# Underactuated Snake Robot in 2D: Dynamics, Locomotion Strategies and Path Tracking

Carlo Barone and Marco Musto

Unversità degli Studi di Napoli Federico II

## INTRODUCTION

The project focuses on the modeling and control of a snake robot, implemented and simulated using MATLAB and Simulink. The choice of this topic stems from the unique characteristics and versatility of snake robots, which make them both scientifically intriguing .
Snake robots are a class of bio-inspired systems that exhibit highly flexible and adaptive locomotion. They can be considered both as terrestrial robots and as underwater vehicles (UUVs), depending on their intended application. This duality enhances their appeal and positions them as valuable tools in diverse environments, from navigating narrow pipelines to exploring terrains.
What makes snake robots particularly fascinating is their distinct mode of locomotion.As mentioned during a lecture in the course, they exploit body undulations and surface friction to move a mechanism that sets them apart. In this sense, their behavior shares conceptual similarities with unicycle robots in terms of underactuation and control challenges, and with legged robots in how friction is harnessed for propulsion.
Motivated by these considerations, we explored current techniques for modeling and controlling snake-like robots. Our goal was to apply and extend the theoretical background acquired in the course, particularly concerning the study of underactuated and redundant systems.
The project is structured as follows:

- **Model of the Snake Robot**, including its interaction with the environment,
- **Control Scheme** adopted for joint regulation and path following,
- **Snake Locomotion Strategies**, inspired by biological and mathematical models,
- **Path Following Strategy**, based on adaptive guidance principles,
- **Simulation and Results**, showing the behavior of the robot under different control and locomotion configurations.

The work has been carried out collaboratively by the authors, Marco and Carlo. Specifically:

- **Marco** was responsible for the *control scheme design* and the development of *locomotion strategies*,

- **Carlo** focused on the *modeling and dynamics* of the snake robot, and the implementation of the *path following approach*,
- The *simulation and results* section was developed jointly.

Both authors have full knowledge of all components of the project, and every part was discussed and refined collaboratively throughout the development process.

# Model of a Snake Robot

The class of snake robots considered in this work comprises wheelless snake robots, whose locomotion relies exclusively on the interaction forces between their body and the environment. Depending on the operational context, the robot may operate in confined environments, such as pipelines or in unstructured open spaces. In this chapter, we present a dynamic model that captures the effect of environmental constraints and frictional anisotropy on locomotion performance.

## Kinematic model of Snake Robot

The snake robot considered in this work consists of N rigid links. Each link is connected to the next via $N-1$ revolute joints. The length of each link is $2\ell$. The mass of each link is $m$ and is assumed to be concentrated at the center of the link, located at a distance $\ell$ from the adjacent revolute joint.
Let the vectors of link center positions along the $x$ and $y$ axes be defined as: $x_c = [x_{c_1}, ..., x_{c_N}]^T \in R^N$ and in y-axis $y_c = [y_{c_1}, ..., y_{c_N}]^T \in R^N$. Similarly, the positions of the start and end points of the links are given by: $x = [x_1, ..., x_{N+1}]^T \in R^{N+1}$ and $y = [x_1, ..., x_{N+1}]^T \in R^{N+1}$.
The angle between the $i$-th link and the $x$-axis is denoted as $\varphi_i$. The vector of all link angles is: $\varphi_i$ and the vector of all these angles is $\varphi = [\varphi_1, ..., \varphi_N]^T \in R^N$. The relative joint angles between consecutive links are defined by: $q_i$, they create the vector $\mathbf{q} = [q_1, ..., q_{N-1}]^T \in R^{N-1}$.

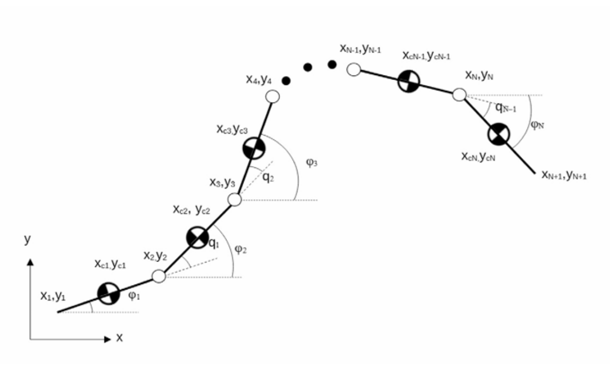

Figure 1: Kinematic parameters

As illustrated in Fig. 1, the kinematic relationships between adjacent joints can be expressed as:

$$\mathbf{J}\mathbf{x} = 2\ell\mathbf{c} \tag{1}$$

$$\mathbf{J}\mathbf{y} = 2\ell\mathbf{s} \tag{2}$$

Let $\mathbf{J} \in R^{N\times(N+1)}$ be an auxiliary matrix. $\mathbf{x} \in R^{N+1}$ the vector of joint positions along the x-axis, and $\mathbf{y} \in R^{N+1}$ the vector of joint positions along the y-axis. $\mathbf{c} \in R^{N}$ and $\mathbf{s} \in R^{N}$ are respectively the vector of individual snake robot links.

$$\mathbf{J} = \begin{bmatrix} -1 & 1 & . & . & . \\ . & -1 & 1 & . & . \\ . & . & . & . & . \\ . & . & . & -1 & 1 \end{bmatrix} \in \mathbf{R}^{N\times(N+1)} \tag{3}$$

$$\begin{aligned} \mathbf{c} &= \left[\cos\varphi_1, \cos\varphi_2, \ldots, \cos\varphi_N\right]^\top \in \mathbf{R}^N, \\ \mathbf{s} &= \left[\sin\varphi_1, \sin\varphi_2, \ldots, \sin\varphi_N\right]^\top \in \mathbf{R}^N \end{aligned} \tag{4}$$

The position of the center of mass (CoM) in the $x$-$y$ plane is given by:

$$\mathbf{p}_c = \begin{bmatrix} p_x \\ p_y \end{bmatrix} = \begin{bmatrix} \frac{1}{Nm}\sum_{i=1}^{N} m x_{c_i} \\ \frac{1}{Nm}\sum_{i=1}^{N} m y_{c_i} \end{bmatrix} \tag{5}$$

## Contact model

A snake robot is subject to interactions with its surrounding environment, primarily ground contact and, when operating within confined spaces such as pipelines, contact with lateral boundaries.
For modeling purposes, it is assumed that frictional forces act at the center of mass (CoM) of each individual link. The friction model employed in snake robot research [1] is **viscous friction**. This model is used to represent ground friction.
The friction force acting on the $i$-th link is formulated as:

$$f_{r,i} = \begin{bmatrix} f_{ti} \\ f_{ni} \end{bmatrix} = -\begin{bmatrix} \gamma_t & 0 \\ 0 & \gamma_n \end{bmatrix} \begin{bmatrix} v_{ti} \\ v_{ni} \end{bmatrix} \tag{6}$$

where $\gamma_t$ and $\gamma_n$ denote the viscous friction coefficients in the tangential and normal directions, respectively, and $v_{ti}$ and $v_{ni}$ are the velocity components of the link in the local contact frame.
To express the local friction forces in the global Cartesian coordinate system, the following rotation matrix is applied:

$$\mathbf{R} = \begin{bmatrix} \cos(\varphi_i) & -\sin(\varphi_i) \\ \sin(\varphi_i) & \cos(\varphi_i) \end{bmatrix} \tag{7}$$

where $\varphi_i$ denotes the orientation angle of the $i$-th link with respect to the global x-axis.

When the robot operates inside a pipeline, additional *side friction forces* arise due to contact between the lateral surfaces of the links and the pipeline walls. These forces are coupled with the ground friction and must be accounted for to accurately predict the motion. The normal contact force between a link and the wall depends on the deformation characteristics of the contact, which can be elastic or compliant. In this work, the contact mechanics are modeled using Hertzian contact theory for non-adhesive elastic contact between a rigid body (the link) and a rigid boundary (the wall).

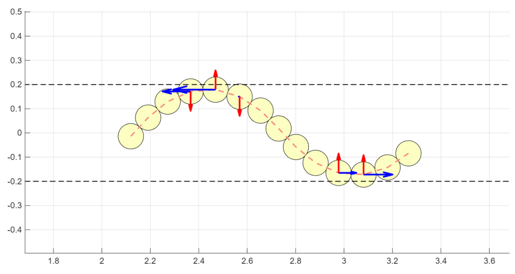

Figure 2: Normal and Tangential forces against wall

As showed in Figure 2 lateral contact force between the snake robot and the side wall of the pipeline, supposing a pipeline along axis x, is applied when the vertical position of the center of mass of the $i$-th link exceeds a defined clearance threshold. Formally, contact is considered active if: $\mathbf{y}_i \geq \frac{d}{2}$ (contact with the upper wall) or $\mathbf{y}_i \leq -\frac{d}{2}$ (contact with the lower wall), where $i \in \{1, \ldots, N\}$ is the index of the link in contact. The effective clearance $d$ is given by $d = D - 2\ell$, where $D$ is the internal diameter of the pipeline and $\ell$ is the half-length of a link.

## Dynamic model

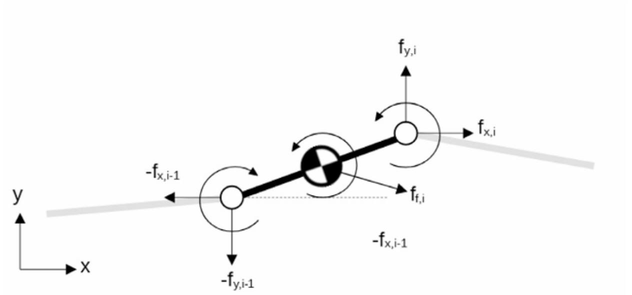

Figure 3: Torques and forces acting on the i-th link

A first step trought the dynamics of the snake has been estimate the number of degrees of freedom (DoF) of a planar snake robot, we apply Grübler's formula for planar mechanisms:

$$\text{DoF} = 3((N+1) - 1 - (N-1)) + (N-1) = N + 2$$

where $N$ is the number of links, $N-1$ is the number of 1-DoF joints (revolute).
This result shows that the snake robot has $N+2$ degrees of freedom in the plane, accounting for $N-1$ joint actuations and 2 additional degrees of freedom due to its floating base (typically the planar position and orientation).
Snake robots are kinematically redundant systems, they have more degrees of freedom than strictly necessary to perform basic movement tasks.
Based on FIgure 3 the translational dynamics of each link are governed by the balance of contact forces, yielding the equations:

$$m\ddot{x}_C = J_f f_x + f_{fx} + f_{wx} \tag{8}$$

$$m\ddot{y}_C = J_f f_y + f_{fy} + f_{wy} \tag{9}$$

where $x_C$ and $y_C$ denote, respectively, the $x$ and $y$ coordinates of the center of each link, and $\mathbf{J}_f$ is an auxiliary incidence matrix defined as:

$$J_f = \begin{bmatrix} 1 & . & . & . & . \\ -1 & 1 & . & . & . \\ . & -1 & 1 & . & . \\ . & . & . & . & . \\ . & . & . & . & 1 \end{bmatrix} \tag{10}$$

The motion of the snake robot CoM is caused only by contact forces affecting the links, namely friction forces $f_f \in R^{2N}$ arising at the interface with the supporting surface, and also from the contact $f_w \in R^{2N}$ caused by pushing the snake robot links against the pipeline side walls.
By consideration of torques acting on the i-th link (Figure 3) we can write the link model as:

$$I\ddot{\varphi} = J_1\mathbf{u} + l\mathbf{S}J_2 f_x + l\mathbf{C}J_3 f_y \tag{11}$$

where $I$ is the moment of inertia of a link, defined as $I = \frac{1}{3}m\ell^2$, $\mathbf{u} = [\,u_1,\, u_2,\, \dots,\, u_{N-1}\,]^\top \in R^{N-1}$ is the vector of joint torques, $\mathbf{C}$ and $\mathbf{S}$ are the cosine and sine diagonal matrix of $\varphi \in R^N$, $\mathbf{f}_x = [\,f_{x1},\, f_{x2},\, \dots,\, f_{x(N-1)}\,]^\top \in R^{N-1}$ denotes the vector of joint constraint forces along the $x$-axis, and $\mathbf{f}_y = [\,f_{y1},\, f_{y2},\, \dots,\, f_{y(N-1)}\,]^\top \in R^{N-1}$ denotes the vector of joint constraint forces along the $y$-axis. The matrices $\mathbf{J}_1$, $\mathbf{J}_2$, and $\mathbf{J}_3$ are auxiliary matrices defined as follows:

$$\mathbf{J}_1 = \mathbf{J}_f \in R^{N\times(N-1)}, \quad \mathbf{J}_2 = -|\mathbf{J}_f| \in R^{N\times(N-1)}, \quad \mathbf{J}_3 = |\mathbf{J}_f| \in R^{N\times(N-1)}.$$

From this model, after we get relations for joint constrains forces, the dynamics of the snake robot can be expressed in the following form:

$$\mathbf{M}\,\ddot{\varphi} + \mathbf{V}\,\dot{\varphi}^2 - \ell\,\mathbf{S}\,\mathbf{R}\,(\mathbf{f}_{fx} + \mathbf{f}_{wx}) + \ell\,\mathbf{C}\,\mathbf{R}\,(\mathbf{f}_{fy} + \mathbf{f}_{wy}) = \mathbf{J}_1\,\mathbf{u} \tag{12}$$

$$\ddot{\mathbf{p}} = \frac{1}{N\,m}\,\mathbf{E}\,(\mathbf{f}_f + \mathbf{f}_w) \tag{13}$$

where:

$$\mathbf{E} = \begin{bmatrix} \mathbf{k}^\top & \mathbf{0}_{1\times N} \\ \mathbf{0}_{1\times N} & \mathbf{k}^\top \end{bmatrix}, \in R^{N\times 2N} \tag{14}$$

is an auxiliary matrix. The matrices $\mathbf{M}$, $\mathbf{V}$, and $\mathbf{R}$ are defined as:

$$\mathbf{M} = I\,\mathbf{I}_N + m\,\ell^2\,\mathbf{S}\,\mathbf{J}_2\,\mathbf{J}_f^{-1}\,\mathbf{P}^{-1}\,|\mathbf{P}|\,\mathbf{S} + m\,\ell^2\,\mathbf{C}\,\mathbf{J}_2\,\mathbf{J}_f^{-1}\,\mathbf{P}^{-1}\,|\mathbf{P}|\,\mathbf{C} \tag{15}$$

$$\mathbf{V} = m\,\ell^2\,\mathbf{S}\,\mathbf{J}_2\,\mathbf{J}_f^{-1}\,\mathbf{P}^{-1}\,|\mathbf{P}|\,\mathbf{C} - m\,\ell^2\,\mathbf{C}\,\mathbf{J}_2\,\mathbf{J}_f^{-1}\,\mathbf{P}^{-1}\,|\mathbf{P}|\,\mathbf{S} \tag{16}$$

$$\mathbf{R} = \mathbf{J}_2\,\mathbf{J}_f^{-1} \tag{17}$$

It is important to note that although the dynamic state vector includes the full set of link angles $\varphi$, $\dot{\varphi}$, $\ddot{\varphi} \in R^N$, actuation is applied only to the relative joint angles between adjacent links. Specifically, the number of actuators is $\mathbf{u} \in R^{N-1}$, which is fewer than the number of configuration variables, rendering the system underactuated. The system cannot impose arbitrary trajectories for all the degrees of freedom independently because there are insufficient control inputs to directly manipulate both the absolute body position and the absolute orientation of each link.
From a modeling perspective, this underactuation implies that it is often more appropriate to express the dynamics in terms of the relative joint angles: $\mathbf{q} = [q_1, q_2, ..., q_{N-1}]^T \in R^{N-1}$. This formulation is advantageous because the vector of joint torques $\mathbf{u}$ and the vector of relative angles $\mathbf{q}$ have the same dimension, simplifying both control design and analysis.
We can set mutual relation between vector $q \in R^{N-1}$ and vector $\varphi \in R^N$ as:

$$\varphi = T\tilde{q} \tag{18}$$

where $\mathbf{T}$ is an upper triangular matrix filled with ones $\in R^{N\times N}$ and $\tilde{q} = [\tilde{q}_1, \tilde{q}_2, ..., \tilde{q}_{N-1}, \varphi_N] \in R^N$.

$$\bar{q} = \begin{bmatrix} \tilde{q} & p \end{bmatrix} \in R^{N+2} \tag{19}$$

This dynamic model can be subsequently modified by a more general equation:

$$\bar{\mathbf{M}}(\tilde{\mathbf{q}})\,\ddot{\tilde{\mathbf{q}}} + \bar{\mathbf{V}}(\tilde{\mathbf{q}}, \dot{\tilde{\mathbf{q}}}) + \mathbf{W}(\tilde{\mathbf{q}})\,\mathbf{f}_e(\tilde{\mathbf{q}},\, \dot{\tilde{\mathbf{q}}},\, \dot{\mathbf{p}}) = \mathbf{B}\mathbf{u}, \tag{20}$$

By considering the dynamic model, the dynamics can be partitioned into actuated and unactuated components:

$$\mathbf{M}_{11}\ddot{\mathbf{q}}_A + \mathbf{M}_{12}\ddot{\mathbf{q}}_U + \bar{\mathbf{V}}_1 + \mathbf{W}_1\,\mathbf{f}_f = \mathbf{u} \tag{21}$$

$$\mathbf{M}_{21}\,\ddot{\mathbf{q}}_A + \mathbf{M}_{22}\;\ddot{\mathbf{q}}_U + \bar{\mathbf{V}}_2 + \mathbf{W}_2\;\mathbf{f}_f = \mathbf{0}_{3\times 1} \tag{22}$$

where $\mathbf{q}_A = \begin{bmatrix} q_1,\, q_2,\, \ldots,\, q_{N-1} \end{bmatrix}^\top \in R^{N-1}$ is the vector of actuated degrees of freedom, and $\mathbf{q}_U = \begin{bmatrix} \varphi_N,\, p_x,\, p_y \end{bmatrix}^\top \in R^3$ is the vector of unactuated degrees of freedom.
Now, we can set a new control input vector as:

$$\ddot{q}_A = \tilde{u} \tag{23}$$

$$\ddot{q}_U = -M_{22}^{-1}(\bar{V}_2 + W_2\mathbf{f}_f + M_{21}\tilde{u}) \tag{24}$$

The model can be expressed in the standard control-affine form by defining the state variables: $x_1 = [\varphi_1, ..., \varphi_{N-1}] \in R^{N-1}$, $x_2 = [\varphi_N, p_x, p_y]$, $x_3 = \dot{x}_1$ and $x_4 = \dot{x}_2$.

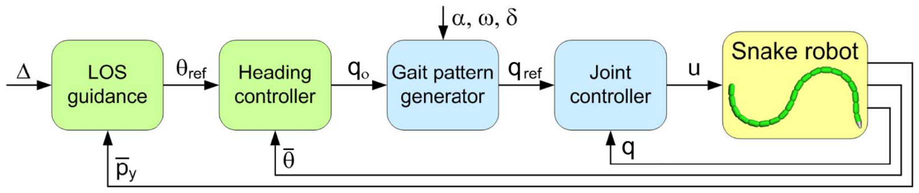

Figure 4: Path following control scheme

The state-space dynamics are:

$$\dot{x} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} x_3 \\ x_4 \\ \tilde{u} \\ A(x) + B(x)\tilde{u} \end{bmatrix} \tag{25}$$

## Control strategy

The control architecture adopted for the snake-like robot is modular and designed to achieve path-following behavior through distributed joint control. The path-following task is treated as a regulation problem and not as a tracking problem, the goal is to remain on the path without any time constraints.
Due to the robot's underactuation, as previously described, we can only control the joints. For this reason, the core of the control scheme is an exponentially stable joint controller, which ensures accurate tracking of reference joint trajectories generated by higher-level components. The control input $\tilde{\mathbf{u}}$ is computed as:

$$\tilde{\mathbf{u}} = \ddot{\boldsymbol{q}}_{\text{ref}} + k_d(\dot{\boldsymbol{q}}_{\text{ref}} - \dot{\boldsymbol{\varphi}}) + k_p(\boldsymbol{q}_{\text{ref}} - \boldsymbol{q}), \tag{26}$$

where $\boldsymbol{q}$ represents the actual joint angles, $\boldsymbol{q}_{\text{ref}}$ the reference joint angles, and $k_p$, $k_d$ are proportional and derivative gains, respectively. This controller ensures the convergence of the joint variables to the desired references with exponential stability.
As shown in Figure 4, the overall control scheme to follow a predefined path is structured as follows:

- **System + Environment**: This block represents the physical snake robot interacting with its environment, as described in the previous chapter. The system outputs include:
  - The joint configuration $\boldsymbol{q}$ (joint angles),
  - The position of the center of mass (CM),
  - The angle $\phi_N$ of the last joint,
  - Their respective velocities.
- **Line of Sight (LOS) Block**: This module is responsible for the path-following behavior. It receives as input the desired trajectory, the current position of the center of mass, and the joint configuration $\boldsymbol{q}$. Based on this information, it computes a direction that indicates the desired heading to maintain alignment with the reference path.

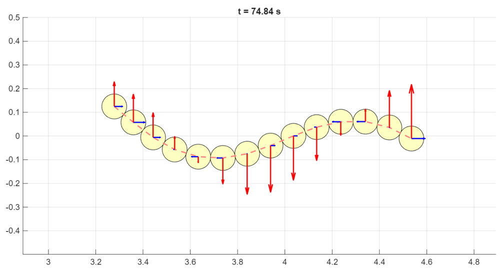

Figure 5: Forces interaction during the slithering

- **Heading Controller**: The input of this block is the desired heading direction from witch it compute the offset of rotation that the snake has to do $q_0$.
- **Central Pattern Generator (CPG)**: The CPG synthesizes rhythmic motion patterns for locomotion. It takes as inputs:
  - The directional offset from the Heading controller block,
  - Parameter values obtained from a planning blocks.

  These inputs are used to modulate the parameters of the oscillatory output to adapt to environmental changes and guide the locomotion direction. The output of the CPG is a set of **reference joint trajectories $\boldsymbol{q}_{\text{ref}}$**, which are then passed to the joint controller.
- **Joint Controller**: As described above, this controller tracks the reference joint trajectories by computing the control inputs $\tilde{\mathbf{u}}$.

The control scheme enables the robot to not only follow a desired path, but also to adapt to environmental disturbances through dynamic modulation of the locomotion pattern.
This structured control approach combines environmental feedback, trajectory guidance, and stable joint regulation, resulting in smooth and adaptive snake-like locomotion. All of these blocks are explained in more detail in the next chapters.

# Snake robot locomotion and CPG

As demonstrated in the model, the snake robot relies on friction forces to achieve locomotion.
Similar to real snakes, movement across a surface is made possible through specific body movements known as gaits. One of the most common gaits is the serpentine gait, illustrated in Figure 5. This gait will be used throughout the project due to its frequent application in literature and its superior performance compared to alternative methods. Nevertheless, the techniques discussed in this chapter can also be used to generate different gaits, depending on the specific requirements.
There are two main approaches to defining a gait:

- Explicit Motion Law - This involves directly defining the movement pattern, such as the well-known Hirose[4] serpentine gait.

- Central Pattern Generator (CPG) - Inspired by biological systems, CPGs are neural circuits that produce rhythmic signals to coordinate motion. These signals can be modulated by high-level commands or sensory feedback, enabling smooth and adaptive locomotion.

In this chapter, we will explore both techniques. Specifically, we will model and implement a low-level CPG controller to generate rhythmic signals capable of producing a slithering gait.

## Base Serpentine Method

The most common form of motion in snake-like robots is based on the *serpenoid curve*. Each joint is controlled using a sinusoidal function defined as:

$$q_i = \alpha \sin(\omega t + (i-1)\delta + q_0)$$

where:

- $\alpha$ is the amplitude,
- $\omega$ is the angular frequency,
- $\delta$ is the phase shift between adjacent joints,
- $q_0$ is a joint offset (used for steering),
- $i \in \{1, 2, \ldots, N-1\}$ is the index of the joint.

The phase shift $\delta$ is typically constant and defined by:

$$\delta = 2\pi \frac{n}{N}$$

where $n$ is the number of complete sine waves along the body.
In biological snakes, amplitude and wave modulation are used to steer and enhance maneuverability, ispired by this steering is achieved by adjusting the offset $q_0$, which laterally shifts the wave, allowing directional control.

### Slithering locomotion analysis

After illustrated the movement of S-shape ,to understand the mechanism of locomotion, it is important to analyze the forces acting on the robot during serpentine motion. As illustrated in Figure 5, the robot propels itself by interacting with the surface through frictional forces generated along the body.
Specifically:

- On straight segments, the body pushes against the ground, generating forward thrust (represented by the blue vectors).
- On curved segments, tangential force components are relatively small, while normal friction forces are more dominant.

The frictional interaction modeled above is introduced using *anisotropic friction coefficients*:

- $\gamma_t$: tangential friction coefficient (along the direction of motion),
- $\gamma_n$: normal friction coefficient (perpendicular to the body).

The disparity between $\gamma_n$ and $\gamma_t$ is essential: higher normal friction reduces backward slip, while lower tangential friction enables lateral motion, resulting in net forward movement.

### Limitations of the Serpenoid Generator

A key limitation of using this type of reference generator for joint angles $q_i$ is its sensitivity to parameter changes. Even minor variations in parameters can lead to abrupt changes in joint trajectories, requiring excessively high torques to follow. This makes the approach impractical for smooth locomotion.
To address this, *Central Pattern Generators (CPGs)* are employed instead. CPGs facilitate smooth gait transitions and significantly reduce joint torque requirements during transitions, making them more suitable for real-time control of snake-like robots.

## CPG Model

As described in the nature tecnique use to generate the motion is the CPG.
Various CPG implementations are used in the literature, and the one adopted here ([5]), it is computationally efficient and show good results also in performance.
A CPG can be represented as a chain-type neural network composed of coupled oscillators. The design of the oscillator model is based on the convergence properties of a gradient system. This design allows for control of frequency, phase difference, and amplitude in a smooth and adaptive way.
The system's potential function always decreases along the gradient direction, converging toward a minimum value in finite time. Even when starting from an arbitrary initial state, the CPG network will converge to the desired phase difference.
As illustrated in Figure **??**, the chain-type CPG network consists of $n$ oscillators (with $n = N - 1$, the number of actuators), each with identical parameters. Let $\varphi_i(t)$ represent the phase of the $i$-th CPG, and $\theta_i(t)$ represent the phase difference between two adjacent CPGs. The desired phase difference is denoted $\tilde{\theta}_i$, that will be equal to $\delta_i$ derived from the serpentine curve model.
To create the model we need to define a potential function, it is created using the generalized coordinates $\Psi$ :

$$\Psi_i = \begin{cases} \varphi_1 - \varphi_2 = \theta_1, & i = 1 \\ \varphi_{n-1} - \varphi_n = \theta_n, & i = n - 1 \\ \varphi_{i+1} + \varphi_{i-1} - 2\varphi_i = \theta_{i-1} - \theta_i, & \text{otherwise} \end{cases} \tag{27}$$

and from this we get the differentiable function :

$$V(\Psi) = \sum_{i=1}^{n} (\Psi_i - \tilde{\Psi}_i)^2 \tag{28}$$

where $\mu_i$ is the coefficient of the convergence velocity and $\tilde{\Psi}_i$ is the generalized coordinates of the desired phase differences.
$V(\Psi)$ can be seen as the gradient system ,deriving and doing a passage to the $\theta$ variables, the gait generator model for the chain-type CPG network is defined as:

$$\begin{bmatrix}\dot{\varphi}_1\\ \dot{\varphi}_2\\ \vdots\\ \dot{\varphi}_n\end{bmatrix} = \begin{bmatrix}\omega_1\\ \omega_2\\ \vdots\\ \omega_n\end{bmatrix} + A\begin{bmatrix}\varphi_1\\ \varphi_2\\ \vdots\\ \varphi_n\end{bmatrix} + B\begin{bmatrix}\theta_1\\ \theta_2\\ \vdots\\ \theta_{n-1}\end{bmatrix} \tag{29}$$

where matrix $A$ is defined as:

$$A = \begin{bmatrix} -\mu_1 & \mu_2 & & & 0 \\ \mu_2 & -2\mu_2 & \mu_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \mu_{n-1} & -2\mu_{n-1} & \mu_{n-1} \\ 0 & & & \mu_n & -\mu_n \end{bmatrix}_{n\times n}$$

and matrix $B$ is:

$$B = \begin{bmatrix} 1 & 0 & & \\ -1 & 1 & & \\ & -1 & \ddots & \\ & & \ddots & 1 \\ 0 & & & -1 \end{bmatrix}_{n\times(n-1)}$$

The convergence rate of the system is governed by matrix $A$ and is influenced by the value of $\mu_i$, in our case the $\mu = 3$ and it is a equal for all the indecies .
The above model return us the phase of the sine , the final equation will follow the definit formula of the serpentine curve and we need to add two PD controllers are used to ensure convergence of the amplitude $R$ and the offset $C$. The full model of the CPG system is:

$$\begin{cases} \dot{\varphi}_i = \omega_i + A\{i,:\}\cdot\Phi + B\{i,:\}\cdot\tilde{\Theta} \\ \ddot{r}_i = a_i\left[\dfrac{a_i}{4}(R_i - r_i) - \dot{r}_i\right] \\ \ddot{c}_i = a_i\left[\dfrac{a_i}{4}(C_i - c_i) - \dot{c}_i\right] \\ x_i = c_i + r_i\sin(\varphi_i) \end{cases} \tag{30}$$

The Figure 6 shows the comparison between the signal generated by the CPG and a standard sine wave during changes in amplitude $R$ and offset $C$. On the left, it is possible to observe a change in amplitude from 0.2 to 0.4, with the $a_i$ gain for the CPG's PD controller set to 25. At 49.5 seconds, the sine wave exhibits a discontinuity, unlike the CPG signal, which maintains a smooth transition. A similar result is observed during the change in the offset $q_0$; the figure shows a transition from 0 to 15 degrees.

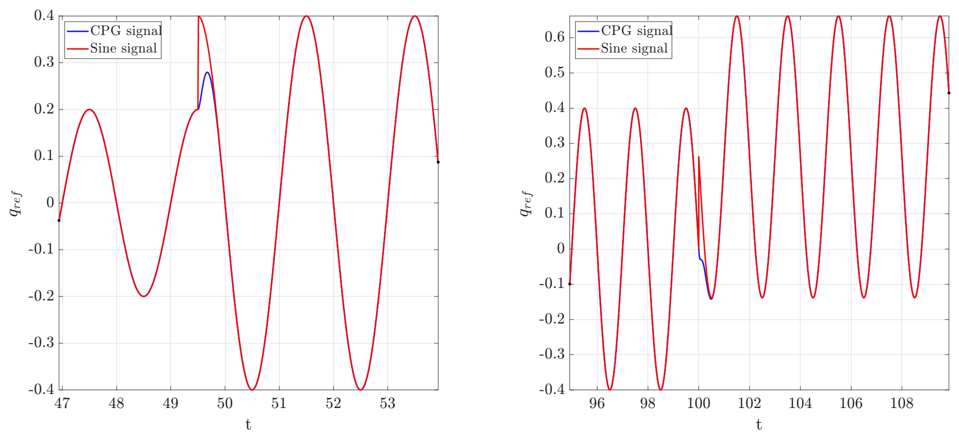

Figure 6: Output comparison between sine function and CPG at the variation of R and C.

**Parameter Smoothing and Head Movement Stabilization** One of the problems encountered when using a CPG (Central Pattern Generator) as a reference for joint control is the abrupt change in configuration when the parameters are suddenly modified. This often leads to high torque demands, making control more difficult and less efficient. To mitigate this, a parameter smoothing system is introduced. This system applies a gradual transition of parameters over a fixed time interval to avoid abrupt changes. The transition is defined by the following function:

$$y = \begin{cases} A_1 & t \le t_1 \\ \alpha \sin\left(\frac{\pi}{t_2 - t_1}\left(t - \frac{t_1 + t_2}{2}\right)\right) + \beta & t_1 < t \le t_2 \\ A_2 & t \ge t_2 \end{cases}$$

$$\alpha = \frac{|A_2 - A_1|}{2}, \quad \beta = \frac{A_1 + A_2}{2}$$

Another issue affecting snake-like robot motion is excessive oscillation of the heading direction. As shown in the control section, the heading is calculated as the sum of the joint angles along the body, and it is highly influenced by the gait parameters. With simple gaits, the heading direction tends to oscillate significantly, sometimes reaching large values. This affects overall stability and control performance.

Moreover, if a camera is mounted on the robot's head, such oscillations degrade image quality and sensor reliability. To address this, a curvature reduction technique inspired by the approach in [5] is adopted. The method introduces a linear reduction factor $P$ that modulates the amplitude along the body from head to tail:

$$P = \left(\frac{k}{N}y + z\right), \quad \forall k \in [0, N], \quad P \in [0, 1]$$

where $z$ and $y$ are linear reduction parameters. Incorporating $P$ into the gait function yields:

$$x_i = P r_i \sin(\varphi) + c_i$$

This modulation enables reduced oscillation at the head while preserving motion at the tail. The resulting behavior is illustrated in Figure 7.

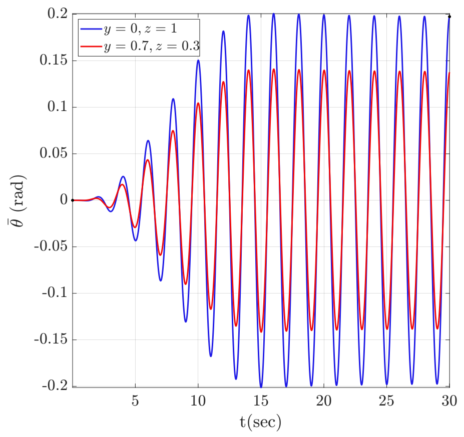

Figure 7: Comparison of head-tail smoothing using different $z$ and $y$ values.

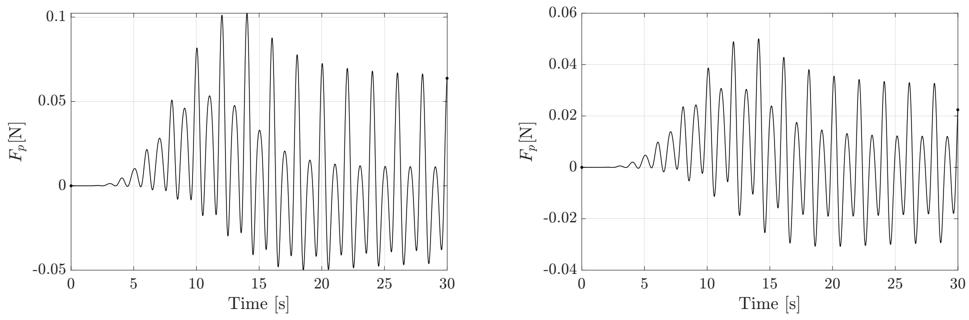

Figure 8: Propulsive forces during a straight movement with linear reduction and without

Figure 7 illustrates a start-up process followed by a straight motion along the x-axis, comparing linear smoothing with no smoothing. It demonstrates how joint amplitude modulation helps reduce heading oscillation in the presence of smoothing. Two scenarios are tested: one with $z = 1$ (no smoothing) and another with $z = 0.3$, $y = 0.7$. The latter configuration results in a more stable robot posture but comes at the cost of reduced speed, as less of the body is used to generate forward propulsion.

This behavior is further highlighted in Figure 8, which shows how the propulsive forces during the start-up phase are almost halved when linear head smoothing is applied, compared to the non-smoothed case. This reduction occurs because the robot uses only part of its body to push forward. Ultimately, the robot achieves lower velocity for a marginal gain in heading stability. This trade-off is only worthwhile in applications that require stable perception, such as when a camera is mounted on the head. Otherwise, alternative gait strategies should be considered.

## Smooth Start-up Strategy

Another critical challenge in CPG-based control is managing the robot's behavior during the start-up phase. When the robot begins from a straight-line configuration with zero initial joint angles, the CPG network needs time to establish steady-state phase differences. During this transient phase, the robot tends to deform uniformly into a semi-circular shape before reaching the intended S-shaped locomotion pattern.
To address this, a smooth start-up strategy is implemented by introducing a scaling parameter $\lambda$, which increases as described above in smooth function from 0 to 1 over a predefined time interval. The joint reference signals are modulated accordingly:

$$x_{oi} = \lambda x_i$$

This technique ensures that the robot starts with the correct phase distribution, immediately forming the desired S-shaped posture and avoiding unwanted initial distortions (as shown in Figure 6).

## Path Following Control

Line-of-Sight (LOS) is a trajectory tracking strategy commonly employed to guide a snake-like robot along a predefined path. The fundamental objective is to continuously minimize the lateral deviation between the robot's center of mass (CoM), denoted by the position vector $\mathbf{p} = [p_x, p_y]$, and the reference trajectory $f(x)$.
At each time instant, the LOS algorithm identifies the point on the reference path nearest to the CoM, defined as $(p_{x,\text{ref}}, p_{y,\text{ref}})$. From this, it determines a reference orientation $\theta_{\text{ref}}$ towards which the robot should align. The angular deviation between the current heading and the reference orientation is subsequently used to compute a corrective phase offset, effectively steering the robot back toward the path.

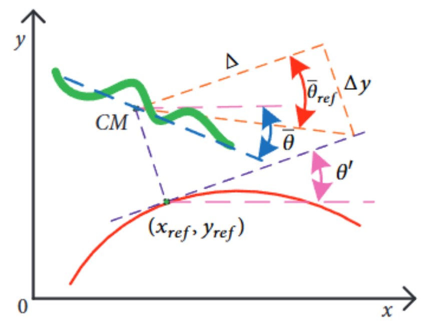

Figure 9: Schematic of the adaptive LOS path-following control

The parameter $\Delta$ determines the baseline influence range of the LOS controller. Smaller values of $\Delta$ correspond to higher resolution and increased trajectory tracking accuracy.

### Principle of Operation

The main purpose of the LOS strategy is to improve the accuracy of path following in scenarios where the robot must execute maneuvers in constrained environments [6].In this work, the direction controller is based on an *adaptive LOS guidance law*, which can be interpreted as a regularization process that, at each instant, projects the robot's position onto the reference path and minimizes the cross-track error.

This control strategy works well for one key reason which can also become a limitation. Thanks to the robot's continuous motion, it always moves forward, and the point it aims to follow updates accordingly. However, if the robot were able to perfectly reach the reference point and stop on it, it could get stuck there, unable to proceed.

The main drawback arises when the reference path includes sharp curves or closely spaced parallel segments. In such cases, the robot might inadvertently cross over to the adjacent segment and begin following a different part of the path, effectively jumping ahead and skipping sections.

As illustrated in Figure 9, the tangent line to the reference trajectory at the projection point $(p_{x,\text{ref}},\, p_{y,\text{ref}})$ defines the desired heading direction. The distance between the robot CoM and the reference point is given by:

$$\gamma = \sqrt{(p_x - p_{x,\text{ref}})^2 + (p_y - p_{y,\text{ref}})^2},$$

and the cross-track error (lateral deviation) is computed as:

$$\Delta y = \gamma \cdot \operatorname{sign}(p_y - p_{y,\text{ref}}).$$

The rationale for introducing the sign function is that it is insufficient to consider only the magnitude of the distance to the reference trajectory; it is also necessary to determine whether the robot is positioned above or below the reference path.

The average orientation of the snake robot, denoted $\bar{\theta}$, is evaluated as the mean of the joint angles along the body:

$$\bar{\theta} = \frac{1}{N}\sum_{i=1}^{N} \varphi_i, \tag{31}$$

where $N$ is the number of links and $\varphi_i$ is the cumulative joint angle of link $i$.

Due to the oscillatory locomotion pattern inherent in snake-like robots, perfect tracking is not achievable. Consequently, the CoM trajectory will exhibit bounded steady-state oscillations about the reference path.

### Reference Orientation and Look-Ahead Distance

To define the desired orientation, the LOS controller computes:

$$\theta_{\text{ref}} = -\arctan\left(\frac{\Delta y}{\Delta}\right), \tag{32}$$

where $\Delta$ is the *look-ahead distance* parameter that modulates the convergence dynamics. The look-ahead distance is adapted online according to the robot's position and heading:

$$\Delta = \Delta_0 - \rho_1 \operatorname{sign}(\bar{\theta}\,\Delta y)\left\lfloor \frac{|\gamma| - \zeta_1}{\sigma_1} \right\rfloor. \tag{33}$$

## Heading Controller and Gain Adaptation

The control law computes the phase offset $q_0$, which governs the correction applied to the robot heading:

$$q_0 = k_\theta\,(\bar{\theta} - \theta_{\text{ref}}), \tag{34}$$

where $k_\theta$ is an adaptive gain defined as:

$$k_\theta = k_{\theta_0} + \rho_2 \left\lfloor \frac{|\gamma| - \zeta_2}{\sigma_1} \right\rfloor + \rho_3 \max\left(0, \left\lfloor \frac{|\bar{\theta} - \zeta_3|}{\sigma_2} \right\rfloor\right). \tag{35}$$

The parameters $\rho_1$, $\rho_2$, and $\rho_3$ assign relative weights to different contributions within the control law, effectively shaping the priority given to specific error components during tracking. In contrast, the sensitivity parameters, denoted as $\zeta_{s1}$, $\zeta_{s2}$, and $\zeta_{s3}$, modulate the responsiveness of the controller to deviations, acting as a form of gain adjustment to tune the control effort. The terms $\sigma_1$ and $\sigma_2$ define smoothing thresholds for the angular correction, which helps the controller maintain stability when the reference trajectory exhibits abrupt curvature variations. The parameter $k_{\theta_0}$ is introduced to adjust the heading correction dynamics and to improve convergence properties when the snake robot deviates from the desired orientation.

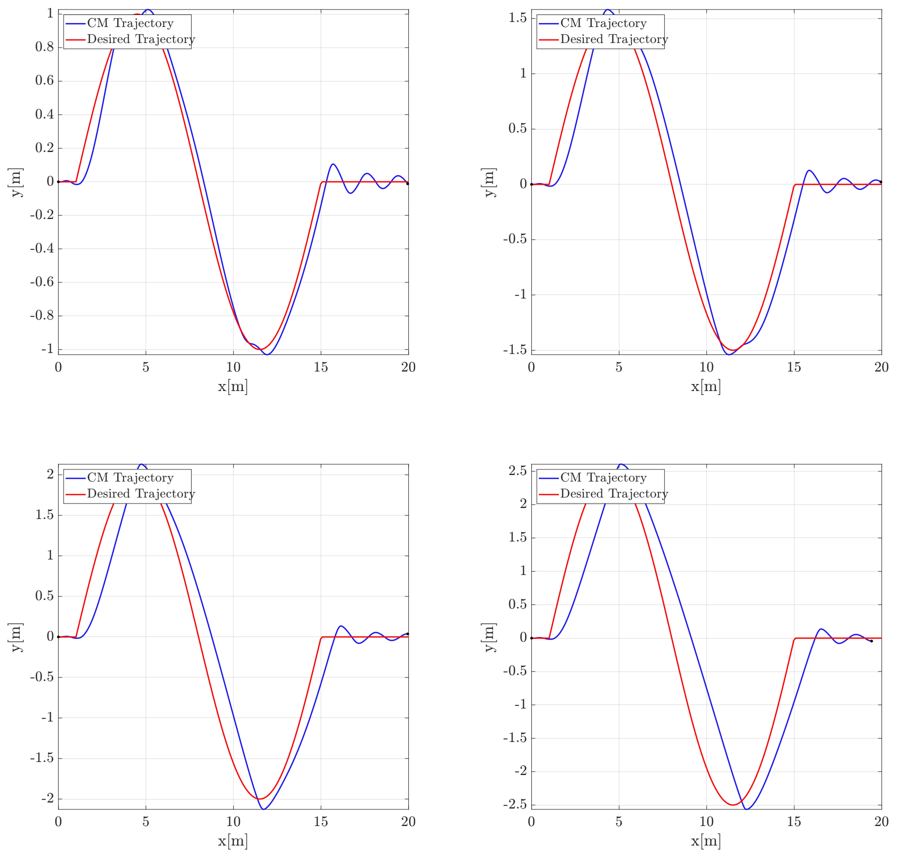

Figure 10: Comparison of trajectory tracking performances for increasing sharpening of the sine curve amplitude$A$. Top-left $A = 1$, Top-right $A = 1.5$, Bottom-left $A = 2$, Bottom-right $A = 2.5$.

The first simulation was conducted using a set of parameters tailored for the Line-of-Sight (LOS) guidance law.
The plots were generated by varying the amplitude **A** in the reference trajectory defined as:

$$y = A\sin(x\frac{\pi}{7}) \tag{36}$$

This parametric function allows us to evaluate the performance of the snake robot in tracking sinusoidal paths of increasing curvature. As the amplitude increases, the trajectory becomes more aggressive, requiring sharper directional changes. The simulation results indicate that the snake robot exhibits degraded tracking performance for higher amplitude values. This behavior highlights the system's limitations in responding to rapid curvature transitions, particularly due to the inertial and frictional constraints inherent in the robot's dynamics. In essence, larger curvature imposes greater demands on the robot's actuation and coordination mechanisms, reducing its ability to maintain close adherence to the desired path.

## Simulation and Results

The simulation is organized into three phases, each illustrating a different locomotion mode of the snake robot. The first phase demonstrates free undulatory motion without environmental contact. The second phase shows propulsion by exploiting contact forces against the pipeline walls. The third phase applies adaptive Line-of-Sight control to track a predefined reference trajectory. Together, these scenarios highlight the robot's capabilities in both open-loop and closed-loop locomotion.
The use parameters are summarized in

Table 1: Simulation Parameters

| Type | Symbol | Value | Type | Symbol | Value |
|---|---|---|---|---|---|
| Link mass | $m$ | 0.4 kg | LOS parameter | $\Delta_0$ | 0.15 |
| Link length | $\ell$ | 0.0525 m | LOS parameter | $\rho_1$ | 0.01 |
| Diameter | $d$ | 0.4 m | LOS parameter | $\zeta_1$ | 0.01 |
| Tangential friction | $\gamma_t$ | 0.15 | LOS parameter | $\sigma_1$ | 0.01 |
| Normal friction | $\gamma_n$ | 0.3 | LOS parameter | $k_{\theta_0}$ | 0.1 |
| Reference amplitude | $A$ | 1 | LOS parameter | $\rho_2$ | 0.001 |
| Proportional gain | $k_p$ | 25 | LOS parameter | $\zeta_2$ | 0.01 |
| Derivative gain | $k_d$ | 10 | LOS parameter | $\zeta_3$ | 15 ° |
| CPG amplitude gain | $a_i$ | 25 | LOS parameter | $\rho_3$ | 0.001 |
| CPG offset gain | $b_i$ | 25 | LOS parameter | $\sigma_2$ | 10 ° |

### Phase 1: Free Locomotion Without Wall Contact

In the initial phase (Figure 11), the robot moves along the **x**-axis without interacting with the pipeline walls. This motion does not require the use of the Line-of-Sight (LOS) guidance law, as the trajectory is simply a straight line and no correction is necessary. The oscillatory motion of the body is generated by imposing a nominal amplitude parameter, set to $R = 0.2$ . With this configuration, the snake maintains a consistent forward progression without contacting the environment.

### Phase 2: Locomotion with Wall Contact

In the second phase (Figure 11), the objective is to demonstrate propulsion through interaction with the pipeline side walls. To ensure that the robot body establishes continuous contact, the amplitude parameter ,after 50 sec ,is increased to $R = 0.4$, resulting in a larger oscillation envelope. In this regime, propulsion is achieved by exploiting the reaction forces generated by friction and normal contact forces against the walls. An important consideration in this phase is that the robot naturally tends to progress in the negative **x**-direction due to the orientation of the undulatory gait relative to the contact points. Therefore, to maintain forward motion along the positive **x**-axis, it is necessary to reverse the commanded direction of locomotion, this is done inverting the $\omega = -\omega$.

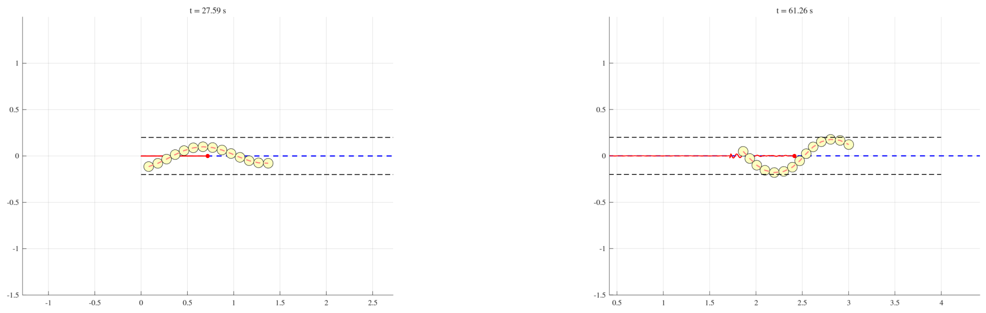

Figure 11: Simulation during the contact on the right and without contact on the left

### Phase 3: Trajectory Tracking Using LOS Control

In the final phase, the snake exit from the pipe and the Line-of-Sight control algorithm is activated to enable trajectory tracking along the predefined sinusoidal reference path.

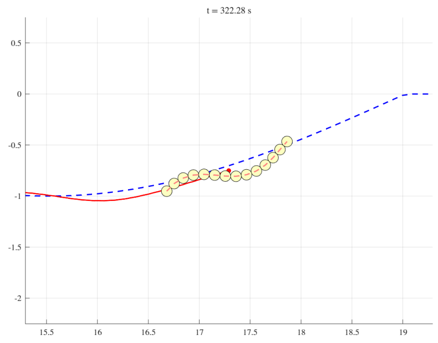

Figure 12: Simulation of path following control

Here, the robot continuously estimates the closest point on the reference trajectory and dynamically adjusts its heading direction to minimize the cross-track error.

### Results

The defined trajectory was tested both with and without linear head-amplitude reduction in the slithering gait. We evaluate tracking performance using the parameter $\gamma$, which measures the distance between the robot's center of mass and the closest point on the reference trajectory.
Figure 13 compares the evolution of $\gamma$ for the two configurations. Both experiments achieve similar tracking accuracy; however, the configuration without head-amplitude reduction yields a significantly higher forward velocity.
In this scenario, head-amplitude reduction does not improve performance and may be omitted in future implementations.

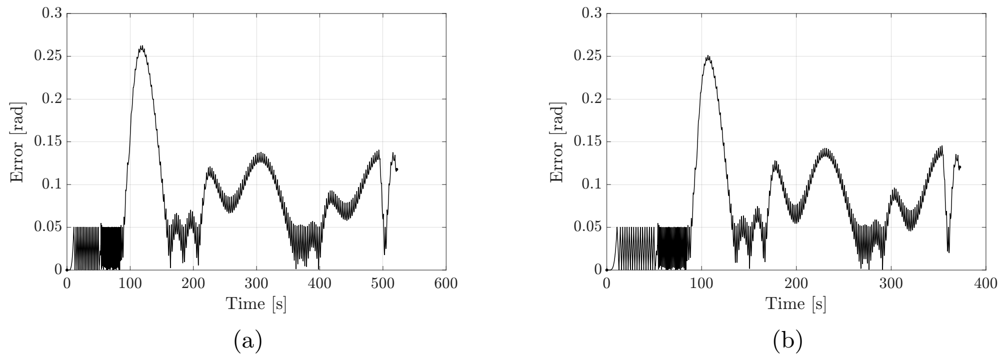

Figure 13: Comparison of tracking error $\gamma$ with (right) and without (left) linear head-amplitude reduction.

## References

[1] Ivan Virgala, Martin Varga, Peter Ján Sinčák, Tomáš Merva, Roman Mykhailyshyn, Michal Kelemen, *Mathematical framework for snake robot motion in a confined space*, Volume 132,2024,Pages 22-40,ISSN 0307-904X, https://doi.org/10.1016/j.apm.2024.04.020

[2]

[3] D. Zhang, Q. Xiao, Z. Cao, R. Huang and Y. Fu, *"Smooth transition of the CPG-based controller for snake-like robots,"* 2017 IEEE International Conference on Robotics and Biomimetics (ROBIO), Macau, Macao, 2017, pp. 2716-2721, doi: 10.1109/ROBIO.2017.8324830.

[4] H. Hirose, *Biologically Inspired Robots*

[5] Zhenshan Bing *Biological-inspired Hierarchical Control of a Snake-like Robot for Autonomous Locomotion*

[6] Cao, Zhengcai, Zhang, Dong, Hu, Biao, Liu, Jinguo, *Adaptive Path Following and Locomotion Optimization of Snake-Like Robot Controlled by the Central Pattern Generator* , Complexity, 2019, 8030374, 13 pages, 2019. https://doi.org/10.1155/2019/8030374