# Implementation of a PSO trajectory planner using kinematic controllers that ensure smooth differential robot velocities

Aldo Aguilar Nadalini
*Mechatronics Engineering Department*
*Universidad del Valle de Guatemala*
Guatemala City, Guatemala
agu15170@uvg.edu.gt

Dr. Luis Alberto Rivera
*Mechatronics Engineering Department*
*Universidad del Valle de Guatemala*
Guatemala City, Guatemala
larivera@uvg.edu.gt

MSc. Miguel Zea Arenales
*Mechatronics Engineering Department*
*Universidad del Valle de Guatemala*
Guatemala City, Guatemala
mezea@uvg.edu.gt

***Abstract*—A multi-agent differential robot system requires a definite algorithm to behave as a swarm with goal searching capabilities. The classic Particle Swarm Optimization (PSO) algorithm is designed for particles with no mass or physical dimensions unlike differential robots. Therefore, the use of the PSO as a trajectory planner is proposed to enable the agents to collectively find the optimal path to a goal in a fitness function. The PSO trajectory planner uses multiple known parameters such as inertia weight, a constriction parameter, two scaling factors and two uniformity factors (for the cognitive and social components of the PSO) optimized for differential robots. The planner needs to take into account the restrictions derived from the kinematic equations of the robotic agents and the finite dimensions of the search space. For that purpose, it is coupled with the necessary controllers to map particle velocities into smooth and continuous differential robot velocities. Four different controllers were tested including: Transformed Unicycle Controller (TUC), Transformed Unicycle with LQR (TUC-LQR), Transformed Unicycle with LQI (TUC-LQI), and a Lyapunov-stable Pose Controller (LSPC). The TUC-LQI was the controller that performed better in terms of achieving smooth continuous differential robot velocities while following the paths generated by the PSO trajectory planner.**

***Index Terms*—PSO trajectory planner; differential robots; swarm intelligence; control; LQI; LQR; PSO.**

## I. Introduction

The Particle Swarm Optimization (PSO) algorithm is a popular tool in the computational intelligence field when it comes to finding the optimal solution of a determined fitness function. It is designed for swarms of particles with no mass or physical dimensions unlike differential robots. The movements that robots can execute are limited, unlike particles which can move in any direction at any velocity. The main objective of this research is to find a way of coupling the searching capabilities of this algorithm to a swarm of differential robots such that it can be used in goal searching applications. To achieve this, the use of a PSO trajectory planner (PSO-TP) algorithm is proposed to make the robots search for a path that minimizes the cost of their own position (X, Y) in a bi-dimensional plane evaluated in a fitness function. In this case, the goal or target to look for would be the absolute minimum of the fitness function. The main challenge of coupling the PSO-TP algorithm to differential robots (i.e. E-Puck robots [1]) is the correct implementation of it considering the kinematic restrictions of the swarm agents, the velocity saturation limits of its wheels and the rate at which its wheels can change speed.

To take this into account, a set of kinematic controllers are designed and tested. They are tasked with the calculation of the differential robot velocities using particle velocities as inputs. This research compares the performance of a Transformed Unicycle proportional controller (TUC) for the diffeomorphic model of a differential robot [2], pose controllers with Lyapunov stability, and LQR/LQI controllers for the diffeomorphic model as well. The objective is to determine which controller performs better when it comes to following the goal path generated by the PSO trajectory planner while at the same time reducing the saturation and hard stops in the velocities of the robots' actuators. In section II, the background of this investigation is presented. Section III shows the structure and implementation of the PSO-TP. In section IV, the structuring of the simulations to test the PSO-TP implementation via the kinematic controllers is explained. Lastly, section V shows the performance results of the PSO-TP coupled with each controller during the simulations.

## II. Background

### A. Classic PSO

The classic PSO algorithm was first proposed in [3]. It consists in a set of possible solutions, referred as particles, located throughout the problem space. Each particle has a fitness value based on its current position and a velocity vector that indicates the direction to which the particle's position will change during the current iteration. The PSO algorithm initializes all particles' position randomly and then it is able to locate optima in a fitness function by updating generations of particle positions. Each particle moves throughout the space by following its new velocity vector calculated by adding the current velocity vector of the particle $\mathbf{V}_i$, the cognitive factor which is the distance between the particle's current position $\mathbf{x}_i$ and the best position found locally by the particle $\mathbf{p}_l$, and

the social factor which is the distance between the particle's current position $\mathbf{x}_i$ and the best position found globally by the entire particle swarm $\mathbf{p}_g$. Therefore, the two vectorial equations used by the PSO algorithm to search for optima are:

$$\mathbf{V}_{i+1} = \varphi \cdot (\omega \mathbf{V}_i + c_1\rho_1(\mathbf{p}_l - \mathbf{x}_i) + c_2\rho_2(\mathbf{p}_g - \mathbf{x}_i)) \quad (1)$$

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \mathbf{V}_{i+1} \quad (2)$$

Equation (1) is the update rule for the particle's velocity. The variable $\varphi$ is a constriction parameter multiplying all velocity factors to limit the maximum step that a particle can take when updating its position inside the problem space [4]. The variable $\omega$ is an inertia parameter first introduced in [5] that weighs the contribution of the current velocity in the calculation of the particle's new velocity. Its value can be computed in several ways including the calculations shown in [6]. Parameters $c_1$ and $c_2$ are scaling parameters for the cognitive and social factors in (1). The effects of different values for the scaling parameters in the particle swarm's behavior are explored in [7]. Variables $\rho_1$ and $\rho_2$ are uniformity factors which have a real random value in the interval (0,1]. These last factors add a key random behavior to the swarm so it can appropriately move in different directions while searching for optima. The update rule for the particle's position is shown in (2), where the new position vector $\mathbf{x}_{i+1}$ is the sum of the particle's current position $\mathbf{x}_i$ vector and the new velocity vector $\mathbf{V}_{i+1}$.

The PSO is ideal for swarm intelligence optimization applications since it is cheap in terms of computational complexity. Particles of the swarm do not need to know any information from other particles other than their found local best positions $\mathbf{p}_l$ to compute the swarm's global best position $\mathbf{p}_g$. Each particle's high individuality using the PSO algorithm makes it easy to implement in applications where there is no central processor calculating the results of each agent (e.g. mobile robots). Another advantage of the PSO when used in robotic applications is that the new position update of each agent depends on its current position. This means that the algorithm computes continuous paths for each agent as they search for optima.

### B. Kinematics of differential robots

Differential robots are non-holonomic systems having three DOF and only two wheel actuators. Therefore, velocity mapping is required to transform the linear ($v$) and angular ($\omega$) velocities of the robot into angular velocities of the robot's wheels ($\Phi_R$, $\Phi_L$). This is done by using the well-known unicycle robot model [2]:

$$\Phi_R = \frac{v + \omega l}{r} \quad (3)$$

$$\Phi_L = \frac{v - \omega l}{r} \quad (4)$$

where $r$ is the wheel radius of the robot and $l$ is the base radius of the robot.

### C. Diffeomorphism of differential robot kinematics

The differential robot model is highly nonlineal and not controllable. Therefore, a diffeomorphism of the original system into a simpler system is required. This type of control transformation is based on the assumption that we can directly control the planar velocities $u_1$ and $u_2$ of a single point in a differential robot [8]. The kinematic equations transforming the planar velocities of the point into differential robot velocities are written as follows:

$$v = u_1 \cdot cos(\theta) + u_2 \cdot sin(\theta) \quad (5)$$

$$\omega = \frac{-u_1}{l} \cdot sin(\theta) + \frac{u_2}{l} \cdot cos(\theta) \quad (6)$$

where all variables are the same as used in the last subsection.

### D. Robot velocity controllers

There are many ways to achieve the movement of a robot from point to point. Several control laws have been developed in different investigations and have resulted in the correct translation of a robot. Control rules can be optimized for velocity smoothness, trajectory smoothness or both.

*1) Point to point planar velocities:* Planar velocity inputs can be calculated in different ways by using the error between the goal position and the actual position of the controlled object. In the case of a system based on a bi-dimensional plane XY, the planar velocities can be computed as follows:

$$\mathbf{u} = I \cdot \tanh(k \cdot \mathbf{e}) \quad (7)$$

where $\mathbf{e}$ is the vector of the position errors between the goal's coordinates and the robot's position [8]. $I$ is a saturation constant for the hyperbolic tangent limiting the output and $k$ is a proportional scaling factor. The velocities can also be computed using LQR control as follows:

$$\mathbf{u} = -\mathbf{K} \cdot (\mathbf{x} - \mathbf{x}_g) + \mathbf{u}_g \quad (8)$$

using the negative state feedback structured by multiplying the resulting LQR matrix $\mathbf{K}$ and the error between the system state $\mathbf{x}$ (robot coordinates) and the operational point ($\mathbf{x}_g$, $\mathbf{u}_g$). The LQR performance can be improved by adding integral action and turning it into an LQI controller with the control law written as follows:

$$\mathbf{u} = -\mathbf{K} \cdot (\mathbf{x} - \mathbf{x}_g) - \mathbf{K}_I \cdot \mathbf{z}_i + \mathbf{u}_g \quad (9)$$

where $\mathbf{z}_i$ is the integral of the difference between a set reference and the actual state of the system calculated during a certain period of time [9].

*2) Pose Controllers:* A useful control technique for goal reaching with smooth trajectories is the pose control. This type of controller tries to minimize the error in three different parameters measured in polar coordinates. Firstly, parameter $\rho$ is the magnitude of the vector between the center of the robot and the goal point, $\alpha$ is the angle between the $\rho$ vector and the horizontal axis of the robot's reference frame, and $\beta$ is the

desired angle of orientation when the robot reaches the goal [10]. The control laws are structured as follows:

$$v = k_\rho \cdot \rho \tag{10}$$

$$w = k_\alpha \cdot \alpha + k_\beta \cdot \beta \tag{11}$$

Some variants of this last pose controller have been developed. In [11], pose control laws were developed using the direct Lyapunov stability criterion and the Barbalat's Lemma. This controller only minimizes the $\rho$ and $\alpha$ parameters. The control laws are structured as follows:

$$v = K_\rho \rho \cdot cos(\alpha); \quad K_\rho > 0 \tag{12}$$

$$\omega = K_\rho sin(\alpha) cos(\alpha) + K_\alpha \alpha; \quad K_\alpha > 0 \tag{13}$$

## III. Structure of the PSO trajectory planner (PSO-TP)

The classic PSO algorithm generates highly irregular trajectories for the swarm particles when updating their position, so following these paths with differential robots would be nearly impossible considering their kinematic restrictions. These restrictions include their mass, size, number of actuators and actuator saturation. However, the PSO velocity vectors generated by computing (1) still work as pointers to new PSO positions $\mathbf{x}_{i+1}$ given by (2). These multiple reference points can be easily tracked by a differential robot. Based on this idea, the PSO trajectory planner is designed as an algorithm that is tasked with simply updating the position of a reference point, or **PSO Marker**. The PSO marker is then tracked by the kinematic controller of the differential robot.

The PSO-TP starts by computing the first position of the PSO Marker $\mathbf{x}_{i+1}$ using (1) and (2). This PSO Marker is then taken by the kinematic controller of the robot as the goal state that the system state $\mathbf{x}_i$ needs to reach. The kinematic controllers use the error between $\mathbf{x}_{i+1}$ and $\mathbf{x}_i$ to generate the control signals, being $\mathbf{x}_{i+1}$ the equilibrium point where the error is zero. While driving towards equilibrium, the robot will describe a smooth and continuous path. After a certain amount of iterations of the kinematic controller, the PSO-TP becomes active again and it updates the location of the PSO Marker $\mathbf{x}_{i+1}$. The kinematic controller starts tracking the new reference point and describes another continuous segment of the trajectory. This process is repeated continuously.

As the PSO algorithm converges to the absolute minima of the fitness function, the PSO Marker converges to this goal as well. Therefore, it drives the robot towards this same point by concatenating each continuous segment described by the agent while tracking each location of the PSO Marker. Note that if the update frequency of the PSO-TP gives the kinematic controller enough time to reach the exact position of the current PSO Marker, the robot would describe the same irregular path of a PSO particle. Therefore, the update frequency of the PSO-TP needs to be increased so that the kinematic controller can only drive the robot towards the current PSO Marker during a short period of time generating a reduced portion of the trajectory needed to reach the marker. These concatenated short trajectory portions create a smooth and continuous path towards the goal followed by the robot as shown in the next figure:

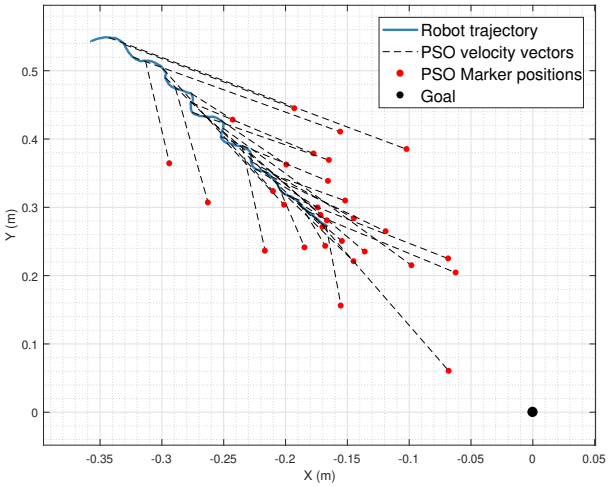


Fig. 1. Smooth trajectory of the PSO Planner

As shown in the last picture, the robot navigates towards the goal (denoted by the black dot at (0,0)) following a continuous path comprised of short trajectory segments that track each updated position of the PSO Marker shown as red dots in the plot.

### A. Tuning the PSO-TP

There are several parameters that influence the behaviour of the PSO trajectory planner. The update rule for the PSO velocity shown in (1) has four parameters that need to be tuned. In this research, the inertia parameter $\omega$ was implemented by using the Linear Decreasing Inertia Weight formula, presented in [6], to reduce the convergence error of the robotic swarm. Secondly, the scaling parameters $c_1$ and $c_2$ were set to a value of 2.05 as used in [4]. The constriction factor $\varphi$ was calculated using the formula presented in [4] as well. The sum of $c_1$ and $c_2$ used in this formula is greater than 4 guaranteeing convergence of the PSO with the resulting value of $\varphi$.

Aside from the native parameters of the PSO, there are other added parameters to tune the PSO-TP. A new scaling parameter $\eta$ multiplying $\mathbf{V}_{i+1}$ in (2) is added to have a direct control over the length of the PSO velocity vectors, thus limiting the acceptable distances bewteen the robot and the PSO Marker. This was done to avoid the PSO Marker surpassing the search space boundaries and causing the kinematic controllers to saturate the robots' actuators. The modified update rule for the particle's position is structured as follows:

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \eta \mathbf{V}_{i+1} \tag{14}$$

As explained before, the update frequency of the PSO Marker is directly related to the smoothness of the resulting path of the robot towards the goal. Therefore, a parameter $P_s$ is used in the algorithm to directly configure the amount of

iterations of the kinematic controllers that will be executed between each activation of the PSO-TP.

## IV. Structure of the PSO-TP simulations for kinematics controller testing

After selecting the adequate PSO-TP parameters to achieve the desired swarm behavior, a simulated world was structured in Webots [12]. The objective was to proceed with the analysis of the PSO-TP implementation on differential robots using 4 different kinematic controllers for velocity mapping. The simulations were performed using swarms of 10 E-Puck robots in a 2x2m space and a sampling period ($T_s$) of 32ms. The amount of agents used in the simulations was determined according to the dimension constrains of the simulated search space to avoid overcrowding. The Sphere benchmark function with absolute minima at (0,0) was used as fitness function $f(\cdot)$ to evaluate the algorithm's performance. Simulated GPS and Compass sensors were used to obtain position and orientation information of each E-Puck agent [12]. All agents were equipped with a simulated radio receiver to implement a fully connected communication topology in the swarm. This was key for the transmission and reception of each agent's local best position $\mathbf{p}_l$ and consequent polling for the global best position $\mathbf{p}_g$ found by the swarm.

As detailed in the previous section, the PSO-TP returns the vectorial position $\mathbf{x}_{i+1}$ of the PSO Marker. The kinematic controllers use the error between this position and the robots current position $\mathbf{x}_i$ to compute the control signals. Four different controllers were tested in the simulations to determine which one gave the robotic swarm the best performance in terms of convergence velocity, trajectory smoothness and smooth velocities (less saturation and hard stops of the robot actuators). The four implemented kinematic controllers were the following:

### A. Transformed Unicycle controller (TUC)

It was structured by coupling the proportional controller shown in (7) with the diffeomorphism equations shown in (5) and (6) to compute the robot's linear and angular velocities. In this case, the saturation constant used in (7) was arbitrarily set to $I = 2$, and the proportional scaling factor $k$ was calculated as follows:

$$k = \frac{1 - e^{-2 \cdot ||\mathbf{e}||}}{2 \cdot ||\mathbf{e}||} \tag{15}$$

### B. Transformed Unicycle with LQR controller (TUC-LQR)

It was structured by coupling the LQR controller shown in (8) with the diffeomorphism equations shown in (5) and (6). Since the diffeomorphism assumes $\dot{\mathbf{x}} = \mathbf{u}$, the LQR gain matrix $\mathbf{K}$ was computed using $\mathbf{A} = \mathbf{0}$, $\mathbf{B} = \mathbf{R} = \mathbf{I}$, and $\mathbf{Q} = 0.1 \cdot \mathbf{I}$. In this case $\mathbf{x}_g = \mathbf{x}_{i+1}$, $\mathbf{x} = \mathbf{x}_i$ and $\mathbf{u}_g = 0$.

### C. Transformed Unicycle with LQI controller (TUC-LQI)

It was structured by coupling the LQI controller shown in (9) with the diffeomorphism equations shown in (5) and (6). The LQI gain matrices $\mathbf{K}$ and $\mathbf{K}_i$ were computed using:

$$\tilde{\mathbf{A}} = \begin{bmatrix} \mathbf{A} & \mathbf{0} \\ -\mathbf{C} & \mathbf{0} \end{bmatrix};\ \tilde{\mathbf{B}} = \begin{bmatrix} \mathbf{B} \\ \mathbf{0} \end{bmatrix} \tag{16}$$

$$\mathbf{Q}_{4\times4} = \mathbf{I};\ \mathbf{R}_{2\times2} = 2000 \cdot \mathbf{I} \tag{17}$$

where diffeomorphism matrices are $\mathbf{A} = \mathbf{0}$ and $\mathbf{B} = \mathbf{I}$. An output matrix $\mathbf{C} = \mathbf{I}$ is used since the GPS sensors allow the direct tracking of the robots' current state $\mathbf{x}_i$. The integral error $\mathbf{z}_i$ is computed by the integration of the difference between the robot's current position $\mathbf{x}_i$ and the swarm's global best position $\mathbf{p}_g$. This was done to add certain stability to the actuator velocities of the robot by adding continuous movement towards the goal. A new proportional control damping coefficient $b_p \in (0, 1]$ is introduced to diminish sudden actuator velocity spikes caused by the PSO Marker position updates. An integral control damping coefficient $b_i \in (0, 1]$ is also presented as a way of reducing robot position oscillations once the goal is reached. The LQI equations are rewritten as follows:

$$\mathbf{u} = -\mathbf{K} \cdot (1 - b_p) \cdot (\mathbf{x}_i - \mathbf{x}_{i+1}) - \mathbf{K}_I \cdot \mathbf{z}_i \tag{18}$$

$$\mathbf{z}_{i+1} = (1 - b_i) \cdot [\mathbf{z}_i + (\mathbf{p}_g - \mathbf{x}_i) \cdot \Delta t] \tag{19}$$

where damping coefficient values were set to $b_p = 0.95$ and $b_i = 0.01$.

### D. Lyapunov-stable Pose Controller (LSPC)

This controller used the control laws presented in (12) and (13) to calculate the linear and angular velocity of the E-Puck robots directly. In this case, position and orientation errors between the goal and the robot's location were mapped to polar coordinates as follows:

$$\rho = \sqrt{e_x^2 + e_y^2} \tag{20}$$

$$\alpha = -\theta + atan2(e_y, e_x) \tag{21}$$

where $\alpha$ belongs to the interval [-$\pi$,$\pi$]. The controller's parameters in (12) and (13) were set to: $k_\rho = 0.01$ and $k_\alpha = 0.5$. If $\alpha$ pointed at the left quadrants of the XY plane, the sign of the linear velocity was inverted.

## V. PSO-TP Simulation Results

Each of the four executed simulations used a different kinematic controller to compare their performance results while tracking the trajectory computed by the PSO-TP. The selected PSO-TP parameters for each case are the following:

TABLE I
PSO-TP configuration depending on kinematic controller

| **Kinematic Controller** | PSO-TP configuration parameters | |
|---|---|---|
| | $P_s$ (iterations) | $\eta$ |
| TUC | 1 | 0.625 |
| TUC-LQR | 5 | 0.250 |
| TUC-LQI | 1 | 0.250 |
| LSPC | 5 | 0.250 |

$P_s$ was set to 5 iterations for the TUC-LQR and LSPC to give the actuators 160ms between velocity shifts since these controllers showed rapid velocity spikes caused by the PSO Marker update. Scaler $\eta$ was set to $0.25$m for the TUC-LQR, TUC-LQI, and LSPC to ensure that the PSO Marker did not surpass the 2x2m search space's boundaries. The TUC controller used $\eta = 0.625$m to stretch the distance between the robot and the PSO Marker with the intention of generating more ample trajectories to minimize early E-Puck collisions caused by the fast controller.

### A. Trajectory smoothness

The E-Pucks were located at different distances and orientations from the goal to vary initial conditions for each robot. Trajectories of the 10 E-Puck robots reaching the goal were plotted to evaluate smoothness of the paths. Figures 2-5 show the trajectory results for all four kinematic controllers. The 2x2m grid represents the problem space, the blue plots are individual E-Puck trajectories, the red dots are initial positions of each E-Puck robot, and the black dot at (0,0) is the absolute minima of the Sphere fitness function.

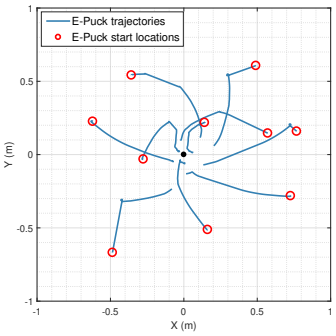


Fig. 2. Simulation with TUC

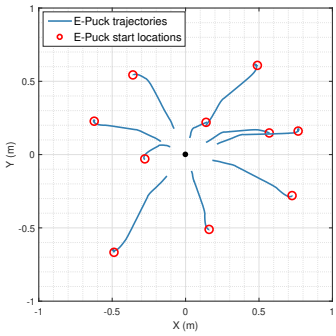


Fig. 3. Simulation with TUC-LQR

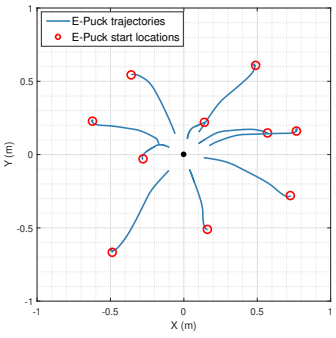


Fig. 4. Simulation with TUC-LQI

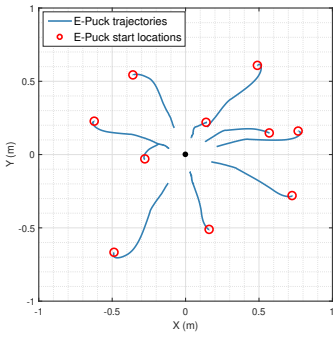


Fig. 5. Simulation with LSPC

The controllers that generated straighter paths towards the goal were the TUC-LQR and the TUC-LQI. The LSPC generated slightly smoother curved trajectories. The TUC controller resulted in more irregular trajectories but it achieved the fastest convergence time (7s). Convegence time for the LSPC and TUC-LQR was the same (25s). Lastly, the TUC-LQI converged in 20s. These three last controllers were considerably slower than the TUC but generated more regular trajectories.

### B. Control smoothness

Measuring the smoothness of the control signals applied to the robot's actuators was also key to evaluate each controller's performance. Firstly, cubic spline interpolations of the resultant actuator velocity curves ($\Phi_R$ and $\Phi_L$) were performed for each controller. Cubic splines have a minimum energy property based on the Euler-Bernoulli beam theory, as shown in [13]. Consequently, the smoothness of a resultant velocity curve can be measured by calculating the minimum energy $W$ required to bend an elastic thin beam to shape it as its interpolated cubic spline $y(x)$. Greater values of $W$ indicate less smooth velocity curves corresponding to aggressive controllers, whereas reduced $W$ values indicate control smoothness. A controller with reduced $W$ is desired to avoid violent velocity changes in the robot's actuators. Wheel velocity data from both actuators of each E-Puck robot in the swarm was collected during the simulations of all four controllers. The bending energy $W$ of a total of 20 interpolated control curves (two actuator velocities for each of the 10 E-Pucks) was calculated for each controller using:

$$W = \frac{1}{2}\int_{x_0}^{x_n} y''(x)^2 dx \tag{22}$$

The next figure shows the smoothness results for all velocity curves generated with each controller:

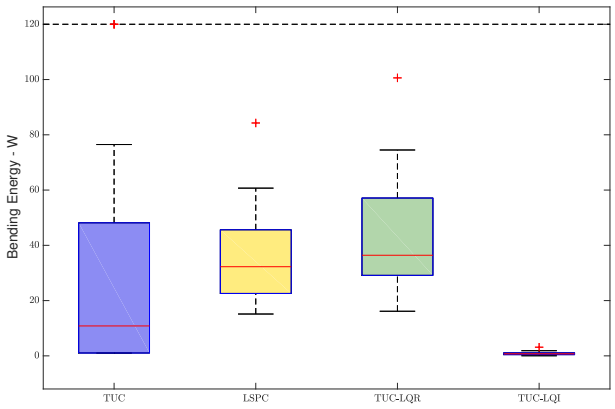


Fig. 6. Control smoothness calculated with bending energy

As shown in Fig.6, the controller that statistically presented the lowest bending energy results (greater smoothness) was the TUC-LQI. The rest of the controllers showed similar distributions of smoothness. The TUC showed the second lowest average bending energy, but the data presented greater distribution indicating strong variation of the control smoothness due to different initial starting conditions of the E-Pucks.

### C. Control saturation rate

Another important aspect of the controllers' performance is the rate of actuator saturation caused by it. Every control

curve generated by each controller was analyzed to determine the ratio between the time that the actuators were saturated at $\pm 6.28$ rad/s and the whole duration of the simulation. The TUC presented actuator saturation between 50% and 90% of the time, whereas the LSPC, TUC-LQR and TUC-LQI presented no saturation whatsoever.

### D. Further testing of the PSO-TP with TUC-LQI controller

The implementation of the PSO-TP using the TUC-LQI controller was also tested using a modified Keane Benchmark function (23) that had two absolute minima at $\mathbf{x}^* = [\pm 0.7, 0]$. The swarm managed to converge to the absolute minima at $\mathbf{x}^* = [+0.7, 0]$ since a robot was closer to that minima influencing $\mathbf{p}_g$ in (1). The smoothness of the control signals did not vary compared to previous tests. The resulting trajectories are shown in Fig.7.

$$f(x,y) = -\frac{\sin^2(2x-y)\sin^2(2x+y)}{\sqrt{2x^2+y^2}} \tag{23}$$

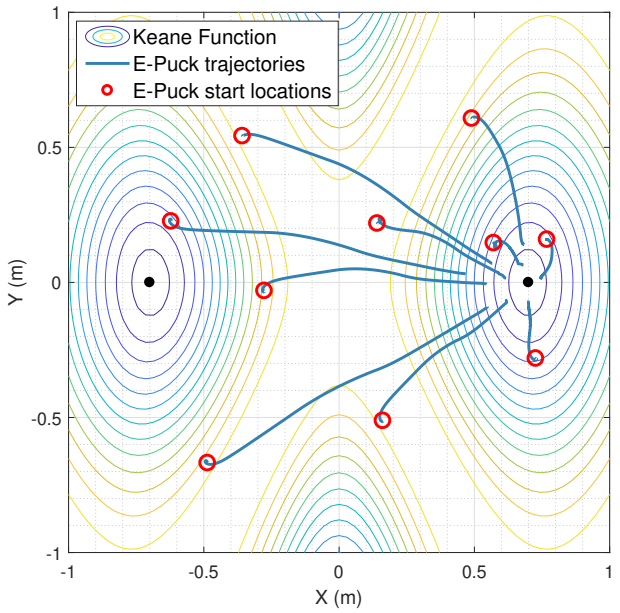


Fig. 7. Swarm trajectories in Keane benchmark function

## VI. Conclusions

The classic PSO parameters incorporated to the PSO-TP (inertia weight $\omega$, constriction factor $\varphi$ and scaling factors $c_1$ & $c_2$) were set to default values used in previous works and the algorithm converged with great accuracy to the goal in a finite amount of time.

The implementation of the PSO Velocity scaler $\eta$ allowed an easier manipulation of the operating range of the PSO-TP. It effectively restricted the positions of the PSO Marker so that they were always within the search space.

The TUC controller presented the fastest convergence time, but it generated the most irregular paths while tracking the PSO Marker. It also presented a high percentage of control saturation which would complicate its implementation in real robots with actuators that may be damaged by this.

The controllers that led to straighter and less curved paths towards the goal in the problem space were the TUC-LQR and TUC-LQI controllers. However, the TUC-LQI performed significantly better in terms of generating the most smooth actuator velocities, with no saturation, no velocity change spikes thanks to the proportional control damper $b_p$, and no oscillations when the goal was reached thanks to the integral control damper $b_i$. The LSPC performed similarly to the TUC-LQR in every sense.

The PSO Trajectory Planner was able to guide a differential robot swarm towards a goal in a finite amount of time generating smooth trajectories, and its implementation using the TUC-LQI kinematic controller ensured smooth control signals with reduced hard stops and no actuator saturation.

## References


[1] M. Bonani, "Epfl e-puck robot." http://www.e-puck.org/. Accessed: 2019-04-29.

[2] K. M. Lynch and F. C. Park, *Modern Robotics*. Cambridge University Press, 2017.

[3] J. Kennedy and R. Eberhart, "Particle swarm optimization," pp. 1–7, IEEE, 1995.

[4] R. C. Eberhart and Y. Shi, "Comparing inertia weights and constriction factors in particle swarm optimization," in *Proceedings of the 2000 congress on evolutionary computation. CEC00 (Cat. No. 00TH8512)*, vol. 1, pp. 84–88, IEEE, 2000.

[5] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *1998 IEEE international conference on evolutionary computation proceedings. IEEE world congress on computational intelligence (Cat. No. 98TH8360)*, pp. 69–73, IEEE, 1998.

[6] J. C. Bansal, P. Singh, M. Saraswat, A. Verma, S. S. Jadon, and A. Abraham, "Inertia weight strategies in particle swarm optimization," in *2011 Third world congress on nature and biologically inspired computing*, pp. 633–640, IEEE, 2011.

[7] T. Beielstein, K. E. Parsopoulos, and M. N. Vrahatis, *Tuning PSO parameters through sensitivity analysis*. Universitätsbibliothek Dortmund, 2002.

[8] F. Martins and A. Brandão, "Velocity-based dynamic model and control." https://bit.ly/32s6jRX, 2018. Accessed: 2019-05-30.

[9] T. K. Priyambodo, "Integral modified linear quadratic regulator method for controlling lateral movement of flying wing in rotational roll mode," in *Journal of Engineering and Applied Sciences*, pp. 463–471, IEEE, 2018.

[10] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, *Introduction to autonomous mobile robots*. MIT press, 2011.

[11] S. K. Malu and J. Majumdar, "Kinematics, localization and control of differential drive mobile robot," *Global Journal of Research In Engineering*, 2014.

[12] Webots, "http://www.cyberbotics.com." Commercial Mobile Robot Simulation Software.

[13] G. Wolberg and I. Alfy, "An energy-minimization framework for monotonic cubic spline interpolation," *Journal of Computational and Applied Mathematics*, vol. 143, no. 2, pp. 145–188, 2002.