A Dynamic Parameter Identification Method for Migrating Control Strategies Between Heterogeneous Wheeled Mobile Robots

by

Jeffrey Laut

A Thesis

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Master of Science

in

Mechanical Engineering

by

May 2011

APPROVED:

Professor Michael A. Demetriou, Thesis Advisor

Professor Stephen S. Nestinger, Thesis Advisor

Professor Gregory S. Fischer, Graduate Committee Member

Professor Nikolaos A. Gatsonis, Graduate Committee Representative

Abstract

Recent works on the control of wheeled mobile robots have shifted from the use of the kinematic model to the use of the dynamic model. Since theoretical results typically treat the inputs to the dynamic model as torques, few experimental results have been provided, as torque is typically not the input to most commercially available robots. Few papers have implemented controllers based on the dynamic model, and those that have did not address the issue of identifying the parameters of the dynamic model.

This work focuses on a method for identifying the parameters of the dynamic model of a wheeled mobile robot. The method is shown to be both effective and easy to implement, and requires no prior knowledge of what the parameters may be. Experimental results on two mobile robots of different scale demonstrate its effectiveness. The estimates of the parameters created by the proposed method are then used in an adaptive controller to verify their accuracy. For future work, this method should be completed autonomously in a two-part manner, onboard the mobile robot. First, the robot should perform the method proposed here to generate an initial parameter estimate, and then use adaptive control to update the estimates.

Acknowledgements

I would like to show my gratitude to my advisors, Prof. Demetriou and Prof. Nestinger, who have both given me excellent guidance and support.

Thanks to the Mechanical Engineering department for the supporting my education through a teaching assistantship. Working with Professors Furlong, Gatsonis, and Sullivan was a rewarding experience.

I am grateful to Prof. Fischer, who provided me with much appreciated feedback regarding the content and structure of my thesis.

Thanks to Yue Wang for the insightful conversations, and thanks to the rest of the friends I have made at WPI, including Jeff Court, Shalin Ye, Yao Wang, Mahdi Agheli, and Brendan Chenelle.

A special thanks to my parents. If it wasn't for them, I probably wouldn't have even gone to college!

Contents

1 Introduction **1**

2 Modeling Differential-Drive Wheeled Mobile Robots **4**

2.1 Kinematics of Wheeled Mobile Robots. 4

2.2 Parameterized Dynamic Model 8

3 Control Based on the Dynamic Model **15**

3.1 Odometry 16

3.2 Inverse Dynamic Control 17

3.3 Adaptation of Dynamic Parameters 21

4 Parameter Estimation **23**

4.1 Proposed method 23

4.2 Simulation Results 29

4.3 Experimental Setup 41

4.4 Experimental Results 44

4.4.1 Khepera III 46

4.4.2 Pioneer 3-DX 59

5 Conclusions and Future Work **69**

5.1 Conclusions 69

5.2 Future Work 70

A Generating a Trajectory **71**

A.1 Figure Eight 71

A.2 Figure Eight with Dither 73

A.3 Alterative Parameterized Dynamic Model 76

B Installing and Using the Khepera III Toolbox **80**

B.1 Installing the toolbox 80

B.2 Writing a program 81

List of Figures

2.1 Coordinates for a disk on plane [9] 5

2.2 Coordinates for unicycle on plane [9] 5

2.3 Differential-drive mobile robot 6

2.4 Terms of dynamic model of differential-drive mobile robot 8

4.1 Legend for parameter evolution plots 30

4.2 Perfect initial estimates 30

4.3 Effect of control gain on parameter convergence 31

4.4 Angular velocity and $\hat{\theta}_6$ 32

4.5 Parameter evolution for figure-eight trajectory 33

4.6 Parameter evolution for figure-eight trajectory with dither 34

4.7 Evolution of $\hat{\theta}_5$ 35

4.8 Parameter evolution 36

4.9 Parameter evolution 37

4.10 Regression 39

4.11 Khepera III and Pioneer 3-DX mobile robots 42

4.12 Khepera III trajectory and error, with adaptation, poor initial estimates 47

4.13 Velocity data 48

4.14 Acceleration data 49

4.15 Plots of linear regression 50
4.16 Khepera III trajectory and error, no adaptation 51
4.17 Khepera III trajectory and error, with adaptation 52
4.18 Comparison of figures 4.16 and 4.17 53
4.19 Khepera III trajectory and error with heavy payload, no adaptation 54
4.20 Khepera III trajectory and error with heavy payload, with adaptation. 55
4.21 Khepera III trajectory and error with light payload, no adaptation. . 56
4.22 Khepera III trajectory and error with light payload, with adaptation. 57
4.23 Comparison of Figures 4.21 and 4.22 58
4.24 Pioneer 3-DX trajectory and error, poor initial estimates, with adaptation. 60
4.25 Velocity data 61
4.26 Acceleration data 62
4.27 Plots of linear regression. 63
4.28 Pioneer 3-DX trajectory and error, adapting all parameters except $\hat{\theta}_4$ and $\hat{\theta}_6$ 65
4.29 Pioneer 3-DX trajectory and error, heavy payload, no adaptation . . 66
4.30 Pioneer 3-DX trajectory and error, heavy payload, with adaptation . 68

List of Tables

2.1 List of terms of dynamic model 9

4.1 Parameter final and initial values 33

4.2 Parameter final and initial values 34

4.3 Khepera III steady state velocity values 45

4.4 Pioneer 3-DX steady state velocity values 45

Chapter 1

Introduction

Wheeled mobile robots are becoming more popular for performing tasks that are too dangerous or tedious for humans, and with the increase in available technology, it is becoming more cost-effective to replace a worker with a mobile robot. Typical environments for wheeled mobile robots include warehouses, factories, military applications, and planetary exploration. In situations where the robot must maneuver with precision, a feedback controller for robot motion is necessary.

Regarding the control of wheeled mobile robots, early papers typically focused on using only the kinematics. It was shown through simulations in [1] that for high speed and high load situations, the dynamics of the robot should be considered for better performance in trajectory tracking. Most of the research regarding control based on the dynamic model has shown only simulation results.

For instance, [2] was the first paper to consider control using the dynamic model with unknown dynamics, and provided torques as inputs to the model, and showed simulation results. [3] provided a controller based on the dynamic model that took inputs from a kinematic controller, and altered them to compensate for the dynam-

ics of a robot. In [4], a combined kinematic and dynamic control law was presented. Their model also assumed torques as input, and provided only simulation results. [5] created a simplified dynamic model, and analyzed the influence of the uncertainties. They too treated the inputs as torques, and provided only simulation results.

Most of the controllers based on the dynamic model considered torques as the inputs, however most commercially available robots do not take torques as inputs. In an effort to design a more practical controller, [6] developed a controller that used voltages as inputs to the dynamic model, but only provided simulation results. [7] created a controller that gave current as input to the dynamic model, and provided experimental results on a custom robot for adaptive steering and adaptive cruise control. More recently [8] provided a dynamic model based on the parameterization from [5], which provided reference velocities as inputs. This is the most practical situation, as most commercially available robots take reference velocities as inputs.

A field of research that seems to have remained mostly untouched, likely due to only the recent use of the dynamic model in control of mobile robots in experiments, is the identification of the parameters of wheeled mobile robots. This work addresses the problem of creating an initial estimate for the parameters of a wheeled mobile robot, and provides experimental results showing both its need and effectiveness.

This thesis is organized as follows; In Chapter 2, first the kinematics of a differential drive mobile robot are reviewed. The dynamic model of a mobile robot which takes velocities as inputs, and includes actuator dynamics is then derived, followed by an overview of odometry as a feedback mechanism. Controlling the robot using the dynamic model is discussed in Chapter 3, assuming that the constants the represent

physical parameters of the robot are perfectly known. To overcome this assumption, an adaptive controller which updates the estimates of the unknown parameters of the robot is shown. To implement the adaptive controller on physical robots, a reasonable initial estimate of the parameters must first be generated. The main contribution is located in Chapter 4, which covers how to create an appropriate initial estimate. Simulation results are provided, and experimental results for two different robots are shown at the end of the chapter.

In the future, we would like to have a two-part scheme that first executes the proposed method to generate initial parameter estimates, and then uses adaptive control to further tune the parameters. This scheme should be completed entirely on-board the robot. Furthermore, we would like the scheme to be easily transferable between heterogeneous mobile robots.

Chapter 2

Modeling Differential-Drive Wheeled Mobile Robots

2.1 Kinematics of Wheeled Mobile Robots.

The pose a mobile robot is defined by three coordinates represented by the vector $\boldsymbol{\xi}$,

$$\boldsymbol{\xi} = \begin{bmatrix} x \\ y \\ \phi \end{bmatrix} \tag{2.1}$$

The set of reachable poses defined by $\mathcal{C}$ encompasses the plane that the robot is maneuvering about. The motion of $\boldsymbol{\xi}$, however, is subject to a nonholonomic constraint. This constraint means that the kinematic model of the robot is not capable of producing instantaneous motion in any direction. To illustrate this, we observe the kinematic constraint of a unicycle model on a plane. Since the unicycle cannot move in the direction perpendicular to the plane containing the wheel, its motion is constrained by $\dot{x}\sin\phi = \dot{y}\cos\phi$. Although this constraint restricts the robot from

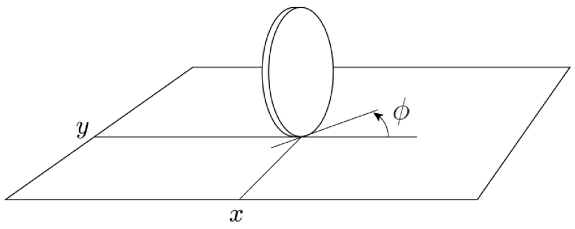


Figure 2.1: Coordinates for a disk on plane [9]

instantaneously moving in any direction, it does not restrict the robot from reaching any pose on the plane it is moving about [9]. The kinematic model of the unicy-

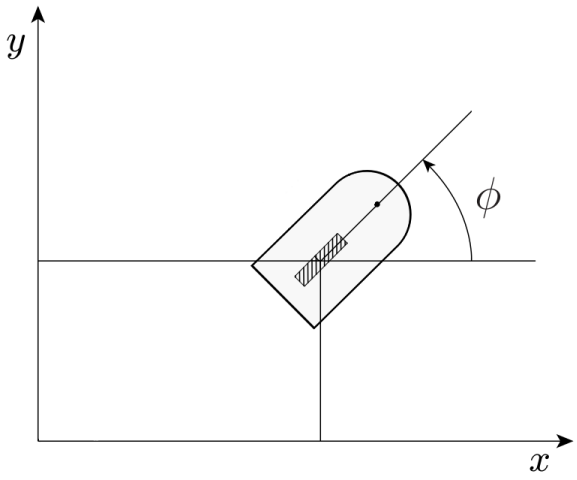


Figure 2.2: Coordinates for unicycle on plane [9]

cle robot relates the forward and angular velocities (v, ω) to Cartesian velocities $(\dot{x}, \dot{y}, \dot{\phi})$. The x component of the forward velocity can be expressed as $\dot{x} = v\cos\phi$, the y component can be expressed as $\dot{y} = v\sin\phi$. The kinematic model can then be represented as

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} \cos\phi & 0 \\ \sin\phi & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{2.2}$$

or

$$\dot{\boldsymbol{\xi}} = \mathbf{S}(\boldsymbol{\xi})\mathbf{v} \tag{2.3}$$

where $\mathbf{v} = \begin{bmatrix} v & \omega \end{bmatrix}^\top$ and $\mathbf{S}(\boldsymbol{\xi})$ is the transformation matrix that relates $\mathbf{v}$ and $\dot{\boldsymbol{\xi}}$.

A differential-drive robot has the same kinematic constraint and kinematic model as the unicycle model. The only additional work needed is to relate the forward and angular velocities of the unicycle with the individual wheel velocities for the differential drive robot.

$$v = \frac{r(\omega_r + \omega_l)}{2} \quad , \quad \omega = \frac{r(\omega_r - \omega_l)}{d} \tag{2.4}$$

where ω_r and ω_l are the right and left wheel rotational velocities, r is the radius of

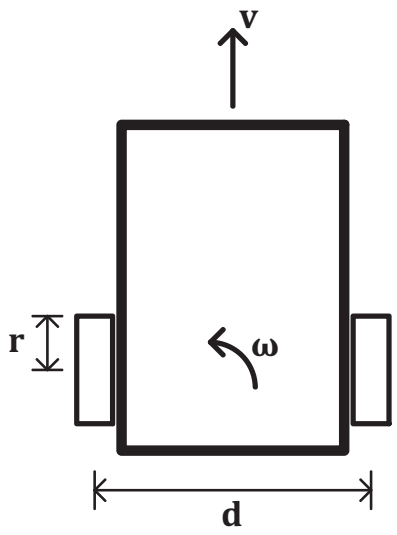


Figure 2.3: Differential-drive mobile robot

the wheels, and d is the distance between the wheels (Figure 2.3).

From the two equations above, the individual wheel velocities can be found as a

function of v and ω,

$$\omega_r = \frac{2v + d\omega}{2r} \quad , \quad \omega_l = \frac{2v - d\omega}{2r} \tag{2.5}$$

Typically, throughout an analysis of a control scheme for a differential drive robot, the unicycle model is used. In this case, the inputs to the kinematic model are still v and ω, and the wheel velocities are only calculated when the controller is physically implemented.

This section provided information on transforming between a robot's forward and angular velocities to Cartesian velocities. As stated previously, early works focused on controlling a mobile robot using only the kinematic model. This assumes that any velocity v or ω given to the robot is instantaneously achieved, and does not account for dynamic effects of the robot. To accurately model the motion of a wheeled mobile robot, the dynamics should be accounted for [1], as a mobile robot in practice cannot achieve instantaneous acceleration. The inputs to the dynamic model generate the forward and angular velocities of the mobile robot, which can then be used with the kinematic model to provide Cartesian velocities. The next section derives a parameterized dynamic model, where the parameters are physical constants of the model, and vary from robot to robot.

2.2 Parameterized Dynamic Model

A parameterized dynamic model of a differential-drive mobile robot, including actuator dynamics, was presented in [5]. This model treated the inputs to the motors as torques. In [8], the same parameterized model was used, with the alteration of using reference velocities as the motor inputs. Their model considered a proportional-derivative controller on the forward and angular velocity inputs. Another common configuration is to have proportional-derivative control on the individual wheel velocities. In this section, the parameterized dynamic model for a mobile robot with proportional-derivative control on the individual wheel velocities is derived. The parameterized dynamic model for a mobile robot with a proportional-derivative controller on the forward and angular velocities is shown in Appendix A.3.

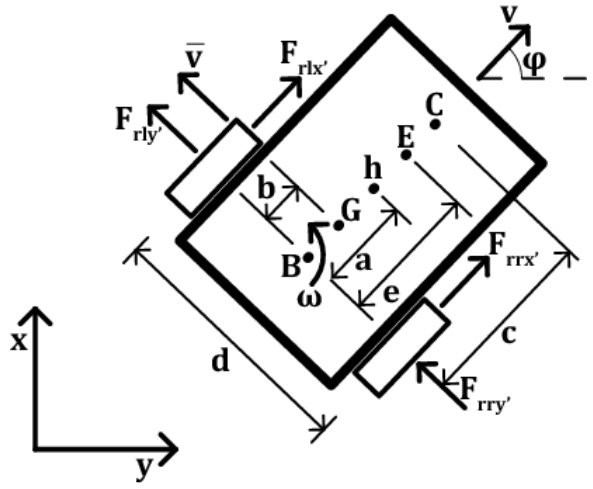


Figure 2.4: Terms of dynamic model of differential-drive mobile robot

G	location of center of mass 重心
B	location of wheel baseline center 两轮中心
E	location of external force 外力作用点
C	location of castor wheel 前轮位置
h	point being tracked 车轨迹点
v	forward velocity 前向速度
$\bar{v}$	lateral velocity 侧向速度
ω	angular velocity 偏转角速度
φ	heading 偏转角
d	distance between driving wheels 车轮距离
b	distance from center of gravity to axle
a	distance from axle to point being tracked
e	distance from axle to external force
c	distance from axle to castor wheel
$F_{rrx'}$	longitudinal tire forces of right wheel 右轮纵向手里
$F_{rry'}$	lateral tire forces of right wheel 右轮横向受力
$F_{rlx'}$	longitudinal tire forces of left wheel
$F_{rly'}$	lateral tire forces of left wheel
$F_{cx'}$	longitudinal force exerted on C by castor
$F_{cy'}$	lateral force exerted on C by castor
$F_{ex'}$	longitudinal force exerted on E
$F_{ey'}$	lateral force exerted on E
τ_e	moment exerted on E

Table 2.1: List of terms of dynamic model

First, we find the robot force and moment equations about local coordinate frame:

$$\begin{aligned}
\Sigma F_{x'} &= m(\dot{v} - \bar{v}\omega) = F_{rlx'} + F_{rrx'} + F_{ex'} + F_{cx'} \\
\Sigma F_{y'} &= m(\dot{\bar{v}} - v\omega) = F_{rly'} + F_{rry'} + F_{ey'} + F_{cy'} \\
\Sigma M_{z'} &= I_z\dot{\omega} = \frac{d}{2}(F_{rrx'} - F_{rlx'}) - b(F_{rly'} + F_{rry'}) \cdots \\
&\quad +(e-b)F_{ey'} + (c-b)F_{cy'} + \tau_e,
\end{aligned} \tag{2.6}$$

where m is the mass of the robot and I_z is the robot moment of inertia about the local z' axis.

The kinematics of the point being tracked, h, are;

$$\begin{aligned}
\dot{x} &= v\cos\psi - \bar{v}\sin\psi - (a-b)\omega\sin\psi \\
\dot{y} &= v\sin\psi - \bar{v}\cos\psi + (a-b)\omega\cos\psi,
\end{aligned} \tag{2.7}$$

with

$$\begin{aligned}
v &= \frac{1}{2}[r(\omega_r + \omega_l) + (v_r^s + v_l^s)] \\
\omega &= \frac{1}{d}[r(\omega_r - \omega_l) + (v_r^s - v_l^s)] \\
\bar{v} &= \frac{b}{d}[r(\omega_r - \omega_l) + (v_r^s + v_l^s)] + \bar{v}^s,
\end{aligned} \tag{2.8}$$

where v is the forward velocity, ω is the angular velocity, $\bar{v}$ is the lateral velocity, v_r and v_l are the left and right linear velocities of each wheel, and any velocity with an s superscript is a velocity due to wheel slippage.

The dynamic model of the drive motors are obtained by neglecting the voltage on the inductance as:

$$\tau_r = k_a(v_r - k_b\omega_r)/R_a \qquad \tau_l = k_a(v_l - k_b\omega_l)/R_a, \tag{2.9}$$

where v_r and v_l are the input voltages applied to the right and left motors, k_b is the motor voltage constant multiplied by the gear ratio, R_a is the electrical resistance constant, τ_r and τ_l are the right and left motor torques multiplied by the gear ratio, and k_a is the torque constant multiplied by the gear ratio.

The dynamics of the combined wheels and motors are:

$$\begin{aligned} I_e\dot{\omega}_r + B_e\omega_r &= \tau_r - F_{rrx'}R_t \\ I_e\dot{\omega}_l + B_e\omega_l &= \tau_l - F_{rlx'}R_t, \end{aligned} \tag{2.10}$$

where I_e is the moment of inertia and B_e is the viscous friction coefficient of the motor, gearbox, and wheel combination, and R_t is the radius of the tire.

The voltage applied to each motor is given by the output of the onboard motor controller. This is a proportional-derivative controller, and the dynamics of the motor input voltage are:

$$\begin{aligned} v_l &= k_P(\omega_{l_{ref}} - \omega_l) + k_D\dot{\omega}_l \\ v_r &= k_P(\omega_{r_{ref}} - \omega_r) + k_D\dot{\omega}_r, \end{aligned} \tag{2.11}$$

where $\omega_{l_{ref}}$ and $\omega_{r_{ref}}$ are the commanded left and right wheel velocities, and ω_l and

ω_r are the actual wheel velocities. The derivatives of the commanded velocities were neglected since the robot cannot be given a reference acceleration.

The above equations are now manipulated in order to arrive at the parameterized dynamic model of the robot.

From the second and third equations of (2.6) we can solve for the angular acceleration,

$$\dot{\omega} = \frac{d}{2I_z}(F_{rrx'} - F_{rlx'}) + \frac{e}{I_z}F_{ey'} + \frac{c}{I_z}F_{cy'} + \frac{1}{I_z}\tau_e - \frac{mb}{I_z}(\dot{\bar{v}} + v\omega). \tag{2.12}$$

Combining equations (2.9) and (2.10),

$$\begin{aligned} F_{rrx'} &= \frac{k_a}{R_a R_t}(v_r - k_b\omega_r) - \frac{I_e}{R_t}\dot{\omega}_r - \frac{B_e}{R_t}\omega_r \\ F_{rlx'} &= \frac{k_a}{R_a R_t}(v_l - k_b\omega_l) - \frac{I_e}{R_t}\dot{\omega}_l - \frac{B_e}{R_t}\omega_l, \end{aligned} \tag{2.13}$$

and from equation (2.9),

$$\begin{aligned} \omega_r + \omega_l &= \frac{2}{r}v - \frac{1}{r}(v_r^s + v_l^s) \\ \omega_r - \omega_l &= \frac{d}{r}\omega - \frac{1}{r}(v_r^s - v_l^s). \end{aligned} \tag{2.14}$$

The above equation along with (2.11) gives,

$$
\begin{aligned}
v_r + v_l &= k_P(\omega_{r_{ref}} + \omega_{l_{ref}}) - k_D(\dot{\omega}_r + \dot{\omega}_l) - \frac{2k_P}{r}v + \frac{k_P}{r}(v_r^s + v_l^s) \qquad (2.15)\\
v_r - v_l &= k_P(\omega_{r_{ref}} - \omega_{l_{ref}}) - k_D(\dot{\omega}_r - \dot{\omega}_l) - \frac{k_P d}{r}\omega + \frac{k_P}{r}(v_r^s - v_l^s).
\end{aligned}
$$

From equation (2.8),

$$
\begin{aligned}
\omega_r + \omega_l &= \frac{2v - (v_r^s + v_l^s)}{r} \qquad (2.16)\\
\omega_r - \omega_l &= \frac{\omega d - (v_r^s - v_l^s)}{r}.
\end{aligned}
$$

Combining the lateral forces acting on each wheel (equation (2.13)) and the angular acceleration (equation (2.12)), and using the second equations of (2.14) and (2.15) as well as the above equation, and using the following parameterization,

$$
\begin{aligned}
\theta_1 &= \frac{R_a R_t m + 2k_a k_D + 2I_e R_a}{2k_a k_P}\\
\theta_2 &= \frac{k_a k_D d^2 + I_e R_a d^2 + 2R_a R_t r(mb^2 + I_z)}{k_a k_P d^2}\\
\theta_3 &= \frac{R_a R_t r m}{2k_a k_P}\\
\theta_4 &= \frac{k_a(k_P + k_b) + 2R_a B_e}{2k_a k_P}\\
\theta_5 &= \frac{2R_a R_t r m b}{k_a k_P d^2}\\
\theta_6 &= \frac{R_a R_t(k_P + k_b) + R_a B_e}{k_a k_P}
\end{aligned}
$$

gives,

$$
\begin{aligned}
\theta_2\dot{\omega} &= \omega_{ref} - \theta_6\omega + \frac{1}{d}\theta_6(v_r^s - v_l^s) + \frac{k_a k_D + R_a I_e}{k_a k_P d}(\dot{v}_r^s - \dot{v}_l^s) +\\
&\quad \frac{2R_a R_t r}{k_a k_p d^2}(eF_{ey'} + cF_{cy'} + \tau_e) - \theta_5\dot{\bar{v}}^s - \theta_5 v\omega. \qquad (2.17)
\end{aligned}
$$

Similarly, using the lateral forces, the sum of forces about the x' axis, and the first equations of (2.14) and (2.15) with (2.16),

$$\begin{aligned}\theta_1\dot{v} &= v_{ref} - \theta_4 v + \frac{1}{2}\theta_4(v_r^s + v_l^s) + \frac{k_a k_D + R_a I_e}{2k_a k_P}(\dot{v}_r^s + \dot{v}_l^s) + \\ &\quad \frac{R_a R_t r}{2k_a k_P}(F_{ex'} + F_{cx'}) + \theta_3\omega^2 + \frac{R_a R_t rm}{2k_a k_P}\omega\bar{v}^s. \end{aligned} \tag{2.18}$$

The combination of equations (2.16) and (2.7), along with the above two equations gives the dynamic model,

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\psi}\\ \dot{v}\\ \dot{\omega}\end{bmatrix} = \begin{bmatrix}v\cos\psi - a\omega\sin\psi\\ v\sin\psi + a\omega\cos\psi\\ \omega\\ \frac{\theta_3}{\theta_1}\omega^2 - \frac{\theta_4}{\theta_1}v\\ -\frac{\theta_5}{\theta_2}v\omega - \frac{\theta_6}{\theta_2}\omega\end{bmatrix} + \begin{bmatrix}0 & 0\\ 0 & 0\\ 0 & 0\\ \frac{1}{\theta_1} & 0\\ 0 & \frac{1}{\theta_2}\end{bmatrix}\begin{bmatrix}v_{ref}\\ \omega_{ref}\end{bmatrix} + \begin{bmatrix}\delta_x\\ \delta_y\\ 0\\ \bar{\delta}_v\\ \bar{\delta}_\omega\end{bmatrix} \tag{2.19}$$

where

$$\begin{bmatrix}\delta_x & \delta_y & 0 & \bar{\delta}_v & \bar{\delta}_\omega\end{bmatrix}^\top$$

is the vector of uncertainty associated with the slip velocities and external forces.

The above model is for a robot that has proportional-derivative control on the forward and angular velocities of the robot. Another common method for regulating the velocity of the robot is to have proportional-derivative control on the individual wheel velocities. This yields the same parameterized model but with different parameters, and is shown in Appendix A.3

Chapter 3

Control Based on the Dynamic Model

If it is desired to have the mobile robot track a trajectory, then a method for creating inputs for the robot to achieve this is necessary. To do this, a control algorithm is used. The control algorithm uses information about the robot's position and velocity and compares it to the desired position and velocity based on the defined trajectory. Using this information, along with the dynamic model of the robot, the algorithm creates inputs for the robot which should drive the robot to track the desired trajectory. Since the position and velocity of the robot are needed as feedback for the controller, odometry, which relates individual wheel rotation to robot position and orientation is used.

This chapter first discusses localization using odometry in Section 3.1. Control using inverse dynamics is discussed in Section 3.2. To improve the performance of the controller, an adaptive controller is derived in Section 3.3.

3.1 Odometry

To utilize a closed-loop controller for controlling the motion of the robot, the robot must be made aware of its current position. In continuous time, the robot can be localized with the forward and angular velocities as

$$\begin{aligned} x(t) &= \int_0^t v(\tau)\cos\left(\phi(\tau)\right)d\tau + x_0 \\ y(t) &= \int_0^t v(\tau)\sin\left(\phi(\tau)\right)d\tau + y_0 \\ \phi(t) &= \int_0^t \omega(\tau)\,d\tau + \phi_0 \end{aligned} \tag{3.1}$$

where x_0, y_0, and ϕ_0 represent the robot's pose at time $t = 0$ [10].

Since physical robots are discrete time devices with incremental encoder data, the current pose has to be estimated. Based on encoder data, the forward and angular displacements (Δs and $\Delta\phi$) are

$$\Delta s = \frac{r(\Delta\phi_r + \Delta\phi_l)}{2} \quad , \quad \Delta\phi = \frac{r(\Delta\phi_r - \Delta\phi_l)}{d} \tag{3.2}$$

where $\Delta\phi_r$ and $\Delta\phi_l$ are the changes in angular position of the right and left wheels.

Using the incremental changes in forward and angular displacements, the new position and orientation of the robot can be estimated as

$$\xi_k = \begin{bmatrix} x_k \\ y_k \\ \phi_k \end{bmatrix} = \begin{bmatrix} x_{k-1} \\ y_{k-1} \\ \phi_{k-1} \end{bmatrix} + \begin{bmatrix} \cos\bar{\phi}_k & 0 \\ \sin\bar{\phi}_k & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \Delta s \\ \Delta\phi \end{bmatrix}, \tag{3.3}$$

where ξ_k is the new position estimate at time t, ξ_{k-1} is the previous estimate at $t - kT$ (where T is the sampling period), and the angle is obtained by a second order Runge-Kutta approximation as $\bar{\phi}_k = \phi_{k-1} + \Delta\phi/2$ [11]

3.2 Inverse Dynamic Control

The dynamic model of the mobile robot can be split into 2 parts - the kinematics and the dynamics [12]. If we look at the bottom half of the dynamic model and neglect the disturbance terms,

$$\begin{bmatrix} \dot{v} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} \frac{\theta_3}{\theta_1}\omega^2 - \frac{\theta_4}{\theta_1}v \\ \\ -\frac{\theta_5}{\theta_2}v\omega - \frac{\theta_6}{\theta_2}\omega \end{bmatrix} + \begin{bmatrix} \frac{1}{\theta_1} & \\ & \\ & \frac{1}{\theta_2} \end{bmatrix} \begin{bmatrix} v_{ref} \\ \omega_{ref} \end{bmatrix}. \tag{3.4}$$

The above system relates the inputs of the robot (v_{ref} and ω_{ref}) to the outputs ($\dot{v}$ and $\dot{\omega}$). Assuming all of the parameters of the dynamic model are known, the inputs can be solved for in terms of the velocities and accelerations. This is known as an *Inverse Dynamic Controller*, as the dynamics of the robot are used to solve for the required input velocities to produce desired accelerations.

Solving for v_{ref} and ω_{ref},

$$\begin{bmatrix} v_{ref} \\ \omega_{ref} \end{bmatrix} = \begin{bmatrix} \dot{v} & 0 & -\omega^2 & v & 0 & 0 \\ 0 & \dot{\omega} & 0 & 0 & v\omega & \omega \end{bmatrix} \times \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \\ \theta_5 \\ \theta_6 \end{bmatrix} \tag{3.5}$$

or

$$\begin{bmatrix} v_{ref} \\ \omega_{ref} \end{bmatrix} = \begin{bmatrix} \theta_1 & 0 \\ 0 & \theta_2 \end{bmatrix} \begin{bmatrix} \dot{v} \\ \dot{\omega} \end{bmatrix} + \begin{bmatrix} 0 & 0 & -\omega^2 & v & 0 & 0 \\ 0 & 0 & 0 & 0 & u\omega & \omega \end{bmatrix} \times \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \\ \theta_5 \\ \theta_6 \end{bmatrix}, \tag{3.6}$$

which has the form

$$\mathbf{v_{ref}} = \mathbf{D}\dot{\mathbf{v}} + \boldsymbol{\eta} \tag{3.7}$$

with

$$\mathbf{v_{ref}} = \begin{bmatrix} v_{ref} \\ \omega_{ref} \end{bmatrix}, \mathbf{v} = \begin{bmatrix} v \\ \omega \end{bmatrix}$$

$$\boldsymbol{\eta} = \begin{bmatrix} 0 & 0 & -\omega^2 & v & 0 & 0 \\ 0 & 0 & 0 & 0 & v\omega & \omega \end{bmatrix} \times \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \\ \theta_5 \\ \theta_6 \end{bmatrix}, \mathbf{D} = \begin{bmatrix} \theta_1 & 0 \\ 0 & \theta_2 \end{bmatrix}$$

If the parameters are perfectly known, equation (3.7) can be used to calculate the required input $\mathbf{v_{ref}}$ to generate the desired output $\mathbf{v}$. Since we would also like to compensate for any positional errors in order to track a desired trajectory, we then

consider a control law of the form [12]

$$\begin{bmatrix} v_{ref} \\ \omega_{ref} \end{bmatrix} = \begin{bmatrix} \theta_1 & 0 \\ 0 & \theta_2 \end{bmatrix} \begin{bmatrix} \sigma_1 \\ \sigma_2 \end{bmatrix} + \begin{bmatrix} 0 & 0 & -\omega^2 & v & 0 & 0 \\ 0 & 0 & 0 & 0 & v\omega & \omega \end{bmatrix} \times \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \\ \theta_5 \\ \theta_6 \end{bmatrix} \tag{3.8}$$

or

$$\mathbf{v_{ref}} = \mathbf{D}\boldsymbol{\sigma} + \boldsymbol{\eta}, \tag{3.9}$$

where

$$\sigma_1 = \dot{v}^k_{ref} + k_v\tilde{v} \tag{3.10}$$

$$\sigma_2 = \dot{\omega}^k_{ref} + k_\omega\tilde{\omega} \tag{3.11}$$

with k_v and k_ω positive constants, and $\tilde{v} = v^k_{ref} - v$ and $\tilde{\omega} = \omega^k_{ref} - \omega$, where v^k_{ref} and ω^k_{ref} are the forward and angular velocities generated by a kinematic controller which tracks the point h.

If we define G as

$$G = \begin{bmatrix} \sigma_1 & 0 & -\omega^2 & v & 0 & 0 \\ 0 & \sigma_2 & 0 & 0 & v\omega & \omega \end{bmatrix}, \tag{3.12}$$

then equation (3.9) can be written as

$$\mathbf{v_{ref}} = \mathbf{G}\boldsymbol{\theta} \tag{3.13}$$

This inverse dynamic controller assumes that the vector of parameters $\boldsymbol{\theta}$ is perfectly known, which is rarely the case. To remedy this situation, an adaptive controller can be used. The adaptive controller is given an initial estimate of the parameters, and then updates each parameter based on the error between what was produced and what was expected.

3.3 Adaptation of Dynamic Parameters

If the parameters are not perfectly known, then their estimates, represented by the vector $\hat{\boldsymbol{\theta}}$, should be used in place of $\boldsymbol{\theta}$. Then the proposed controller has the form

$$\begin{aligned} \mathbf{v_{ref}} &= \mathbf{G}\hat{\boldsymbol{\theta}} \\ &= \mathbf{G}\boldsymbol{\theta} + \mathbf{G}\tilde{\boldsymbol{\theta}} \\ &= \mathbf{D}\boldsymbol{\sigma} + \boldsymbol{\eta} + \mathbf{G}\tilde{\boldsymbol{\theta}} \end{aligned} \tag{3.14}$$

where $\tilde{\boldsymbol{\theta}} = \hat{\boldsymbol{\theta}} - \boldsymbol{\theta}$

If we set equation (3.7) equal to equation (3.14) then we have

$$\mathbf{D}\dot{\mathbf{v}} + \boldsymbol{\eta} = \mathbf{D}\boldsymbol{\sigma} + \boldsymbol{\eta} + \mathbf{G}\tilde{\boldsymbol{\theta}} \tag{3.15}$$

Cancelling the $\boldsymbol{\eta}$ term on both sides and rearranging,

$$\mathbf{D}(\boldsymbol{\sigma} - \dot{\mathbf{v}}) = -\mathbf{G}\tilde{\boldsymbol{\theta}} \tag{3.16}$$

Since $\boldsymbol{\sigma} = \dot{\mathbf{v}}^{\mathbf{k}}_{\mathbf{ref}} + \mathbf{K}\tilde{\mathbf{v}}$, then $\boldsymbol{\sigma} - \dot{\mathbf{v}} = \dot{\tilde{\mathbf{v}}} + \mathbf{K}\tilde{\mathbf{v}}$ with $\tilde{\mathbf{v}} = \mathbf{v}^{\mathbf{k}}_{\mathbf{ref}} - \mathbf{v}$ and $\mathbf{K} = \begin{bmatrix} k_v & 0 \\ 0 & k_\omega \end{bmatrix}$, then equation (3.16) becomes

$$\mathbf{D}(\dot{\tilde{\mathbf{v}}} + \mathbf{K}\tilde{\mathbf{v}}) = -\mathbf{G}\tilde{\boldsymbol{\theta}} \tag{3.17}$$

The error equation can now be written as $\dot{\tilde{\mathbf{v}}}$,

$$\dot{\tilde{\mathbf{v}}} = -\mathbf{D}^{-\mathbf{1}}\mathbf{G}\tilde{\boldsymbol{\theta}} - \mathbf{K}\tilde{\mathbf{v}} \tag{3.18}$$

A Lyapunov function is then chosen as

$$V = \frac{1}{2}\tilde{\mathbf{v}}^\top \mathbf{D}\tilde{\mathbf{v}} + \frac{1}{2}\tilde{\boldsymbol{\theta}}^\top \boldsymbol{\gamma}^{-\mathbf{1}}\tilde{\boldsymbol{\theta}}, \tag{3.19}$$

where $\boldsymbol{\gamma}$ is a positive-definite 6×6 matrix, and $\mathbf{D}$ is positive definite due to the physical properties of the robot. Differentiating V with respect to time, and noticing that $\dot{\tilde{\boldsymbol{\theta}}} = \dot{\hat{\boldsymbol{\theta}}}$ since $\boldsymbol{\theta}$ is a constant,

$$\dot{V} = -\tilde{\mathbf{v}}^\top \mathbf{D}\mathbf{K}\tilde{\mathbf{v}} - \tilde{\mathbf{v}}^\top \mathbf{G}\tilde{\boldsymbol{\theta}} + \tilde{\boldsymbol{\theta}}^\top \boldsymbol{\gamma}^{-\mathbf{1}}\dot{\hat{\boldsymbol{\theta}}}. \tag{3.20}$$

Thus, if the parameter-update law is chosen to be [12]

$$\dot{\hat{\boldsymbol{\theta}}} = \boldsymbol{\gamma}\mathbf{G}^\top \tilde{\mathbf{v}}, \tag{3.21}$$

then

$$\dot{V} = -\tilde{\mathbf{v}}^\top \mathbf{D}\mathbf{K}\tilde{\mathbf{v}}, \tag{3.22}$$

and $\dot{V} \leq 0$, which means that $\tilde{\mathbf{v}}, \tilde{\boldsymbol{\theta}} \in L_\infty$. Equation (3.22) can be integrated to obtain

$$V(T) + \int_0^T \tilde{\mathbf{v}}^\top \mathbf{D}\mathbf{K}\tilde{\mathbf{v}}\, dt \leq V_0, \tag{3.23}$$

then

$$V(T) + \lambda_{min}(\mathbf{DK}) \int_0^T \tilde{\mathbf{v}}^\top \tilde{\mathbf{v}}\, dt \leq V_0, \tag{3.24}$$

which implies that $\tilde{v} \in L_2$. Using the fact that $\tilde{\theta}, \tilde{v} \in L_\infty$ we then have from equation (3.18) that $\dot{\tilde{v}} \in L_\infty$. Since $\tilde{v} \in L_2 \cap L_\infty$ with $\dot{\tilde{v}} \in L_\infty$, then an application of Barbalat's Lemma gives $\tilde{v} \to 0$ as $t \to \infty$.

Chapter 4

Parameter Estimation

4.1 Proposed method

Although in simulation any initial parameter estimate is sufficient for convergence to the true value, in practice, the initial estimate must be close to the true value. Thus, before the adaptive controller can be implemented, an acceptable estimate of the parameters must be produced. Of the methods that were developed, all are effective in simulation, but only one is effective in practice. These methods are outlined below.

Parameter projection

The parameter projection method utilizes prior knowledge of physical constants of the robot in question to develop bounds on the parameter estimates. To create the bounds, the terms that make up each parameter should be considered. For instance, for $\hat{\theta}_5$, if it is known that the mass m of the robot is somewhere between 8 and $9kg$, then this knowledge can be used to develop a range for $\hat{\theta}_5$. However, all of the terms that comprise each parameter must be estimated in order to create such a

range. If a term is known with complete certainty (for example, we may know the proportional and derivative gains of the low-level controller), then these values can be treated as constants when creating the parameter ranges. Once the bounds are acquired, the initial estimate can be obtained by taking the average of the upper and lower bound. The values of the bounds are then used in the parameter update law to restrict a parameter from leaving the bounds. For example, if it is known that θ_3 is negative, but the value of $\hat{\theta}_3$ is zero and $\dot{\hat{\theta}}_3$ is positive (such that $\hat{\theta}_3$ will become positive), then the parameter update law for that term is disabled. In the simulation environment, where the parameters of the dynamic model of the robot are perfectly known, it is easy to produce these bounds. In practice, however, it is not so straightforward. The designer of the controller may not have access to the knowledge of some of the physical constants. For instance, the proportional-derivative gains of the motor controllers may be something not published by the manufacturer. The properties of the motors themselves, i.e. rotor inertia, the voltage constant, and the torque constant may not be available from the motor manufacturer (or worse, the motor manufacturer may be unknown). This is the main drawback of the parameter projection method - we must have a reasonable estimate of all of the terms in each parameter. If the initial estimate is not close to the true value, the estimate may not converge. Alternatively, the bounds produced may not enclose the true value.

Linear regression

The main drawback of the parameter projection method is that we need to have reasonable estimates for each term of each parameter. Ideally, we would like a method for generating initial parameter estimates that requires no a priori knowledge of the terms comprising the parameters. Such a method would be convenient in that it would be easily transferable between robots of different dynamics, and relieves

any concerns of finding estimates for each term of the parameters. If we consider the parameterized dynamic equation of the robot,

$$\begin{bmatrix} v_{ref} \\ \omega_{ref} \end{bmatrix} = \begin{bmatrix} \dot{v} & 0 & -\omega^2 & v & 0 & 0 \\ 0 & \dot{\omega} & 0 & 0 & v\omega & \omega \end{bmatrix} \times \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \\ \theta_5 \\ \theta_6 \end{bmatrix}, \tag{4.1}$$

the parameters can be obtained using the knowledge of v_{ref}, ω_{ref}, and the regressor matrix. In a noise-free situation, the parameters can be solved for exactly. In reality, a linear regression can be used to produce the best estimate based on the inputs provided and the outputs measured. On most commercially available robots, the accelerations $\dot{v}$ and $\dot{\omega}$ are not available for measurement, and therefore must be estimated using the velocity data. Once the accelerations are estimated, the regressor matrix can be constructed, and a linear regression can be applied. In both simulation and experimental results, this has yielded acceptable values for the initial parameter estimates. The proposed method is as follows;

1. Excite the robot's v_{ref} and ω_{ref} inputs with sufficiently rich signals.
2. Record the robot's v and ω outputs.
3. After all of the data is collected, filter the recorded velocities.
4. Differentiate the filtered velocities to produce estimates of $\dot{v}$ and $\dot{\omega}$.
5. Construct the regressor matrix from the filtered and estimated values.
6. Perform a linear regression to obtain the estimates of the parameters.

7. Using the estimates obtained from the linear regression, the inverse dynamic controller can now be used, along with the adaptive controller to update the parameter estimates if needed.

To perform the linear regression, equation (4.1) is broken into two equations;

$$v_{ref} = \begin{bmatrix} \dot{v} & -\omega^2 & v \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_3 \\ \theta_4 \end{bmatrix} \quad , \quad \omega_{ref} = \begin{bmatrix} \dot{\omega} & v\omega & \omega \end{bmatrix} \begin{bmatrix} \theta_2 \\ \theta_5 \\ \theta_6 \end{bmatrix} \tag{4.2}$$

For the case of the forward velocity, if data is then recorded with n observations, we have

$$\begin{bmatrix} v_{ref_1} \\ v_{ref_2} \\ \vdots \\ v_{ref_n} \end{bmatrix} = \begin{bmatrix} \dot{v}_1 & -\omega_1^2 & v_1 \\ \dot{v}_2 & -\omega_2^2 & v_2 \\ \vdots & \vdots & \vdots \\ \dot{v}_n & -\omega_n^2 & v_n \end{bmatrix} \begin{bmatrix} \hat{\theta}_1 \\ \hat{\theta}_3 \\ \hat{\theta}_4 \end{bmatrix} \tag{4.3}$$

which can be written as

$$\mathbf{Y} = \mathbf{X}\mathbf{b} \tag{4.4}$$

where $\mathbf{Y}$ is the predictor vector of length n containing the inputs v_{ref}, $\mathbf{X}$ is the n by 3 regressor matrix, and $\mathbf{b}$ is the vector of the three parameter estimates. If the matrix $\mathbf{X}$ is of rank 3, then we can solve for the parameter vector $\mathbf{b}$ by left-multiplying both sides by the pseudoinverse of $\mathbf{X}$ [13]. Then, we have

$$\mathbf{b} = (\mathbf{X}^\top \mathbf{X})^{-\mathbf{1}} \mathbf{X}^\top \mathbf{Y} \tag{4.5}$$

or

$$\begin{bmatrix} \hat{\theta}_1 \\ \hat{\theta}_3 \\ \hat{\theta}_4 \end{bmatrix} = \left\{ \begin{bmatrix} \dot{v}_1 & \dot{v}_2 & \dots & \dot{v}_n \\ -\omega_1^2 & -\omega_2^2 & \dots & -\omega_n^2 \\ v_1 & v_2 & \dots & v_n \end{bmatrix} \begin{bmatrix} \dot{v}_1 & -\omega_1^2 & v_1 \\ \dot{v}_2 & -\omega_2^2 & v_2 \\ \vdots & \vdots & \vdots \\ \dot{v}_n & -\omega_n^2 & v_n \end{bmatrix} \right\}^{-1} \times \begin{bmatrix} \dot{v}_1 & \dot{v}_2 & \dots & \dot{v}_n \\ -\omega_1^2 & -\omega_2^2 & \dots & -\omega_n^2 \\ v_1 & v_2 & \dots & v_n \end{bmatrix} \begin{bmatrix} v_{ref_1} \\ v_{ref_2} \\ \vdots \\ v_{ref_n} \end{bmatrix} \tag{4.6}$$

Similarly, for ω_{ref},

$$\begin{bmatrix} \hat{\theta}_2 \\ \hat{\theta}_5 \\ \hat{\theta}_6 \end{bmatrix} = \left\{ \begin{bmatrix} \dot{\omega}_1 & \dot{\omega}_2 & \dots & \dot{\omega}_n \\ v_1\omega_1 & v_2\omega_2 & \dots & v_n\omega_n \\ \omega_1 & \omega_2 & \dots & \omega_n \end{bmatrix} \begin{bmatrix} \dot{\omega}_1 & v_1\omega_1 & \omega_1 \\ \dot{\omega}_2 & v_1\omega_2 & \omega_2 \\ \vdots & \vdots & \vdots \\ \dot{\omega}_n & v_1\omega_n & \omega_n \end{bmatrix} \right\}^{-1} \times \begin{bmatrix} \dot{\omega}_1 & \dot{\omega}_2 & \dots & \dot{\omega}_n \\ v_1\omega_1 & v_2\omega_2 & \dots & v_n\omega_n \\ \omega_1 & \omega_2 & \dots & \omega_n \end{bmatrix} \begin{bmatrix} \omega_{ref_1} \\ \omega_{ref_2} \\ \vdots \\ \omega_{ref_n} \end{bmatrix} \tag{4.7}$$

This method is appealing in that it requires no knowledge of the terms that comprise the parameters. It can therefore easily be applied to any robot that shares the same parameterization, even if the parameters themselves have a different structure. For instance, any differential drive robot will have the parameterization used here. However, a differential drive robot that has a low-level proportional-derivate controller

on forward and angular velocities will have parameters of a different form than one that has proportional-derivate control on individual wheel velocities. Regardless of this difference, since both robots share the same form of the parameterized model, this method will work for both robots. This method can also be completely automated such that all the programmer needs to do is load the control algorithm onto the robot. The robot can then perform the above steps on-board to obtain the parameter estimates autonomously.

In a variation on the above approach, two of the parameters can first easily be obtained. Observing equation (4.1), if the robot has only constant forward velocity and a constant input v_{ref}, then we have a simple case where $v_{ref} = \hat{\theta}_4 v$. Similarly for purely constant angular velocity, $\omega_{ref} = \hat{\theta}_6 \omega$. Intuitively, these values should be close to unity, as we expect the robot's low-level controller to achieve and maintain the reference inputs. The robot can therefore be subjected to a constant forward velocity, and after a steady-state velocity is reached, dividing v_{ref} by v will yield an estimate for $\hat{\theta}_4$. Doing the same for the angular velocity thereby produces two out of the six unknown parameters.

4.2 Simulation Results

Computer simulations were performed using *Matlab*. The dynamic model of the robot and the parameter update law were modeled as an 11 state system of first order differential equations, where the first 5 states were the robot dynamics, and the last 6 states were the parameter estimates of the robot. The kinematic control law used to generate the v_{ref}^k and ω_{ref}^k inputs to the dynamic model is taken from [11] as

$$\begin{aligned} v_{ref}^k &= v_d \cos\tilde{\varphi} + k_1(\tilde{x}\cos\varphi + \tilde{y}\sin\varphi) \\ \omega_{ref}^k &= \omega_d + k_2 v_d \frac{\sin\tilde{\varphi}}{\tilde{\varphi}}(\tilde{y}\cos\varphi - \tilde{x}\sin\varphi) + k_3\tilde{\varphi} \end{aligned} \tag{4.8}$$

where

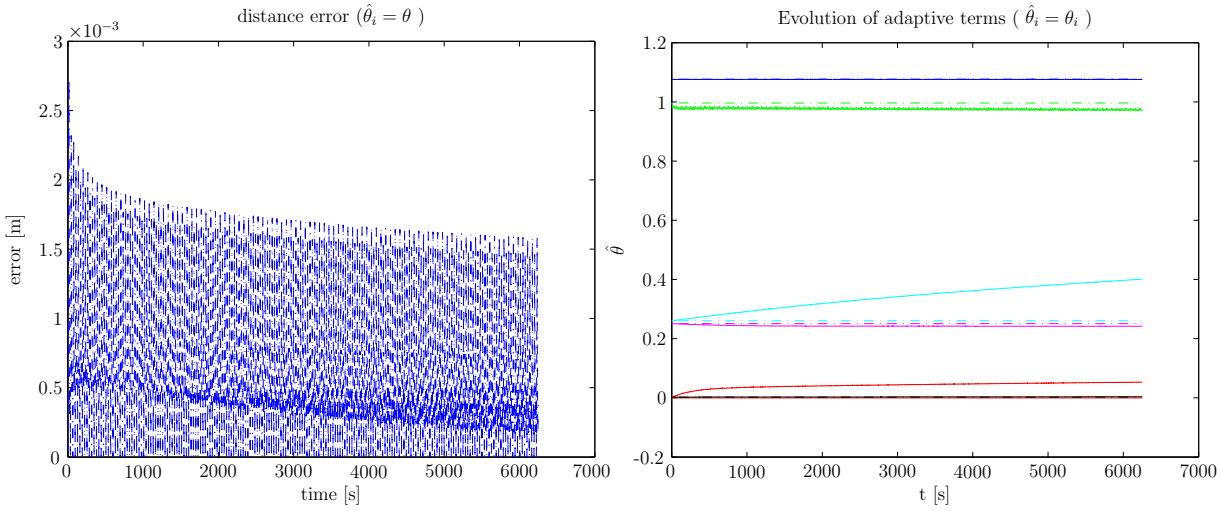
$$\begin{aligned} \tilde{x} &= x_d - x \\ \tilde{y} &= y_d - y \\ \tilde{\varphi} &= \varphi_d - \varphi \end{aligned}$$

Using this kinematic control law, the point h being tracked in figure 2.4 is on the axle, and therefore $h = 0$. The purpose of the kinematic control law above is to generate reference velocities to keep the robot on a desired trajectory. The trajectory is defined by $x_d(t)$, $y_d(t)$, and $\varphi_d(t)$. The trajectory can be composed of any continuously differentiable functions defining $x_d(t)$ and $y_d(t)$. $\varphi_d(t)$ is defined by the angle formed by the velocity resulting from $x_d(t)$ and $y_d(t)$. In all simulations presented here, the robot is following a figure-eight style trajectory as defined in Appendix A.

Simulations were performed to analyze the performance of the adaptive controller in different scenarios. First, a simulation was run using perfect parameter estimates and perfect initial conditions for position, but the robot having an initial velocity of zero, which is not the initial desired velocity. Although the parameter estimates begin at their true values, the values changed because of the error in initial velocity.

$\hat{\theta}_1$, $\hat{\theta}_2$, $\hat{\theta}_3$, $\hat{\theta}_4$, $\hat{\theta}_5$, $\hat{\theta}_6$

Figure 4.1: Legend for parameter evolution plots



(a) Distance error

(b) Parameter evolution

Figure 4.2: Perfect initial estimates

Figure 4.2(a) shows the error between the actual position and desired position along the trajectory versus time for 50 iterations of the figure-eight trajectory. As time increases the error decreases. In Figure 4.2(b), it can be observed that even though the initial parameter estimates are perfect, they do not remain constant due to the imperfect initial condition on velocity.

Effect of control gains on parameter convergence

In simulation, any positive control gain is suitable. Given a proper excitation, a large control gain will drive the parameter estimate to the true value faster than a smaller gain. This is demonstrated in Figure 4.3. Looking at Figure 4.3, we

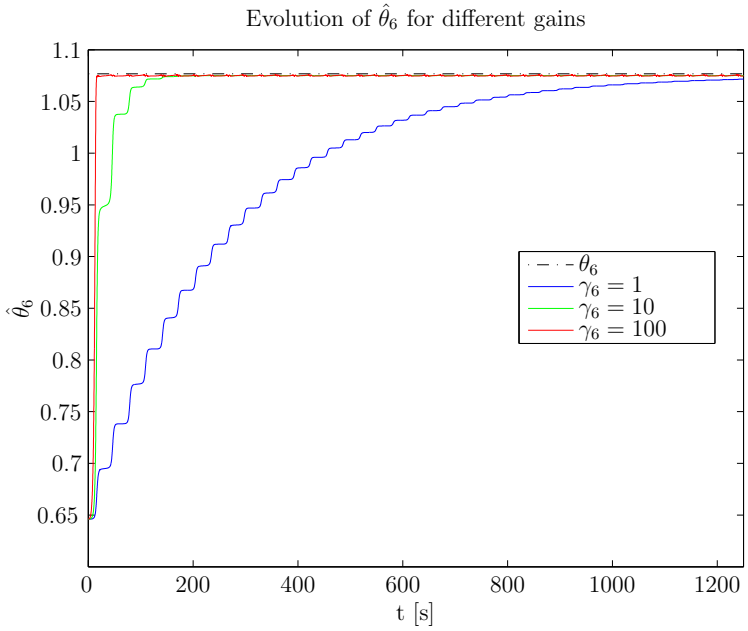


Figure 4.3: Effect of control gain on parameter convergence

can see that that $\hat{\theta}_6$ updates periodically. This is due to the input provided to the robot. Observing that $\dot{\hat{\theta}}_6 = \gamma_6\omega(\omega^k_{ref} - \omega)$ shows that $\hat{\theta}_6$ will not change under two conditions; If the angular velocity of the robot, ω is zero, or if the commanded angular velocity ω^k_{ref} matches ω.

Figure 4.4 shows that $\hat{\theta}_6$ changes the least when ω is close to zero. It also shows that as time progresses, for a given angular velocity the change in $\hat{\theta}_6$ decreases, suggesting that the controller is becoming more accurate in producing an angular velocity ω close to ω^k_{ref}.

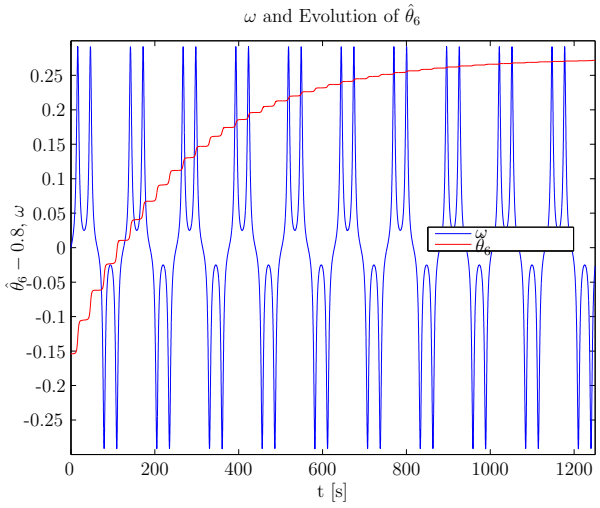


Figure 4.4: Angular velocity and $\hat{\theta}_6$

Effect of input signal on parameter convergence

From the results in Figure 4.4, it is obvious that the input to the system has a strong impact on the convergence of parameters. Here, the robot is again following the figure-eight trajectory. For a given set of initial estimates, the parameters may converge to different values in the case of different input signals. Below are examples of two simulations with identical initial parameters and perfect initial conditions for the position and velocity of the robot.

For the case of a simple figure-eight style trajectory, all terms except $\hat{\theta}_1$ and $\hat{\theta}_5$ approach their true values.

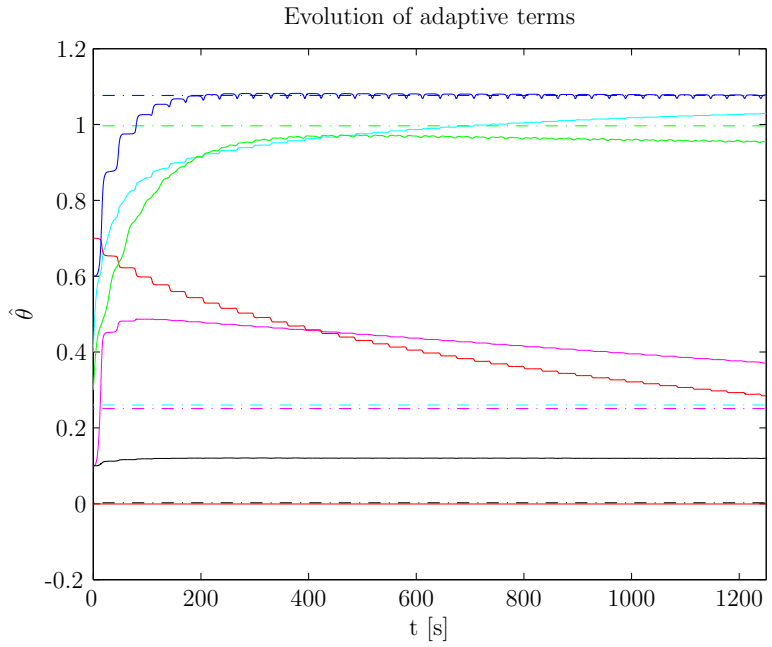


Figure 4.5: Parameter evolution for figure-eight trajectory

$\hat{\theta}$	initial	final	true
1	0.4000	1.0291	0.2604
2	0.1000	0.3711	0.2509
3	0.7000	0.2847	-0.0004
4	0.3000	0.9539	0.9965
5	0.1000	0.1198	0.0026
6	0.6000	1.0779	1.0768

Table 4.1: Parameter final and initial values

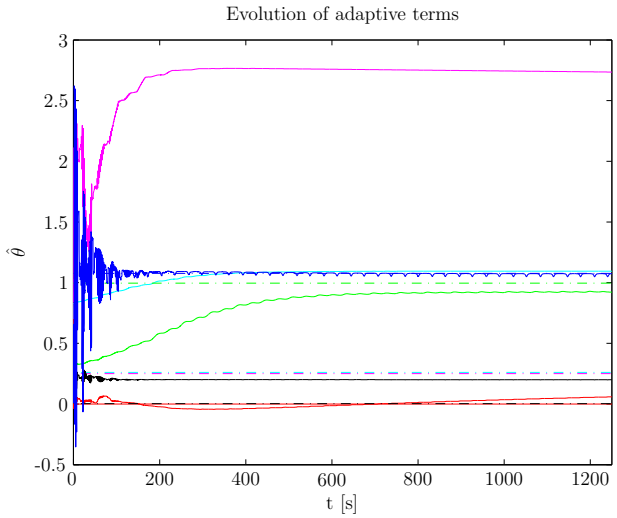


Figure 4.6: Parameter evolution for figure-eight trajectory with dither

For the case of a figure-eight style trajectory with a dither signal, all terms except $\hat{\theta}_2$ approach their true values.

$\hat{\theta}$	initial	final	true
1	0.4000	1.0939	0.2604
2	0.1000	2.7351	0.2509
3	0.7000	0.0595	-0.0004
4	0.3000	0.9216	0.9965
5	0.1000	0.2005	0.0026
6	0.6000	1.0739	1.0768

Table 4.2: Parameter final and initial values

Effect of other parameters on parameter convergence

The convergence of a particular parameter can depend on the values of other parameters. This is demonstrated in the two simulations below.

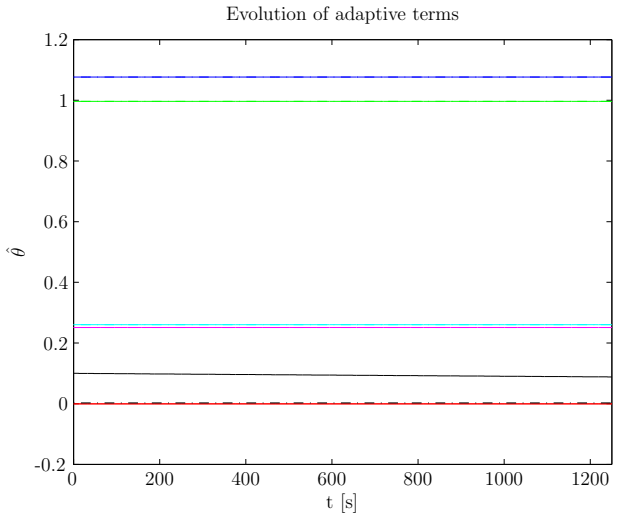
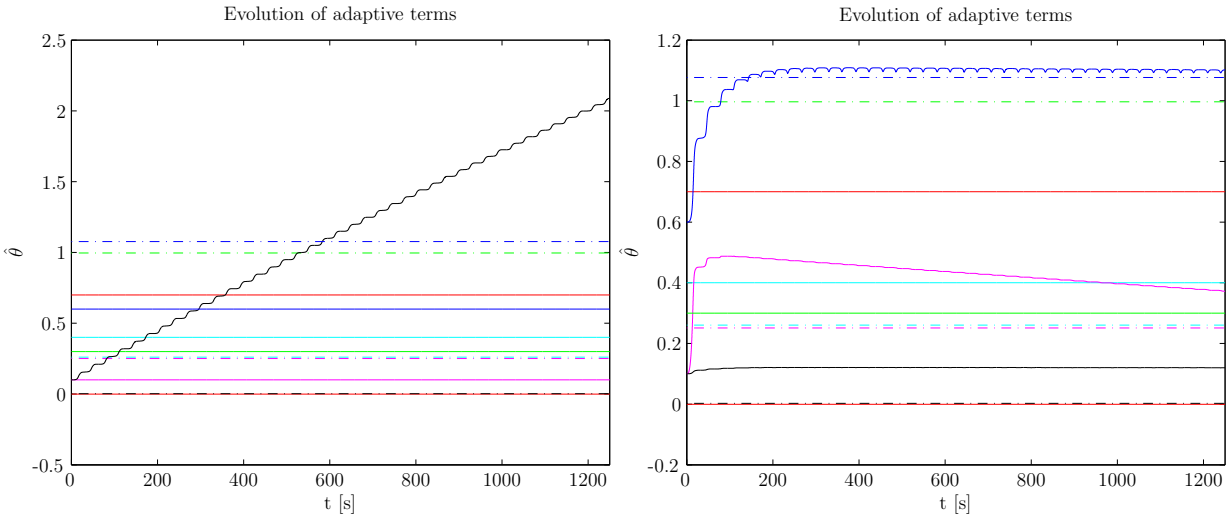


Figure 4.7: Evolution of $\hat{\theta}_5$

In Figure 4.7, all parameter estimates are perfect, except for $\hat{\theta}_5$ (represented by the solid black line). The only nonzero adaptive gain is $\gamma_5 = 10$, such that only $\hat{\theta}_5$ is being updated. From the figure, we can see that $\hat{\theta}_5$ is slowly approaching its true value. Figure 4.8(a) shows the evolution of $\hat{\theta}_5$ for non-perfect parameter estimates, with only $\hat{\theta}_5$ being updated. In this case, $\hat{\theta}_5$ moves away from its true value.

Considering that $\dot{\hat{\theta}}_5 = \gamma_5(\omega_{ref}^k - \omega)$, we know that $\dot{\hat{\theta}}_5$ depends on ω. If we look at the dynamic model of the robot, it can be seen that the angular velocity ω of the robot contains the parameters θ_2, θ_5, and θ_6. Thus if the estimates for θ_2 and θ_6 are inaccurate, and we try to adapt $\hat{\theta}_5$, the value for $\hat{\theta}_5$ may converge to some other value to compensate for the error in angular velocity being produced due to the inaccuracies in $\dot{\hat{\theta}}_2$ and $\dot{\hat{\theta}}_6$



(a) updating $\hat{\theta}_5$ only

(b) updating $\hat{\theta}_2$, $\hat{\theta}_5$, and $\hat{\theta}_6$

Figure 4.8: Parameter evolution

Under the same initial conditions and input signals, if all terms relating to angular velocity are adapted ($\hat{\theta}_2$, $\hat{\theta}_5$, and $\hat{\theta}_6$), then the parameters approach their true values, as shown in Figure 4.8(b).

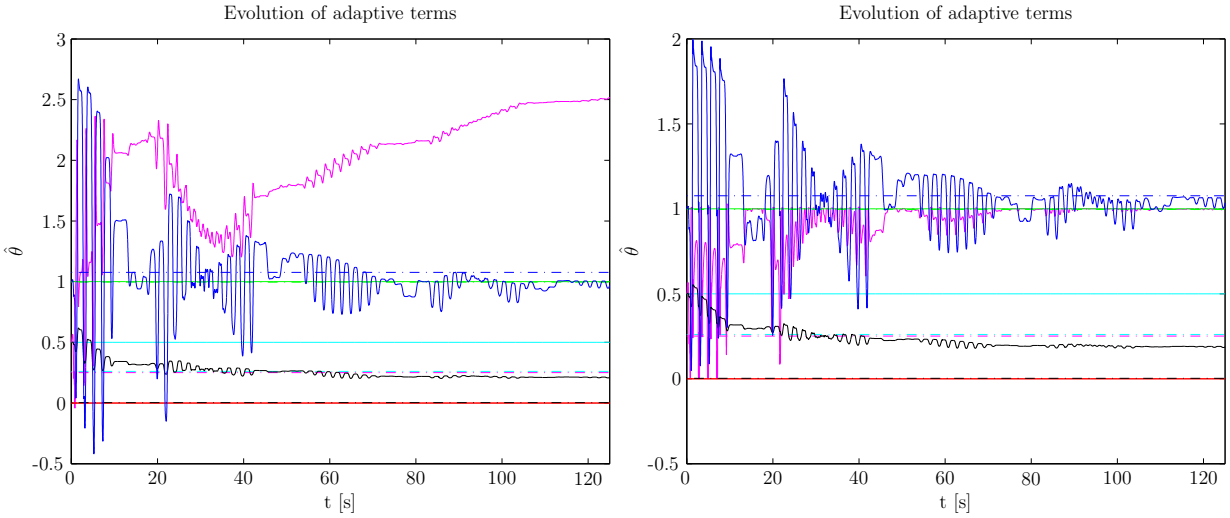
Parameter projection

Since the parameters of the dynamic model represent combinations of physical constants of the robot, most of which are known or can be estimated with some upper and lower bound, bounds on the parameter estimates can be made. Using these upper and lower bounds, the adaptive law can be modified to constrain each parameter estimate to remain within the predefined bounds [14].

If the following parameter update considered,

$$\dot{\hat{\theta}}_i = \begin{cases} 0 & \text{if} \quad \hat{\theta}_i \geq \hat{\theta}_{i_{max}} \quad \text{and} \quad \gamma_i \mathbf{g_i}^\top \tilde{\mathbf{v}} > 0 \\ \gamma_i \mathbf{g_i}^\top \tilde{\mathbf{v}} & \text{if} \quad \hat{\theta}_{i_{min}} < \hat{\theta}_i < \hat{\theta}_{i_{max}} \\ 0 & \text{if} \quad \hat{\theta}_i \leq \hat{\theta}_{i_{min}} \quad \text{and} \quad \gamma_i \mathbf{g_i}^\top \tilde{\mathbf{v}} < 0 \end{cases} \tag{4.9}$$

then the adaptive law is only active when each parameter is within its bounds.



(a) updating $\hat{\theta}_5$ only

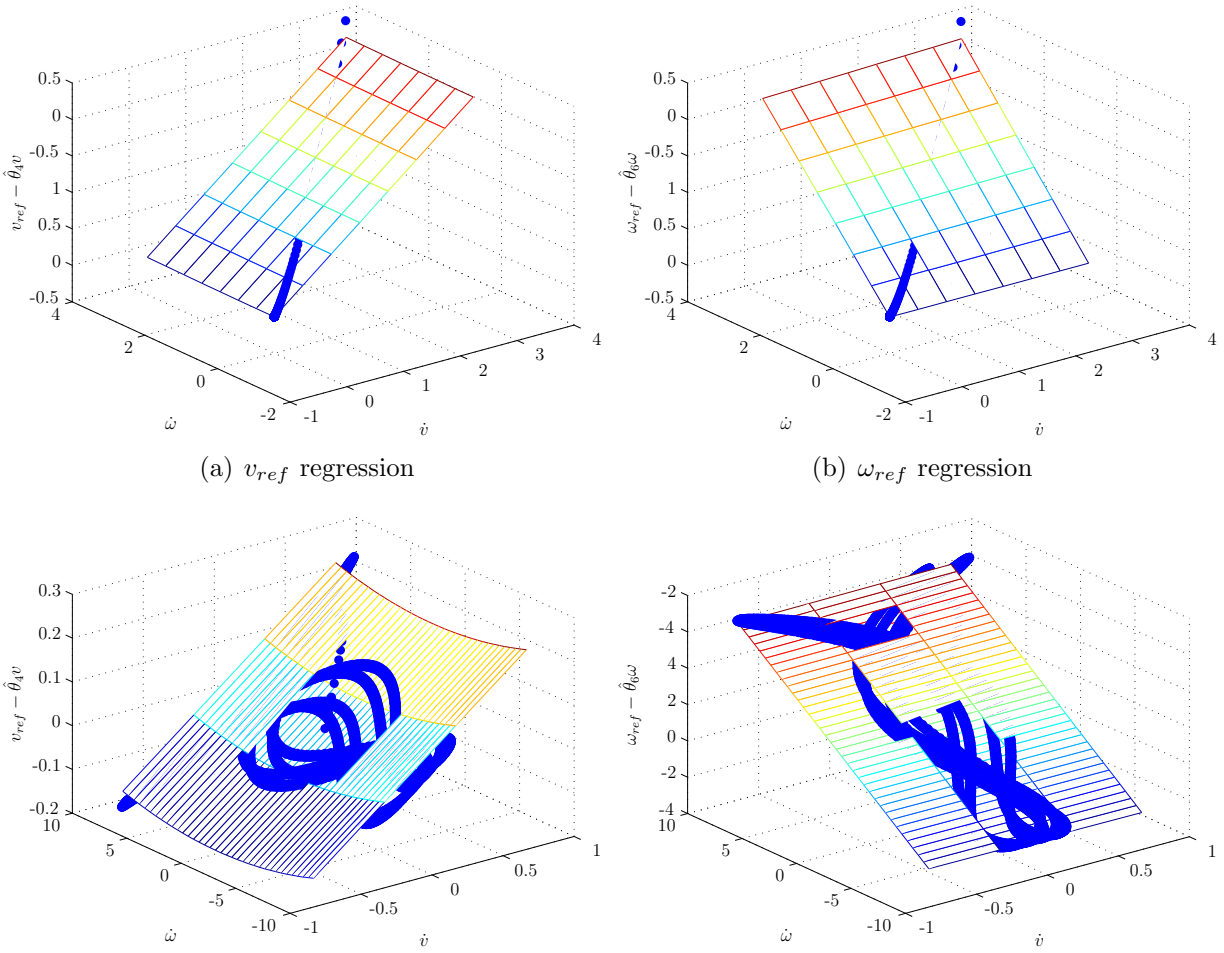
(b) updating $\hat{\theta}_2$, $\hat{\theta}_5$, and $\hat{\theta}_6$

Figure 4.9: Parameter evolution

The simulation results for the linear regression method proposed in section 4.1 are shown in Figure 4.10. For the first case (Figures 4.10(a) and 4.10(b)), the system is given an input signal of $v_{ref} = 1$ and $\omega_{ref} = 1$. For the second case, the system was given inputs of

$$\begin{aligned} v_{ref} &= 0.2\sin(t) + 0.1\sin(1.5t) + 0.1\sin(3t) + 0.1\sin(0.1t)\ldots \\ &\quad + 0.08\sin(0.013t) + 0.1\sin(5t) \\ \omega_{ref} &= (5\pi/3)\sin(2t) + 0.1\sin(0.1t) + 0.1\sin(0.09t) \end{aligned}$$

Linear regression



(a) v_{ref} regression

(b) ω_{ref} regression

(c) v_{ref} regression

(d) ω_{ref} regression

Figure 4.10: Regression

It was previously shown that the inputs provided to the dynamic model had a strong impact on parameter convergence. In figure 4.10, this is shown for the case of linear regression. It is clear that for the case where a more complex input signal is provided (figures 4.10(c) and 4.10(d)), it is easier to define the surface that is created by the datapoints. For this reason, when collecting data for a linear regression, a signal that provides sufficient excitation should be chosen. In general, the signal should contain at least half as many distinct frequencies as there are unknown parameters [14].

4.3 Experimental Setup

The proposed controller is tested on a Pioneer 3-DX and a Khepera III (figure 4.11). The Pioneer 3-DX is a larger differential drive mobile robot, having a mass of 9 kilograms. It has a maximum forward velocity of 1.4 meters per second, and a track width of 34 centimeters. The Khepera III is a much smaller robot, with a mass of 742 grams. It has a maximum forward velocity of 500 millimeters per second, and a track width of 89 millimeters. Because of the difference in size between these robots, one can expect different dynamics between the two. Both robots take a forward and angular reference velocity as inputs. It is worth noting that the Pioneer achieves the reference velocities via proportional-derivative control on the forward and angular velocities of the robot, whereas the Khepera III uses proportional-derivative control on individual wheel velocities.

The Pioneer 3-DX has an onboard computer with a 1.8 GHz processor, 512 megabytes of RAM and uses Debian Linux as an operating system. The Khepera III utilizes a Gumstix Verdex computer-on-module operating at 600 MHz, with 128 megabytes of RAM, and running OpenEmbedded Angstrom Linux as the operating system.

Since these two robots are made by different manufacturers, they do not use the same libraries to interface the software with the hardware. For instance, the command to specify a forward velocity on the Pioneer 3-DX is not the same command used on the Khepera III. A piece of software that addresses this issue is *Player* [15]. Player is a "cross-platform robot device interface" which provides a uniform interface from robot to robot (provided that the robot is supported). Player is essentially an additional layer of software on top of the basic libraries used to interface with the robot. It is a very useful piece of software if the code being written is planned to be used on multiple different platforms. Player also has the capability of being

executed remotely. For instance, the program can be run on a base station, while commanding motors and reading sensors by wireless internet. In the experiments performed here, the program is run on the physical robot to eliminate any time delays that may be associated with wireless communication. There is a disadvantage of using player, namely the added computation time required. On the Pioneer 3-DX, this is not much of an issue since it has a relatively fast processor. On the Khepera III however, the sample time of the algorithm being implemented using Player is about 0.8 second (compared to 0.3 second on the Pioneer 3-DX). For this reason, using Player on the Khepera III is not a viable option. Instead, the Khepera III toolbox is used. The Khepera III toolbox provides libraries, modules, and scripts for interfacing with the Khepera III. Using the Khepera III toolbox decreased the sample time substantially to about 0.02 second. This does take away the convenience of being able to run the same code on both robots, as was the case with Player. Since the main advantage of using Player is now not applicable, there is no reason to suffer the additional computational time in using it on the Pioneer. The pioneer is therefore programmed in its native API, *Aria*. Information on programming in Aria can be found in [16].

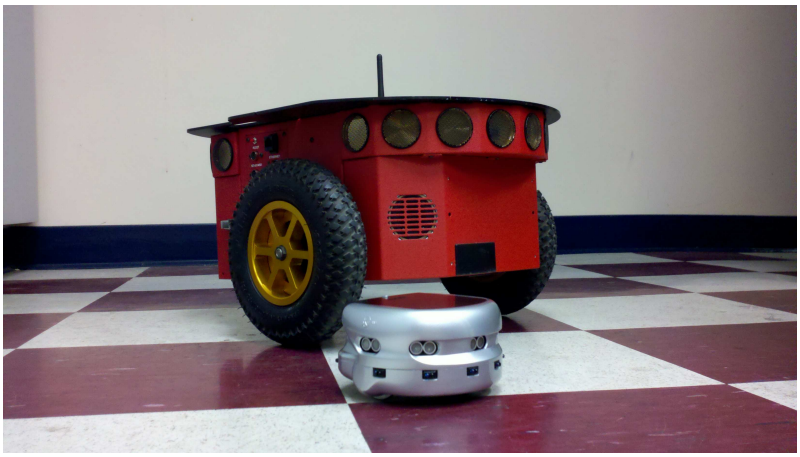


Figure 4.11: Khepera III and Pioneer 3-DX mobile robots

In the experiments performed in the subsequent sections, all localization is based off of each robot's onboard odometry. During each iteration of the program, the time elapsed from the start of the program is calculated. Based on the trajectory programmed, the robot calculates a desired position, orientation, and velocity. This data is recorded in a file *desired*. Each iteration, the robot also polls the encoders, and calculates an estimate of its current position, which is stored in *actual*. From the files *desired* and *actual*, plots of the desired and actual trajectories can be generated. Considering that the files store the x position in the first column, the y position in the second column, and the orientation ϕ in the third column, a plot of the trajectories can be generated in *Matlab* by executing the following commands;

```
load actual;
load desired;
plot(desired(:,1),desired(:,2),'-b');
hold on
plot(actual(:,1),actual(:,2),'-r');
```

The distance error is calculated by the following equation;

$$\text{error} = \sqrt{(x_d - x)^2 + (y_d - y)^2}$$

The distance error is the main performance metric in the experiments performed, and is what is trying to be minimized.

4.4 Experimental Results

To demonstrate the necessity of generating reasonable initial parameter estimates and the effectiveness of the proposed method for doing so, various experiments were performed. The experiments were performed on two different robots, a Khepera III and a Pioneer 3-DX. Both of these robots are similar in that they are differentially driven, but differ in size. The difference in scale of the robots should add confidence that the proposed method is effective regardless of the actual parameters being estimated, and requires no modification to perform on different systems. To show the effect of varying the dynamics of the robot, each robot is subject to a payload. The adaptive control law will adjust the parameter estimates to compensate for the change in dynamics introduced by the payload.

As discussed in Section 4.1, the values of $\hat{\theta}_4$ and $\hat{\theta}_6$ are expected to be close to 1. It will be shown that both for the Khepera III and Pioneer 3-DX, the initial estimates for these two parameters are indeed close to 1. If we observe the terms that comprise each parameter (see Section 2.2), it should be noticed that the only parameters that would be effected by an additional payload are θ_1, θ_2, θ_3, and θ_5, since these are the only parameters that contain mass, inertia, and the location of the center of gravity. Therefore, $\hat{\theta}_4$ and $\hat{\theta}_6$ can be held as constant during adaptation.

To further verify this assumption, each robot was subjected to various forward and angular velocities, both with and without a payload. Comparing the steady-state value with the reference value shows that the assumption is indeed valid.

(a) without payload

v_{ref}	$v_{steadystate}$
0.1	0.1000
0.2	0.1999
0.3	0.3001

ω_{ref}	$\omega_{steadystate}$
2.0	1.9987
4.0	4.0004
6.0	5.9990

(b) with payload

v_{ref}	$v_{steadystate}$
0.1	0.0999
0.2	0.1999
0.3	0.2935

ω_{ref}	$\omega_{steadystate}$
2.0	1.9987
4.0	3.9987
6.0	5.9207

Table 4.3: Khepera III steady state velocity values

(a) without payload

v_{ref}	$v_{steadystate}$
0.1	0.0985
0.3	0.2927
0.5	0.4784

ω_{ref}	$\omega_{steadystate}$
0.5	0.4771
1.0	0.9821
1.5	1.4597

(b) with payload

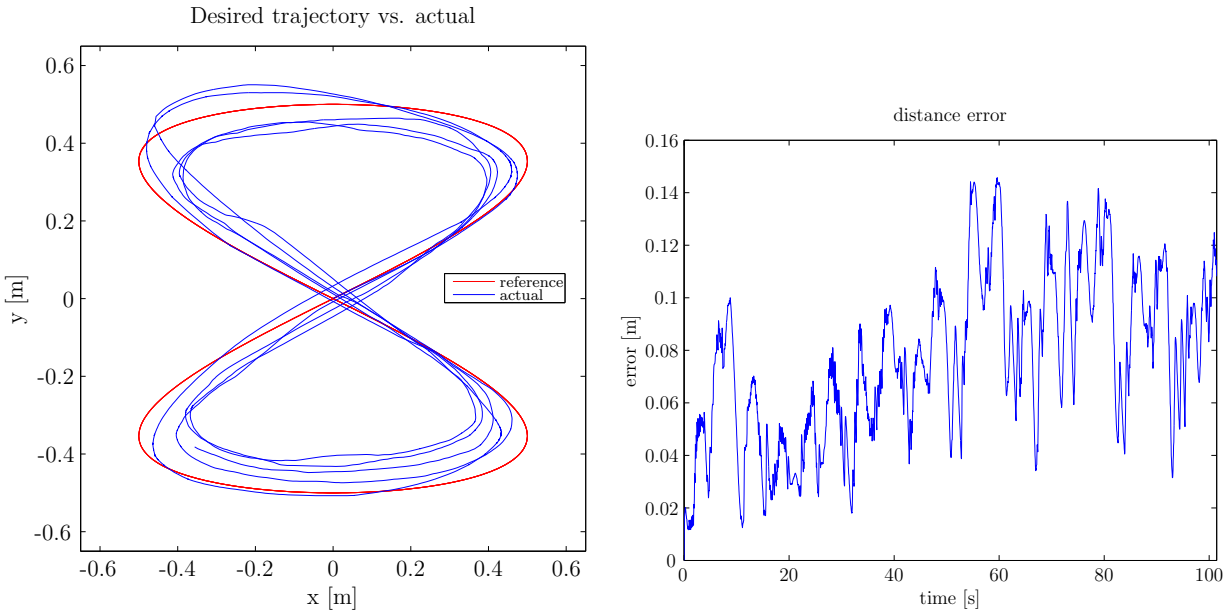
v_{ref}	$v_{steadystate}$
0.1	0.0985
0.3	0.2917
0.5	0.4721

ω_{ref}	$\omega_{steadystate}$
0.5	0.4840
1.0	0.9847
1.5	1.4756

Table 4.4: Pioneer 3-DX steady state velocity values

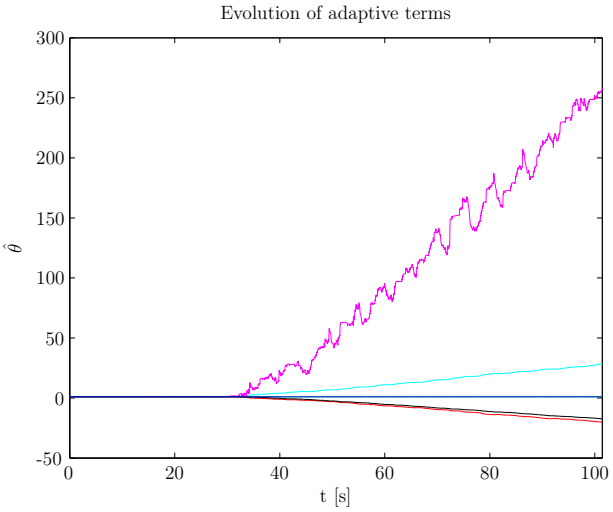
4.4.1 Khepera III

In simulation, it was shown that the initial parameter error does not need to be small for the parameter estimates to converge. In reality, this is not the case. To illustrate this, the adaptive controller was implemented on the Khepera III with all initial parameter estimates chosen as 1. The robot is commanded to follow a figure-eight style trajectory. At $t = 30$ seconds, the parameter adaptation begins. At this time, it can be seen in figure 4.12(b) that the distance error begins to increase with time, indicating a degradation in performance. Figure 4.12(c) shows the parameter estimates. Although the true values are not known, it is clear that some of the parameters are diverging quickly.

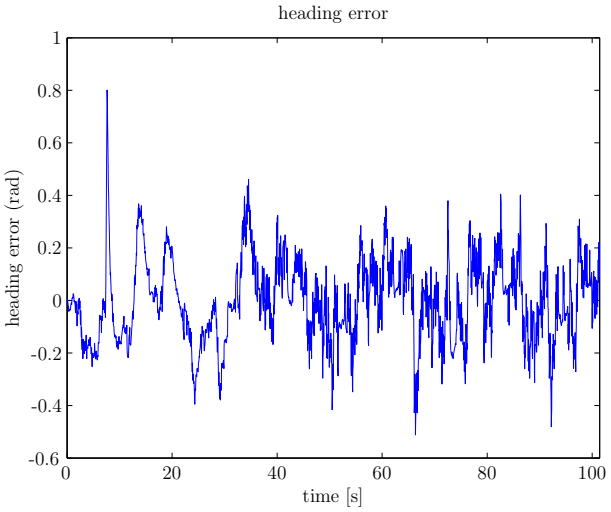


(a) Trajectory

(b) Distance Error



(c) Parameter Evolution



(d) Heading Error

Figure 4.12: Khepera III trajectory and error, with adaptation, poor initial estimates

From the results shown in figure 4.12, it is obvious that a more intelligent estimate of the initial parameters is needed. Using the method outlined in section 4.1, a more reasonable initial estimate is generated.

First, the robot is given inputs of

$$\begin{aligned} v_{ref} &= 0.1 + 0.05\sin(2t) + 0.05\sin(t) + 0.05\sin(t/3) + 0.05\sin(t/11) \\ \omega_{ref} &= 1.5 + 0.5\sin(3t/2) + 0.5\sin(t) + 0.25\sin(t/2) + 0.25\sin(t/5) \end{aligned}$$

The forward and angular velocities were recorded by the robot as it moved. After 2 minutes, the robot was stopped and the data was collected. The velocity data was filtered (figure 4.13)and differentiated to obtain the accelerations (figure 4.14).

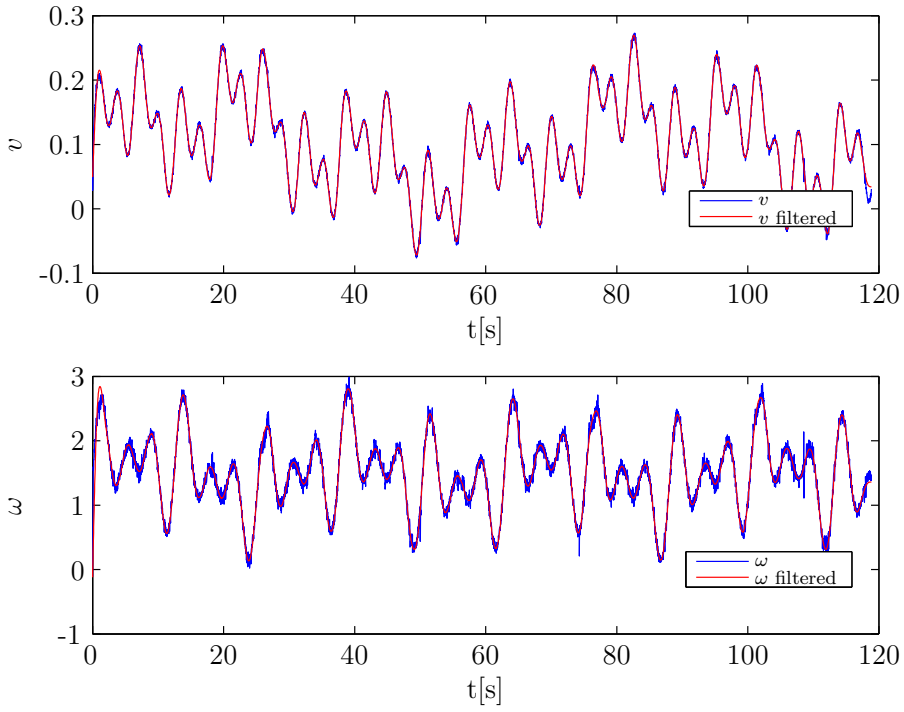


Figure 4.13: Velocity data

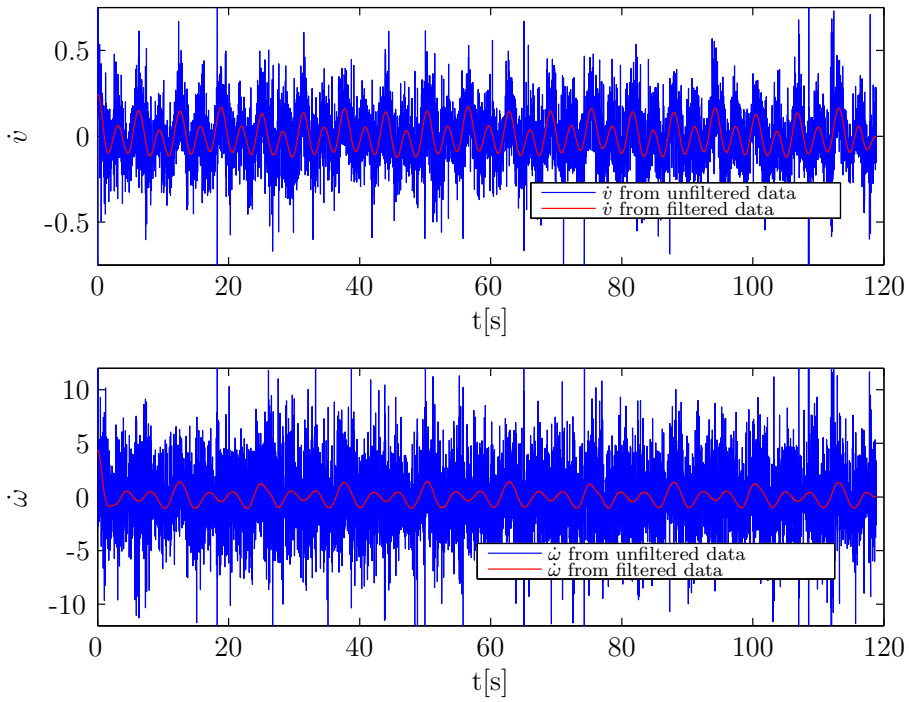
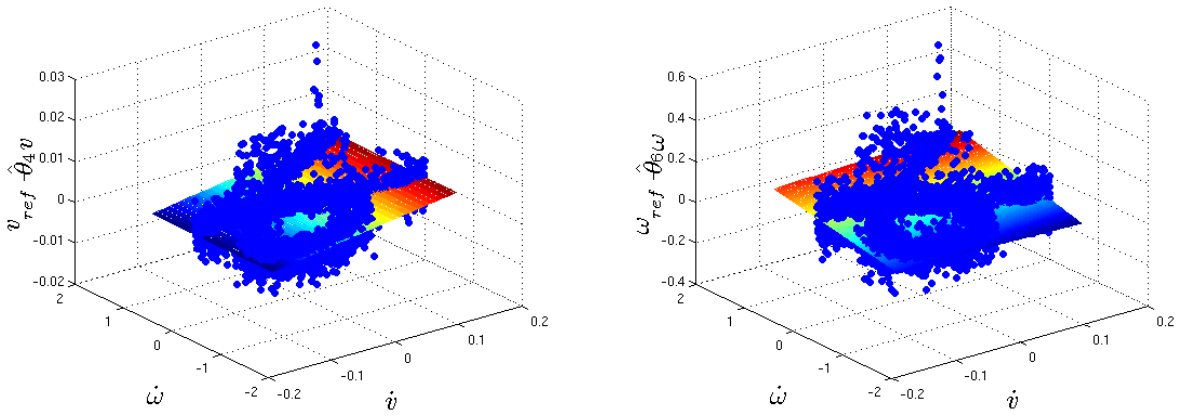


Figure 4.14: Acceleration data

Although the velocity data of figure 4.13 doesn't appear to be too noisy, it is obvious that the acceleration data acquired by differentiating the noisy signal is useless. Here, the data was filtered once forward, and again in reverse to eliminate any added delay. The type of filter used was a Butterworth filter with a cutoff frequency of 0.5 Hz. Using the filtered signals, a linear regression can be performed to obtain estimates of the parameters.



(a) v_{ref} regression

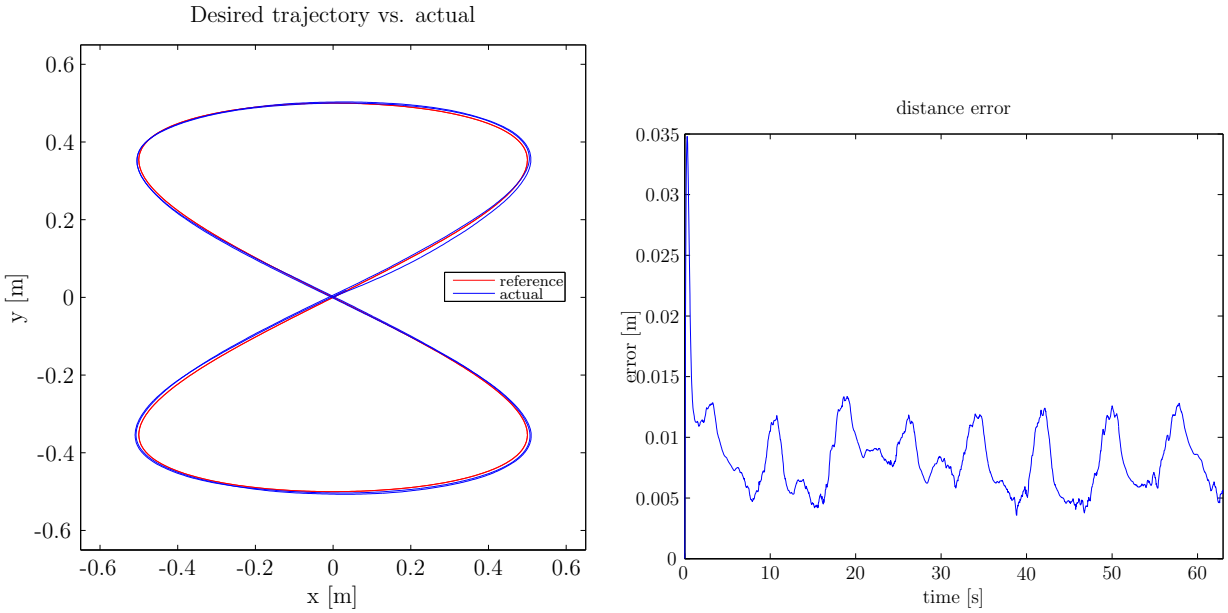
(b) ω_{ref} regression

Figure 4.15: Plots of linear regression

Performing the linear regression as described in equations (4.6) and (4.7) on the filtered signals, $\hat{\boldsymbol{\theta}}$ is found to be

$$\begin{aligned}
\hat{\theta}_1 &= 0.0228 \\
\hat{\theta}_2 &= 0.0568 \\
\hat{\theta}_3 &= -0.0001 \\
\hat{\theta}_4 &= 1.0030 \\
\hat{\theta}_5 &= 0.0732 \\
\hat{\theta}_6 &= 0.9981
\end{aligned}$$

As discussed in Section 4.1 (pg. 27), $\hat{\theta}_4$ and $\hat{\theta}_6$ were expected to be close to 1. Since our values for these parameters obtained from the linear regression are close to 1, this is a good indicator that the parameter estimates may be accurate.

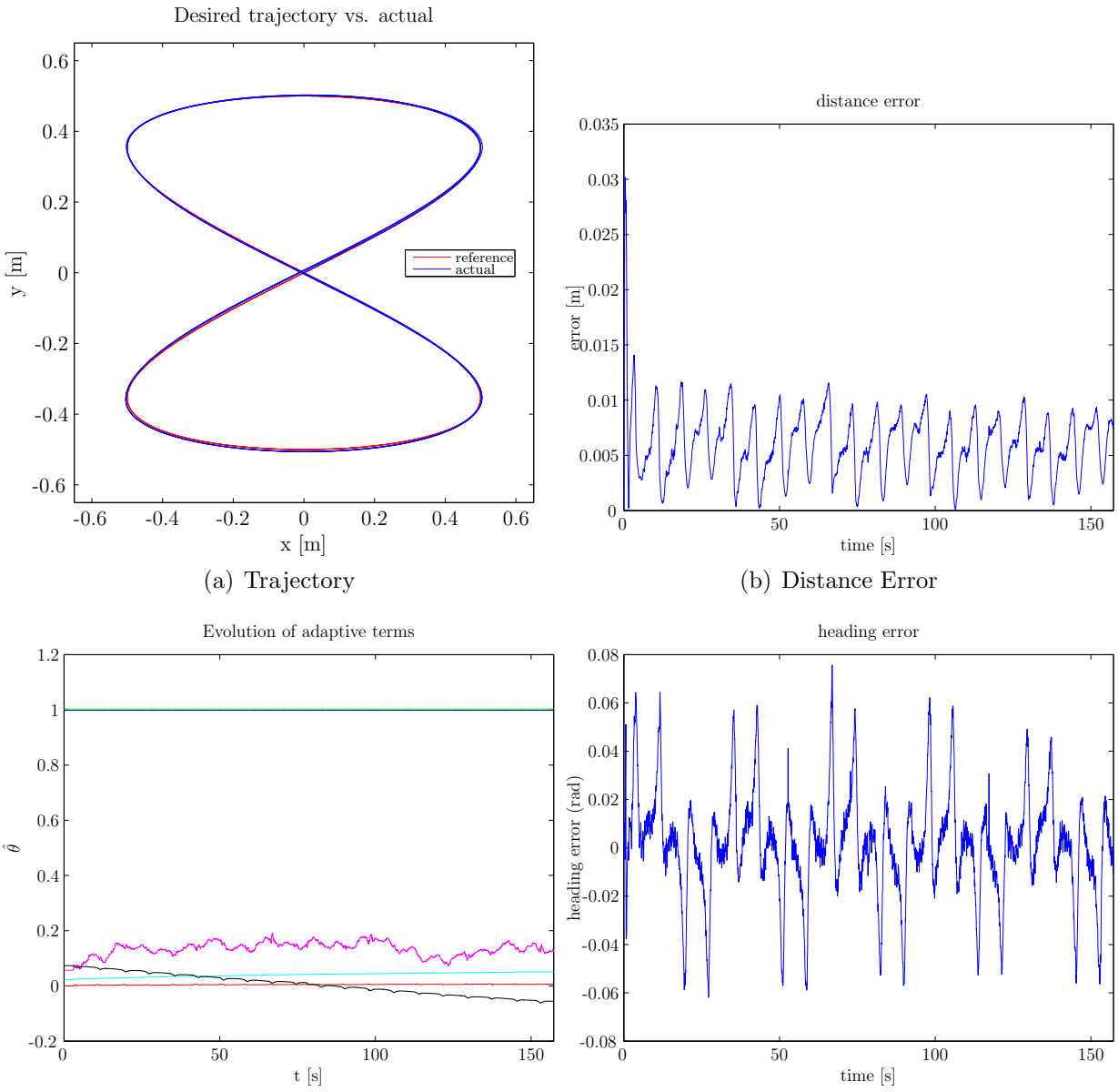


(a) Trajectory (b) Error

Figure 4.16: Khepera III trajectory and error, no adaptation

Using the parameter estimates from the previous page, the controller is again implemented on the Khepera III to follow a figure-eight style trajectory. In this case, there is no parameter adaptation so the accuracy of the initial estimates generated by the proposed method can be better evaluated.

In Figure 4.16, we can see a drastic increase in performance in comparison with the case of all parameters initially chosen as 1 (pg 47).



(a) Trajectory

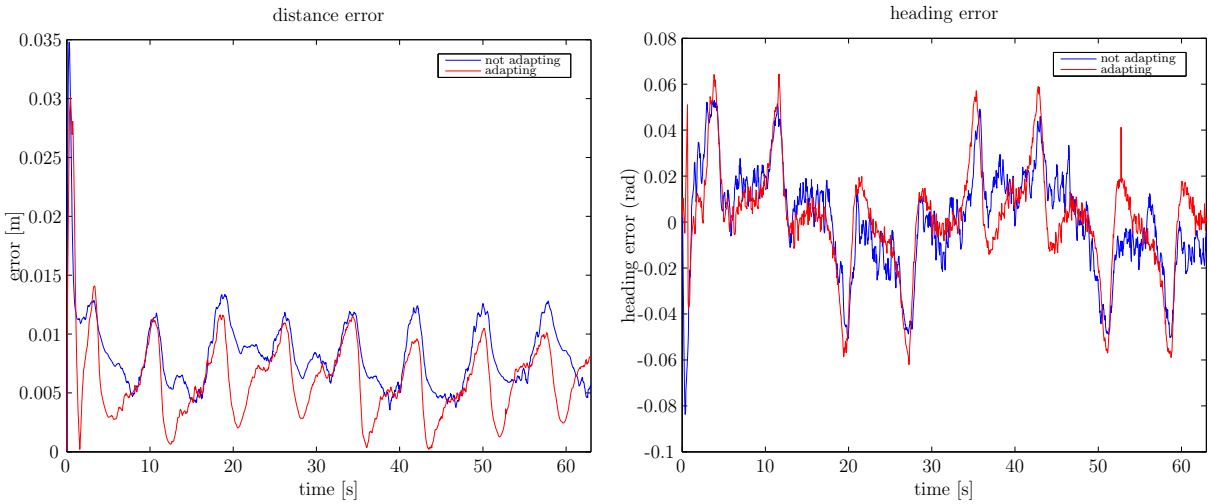
(b) Distance Error

(c) Parameter Evolution

(d) Heading Error

Figure 4.17: Khepera III trajectory and error, with adaptation

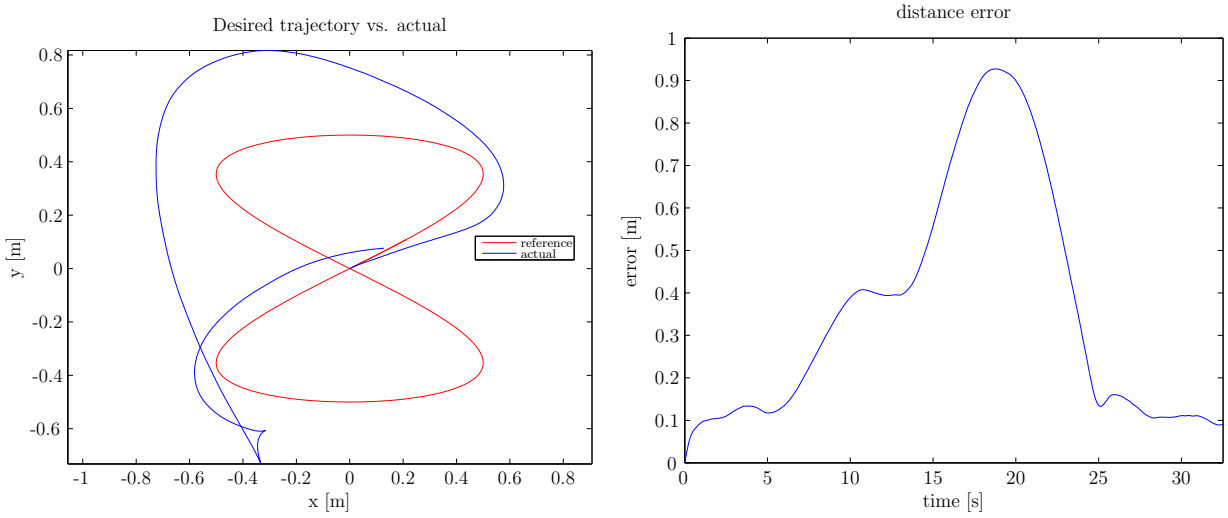
Although in Figure 4.16, the performance of the controller is acceptable with using the initial parameter estimates, and no parameter adaptation, there is still room for improvement. Figure 4.17 shows the same scenario, but with adaptation beginning at $t = 3$ seconds. A slight decrease in distance error can be noticed, along with a small change in the parameter estimates. A comparison of the errors in the two scenarios is shown in Figure 4.18.



(a) Distance Error

(b) Heading Error

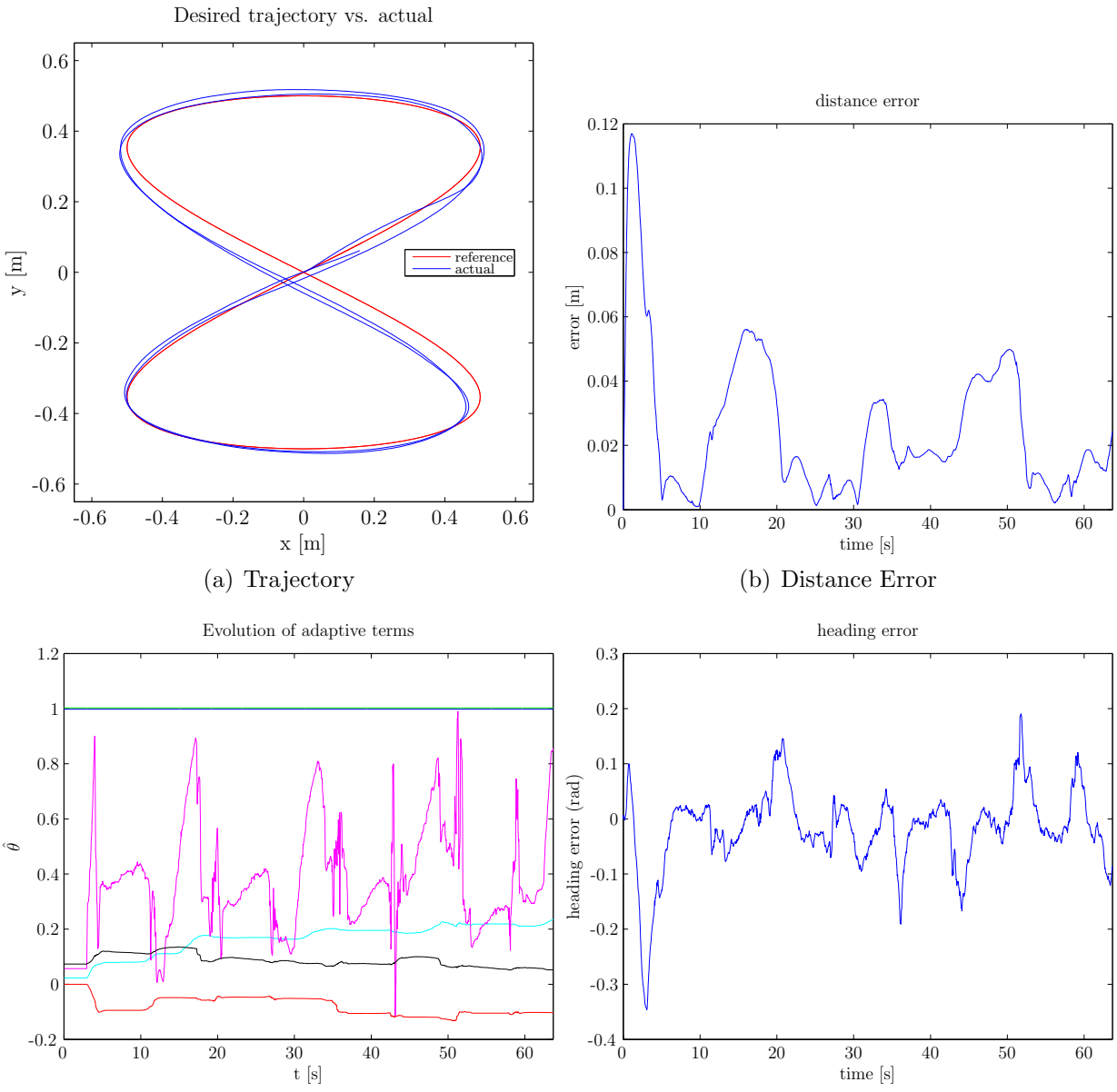
Figure 4.18: Comparison of figures 4.16 and 4.17



(a) Trajectory (b) Error

Figure 4.19: Khepera III trajectory and error with heavy payload, no adaptation

To demonstrate the effectiveness of the adaptive controller even when a payload is added to thereby alter the parameters of the dynamic model, a heavy payload is added to the Khepera III. First, the controller is used without adaptation (Figure 4.19). Obviously, the controller is demonstrating poor performance when a payload is added and the parameter estimates have no been changed. The adaptive controller was then used (Figure 4.20), and performance improved greatly. Although there is still room for improvement in the trajectory tracking, the experiment was stopped because the Khepera III is not suited to carry such a heavy payload.



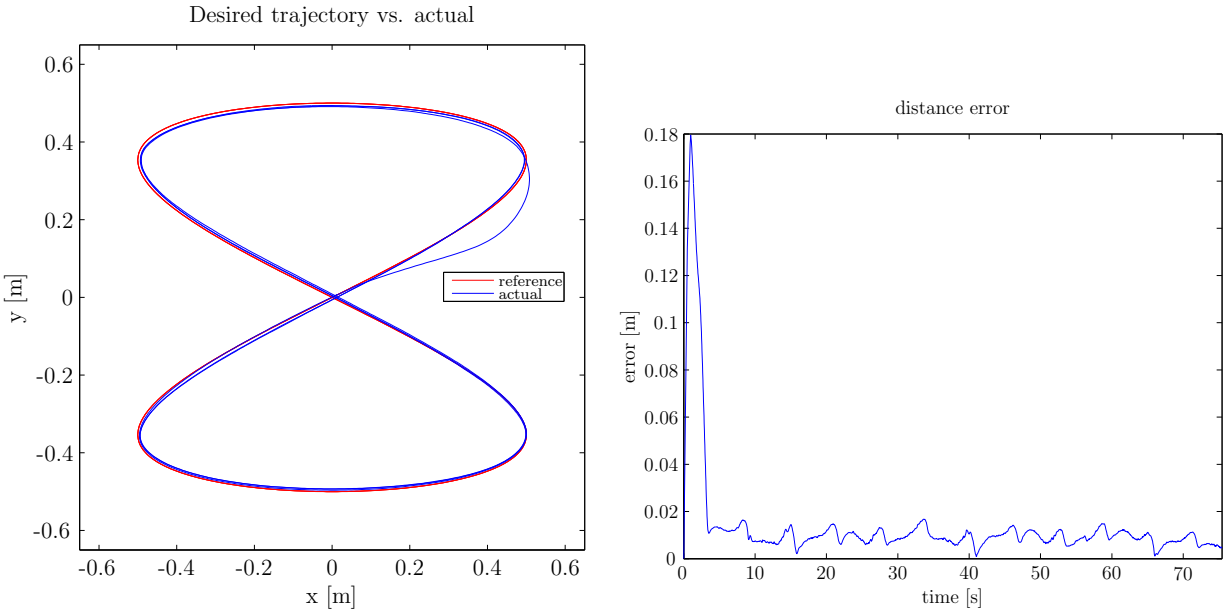
(a) Trajectory

(b) Distance Error

(c) Parameter Evolution

(d) Heading Error

Figure 4.20: Khepera III trajectory and error with heavy payload, with adaptation.

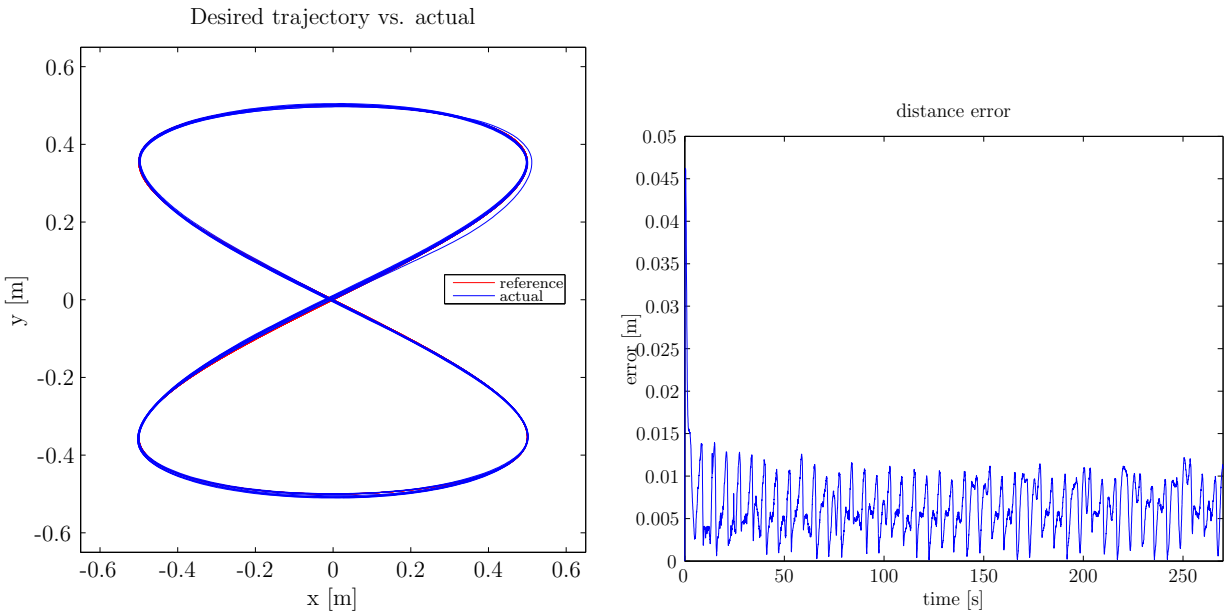


(a) Trajectory (b) Error

Figure 4.21: Khepera III trajectory and error with light payload, no adaptation.

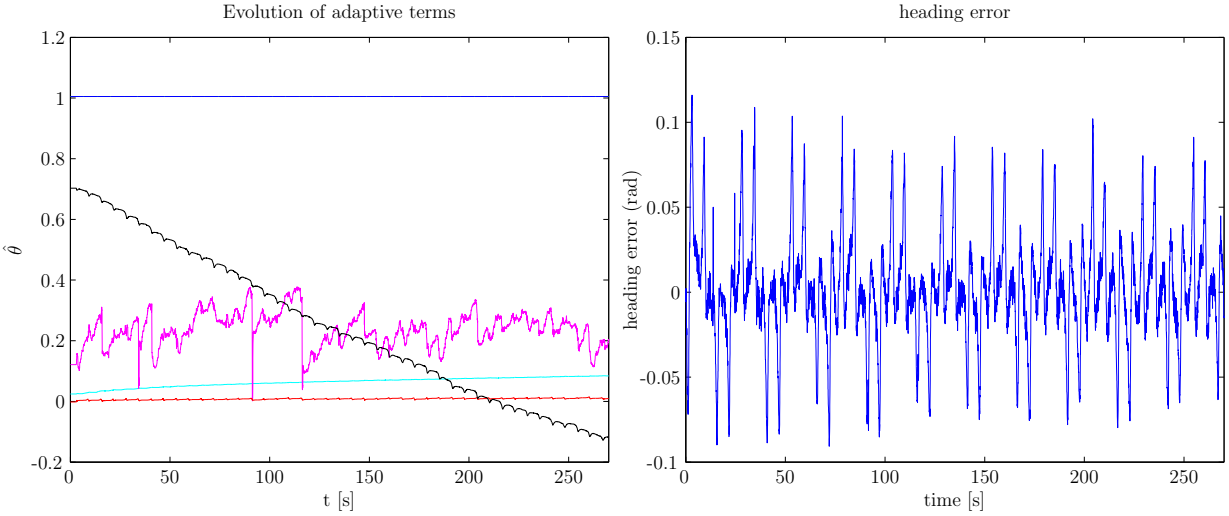
The Khepera III then had a lighter payload attached, and the initial parameter estimates were again generated using the proposed method to demonstrate that it is effective under different loading conditions using the same input signals and method performed initially (page 48). First, the robot with the light payload was controlled using the initial estimates without adaptation, as shown in Figure 4.21. Here, the controller is performing well, again showing that the initial parameter estimate generated by the proposed method is accurate.

The experiment was then executed again using adaptation, and an increase in performance is noticed (Figure 4.22).



(a) Trajectory

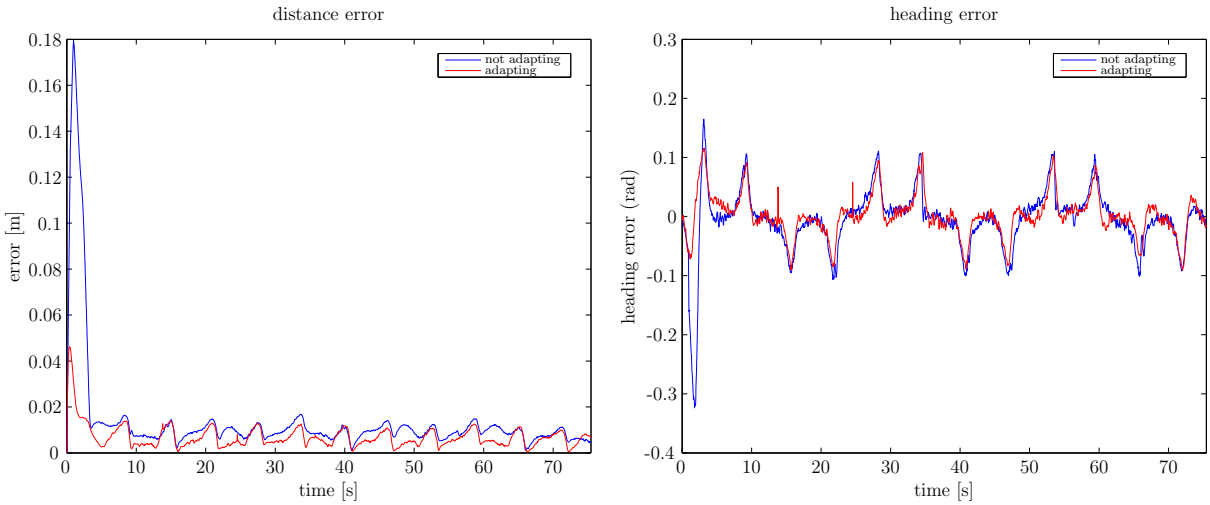
(b) Distance Error



(c) Parameter Evolution

(d) Heading Error

Figure 4.22: Khepera III trajectory and error with light payload, with adaptation.



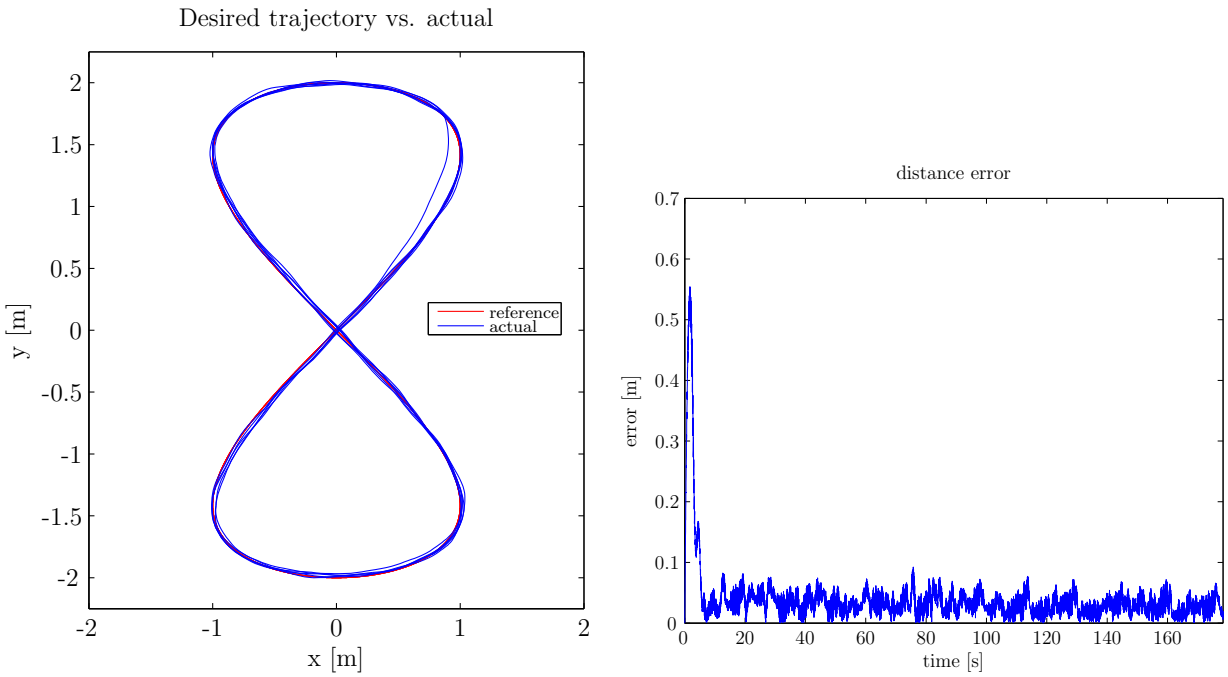
(a) Trajectory

(b) Distance Error

Figure 4.23: Comparison of Figures 4.21 and 4.22

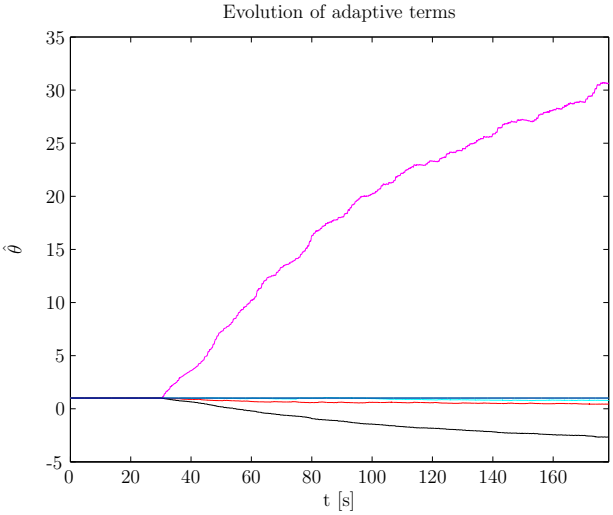
4.4.2 Pioneer 3-DX

As with the Khepera III, we first demonstrate the with poor initial estimates for the parameters, the parameters may not converge. In Figure 4.24, all parameter estimates are initially one. Again, this shows that a more intelligent initial estimate is needed.

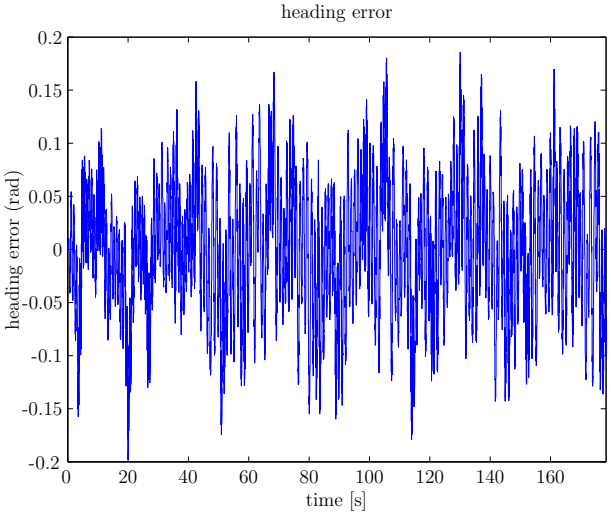


(a) Trajectory

(b) Error



(c) Parameter Evolution



(d) Heading Error

Figure 4.24: Pioneer 3-DX trajectory and error, poor initial estimates, with adaptation.

First, the robot is given inputs of

$$\begin{aligned} v_{ref} &= 0.4 + 0.1\sin(2t) + 0.1\sin(t) + 0.1\sin(t/3) + 0.1\sin(t/11) \\ \omega_{ref} &= 0.75 + 0.25\sin(3t/2) + 0.25\sin(t) + 0.25\sin(t/2) + 0.25\sin(t/5) \end{aligned}$$

The forward and angular velocities were recorded by the robot as it moved. After about 2 and a half minutes, the robot was stopped and the data was collected. The velocity data was filtered (figure 4.25)and differentiated to obtain the accelerations (figure 4.26).

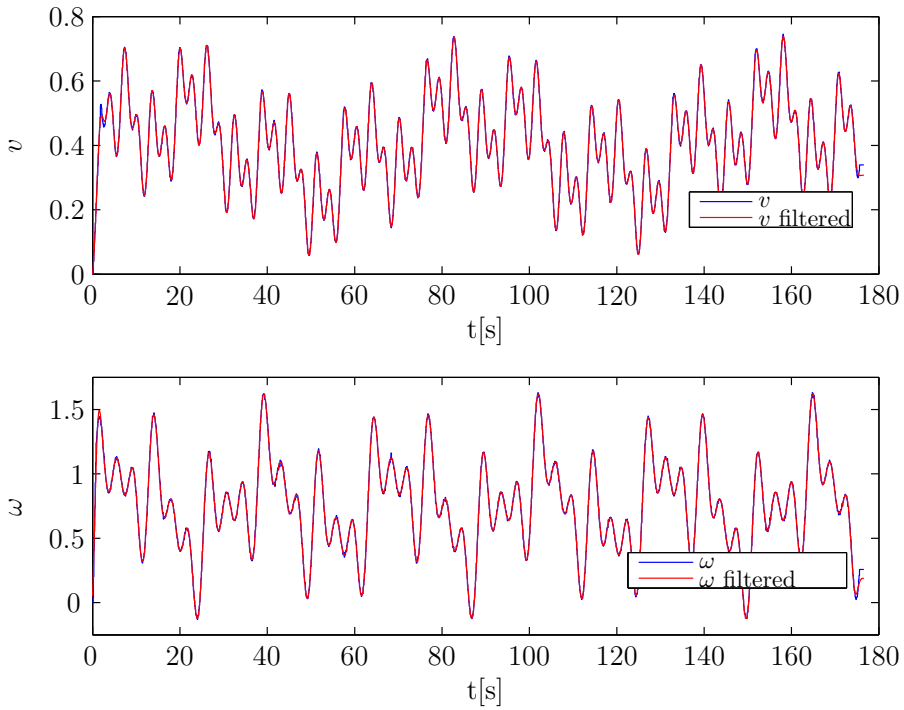


Figure 4.25: Velocity data

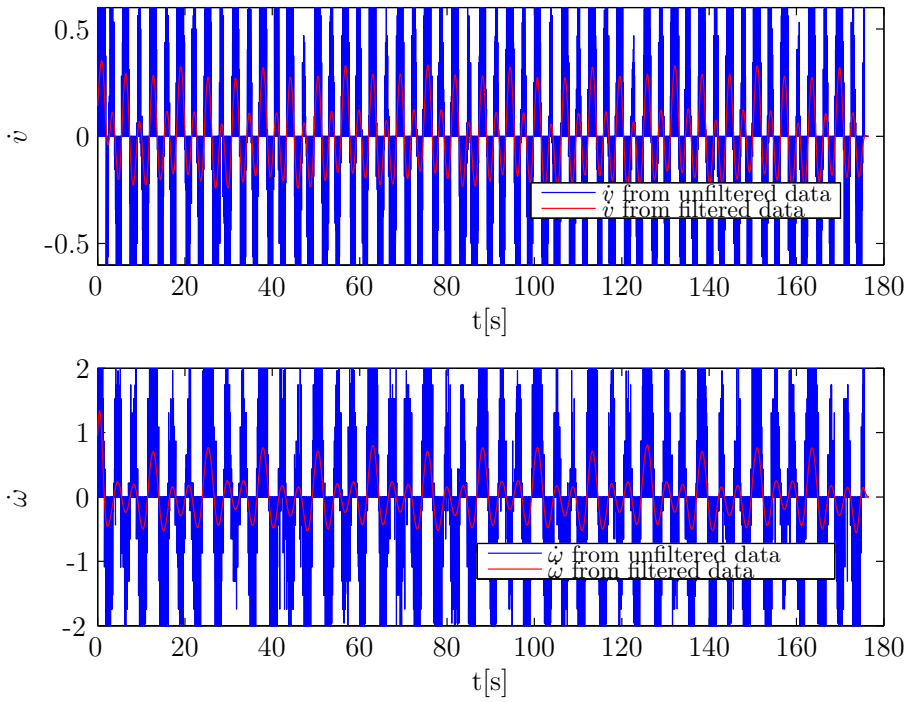
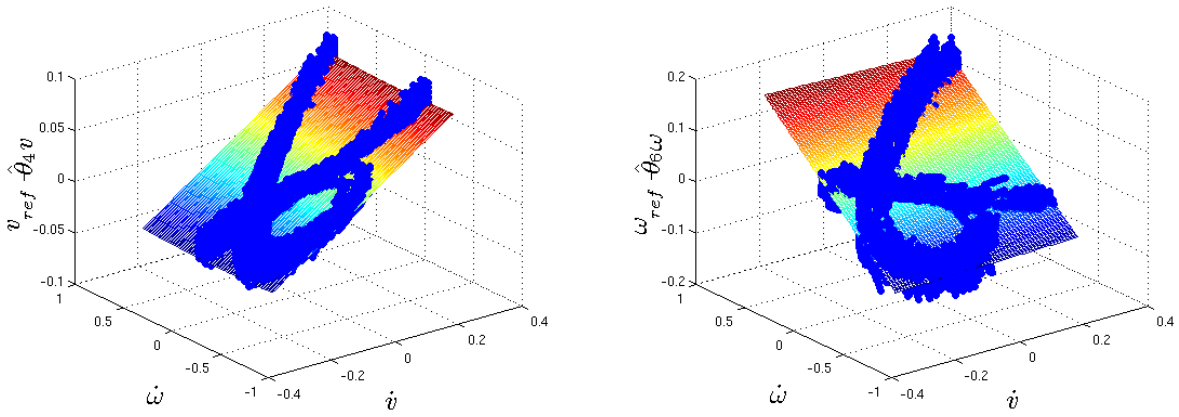


Figure 4.26: Acceleration data

Again, the velocity data of figure 4.25 doesn't appear to be too noisy, but the acceleration data derived from the raw velocities is not good. This demonstrates the importance of filtering any signal before differentiating it. The data was filtered the same way as the case for the Khepera, once forward, and again in reverse to eliminate any added delay using a Butterworth filter with a cutoff frequency of 0.5 Hz. Using the filtered signals, a linear regression can be performed to obtain estimates of the parameters.



(a) v_{ref} regression

(b) ω_{ref} regression

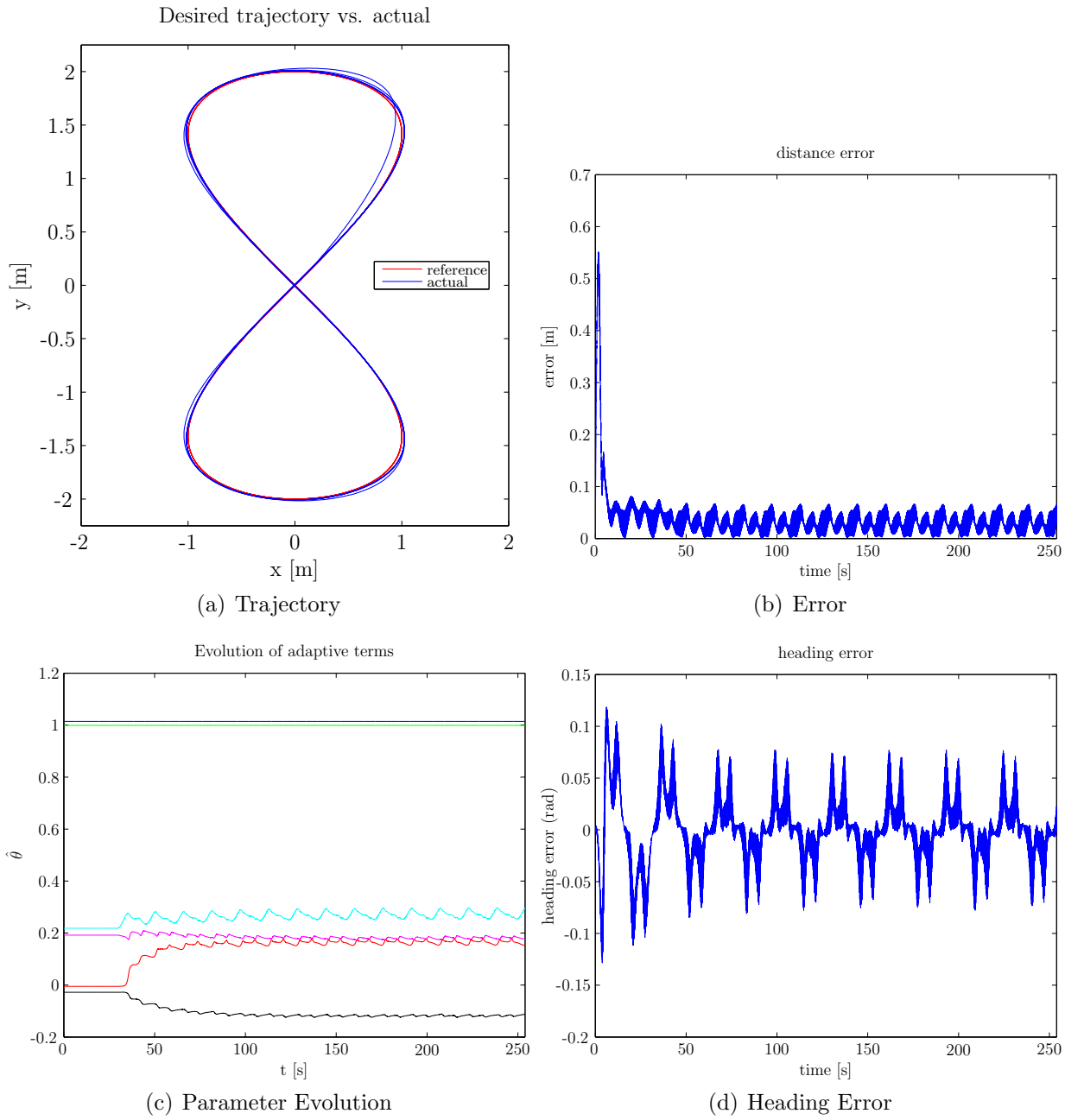
Figure 4.27: Plots of linear regression.

Performing the linear regression as described in equations (4.6) and (4.7) on the filtered signals, $\hat{\boldsymbol{\theta}}$ is found to be

$$\begin{aligned}
\hat{\theta}_1 &= 0.2183 \\
\hat{\theta}_2 &= 0.1918 \\
\hat{\theta}_3 &= -0.0050 \\
\hat{\theta}_4 &= 0.9993 \\
\hat{\theta}_5 &= -0.0279 \\
\hat{\theta}_6 &= 1.0142.
\end{aligned}$$

Once again, the values obtained for $\hat{\theta}_4$ and $\hat{\theta}_6$ are close to one.

In Figure 4.28, the Pioneer 3-DX is using the parameter estimates generated by the proposed method, and at $t = 30$ seconds, adaptation begins. Only a small decrease in the distance error can be noticed, suggesting that initial parameter estimates are decent. Viewing Figure 4.28(c), the parameters quickly converge to new values not far from the initial estimates.



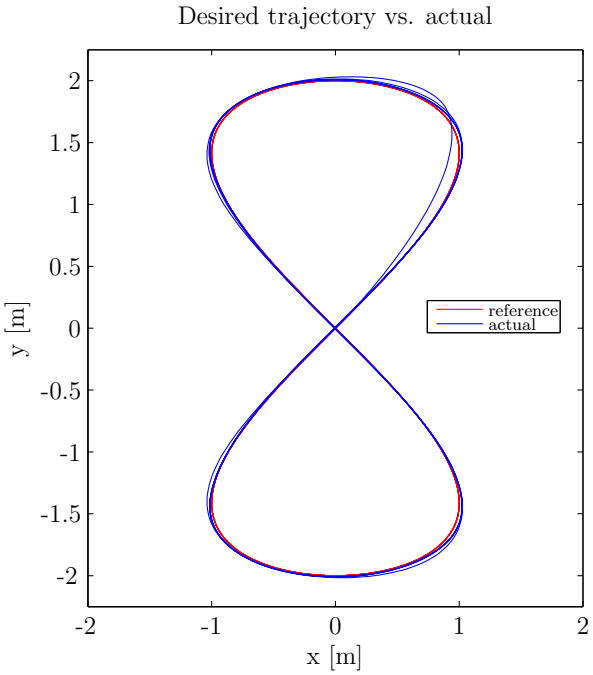
(a) Trajectory

(b) Error

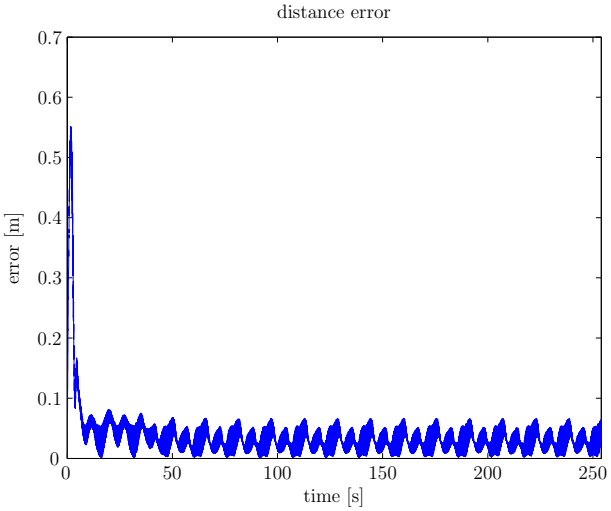
(c) Parameter Evolution

(d) Heading Error

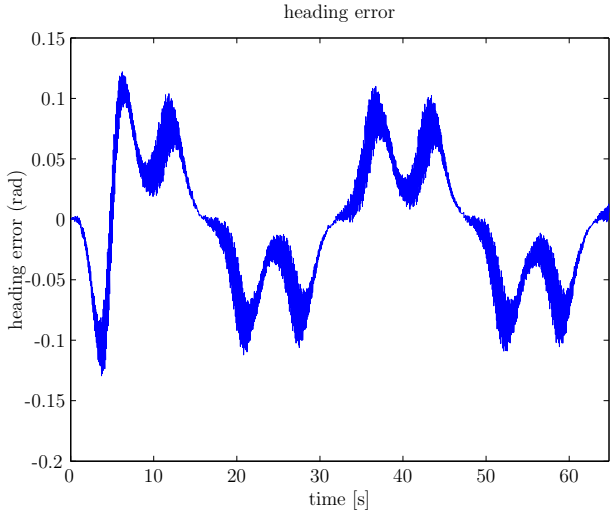
Figure 4.28: Pioneer 3-DX trajectory and error, adapting all parameters except $\hat{\theta}_4$ and $\hat{\theta}_6$



(a) Trajectory



(b) Error

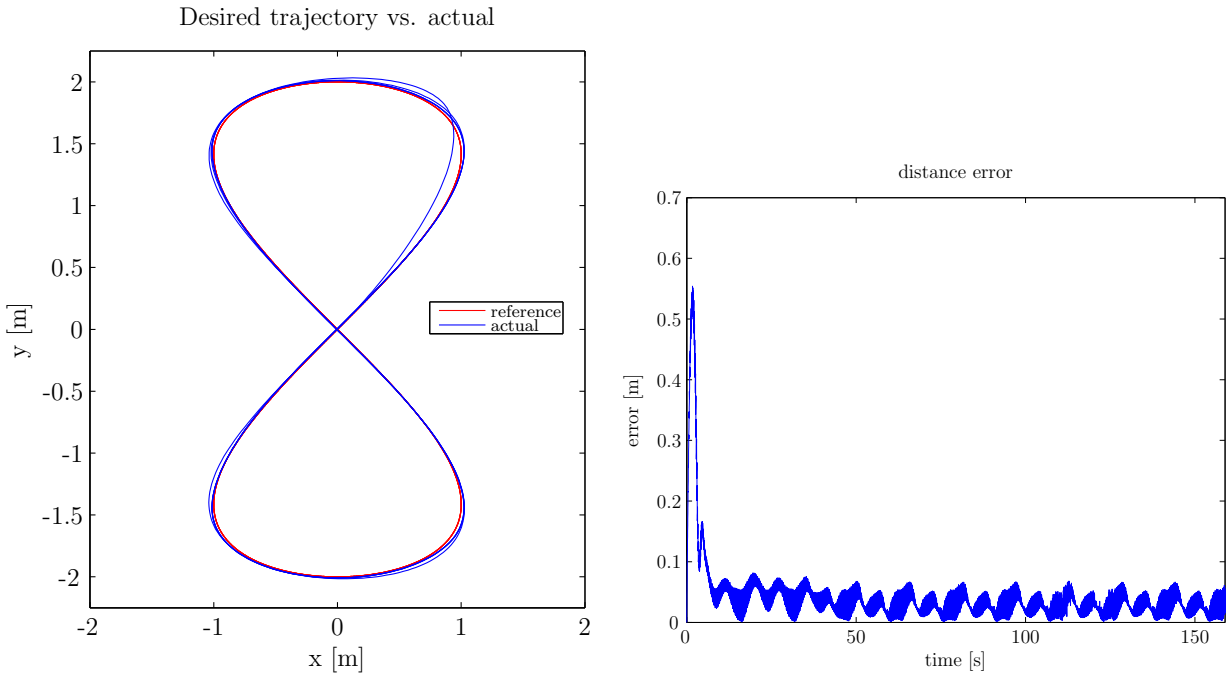


(c) Heading Error

Figure 4.29: Pioneer 3-DX trajectory and error, heavy payload, no adaptation .

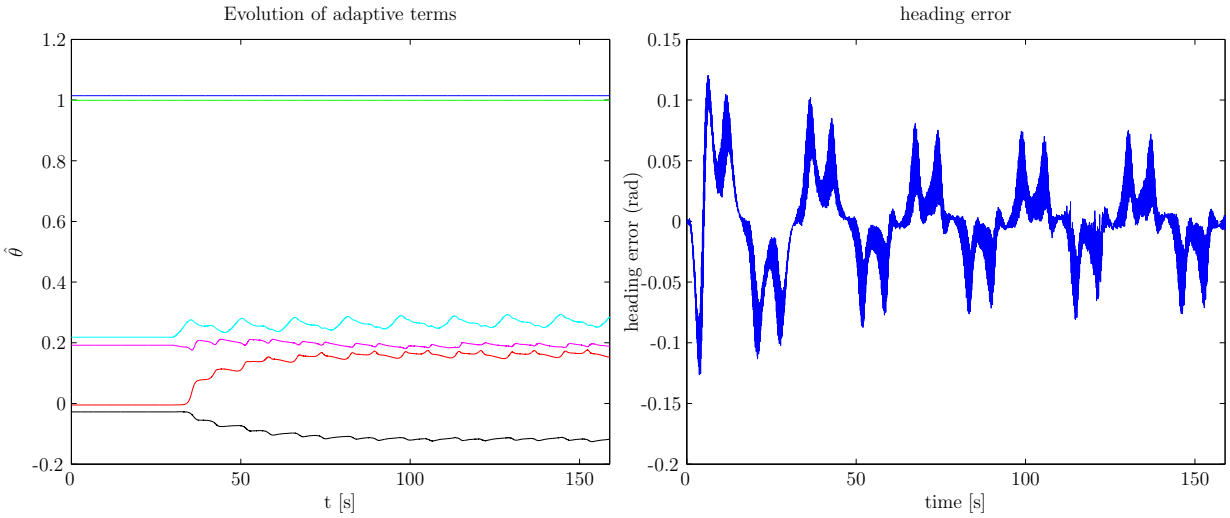
The Pioneer 3-DX was then subjected to a heavy payload. The results of tracking a figure-eight trajectory using the inverse dynamic controller with the estimates for the robot with no payload without adaptation are shown in Figure 4.29.

The same scenario was performed again using adaptation, shown in Figure 4.30. Again, there is a slight improvement in performance after updating is enabled at $t = 30$ seconds, and the parameters quickly converge to new values. It should also be noted that a payload has less of an effect on the larger robot than the smaller one, where the Khepera III was not even able to track a trajectory when a large payload was placed on it.



(a) Trajectory

(b) Error



(c) Parameter Evolution

(d) Heading Error

Figure 4.30: Pioneer 3-DX trajectory and error, heavy payload, with adaptation .

Chapter 5

Conclusions and Future Work

5.1 Conclusions

In this work, a review of the kinematics and dynamics of a differential drive wheeled mobile robot was given. An adaptive controller for the parameters of the dynamic model was also studied. In the physical implementation of the adaptive controller based on the dynamic model, it was noticed that the accuracy of the initial estimate of the parameters is important for the convergence of the parameter estimates, as well as the performance of the inverse dynamic controller. A method for generating an initial parameter estimate was proposed. The estimate was created by giving the robot forward and angular reference velocities, collecting the actual velocities, and performing a linear regression on the data. This method allowed for use on different robotic platforms, and provided acceptable initial estimates. Experimental results, performed on two differentially-driven mobile robots, showed that the proposed method is effective in generating initial estimates. The method proposed here is attractive in that it requires absolutely no knowledge of what the parameters might be, and can easily be implemented on platforms of whose parameters are very

different, without any adjustment.

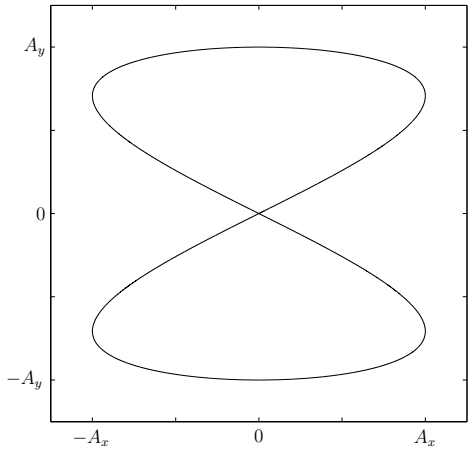
5.2 Future Work

In this work, a robot was given input signals, and data was collected. The initial parameters were then generated on a separate computer. For future work, the initial parameter estimation should be completely autonomously by the robot. The controller applied in the experimental results section did not use any form of saturation control, so extra care was taken when generating trajectories so the robot's actuators would not saturate. An improvement to the current controller would be to incorporate saturation control. The excitation signals used when collecting data for the linear regression were chosen empirically, but it is suspected that there exist excitation signals that would allow for the identification of the parameters in a minimum amount of time. Performing the linear regression with projection, using information as to what the range of parameter estimates may be, could possibly yield more accurate initial estimates.

Appendix A

Generating a Trajectory

A.1 Figure Eight



A figure-eight style trajectory is convenient because it contains an equal number of left and right hand turns, and the trajectory can be run indefinitely while occupying a finite space. To define a figure-eight trajectory, the desired x position should be a sinusoid of frequency $2f$, and the desired y position should be a sinusoid of frequency f.

$$\begin{aligned} x_d &= A_x \sin(2ft) \\ y_d &= A_y \sin(ft) \end{aligned} \tag{A.1}$$

where A_x and A_y are the amplitudes in the x and y directions.

In order to feed the trajectory into a controller, the heading angle, forward velocity, and angular velocity must also be defined. If a dither signal is to be added, the angular acceleration is needed. The desired forward velocity of the robot is found by taking the derivative of the desired x and y positions and taking the square root of the sum of their squares:

$$\dot{x}_d = 2fA_x \cos(2ft) \tag{A.2}$$

$$\dot{y}_d = fA_y \cos(ft) \tag{A.3}$$

thus

$$v_d = \sqrt{\dot{x}_d^2 + \dot{y}_d^2} \tag{A.4}$$

and the desired heading angle is defined by:

$$\phi_d = atan2(\dot{y}_d, \dot{x}_d) \tag{A.5}$$

where $atan2$ is the 360-degree variation of arctan .

Similarly, the desired accelerations and jerks are:

$$\ddot{x}_d = -4f^2 A_x \sin(2ft) \tag{A.6}$$

$$\ddot{y}_d = -f^2 A_y \sin(ft) \tag{A.7}$$

$$\tag{A.8}$$

$$\dddot{x}_d = -8f^3 A_x \cos(2ft) \tag{A.9}$$

$$\dddot{y}_d = -f^3 A_y \cos(ft) \tag{A.10}$$

The desired angular velocity and acceleration are defined as follows:

$$\omega_d = \frac{\ddot{y}_d \dot{x}_d - \ddot{x}_d \dot{y}_d}{\dot{x}_d^2 + \dot{y}_d^2} \tag{A.11}$$

$$\alpha_d = \frac{(\dddot{y}_d \dot{x}_d - \dddot{x}_d \dot{y}_d)(\dot{x}_d^2 + \dot{y}_d^2) - (\ddot{y}_d \dot{x}_d - \ddot{x}_d \dot{y}_d)(2\dot{x}_d \ddot{x}_d + 2\dot{y}_d \ddot{y}_d)}{(\dot{x}_d^2 + \dot{y}_d^2)^2} \tag{A.12}$$

A.2 Figure Eight with Dither

The dither signal is added to the robot's local x coordinate, and can be defined as:

$$d = g \sin(ht) \cos(kt) e^{lt} \tag{A.13}$$

In order to add the dither signal to a trajectory, the first and second derivatives

must be known, and are:

$$\dot{d} = g[h\cos(ht)\cos(kt) - k\sin(ht)\sin(kt) + l\sin(ht)\cos(kt)]e^{lt} \quad (A.14)$$

$$\ddot{d} = g[(l^2 - h^2 - k^2)\sin(ht)\cos(kt) - 2hk\cos(ht)\sin(kt) + \quad (A.15)$$

$$2hl\cos(ht)\cos(kt) - 2kl\sin(ht)\sin(kt)]e^{lt} \quad (A.16)$$

The desired position with dither can then be defined as:

$$x_{d_{dither}} = x_d + d\cos(\frac{\pi}{2} - \phi_d) \quad (A.17)$$

$$= x_d + d\sin\phi_d \quad (A.18)$$

$$(A.19)$$

$$y_{d_{dither}} = y_d + d\sin(\frac{\pi}{2} - \phi_d) \quad (A.20)$$

$$= y_d + d\cos\phi_d \quad (A.21)$$

The remainder of the necessary parameters are found as they were previously:

$$\dot{x}_{d_{dither}} = \dot{x}_d + \dot{d}\sin\phi_d + d\omega_d\cos\phi_d \quad (A.22)$$

$$\dot{y}_{d_{dither}} = \dot{y}_d + \dot{d}\cos\phi_d - d\omega_d\sin\phi_d \quad (A.23)$$

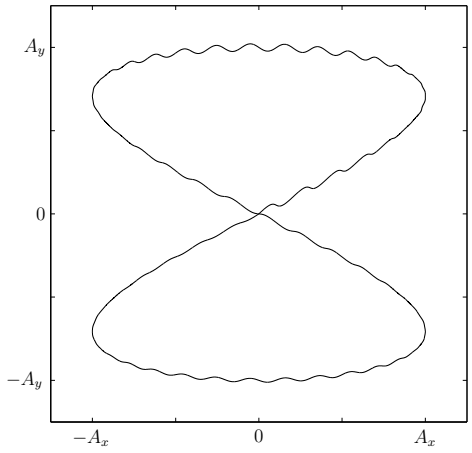
$$v_{d_{dither}} = \sqrt{\dot{x}^2_{d_{dither}} + \dot{y}^2_{d_{dither}}} \quad (A.24)$$

$$\phi_{d_{dither}} = atan2(\dot{y}_{d_{dither}}, \dot{x}_{d_{dither}}) \quad (A.25)$$

$$\ddot{x}_{d_{dither}} = \ddot{x}_d + \ddot{d}\sin\phi_d + 2\dot{d}\omega_d\cos\phi_d - d\omega_d^2\sin\phi_d \quad \text{(A.26)}$$

$$\ddot{y}_{d_{dither}} = \ddot{y}_d + \ddot{d}\cos\phi_d - 2\dot{d}\omega_d\sin\phi_d - d\omega_d^2\cos\phi_d \quad \text{(A.27)}$$

$$\omega_{d_{dither}} = \frac{\ddot{y}_{d_{dither}}\dot{x}_{d_{dither}} - \ddot{x}_{d_{dither}}\dot{y}_{d_{dither}}}{\dot{x}_{d_{dither}}^2 + \dot{y}_{d_{dither}}^2} \quad \text{(A.28)}$$



A.3 Alterative Parameterized Dynamic Model

Referring to pages 8 and 9 for the list of terms of the dynamic model, and the rest of the derivation in section 2.2, the parameterized dynamic model for a differential drive wheeled mobile robot with proportional-derivative gains on the forward and angular velocties is obtained with some changes.

First, equation (2.11) is changed to

$$\begin{bmatrix} v_v \\ v_\omega \end{bmatrix} = \begin{bmatrix} K_{PT}(v_{ref} - v_{me}) - K_{DT}\dot{v}_{me} \\ K_{PR}(\omega_{ref} - \omega_{me}) - K_{DR}\dot{\omega}_{me} \end{bmatrix} \tag{A.29}$$

with

$$v_{me} = \frac{1}{2}[r(\omega_r + \omega_l)] \qquad \omega_{me} = \frac{1}{d}[r(\omega_r - \omega_l)] \tag{A.30}$$

$$v_v = \frac{v_l + v_r}{2} \qquad v_\omega = \frac{v_r - v_l}{2} \tag{A.31}$$

where v_{me} and ω_{me} are the measured forward and angular velocities, and v_{ref} and ω_{ref} are the commanded forward and angular velocities. The derivatives of the commanded velocities were neglected.

Now, the parameterized dynamic model is solved similar to section 2.2

From the second and third equations of (2.6) we can solve for the angular acceleration,

$$\dot{\omega} = \frac{d}{2I_z}(F_{rrx'} - F_{rlx'}) + \frac{e}{I_z}F_{ey'} + \frac{c}{I_z}F_{cy'} + \frac{1}{I_z}\tau_e - \frac{mb}{I_z}(\dot{\bar{v}} + v\omega) \tag{A.32}$$

Combining equations (2.9) and (2.10),

$$\begin{aligned} F_{rrx'} &= \frac{k_a}{R_a R_t}(v_r - k_b\omega_r) - \frac{I_e}{R_t}\dot{\omega}_r - \frac{B_e}{R_t}\omega_r \\ F_{rlx'} &= \frac{k_a}{R_a R_t}(v_l - k_b\omega_l) - \frac{I_e}{R_t}\dot{\omega}_l - \frac{B_e}{R_t}\omega_l \end{aligned} \tag{A.33}$$

And from equation (2.9),

$$\begin{aligned} \omega_r + \omega_l &= \frac{2}{r}v - \frac{1}{r}(v_r^s + v_l^s) \\ \omega_r - \omega_l &= \frac{d}{r}\omega - \frac{1}{r}(v_r^s - v_l^s) \end{aligned} \tag{A.34}$$

The above equation along with (A.29) gives,

$$\begin{aligned} v_r + v_l &= 2k_{PT}\left(v_{ref} - v + \frac{1}{2}(v_r^s + v_l^s)\right) - 2k_{DT}\left(\dot{v} - \frac{1}{2}(v_r^s + v_l^s)\right) \\ v_r - v_l &= 2k_{PR}\left(\omega_{ref} - \omega + \frac{1}{d}(v_r^s - v_l^s)\right) - 2k_{DR}\left(\dot{\omega} - \frac{1}{d}(v_r^s - v_l^s)\right) \end{aligned} \tag{A.35}$$

The lateral velocity is found from the second and third equations of (2.8) as,

$$\bar{v} = b\omega + \bar{v}^s \tag{A.36}$$

Combining the lateral forces acting on each wheel (equation (2.13)) and the angular acceleration (equation (A.32)), and using the second equations of (A.34) and (A.35)

as well as the above equation for lateral velocity,

$$\begin{aligned}\theta_2\dot{\omega} &= \omega_{ref} - \theta_6\omega + \frac{1}{d}\theta_6(v_r^s - v_l^s) + \left(\frac{k_{DR}}{dk_{PR}} + \frac{R_a I_e}{2k_{PR}k_a r}\right)(\dot{v}_r^s - \dot{v}_l^s) + \\ &\quad \frac{R_a R_t}{dk_{PR}k_a}(eF_{ey'} + cF_{cy'} + \tau_e) - \theta_5\dot{\bar{v}}^s - \theta_5 v\omega \end{aligned} \tag{A.37}$$

Similarly, using the lateral forces, the sum of forces about the x' axis, and the first equations of (A.34) and (A.35) with (A.36),

$$\begin{aligned}\theta_1\dot{v} &= v_{ref} - \theta_4 v + \frac{1}{2}\theta_4(v_r^s + v_l^s) + \left(\frac{k_{DT}}{2k_{PT}} + \frac{R_a I_e}{2k_{PT}k_a r}\right)(\dot{v}_r^s + \dot{v}_l^s) + \\ &\quad \frac{R_a R_t}{2k_{PT}k_a}(F_{ex'} + F_{cx'}) + \theta_3\omega^2 + \frac{R_a R_t m}{2k_{PT}k_a}\omega\bar{v}^s \end{aligned} \tag{A.38}$$

The combination of equations (A.36) and (2.7), along with the above two equations gives the dynamic model,

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\psi}\\ \dot{v}\\ \dot{\omega}\end{bmatrix} = \begin{bmatrix} v\cos\psi - a\omega\sin\psi \\ v\sin\psi + a\omega\cos\psi \\ \omega \\ \frac{\theta_3}{\theta_1}\omega^2 - \frac{\theta_4}{\theta_1}v \\ -\frac{\theta_5}{\theta_2}v\omega - \frac{\theta_6}{\theta_2}\omega \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \frac{1}{\theta_1} & 0 \\ 0 & \frac{1}{\theta_2}\end{bmatrix}\begin{bmatrix} v_{ref} \\ \omega_{ref}\end{bmatrix} + \begin{bmatrix}\delta_x \\ \delta_y \\ 0 \\ \bar{\delta}_v \\ \bar{\delta}_\omega\end{bmatrix} \tag{A.39}$$

with

$$
\begin{aligned}
\theta_1 &= \left(\frac{R_a}{k_a}\right)/(2rk_{PT}) \\
\theta_2 &= \left(\frac{R_a}{k_a}(I_e d^2 + 2R_t r(I_z + mb^2)) + 2rdk_{DR}\right)/(2rdk_{PR}) \\
\theta_3 &= \frac{R_a}{k_a} mbR_t/(2k_{PT}) \\
\theta_4 &= \frac{R_a}{k_a}\left(\frac{k_a k_b}{R_a} + B_e\right)/(rk_{PT}) + 1 \\
\theta_5 &= \frac{R_a}{k_a} mbR_t/(dk_{PR}) \\
\theta_6 &= \frac{R_a}{k_a}\left(\frac{k_a k_b}{R_a} + B_e\right) d/(2rk_{PR}) + 1
\end{aligned}
$$

Appendix B

Installing and Using the Khepera III Toolbox

B.1 Installing the toolbox

The Khepera III toolbox requires nothing to be done onboard the Khepera.

First, download the latest revision of the toolbox;

```
svn checkout https://khepera3toolbox.svn.sourceforge.net/svnroot/khepera3toolbox
```

The Khepera III toolbox is now installed in the directory in which `svn` was run.

The following environment variables need to be added to `~/.bashrc`.

First, type

```
vi ~/.bashrc
```

Then, at the bottom of the file, add

```
export K3_ROOT=/path/to/your/khepera3toolbox
```

```
export PATH=$PATH:$K3_ROOT/Scripts
```

A new terminal window will need to be opened for bash to recognize the new environment variables (alternatively, the user can just type `bash` in the current terminal).

B.2 Writing a program

First, the cross-compiler for the ARM architecture must be installed. This is availeble from K-team's website by downloading the Korebot toolchain. Once the toolchain is downloaded, the path to the ARM compiler must be added to `~/.bashrc`. In `~/.bashrc`, add

```
export PATH=$PATH:/path/to/your/arm/toolchain/bin
```

Now, to make a new program, in the `Programs` directory of the Khepera III toolbox, type

```
k3-create-program program_name
```

where `program_name` is the name of the program you wish to create. Doing this creates a template file in C with all of the necessary header files, as well as a makefile within a directory named after the program just created. Entering the directory of the program and typing `make` builds the program.

The program can be placed on the Khepera by typing

```
k3put +ip_address program_name
```

Where `ip_address` is the IP address of the Khepera III robot being used.

Bibliography

[1] F. D. Boyden and S. A. Velinsky, "Dynamic modeling of wheeled mobile robots for high load applications," in *IEEE International conference on Robotics and Automation*, August 2002.

[2] T. Fukao, H. Nakagawa, and N. Adachi, "Adaptive tracking control of a nonholonomic mobile robot," in *IEEE Transactions on Robotics and Automation*, October 2000.

[3] M. S. Kim, J. H. Shin, and J. J. Lee, "Design of a robust adaptive controller for a mobile robot," in *Internation Conference on Intelligent Robots and Systems*, 2000.

[4] R. Fierro and F. L. Lewis, "Control of a nonholonomic mobile robot: Backstepping kinematics into dynamics," *Journal of Robotic Systems*, vol. 14, pp. 149–163, 1997.

[5] Y. Zhang, D. Hong, J. H. Chung, and S. Velinsky, "Dynamic model based robust tracking control of a differentially steered wheeled mobile robot," in *Proceedings of the American Control Conference*, June 1998.

[6] J. S. Choi and B. K. Kim, "Near minimum-time direct voltage control algorithms for wheeled mobile robots with current and voltage constraints," in *Robotica*, 2001.

[7] R. Rajagopalan and N. Barakat, "Velocity control of wheeled mobile robots using computed torque control and its performance for a differentially driven robot," *Journal of Robotic Systems*, vol. 14, pp. 325–340, 1997.

[8] C. D. L. Cruz and R. Carelli, "Dynamic model based formation control and obstacle avoidance of multi-robot systems," *Robotica*, vol. 26, pp. 345–356, 2008.

[9] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics; Modelling, Planning and Control*. London, England: Springer-Verlag, 2009.

[10] G. Dudek and M. Jenkin, *Handbook of Robotics*, ch. 20, Inertial Sensors, GPS, and Odometry. Springer-Verlag, 2008.

[11] G. Oriolo, A. D. Luca, and M. Vendittelli, "Wmr control via dynamic feedback linearization: design, implementation, and experimental validation," in *IEEE Transactions on Control Systems Technology*, November 2002.

[12] F. N. Martins, W. C. Celeste, and R. Carelli, "An adaptive dynamic controller for autonomous mobile robot trajectory tracking," *Control Engineering Practice*, vol. 16, pp. 1354–1363, 2008.

[13] N. R. Draper and H. Smith, *Applied regression analysis*. New York, New York: John Wiley & Sons, 1998.

[14] P. Ioannou and J. Sun, *Robust Adaptive Control*. Prentice Hall, 1995.

[15] "Player/stage." `http://playerstage.sourceforge.net/`, May 2011.

[16] A. Whitbrook, *Programming mobile robots with Aria and Player*. London, England: Springer-Verlag, 2010.

[17] F. Reyes and R. Kelly, "On parameter identification of robot manipulators," in *Proceedings of the 1997 IEEE international conference on robotics and automation*, April 1997.

[18] G. Campion, G. Bastin, and B. D'andrea-Novel, "Structural properties and classification of kinematic and dynamic models of wheeled mobile robots," in *IEEE Transactions on Robotics and Automation*, February 1996.

[19] "Khepera iii toolbox." `http://en.wikibooks.org/wiki/Khepera_III_Toolbox`, May 2011.

[20] H. M. Deitel, *How to Program C*. Upper Saddle River, New Jersey: Prentice Hall, 2001.

[21] W. Khalil, *Modeling, Identification, and Control of Robots*. Butterworth-Heinemann, 2004.