) represent a constraint on u that, if enforced, ensures the forward invariance of the set S. Among all the values of u that satisfy this affine inequality constraint, we proceed as for stability-like tasks, and pick the control input which minimizes its norm. We can do so by solving the following convex QP:

$$\begin{aligned} &\underset{u}{\text{minimize}}\ \|u\|^2 \\ &\text{subject to}\ \frac{\partial h}{\partial x} f_0(x) + \frac{\partial h}{\partial x} f_1(x) u + \alpha(h(x)) \geq 0. \end{aligned} \tag{44}$$

The controller $u^\star$ solution of (44) is the *min-norm controller*, or the *minimum-energy controller*, which renders the set $S = \{x \in \mathbb{R}^n : h(x) \geq 0\}$ forward invariant.

Remark 3 *If the system model is purely kinematic, i.e. no dynamic effects like inertia and centrifugal forces are considered (as in the case of kinematic models of manipulators,* (2)*, and mobile robots,* (4)*), then the solution to* (44) *is simply $u^\star = 0$. This corresponds to the fact that, if $x \in S$ and we can instantaneously stop the motion of x by setting $u = 0$, then to remain within S, we just need x to not move. And $u = 0$ achieves that in the (globally) laziest possible way—by globally minimizing $\|u\|^2$.*

Combining Stability-like and Invariance-like Tasks

Remark 3 highlights the fact that, if we only have invariance-like tasks, we might get a robot that just does not move, does not do anything. This condition comes from the fact that the invariance constraint (42) is *not active*. More interesting behaviors, can be achieved by combining stability-like and invariance-like tasks.

Thanks to the constraint-based formulation, where tasks are encoded as constraints, we can simply combine multiple tasks by enforcing multiple constraints, as follows:

$$\begin{aligned} &\underset{u}{\text{minimize}}\ \|u\|^2 \\ &\text{subject to}\ \frac{\partial V}{\partial x} f_0(x) + \frac{\partial V}{\partial x} f_1(x) u + \alpha(V(x)) \leq 0 \\ &\qquad\qquad\ \frac{\partial h}{\partial x} f_0(x) + \frac{\partial h}{\partial x} f_1(x) u + \alpha(h(x)) \geq 0, \end{aligned} \tag{45}$$

where the first constraint, constructed using the control Lyapunov function V, encodes the stability-like task, and the second constraint, constructed using the control barrier function h, encodes the invariance-like task.

Remark 4 *Proceeding as in Example 10 to solve (45) might not lead to a closed-form solution $u^\star$ due to the multiple constraints.*

Remark 5 *When we have multiple constraints, we need to worry about the infeasibility of the optimization problem. If V is a CLF and h is a CBF, the two constraints alone can be satisfied, by definition of CLF and CBF. Nevertheless, there might be no u that satisfies both constraints at the same time.*

Remark 6 *The optimization problem (45) is still a convex QP and can be solved in a reliable and efficient way using numerical methods, which will either return the optimal solution $u^\star$ or certify that there does not exist any.*

Example 11 (Example 10 cont.) *Assume that, besides driving the state x of a single integrator robot to 0, we would also like it to avoid a ball-shaped unsafe region of radius d centered at x_o (see Fig. 21). The set S we would like to be invariant is given by the following expression:*

$$S = \{x \in \mathbb{R}^n : \|x - x_o\|^2 \geq d^2\}. \tag{46}$$

If $\bar{x} \in S$, then the distance between $\bar{x}$ and x_o is larger than d, i.e., $\bar{x}$ is not in the unsafe region. A function h whose 0-superlevel set is S is the following:

$$h : \mathbb{R}^n \to \mathbb{R} : x \mapsto \|x - x_o\|^2 - d^2, \tag{47}$$

so that if $\bar{x}$ is such that $h(\bar{x}) \geq 0$, then $\bar{x} \in S$.

Adding the CBF constraint (42) to (35) results in the following QP:

$$\begin{aligned} &\underset{u}{\text{minimize}}\ \|u\|^2 \\ &\text{subject to}\ 2x^T u + \|x\|^2 \leq 0 \\ &\qquad\qquad \underbrace{2(x - x_o)^T u}_{\frac{\partial h}{\partial x} f_1(x) u} + \underbrace{\|x - x_o\|^2 - d^2}_{\alpha(h(x))} \geq 0. \end{aligned} \tag{48}$$

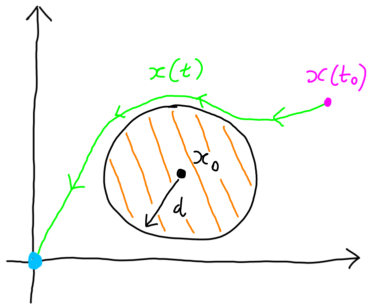


Figure 21: Combination of stability-like and invariance-like tasks. We would like x to reach the origin depicted as a blue dot, while avoiding the region crosshatched in orange.

The feasibility concerns highlighted in Remark 5 can be mitigated by relaxing the stability-like task by introducing a so-called slack variable $\delta \in \mathbb{R}$ in (48) as follows:

$$\begin{aligned} &\underset{u,\delta}{\text{minimize}}\ \|u\|^2 + \kappa\delta^2 \\ &\text{subject to}\ 2x^T u + \|x\|^2 \leq \delta \\ &\qquad\qquad 2(x - x_o)^T u + \|x - x_o\|^2 - d^2 \geq 0. \end{aligned} \tag{49}$$

The slack variable δ relaxes the stability-like constraint. The behavior resulting from the implementation of $u^\star$ solution of (49) consists in trading off the execution of the stability-like task for the execution of the invariance-like task, so that avoiding the unsafe region (invariance-like task) has higher priority with respect to driving x to 0 (stability-like task).

`Link to Google Colab for invariance-like tasks of manipulators and mobile robots.`

References

Aaron D Ames, Samuel Coogan, Magnus Egerstedt, Gennaro Notomista, Koushil Sreenath, and Paulo Tabuada. Control barrier functions: Theory and applications. In *2019 18th European control conference (ECC)*, pages 3420–3431. IEEE, 2019.

Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

Hassan K Khalil. *Nonlinear control*. Pearson New York, 2015.

Alessandro De Luca and Giuseppe Oriolo. Modelling and control of nonholonomic mechanical systems. In *Kinematics and dynamics of multi-body systems*, pages 277–342. Springer, 1995.

Richard M. Murray, Zexiang Li, and S. Shankar Sastry. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994.

Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, and Giuseppe Oriolo. *Robotics: modelling, planning and control*. Springer Science & Business Media, 2010.

Eduardo D Sontag. A lyapunov-like characterization of asymptotic controllability. *SIAM journal on control and optimization*, 21(3):462–471, 1983.

Eduardo D Sontag. A 'universal' construction of Artstein's theorem on nonlinear stabilization. *Systems & control letters*, 13(2):117–123, 1989.

Mark W. Spong, Seth Hutchinson, and Mathukumalli Vidyasagar. *Robot modeling and control*. John Wiley & Sons, 2020.