Politecnico di Milano

AUTOMATION AND CONTROL ENGINEERING

CONSTRAINED NUMERICAL OPTIMIZATION FOR ESTIMATION AND CONTROL

REPORT

Optimal Control Problem: optimization of the trajectory of a snake robot

Student Name	**Student ID**
Massimiliano Manenti	10564189
Riccardo Monaco	10559982
Alberto Ortolan	10609776

Symbol	Description
α	Parameter of the serpenoid curve (amplitude)
α_j	Optimization variable, parameter of the reference serpenoid curve (j-th time interval)
α_{min} (α_{max})	Minimum (maximum) value that the optimization variables α_j can be
α_{ref}	Parameter of the reference serpenoid curve (amplitude)
β	Parameter of the serpenoid curve (phase shift)
β_j	Optimization variable, parameter of the reference serpenoid curve (j-th time interval)
β_{min} (β_{max})	Minimum (maximum) value that the optimization variables β_j can be
β_{ref}	Parameter of the reference serpenoid curve (phase shift)
C	Matrix present in the linear inequality constraints
c_n (c_t)	Normal (tangential) viscous friction coefficient
d	Vector present in the linear inequality constraints
$f(v_{opt})$	Cost function of the optimization problem
$f_1(v_{opt})$	Addend of the cost function (goal position)
$F_1(v_{opt})$	Vector composing $f_1(v_{opt})$
$f_2(v_{opt})$	Addend of the cost function (goal position, last interval)
$F_2(v_{opt})$	Vector composing $f_2(v_{opt})$
$f_3(v_{opt})$	Addend of the cost function (velocities)
$F_3(v_{opt})$	Vector composing $f_3(v_{opt})$
f_i	Friction force acting on the *i-th* link
f_{i_n} (f_{i_t})	Normal (tangential) component of f_i
γ	Parameter of the serpenoid curve (offset)
γ_j	Optimization variable, parameter of the reference serpenoid curve (j-th time interval)
γ_{min} (γ_{max})	Minimum (maximum) value that the optimization variables γ_j can be
γ_{ref}	Parameter of the reference serpenoid curve (offset)
$h(v_{opt})$	nonlinear inequality constraints
J	Moment of inertia of each link
k_P	Proportional controller gain
k_D	Derivative controller gain
l	Half length of a link
$last\_interval$	Time step in which the last interval starts
m	Mass of each link
n	Number of links composing the snake robot
$Nsim$	Last time step of the simulation
n_{vary}	Number of intervals in which the simulation is divided (opt. var. are constant in each interval)
ω	Parameter of the serpenoid curve (frequency)
ω_j	Optimization variable, parameter of the reference serpenoid curve (j-th time interval)
ω_{min} (ω_{max})	Minimum (maximum) value that the optimization variables ω_j can be
ω_{ref}	Parameter of the reference serpenoid curve (frequency)
ϕ_i	Angle between the *i+1-th* link and the *i-th* link ($\phi_{i,t}$ when emphasizing the time dependence)
ϕ_{ref_i}	Reference signal related to angle ϕ_i
p_x (p_y)	Coordinates of the COM of the whole robot ($p_{x,t}$ ($p_{y,t}$) when emphasizing the time dependence)
q (Q)	Weight related to $f_1(v_{opt})$ ($F_1(v_{opt})$)
q_f (Q_f)	Weight related to $f_2(v_{opt})$ ($F_1(v_{opt})$)
q_v (Q_v)	Weight related to $f_3(v_{opt})$ ($F_1(v_{opt})$)
Q_{norm}	Normalization term related to $f_1(v_{opt})$
Q_{f_norm}	Normalization term related to $f_2(v_{opt})$
Q_{v_norm}	Normalization term related to $f_3(v_{opt})$
σ	Small and positive parameter used in Gauss-Newton
$step_{phi}$	Number of time steps; every $step_{phi}$ steps the ϕ_i angles are constrained
θ_i	Global orientation of the *i-th* link
$\bar{u}$	New set of control inputs (after partial feedback linearization)
v_i	Speed of the center of mass of the *i-th* link
v_{opt}	Vector of optimization variables (parameters of the reference serpenoid curve in each interval)
$\tilde{x}$	State vector of the model
$(x_i,\ y_i)$	Coordinates of the center of mass of the *i-th* link
$\tilde{y}$	Output vector of the model

1 Introduction

Research on snake robots has been conducted for several decades. The development and control of such robots, especially wheeless, is generally quite challenging for two reasons. First of all, a snake robot has many degrees of freedom, which means that the physical mechanism will be composed of a complex interconnection of sensors, actuators, and control logic. Moreover, the many degrees of freedom are related to a complex nonlinear dynamic.
In second place, the interaction with the environment is more complicated for a snake robot than for other types of mobile robots. In fact, propulsion requires synchronised motion of the entire body in order to produce appropriate traction forces.
However, even though the motion is challenging, nowadays a lot of effort has been done worldwide and literature about both control and mechanical design has been developed and studied. The reason why this happened is related to the benefits of a multi degrees of freedom wheeless robot, which essentially consist in the ability to move successfully in bumpy terrains and restrictive environments, such as the ones present in post earthquake rescue operations or car-accidents assistance in tunnels.
The focus of this project is the exploitation of simulation and optimization techniques applied to a dynamical model of a snake robot, in order to make it reach a goal position in a 2D environment. The study is motivated by the appeal of such systems and relative applications.

2 Problem description

The project aims at finding the optimal reference signals that, given to the snake robot system, make it move from a starting point to a desired goal position. Needless to say, the definition of "optimal reference signals" is strictly bounded to the optimization problem definition.
The task can be classified as an Optimal Control Problem: once the reference signals are found through the optimization routine, since the control law is known and the measurements are available, one can easily compute the control inputs that should bring the snake robot to the goal position.
The following assumptions are taken into account:

1. Given the complexity of the robot dynamics, a 2D model is studied and implemented.
2. Among the different motion patterns employable by a snake robot, lateral undulation is the type of snake locomotion that has been considered. In order to generate propulsion, this kind of motion requires anisotropic friction, which is assumed to be present.
3. A viscous dynamical friction model was employed, in order to avoid excessive mathematical complexity.
4. Both the starting point and the goal refer to the center of mass of the snake.
5. The actuation has been considered ideal, in the sense that actuators do not have power nor technical limitations.
6. All states are assumed measurable.
7. It is assumed that no disturbance is acting on the system.

3 Model

3.1 Kinematic model

When talking about a snake robot, it is intended a modular robot that consists of n rigid links, each of length $2l$, interconnected by n-1 motorized joints. The following picture shows a graphical representation of it:

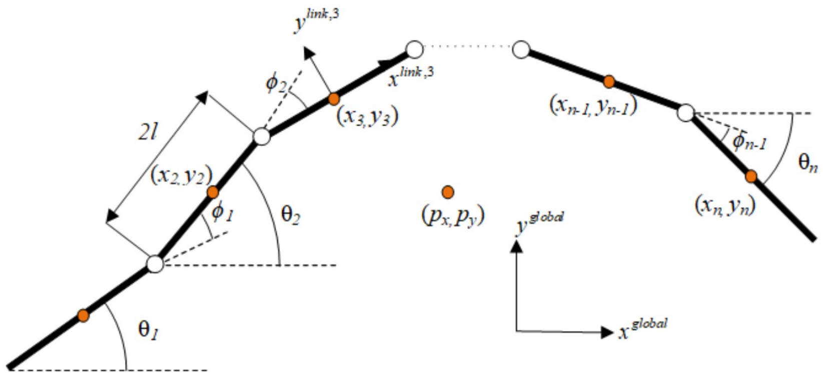

Figure 1: 2D graphical representation of a snake robot

Here, variables and elements of the model are introduced and explained.
$(x_i,\ y_i)$ describe the global coordinates of the COM (centre of mass) of link i, while $(p_x,\ p_y)$ are the global coordinates of the COM of the entire robot. θ_i represents the orientation of the *i-th* link, whereas ϕ_i is the angle between the *i+1-th* link and the *i-th* link and can be expressed as:

$$\phi_i = \theta_i - \theta_{i+1}$$

Moreover, the actuators are placed in the joints.
Both the link angles and the joint angles can be expressed in the vector form as $\theta = [\theta_1, ..., \theta_n]^T$ and $\phi = [\phi_1, ..., \phi_{n-1}]^T$.

3.2 Friction model

A planar snake robot achieves forward propulsion on a flat surface by generating ground friction forces that propel it forward. The ground friction model is therefore a key part of the dynamics of the snake robot.
In this project, anysotropic viscous friction is considered, so the friction force f_i acting on a link i can be computed as (making explicit the tangential and normal components):

$$\begin{bmatrix} f_{i\_t} \\ f_{i\_n} \end{bmatrix} = -\begin{bmatrix} c_t & 0 \\ 0 & c_n \end{bmatrix} v_i$$

with v_i being the speed of the center of mass of link i and c$_t$ and c$_n$ respectively the tangential and normal viscous friction coefficient, with c$_t \neq c\ _n$ (assumption number 2).

3.3 Dynamical model

In this section, the final expression of the dynamical model is reported. The mathematical reasoning followed to obtain the final equations will not be mentioned in this report due to the complexity of the model, however it can be found in Chapter 2 of [1] . Still, it is important to mention that the model presented here is the result of a partial feedback linearization of the system. The model takes as input the ideal actuation, $\bar{u}$, and returns as outputs the entire state, so all angles and position of the center of mass and respective derivatives.

$\bar{u} = [\ddot{\phi}_1, ..., \ddot{\phi}_{n-1}]^T$
$\tilde{x} = [\phi_1, ..., \phi_{n-1}, \theta_n, p_x, p_y, \dot{\phi}_1, ..., \dot{\phi}_{n-1}, \dot{\theta}_n, \dot{p}_x, \dot{p}_y]^T$
$\tilde{y} = \tilde{x}$

The dynamical model equations can be, therefore, expressed as:

$$\dot{\tilde{x}} = \begin{bmatrix} \dot{\tilde{x}}_1 \\ \dot{\tilde{x}}_2 \\ \dot{\tilde{x}}_3 \\ \dot{\tilde{x}}_4 \end{bmatrix} = \begin{bmatrix} \tilde{x}_3 \\ \tilde{x}_4 \\ \bar{u} \\ A(\tilde{x}) + B(\tilde{x}_1)\bar{u} \end{bmatrix}$$

Where:

$\tilde{x}_1 = [\phi_1, ..., \phi_{n-1}]^T$, $\tilde{x}_2 = [\theta_n, p_x, p_y]^T$, $\tilde{x}_3 = [\dot{\phi}_1, ..., \dot{\phi}_{n-1}]^T$, $\tilde{x}_4 = [\dot{p}_x, \dot{p}_y]^T$

$A(\tilde{x})$ and $B(\tilde{x}_1)$ are suitable matrices which structure can be seen in Chapter 2 of [1].

3.4 Control system

Once the dynamical model has been introduced, the focus moves on the control scheme. This consists of a closed loop scheme in which the joint angles and joint velocities are given as feedback and a partial feedback linearization is performed. Feedback linearization is needed in order to deal with the non linearities that are present in the system.

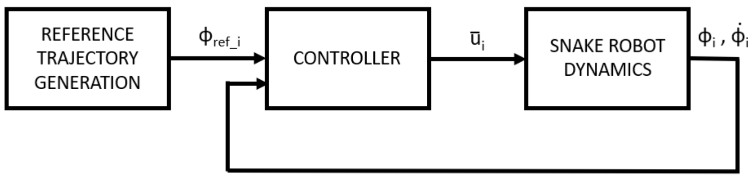

Figure 2: The three main blocks of the control scheme

In order to propel the snake robot forward, it has been shown in literature [2] that the joints should follow time signals in the form:

$$\phi_i = \alpha sin(\omega t + (i-1)\beta) + \gamma \qquad i = \{1, ..., n-1\}$$

This function is known as "serpenoid curve". α and ω are the amplitude and frequency, respectively, of the sinusoidal joint motion, β determines the phase shift between the joints and γ is the joint offset and used to control the direction of the locomotion. i is the index referring to the $i-th$ joint, whereas t represents time.

For this reason, the reference signal is in the form:

$$\phi_{ref\_i} = \alpha_{ref} sin(\omega_{ref} t + (i-1)\beta_{ref}) + \gamma_{ref} \qquad i = \{1, ..., n-1\}$$

The objective of the optimization routine will be to tune the parameters of the reference serpenoid curve in order to obtain a good control on the motion of the snake robot.

The control law for link i is given by:

$$\bar{u}_i = k_P(\phi_{ref\_i} - \phi_i) - k_D(\dot{\phi}_{ref\_i} - \dot{\phi}_i) + \ddot{\phi}_{ref\_i}$$

where k_P and k_D are controller gains (both > 0). This control action has been chosen for the exponential stability of its error dynamics (see Section 4 of [1]).

4 Optimization problem

4.1 Problem abstraction

As said in the previous sections, the purpose of the optimization routine is to tune the parameters of the reference serpenoid curve in order to force the robot to reach the goal position minimizing a certain cost function and trying to satisfy some constraints. The overall problem takes the form of:

$$\begin{aligned} \min_{v_{opt}} \quad & f(v_{opt}) \\ \text{s.t.} \quad & \textit{dynamic (including controller)} \\ & Cv_{opt} - d \geq 0 \\ & h(v_{opt}) \geq 0 \end{aligned} \tag{1}$$

In this section, the various elements are presented and explained.
A SQP method is used to solve the optimization problem (more datails in Section [4.2]).

4.1.1 Optimization variables

The chosen optimization variables are the parameters of the reference serpenoid curve: $\alpha_j, \omega_j, \beta_j$ and γ_j, the index j underlines the fact that these parameters vary with time. In order to be able to perform complex trajectories, it would be preferable to allow the parameters to change at each time step, however it would be too complex of a problem to solve due to the huge number of optimization variables. Therefore, it has been chosen to divide the entire simulation time into *n_vary* intervals and force the optimization variables to stay constant for the entire duration of each interval (the index j refers to the *j-th* interval). *n_vary* is a designer-chosen parameter and the total number of optimization variables is 4x*n_vary*.
Relative to the duration of each time interval, it was thought that the snake robot movement would be of three types during the simulation: initial adjustment, forward movement and final stopping maneuvers. The first interval could be "delicate", in the sense that the snake robot could try to assume critical configurations trying to self-align with the goal position. The other part of the simulation that could be critical is the last one, since the snake robot would try to stop (set the linear velocity of the COM to zero) and reach the final position (COM coinciding with the goal position). Whereas the central part of the simulation is thought to be less critical, since it is assumed that the snake robot would try to go straight.
For the above reasoning, in the script there is the possibility to change the length of the first and last interval (changing $first\_interval$ and $last\_interval$ in the $main\_snake$ script, which are respectively the time step in which the first interval ends and the time step in which the last interval starts), whereas the intervals in between divide the remaining time in constant length intervals (note that if one needs more design freedom in the central part of the simulation, it is sufficient to increase n_vary).
In this way, by changing n_vary, $Tend$ (duration of the simulation), $first\_interval$ and $last\_interval$, the user of the algorithm has many tools to deal with the number of intervals and their length.

The optimization variables vector looks like:

$$v_{opt} = \begin{bmatrix}\alpha_1 & \omega_1 & \beta_1 & \gamma_1 & \dots & \alpha_{n\_vary} & \omega_{n\_vary} & \beta_{n\_vary} & \gamma_{n\_vary}\end{bmatrix}^T \qquad v_{opt} \in \Re^{4n\_vary}$$

4.1.2 Cost function

The chosen cost function $f(v_{opt}) : \Re^{4n\_vary} \to \Re$ is characterized by three main objectives (each one defined on the same domain and codomain of $f(v_{opt})$):

$$f(v_{opt}) = f_1(v_{opt}) + f_2(v_{opt}) + f_3(v_{opt})$$

$$f_1(v_{opt}) = \sum_{t=1}^{Nsim-1} q^2 \cdot ((x_{goal} - p_{x,t})^2 + (y_{goal} - p_{y,t})^2) = F_1(v_{opt})' \cdot Q^2 \cdot F_1(v_{opt})$$

$$f_2(v_{opt}) = q_f^2((x_{goal} - p_{x,Nsim})^2 + (y_{goal} - p_{y,Nsim})^2) = F_2(v_{opt})' Q_f^2 F_2(v_{opt})$$

$$f_3(v_{opt}) = \sum_{t=last\_interval+1}^{Nsim} q_v^2(\dot{p}_{x,t}^2 + \dot{p}_{y,t}^2) = F_3(v_{opt})' Q_v^2 F_3(v_{opt})$$

$$F_1(v_{opt}) = \begin{bmatrix} x_{goal} - p_{x,1} \\ \cdots \\ x_{goal} - p_{x,Nsim-1} \\ y_{goal} - p_{y,1} \\ \cdots \\ y_{goal} - p_{y,Nsim-1} \end{bmatrix} \quad F_2(v_{opt}) = \begin{bmatrix} x_{goal} - p_{x,Nsim} \\ y_{goal} - p_{y,Nsim} \end{bmatrix} \quad F_3(v_{opt}) = \begin{bmatrix} -\dot{p}_{x,last\_interval+1} \\ \cdots \\ -\dot{p}_{x,Nsim} \\ -\dot{p}_{y,last\_interval+1} \\ \cdots \\ -\dot{p}_{y,Nsim} \end{bmatrix}$$

- $q,\ q_f,\ q_v \in \Re$, $Q = diag(q) \in \Re^{2(Nsim-1)\times 2(Nsim-1)}$, $Q_f = diag(q_f) \in \Re^{2\times 2}$, $Q_v = diag(q_v) \in \Re^{2(Nsim-last\_interval)\times 2(Nsim-last\_interval)}$. The domains and codomains of $F_1(v_{opt})$, $F_2(v_{opt})$, $F_3(v_{opt})$ can be easily found, according to the other terms present in the optimization problem. *diag*() represents the diagonal operator (it creates a square matrix composed of zeros, with the exception of the elements on the diagonal, which take the value given as input).
- *Nsim* represents the number of time steps of the simulation.
  last_interval represents the time step in which the last interval starts.
- $f_1(v_{opt})$ is directly related to the weighted distance of the center of mass of the snake from the goal in each time instant, except for the last one.
- $f_2(v_{opt})$ is directly related to the weighted distance of the center of mass of the snake to the goal in the last time instant.
- $f_3(v_{opt})$ is directly related to the weighted linear velocities of the center of mass of the robot. The velocities considered are the ones of the last interval, so that the robot remains still in that period of time, hopefully once it has reached the goal.
- Since we have assumed to have an ideal control action, the control action is not weighted in the cost function.

Without considering the weights Q, Q_f and Q_v, the various terms of the cost function belong to different order of magnitudes. For example $f_1(v_{opt})$ accounts for all the time steps, with the exception of the last one, which is the only time step considered in $f_2(v_{opt})$ (one can clearly see that the order of magnitude of $f_1(v_{opt})$ will be greater than the one of $f_2(v_{opt})$). For this reason the cost contributions $f_1(v_{opt})$, $f_2(v_{opt})$, $f_3(v_{opt})$ should be normalized. The normalization is considered in the coefficients Q, Q_f, Q_v. Correctly tuning Q, Q_f and Q_v is important since the cost function presents more than one objective, the user should be able to decide how much relevant

each component is in the overall cost.
The form of a generic weight is:

$$Q_{cost} = \frac{Q_{weight}}{Q_{normalization}}$$

Qcost is the final weight that multiplies the single element of the cost function.
Qweight represents the design parameter introduced to manage the presence of more than one objective.
Qnormalization is the normalization weight that should bring the terms of the cost function to closer orders of magnitude.

The three *Qnormalization* used in the script aim at bringing the values of every component of the cost function inside the range $(0, 1]$, . Their values are:

$$related\ to\ Q\ (i.e.,\ f_1(v_{opt})):\ Q_{norm} = (N_{sim} - 1)(x^2_{goal} + y^2_{goal})$$

$$related\ to\ Q_f\ (i.e.,\ f_2(v_{opt})):\ Q_{f\_norm} = (x^2_{goal} + y^2_{goal})$$

$$related\ to\ Q_v\ (i.e.,\ f_3(v_{opt})):\ Q_{v\_norm} = (N_{sim} - last\_interval)2v^2_{max}$$

Qnorm represents the cost given by $f_1(v_{opt})$ when Q is set to 1 and the snake robot stays in the origin for the whole simulation. *Qf_norm* has a similar meaning, but it is related to $f_2(v_{opt})$ and accounts only for the last time instant. *Qv_norm* considers the case in which the snake robot is moving at the maximum velocity v_{max} for the time interval in which $f_3(v_{opt})$ acts (note that v_{max} is tuned relying on literature [1] and on simulations) .

During the development of the optimization problem, other terms in the cost function have been considered but eventually they were abandoned for different reasons. For example, it has been removed a weight on the α_j of the last interval. Its goal was to favour the stop of the robot at the end of the simulation, trying to give a constant reference to the joint angles (see Section 3.4). Clearly this idea is dominated by the cost term $f_3(v_{opt})$.
Another term that was present in the cost function tried to avoid the robot self-intersection, weighting the $\phi_{i,t}$ angles different from 0. It has been decided to insert a hard constraint in order to deal with this problem.

4.1.3 Constraints

Three main types of constraints were added to the optimization problem:

1. Dynamics of the system. The dynamic constraint is enforced in the simulation (considering also the controller) and won't be furtherly analyzed.
2. Saturation constraints on the optimization variables, so on the serpenoid curve parameters ($Cv_{opt} - d \geq 0$).
3. Anti-crouch constraints ($h(v_{opt}) \geq 0$).

The second type of constraints are linear constraints. They serve to force the optimization variables to assume values that make sense from a physical point of view. For example, it is inconvenient to have $\beta = 0$ because it drifts away from the original idea of snake imitating motion pattern (the snake is moving, but not slithering forward). The limits considered come from literature and our experience in the development of the project.

The linear constraints are in the form:

$$\alpha_{min} \leq \alpha_j \leq \alpha_{max}$$
$$\omega_{min} \leq \omega_j \leq \omega_{max}$$
$$\beta_{min} \leq \beta_j \leq \beta_{max}$$

$$\gamma_{min} \leq \gamma_j \leq \gamma_{max}$$
$$with\ j = 1, 2, ..., n\_vary$$

note that they can be cast in a linear matrix inequality as:

$$Cv_{opt} - d \geq 0$$

with $C \in \Re^{8n\_vary \times 4n\_vary}$ $d \in \Re^{8n\_vary \times 1}$ and:

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & ... \\ 0 & 1 & 0 & 0 & ... \\ 0 & 0 & 1 & 0 & ... \\ 0 & 0 & 0 & 0 & ... \\ .. & & & & \\ -1 & 0 & 0 & 0 & ... \\ 0 & -1 & 0 & 0 & ... \\ 0 & 0 & -1 & 0 & ... \\ 0 & 0 & 0 & 0 & ... \\ .. & & & & \end{bmatrix} \quad d = \begin{bmatrix} \alpha_{min} \\ \omega_{min} \\ \beta_{min} \\ 0 \\ .. \\ -\alpha_{max} \\ -\omega_{max} \\ -\beta_{max} \\ 0 \\ .. \end{bmatrix}$$

In the implementation, the optimization variables related to γ_j were not constrained, for this reason the terms in C that multiply γ_j are set to 0 (they give origin to a $0 = 0$ equation).

The third type of constraints is instead nonlinear. In particular, these constraints are called anti-crouch because they are used to prevent the robot from self-hitting and from self-coiling up. The idea is that, if all ϕ_i angles can never go beyond a certain ϕ_{max}, then the snake won't be able to self-hit. The value of ϕ_{max} is chosen as the value for which the snake forms a regular polygon (setting all the $\phi_i = \phi_{max}$). For example, in the image below ϕ_{max} is equal to 72° (setting $\phi_1 = \phi_2 = \phi_3 = \phi_4 = 72°$ results in a pentagon, with the head point H and tail point T coinciding):

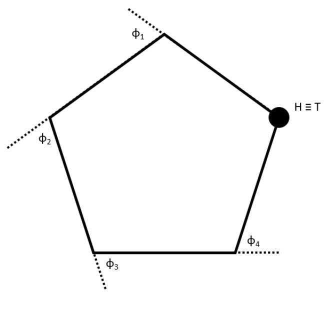

Figure 3: Example of a forbidden configuration of the robot

The anti-crouch constraints are in the form:

$$\phi_{min} < \phi_{i,t} < \phi_{max} \quad with\ i = 1, 2, ..., n-1 \ \ and \ \ t = 1, 2, ..., N_{sim}$$

which can easily be enforced using a nonlinear function $h(v_{opt})$ in the simulation phase.
Since the number of constraints could be high (the number of linear constraints is $8 \times n\_vary$ and the number of non linear constraints is $(n-1) \times Nsim$) and the constraint on $\phi_{i,t}$ is very conservative (the head and the tail of the robot touch if at the same time instant t, the various $\phi_{i,t}$ are all at their maximum value or all at their minimum value), it has been decided to introduce in the MATLAB script the possibility to constrain the $\phi_{i,t}$ angles, not at every time instant, but every $step_{phi}$ steps.
Moreover, the $step_{phi}$ value is not fixed, but it takes three decidable values: one for the initial adjustment, one for the forward movement and one for the final stopping maneuvers. Following

the reasoning of Section 4.1.1, a low value for the first and last $step_{phi}$ is suggested.

4.2 Numerical optimization

In this paragraph, some technical information about the optimization methodologies adopted in this project, are reported.
The optimization routine used is based on a discrete-time simulation of the snake behaviour which is obtained from the continuous-time model differential equations applying Euler-forward discretization, so:

$$\frac{dz}{dt} \simeq \frac{z(t+1)-z(t)}{T_s}$$

Which allows, starting from the continuous-time model and the state at time t $(z(t))$, to compute the state at next time instant.

$$z_{t+1} = z_t + T_s \dot{z}$$

The optimization routine is based on a SQP algorithm, where both Gauss Newton and BFGS can be used to compute the estimation of the Hessian at each iteration.
Note that in order to guarantee the Hessian to be positive definite in GN, a small and positive parameter σ was added on every element of the diagonal of the Hessian,
The computation of the derivatives necessary to make the algorithm work was carried out with Forward Finite Difference or Central Finite Difference. This choice was taken because the model was very complex and it would have been useful to resort to an automatic differentiation method, avoiding the effort of analytically computing all the derivatives.

5 Results

5.1 Preliminary considerations

Before starting discussing the results, some considerations are given:

- It must be noted that the running time of the optimization procedure is quite high. In order to reduce it, codegen has been employed.
- Sometimes the computation returns *"all constraints are satisfied"* even though actually some $\phi_{i,t}$ exceed the ϕ_{min} (or ϕ_{max}) values. The reason why this happens is the fact that $step_{phi}$ is not always equal to 1, so $\phi_{i,t}$ are not constrained at every time instant t (see Section 4.1.3). However, it must be remembered that these are conservative constraints, so the violation of the upper or lower bound doesn't necessarily imply the robot self-hitting. Therefore, the real acceptability of the solution can be accessed via visual inspection in the animation or by checking if the constraint is violated by <u>all</u> the $\phi_{i,t}$ at the <u>same time</u>.
- The solution is related closely to the simulation time selected by the user. A simulation time much bigger than the one needed causes the robot to choose a wider and smoother path. An even bigger simulation time makes the robot spend the last intervals available for adjusting its COM to the goal, however this behaviour is unnecessary. On the contrary, a too small simulation time results in the robot not being able to reach the goal.
- The simulations in the report have been performed with Central Difference as derivation method and with Gauss-Newton as Hessian computation method. There was not a dominant method between FD and CD, neither between GN and BFGS (considering cost function value and efficiency).

5.2 Weights discussion

Looking at the weights of the cost function terms:

- if q_v is zero, the robot is not able to decelerate before reaching the goal. If instead its value is too high, the robot does not successfully reach the goal;
- if q_f is zero, the robot does not reach the goal. In particular, it is preferable to have q_f bigger than both q_v and q;
- if q is zero, the robot is still able to reach the goal but its behavior is quite aggressive, so the performance is not satisfactory.

Fixing q and searching for Pareto optimal q_f and q_v, the weights have been tuned.

5.3 Example tests

In the following, an example test is showed in order to present the performance of the optimization routine. The example test here reported is characterized by the following parameters: goal (3,5) m, simulation time: 11s, sampling time: 0.1s, max number of iterations: 300, number of iterations performed: 300, initial guesses: $\alpha_0 = \pi/12 \ rad$, $\beta_0 = 10 \ rad$, $\omega_0 = \pi/16 \ rad/s$, $\gamma_0 = 0 \ rad$, weights: $q^2 = 0.1$ and $q_f^2 = 0.89$. $q_v^2 = 0.01$.
The COM trajectory and absolute value of the velocity of the COM is here reported:.

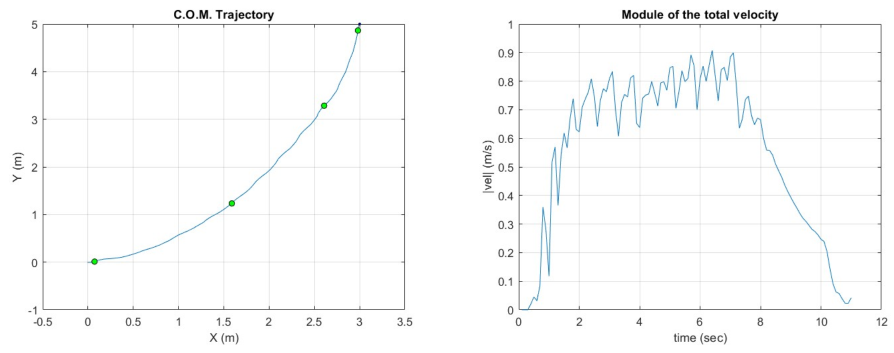

Figure 4: Trajectory of COM (left) and velocity of center of mass (right):

Looking at COM trajectory plot, it can be noticed that the center of mass reaches exactly the goal position (3,5) m. Moreover, in the plot the green points correspond to a change of the optimization variables value (4 changes in total). Note that the v_{max} chosen in the script as the normalization term for Q_{v_norm} is $1m/s$, close to the maximum value of the velocity in the plot (see Section [4.1.2] for further explanation).

Concerning the optimization variables (Figure 5), their plots show that they remain inside the limits expressed by the saturation constraints. An important thing to note is that the value of α decreases at the end of the simulation. This behaviour is related to the fact that the amplitude of the serpenoid curve is given by α (Section 3.4). A lower α at the end of the simulation usually implies lower velocities near the goal position.

The plots of the first three ϕ_i are reported in Figure 6. They are showed just as specimen of the behaviour of all these variables.

It can be seen that the reference values of ϕ_i are not perfectly tracked. This is caused by an imperfect tracking action of the controller. Notice that a perfect controller is not needed: the reference signals could be impossible to track, but the optimization routine gives as output those reference signals since the system would act optimally when the controller tries to track them.

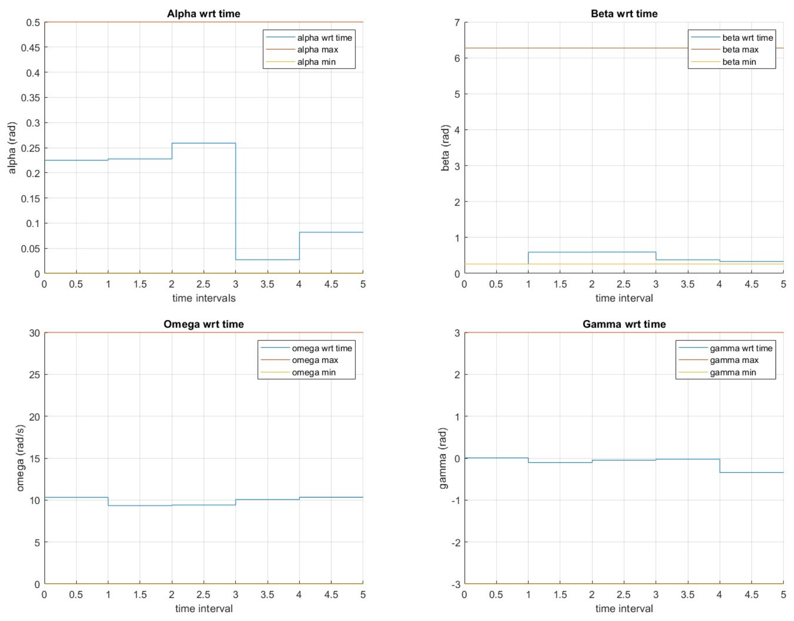

Figure 5: Plots of optimization variables with respect to their limits

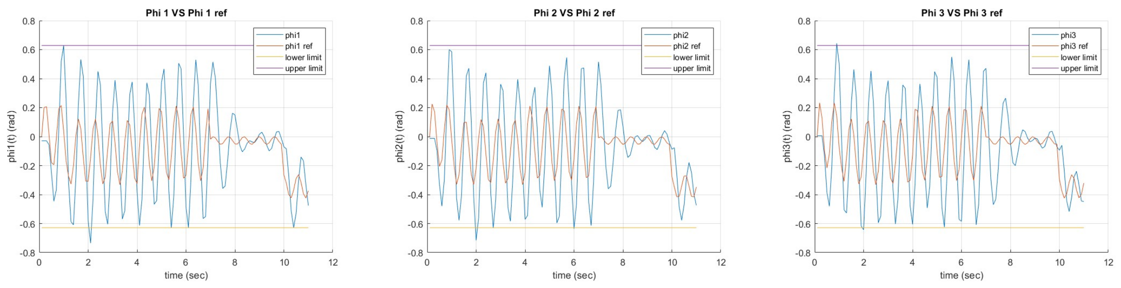

Figure 6: Plots of ϕ_1 (left) and ϕ_2 (center), ϕ_3 (right)

As previously mentioned, one of the main assumptions was the ideal actuation. For completeness, it is shown also the behaviour of at least a few components of u (Figure 7), even though the amplitude of the signals and their rate of change are not limited. Important is to notice that the plotted u is not the one considered as the input for the dynamical model, but rather that one together with the action necessary to linearize the model following the partial feedback linearization theory. In other words, it represents the actual request of control action that the regulator would demand to the actuator.

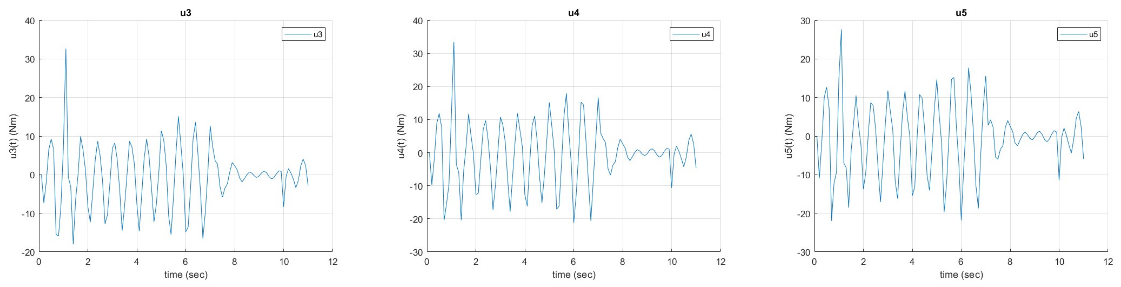

Figure 7: Plots of three control actions

Other simulations have been performed, with different goal positions (even behind the robot). It has always been found values of the parameters for which the robot was able to reach the goal.

5.4 Initial condition problem

The Optimal Control Problem can present different solutions (even with the same goal) based on the initial guesses used to initialize the algorithm. The user can reduce the dependency from these initial guesses by modifying the *for* loop at the beginning of the *main_snake* file (e.g. in the script with ”for initial_conditions_index = 1:1:15 [...] end” 15 initial guesses are made). Note that the first initial guess is always user chosen (in this way, if one has an initial guess that is working well, he can use it).
Basically, each loop will solve the optimization problem starting from a different initial condition and the program returns the solution characterized by the smallest cost. In doing so, however, the script becomes harder to manage computationally and the simulation time increases dramatically.

6 Conclusions

The project aims to solve an Optimal Control Problem consisting in the optimization of a snake robot trajectory in a 2D environment. Starting from a dynamical model of the snake robot and a controller, the problem has been solved by optimizing the parameters of the serpenoid curve (given as reference signal to the model). These parameters are time-varying but piece-wise constant. The problem is characterized by a multi-objective cost function, which weights has been tuned using a Pareto Optimal approach. The robot successfully reaches the goal in each test, following a suitable trajectory.
As a future step, the actuator dynamic could be considered, inserting limits and/or terms in the cost function related to it. That being the case, a more realistic problem would be faced.

7 References

[1] Liljebäck, Pål, ”Snake robots: modelling, mechatronics, and control” London: Springer, 2013.

[2] Hirose, Shigeo. ”Biologically inspired robots. Snake-Like Locomotors and Manipulators” 1993.

[3] Yesim A. Baysal, Ismail H. Altas, ”Modelling and Simulation of a Wheel-Less Snake Robot”, 2020