Machine Learning Based Kinematic Control of Different Drive Mobile Robots

Manoj Rajagopalan
CS229 Autumn 2023 Final Project

Abstract

We present two approaches - model-based and model-free - to learning a kinematic controller that helps differential drive mobile robots overcome a variety of defects that affect its ability to follow a reference trajectory. The model-based approach solves a Linear Regression problem using Stochastic Gradient Descent to learn dynamical-model-parameters, and then applies them to calculate adaptations to the reference control policy to restore robot operation. The model-free approach uses a fully-connected neural network to directly learn adapted values of the reference control policy without requiring any dynamical model. In both cases, we implement a kinematic controller that incorporates the learning and restores near-correct operation for a variety of defect patterns. We show that by generating a dataset containing a small number of figure-of-8 trajectories (various radii and speeds) we complete learning in only a few seconds, allowing our procedures to be personalized to each robot-instance's unique deviation from factory/design and thereby improving longevity.

Introduction

Differential drive mobile robots (**DDMR**) are the simplest wheeled robots. Fig. 1 shows a schematic construction and the coordinate system used. Three wheels provide mobility; two identical wheels (left and right) share the same axis and are independently powered (by motors) while the third is usually a passive (castor) wheel. Combinations of angular velocities of the wheels (φ_l and φ_r) generate a variety of motions: identical values induce straight line motion; equal and opposite values induce rotation-in-place; other combinations generate various curves. The wheels are separated by a distance (baseline) of $2L$. The center-of-mass (CoM, also the origin of the robot's local coordinate frame $\{x_R, y_R\}$) is lies on the wheels' axis at the mid-point; its coordinates in the inertial (global) frame are denoted (x,y). The wheel radii are R_l and R_r respectively; typically they are the same. The CoM's longitudinal velocity is denoted v, and the robot's angular velocity about the center-of-mass is denoted ω.

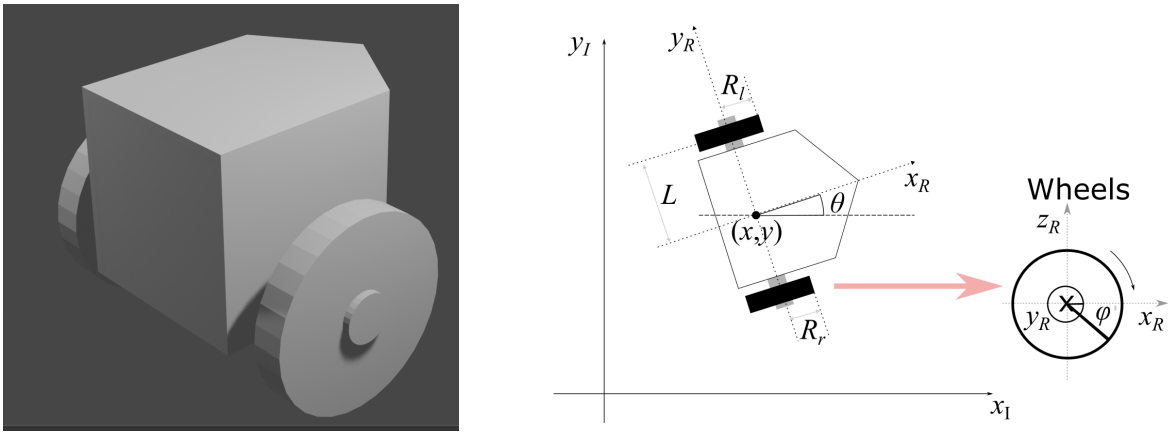


Fig. 1: 3D and schematic view of the DDMR (latter showing both Inertial and Robot coordinate frames as well as important geometrical parameters defining the system).

When a DDMR receives new destination coordinates, its mobility logic typically generates a ***Path Plan*** first, based on a pre-generated map and knowledge of its current pose (***Localization***). The path plan is a just sequence of (x,y) coordinates along the path. DDMR systems have the special property of being ***Differentially Flat*** ; this allows direct pre-computation (***open-loop***) of a ***Reference Trajectory*** from the

path-plan. The trajectory is a time-series of control inputs and expected output states (position, orientation etc.)) for the entire path. The reference trajectory is then fed into a low-level (feedback) controller which typically runs at 10 Hz or higher, commanding the left and right wheel motor angular velocities to achieve the reference trajectory.

The reference trajectory is typically expressed in the ***Body Frame*** as a sequence of pairs - longitudinal velocity (v) and angular velocity (ω) - at each time-step. This is independent of the construction of the robot. Lower-level controllers must translate the desired v and ω to the $\dot{\varphi}_l$ and $\dot{\varphi}_r$ to be commanded at respective wheels. Such calculations typically require accurate knowledge of robot geometry - L, R_l, R_r. However, fabrication defects, environmental noise, and wear-and-tear change these parameters (over time) distorting robot operation and reducing operational longevity.

In this work, we explore two approaches for learning low-level controllers that translate desired body-frame controls (v, ω) into actual wheel-frame controls ($\dot{\varphi}_l, \dot{\varphi}_r$) for a robot's specific defect-pattern. The first is a (model-based) ***System Identification*** approach - we assume a dynamical mathematical model and learn its parameters from data. The second is a model-free approach which bypasses System Identification completely, and directly learns the to predict ($\dot{\varphi}_l, \dot{\varphi}_r$) from ($v, \omega$) input. If successful, this will pave the way for robots to self-heal in remote environments.

Related work

[1] demonstrate a kinematic controller based on RNN with a single hidden layer (tansigmoidal activation) to track a reference trajectory (make the error zero). [2] do similar with a multi-layer perceptron architecture. [3] solve the same trajectory-tracking problem by expressing it as Model Predictive Control and solving the associated Quadratic Programming problem using Varying Parameter Convergent Differential Neural Networks (VPCDNN). [4] perform system identification, to relate electric motor drive currents (inputs) to the linear and angular velocities, using a 100-activation-hidden-layer feed-forward neural network.

Dataset

All our work is done in simulation.

To generate our training set, we first tele-operate our robots (as in realistic scenarios) to perform figure-of-8 patterns with different radii and speeds to cover the input space. We tabulate commanded inputs (wheel-frame controls - $\dot{\varphi}_l$ and $\dot{\varphi}_r$) with corresponding (noisy) measurements outputs (state: x, $y, \theta, \varphi_l, \varphi_r$ and speeds: v, ω). We inject Gaussian noise into all our simulated measurements.

We perform the above on 6 categories test robots each with a unique defect pattern - (1) smaller left wheel, (2) larger left wheel, (3) smaller baseline, (4) larger baseline, (5) "noisy": left wheel with one localized protrusion, and (6) "noisier": both wheels have multiple protrusions of different angular widths and different locations. Appendix B describes the geometry for each. Fig. 2 shows the figures-of-8 for the Noisier robot; its right side shows noisy measurements (left is ground-truth) which are inputs to our learning algorithm. We use similar patterns for the other 5 robots as well (Appendix C). The number of training examples for each robot is:

SmallerLeftWheel: 203	LargerLeftWheel: 144	Noisy: 173
SmallerBaseline: 115	LargerBaseline: 173	Noisier: 115

We present two learning approaches. In ***System-ID based learning***, our inputs are $(x, y, \theta, \varphi_l, \varphi_r, \dot{\varphi}_l, \dot{\varphi}_r)$ and our labels are (v, ω). We then implement a controller which accepts (v, ω) as inputs along with $(x, y, \theta, \varphi_l, \varphi_r)$ and outputs $(\dot{\varphi}_l, \dot{\varphi}_r)$ In ***model-free learning***, our inputs are $(x, y, \theta, \varphi_l, \varphi_r, v, \omega)$ and our labels are $(\dot{\varphi}_l, \dot{\varphi}_r)$. Note that robot pose (x, y, θ) does not affect either case; both approaches must learn to ignore these.

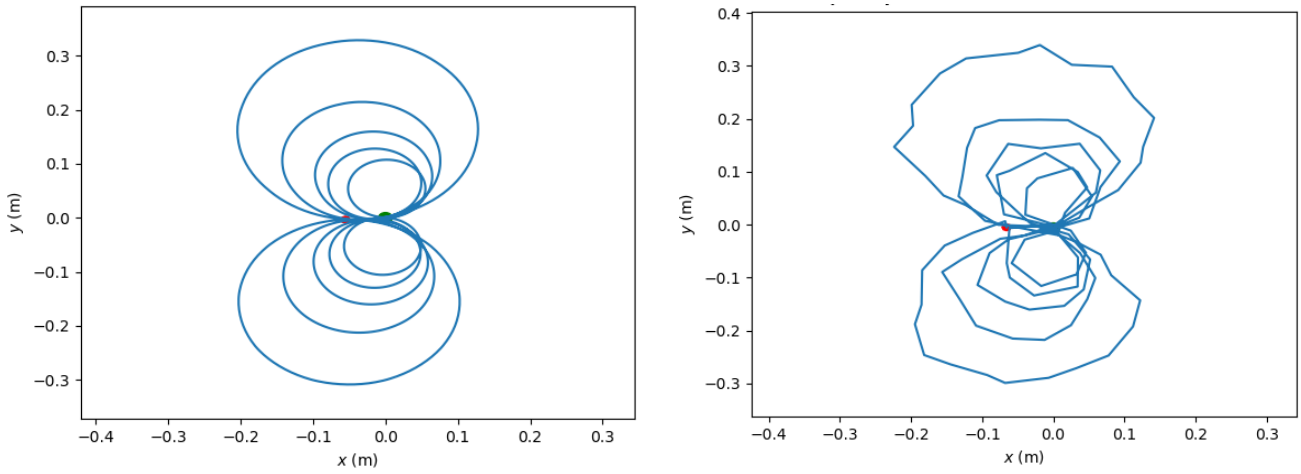


Fig. 2: Ground-truth dataset trajectory (left) and its measured variant (right) which includes Gaussian noise.

Methods

Model-Based Control via System Identification

In this approach, we adopt the following well-known dynamical mathematical model relating the two control frames, and learn the system parameters - L, R_l, R_r - that best fit our training examples.

$$\begin{bmatrix} v(t) \\ \omega(t) \end{bmatrix} = \frac{1}{2} \begin{bmatrix} R_r(\varphi_r(t)) & R_l(\varphi_l(t)) \\ R_r(\varphi_r(t))/L & -R_l(\varphi_l(t))/L \end{bmatrix} \begin{bmatrix} \dot{\varphi}_l(t) \\ \dot{\varphi}_r(t) \end{bmatrix} \quad (1)$$

We allow R_l, R_r to be functions of φ_l, φ_r respectively in order to capture distortions. Note the linearity in R_l, R_r and $1/L$, for any fixed wheel-orientation (φ_l - φ_r combination). For an ideal (perfectly cylindrical) wheel, R_l and R_r are constants, independent of φ_l and φ_r.

System definition: In order to learn the wheels' distortions, we discretize each wheel into N_φ equal arcs, each with its own radius. We learn the $(2N_\varphi + 1)$ parameters - $L, R_{l,n}, R_{r,n}, n \in [1 \,..\, N_\varphi]$ - for our model by performing Regression using Stochastic Gradient Descent. Note that for any given time-instant the two wheels are in 1 out of N_φ^2 combinations of $\varphi_l^{(n)}$ and $\varphi_r^{(n)}$.

Loss function: We allow each wheel to spin no faster than 8 rotations/second; this provides ~ 1 m/s linear speed for wheels of radius 2 cm. Because the associated $\dot{\varphi}$ can be as high as 50 radians/s, and ω lies in this scale, applying the simple MSE loss function the $[v\ \omega]^T$ (error) vector risks providing lower weightage to v. Therefore we adopt a "Mahalanobis" loss function, inspired by the eponymous distance measure,

$$l(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}) := || \begin{bmatrix} \Delta v^{(i)} \\ \Delta \omega^{(i)} \end{bmatrix} ||_{2,mah} := \frac{1}{2} \begin{bmatrix} \Delta v^{(i)} & \Delta \omega^{(i)} \end{bmatrix} \begin{bmatrix} \kappa^2 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \Delta v^{(i)} \\ \Delta \omega^{(i)} \end{bmatrix}$$

$$\begin{bmatrix} \Delta v^{(i)} \\ \Delta \omega^{(i)} \end{bmatrix} := \begin{bmatrix} \hat{v}^{(i)} \\ \hat{\omega}^{(i)} \end{bmatrix} - \begin{bmatrix} v^{(i)} \\ \omega^{(i)} \end{bmatrix} \tag{2}$$

$$[\hat{v}^{(i)} \hat{\omega}^{(i)}]^T := \mathbf{f}(\mathbf{x}^{(i)}; \zeta) \qquad \mathbf{x} := [x, y, \theta, \varphi_l, \varphi_r]^T \qquad \mathbf{y} := [v^{(i)} \omega^{(i)}]^T$$

where κ^2 can be thought of the ratio of variances for v and ω that make the scales comparable, and NN learns $\mathbf{f}$ is the function, with parameters ζ, that the neural network is learning. The gradients are,

$$\frac{\partial l}{\partial R_{l,n}} = \frac{\dot{\varphi}_l}{2} \left[\kappa^2 (\hat{v}^{(i)} - v^{(i)}) - \frac{(\hat{\omega}^{(i)} - \omega^{(i)})}{L} \right]$$

$$\frac{\partial l}{\partial R_{r,n}} = \frac{\dot{\varphi}_r}{2} \left[\kappa^2 (\hat{v}^{(i)} - v^{(i)}) + \frac{(\hat{\omega}^{(i)} - \omega^{(i)})}{L} \right] \tag{3}$$

$$\frac{\partial l}{1/L} = (\hat{\omega}^{(i)} - \omega^{(i)})(R_r \dot{\varphi}_r - R_l \dot{\varphi}_l)$$

Model-based kinematic controller: Once we learn the faulty robot's actual geometry - L, $R_{l,n}$, $R_{r,n}$, $n \in [1 \,..\, N_\varphi]$ - we implement a controller which inverts Eqn. (1) to calculate the instantaneous $\dot{\varphi}_l$, $\dot{\varphi}_r$ from the desired body-frame (v, ω). The measured state of the robot, available to the controller, includes the angular orientations - φ_l and φ_r - of the wheels; this marks the specific angular arc in each wheel that is making contact with the ground.

Model-Free Control via Neural Networks

In this approach we implement a fully connected neural network that accepts 7 input-fields - x, y, θ, φ_l, φ_r, v, ω - and predicts 2 output-fields - $\dot{\varphi}_l$, $\dot{\varphi}_r$. We use a fully-connected neural network with MSE loss and ADAM optimizer with L2 regularization. We tried a second approach where we strip out the x, y, θ in the inputs, making them only 4-wide.

Model-free kinematic controller: Since our network learns to predict $\dot{\varphi}_l$, $\dot{\varphi}_r$ from v, ω (and state), implementing a kinematic controller is simply a matter of concatenating the state and body-frame controls and running an inference cycle through the neural network. The output $\dot{\varphi}_l$, $\dot{\varphi}_r$ are used to command the wheels.

Results

We implement our approaches with about 1400 lines of Python3 code [GitHub link]. We performed trajectory simulation using `scipy.integrate.solve_ivp()` with a time-step of 10 ms (hence, at 100 Hz). To mimic real-world measurement, we decimated simulation results by sampling at 100 ms intervals (hence, at 10 Hz). We add Gaussian noise to the trajectories being used as datasets (multi-figure-of-8 pattern). Programs were run on a Macbook Pro with 2.9 GHz 6-core Intel i9 CPU, 32 GB RAM, MacOS Sonoma v14.1.

System-identification based kinematic control

We ran SGD for 1000 epochs with a batch-size of 20. We used a learning rate, $\alpha=10^{-4}$. We found higher values to introduce numerical instability. We identified the hyperparameters by trial-and-error.

The associated total runtime is 2-4 seconds. **Appendix D** shows the performance of each robot with a kinematic controller that translates desired (v,ω) to wheel $(\dot{\varphi}_l, \dot{\varphi}_r)$.

- Empirically, we found κ^2 = 10 to be a good value.
- Shuffling the dataset trajectory before SGD significantly improved the quality of result.
- All 6 ideal trajectories are reproduced very well; the maximum deviation at any point < 0.001 m.
- However, the system is not identified correctly; both baseline and wheel-radii (variation with angle) are identified incorrectly. However, errors introduced by one are compensated by the other; the observable behavior of the system is very good. This indicates some ambiguity in the dynamical model.

Model-free kinematic control

We implemented a fully-connected neural network, with the structure shown to the right, in PyTorch. This is a 4-layer network. We initialized 1D-tensor parameters random values from the canonical Normal distribution. For higher-dimensional tensors we adopted Xavier-uniform initialization. We found, by trial-and-error, the following hyperparameter values to work well.

- learning rate, α=0.001
- batch-size = 50
- number of epochs = 2000

```
nn.Sequential(nn.Linear(7, 20)
              nn.ReLU(),
              nn.Linear(20,40)
              nn.Sigmoid(),
              nn.Linear(40,20)
              nn.ReLU(),
              nn.Linear(20,2))
```

Appendix E presents the trajectories traversed by our 6 test robots, with a kinematic controller that uses the above NN (trained) to translate desired body-frame controls to robot-specific wheel-frame values. Even though the values of loss were smallest for the above configuration in our experiments, we observe that the performance of the controller is still very far from satisfactory. However, we note that there has been systematic improvement relative to simpler hyperparameter configurations we explored:

- 2- and 3-layer networks
- 20, 30 hidden units/layer
- number of epochs varied across 100 and 1000
- α = 0.01

Conclusion/Future Work

While system-identification-based kinematic control performs observably well, the internal ambiguities cast doubt on its reliability. Our datasets are significantly small; this definitely affects our results adversely. The significantly poor performance of the NN-based approach (high bias situation) is confirmation. But this is also an indication that our 6 reference trajectories (straight, spin, circle-CW, circle-CCW, figure-of-8, and tri-wave-φ) provide reasonably good test data, at least for NN training (though they may not be sufficient). The system-identification approach is light-weight and offers a fast controller; the model-free approach is expectedly slower in training but its slower inference challenges its suitability for real-time control.

References

[1] Velagic J, Osmic N, Lacevic B, “Neural Network Controller for Mobile Robot Motion Control”, Int. J. Electrical & Computer Engg., 3:7 2008.

[2] Al-Shibaany ZY, Hedley J, Bicker R, "Design of an adaptive neural kinematic controller for a National Instrument mobile robot system," 2012 IEEE International Conference on Control System, Computing and Engineering, Penang, Malaysia, 2012, pp. 623-628, doi: 10.1109/ICCSCE.2012.6487220.

[3] Hu Y, Su H, Zhang L, Miao S, Chen G, Knoll A. “Nonlinear model predictive control for mobile robot using varying-parameter convergent differential neural network”. Robotics. 2019 Jul 31;8(3):64.

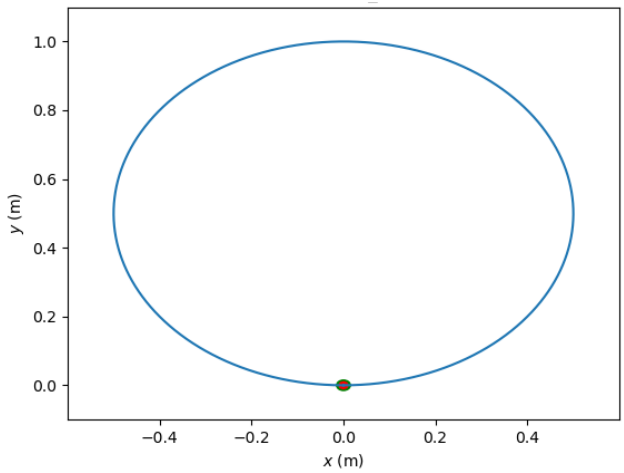
[4] Khan MA, Baig D-e-Z, Ali H, Ashraf B, Khan S, Wadood A, Kamal T, “Efficient System Identification of a Two-Wheeled Robot (TWR) Using Feed-Forward Neural Networks”. Electronics. 2022; 11(21):3584. https://doi.org/10.3390/electronics11213584

Appendix A: Ideal robot geometry and test trajectories

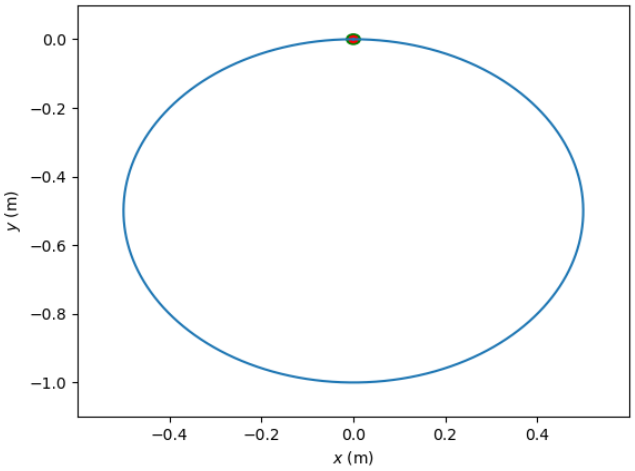
Our perfect, reference DDMR has the following geometry:

- baseline: 10 cm
- wheel radii: 2 cm each

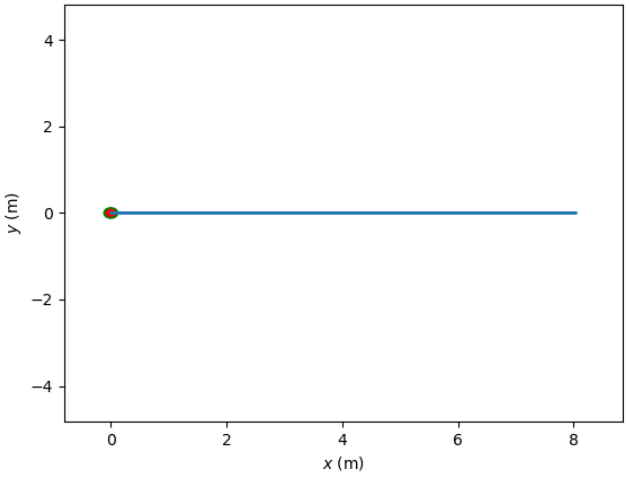
Fig. A1 and Fig. A2 show our test-set: a set of paths, with varying complexity, traversed by this Ideal robot. Appendix B shows the extent to which our faulty robots deviate from the above paths. Our learning-based controllers must restore that aberrant behavior to what we see here, above.



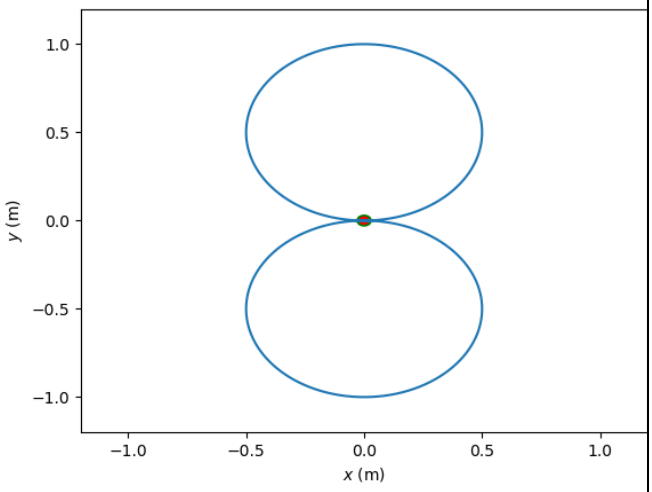
(a) Counter-clockwise loop



(b) Clockwise loop

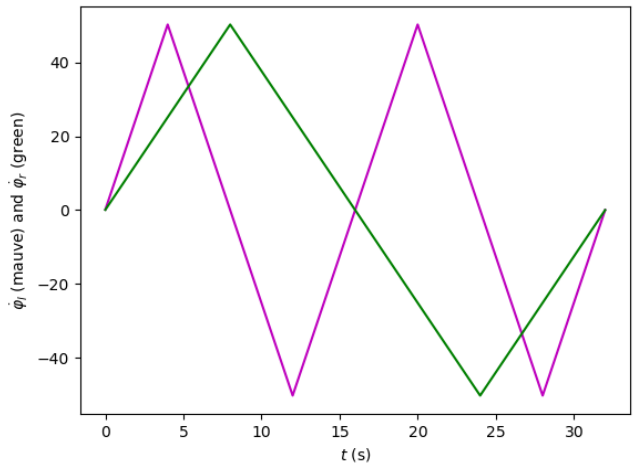


(c) Straight line

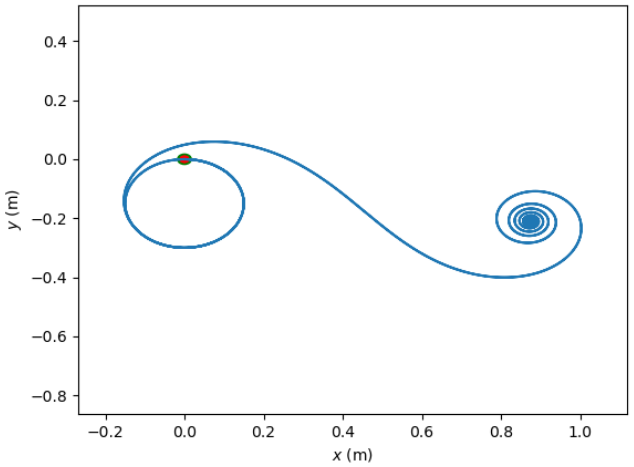


(d) Figure-of-8

Fig. A1: Various focussed test paths generated by our ideal robot. Red dot shows origin, green dot shows destination. They coincide in all the above.



(e) "Tri-wave" wheel-frame control policy



(f) Path generated by the "tri-wave" wheel-frame control policy

Fig. A2: A special control policy created to exercise various combinations of control values, and test path traversed by our ideal robot by executing this policy. Red dot shows origin, green dot shows destination. They coincide in all the above.

Appendix B: Test robots

Geometrical details for our 6 test robots are presented below. Recall that the Ideal robot has a baseline of 10cm and wheel radius of 2 cm.

1. **SmallerLeftWheel**: like Ideal but with left-wheel-radius of 1.5 cm.
2. **LargerLeftWheel**: like Ideal but with left-wheel-radius of 2.5 cm.
3. **SmallerBaseline**: like Idea but with baseline of 8 cm.
4. **LargerBaseline**: like Idea but with baseline of 12 cm.
5. **Noisy**:
    a. baseline = 12 cm
    b. left wheel radius = 2 cm
        i. left wheel has one 0.5 cm protrusion spanning 35° out of 360°.
    c. right wheel radius = 2cm
6. **Noisier**:
    a. baseline = 8 cm
    b. left wheel radius = 2 cm
        i. one protrusion of 5 cm spanning 20° out of 360°.
    c. right wheel radius = 2 cm
        i. one flattening by 0.4 cm (i.e. effective radius = 1.6 cm) spanning 10° out of 360°.
        ii. one flattening by 0.2 cm (i.e. effective radius = 1.8 cm) spanning 10° out of 360°.

Appendix B1: Uncorrected performance of SmallerLeftWheel robot

Fig. B1a and Fig. B1b below show how a robot with a smaller left wheel (1.5 cm radius; right wheel 2.0 cm radius; baseline 10 cm) performs when subjected to the wheel-frame control policies from the ideal robot *without any controller*. We see that the deviations are significant.

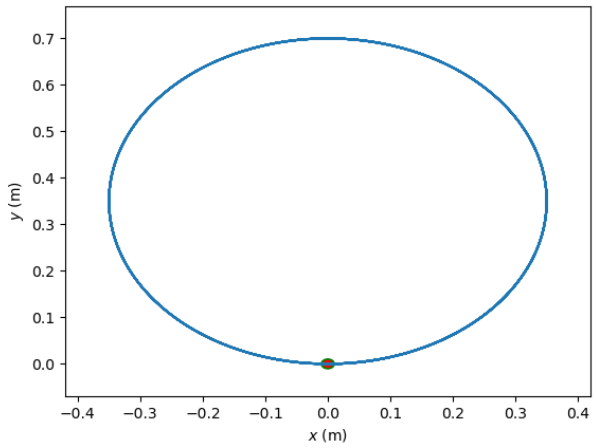
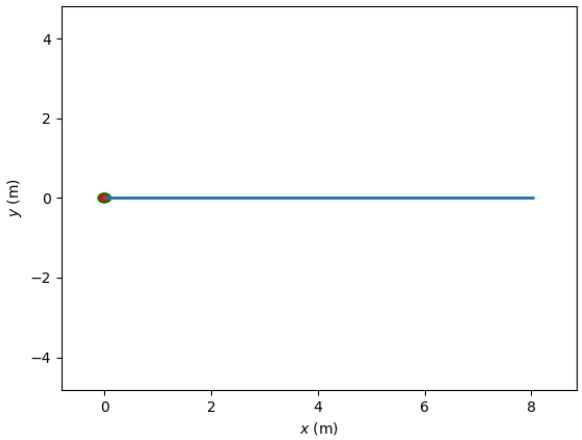
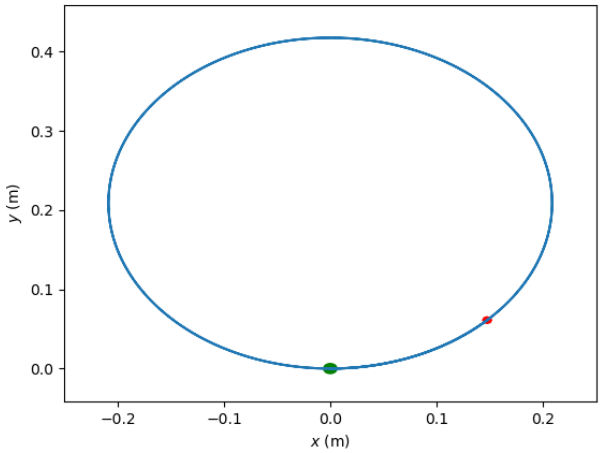
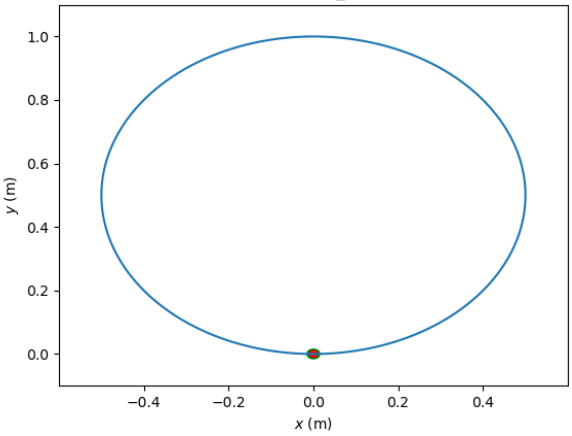
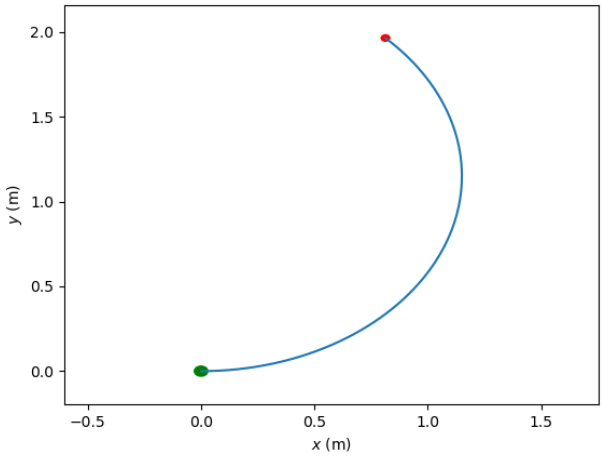
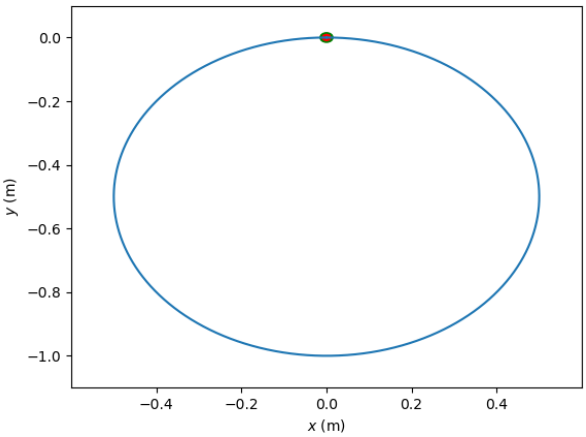
	Faulty robot	Ideal robot
Straight		
Circle-CCW		
Circle-CW		

Fig. B1a: Uncorrected performance of SmallerLeftWheel robot when subject to some of the ideal robot's wheel-frame control policies.

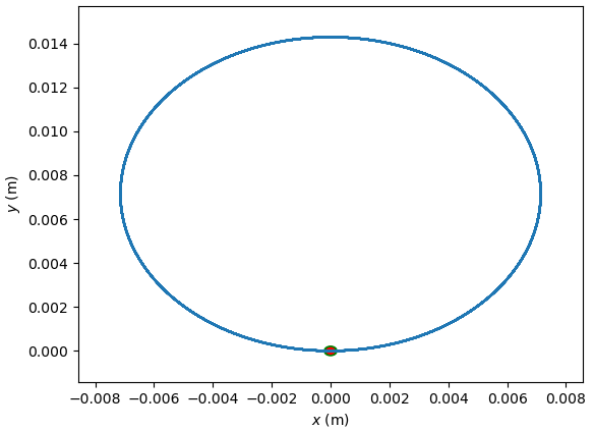
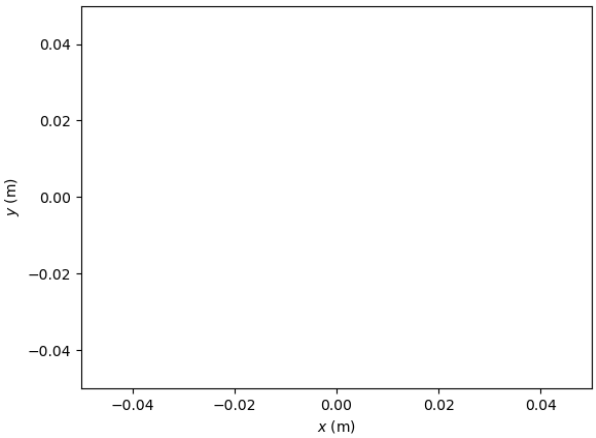
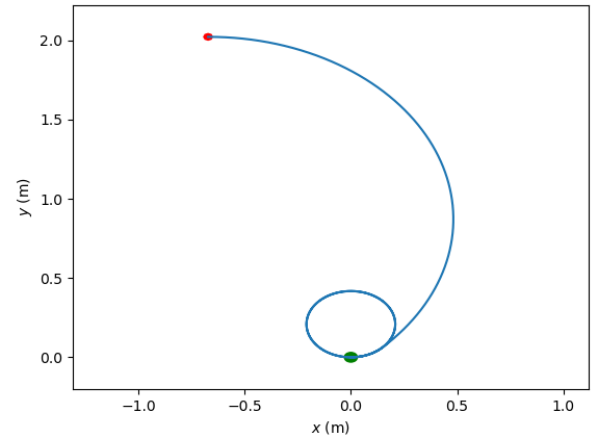
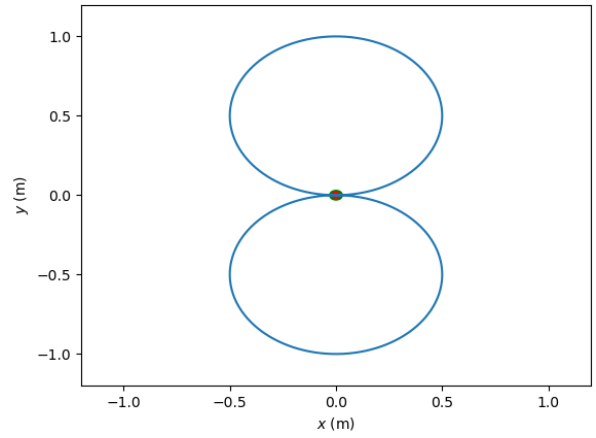
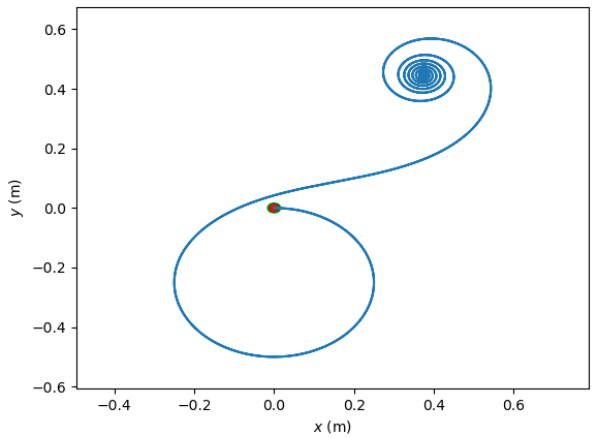
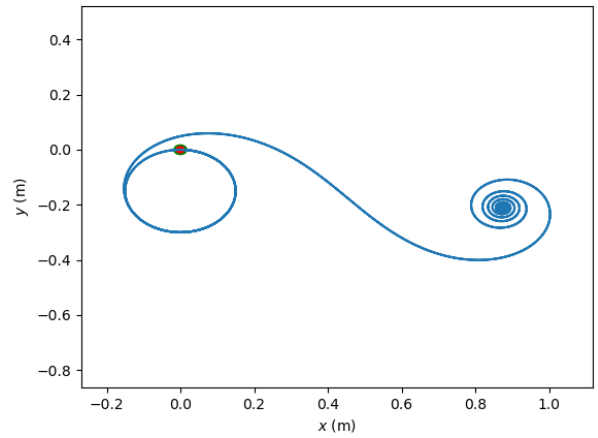
	Faulty robot	Ideal robot
Spin in-place		 (intentionally blank: no net motion)
Figure-of-8		
"Tri-wave" φ		

Fig. B1b: Uncorrected performance of SmallerLeftWheel robot when subject to the remainder of the ideal robot's wheel-frame control policies.

Appendix B2: Uncorrected performance of LargerLeftWheel robot

Fig. B2a and Fig. B2b below show how a robot with a larger left wheel (2.5 cm radius; right wheel 2.0 cm radius; baseline 10 cm) performs when subjected to the wheel-frame control policies from the ideal robot. We see that the deviations are significant.

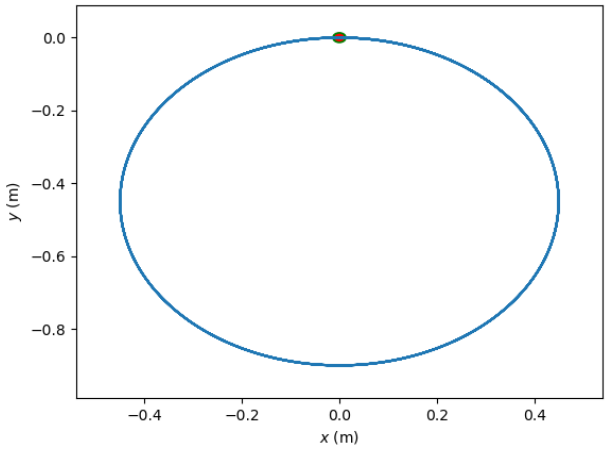
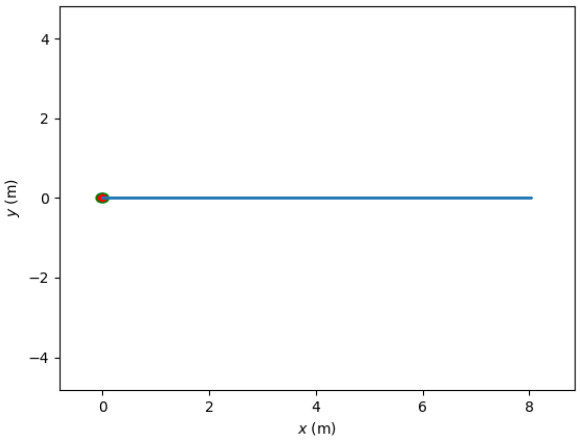
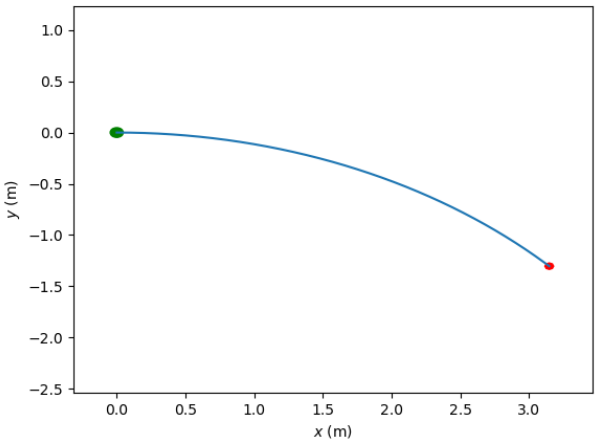
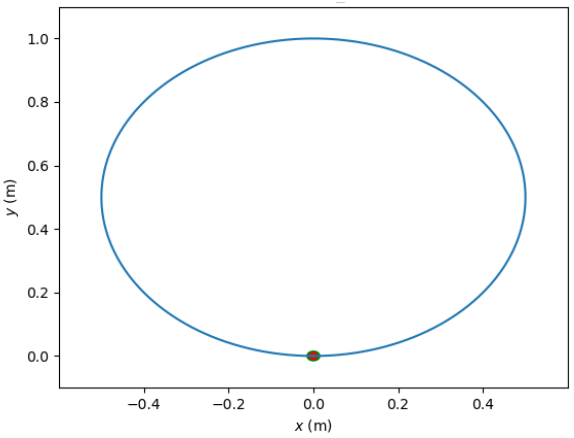
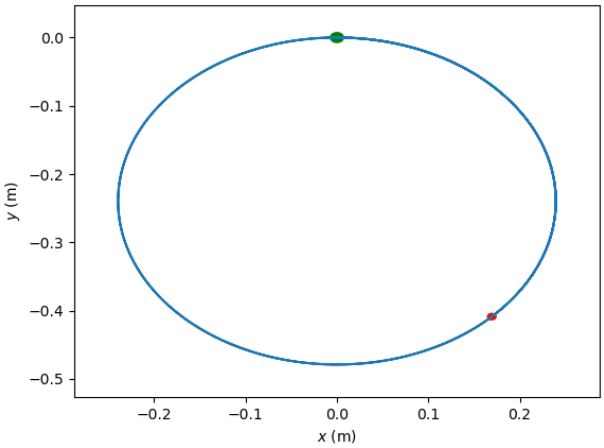
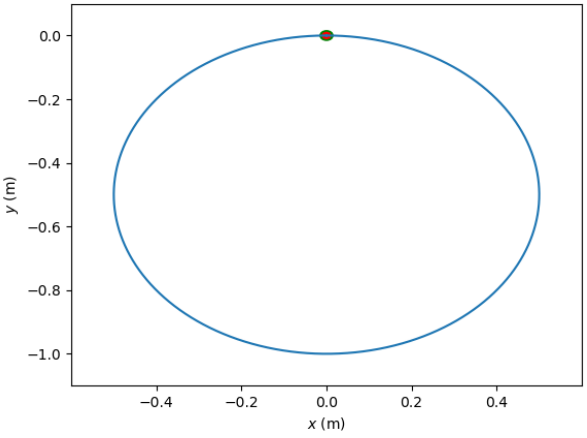
	Faulty robot	Ideal robot
Straight		
Circle-CCW		
Circle-CW		

Fig. B2a: Uncorrected performance of LargerLeftWheel robot when subject to some of the ideal robot's wheel-frame control policies.

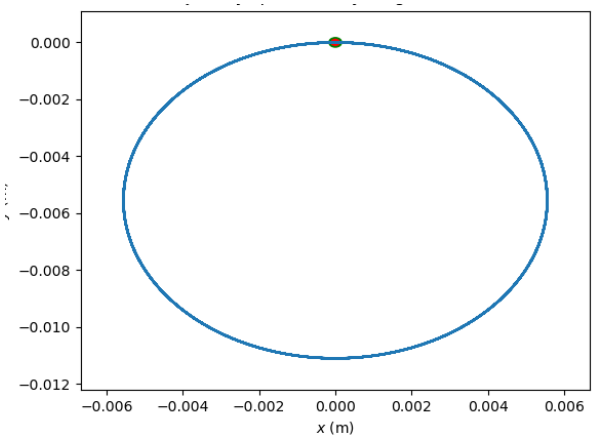
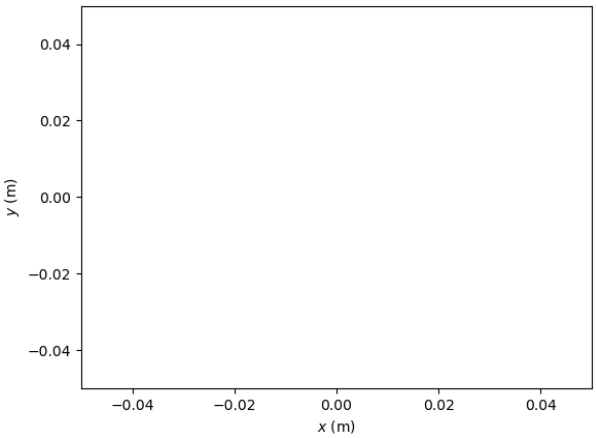
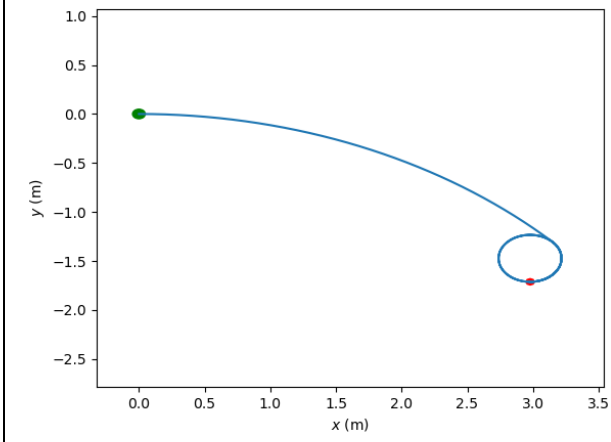
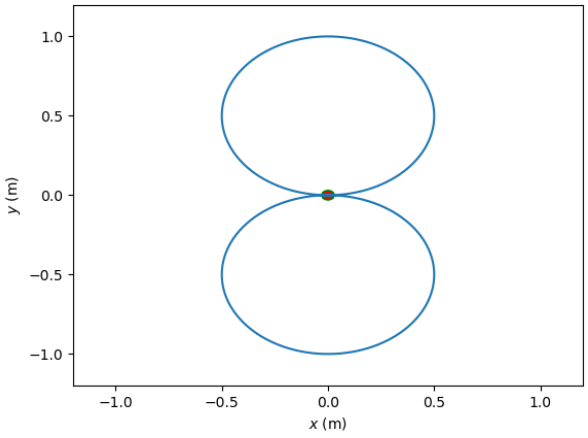
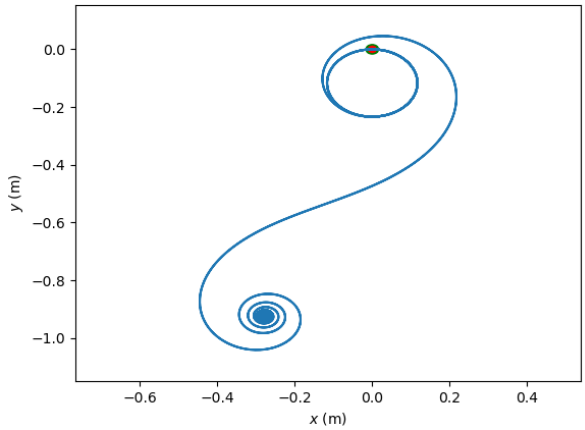
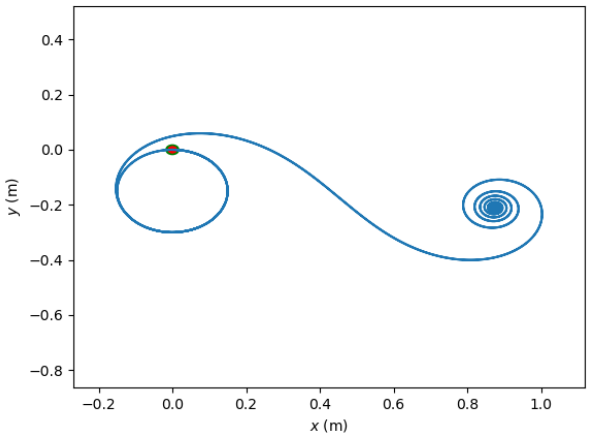
	Faulty robot	Ideal robot
Spin in-place		 (intentionally blank: no net motion)
Figure-of-8		
"Tri-wave" φ		

Fig. B2b: Uncorrected performance of LargerLeftWheel robot when subject to the remainder of the ideal robot's wheel-frame control policies.

Appendix B3: Uncorrected performance of SmallerBaseline robot

Fig. B3a and Fig. B3b below show how a robot with a smaller baseline (8 cm instead of 10 cm; wheels are 2.0 cm radius) performs when subjected to the wheel-frame control policies from the ideal robot. We see that the deviations are significant.

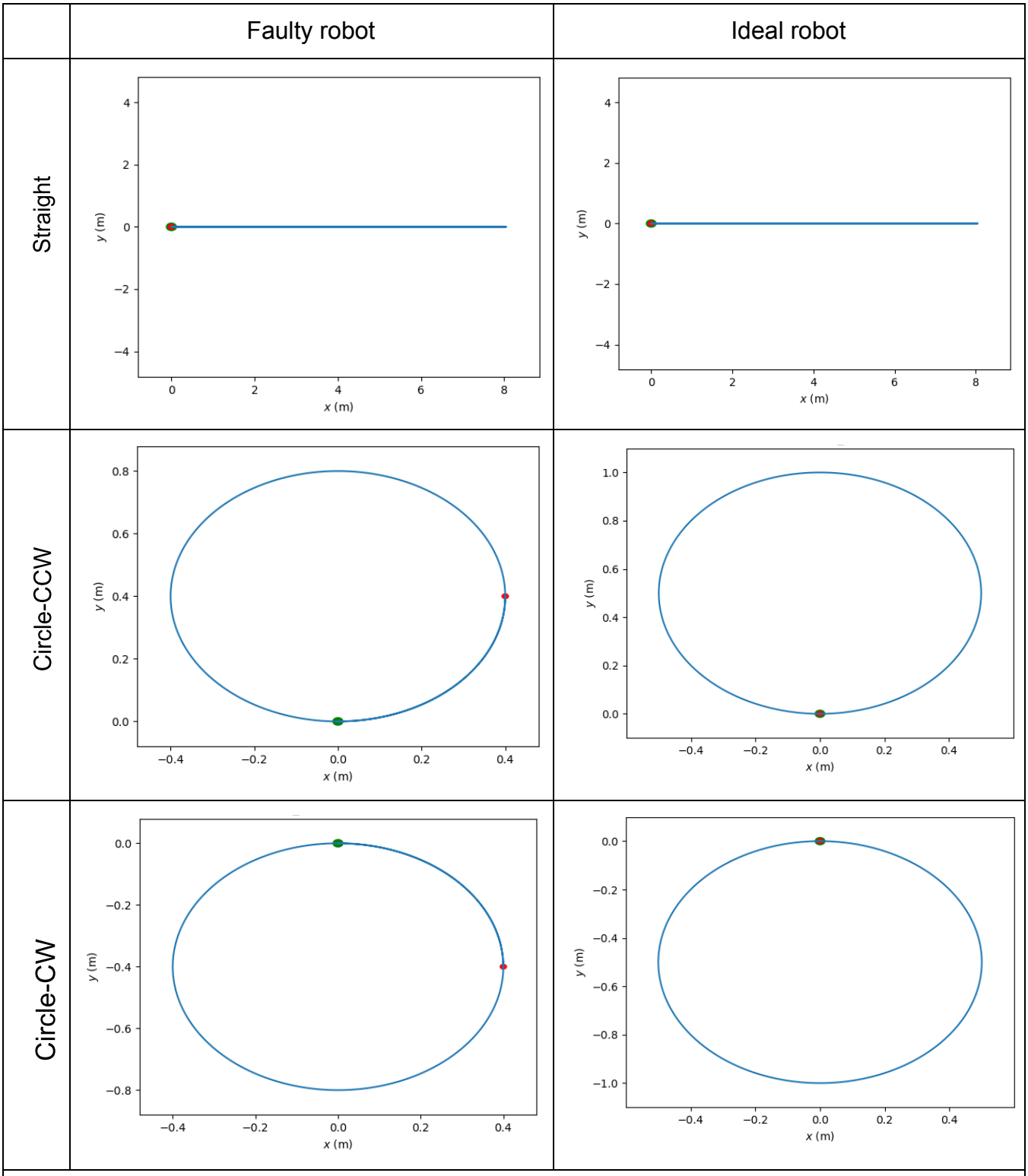


Fig. B3a: Uncorrected performance of SmallerBaseline robot when subject to some of the ideal robot's wheel-frame control policies.

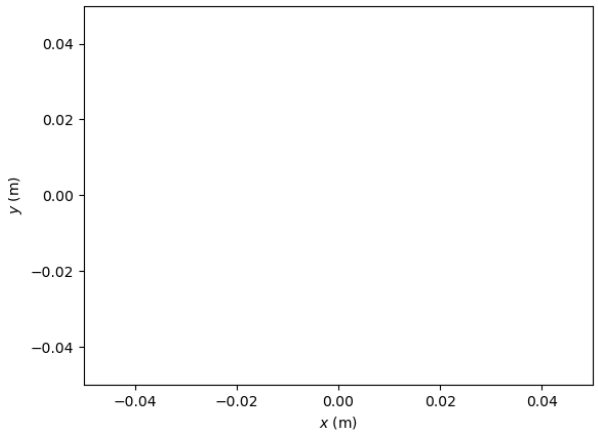
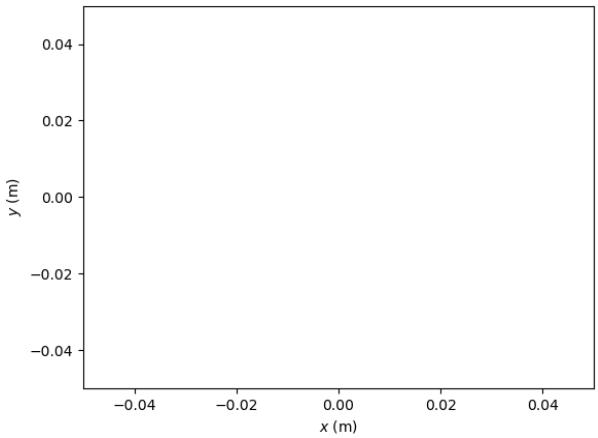
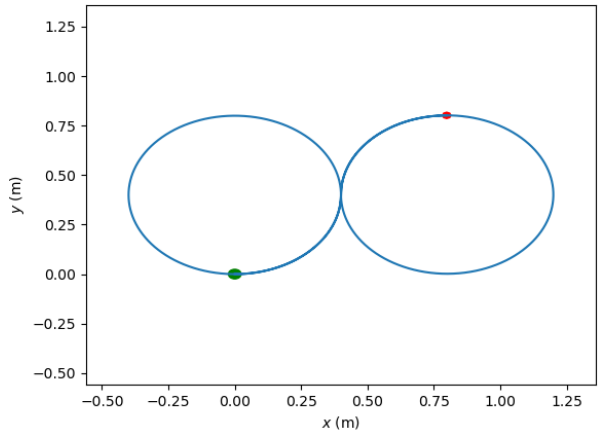
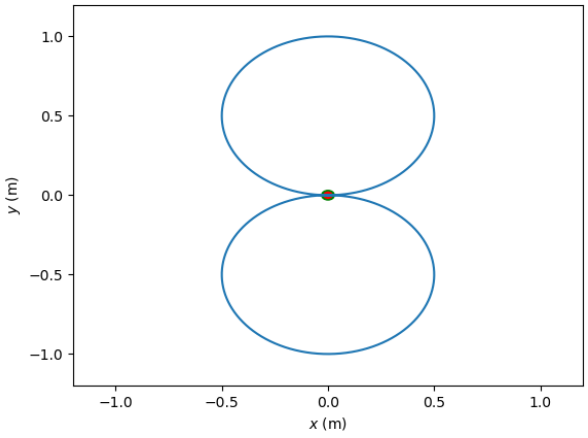
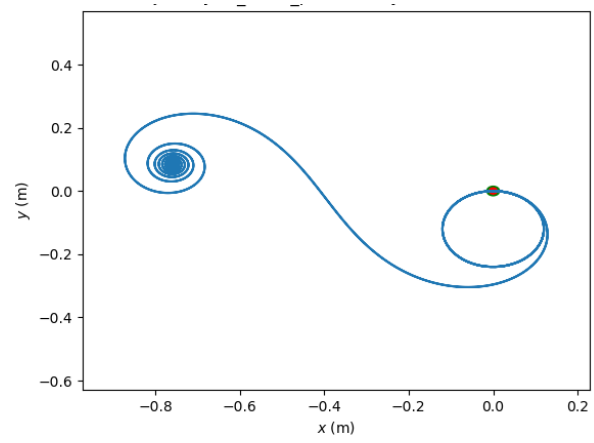
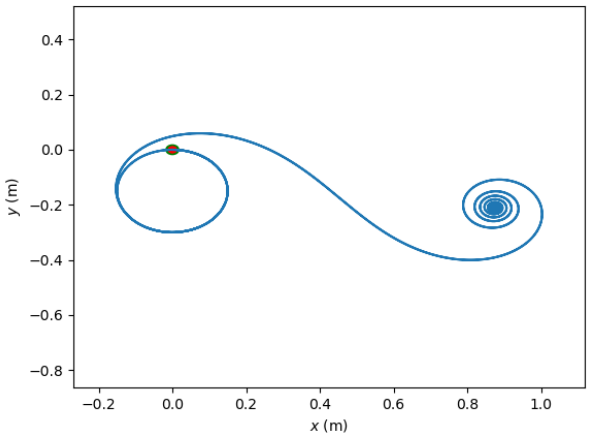
	Faulty robot	Ideal robot
Spin in-place	 (intentionally blank: no net motion)	 (intentionally blank: no net motion)
Figure-of-8		
“Tri-wave” φ		

Fig. B3b: Uncorrected performance of SmallerBaseline robot when subject to the remainder of the ideal robot’s wheel-frame control policies.

Appendix B4: Uncorrected performance of LargerBaseline robot

Fig. B4a and Fig. B4b below show how a robot with a larger baseline (12 cm instead of 10 cm; wheels are 2.0 cm radius) performs when subjected to the wheel-frame control policies from the ideal robot. We see that the deviations are significant.

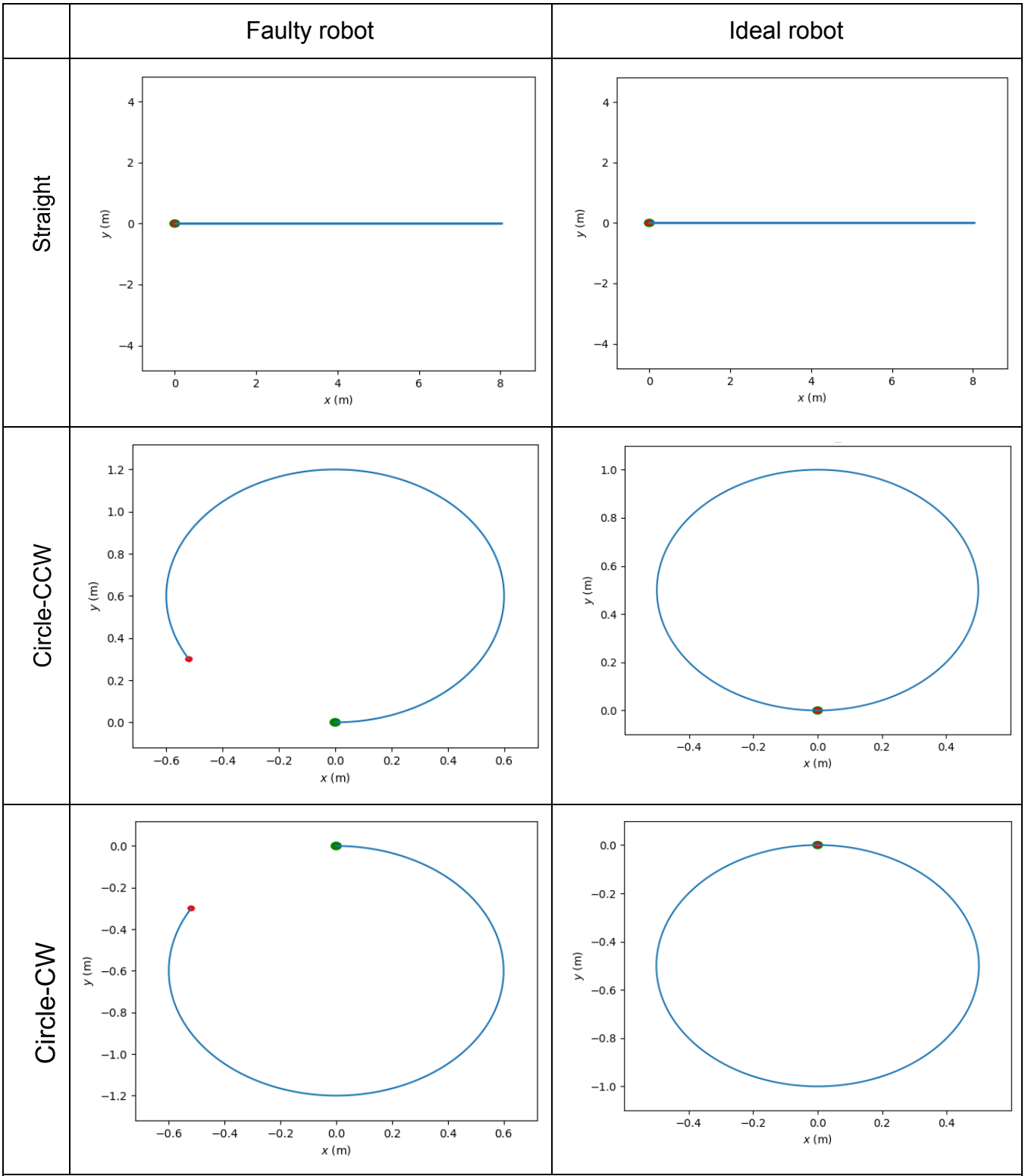


Fig. B4a: Uncorrected performance of LargerBaseline robot when subject to some of the ideal robot's wheel-frame control policies.

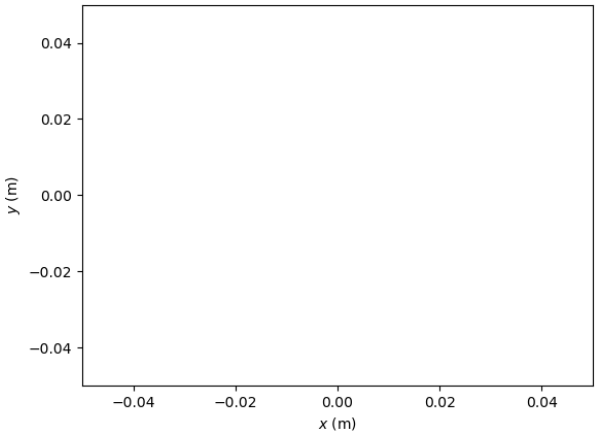
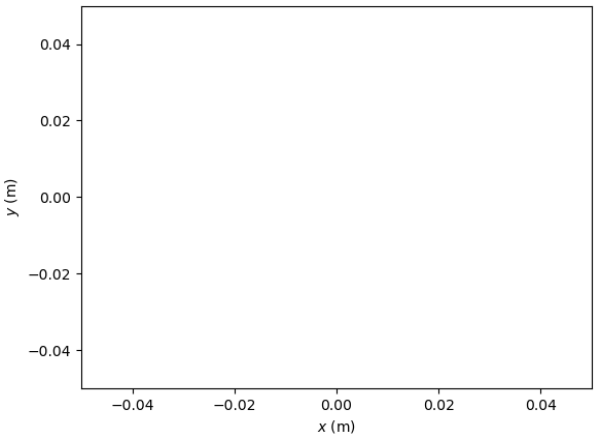
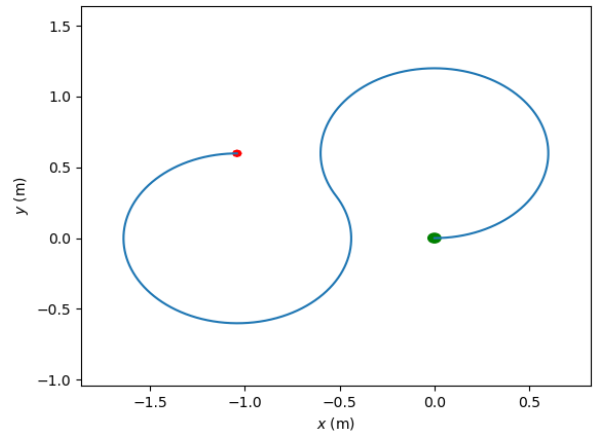
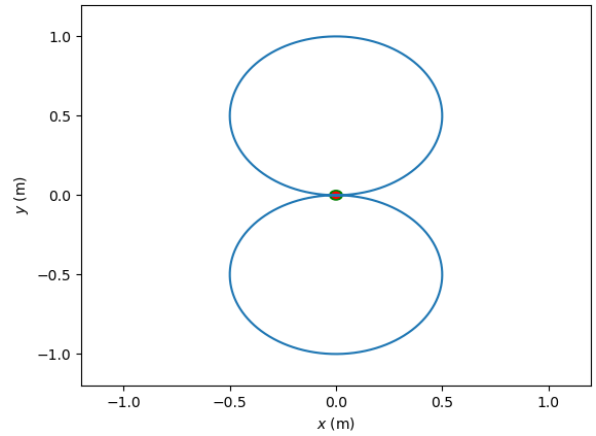
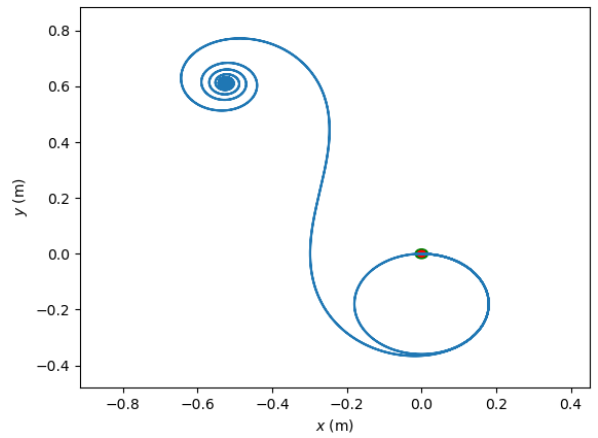
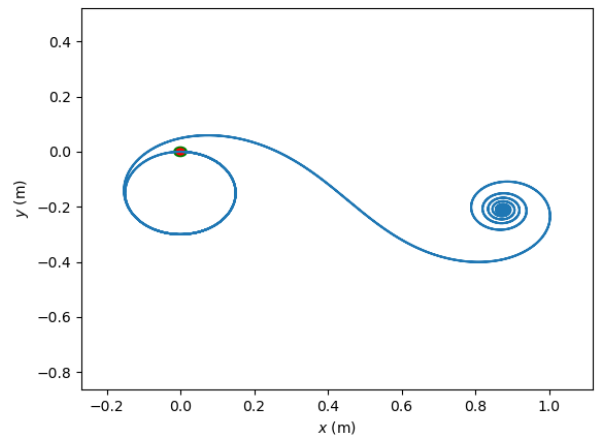
	Faulty robot	Ideal robot
Spin in-place	 (intentionally blank: no net motion)	 (intentionally blank: no net motion)
Figure-of-8		
“Tri-wave” φ		

Fig. B4b: Uncorrected performance of LargerBaseline robot when subject to the remainder of the ideal robot’s wheel-frame control policies.

Appendix B5: Uncorrected performance of “Noisy” robot

Fig. B5a and Fig. B5b below show how a robot with multiple defects performs when subjected to the wheel-frame control policies from the ideal robot. This robot has larger baseline (12 cm instead of 10 cm and one protrusion on left wheel (2.5 cm radius for 35° of 360°; otherwise 2.0 cm for both wheels).

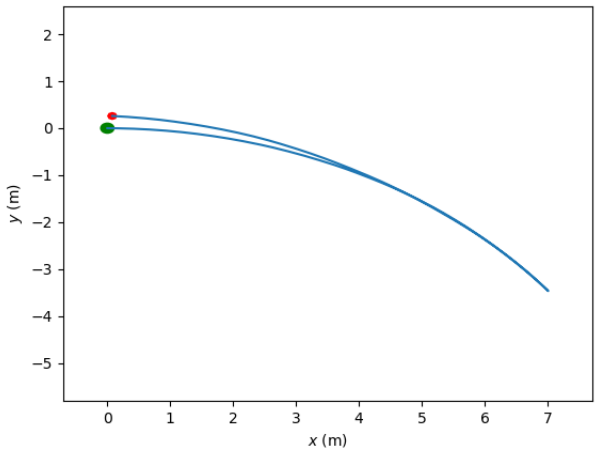
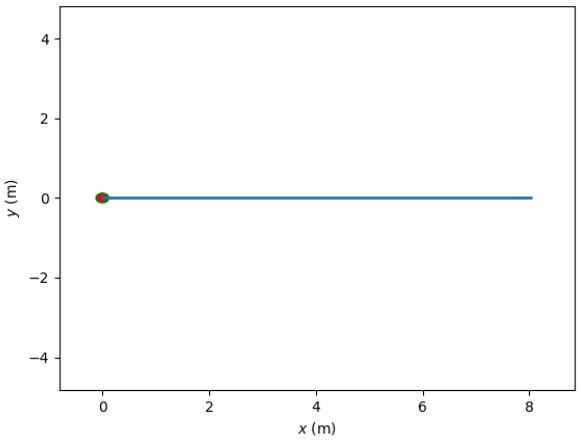
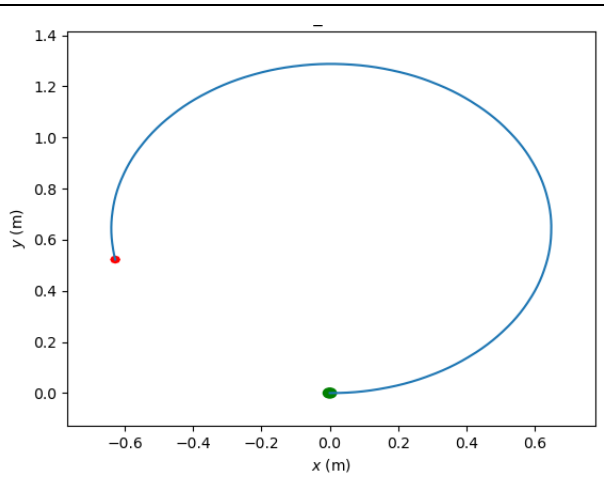
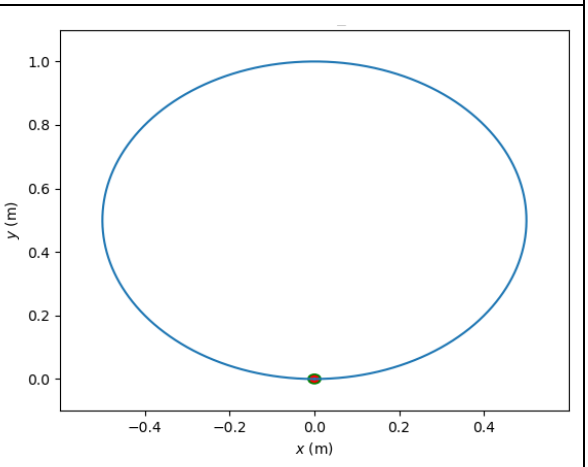
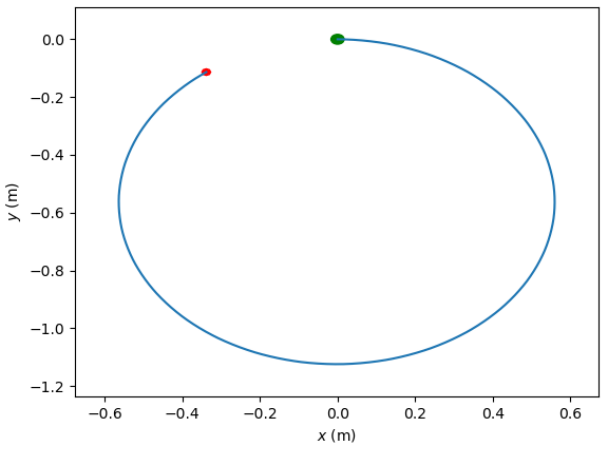
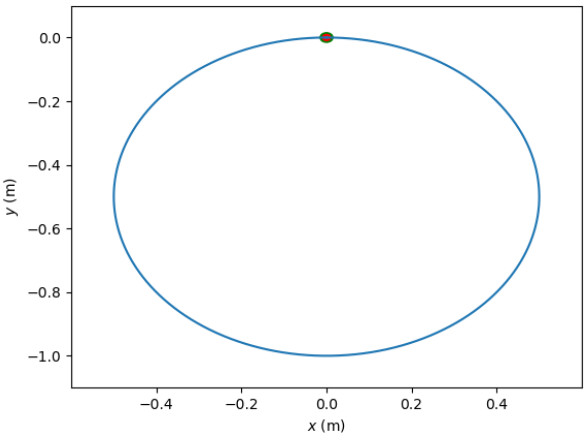
	Faulty robot	Ideal robot
Straight		
Circle-CCW		
Circle-CW		

Fig. B5a: Uncorrected performance of “Noisy” robot when subject to some of the ideal robot’s wheel-frame control policies.

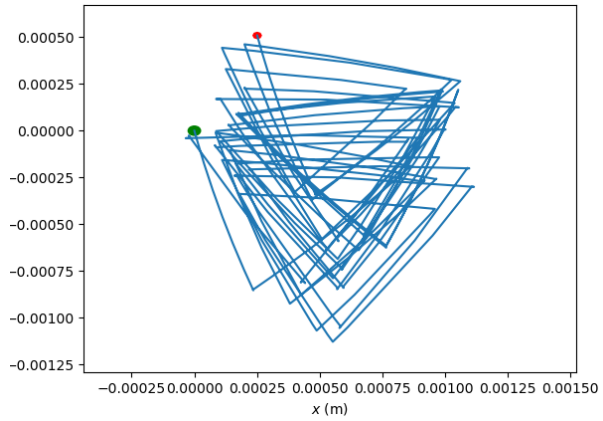
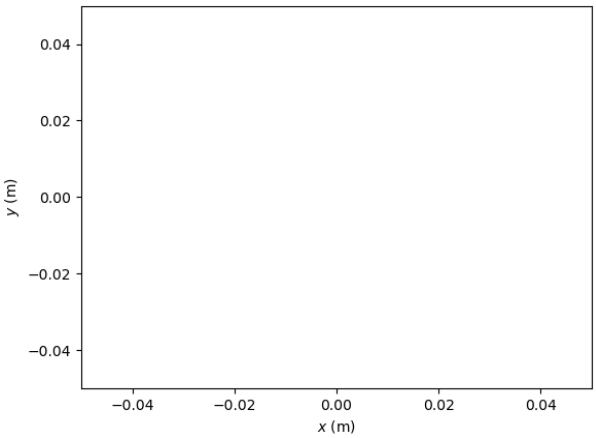
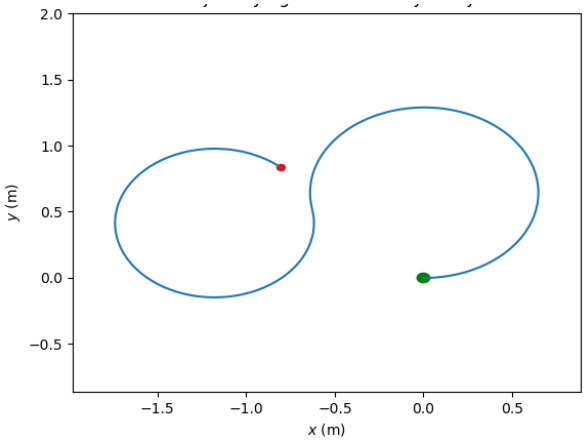
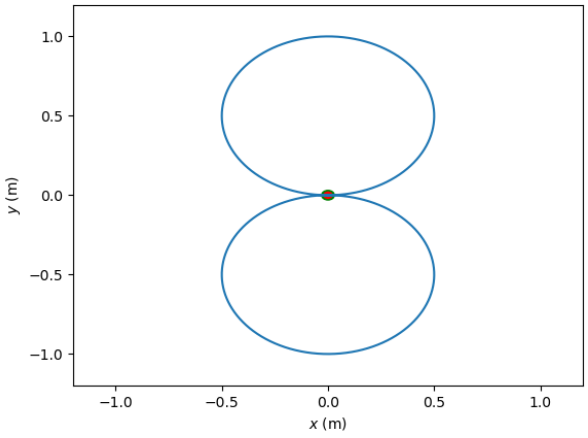
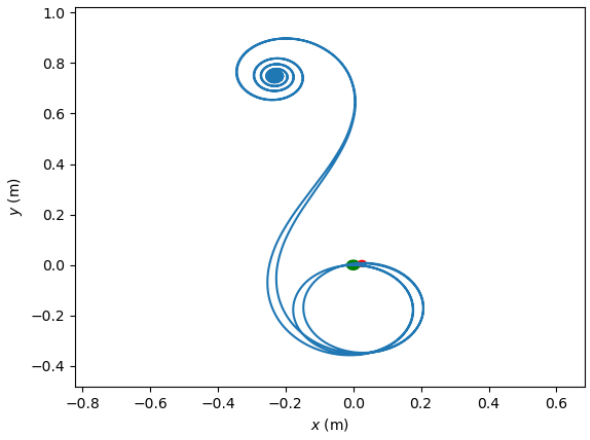
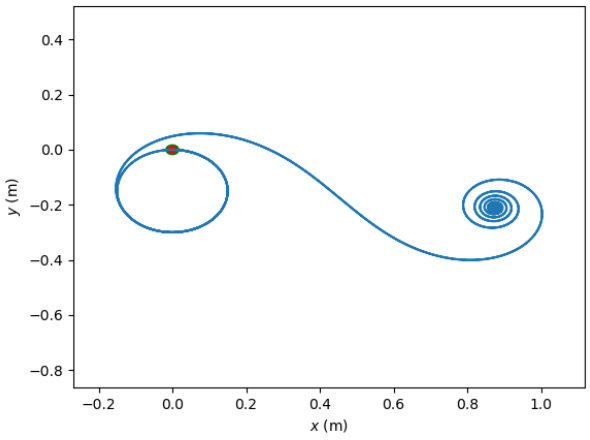
	Faulty robot	Ideal robot
Spin in-place		 (intentionally blank: no net motion)
Figure-of-8		
"Tri-wave" φ		

Fig. B5b: Uncorrected performance of "Noisy" robot when subject to the remainder of the ideal robot's wheel-frame control policies.

Appendix B6: Uncorrected performance of “Noisy” robot

Fig. B6a and Fig. B6b below show how a second robot with even more defects performs when subjected to the wheel-frame control policies from the ideal robot. This robot has smaller baseline (8 cm instead of 10 cm, one 0.5 cm protrusion (20° extent) on the left wheel and two flat spots on the right wheel - 0.4 cm (10° extent) and 0.2 cm (10° extent also). Otherwise the wheel-radius is 2 cm.

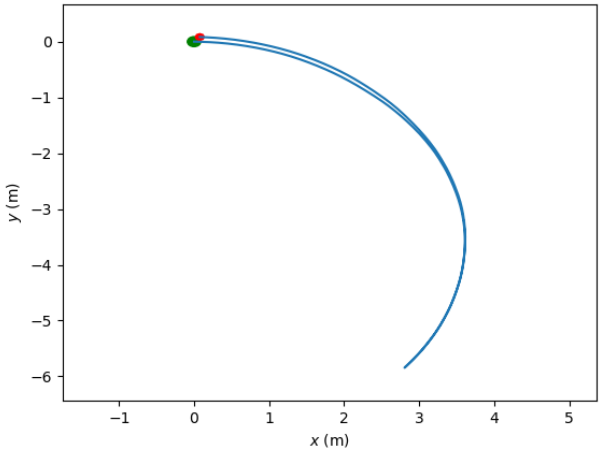
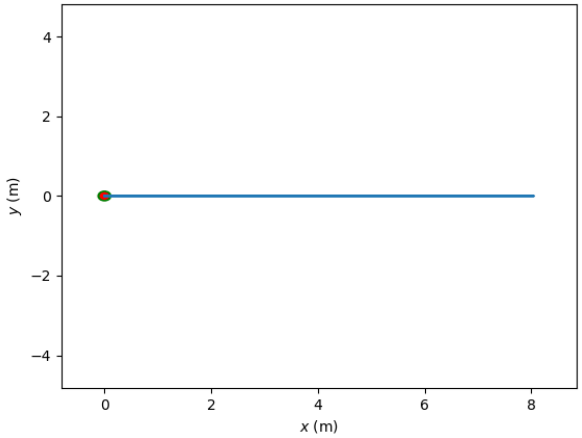
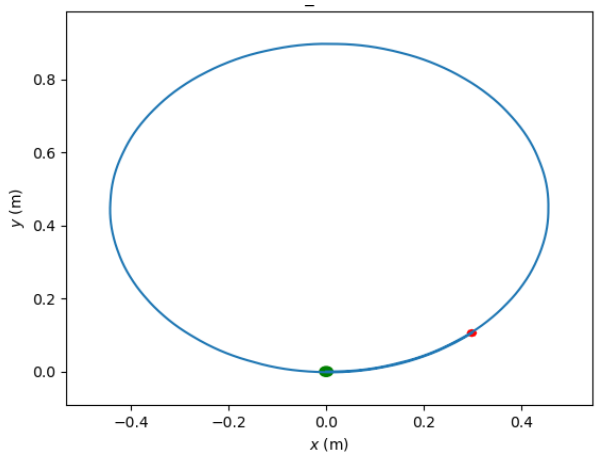
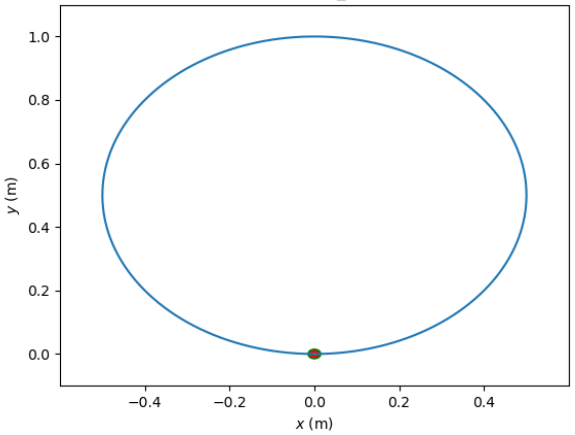
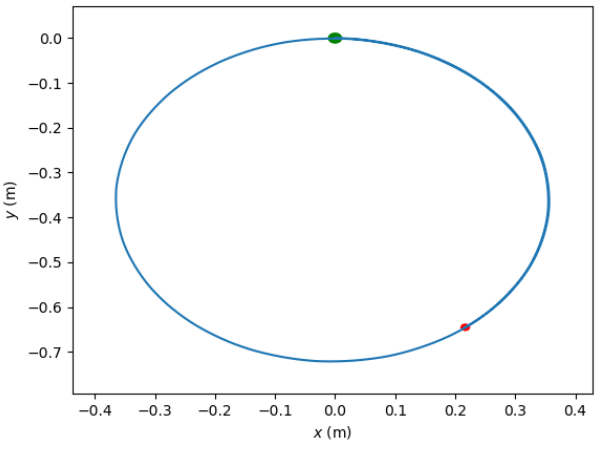
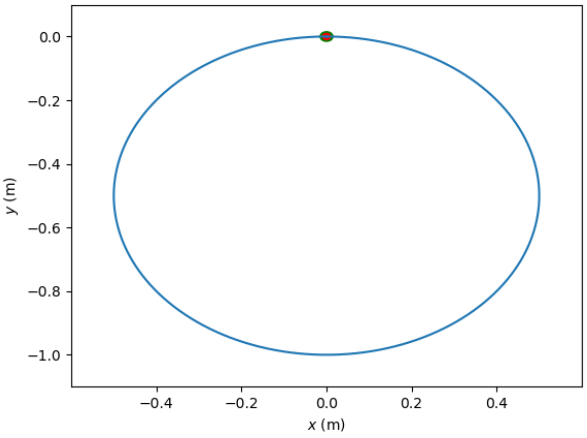
	Faulty robot	Ideal robot
Straight		
Circle-CCW		
Circle-CW		

Fig. B6a: Uncorrected performance of “Noisy” robot when subject to some of the ideal robot’s wheel-frame control policies.

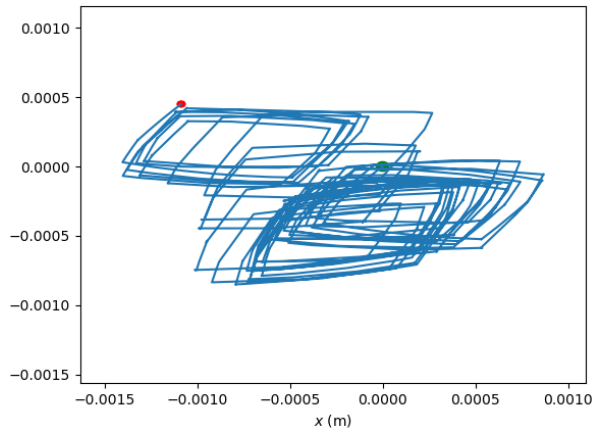
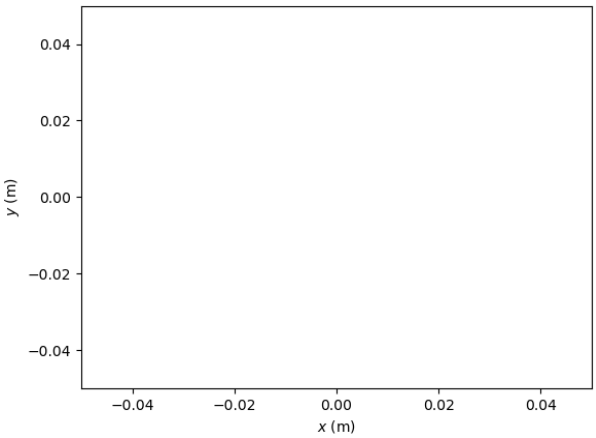
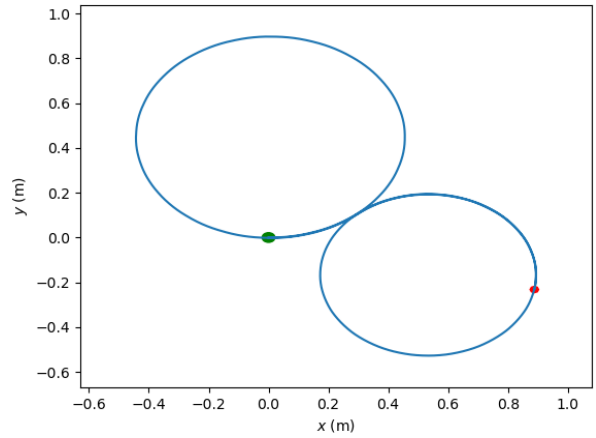
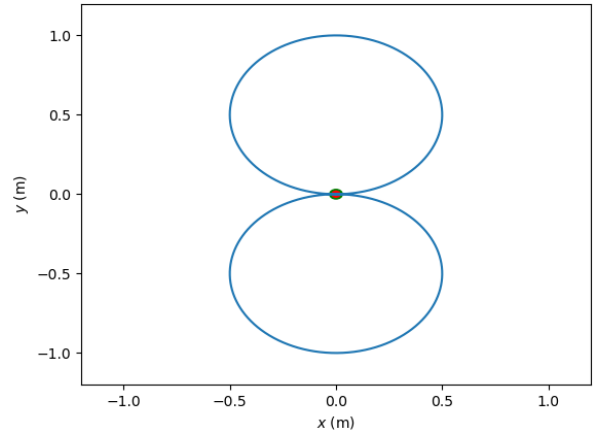
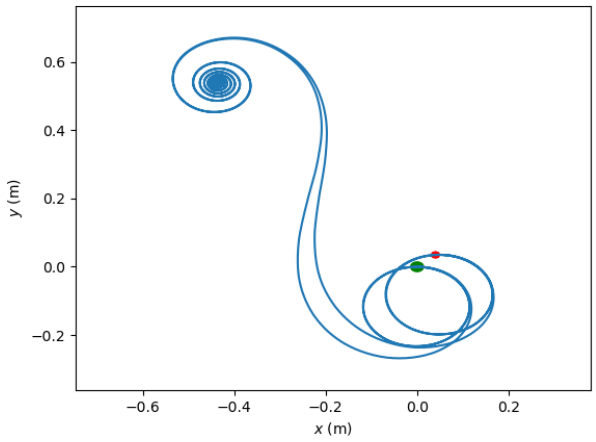
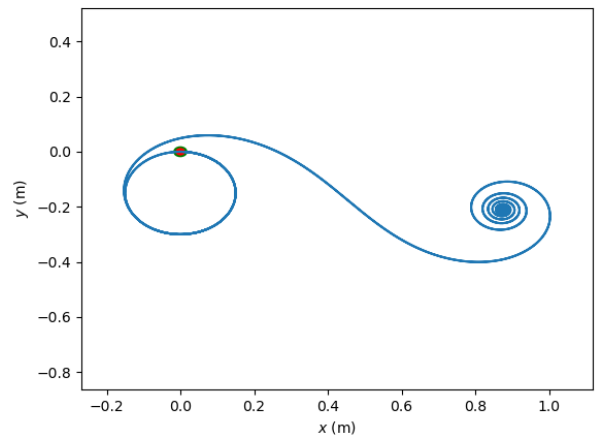
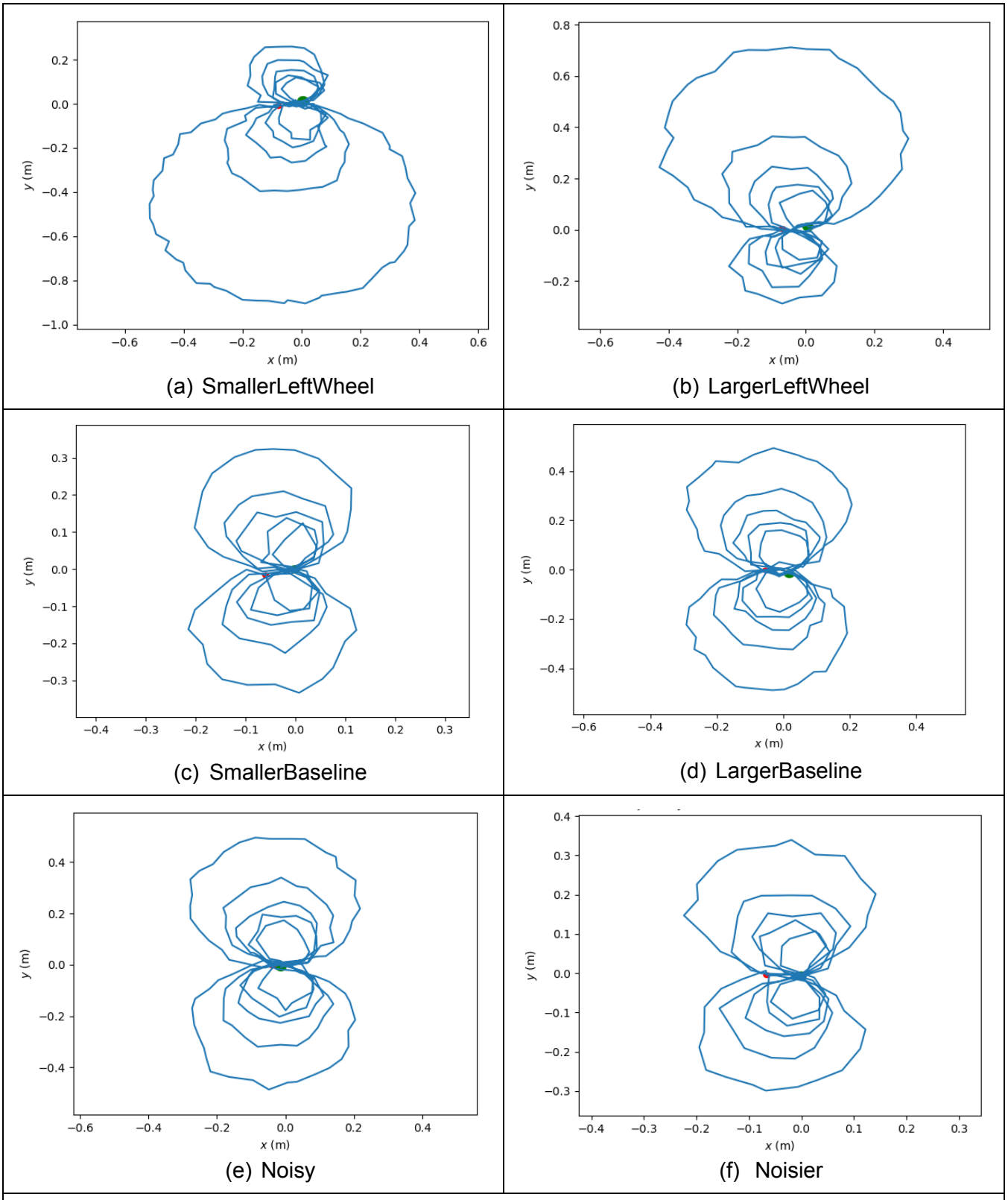
	Faulty robot	Ideal robot
Spin in-place		 (intentionally blank: no net motion)
Figure-of-8		
"Tri-wave" φ		

Fig. B6b: Uncorrected performance of "Noisy" robot when subject to the remainder of the ideal robot's wheel-frame control policies.

Appendix C: Dataset trajectories for the 6 robots

Fig. C1 shows the dataset trajectories used for each of the 6 robots. All were subject to the same combinations of wheel-frame control policy - $\dot{\varphi}_l$ and $\dot{\varphi}_r$. Each takes a different time and path to complete corresponding lobes of the figure-of-8.



(a) SmallerLeftWheel

(b) LargerLeftWheel

(c) SmallerBaseline

(d) LargerBaseline

(e) Noisy

(f) Noisier

Fig. C1: Dataset trajectories for the 6 robots.

Appendix D1: SysID-based control of SmallerLeftWheel robot

Fig. D1a and D1b shows the paths generated by the robots with the smaller wheel when using the System-Identification to create a kinematic controller to translate the desired body-frame control policy (v,ω) into actual wheel-frame control policy ($\dot{\varphi}_l$, $\dot{\varphi}_r$) specific to the robot with the smaller left wheel. Left column overlays the traversed path (black) on top of that traversed by the ideal robot model (red). The right column shows point-wise separation between the two paths along the trajectory.

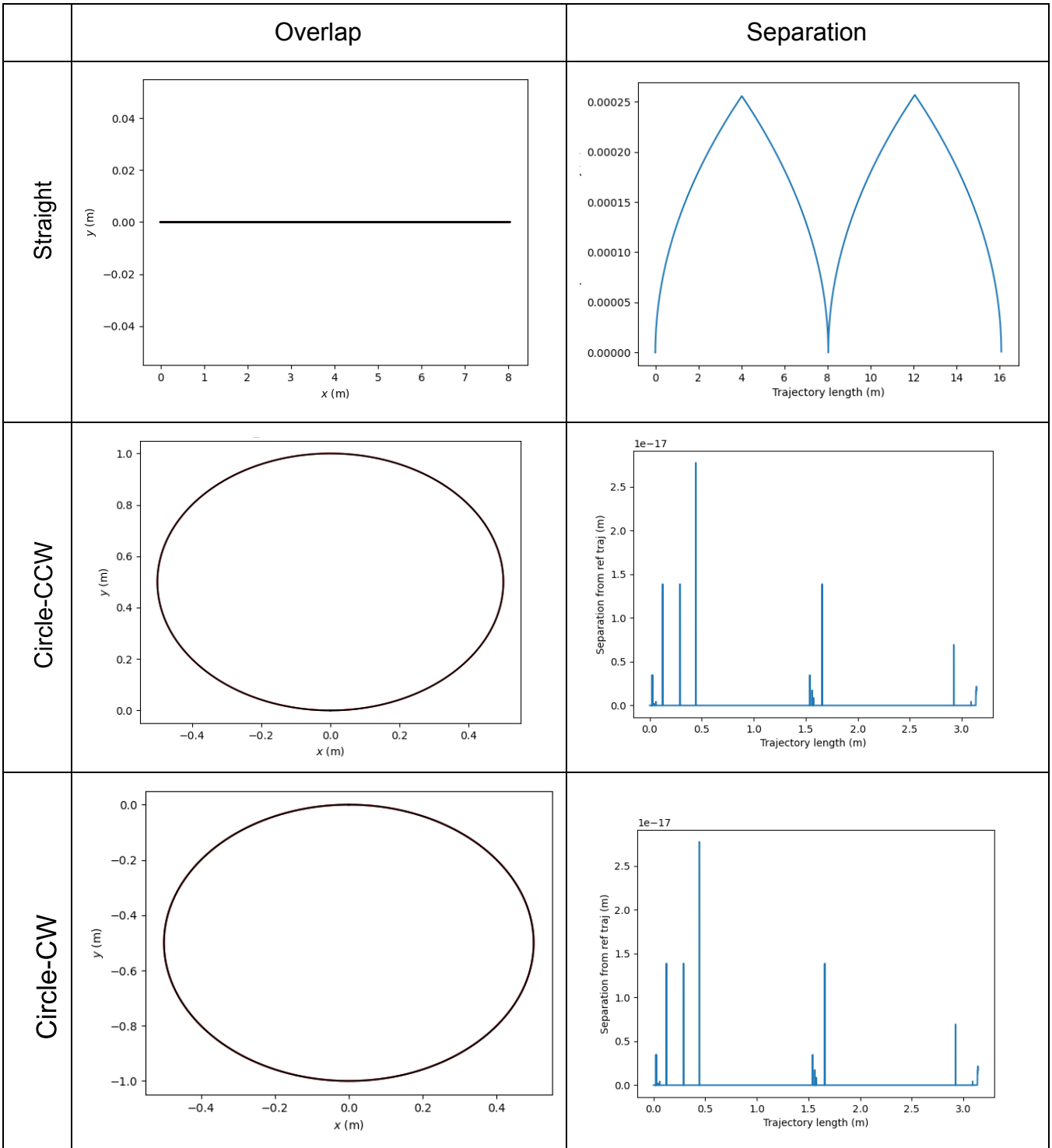


Fig. D1a: Corrected performance of “SmallerLeftWheel” robot when subject to some of the ideal robot’s

wheel-frame control policies. The controller is based on system-identification of the baseline and wheel-radii.

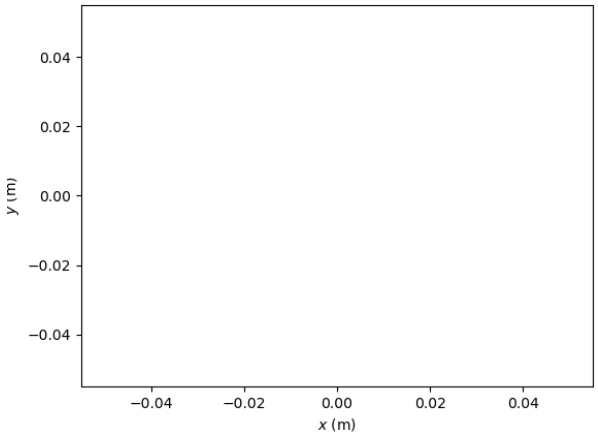
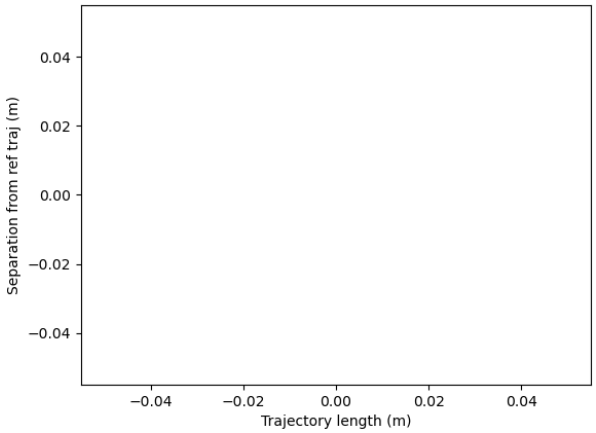
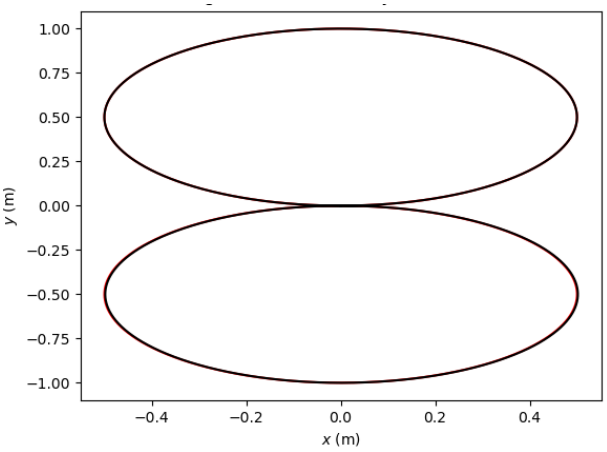
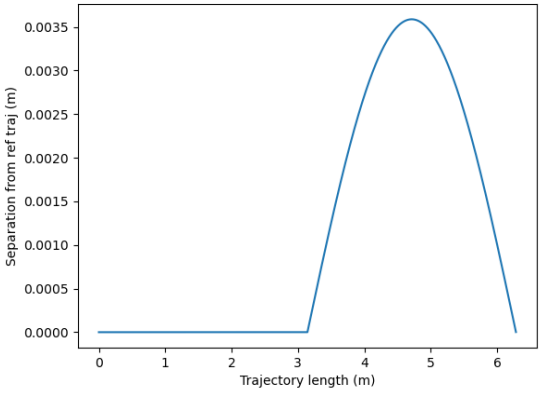
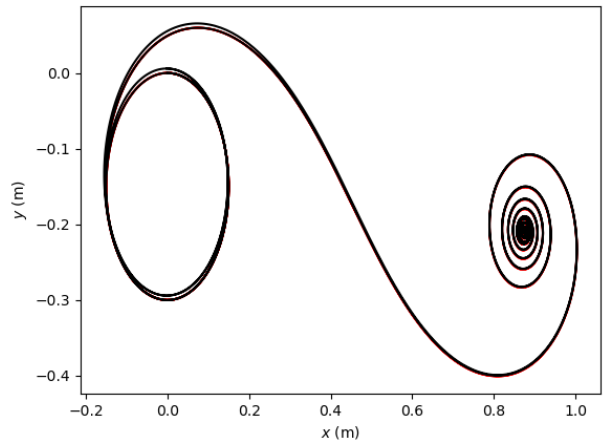
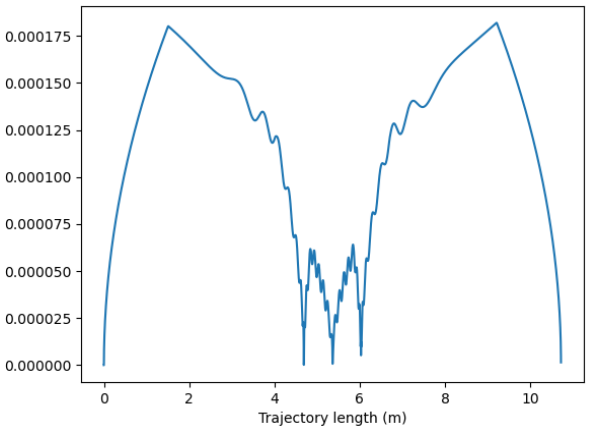
	Overlap	Separation
Spin	 *(intentionally blank - no net motion)*	 *(intentionally blank - no net motion)*
Figure-of-8		
"Tri-wave" φ		

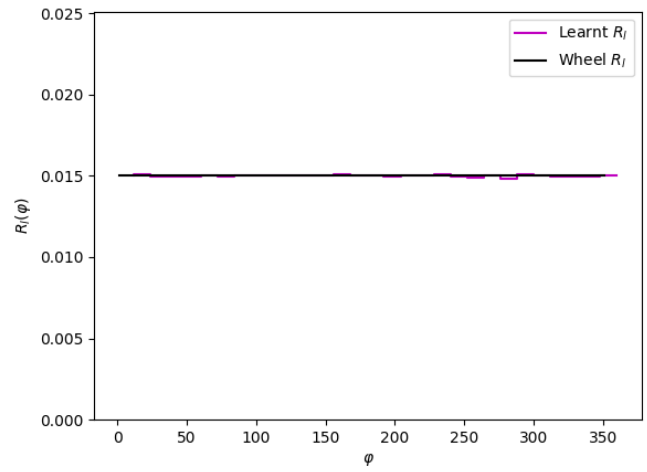
Fig. D1b: Fig. D1a: Corrected performance of "SmallerLeftWheel" robot when subject to the remainder of the ideal robot's wheel-frame control policies. The controller is based on system-identification of the baseline and wheel-radii variations.

The table below shows the maximum distance (meters) of executed trajectory from expectation, for

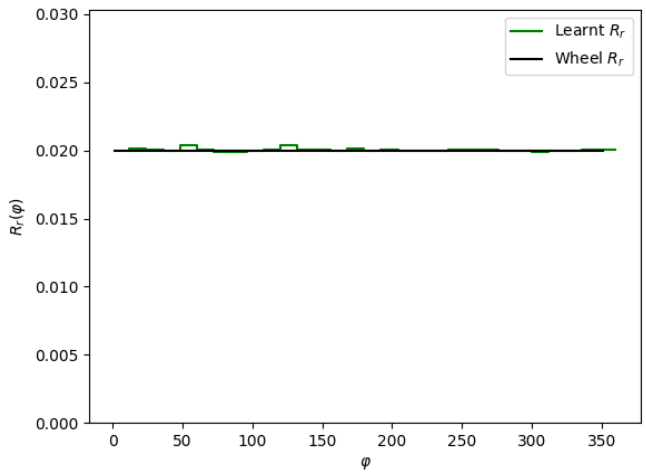
Straight	Circle-CCW	Circle-CW	Spin in-place	Figure-of-8	Tri-wave φ
0.0003	0	0	0	0.0036	0.0002

The learnt baseline was 10 cm which matches the actual baseline.

Fig. D1c shows the learnt radii as a function of wheel-angle, for the left and right wheels.



(a) Left-wheel radii



(b) Right-wheel radii

Fig. D1c: Learnt radii for left and right wheels for the “SmallerLeftWheel” robot.

Appendix D2: SysID-based control of LargerLeftWheel robot

Fig. D2a and D2b shows the paths generated by the robot with the larger wheel when using the System-Identification to create a kinematic controller to translate the desired body-frame control policy (v,ω) into actual wheel-frame control policy ($\dot{\varphi}_l$, $\dot{\varphi}_r$) specific to the robot with the smaller left wheel. Left column overlays the traversed path (black) on top of that traversed by the ideal robot model (red). The right column shows point-wise separation between the two paths along the trajectory.

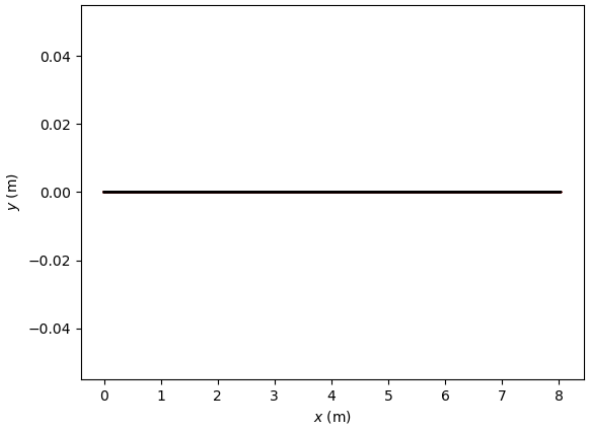
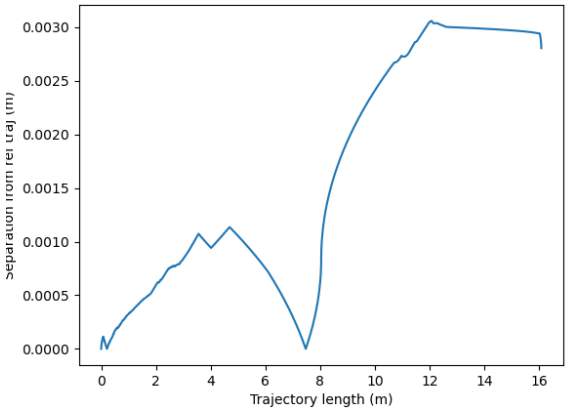
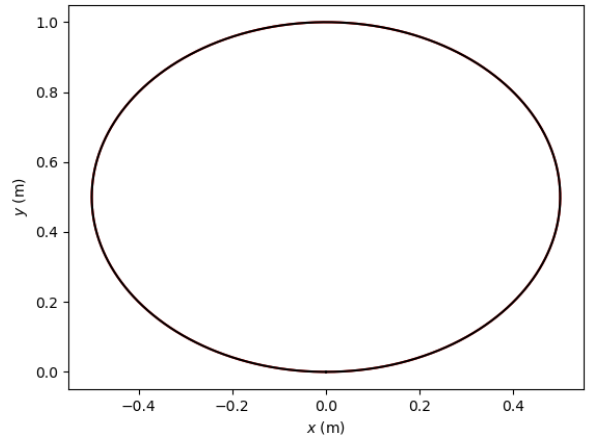
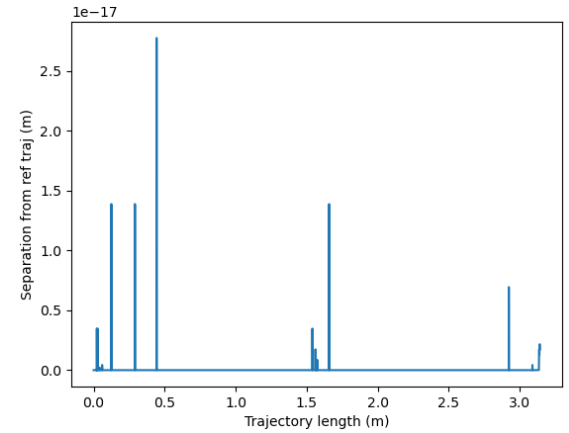
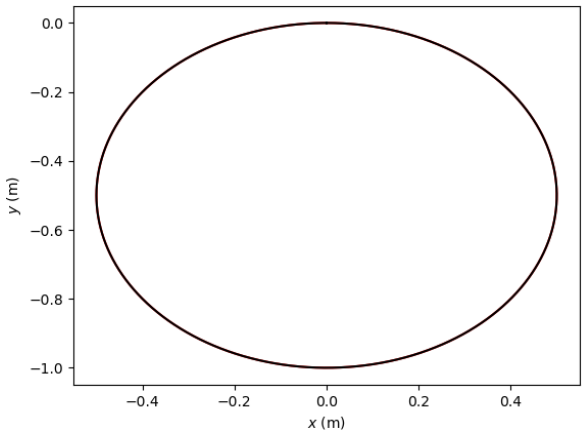
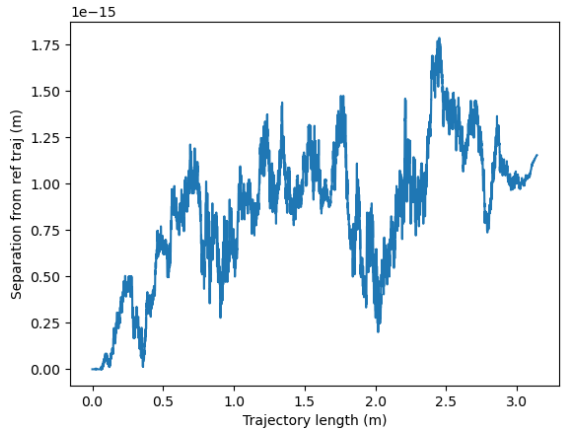
	Overlap	Separation
Straight		
Circle-CCW		
Circle-CW		

Fig. D2a: Corrected performance of “LargerLeftWheel” robot when subject to some of the ideal robot’s wheel-frame control policies. The controller is based on system-identification of the baseline and wheel-radii.

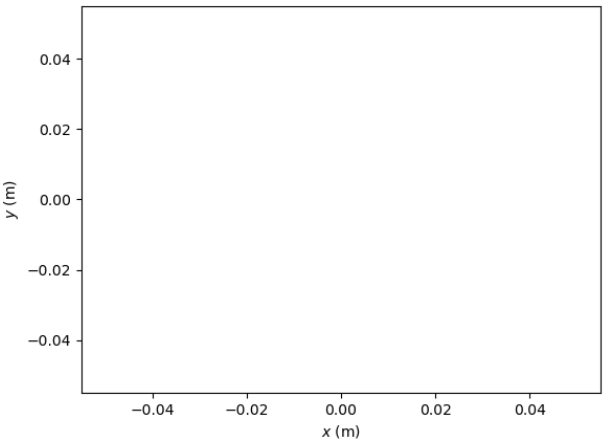
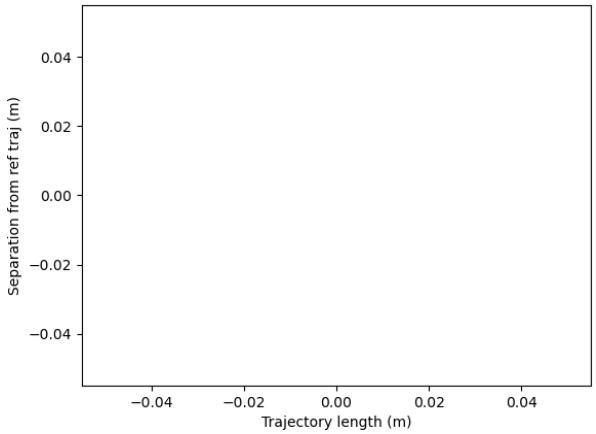
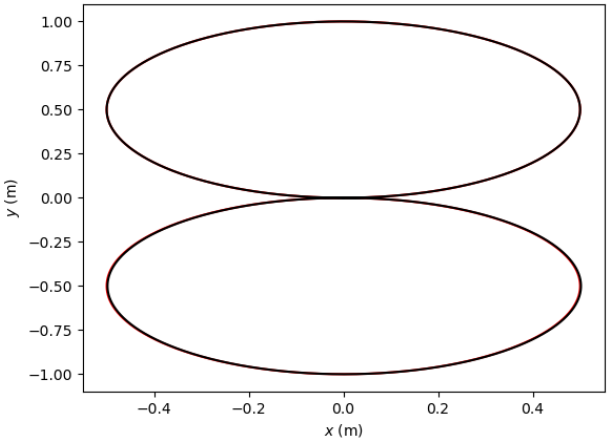
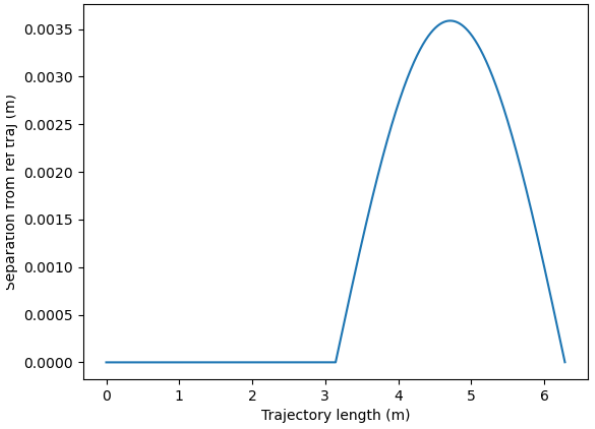
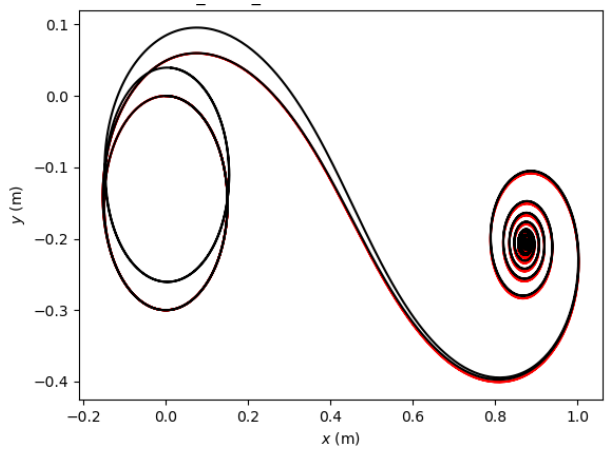
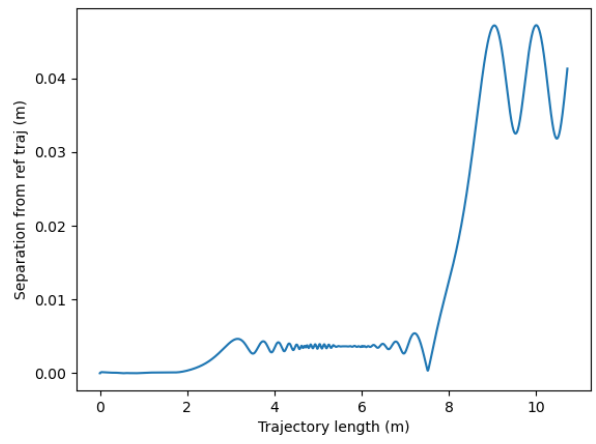
	Overlap	Separation
Spin	 *(intentionally blank - no net motion)*	 *(intentionally blank - no net motion)*
Figure-of-8		
"Tri-wave" φ		

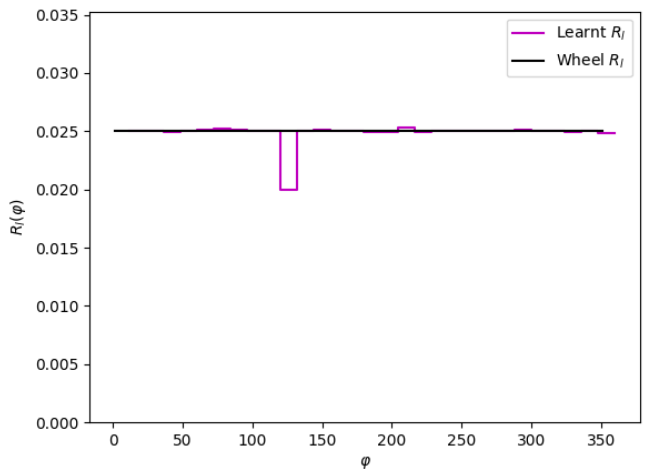
Fig. D2b: Corrected performance of "LargerLeftWheel" robot when subject to the remainder of the ideal robot's wheel-frame control policies. The controller is based on system-identification of the baseline and wheel-radii variations.

The table below shows the maximum distance (meters) of executed trajectory from expectation, for

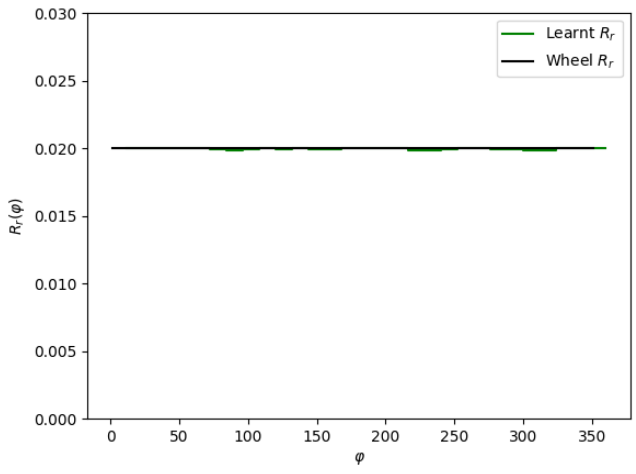
Straight	Circle-CCW	Circle-CW	Spin in-place	Figure-of-8	Tri-wave φ
0.0001	0	0	0	0.0036	0.0002

The learnt baseline was 10 cm which matches the actual baseline.

Fig. D2c shows the learnt radii as a function of wheel-angle, for the left and right wheels.



Left-wheel radii



Right-wheel radii

Fig. D2c: Learnt radii for left and right wheels for the “LargerLeftWheel” robot.

Appendix D3: SysID-based control of SmallerBaseline robot

Fig. D3a and D3b shows the paths generated by the robot with the smaller baseline when using the System-Identification to create a kinematic controller to translate the desired body-frame control policy (v,ω) into actual wheel-frame control policy ($\dot{\varphi}_l$, $\dot{\varphi}_r$) specific to the robot with the smaller left wheel. Left column overlays the traversed path (black) on top of that traversed by the ideal robot model (red). The right column shows point-wise separation between the two paths along the trajectory.

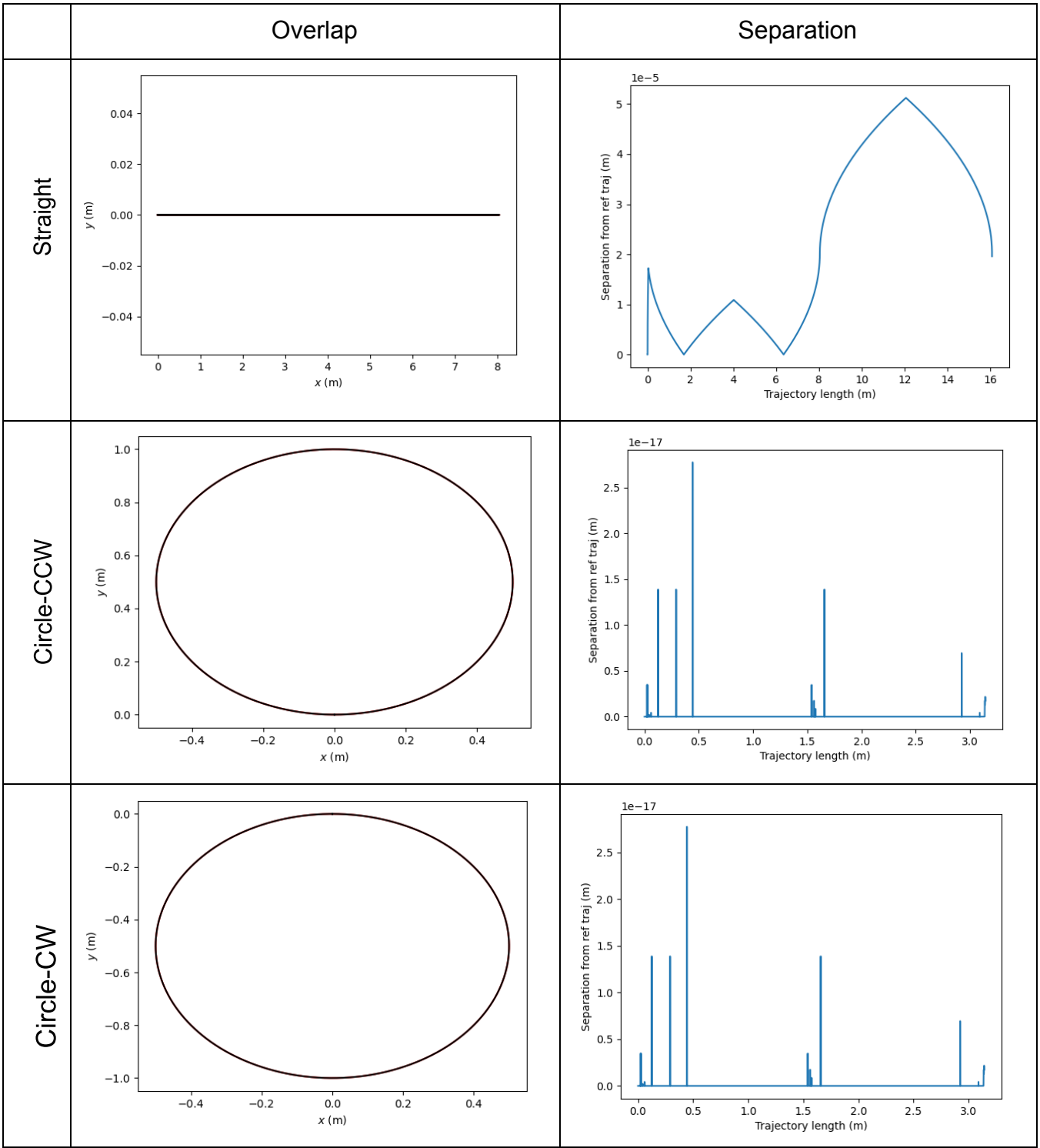


Fig. D3a: Corrected performance of "SmallerBaseline" robot when subject to some of the ideal robot's wheel-frame control policies. The controller is based on system-identification of the baseline and wheel-radii.

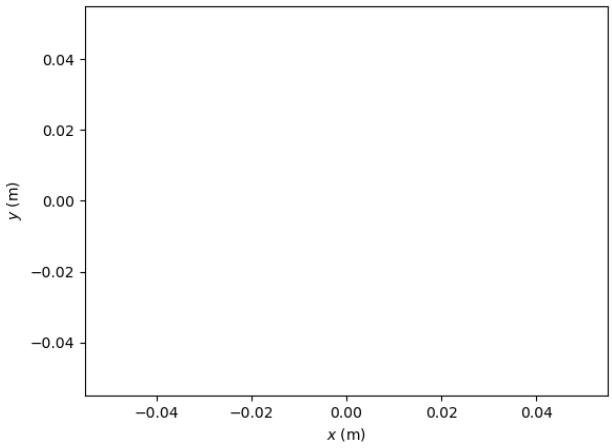
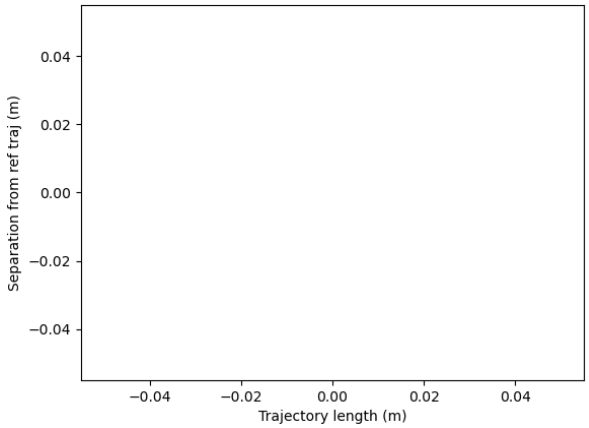
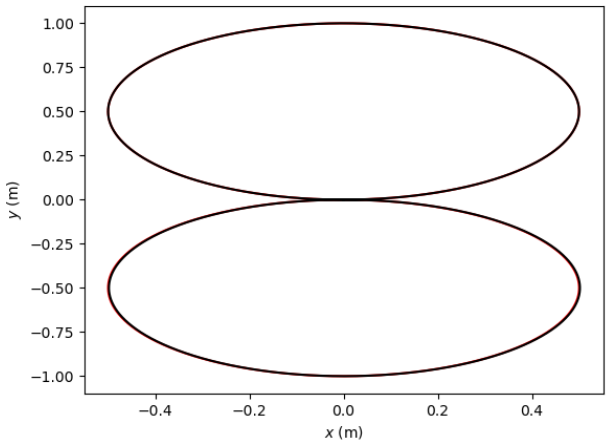
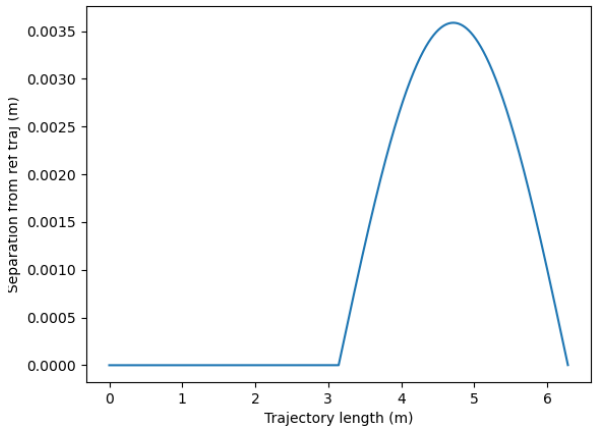
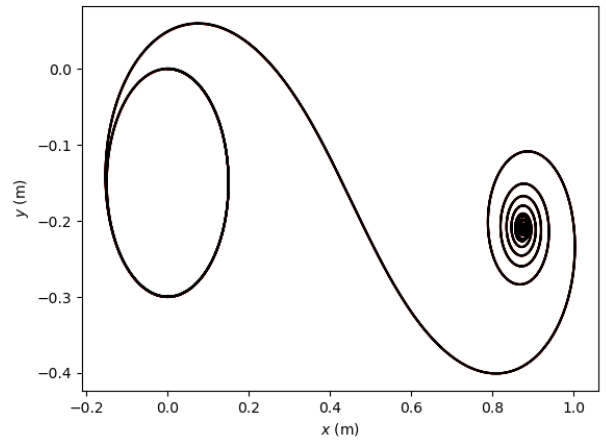
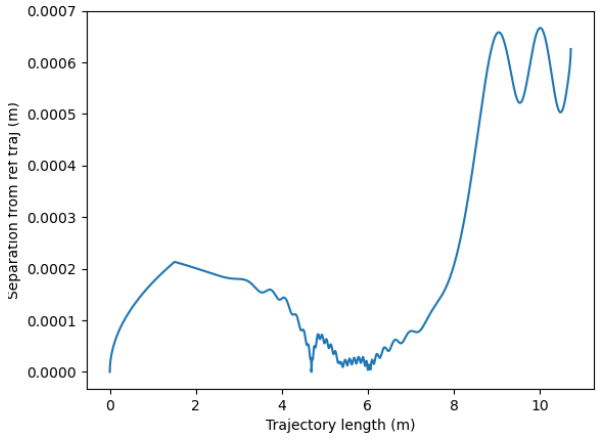
	Overlap	Separation
Spin	 *(intentionally blank - no net motion)*	 *(intentionally blank - no net motion)*
Figure-of-8		
"Tri-wave" φ		

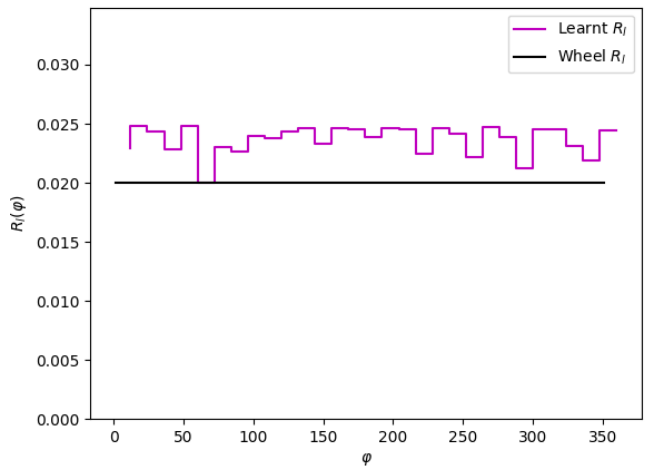
Fig. D3b: Corrected performance of "SmallerBaseline" robot when subject to the remainder of the ideal robot's wheel-frame control policies. The controller is based on system-identification of the baseline and wheel-radii variations.

The table below shows the maximum distance (meters) of executed trajectory from expectation, for

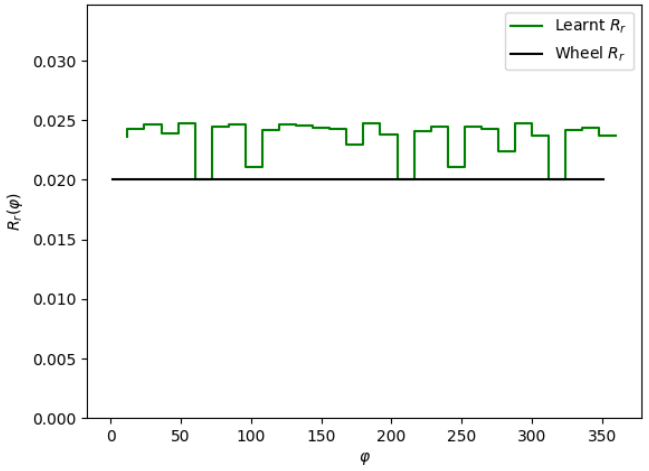
Straight	Circle-CCW	Circle-CW	Spin in-place	Figure-of-8	Tri-wave φ
0.0002	0	0	0	0.0036	0.0002

The learnt baseline was 10 cm which exceeds the actual baseline of 8 cm.

Fig. D3c shows the learnt radii as a function of wheel-angle, for the left and right wheels.



Left-wheel radii



Right-wheel radii

Fig. D3c: Learnt radii for left and right wheels for the “SmallerBaseline” robot.

The procedure transfers the learning entirely on to the wheel radii and ignores the baseline.

Appendix D4: SysID-based control of LargerBaseline robot

Fig. D4a and D4b shows the paths generated by the robot with the larger baseline when using the System-Identification to create a kinematic controller to translate the desired body-frame control policy (v,ω) into actual wheel-frame control policy ($\dot{\varphi}_l$, $\dot{\varphi}_r$) specific to the robot with the smaller left wheel. Left column overlays the traversed path (black) on top of that traversed by the ideal robot model (red). The right column shows point-wise separation between the two paths along the trajectory.

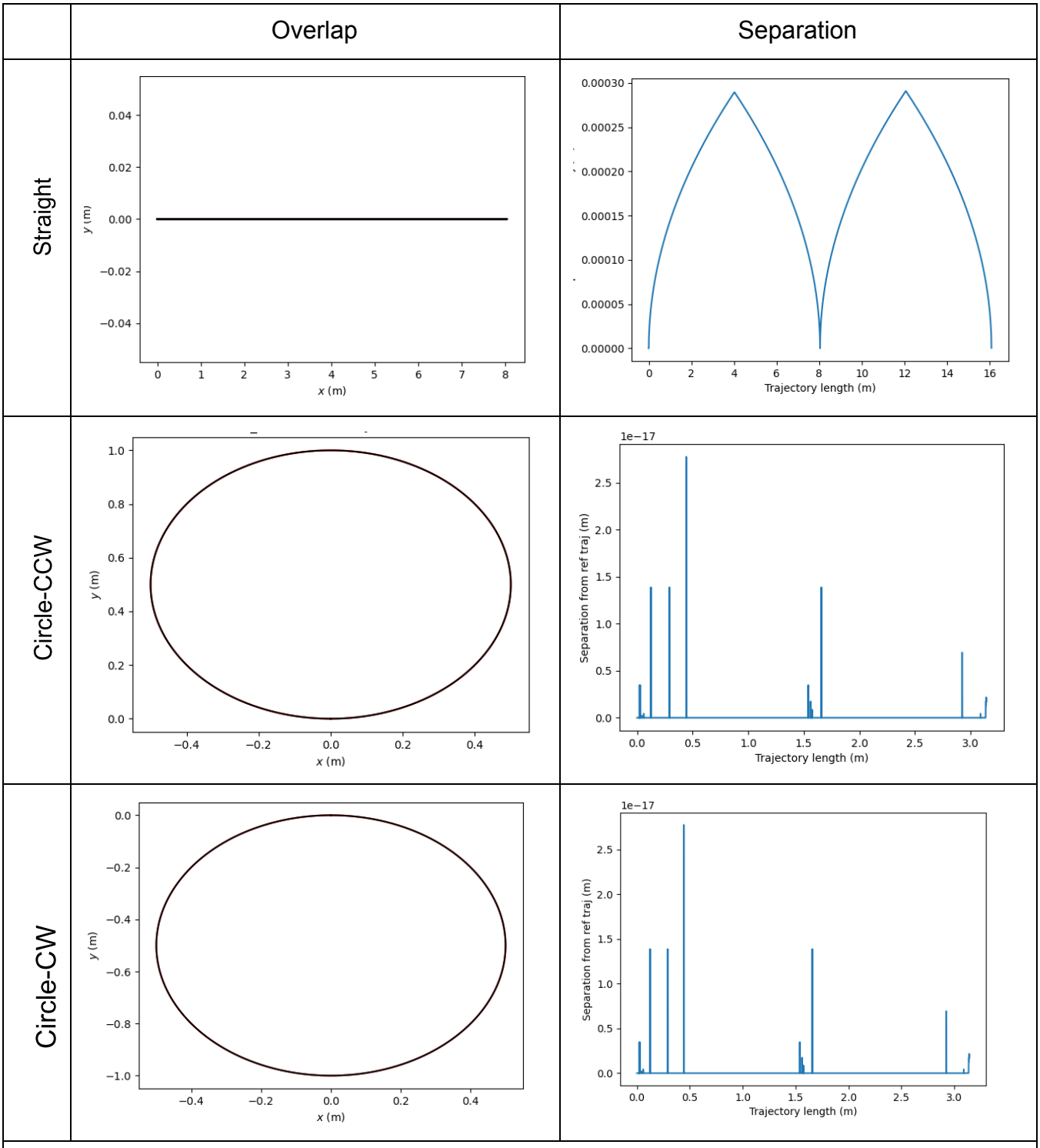


Fig. D4a: Corrected performance of "LargerrBaseline" robot when subject to some of the ideal robot's wheel-frame control policies. The controller is based on system-identification of the baseline and wheel-radii.

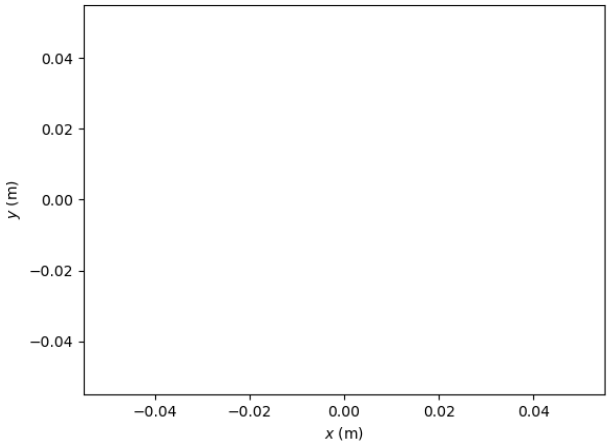
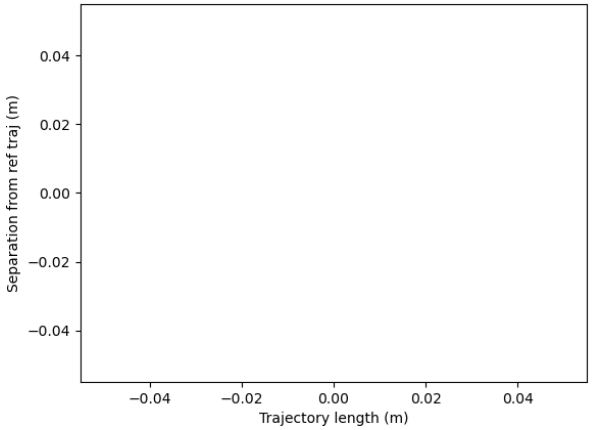
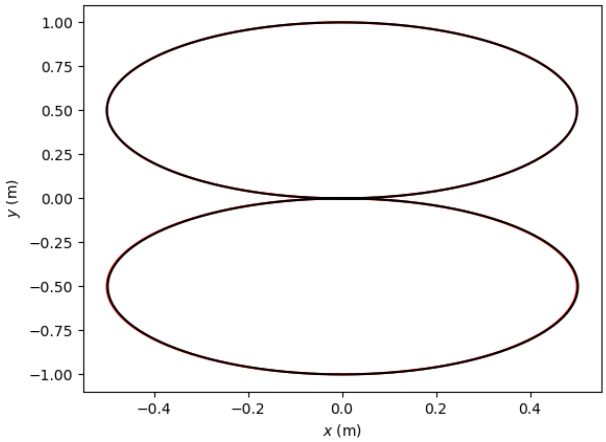
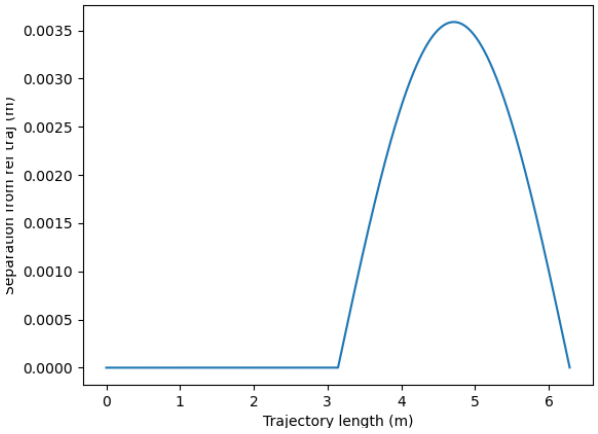
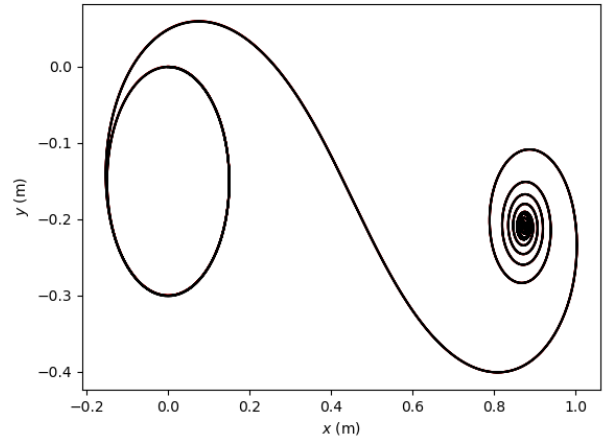
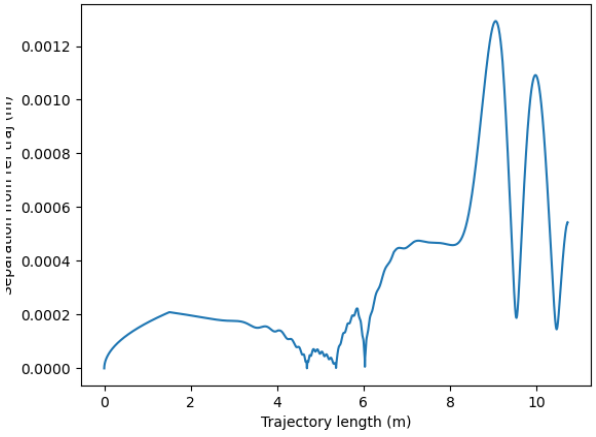
	Overlap	Separation
Spin	 *(intentionally blank - no net motion)*	 *(intentionally blank - no net motion)*
Figure-of-8		
"Tri-wave" φ		

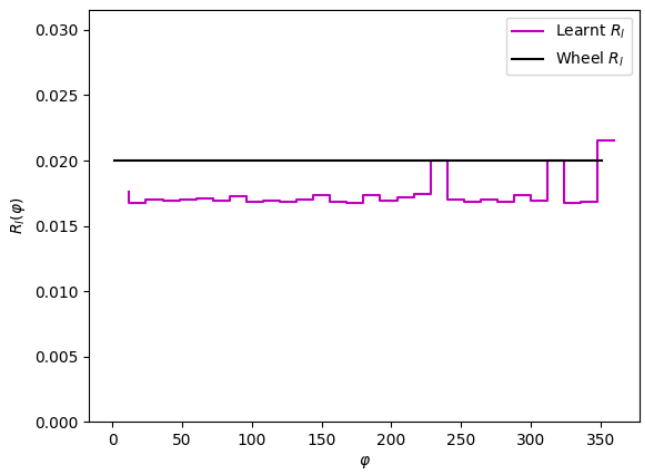
Fig. D4b: Corrected performance of "LargerBaseline" robot when subject to the remainder of the ideal robot's wheel-frame control policies. The controller is based on system-identification of the baseline and wheel-radii variations.

The table below shows the maximum distance (meters) of executed trajectory from expectation, for

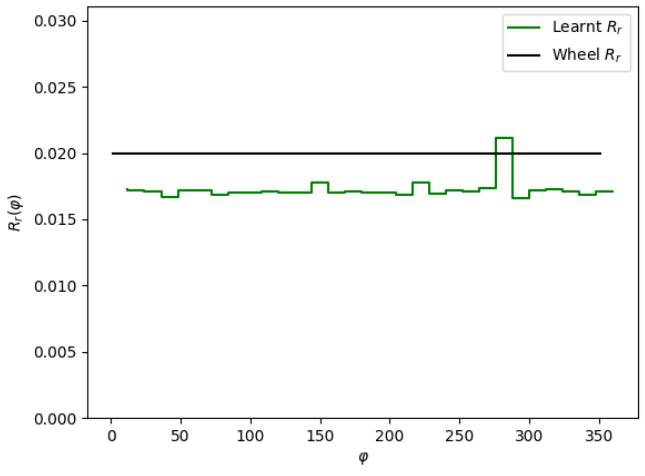
Straight	Circle-CCW	Circle-CW	Spin in-place	Figure-of-8	Tri-wave φ
0.0002	0	0	0	0.0036	0.0018

The learnt baseline was 10 cm which does not meet the actual baseline of 12 cm.

Fig. D4c shows the learnt radii as a function of wheel-angle, for the left and right wheels.



Left-wheel radii



Right-wheel radii

Fig. D4c: Learnt radii for left and right wheels for the “LargerBaseline” robot.

The procedure transfers the learning entirely on to the wheel radii and ignores the baseline.

Appendix D5: SysID-based control of “Noisy” robot

Fig. D5a and D5b shows the paths generated by the robot with the noisy construction when using the System-Identification to create a kinematic controller to translate the desired body-frame control policy (v,ω) into actual wheel-frame control policy ($\dot{\varphi}_l$, $\dot{\varphi}_r$) specific to the robot with the smaller left wheel. Left column overlays the traversed path (black) on top of that traversed by the ideal robot model (red). The right column shows point-wise separation between the two paths along the trajectory.

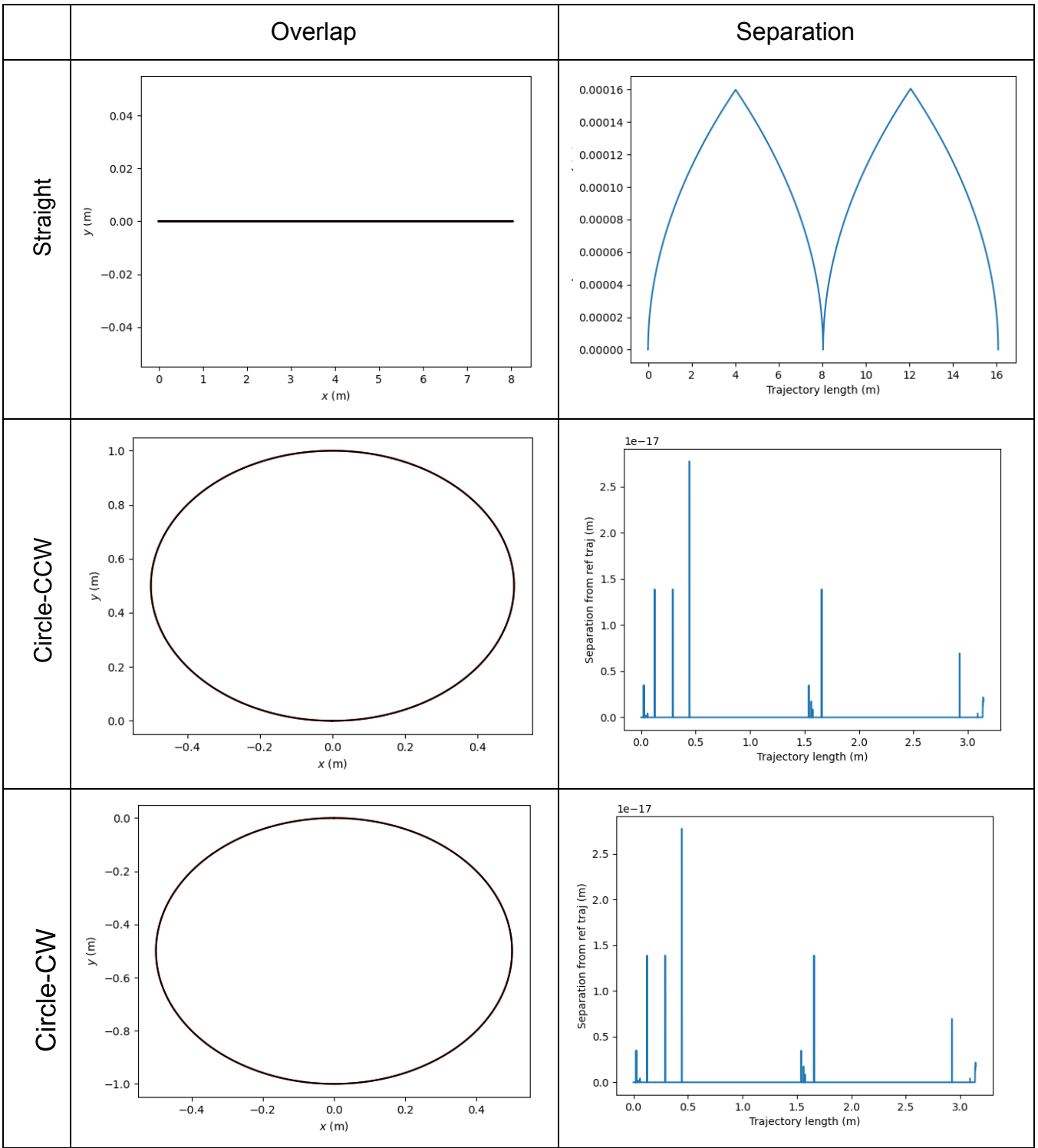


Fig. D5a: Corrected performance of “Noisy” robot when subject to some of the ideal robot’s wheel-frame control policies. The controller is based on system-identification of the baseline and wheel-radii.

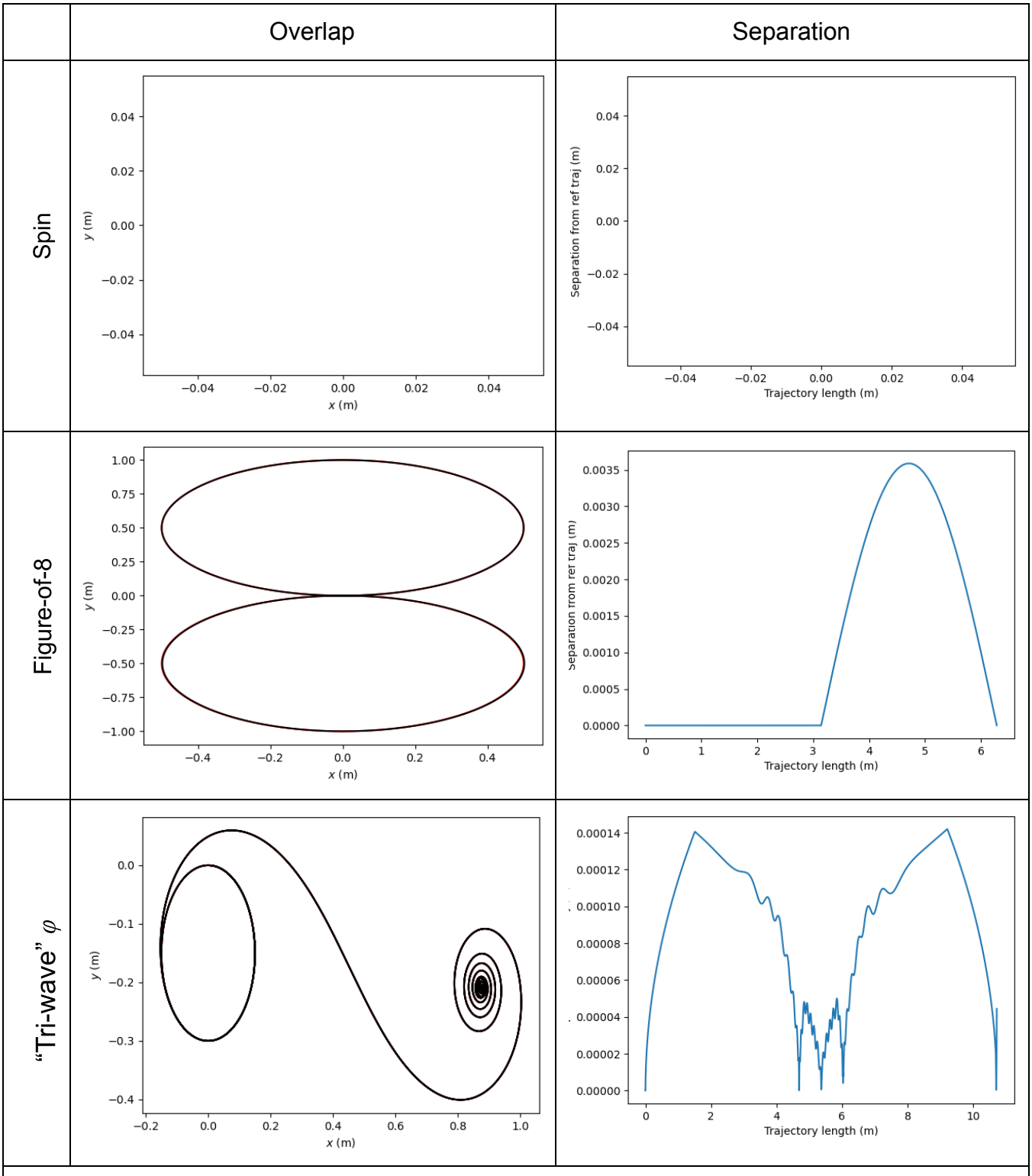


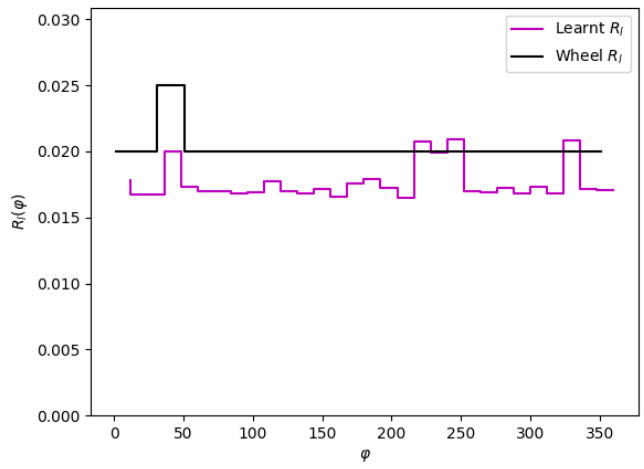
Fig. D5b: Corrected performance of "Noisy" robot when subject to the remainder of the ideal robot's wheel-frame control policies. The controller is based on system-identification of the baseline and wheel-radii variations.

The table below shows the maximum distance (meters) of executed trajectory from expectation, for

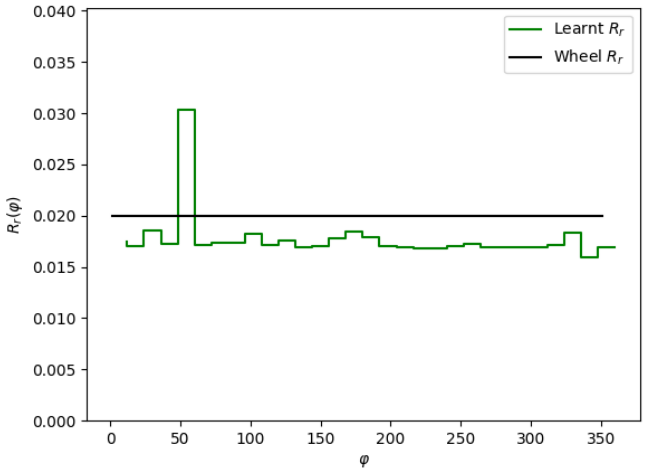
Straight	Circle-CCW	Circle-CW	Spin in-place	Figure-of-8	Tri-wave φ
0.0002	0	0	0	0.0036	0.0002

The learnt baseline was 10 cm which does not meet the actual baseline of 12 cm.

Fig. D5c shows the learnt radii as a function of wheel-angle, for the left and right wheels.



Left-wheel radii



Right-wheel radii

Fig. D5c: Learnt radii for left and right wheels for the “Noisy” robot.

The procedure transfers the learning entirely on to the wheel radii and ignores the baseline.

Appendix D6: SysID-based control of “Noisier” robot

Fig. D6a and D6b shows the paths generated by the robot with the noisier construction when using the System-Identification to create a kinematic controller to translate the desired body-frame control policy (v,ω) into actual wheel-frame control policy ($\dot{\varphi}_l$, $\dot{\varphi}_r$) specific to the robot with the smaller left wheel. Left column overlays the traversed path (black) on top of that traversed by the ideal robot model (red). The right column shows point-wise separation between the two paths along the trajectory.

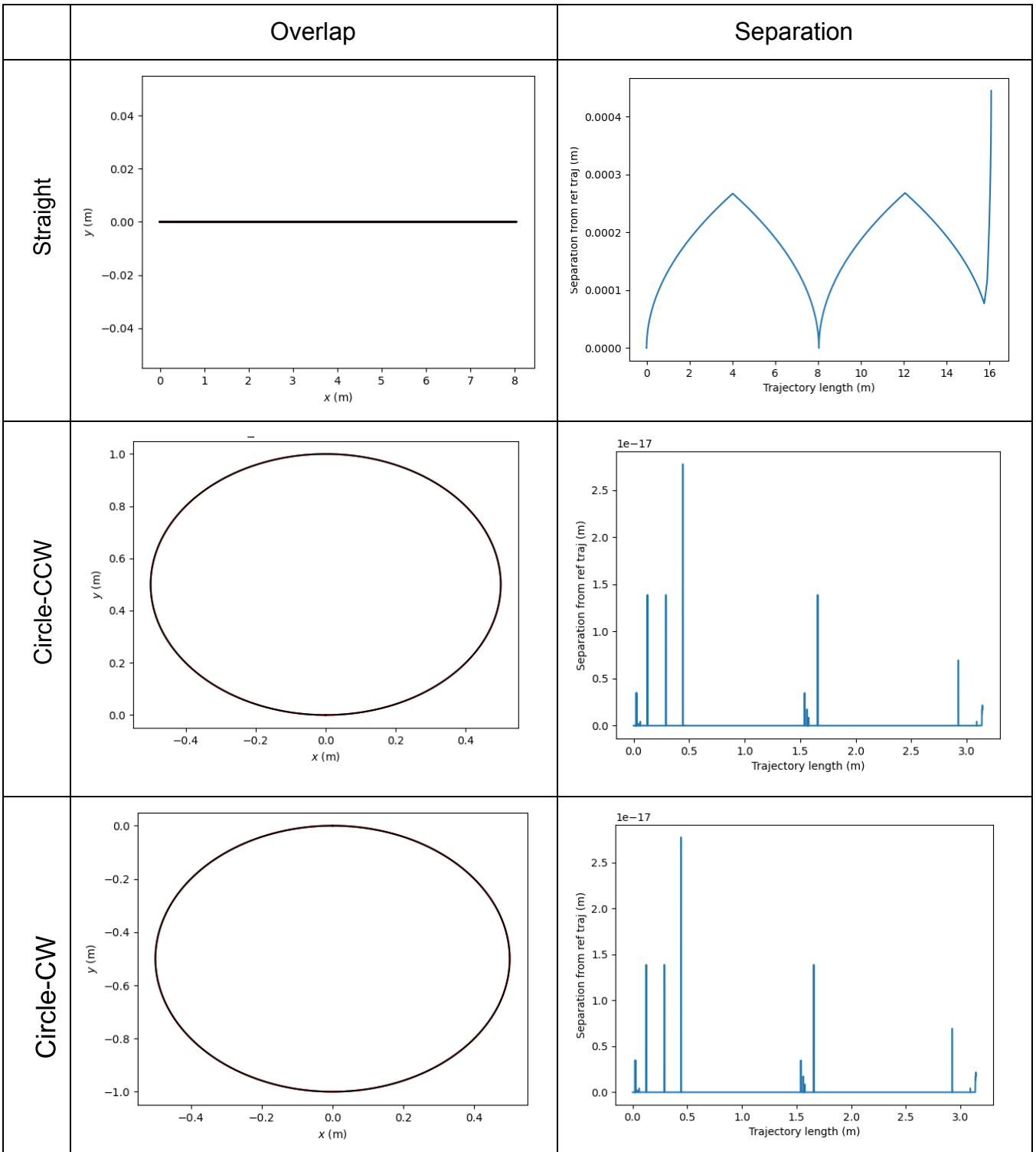


Fig. D6a: Corrected performance of “Noisier” robot when subject to some of the ideal robot’s wheel-frame control policies. The controller is based on system-identification of the baseline and wheel-radii.

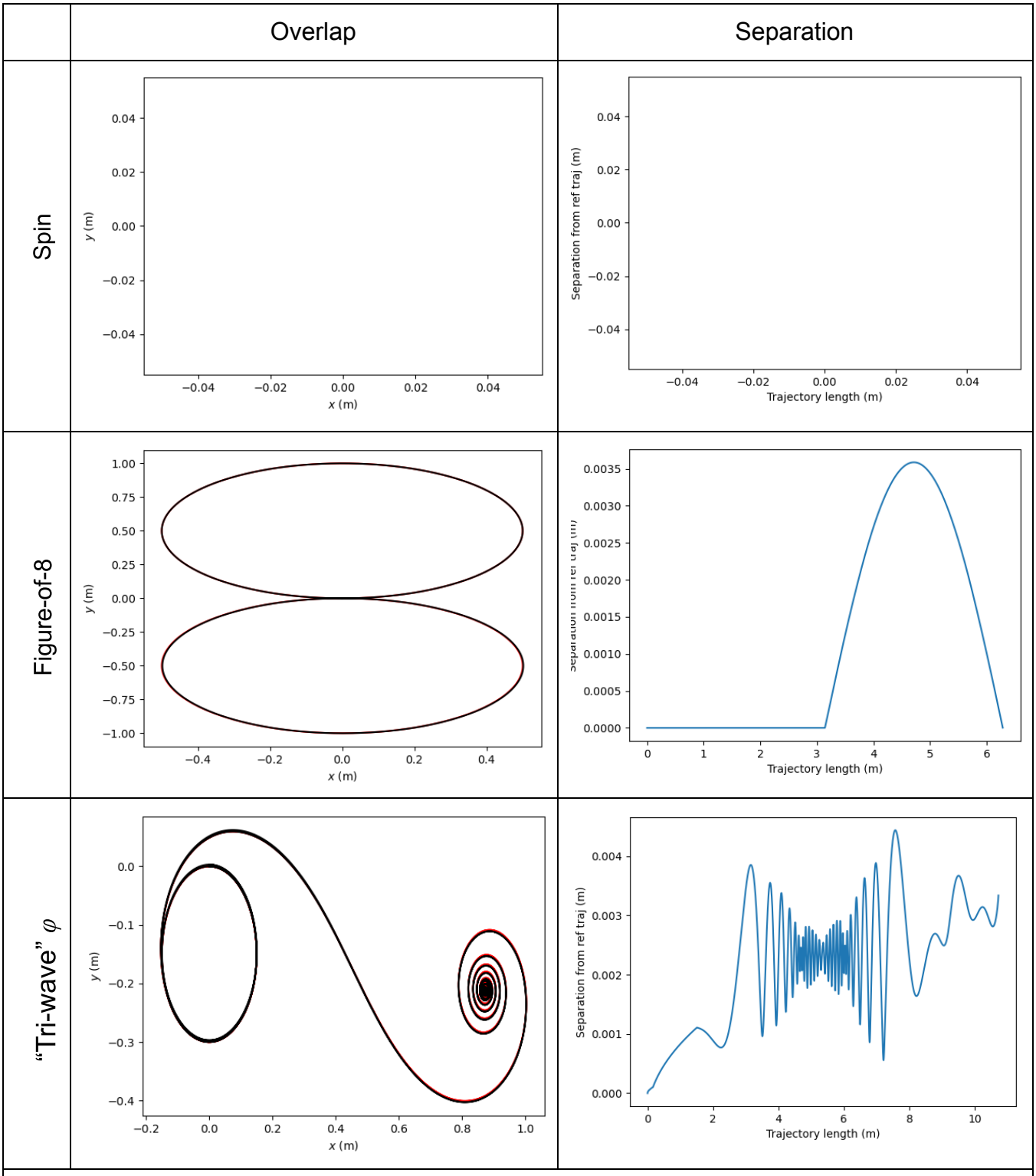


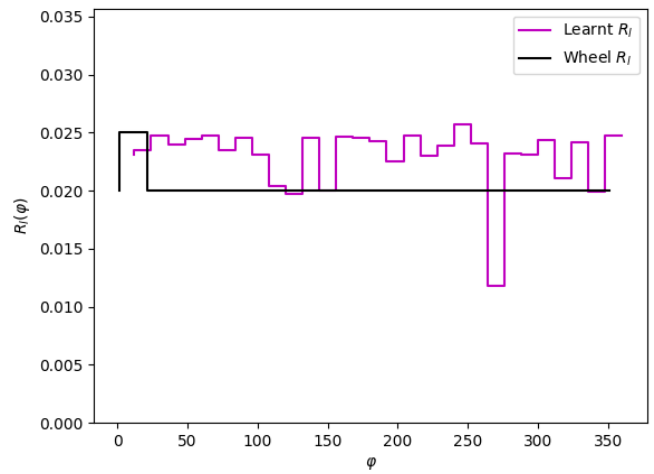
Fig. D6b: Corrected performance of "Noisier" robot when subject to the remainder of the ideal robot's wheel-frame control policies. The controller is based on system-identification of the baseline and wheel-radii variations.

The table below shows the maximum distance (meters) of executed trajectory from expectation, for

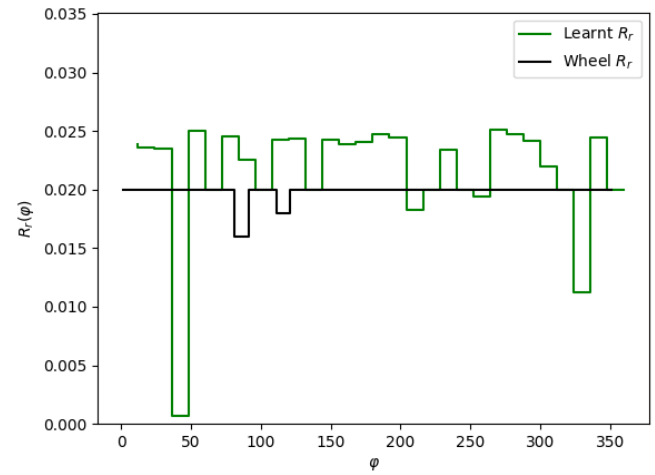
Straight	Circle-CCW	Circle-CW	Spin in-place	Figure-of-8	Tri-wave φ
0.0044	0	0	0	0.0033	0.0030

The learnt baseline was 10 cm which exceeds the actual baseline of 8 cm.

Fig. D6c shows the learnt radii as a function of wheel-angle, for the left and right wheels.



Left-wheel radii



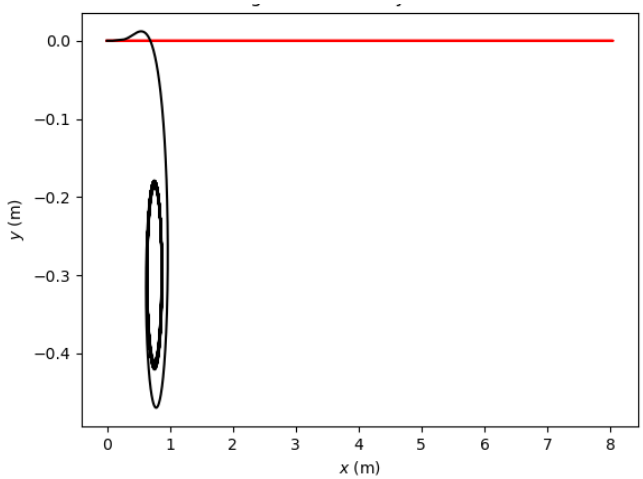
Right-wheel radii

Fig. D6c: Learnt radii for left and right wheels for the “Noisier” robot.

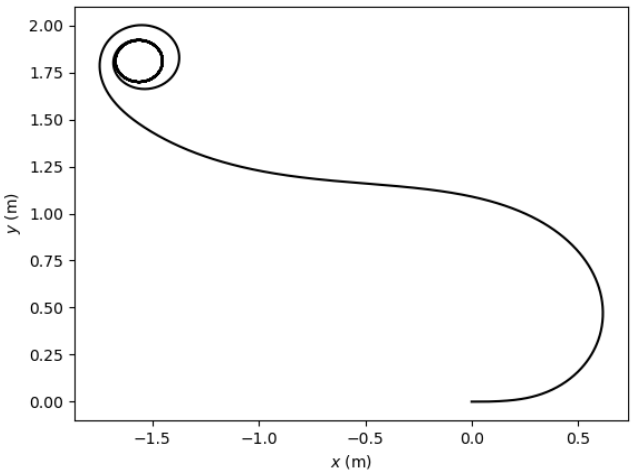
The procedure transfers the learning entirely on to the wheel radii and ignores the baseline.

Appendix E1: NN control of SmallerLeftWheel robot

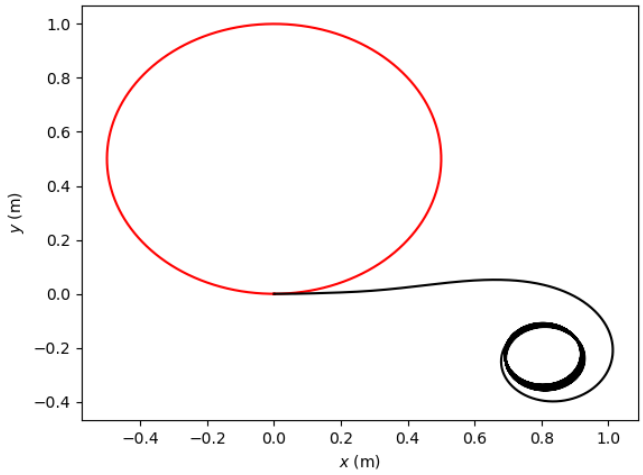
The figures below overlay the ideal trajectory (red) with that traversed by the robot with a kinematic controller incorporating the trained, fully-connected neural network.



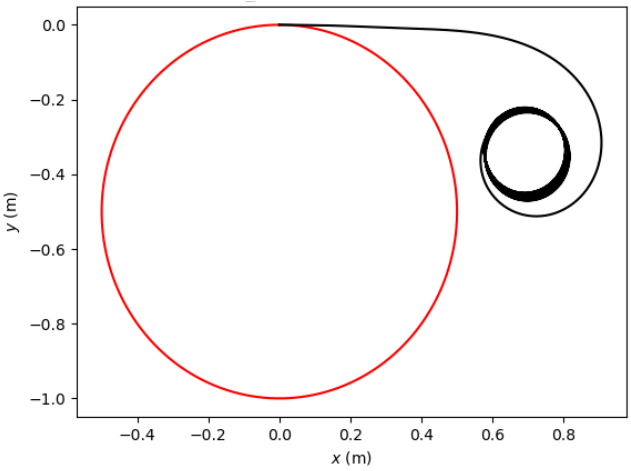
Straight line



Spin in-place



Circle-CCW



Circle-CW

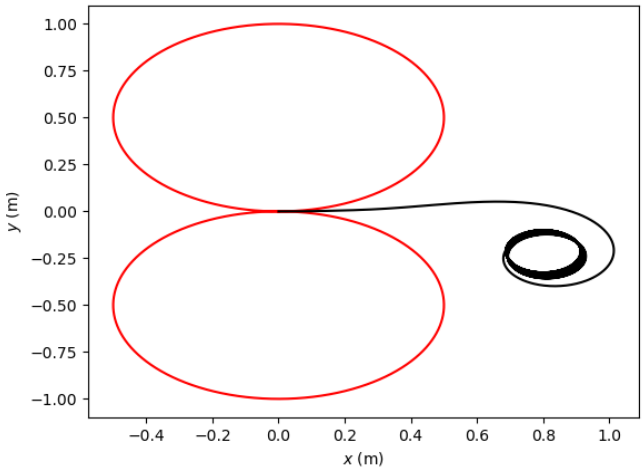
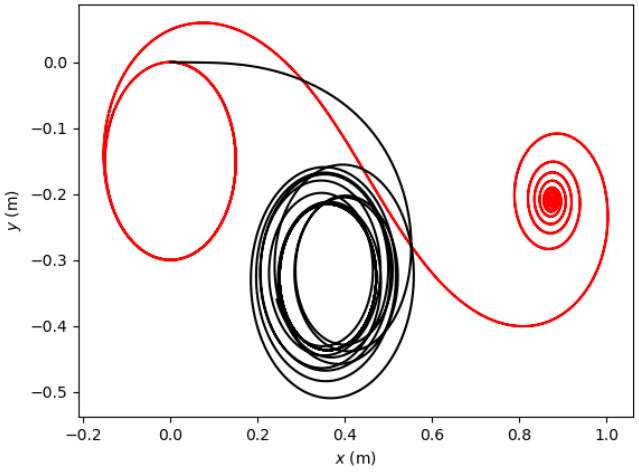


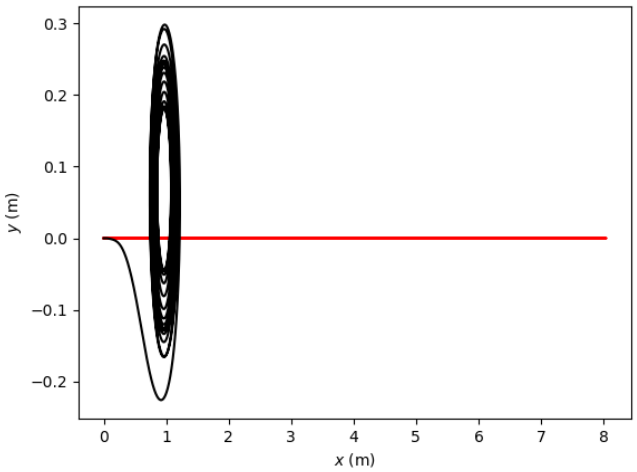
Figure-of-8



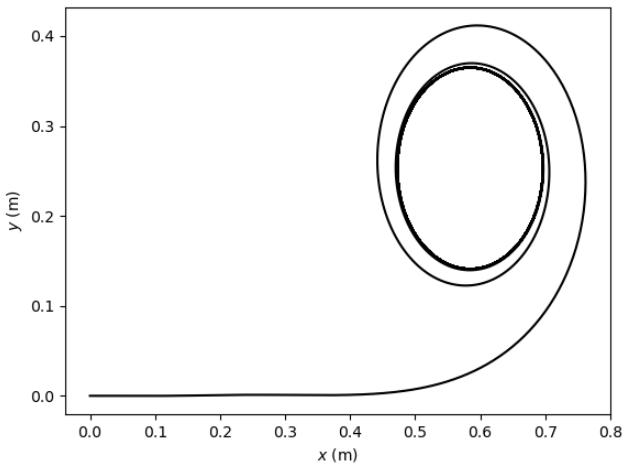
Tri-wave φ

Appendix E2: NN control of LargerLeftWheel robot

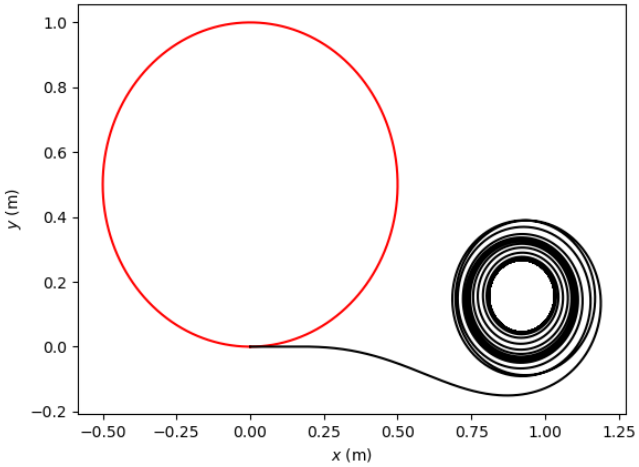
The figures below overlay the ideal trajectory (red) with that traversed by the robot with a kinematic controller incorporating the trained, fully-connected neural network.



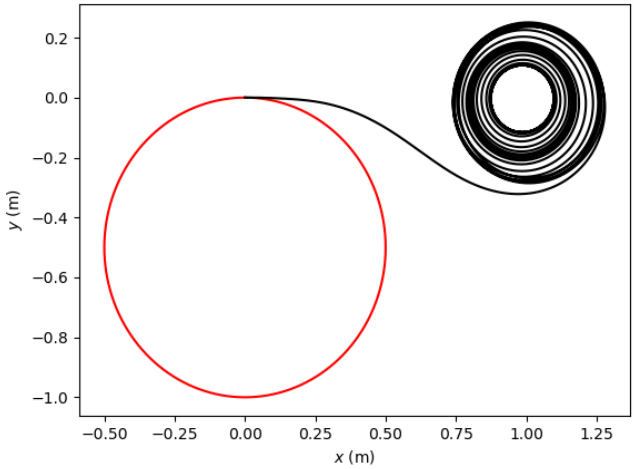
Straight line



Spin in-place



Circle-CCW



Circle-CW

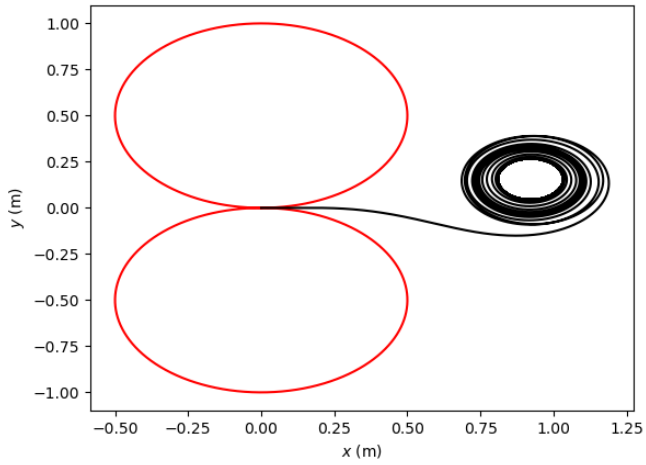
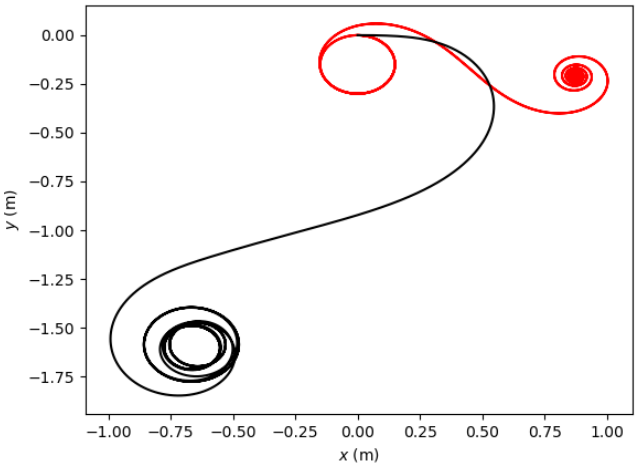


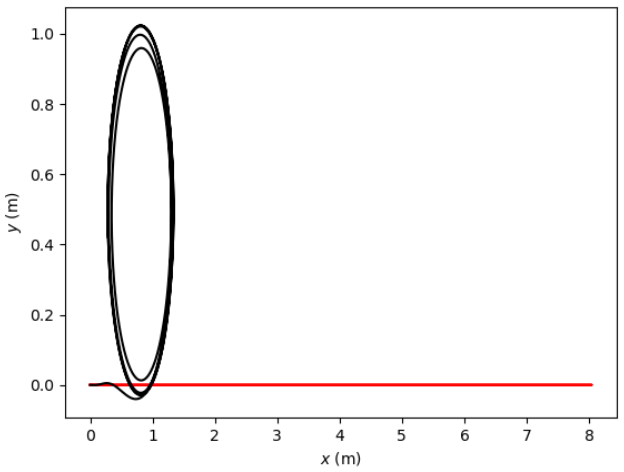
Figure-of-8



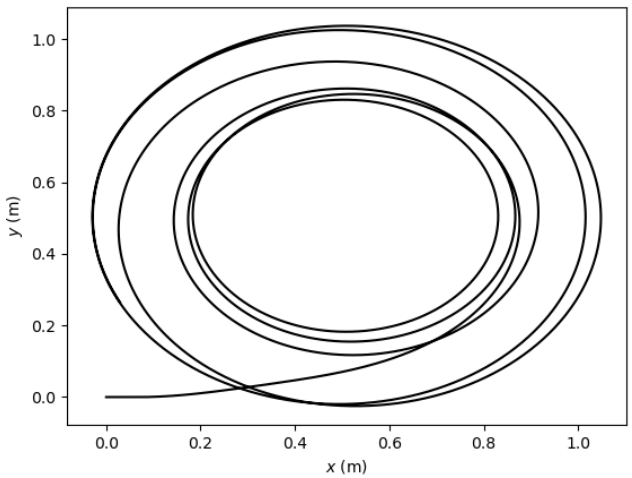
Tri-wave φ

Appendix E3: NN control of SmallerBaseline robot

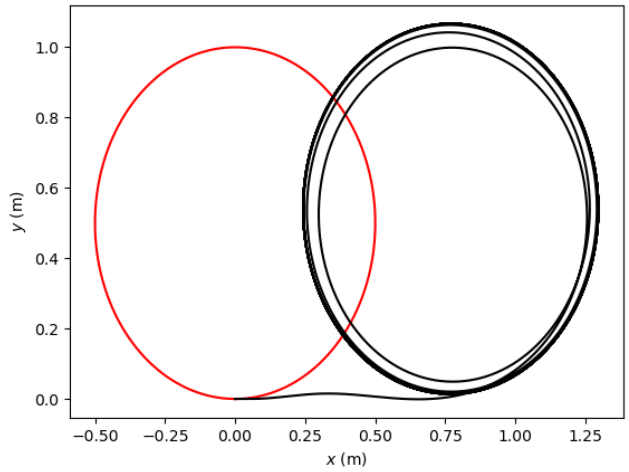
The figures below overlay the ideal trajectory (red) with that traversed by the robot with a kinematic controller incorporating the trained, fully-connected neural network.



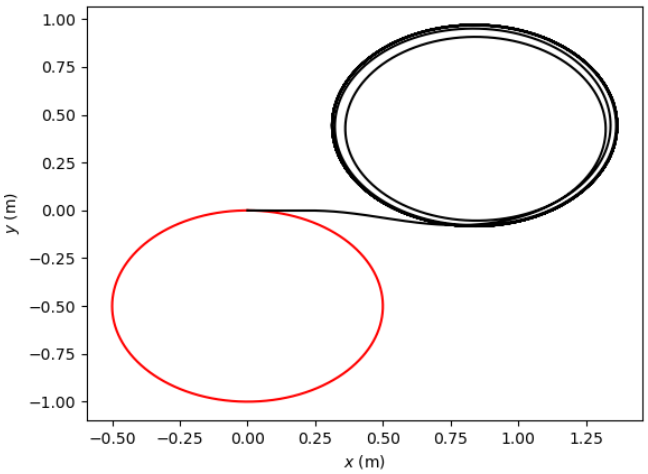
Straight line



Spin in-place



Circle-CCW



Circle-CW

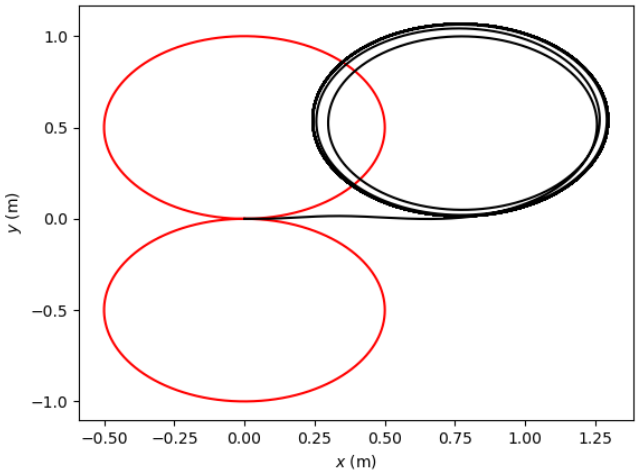
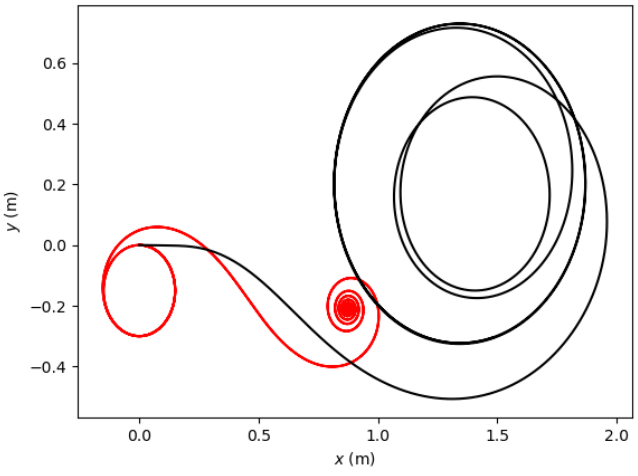


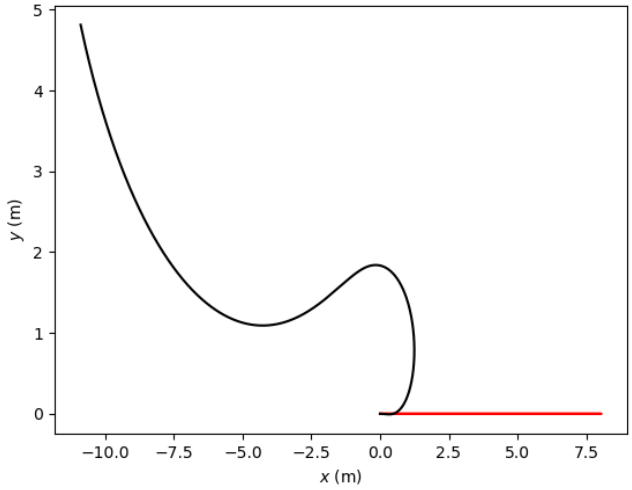
Figure-of-8



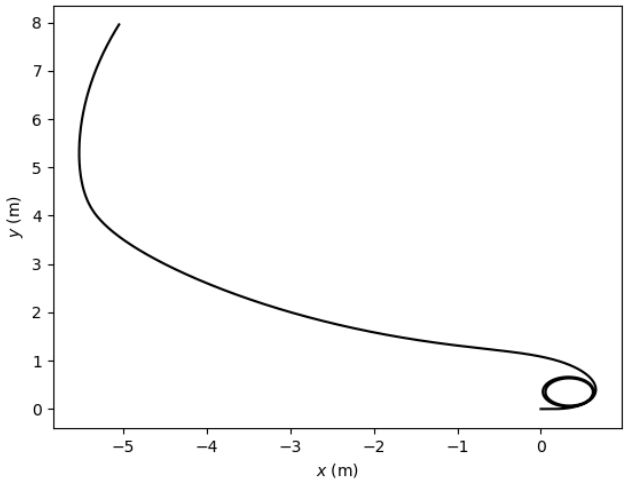
Tri-wave φ

Appendix E4: NN control of LargerBaseline robot

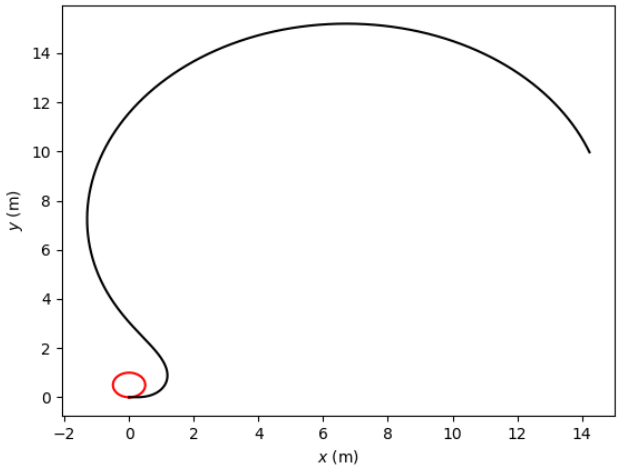
The figures below overlay the ideal trajectory (red) with that traversed by the robot with a kinematic controller incorporating the trained, fully-connected neural network.



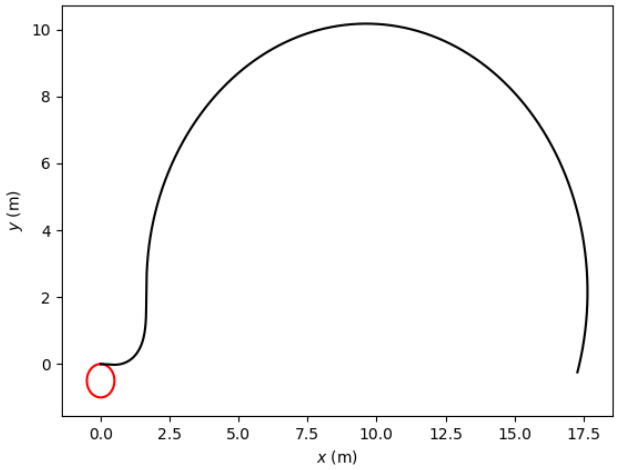
Straight line



Spin in-place



Circle-CCW



Circle-CW

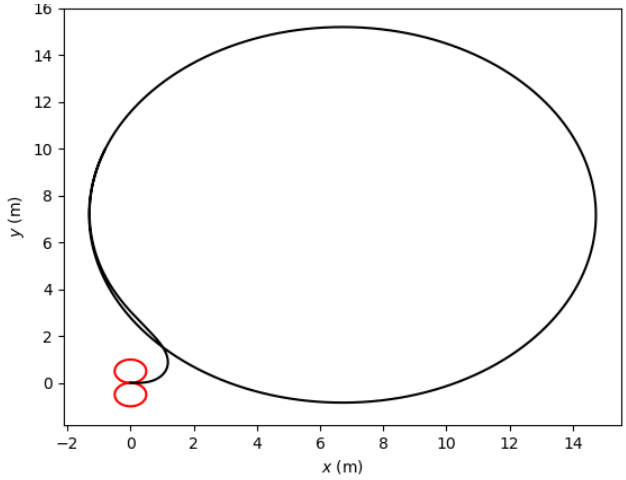
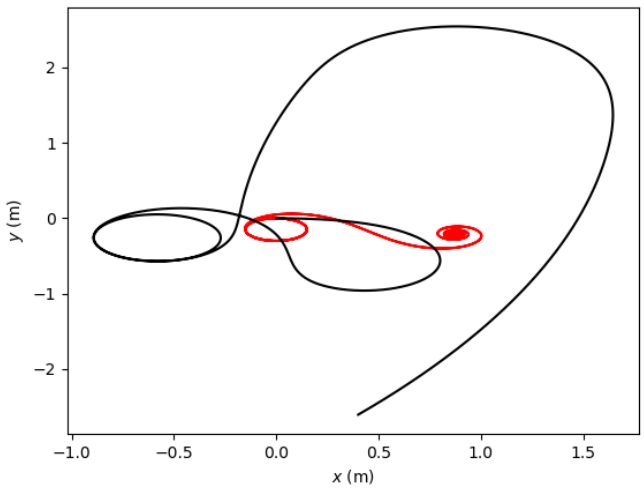


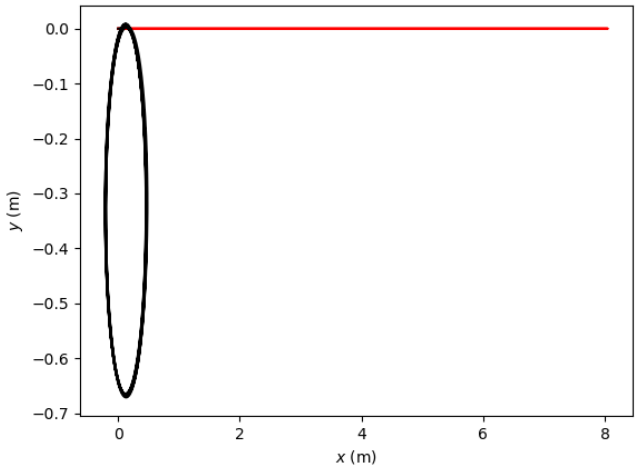
Figure-of-8



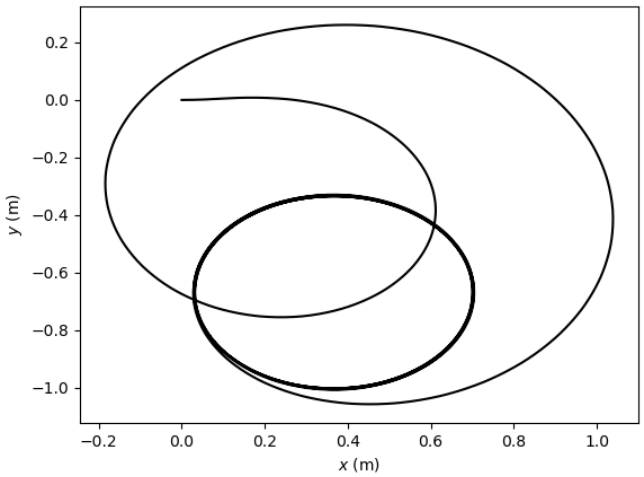
Tri-wave φ

Appendix E5: NN control of Noisy robot

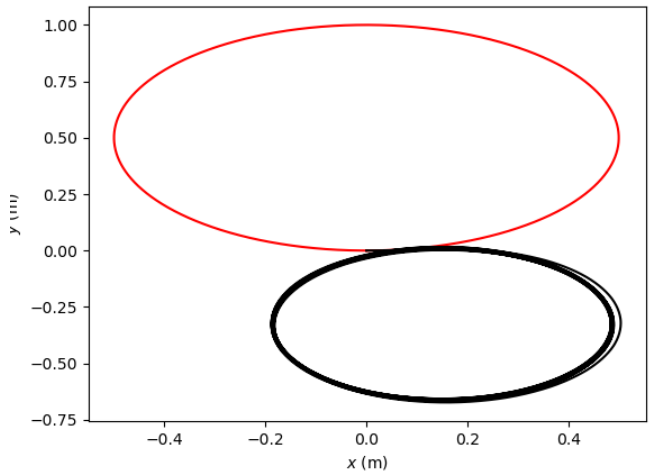
The figures below overlay the ideal trajectory (red) with that traversed by the robot with a kinematic controller incorporating the trained, fully-connected neural network.



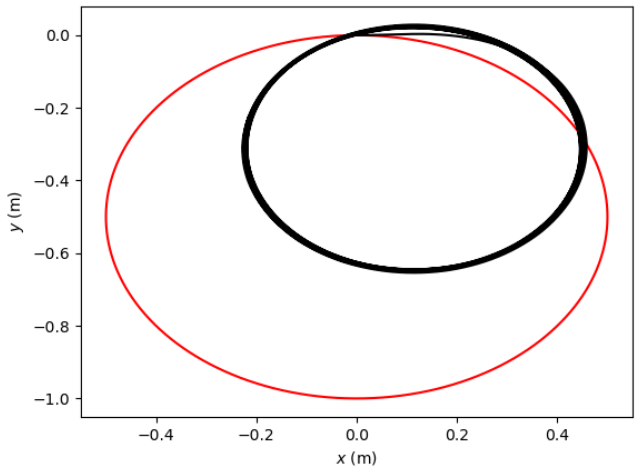
Straight line



Spin in-place



Circle-CCW



Circle-CW

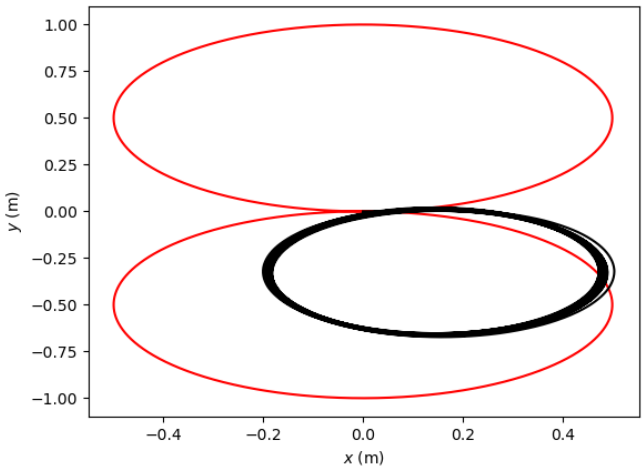
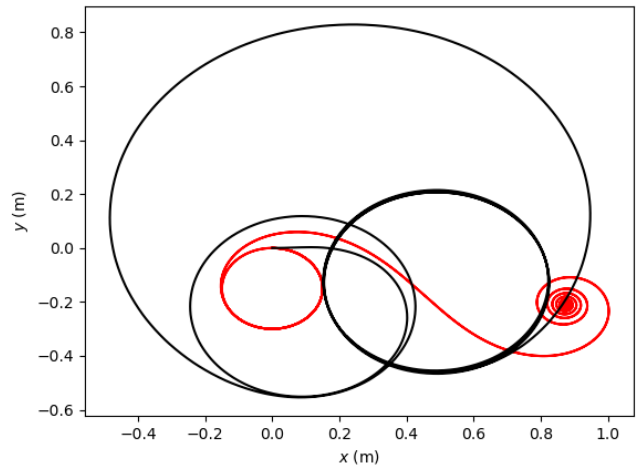


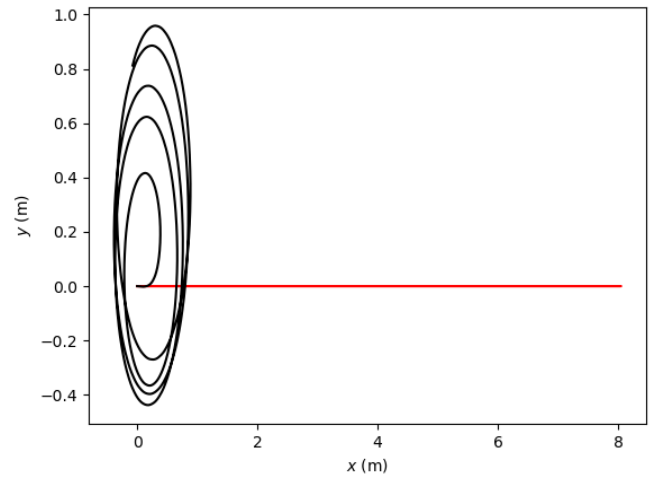
Figure-of-8



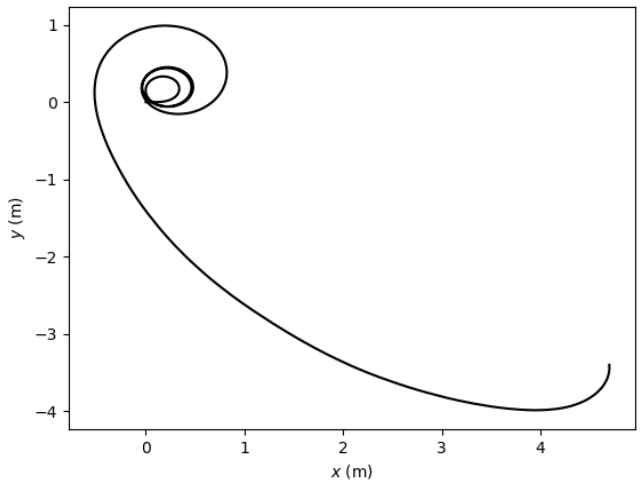
Tri-wave φ

Appendix E6: NN control of Noisier robot

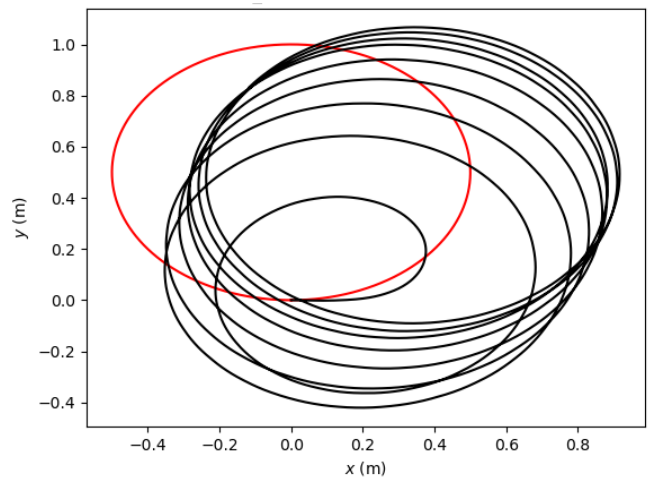
The figures below overlay the ideal trajectory (red) with that traversed by the robot with a kinematic controller incorporating the trained, fully-connected neural network.



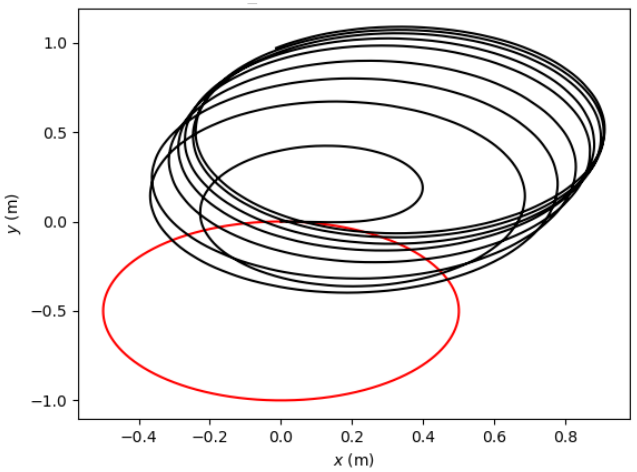
Straight line



Spin in-place



Circle-CCW



Circle-CW

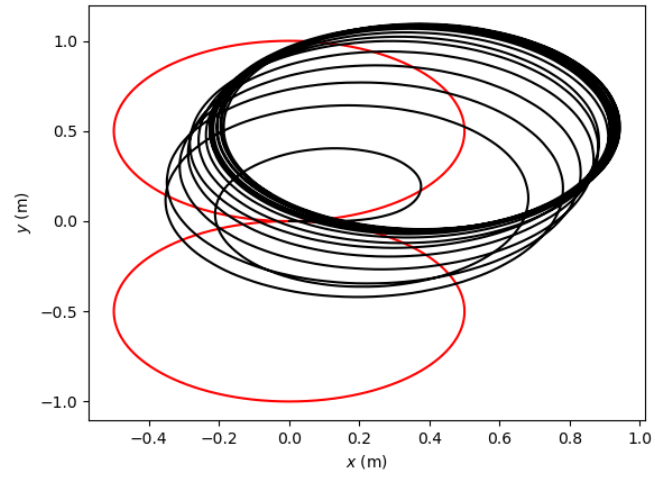
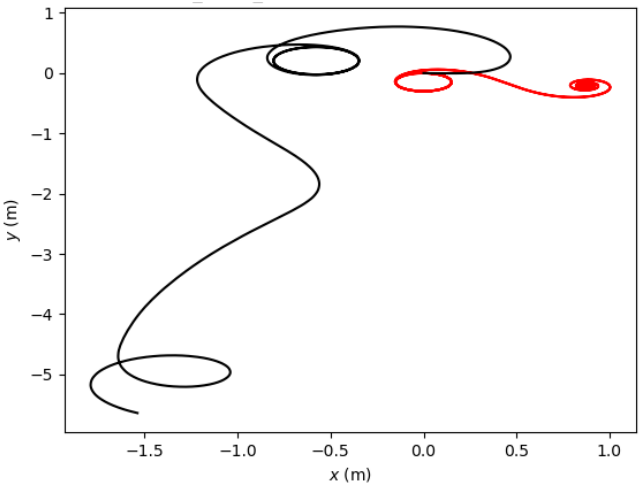


Figure-of-8



Tri-wave φ