86739: Mobile Robots
Mobile robot control

Date: Mon 0:00, 23.05.16

Prof. G. Garcia, Prof. P. Martinet

Rabbia Asghar, BEng, Ernest Skrzypczyk, BSc

Contents

List of Figures

Chapter 1

Robot type (2, 0)

Direct kinematic model

Type (2, 0) mobile robot is more commonly known as a differential drive robot or a unicycle.

Theoretical model

Kinematics of a differential drive robot can be modelled based on the non slipping and non skidding constraints. To define the kinematic model and implement it in Simulink following definitions are taken in accordance to the data provided in the laboratory:

- $(V, \omega)^T$: cartesian velocities of the mobile robot in m/s and rad/s
- $(x, y, \theta)^T$): posture coordinates of the mobile robot
- (φ_1, φ_2): the fixed wheel angles, for right and left wheels respectively
- r: radius of the wheels in m
- L: distance of the fixed wheel in m from the mobile frame ($l = 2L$)

Right wheel velocity can be defined in terms of φ_1 and also in terms of ω as shown in equations (1.1) and (1.2).

$$V_R = r\dot{\varphi}_1 \tag{1.1}$$

$$V_R = \omega(R + L) \tag{1.2}$$

where R is the distance of the robot from $i$nstantaneous $c$entre of $r$otation (ICR). Similarly, left wheel velocity can be defined in terms of φ_2 and also in terms of ω as shown in equations (1.3) and (1.4).

$$V_L = r\dot{\varphi}_2 \tag{1.3}$$

$$V_L = \omega(R - L) \tag{1.4}$$

Substituting equations (1.1), (1.2), (1.3) and (1.4) together, ω can be expressed in (1.5).

$$\omega = \frac{r\dot{\varphi}_1 - r\dot{\varphi}_2}{2L} \tag{1.5}$$

Velocity of the robot V (1.6) can be expressed as average of V_R and V_L.

$$V = \frac{r\dot{\varphi}_1 + r\dot{\varphi}_2}{2L} \tag{1.6}$$

Following the above derivation, V and ω can now be expressed together in terms of $\dot{\varphi}_1$ and $\dot{\varphi}_2$.

$$\begin{bmatrix} V \\ \omega \end{bmatrix} = \begin{bmatrix} r/2 & r/2 \\ r/2L & r/2L \end{bmatrix} \begin{bmatrix} \dot{\varphi}_1 \\ \dot{\varphi}_2 \end{bmatrix} \tag{1.7}$$

(1.7) is implemented in the *d*irect *k*inematic *m*odel block in the Simulink model. In order to define inverse kinematic model, $\dot{\varphi}_1$ and $\dot{\varphi}_2$ is expressed in terms of V and ω (1.8).

$$\begin{bmatrix} \dot{\varphi}_1 \\ \dot{\varphi}_2 \end{bmatrix} = \begin{bmatrix} 1/r & L/r \\ 1/r & -L/r \end{bmatrix} \begin{bmatrix} V \\ \omega \end{bmatrix} \tag{1.8}$$

Velocity in x and y direction of the reference frame can be computed using following relation.

$$V_x = V\cos\theta \tag{1.9}$$

$$V_y = V\sin\theta \tag{1.10}$$

Given initial posture of the robot, V_x, V_y and ω can be integrated to find the new position and orientation of robot. This is implemented in the localization block in the Simulink model. Posture kinematic model ξ of a (2, 0) mobile robot is defined in equation (1.11),

$$\dot{\xi} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} u \tag{1.11}$$

where u is the control vector.

$$u = \begin{bmatrix} V \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} V \\ \omega \end{bmatrix} \tag{1.12}$$

Static decoupling control

First, the static feedback control is implemented on the (2, 0) mobile robot model.

Theoretical study

Static feedback control can not be used to control posture of the robot, since it leads to singularity. However, a point located at a distance from mobile frames origin, that is not situated at the ${}^{m}y$ axis can be controlled. A point P' on the x axis of the mobile frame is taken to implement static feedback control. P' is represented as h in the equation (1.13) and in the Simulink model. The distance of point P' from the origin of the mobile frame O_m is defined as d and selected as $15cm$ in the Simulink model.

$$h = \begin{bmatrix} x + d\cos(\theta) \\ y + d\sin(\theta) \end{bmatrix} \tag{1.13}$$

Coordinates x and y of point P' can be controlled and are represented as subsystem z_1, whereas the orientation θ of the robot cannot and is represented as z_2. Using static feedback control, we can control x and y coordinates of point P' but the orientation of the robot can not be controlled. We assign z_1 as the subsystem that can be controlled and is equal to h, while z_2 is the other subsystem that is equal to orientation of the robot θ. Equation (1.14) represents h^d as desired trajectory for the controllable subsystem ${z_1}^d$. A circular trajectory with counter-clockwise direction is specified as the desired trajectory.

$$h^d = \begin{bmatrix} R\cos(\omega^d \cdot t) \\ R\sin(\omega^d \cdot t) \end{bmatrix} \tag{1.14}$$

In order to track the desired trajectory error $e(t)$ is defined as the difference in desired trajectory and the current position of the robot, as shown in equation (1.15).

$$e(t) = {z_1}^d - z_1 \tag{1.15}$$

Taking the first time derivative of $e(t)$ results in

$$\dot{e}(t) = \dot{z}_1^d - \dot{z}_1 \tag{1.16}$$

where $\dot{z}_1^d$ equals

$$\dot{z}_1^d = \begin{bmatrix} -R\omega^d \sin(\omega^d \cdot t) \\ R\omega^d \cos(\omega^d \cdot t) \end{bmatrix} \tag{1.17}$$

and $\dot{z}_1$ equals

$$\dot{z}_1 = \begin{bmatrix} cos(\theta) \cdot V - d\sin(\theta) \cdot \dot{\theta} \\ sin(\theta) \cdot V + d\cos(\theta) \cdot \dot{\theta} \end{bmatrix} \tag{1.18}$$

$\dot{z}_1$ can now be expressed in terms of control vector u.

$$\dot{z}_1 = K(\theta) \cdot \begin{bmatrix} V \\ \omega \end{bmatrix} \tag{1.19}$$

where

$$K(\theta) = \begin{bmatrix} cos(\theta) & -d\sin(\theta) \\ sin(\theta) & d\cos(\theta) \end{bmatrix} \tag{1.20}$$

$K(\theta)$ can be inverted since $d \neq 0$. In order for the error to be asymptotically stable and converge to zero, the equation (1.21) is specified.

$$\dot{e}(t) + K_p \cdot e(t) = 0 \tag{1.21}$$

A positive value of K_p ensures that poles of the error are located in the left half of the complex plane and thus the error is asymptotically stable. The control parameter K_p will need to be tuned for the desired control response, which is user and application dependent. Substituting (1.21) with (1.15) and (1.16) results in

$$\dot{z}_1^d - \dot{z}_1 + K_p \cdot ({z_1}^d - z_1) = 0 \tag{1.22}$$

$$K(\theta) \cdot \begin{bmatrix} V \\ \omega \end{bmatrix} = \dot{z}_1^d + K_p \cdot ({z_1}^d - z_1) \tag{1.23}$$

We call the right hand side of (1.23) auxiliary control w. Static state feedback control u can be expressed by following equation.

$$u = K^{-1}(\theta) \cdot w \tag{1.24}$$

where

$$K^{-1}(\theta) = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\frac{\sin(\theta)}{d} & \frac{\cos(\theta)}{d} \end{bmatrix} \tag{1.25}$$

Validation and simulation

There is no direct control of the position of the robot with respect to its mobile frame O_m, however control of a point on the robot results indirectly in controlling the position and orientation of the robot, because of its physical structure. Consequent of that it can be observed that the robot is able to maintain desired angular velocity ω^d. However there is constant error θ^e in the orientation of the robot. Using the desired trajectory (1.14) allowed to compute desired orientation and determine error during simulation.

$$\theta^d = \omega^d \cdot t + \pi/2 \tag{1.26}$$

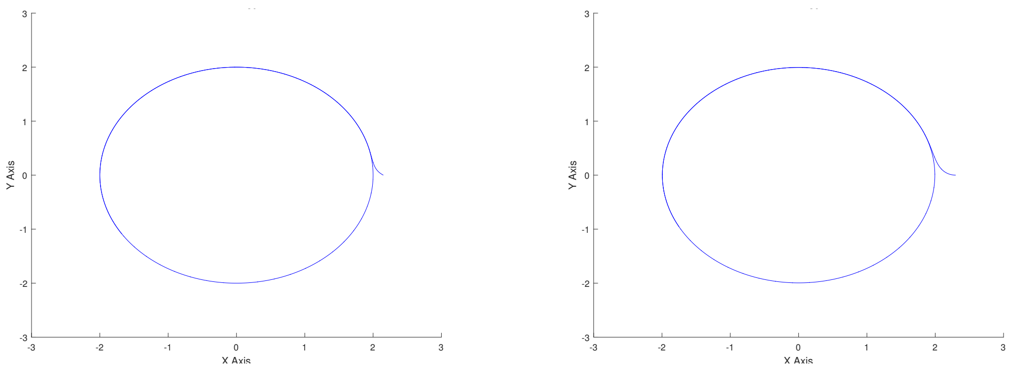

Figure 1.1: Trajectory of point P' and robot position using static decoupling control for accurate robot parameters.

The simulation was performed with tuned control gain to the value $K_p = 10$. It can be observed from the plots [1] and [1], that static feedback control can act on the trajectory of point P' very well. The error in x position of the point P' reduced to 0 without any overshoot and settles within $1s$. The error in y position of the graph was negligible of the order $10^{-11}m$. As discussed above the error in θ did not converge to zero. It rises from $-\pi/2$ to zero and then overshoots to a constant value of approximately $0.8rad$.

The simulation was then carried out with $\pm 10\%$ error in the radii of the wheels and the trackgauge in inverse kinematic model. This was done to simulate errors made in measurements of the physical robot

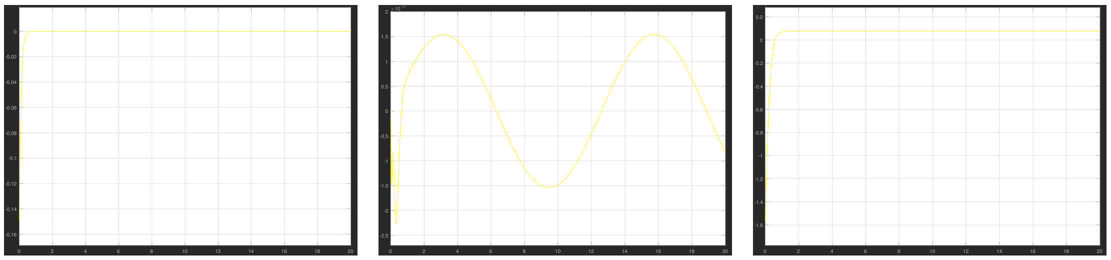

Figure 1.2: Errors in x, y and θ using static decoupling control for accurate robot parameters.

model and changes in these parameters in time due to wear and tear or random events like accidents. In the simulation of $\pm 10\%$ error in robot parameters, the trajectory for P' and robot position appeared as well as for the default case. However, the plots of error [1] in x and [1], in y revealed $\pm 0.01m$ oscillations around the desired trajectory. The error in θ settled in constant value of $0.08rad$.

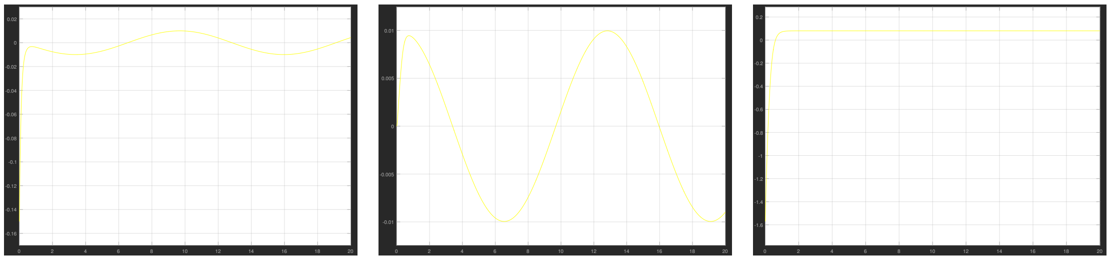

Figure 1.3: Errors in x, y and θ using static decoupling control with $+10\%$ error in robot parameters.

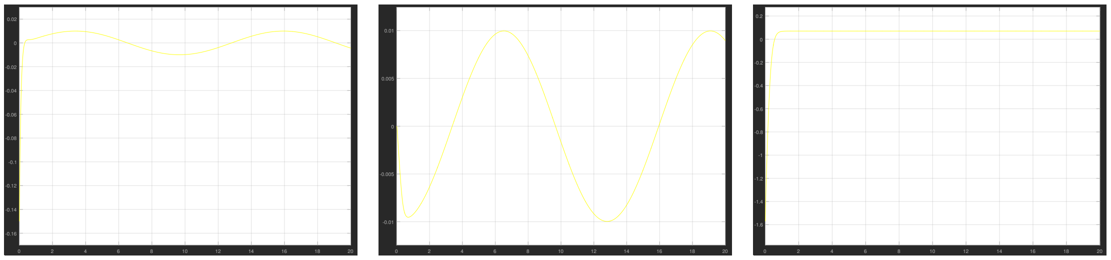

Figure 1.4: Errors in x, y and θ using static decoupling control with -10% error in robot parameters.

From simulations with different than default parameter values it has been observed that the orientation error θ^e is proportional to the radius of the desired trajectory. With default values and a very small distance $d < 0.005m$ in control block following the trajectory becomes erratic, unable to follow it properly. However from a practical point of view such a small distance d enables point tracking in the origin of the mobile frame. Tolerances in the measurement of the physical model alone would most probabk2ly be above such a low value. Gain K_p can be set to a value $5 \div 100$ with default simulation scenario without showing any unusual behavior. Up to a value of 250 the simulation runs fine for $50s$, however at 300 it fails already in the first few seconds. Main determinants for a proper control are the values of distance d and desired angular velocity ω^d. For $\omega^d > 19$, for example $20rad/s$ makes the mobile platform unable to follow the trajectory. Higher ω results in overshooting from the trajectory after final point has been determined. The robot does not return to the final point leaving significant errors in x and y.

In order to study position regulation problem, the desired trajectory (1.14) was followed for a specified amount of time then to be abruptly stopped at a fixed desired final position. The results can be seen in the plot [1]. It was observed that the robot decelerates when it reaches final position, leaves the previous trajectory and then stops. The smaller the value of K_p, the larger the overshoot. Once the robot stopped, it was unable to return to the desired final position.

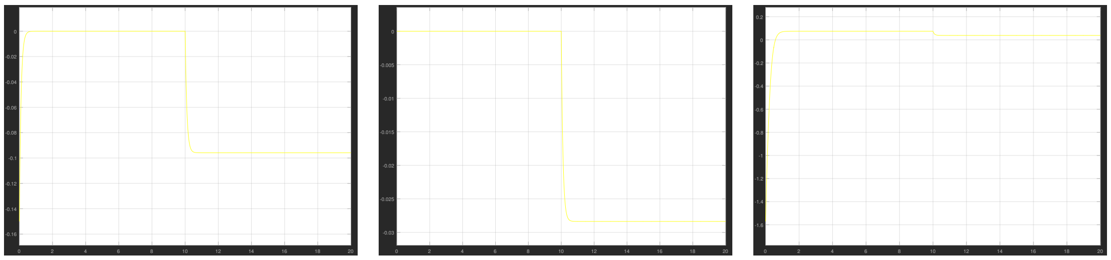

Figure 1.5: Errors in x, y and θ when the robot stops at fixed desired final position using static decoupling control with accurate robot parameters.

It is also important to note that the simulation is based solely on the derived kinematic model of (2,0) robot presented in section *Theoretical model*. The inertia of the robot, which is a part of the dynamic model, has not been taken into account. A more realistic and practical technique to define desired trajectory would include gradually decelerating to the fixed final position in order to prevent overshoot.

Dynamic decoupling control

Theoretical study

For the (2, 0) mobile robot, the position of the robot, O_m can be controlled using dynamic feedback control. We proceed similarly to the static feedback control. Point P is represented as h in the equation (1.27) and in the Simulink model.

$$h = \begin{bmatrix} x \\ y \end{bmatrix} \tag{1.27}$$

In order to implement dynamic feedback control, we find time derivative of h to the order of relative degree of the system. Taking the first time derivative of h to get

$$\dot{h} = \begin{bmatrix} V\cos(\theta) \\ V\sin(\theta) \end{bmatrix} \tag{1.28}$$

$$\ddot{h} = \begin{bmatrix} \dot{V}\cos(\theta) - V\sin(\theta)\cdot\dot{\theta} \\ \dot{V}\sin(\theta) + V\sin(\theta)\cdot\dot{\theta} \end{bmatrix} \tag{1.29}$$

$\ddot{h}$ can now be expressed in terms of new control vector u.

$$\ddot{h} = F(\theta)\cdot\begin{bmatrix} \dot{V} \\ \omega \end{bmatrix} \tag{1.30}$$

where

$$F(\theta) = \begin{bmatrix} \cos(\theta) & -V\sin(\theta) \\ \sin(\theta) & V\sin(\theta) \end{bmatrix} \tag{1.31}$$

$F(\theta)$ can be inversed as long as $V \neq 0$. In order for the error to asymptotically go to zero, we define the equation.

$$\ddot{e}_{(t)} + K_d \cdot \dot{e}_{(t)} + K_p \cdot e(t) = 0 \tag{1.32}$$

Positive values of K_p and K_d ensure that poles of the error are located in the left half of the complex plane guarantying the error is asymptotically stable. The control parameters K_p and K_d have to be tuned for the desired control response, which is user and application dependent. Substituting $e(t)$ with h and h^d results in

$$\dot{h}^d - \dot{h} + K_p \cdot (h^d - h) = 0 \tag{1.33}$$

$$F(\theta) \cdot \begin{bmatrix} \dot{V} \\ \omega \end{bmatrix} = \dot{h}^d + K_p \cdot (h^d - h) \tag{1.34}$$

The right hand side of (1.34) is auxiliary control w. Dynamic state feedback control u can be expressed by following equation (1.35)

$$u = F^{-1}(\theta) \cdot w \tag{1.35}$$

where

$$F^{-1}(\theta) = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\frac{\sin(\theta)}{V} & \frac{\cos(\theta)}{V} \end{bmatrix} \tag{1.36}$$

Validation and simulation

In setting up the Simulink model for dynamic state feedback control, one of the hurdles was that velocity V of the robot could not be zero. This is because $V = 0$ resulted in singularity in control block (1.36). In order to overcome this, the model was assigned initial wheel angular velocities $\dot{\phi}_1$ and $\dot{\phi}_2$ for short interval of time. This ensured that before the control was implemented in the simulation, velocity V was not zero. This was only allowed for $0.1s$ of the simulation time so that the initial position of the robot did not change significantly before the implementation of dynamic feedback control. It must be noted that this step was only taken to perform simulation. For practical control of the robot, it would be wise to gradually accelerate the robot until significant velocity is achieved and then switch to dynamic feedback control. Another alternative would be to implement static state feedback initially and once the robot gains speed the control can revert to dynamic state feedback.

The simulation was tuned to values of $K_p = 20$ and $K_d = 10$. With the dynamic feedback technique the control of the position of the robot is defined by its mobile frame O_m and thus indirectly controlling the orientation. The robot is able to maintain desired ω^d and track desired orientation θ^d of the robot. The plots for posture error can be seen in [1]. The error in θ converges to zero within $2s$. The error in x and y observed were larger than error observed in static. That is due to the initial velocities assigned to dynamic model to overcome the singularity issue. The error in y overshoots to $0.16m$ and then settles to zero. The

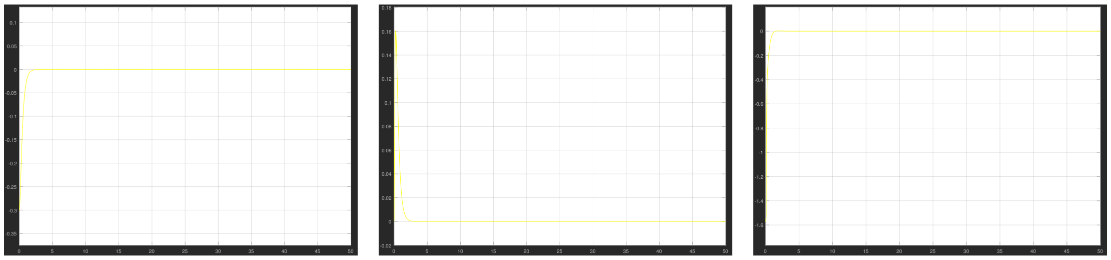
Figure 1.6: Errors in x, y and θ using static decoupling control for accurate robot parameters.

error in x settles to zero within $2.5s$ without overshooting. These errors can be significantly reduced with the appropriate technique to deal with the condition $V = 0$.

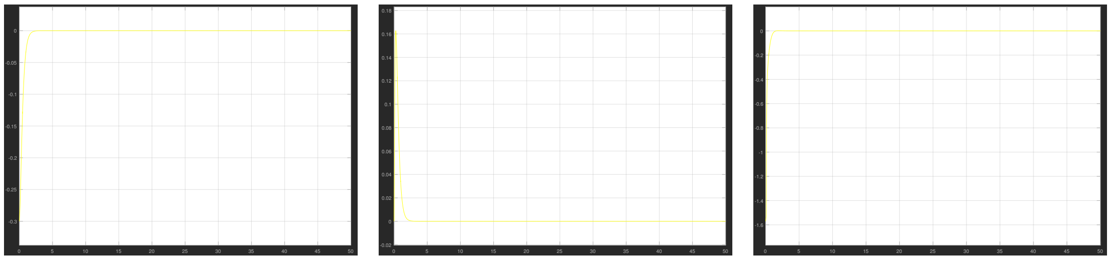
Figure 1.7: Errors in x, y and θ using static decoupling control with $+10\%$ error in robot parameters.

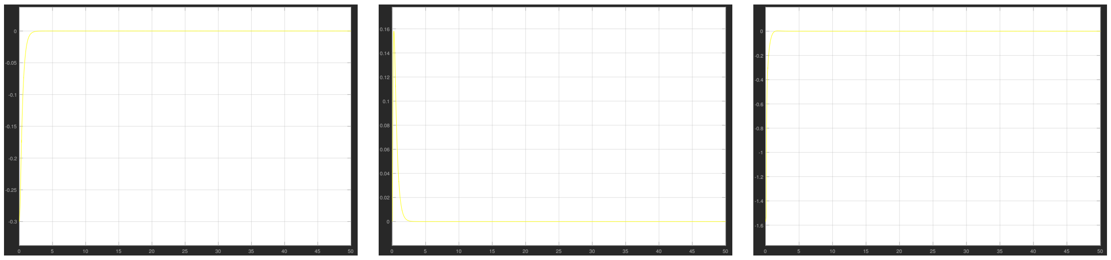
Figure 1.8: Errors in x, y and θ using static decoupling control with -10% error in robot parameters.

The simulation was then carried out with $\pm10\%$ error in the radius of the wheels and the trackgauge in inverse kinematic model, similar to the static feedback technique. For both $+10\%$ and -10% error in the robot parameters, the dynamic feedback control showed robust control, [1] and [1]. The error in x, y and θ, all settled to zero with in 3 unit of simulation time.

Lyapunov control law

Another technique of control the posture ξ of the robot is by using a control-Lyapunov function.

Theoretical study

A control-Lyapunov function is defined such that $V(x)$ is continuously differentiable, positive-definite and has relative degree 1 and its first order time derivative $\dot{V}(x)$ is negative-definite.

$$\frac{\partial \dot{V}(x,u)}{\partial u} \neq 0 \tag{1.37}$$

$$\dot{V}(x) = \frac{\partial V(x)}{\partial x} \cdot f(x,u) < 0 \tag{1.38}$$

First, the posture of the robot is defined as ξ as shown in (1.39)

$$\xi = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \tag{1.39}$$

The desired posture for the robot ξ^d is defined according to (1.14).

$$\xi^d = \begin{bmatrix} R\cos(\omega^d \cdot t) \\ R\sin(\omega^d \cdot t) \\ \omega^d \cdot t + pi/2 \end{bmatrix} \tag{1.40}$$

Posture tracking error ξ^e can be expressed as the difference between the desired and actual posture.

$$\xi^e = \xi^d - \xi \tag{1.41}$$

Posture error in (1.41) is expressed in absolute frame. It can be specified in mobile frame by multiplying it with ${}^mR_0$.

$${}^m\xi^e = \begin{bmatrix} {}^mx^e \\ {}^my^e \\ {}^m\theta^e \end{bmatrix} =^m R_0[{}^0\xi^d -^0 \xi] \tag{1.42}$$

Control error for Lyapunov function is defined by:

$$u^e = u^d - u \tag{1.43}$$

where $u^d = (V^d, \omega^d)^T$ represents the desired control velocities of the mobile robot and $u = (V, \omega)^T$ represents the current control velocities of the mobile robot.

Desired control u^d can be defined by the desired posture of the robot (1.40).

$$u^d = \begin{bmatrix} R \cdot \omega^d \\ \omega^d \end{bmatrix} \tag{1.44}$$

Computing time derivative of ${}^m\xi^e$ gives

$${}^m\dot{\xi}^e = {}^m\dot{R}_0[{}^m\xi^d - {}^m\xi] + {}^mR_0[{}^m\dot{\xi}^d - {}^m\dot{\xi}] \tag{1.45}$$

Equation (1.45) yields

$${}^m\dot{\xi}^e = \begin{bmatrix} 0 & \omega^d & \frac{V^d(\cos({}^m\theta^e)-1)}{{}^m\theta^e} \\ -\omega^d & 0 & \frac{V^d\sin({}^m\theta^e)}{{}^m\theta^e} \\ 0 & 0 & 0 \end{bmatrix} {}^m\xi^e + \begin{bmatrix} 1 & -{}^my^e \\ 0 & {}^mx^e \\ 0 & 1 \end{bmatrix} u^e \tag{1.46}$$

A possible Lyapunov function candidate for the (2,0) robot can be defined as shown in (1.47), (1.48). It can be seen that $V(X)$ is positive-definite by definition for $X \neq 0$.

$$V(X) = \frac{1}{2}({}^m x^{e2} + {}^m y^{e2} + \frac{{}^m\theta^{e2}}{K_y}) \tag{1.47}$$

where

$$X = {}^m\xi^e = \begin{bmatrix} {}^m x^e \\ {}^m y^e \\ {}^m\theta^e \end{bmatrix} \tag{1.48}$$

First order time derivative $\dot{V}(X)$ of the Lyapunov function is expressed in (1.49).

$$\dot{V}(X) = \begin{bmatrix} {}^m x^e & {}^m y^e & \frac{{}^m\theta^e}{K_y} \end{bmatrix} \begin{bmatrix} {}^m\dot{x}_e \\ {}^m\dot{y}_e \\ {}^m\dot{\theta}_e \end{bmatrix} \tag{1.49}$$

For $V(X)$ to be a Lyapunov function, $\dot{V}(X)$ must be less than zero for $X \neq 0$. Substituting (1.46) in (1.49) and expanding it yields the relation shown in (1.50).

$$[V^d \cdot (\cos({}^m\theta^e) - 1) + V^e] \cdot {}^m x^e + [V^d \cdot \frac{\sin({}^m\theta^e)}{{}^m\theta^e} \cdot {}^m y^e + \frac{{}^m\omega^e}{K_y}] \cdot {}^m\theta^e \leqslant 0 \tag{1.50}$$

A possible way to confirm that $\dot{V}(X)$ is negative semi-definite is to make following imposition in (1.50).

$$[V^d \cdot (\cos({}^m\theta^e) - 1) + {}^m V^e] = -K_x \cdot {}^m x^e \tag{1.51}$$

and

$$[V^d \cdot \frac{\sin({}^m\theta^e)}{{}^m\theta^e} \cdot {}^m y^e + \frac{{}^m\omega^e}{K_y}] = -\frac{K_\theta}{K_y} {}^m\theta^e \tag{1.52}$$

By rearranging the above expressions u^e can be defined.

$$u^e = \begin{bmatrix} V^e \\ \omega^e \end{bmatrix} = \begin{bmatrix} -V^d \cdot (\cos({}^m\theta^e) - 1) - K_x \cdot {}^m x^e \\ -K_y V^d \cdot \frac{\sin({}^m\theta^e)}{{}^m\theta^e} \cdot {}^m y^e - K_\theta \cdot {}^m\theta^e \end{bmatrix} \tag{1.53}$$

Substituting (1.54) in (1.43), gives the expression for the control vector u.

$$u = \begin{bmatrix} V \\ \omega \end{bmatrix} = \begin{bmatrix} -V^d \cdot \cos({}^m\theta^e) + K_x \cdot {}^m x^e \\ \omega^d + K_y V^d \cdot \frac{\sin({}^m\theta^e)}{{}^m\theta^e} \cdot {}^m y^e + K_\theta \cdot {}^m\theta^e \end{bmatrix} \tag{1.54}$$

With the control function u as shown in (1.54), $\dot{V}(X)$ is negative-definite for $X \neq 0$. For X belonging to domain $\mathbf{D} \subset \Re^3$ control-Lyapunov function $V(X)$ is asymptotically stable for domain $\mathbf{D}$.

The term $\frac{\sin({}^m\theta^e)}{{}^m\theta^e}$ in (1.54) does not introduce a singularity, since its Taylor series expansion equals $1 - \frac{{}^m\theta^{e(2)}}{3!} + \frac{{}^m\theta^{e(4)}}{5!} \ldots$.

Validation and simulation

Lyapunov-based control directly controls error in the posture of the robot, ergo positional errors x and y and orientation θ.

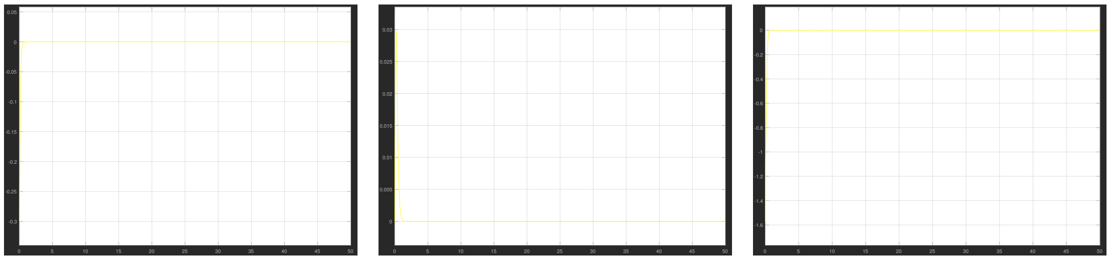

Figure 1.9: Errors in x, y and θ using static decoupling control for accurate robot parameters.

The Simulink model was tuned to values $K_x = 5.4$, $K_y = 11.3$ and $K_\theta = 44.8$. It can be observed from the plots [1] that the static feedback control can control the posture of the robot very well. The error in x position and θ orientation converges to zero within $1.5s$. The error in y overshoots to $0.03m$ before it settles to zero.

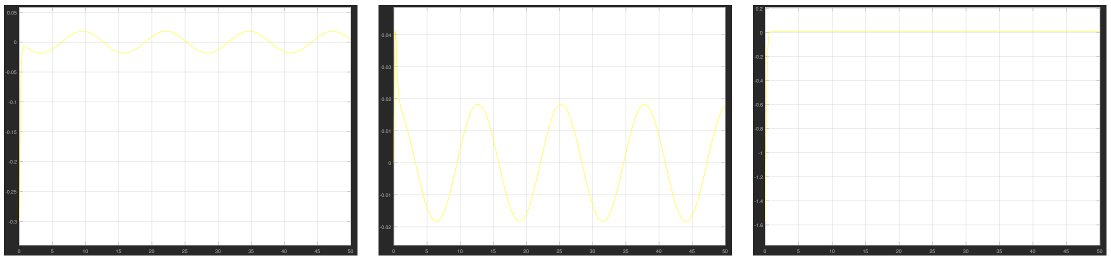

Figure 1.10: Errors in x, y and θ using static decoupling control with $+10\%$ error in robot parameters.

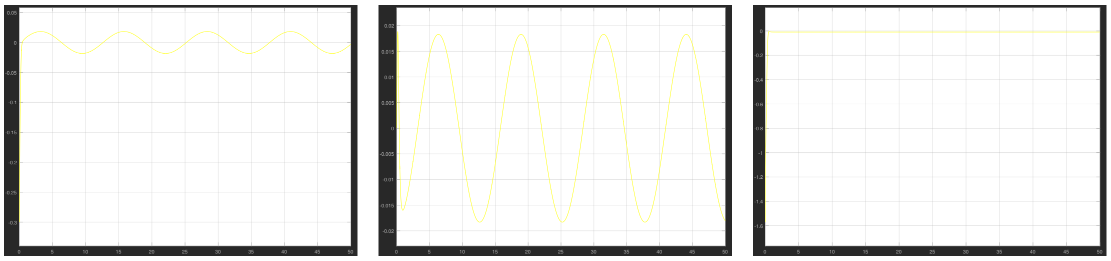

Figure 1.11: Errors in x, y and θ using static decoupling control with -10% error in robot parameters.

The simulation was then carried out with $\pm 10\%$ error in the radii of the wheels and the trackgauge in inverse kinematic model. Lyapunov based control did not offer as robust control as offered by dynamic state feedback. For both $+10\%$ and -10% errors in robot parameters, oscillations within $0.02m$ were observed in the x and y position [1], [1]. Constant errors in orientation were observed for both scenarios with $0.01rad$ for $+10\%$ error and $-0.01rad$ for -10% error in robot parameters.

Chapter 2

Robot type (1, 1)

Direct kinematic model

Modelling of a mobile robot in two dimensional space begins with parametrization of a theoretical model representing the physical mobile platform as portrayed by figure [2.1] for robot type (1, 1). This type of robot is the classical model used to represent a car, where a single central steering wheel is used instead of two.

Theoretical model

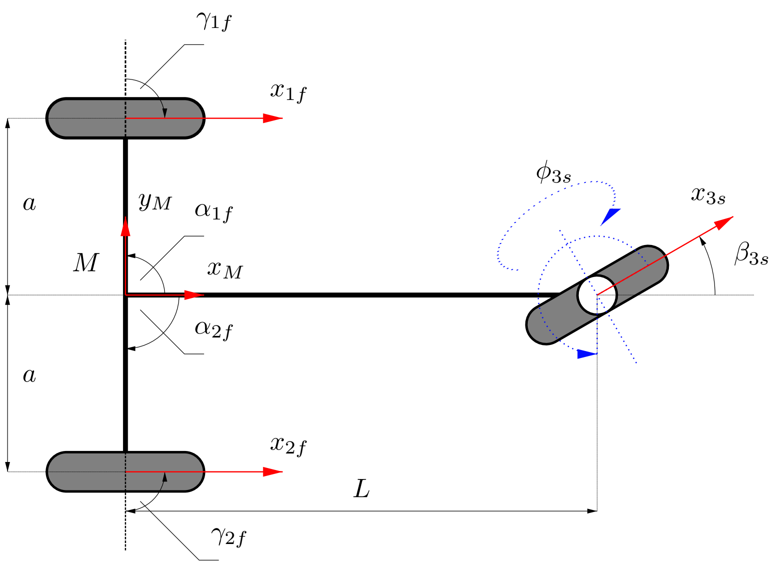

Figure 2.1: Theoretical model of a type 1,1 wheeled mobile robot. (Mobile platform and wheel frames, motorization of wheels.)

Parametrization

After defining the schematic theoretical model the robot frame with the origin in point M is selected considering symmetrical and simplification of calculations. Next wheel assemblies are numbered and their frames, reduced to x_i only, drawn.

Tables 2.1 and 2.2 holds the results of the parametrization process in accordance with figure [2.1].

Table 2.1: Parametrization of robot type (1, 1)

Wheel	L	d	α	β	γ	φ	ψ
1_f	a	0	$\pi/2$	0	$-\pi/2$	φ_{1f}	0
2_f	a	0	$-\pi/2$	0	$\pi/2$	φ_{2f}	0
3_s	L	0	0	β_{3s}	0	φ_{3s}	β_{3s}

Table 2.2: Characteristics of robot type (1, 1)

Degree of $0 \leqslant \delta_s + \delta_f \leqslant 2$	mobility δ_m $2 \leqslant \delta_m + \delta_s \leqslant 3$	steerability δ_s $0 \leqslant \delta_s \leqslant 2$	fixed ICR δ_f $0 \leqslant \delta_f \leqslant 1$	maneuverability $\delta_M = \delta_m + \delta_s$
	1	1	1	2
Wheels	fixed N_f	steerable N_s	castor N_c	total N_t
	2	1	0	3

The parametrization process ends with defining the configuration vector q:

$$q = \begin{bmatrix} x \\ y \\ \theta \\ \beta_3 s \\ \varphi_1 f \\ \varphi_2 f \\ \varphi_3 s \end{bmatrix} = \begin{bmatrix} \xi \\ \beta_3 s \\ \varphi_1 f \\ \varphi_2 f \\ \varphi_3 s \end{bmatrix} \tag{2.1}$$

Constraints

Work hypotheses include nonholonomic constraints, also known as pure rolling constraints, for wheels:

- no slipping – tangential speed $v_t = 0$,
- no skidding – normal speed $v_n = 0$,
- rotational slipping acceptable – rotational speed ω_z.

For a generic wheel those constraints can be expressed in equation (2.2).

$$\begin{bmatrix} v_t \\ v_n \end{bmatrix} = \begin{bmatrix} cos(\psi) & sin(\psi) & d \cdot sin(\gamma) + L \cdot sin(\beta + \gamma) \\ -sin(\psi) & cos(\psi) & d \cdot cos(\gamma) + L \cdot cos(\beta + \gamma) \end{bmatrix} \begin{bmatrix} cos(\theta) & sin(\theta) & 0 \\ -sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \dot{\xi} + \begin{bmatrix} -r \\ 0 \end{bmatrix} \dot{\varphi} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{2.2}$$

Fixed wheels

Following the equation (2.2) results in (2.3), (2.4) for fixed wheel x_{1f} and in (2.5) for fixed wheel x_{2f}.

$$\begin{bmatrix} v_t \\ v_n \end{bmatrix} = \begin{bmatrix} cos(0) & sin(0) & L \cdot sin(-\pi/2) \\ -sin(0) & cos(0) & L \cdot cos(-\pi/2) \end{bmatrix} {}^m\Omega_0 \cdot \dot{\xi} + \begin{bmatrix} -r \\ 0 \end{bmatrix} \dot{\varphi}_{1f} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{2.3}$$

$$\begin{cases} r\dot{\varphi}_{1f} & = & {}^m\dot{x} - a \cdot {}^m\dot{\theta} \\ {}^m\dot{y} & = & 0 \end{cases} \tag{2.4}$$

$$\left\{\begin{array}{rcl} r\dot{\varphi}_{2f} & = & {}^m\dot{x} + a \cdot {}^m\dot{\theta} \\ {}^m\dot{y} & = & 0 \end{array}\right. \tag{2.5}$$

Steerable wheels

Following the equation (2.2) results in (2.6) and (2.7) for steerable wheel x_{3s}.

$$\begin{bmatrix} v_t \\ v_n \end{bmatrix} = \begin{bmatrix} cos(0) & sin(0) & L \cdot sin(-\pi/2) \\ -sin(0) & cos(0) & L \cdot cos(-\pi/2) \end{bmatrix} {}^m\Omega_0 \cdot \dot{\xi} + \begin{bmatrix} -r \\ 0 \end{bmatrix} \dot{\varphi}_{1f} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{2.6}$$

$$\left\{\begin{array}{rcl} r\dot{\varphi}_{3s} & = & {}^m\dot{x}cos(\beta_{3s}) + {}^m\dot{y}sin(\beta_{3s}) + {}^m\dot{\theta}L \cdot sin(\beta_{3s}) \\ 0 & = & {}^m\dot{y}cos(\beta_{3s}) - {}^m\dot{x}sin(\beta_{3s}) + {}^m\dot{\theta}L \cdot cos(\beta_{3s}) \end{array}\right. \tag{2.7}$$

Pure rolling constraints

From sections 2 and 2 the matrix form for all wheels defining the non slipping

$$\underbrace{\begin{bmatrix} 1 & 0 & -a \\ 1 & 0 & a \\ cos(\beta_{3s}) & sin(\beta_{3s}) & L \cdot sin(\beta_{3s}) \end{bmatrix}}_{J_1} {}^m\dot{\xi}_0 + \underbrace{\begin{bmatrix} -r & 0 & 0 \\ 0 & -r & 0 \\ 0 & 0 & -r \end{bmatrix}}_{J_2} \begin{bmatrix} \dot{\varphi}_{1f} \\ \dot{\varphi}_{2f} \\ \dot{\varphi}_{3s} \end{bmatrix} = 0 \tag{2.8}$$

and non skidding constraints

$$\underbrace{\begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ -sin(\beta_{3s}) & cos(\beta_{3s}) & L \cdot cos(\beta_{3s}) \end{bmatrix}}_{C_1} {}^m\dot{\xi}_0 = 0 \tag{2.9}$$

leads to definition of the rotation speed vector of the mobile platform $\dot{\varphi}$ in equation (2.10).

$$\dot{\varphi} = \frac{1}{r}\left[J_1 {}^m\dot{\xi}_0\right] \tag{2.10}$$

Posture kinematic model

Since only non skidding constraints of fixed and steering wheels can limit the mobility of the robot, and in case of this (1, 1) robot those are the only used types of wheels, $C_1 = C_1^*(\beta_{3s})$ and therefore twist in reference to the mobile platform ${}^m\dot{\xi}$ can be decomposed on a base of kernel of C_1^* as portrayed in (2.11).

$${}^m\dot{\xi} = {}^m\Omega_0 \cdot \dot{\xi} = \Sigma(\beta_{3s})u_m \quad \epsilon \; ker(C_1^*(\beta_{3s})) \tag{2.11}$$

Since the robot has a steerable wheel, it's angular position must be controlled, therefore the base of kernel Σ is a function of the steerable and not a constant. Out of several possible bases of kernel of C_1^* presented in (2.12).

$$\Sigma_1 = \begin{bmatrix} L/\operatorname{tg}(\beta_{3s}) \\ 0 \\ 1 \end{bmatrix}, \quad \Sigma_2 = \begin{bmatrix} 1 \\ 0 \\ \operatorname{tg}(\beta_{3s})/L \end{bmatrix}, \quad \Sigma_3 = \begin{bmatrix} L \cdot cos(\beta_{3s}) \\ 0 \\ sin(\beta_{3s}) \end{bmatrix} \tag{2.12}$$

Σ_1 introduces singularities at $k\frac{\pi}{2}$ and at 0, Σ_1 introduces singularities at $k\frac{\pi}{2}$ and Σ_3 made the control vector u computations complex. Σ_2 was chosen as the base of kernel of C_1^* with the control vector u defined as shown in (2.15) in accordance to derivation (2.13) and (2.14) in new state vector $\dot{z}$, which is the posture kinematic model of the selected mobile robot of type (1, 1).

$$\begin{cases} {}^0\dot{\xi} & = & {}^0\Omega_m(\theta) \cdot \Sigma(\beta_{3s}) \cdot u_m \\ \dot{\beta}_{3s} & = & u_s \end{cases} \tag{2.13}$$

$$\dot{z} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\beta_{3s}} \end{bmatrix} = B(z) \cdot u = \begin{bmatrix} {}^0\Omega_m(\theta) \cdot \Sigma(\beta_{3s}) & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} V \\ \dot{\beta}_{3s} \end{bmatrix} \tag{2.14}$$

$$u = \begin{bmatrix} u_m \\ u_s \end{bmatrix} = \begin{bmatrix} V \\ \dot{\beta}_{3s} \end{bmatrix} \tag{2.15}$$

Configuration kinematic model

In order to motorize the robot the configuration kinematic model needs to be determined. It is an expansion of the direct kinematic model (2.14), holding the state of all wheel assemblies. From the control vector q (2.1) the configuration kinematic model (2.16) is derived with respect to the fixed frame. Since there are no castor wheels in robot (1, 1) and only one steerable wheel, the matrix $D(\beta c)$ does not exist and $E(\beta s, \beta c)$ only depends on that steerable wheel x_{3s}.

$$\dot{q} = \begin{bmatrix} \dot{\xi} \\ \dot{\beta}_s \\ \dot{\beta}_c \\ \dot{\varphi} \end{bmatrix} = S(q) \cdot u = \begin{bmatrix} {}^0\Omega_m(\theta) \cdot \Sigma(\beta_{3s}) & 0 \\ 0 & I \\ E(\beta_{3s}) \cdot \Sigma(\beta_{3s}) & 0 \end{bmatrix} \begin{bmatrix} u_m \\ u_s \end{bmatrix} \tag{2.16}$$

The matrix $E(\beta_{3s})$ is defined as equation (2.17) with substitution of matrix Σ with respect to the fixed frame results in the configuration kinematic model $\dot{q}$ (2.18).

$$E(\beta_{3s}) = -J_2^{-1} \cdot J_1(\beta_s, \beta_c) = \frac{1}{r} \begin{bmatrix} 1 & 0 & -a \\ 1 & 0 & a \\ cos(\beta_{3s}) & sin(\beta_{3s}) & L \cdot sin(\beta_{3s}) \end{bmatrix} \tag{2.17}$$

$$\dot{q} = \begin{bmatrix} cos(\theta) & 0 \\ sin(\theta) & 0 \\ \operatorname{tg}(\beta_{3s})/L & 0 \\ 0 & 1 \\ \frac{1}{r}\begin{pmatrix} 1 - a \cdot \operatorname{tg}(\beta_{3s})/L \\ 1 + a \cdot \operatorname{tg}(\beta_{3s})/L \\ cos(\beta_{3s}) + sin(\beta_{3s}) \cdot \operatorname{tg}(\beta_{3s}) \end{pmatrix} & 0 \end{bmatrix} \begin{bmatrix} V \\ \dot{\beta}_{3s} \end{bmatrix} \tag{2.18}$$

Motorization of wheels

From the configuration kinematic model $\dot{q}$ (2.18) the matrices $D(\beta c)$ and $E(\beta s, \beta c)$ need to be extracted for the motorization process, however since there are no castor wheels in robot type (1, 1) and therefore no $D(\beta c)$ matrix, only matrix $E(\beta s)$ is extracted. This results in equation (2.19), representing only $\dot{\varphi}$ in this case, for all wheels.

$$\dot{\varphi} = \begin{bmatrix} \dot{\varphi}_{1f} \\ \dot{\varphi}_{2f} \\ \dot{\varphi}_{3s} \end{bmatrix} = F(\beta_s) \cdot u_m = \frac{V}{r} \begin{bmatrix} 1 - a \cdot \text{tg}(\beta_{3s})/L \\ 1 + a \cdot \text{tg}(\beta_{3s})/L \\ cos(\beta_{3s}) + sin(\beta_{3s}) \cdot \text{tg}(\beta_{3s}) \end{bmatrix} \tag{2.19}$$

The choice of actuators of a wheeled mobile robot depends on many factors including physical layout of the robot, cost, robustness and reliability, user and application specifics as well as space and cargo load limitations and others. Usually because of simplicity of the construction process and availability, fixed wheels are of lowest cost, easy to exchange or repair, robust and reliable. Cargo wheels have limitations with load weight put on the orientation joint and fully motorized steerable wheels can get mechanically quite complex.

For the presented mobile robot of type (1, 1) the full motorization of the steerable wheel x_{3s} has been chosen, making fixed wheels passive, as presented in figure [2.1]. The decision is mostly motivated by practical aspects like power and signal cables wiring and symmetry of weight distribution. The steerable must have ground contact in order for the mobile robot to move properly, irrelevant of the fact which wheel assemblies are motorized.

For the direct kinematic model in Simulink, integrator block is used to compute orientation of the steering wheel β_{3s} from $\dot{\beta_{3s}}$. Using equation (2.18) and (2.19), cartesian velocities of the mobile robot V and ω are then computed from $\dot{\varphi}_{3s}$ and β_{3s} (2.20), (2.21).

$$V = r\dot{\varphi_{3s}} \cos \beta_{3s} \tag{2.20}$$

$$\omega = \frac{r}{L} \dot{\varphi_{3s}} \sin \beta_{3s} \tag{2.21}$$

The inverse kinematic block computes $\dot{\varphi}_{3s}$ from V and β_{3s} (2.22)

$$\dot{\varphi_{3s}} = \frac{V}{r \cos \beta_{3s}} \tag{2.22}$$

Static decoupling control

Analogous to the (2, 0) robot static state feedback has been derived and implemented.

Theoretical study

Static feedback control can not be used to control position of the robot, since it leads to singularity. However, a point attached to the plane of the steering wheel, except its center can be controlled. This point P' is represented as h in the Simunlink model. For the implemented (1,1) robot, L represents the distance between O_m and steering wheel in the direction of x_M [2.1]. d represents the distance of point P' from the center of steering wheel.

$$h = \begin{bmatrix} x + L\cos(\theta) + d\cos(\theta + \beta_{3s}) \\ y + L\sin(\theta) + d\cos(\theta + \beta_{3s}) \end{bmatrix} \tag{2.23}$$

Coordinates x and y of point P' can be controlled using static state feedback and are represented as subsystem z_1, whereas the orientation θ and steering angle β_{3s} cannot and are represented as subsystem z_2. Same desired trajectory is defined for the robot as was defined for (2,0) robot. The derivation process for feedback control is same as defined in section 1and is thus not repeated here.

The control vector for (1,1) robot is different and consists of V and $\dot{\beta_{3s}}$.

$$u = \begin{bmatrix} V \\ \dot{\beta_{3s}} \end{bmatrix} \tag{2.24}$$

$\dot{z}_1$ can be expressed in terms of control vector u.

$$\dot{z}_1 = K(\theta) \cdot \begin{bmatrix} V \\ \dot{\beta_{3s}} \end{bmatrix} \tag{2.25}$$

where

$$K(\theta) = \begin{bmatrix} \cos(\theta) - \mathrm{tg}(\beta_{3s}) \sin(\theta) - \frac{d}{L} \mathrm{tg}(\beta_{3s}) \sin(\theta + \beta_{3s}) & -d \sin(\theta + \beta_{3s}) \\ \sin(\theta) + \mathrm{tg}(\beta_{3s}) \cos(\theta) + \frac{d}{L} \mathrm{tg}(\beta_{3s}) \cos(\theta + \beta_{3s}) & d \cos(\theta + \beta_{3s}) \end{bmatrix} \tag{2.26}$$

Computing $K^{-1}(\theta)$ yields equation (2.27).

$$K^{-1}(\theta) = \frac{1}{|K(\theta)|} \cdot \begin{bmatrix} d \cos(\theta + \beta_{3s}) & d \sin(\theta + \beta_{3s}) \\ -\sin(\theta) - \mathrm{tg}(\beta_{3s}) \left(\cos(\theta) - \frac{d}{L} \cos(\theta + \beta_{3s})\right) & \cos(\theta) - \mathrm{tg}(\beta_{3s}) \left(\sin(\theta) - \frac{d}{L} \sin(\theta + \beta_{3s})\right) \end{bmatrix} \tag{2.27}$$

where

$$|K(\theta)| = d \cdot [\sin(\theta + \beta_{3s}) \sin\theta + \cos(\theta + \beta_{3s}) \cos\theta - \mathrm{tg}(\beta_{3s}) (\cos(\theta + \beta_{3s}) \sin\theta - \sin(\theta + \beta_{3s}) \cos\theta)] \tag{2.28}$$

It can be seen that determinant of $K(\theta)$ in (2.28), $|K(\theta)|$ equals zero when $d = 0$.

Validation and simulation

Similarly to the type (2, 0) robot there is no direct control of the robots position with static feedback control, however a point attached to the plane of the steering wheel with the exception of its center can be controlled. This results in an indirect control of the position and orientation of the mobile platform with a constant error θ^e.

Simulation was performed with the gain $K_p = 5$. The plots for trajectory and errors are shown in [2] and [2]. The error in y axis of the mobile frame is oscillating at a negligible level. The error in x is converging to zero in a very short time, resulting in a very well control of trajectory of point P'. The orientation error reaches a constant value of $0.14 rad$ being unable to converge to zero.

Changing the parameters of the simulation with $\pm 10\%$ error in the radii of the wheels and the trackgauge in inverse kinematic model resulted in oscillations around $\pm 0.02 m$ of positional errors x and y, while the orientation error behaved without noticeable difference converging a constant value of $0.14 rad$. The results are shown in plots [2] and [2].

When the trajectory generation stops the mobile robot runs out of the trajectory in a proportional way to the desired angular velocity ω^d leaving significant positional errors in x and y. However the orientation

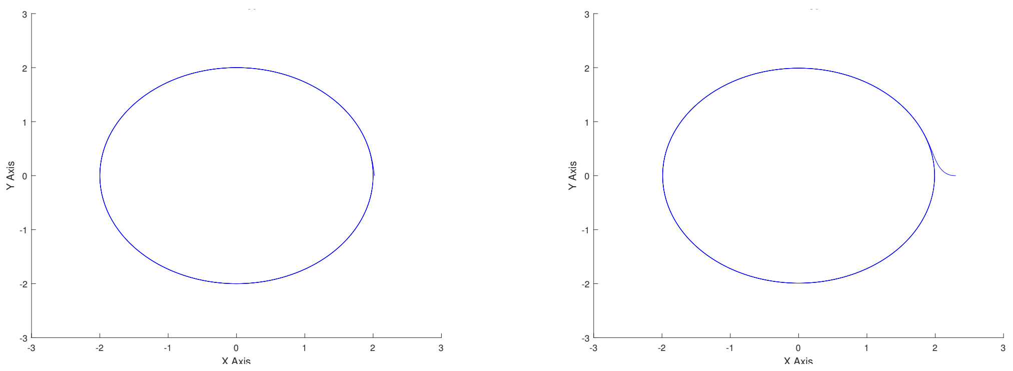

Figure 2.2: Trajectory of point P' and robot position using static decoupling control for accurate robot parameters.

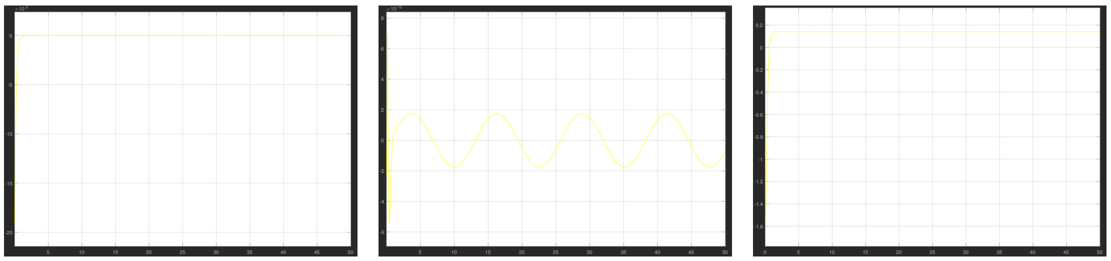

Figure 2.3: Errors in x, y and θ using static decoupling control for accurate robot parameters.

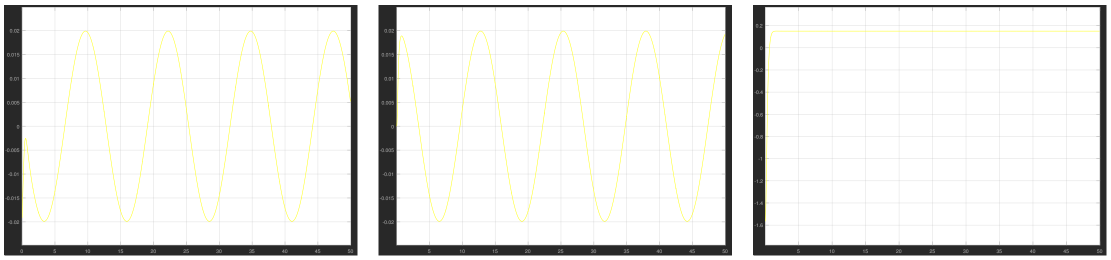

Figure 2.4: Errors in x, y and θ using static decoupling control with +10% error in robot parameters.

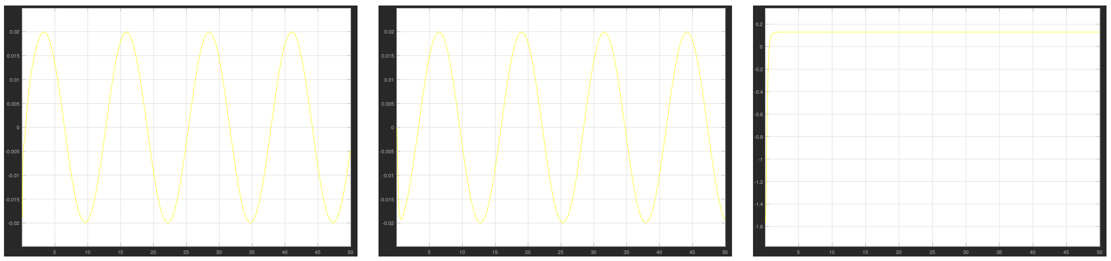

Figure 2.5: Errors in x, y and θ using static decoupling control with −10% error in robot parameters.

error θ^e gets significantly corrected after final desired point has been determined. The plots for the errors are shown in [2]. This correction is inversely proportional to the gain K_p.

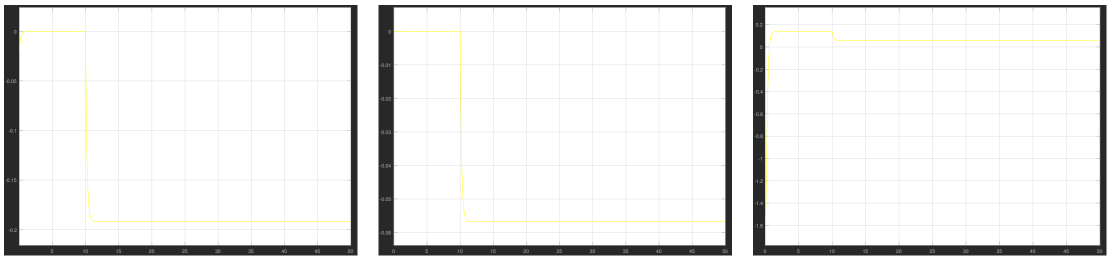

Figure 2.6: Errors in x, y and θ when the robot stops at fixed desired final position using static decoupling control with accurate robot parameters.

Again a more realistic and practical approach would be to let the desired trajectory decelerate slowly to a final position accounting the inertial effects.

Dynamic decoupling control

The position of the type (1, 1) robot O_m can be controlled using dynamic feedback control.

Theoretical study

The details for derivation and developing model for dynamic feedback control can be reviewed in section *1. Theoretical study.* Point O_m is represented as h in the Simulink model (2.29).

$$h = \begin{bmatrix} x \\ y \end{bmatrix} \tag{2.29}$$

Second order time derivative of h can be represented in terms of $\dot{V}$ and $\mathrm{tg}(\beta_{3s})$ (2.30).

$$\ddot{h} = F(\theta) \cdot \begin{bmatrix} \dot{V} \\ tan(\beta_{3s}) \end{bmatrix} \tag{2.30}$$

where

$$F(\theta) = \begin{bmatrix} \cos(\theta) & -\frac{V^2}{L}\sin(\theta) \\ \sin(\theta) & \frac{V^2}{L}\sin(\theta) \end{bmatrix} \tag{2.31}$$

Computing inverse of $F(\theta)$ yields (2.32)

$$F^{-1}(\theta) = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\frac{L}{V^2}\sin(\theta) & \frac{L}{V^2}\cos(\theta) \end{bmatrix} \tag{2.32}$$

Similar to the case of (2,0) robot, $F(\theta)$ can only be inversed when $V \neq 0$.

Validation and simulation

As with the (2, 0) robot, the velocity introduced singularity has been dealt with using initial values for the angular velocity of the steering wheel $\dot{\varphi}_{3s}$ and its initial orientation β_{3s}. Details of this problem have been discussed in *1. Validation and simulation*.

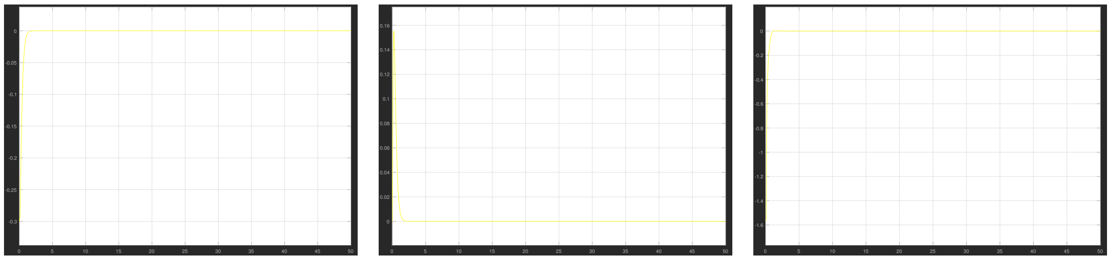

Figure 2.7: Errors in x, y and θ using static decoupling control for accurate robot parameters.

Simulation has been performed with tuned values of gains $K_p = 25$ and $K_d = 10$. The implemented dynamic feedback technique allows for direct control of the position of the mobile frame and indirectly its orientation.

In the default scenario all errors, in position and orientation, converge to zero [2]. The error in y overshoots to $0.16m$ due to commutation between fixed values and actual trajectory generation. Reduction of the errors is expected to occur when using an appropriate technique for the singularity $V = 0$ issue.

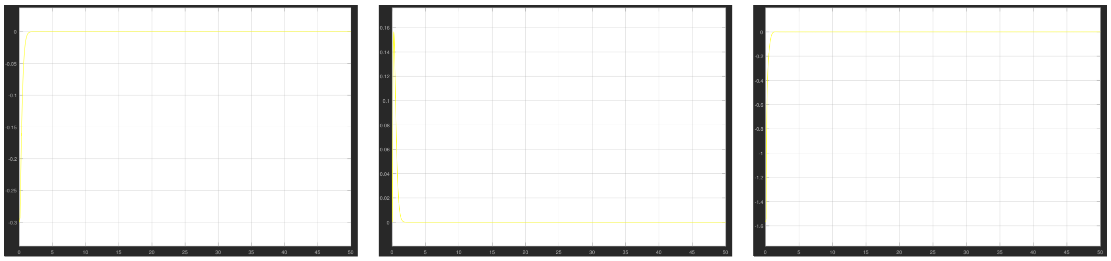

Figure 2.8: Errors in x, y and θ using static decoupling control with $+10\%$ error in robot parameters.

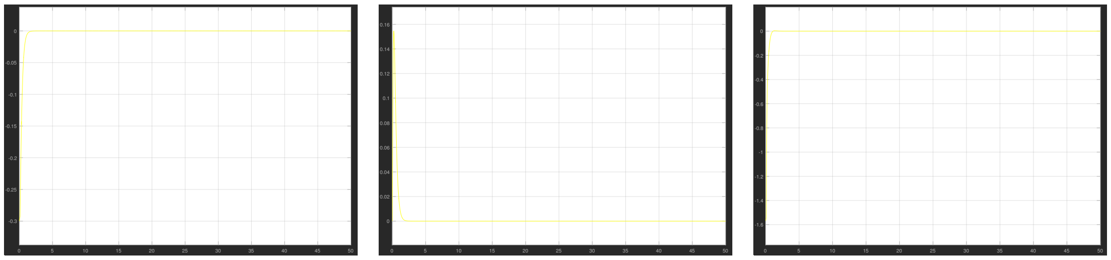

Figure 2.9: Errors in x, y and θ using static decoupling control with -10% error in robot parameters.

Simulation performed with $\pm 10\%$ errors in physical parameters of the robot does not influence the tracking performance in any significant way, as can be seen in the plots [2] and [2]. The control is robust even with abstractly high values for radii of the wheels going up to $1.5m$ in the inverse kinematic model. Depending on the magnitude of differences between control and physical models the errors converge relatively quickly. In the default scenario all converge to zero below $3s$ with a slight increase of time for the scenarios including errors.

Lyapunov control law

Theoretical study

For the (1,1) robot the same Lyapunov function (2.33) was used that was developed for (2,0) robot as shown in (1.47) on page 10 to control the posture of the robot.

$$V(X) = \frac{1}{2}({}^{m}x^{e2} + {}^{m}y^{e2} + \frac{{}^{m}\theta^{e2}}{K_y}) \tag{2.33}$$

This is because the position and orientation of the robot can be expressed in terms of V as previously shown in (2.14) and expressed again in (2.34). For the given problem statement, no desired β_{3s} is assigned.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ \text{tg}(\beta_{3s})/L \end{bmatrix} \cdot V \tag{2.34}$$

$$\tag{2.35}$$

In order to use the same Lyapunov function a minor addition is required. Rotation angle β_{3s} is computed from the Lyapunov control: V and ω. This can simply be derived from (2.34)

$$\beta_{3s} = tan^{-1}(\frac{L\omega}{V}) \tag{2.36}$$

Validation and simulation

Since the same Lyapunov function (2.33) was used to control errors in the posture of the robot, the Simulink model was used with the same tuned values as those used with type (2,0) robot, that is $K_x = 5.4$, $K_y = 11.3$ and $K_\theta = 44.8$. The main difference in the simulation included changes to the inverse kinematic model to accustom the different control vector u, since the control was conducted through the steerable wheel alone in this case.

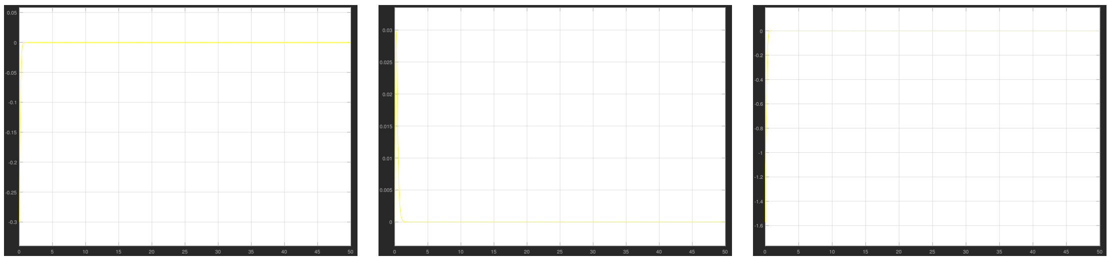

Figure 2.10: Errors in x, y and θ using static decoupling control for accurate robot parameters.

From the generated plots [2] it can be determined that all errors converge to zero relatively quickly in the default scenario, not exceeding $1.5s$. Posture of the robot is controlled smoothly according to the desired trajectory. The error in y position overshoots again to $0.03m$ before it settles to zero.

Introducing $\pm10\%$ errors in the radii of the wheels and the trackgauge in the inverse kinematic model results in relatively robust control. Very similar results, [2] and [**??**], were achieved in comparison to type (2,0) robot. Again the Lyapunov control did not offer as robust control as with dynamic state feedback. Oscillations within $0.02m$ were observed in the x and y positions. The orientation error was at approximately $0.01rad$

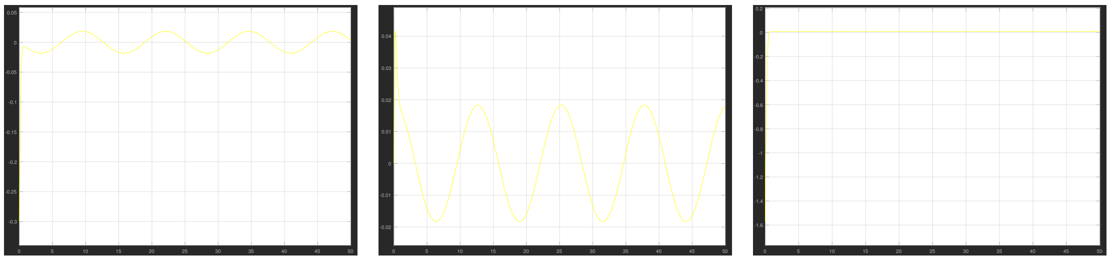

Figure 2.11: Errors in x, y and θ using static decoupling control with $+10\%$ error in robot parameters.

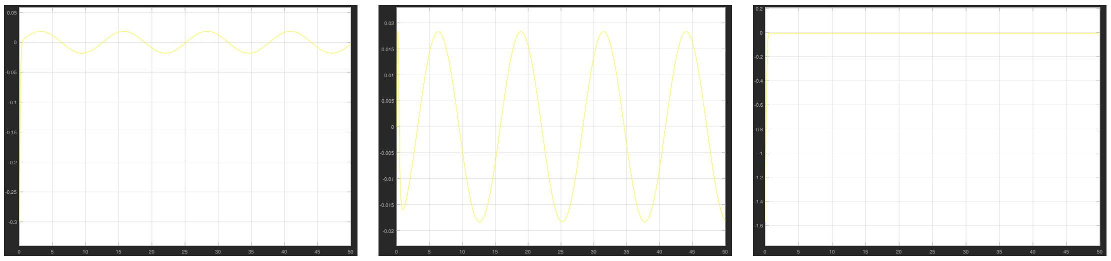

Figure 2.12: Errors in x, y and θ using static decoupling control with -10% error in robot parameters.

for $+10\%$ error and $-0.01 rad$ for -10%. The oscillations in position are proportional to the error value as well as is the orientation converging to a constant value.

Conclusions

From the validation and simulation parts in each section for both robot types (2, 0) and (1, 1) an overview of performance of static and dynamic decoupling control as well as Lyapunov control can be determined.

Analyzing the robot position and errors graphs concludes that all presented control techniques have their strong and weak aspects. All perform well in the ideal scenario, which includes no errors in determining physical parameters of the robot like the wheel radii and no change due to wear or other internal or external influences. There is a significant difference once the parameters differ from the used robot model. This is critical for mobile robots in general, since many parameters undergo change in time, even when optimal working condition are guaranteed. But even more important is the ability to deal with random events like a damaged wheel or axis significantly changing the physical model.

Static feedback control provides indirect control of position and orientation, similar to pulling a cart. There are negligible errors in position in the default scenario, while orientation error evolves to a constant value. Errors in parameters of the robot result in position and orientation error that are proportional and of the same behavior as in default scenario. The implemented control does not reach a final position determined before the end of the simulation. However this was done abruptly, which is not practical, alone because of inertia. A desired trajectory generation taking into account deceleration towards the final point is expected to work with similar results as in the ideal case. One aspect worth noting about the static feedback control is that the distance of the tracked point to the origin of the mobile platform can be very small resulting in pseudo tracking of the robot using only its frame.

The implemented dynamic decoupling control allowed for direct control of position of the robot and therefore indirectly its orientation. Dynamic feedback control introduced a singularity problem around linear velocity $V = 0$. Simulations were performed by switching between a set of fixed and generated values of the outputs of the inverse kinematic model. This is a crude but sufficient solution for performed simulations. A more robust solution should be implemented to overcome the singularity aspect, like switching between static and dynamic feedback control based on the velocity. The implemented control performed very well, positional and orientation errors converged to zero within relatively small time around $2s$. Impressive was the almost identical performance with errors in radii and trackgauge in the inverse kinematic model for the type (1, 1) robot. The dynamic feedback control handled even extraordinarily high errors of order of magnitude of $1.5m$. The (2, 0) model handled the errors worse, resulting in still relatively small oscillations in positional errors and a constant error in orientation. Even though both robot types have a degree of maneuverability $\delta_M = 2$, this behavior might be related to the fully motorized steerable wheel. For the robot type (1, 1) such performance ensures that the dynamic control is well designed to deal with changing systems, once the singularity problems have been overcome. During the experimentation phase it has been established that depending on the initial orientation the platform might move in reverse.

Lyapunov control ensures direct control of the posture and results in what can be considered most smooth transition of the mobile platform for the used candidate function (1.47) to the desired trajectory. This can be a significant aspect if there are restriction in the physical robot. Also the Lyapunov control function allows for more direct customization to accustom for specific application and user needs. For example limits on the angular velocity could be imposed to prevent from possibly dangerous maneuvers when carrying heavy loads. Lyapunov control performs relatively well with errors in robots parameters in the inverse kinematic model, however the control was not as robust in any robot as with dynamic feedback control with robot type (1, 1). Similar oscillations have been achieved with both robot types in positional errors and evolution of the orientation error to a constant value as with dynamic control and robot type (2, 0). Posture errors are proportional to the parameter errors. Is it speculated that the same Lyapunov control function can be used for similar problem statements with only modifications necessary to accustom different control vectors u. This could result in a significant reduction of workload dealing with control of robots that can change their type depending on the situation. Possibly even whole families of different robot types could be controlled using the same or similar functions.

Concluding presented results, it can be stated that dynamic control is very well suited for certain types of robot where changes in the physical model are likely to happen or occur constantly over time. Static feedback could be used for the dynamic control when reaching singularity, otherwise its performance is not convincing. Lyapunov approach offers a similar performance to the dynamic feedback without the singularity problem and is the only presented control scheme that acts directly on the posture of the mobile robot.

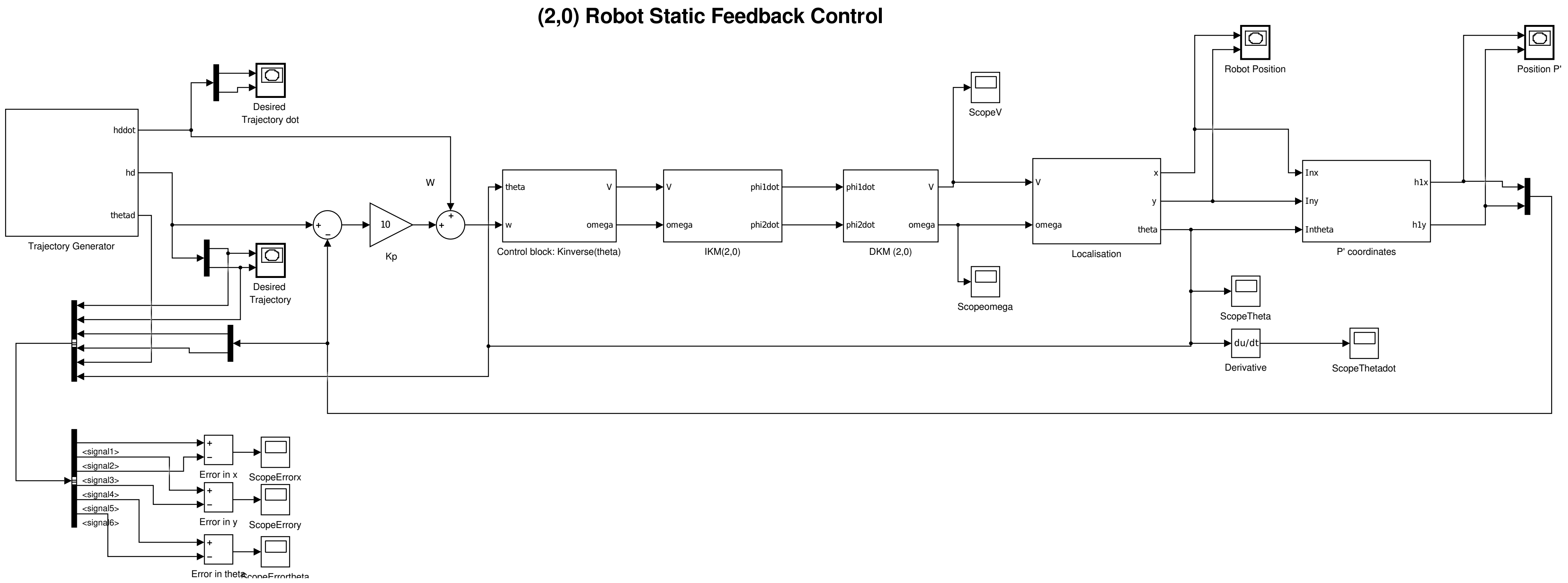

Figure 1: Simulink static decoupling control schematic for the type (2, 0) robot.

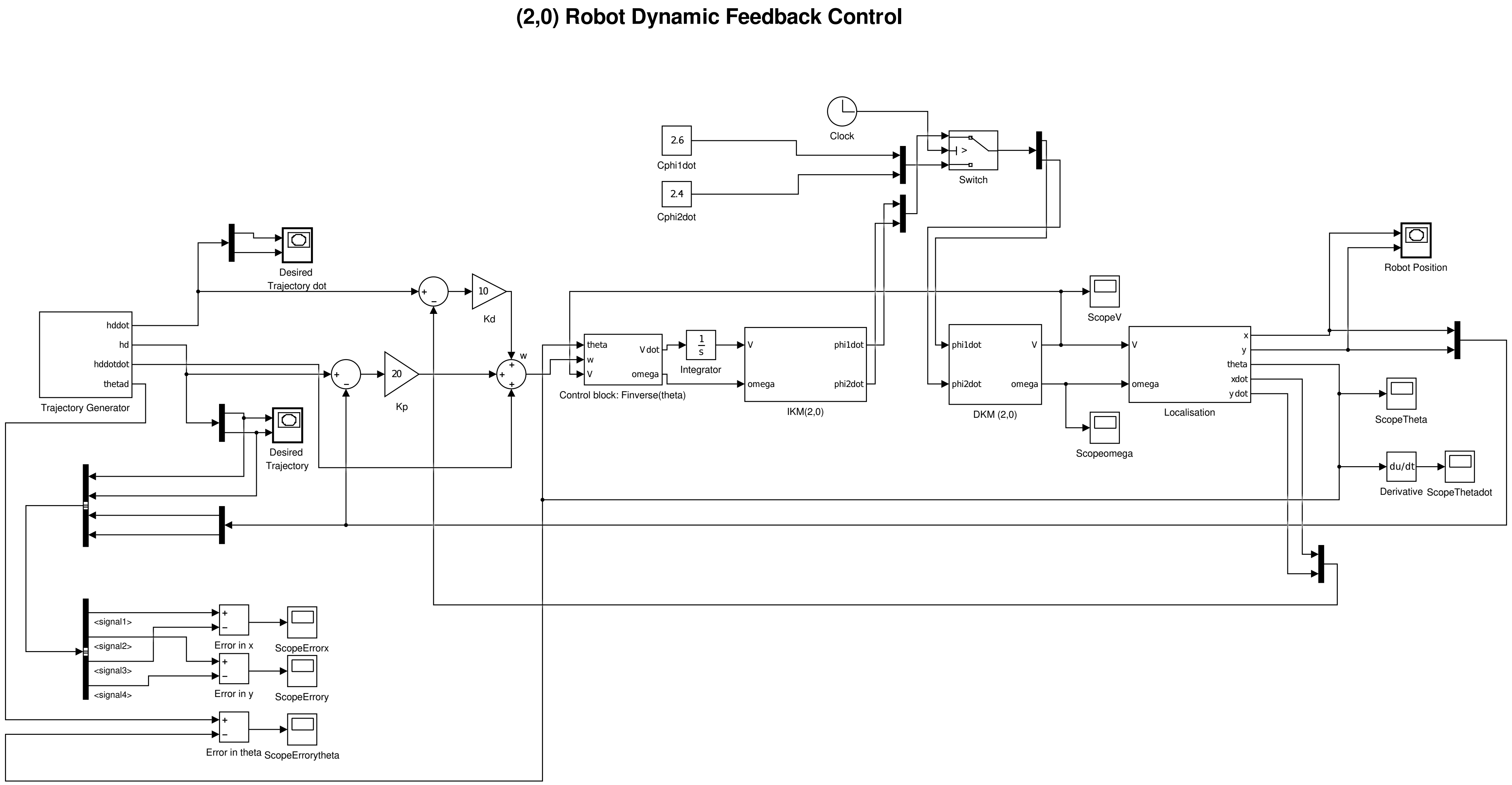

Figure 2: Simulink dynamic decoupling control schematic for the type (2, 0) robot.

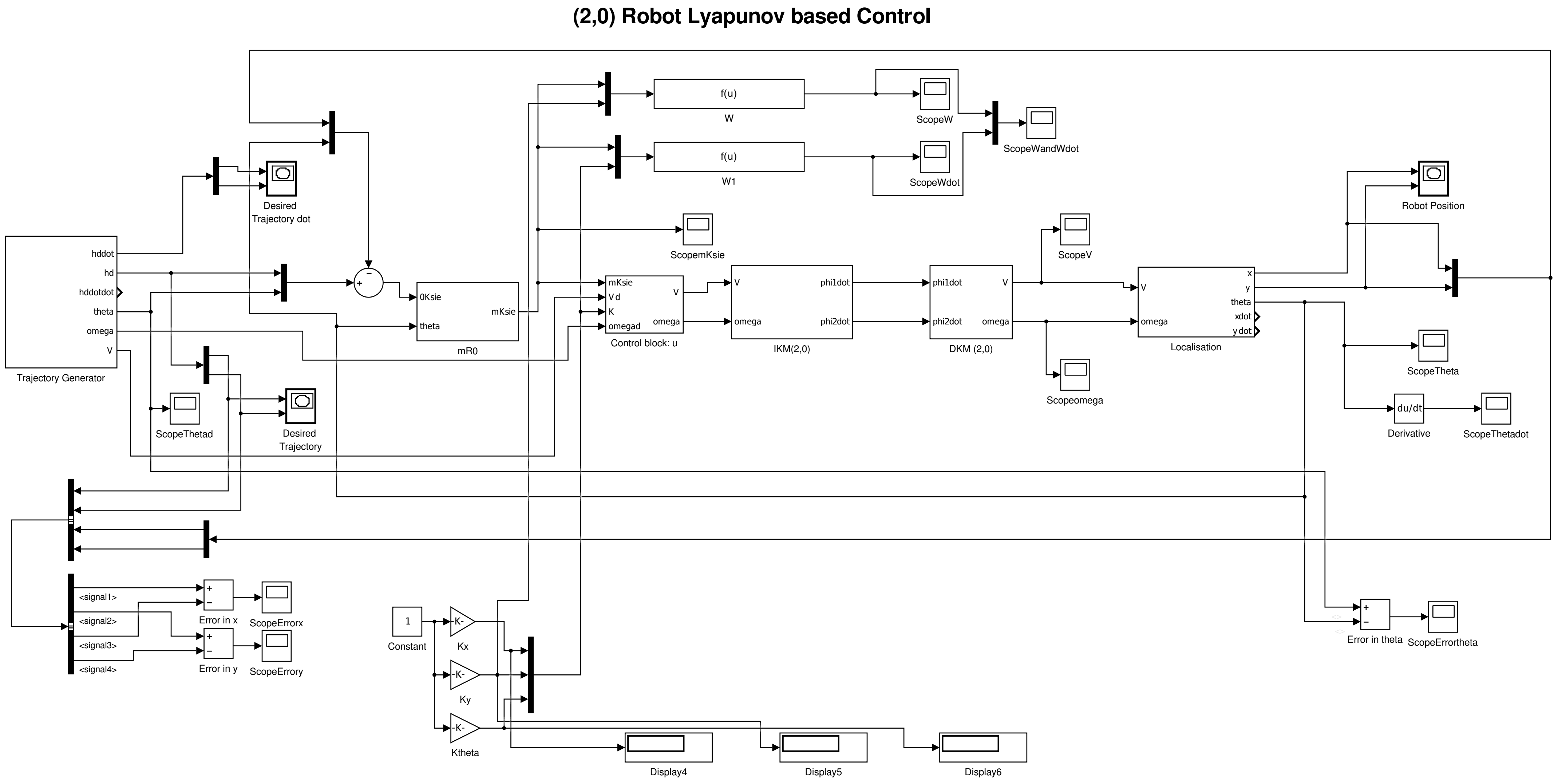

Figure 3: Simulink Lyaponov control schematic for the type (2, 0) robot.

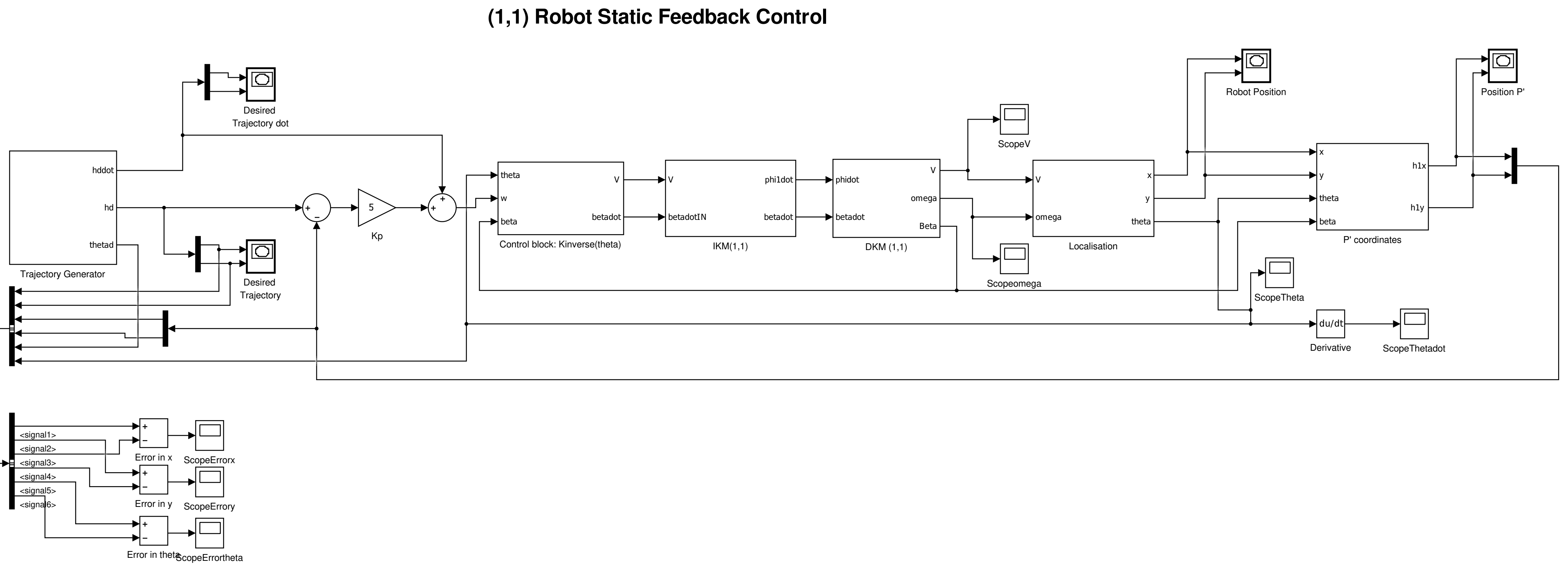

Figure 4: Simulink static decoupling control schematic for the type (1, 1) robot.

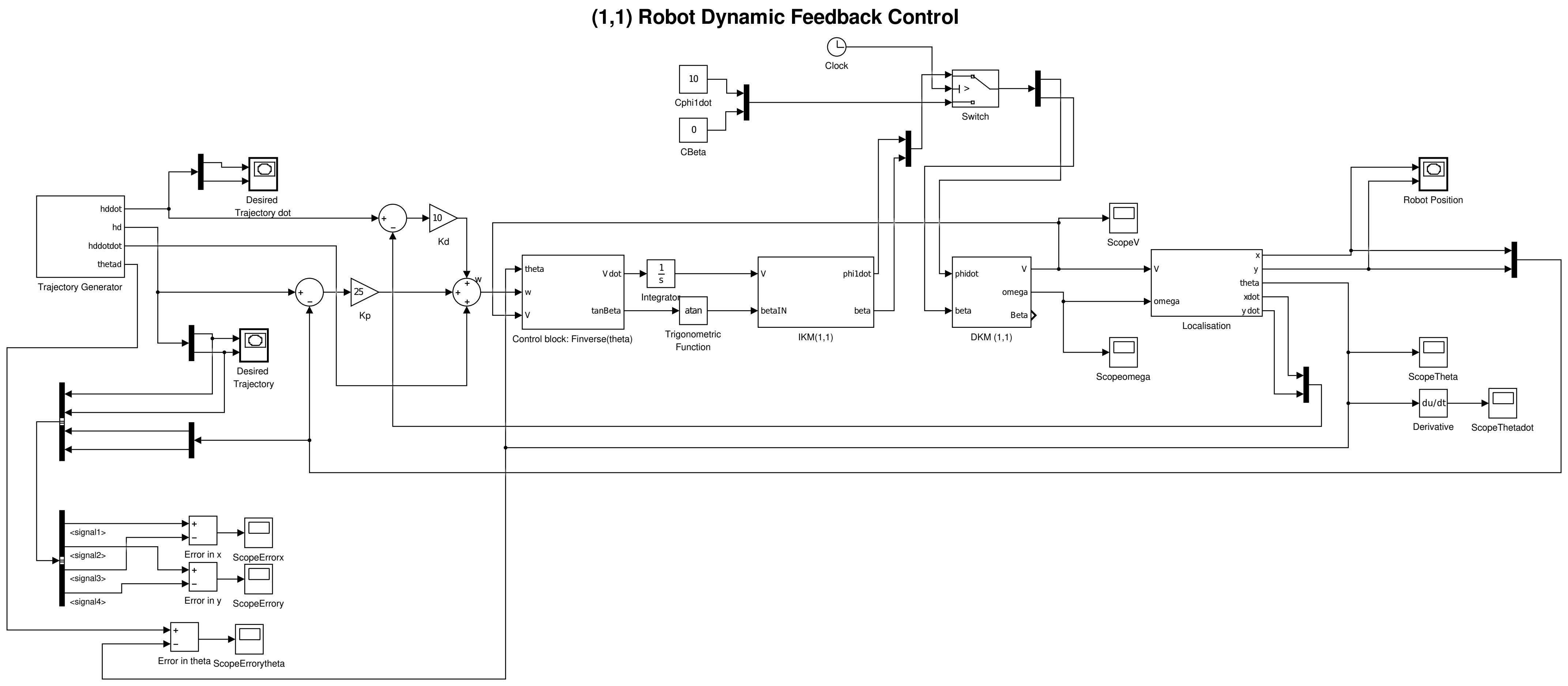

Figure 5: Simulink dynamic decoupling control schematic for the type (1, 1) robot.

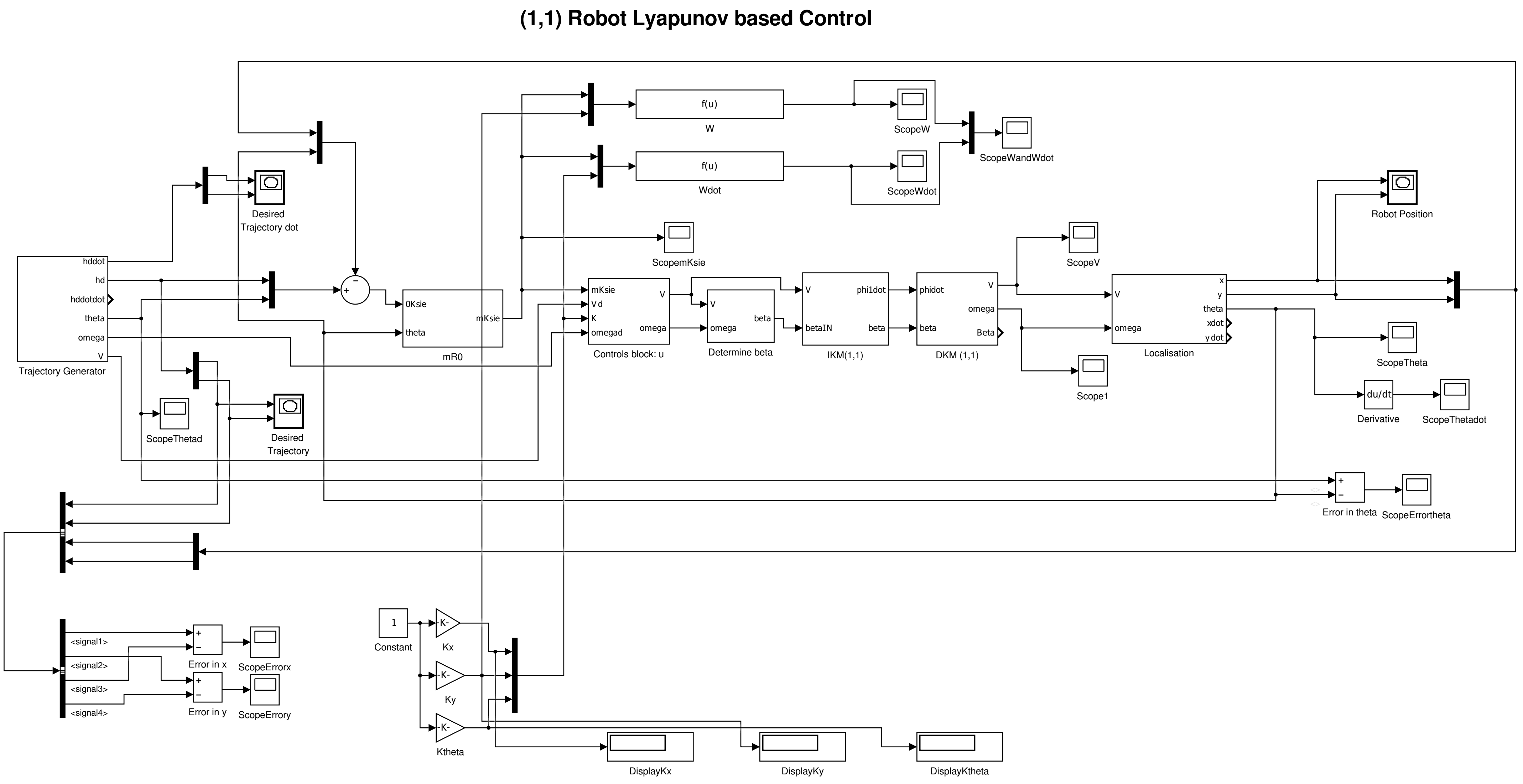

Figure 6: Simulink Lyaponov control schematic for the type (1, 1) robot.