Introduction to computer based control systems
Process lab 2

Name:	
Program:	
Date:	
Approved:	Sign

UPPSALA UNIVERSITY

DEPARTMENT OF SYSTEM AND CONTROL

1 Introduction

In this lab session you will first derive a mathematical model for the system, a robotic car, from the laws of mechanics. This is largely the same derivation as given in Computer Lab 1, but here the result is given as a transfer function instead of the state space formulation. The model will contain parameters that will determined experimentally by using system identification techniques. Finally you will tune and implement a PID controller that will be able to keep the vehicle on track.

2 Modeling of the system

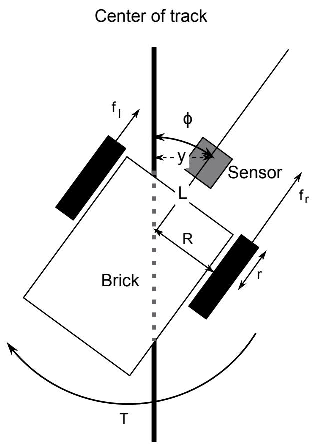

Figure 1: Robot viewed from above.

Table 1: List of variables

Variable	Description	Unit
T	Torque	Nm
R	Wheel axis radius	m
ϕ	Angle to track	rad
$f_{r,l}$	Force on left and right wheel	N
$\tau_{r,l}$	Torque on left and right wheel	$N \cdot m$
$\theta_{r,l}$	Rotation angle of left and right wheel	rad
I	Moment of inertia for robot around the center of rotation	$kg \cdot m^2$
L	Distnace from center of rotation to light sensor	m
y	Distance from center of track	m

Mechanical modeling of the car

The vehicle will be given a constant forward speed v and its direction of travel will be controlled by a feedback. An equal offset voltage u_0 is applied to both motors yielding the forward velocity v, and steering is performed by a differential voltage Δu. Denoting the voltage given to the left and right motor u_l and u_r respectively, this can be expressed as

$$\begin{aligned} u_l &= u_0 + \Delta u/2 \\ u_r &= u_0 - \Delta u/2 \end{aligned}$$

The aim is to keep the vehicle (or actually the position sensor) at the center of the track. The output of the system is thus the distance from the sensor to the center of the track y, and the input signal is the differential voltage Δu applied to the motors. In the controller program, Δu will be expressed as a percentage of the maximum voltage and this input will be denoted $u(t)$. A model that gives the relationship between the distance to the center of track y and the percentage of the differential voltage u is thus sought in the form of transfer function $G(s)$

$$Y(s) = G(s)U(s).$$

The movement of the robot can be decomposed into a translational movement and a rotational movement.

Translational movement

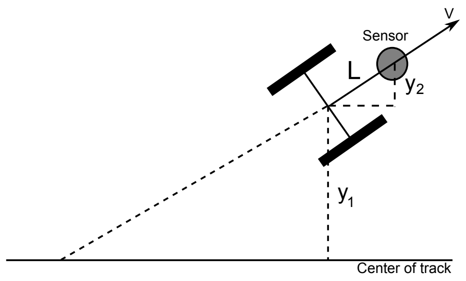

Figure 2: Translational part y_1 and rotational part y_2 of the total distance to the center of track y

The translational movement in question occurs in the direction perpendicular to the track that the vehicle is traveling along. This is described by the two distances y_1 and y_2, as shown in Fig. 2. The distance y_2 is the integrated velocity component perpendicular to the track while y_2 is the distance component perpendicular to the track due to the sensor rotation by the angle ϕ. Mathematically this can be expressed as:

$$\begin{aligned}
y(t) &= y_1(t) + y_2(t) \\
y_1(t) &= \int_0^t v \cdot \sin(\phi(\tau)) d\tau \\
y_2(t) &= L \cdot \sin(\phi(t))
\end{aligned}$$

For small values of ϕ, $\sin(\cdot)$ can be replaced by the angle itself, *i.e.* $sin(\phi) \approx \phi$ which gives

$$y(t) = y_1(t) + y_2(t) \approx L\phi(t) + v\int_0^t \phi(\tau) d\tau \qquad (1)$$

The translational velocity of the vehicle is proportional to the offset voltage u_0

$$v = K_u u_o$$

Note that the "constant" K_u depends on the offset voltage $K_u = K_u(u_0)$, e.g. a doubling of u_0 will not double v. However for a constant u_0 this K_u can be regarded as a constant.

Laplace transformation of Eq. 1 gives

$$Y(s) = (L + \frac{v}{s})\Phi(s)$$

The transfer function from $\phi(t)$ to the distance to the center of the track is thus given by

$$G_T(s) = L + \frac{v}{s} = L + \frac{K_u}{s} u_0$$

Rotational movement

The laws of mechanics give the following relations

$$\begin{aligned} T &= I\ddot{\phi} && (2)\\ T &= (f_l - f_l)R && (3)\\ f_{r,l} &= \frac{\tau_{r,l}}{r} && (4)\\ \phi &= r\frac{\theta_l - \theta_r}{2R} && (5) \end{aligned}$$

Combining Eq. 2-4 together gives

$$(\tau_l - \tau_r)\frac{R}{r} = I\ddot{\phi} \qquad (6)$$

This relates the torque exerted by the left and right wheel to the turning angle ϕ. Since the input is the voltage to the left and right motor, a DC motor model that describes the relation between the input voltage and the output torque is needed. The total torque τ_{tot} that the motor will exert is proportional to the input voltage, i.e. $(\tau_{tot})_{r,l} = K_\tau u_{r,l}$. To produce an angular acceleration of the wheel requires a torque of $\tau_w = I_w\ddot{\theta}$. Furthermore, there is a torque required to compensate for the friction and back emf of the motor that is proportional to the angular velocity of the wheel $\tau_f = K_b\dot{\theta}$. The remainder of the torque τ_l will act on the load, i.e. move the vehicle. This gives the relationship (dropping the indexes l and r for left and right motors)

$$\begin{aligned} \tau_{tot} &= \tau_w + \tau_f + \tau_l \Leftrightarrow\\ \tau_l &= \tau_{tot} - \tau_f - \tau_w\\ &= K_\tau u - I_w\ddot{\theta} - K_b\dot{\theta} && (7) \end{aligned}$$

Now, inserting Eq. 7 into Eq. 6 yields

$$\frac{rI}{R}\ddot{\phi} = K_\tau(u_l - u_r) - K_b(\dot{\theta}_l - \dot{\theta}_r) - I_w(\ddot{\theta}_l - \ddot{\theta}_r)$$

Using Eq. 5, this can be written as

$$\begin{aligned} \frac{rI}{R}\ddot{\phi} &= K_\tau\Delta u - \frac{2R}{r}K_b\dot{\phi} - \frac{2R}{r}I_w\ddot{\phi} \Longleftrightarrow\\ (\frac{rI}{R} + \frac{2R}{r}I_w)\ddot{\phi} &= -\frac{2R}{r}K_b\dot{\phi} + K_\tau\Delta u && (8) \end{aligned}$$

where $\Delta u = u_l - r$. This second order differential equation gives the dynamic relationship between the input Δu and the output ϕ. Since $u(t) = K_\Delta\Delta u(t)$ for some constant K_Δ, the transfer function from u to the angle ϕ is thus given by Laplace transform of Eq. 8 which gives

$$\Phi(s) = \frac{K}{s(Ts+1)}U(s)$$

where

$$\begin{aligned} K &= \frac{K_\Delta K_\tau r}{2RK_b} \\ T &= \frac{1}{K_b}(\frac{r^2 I}{2R^2} + I_w) \end{aligned}$$

The transfer function from u to ϕ is thus

$$G_R(s) = \frac{K}{s(Ts+1)}$$

The transfer function for the whole system

The transfer function for the whole system is given by $G_R(s)$ in series with $G_T(s)$ as shown in Fig. 3.

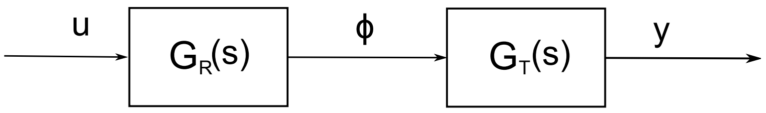

Figure 3: Block diagram for the system.

$$G(s) = G_R(s)G_T(s) = K\frac{Ls+v}{s^2(Ts+1)}$$

3 Parameter identification

The transfer function $G(s)$ has the parameters K, T, v, and L that have to be determined. This can be done using system identification techniques. Note that these are the same parameters that are used in the state space formulation in Computer Lab 1.

3.1 Identifying $G_T(s)$

First of all: Open Matlab, get to the lab directory, type **addLabPath** to get access to all required m-files.

Task: Determine the parameters v and L of $G_T(s)$:

- Use a ruler to determine the distance from the center of rotation to the middle of the sensor L.
- Open **actuatorTest.m** by typing **open actuatorTest** in Matlabs command prompt. Make the vehicle run forward for 3 seconds by applying 40% of the maximum voltage to the motors. Measure the distance that the vehicle has moved during the 3 seconds and determine v from this experiment.

What values of L and v have you obtained?

3.2 Identifying $G_R(s)$

The transfer function from u to ϕ is $G_R(s)$. This is a second order transfer function. The transfer function from u to $\dot{\phi}$ is given by $sG_R(s) = \frac{K}{Ts+1}$ which is a first order transfer function but with the same parameters K and T. The system response to a step of $u = 60\%$ will have the signal shape similar to that in Fig. 4. It can be seen that the values K and T can be estimated from the step response since $y(T) = 0.63 \cdot 60K$ and $60K$ is the static gain of the step.

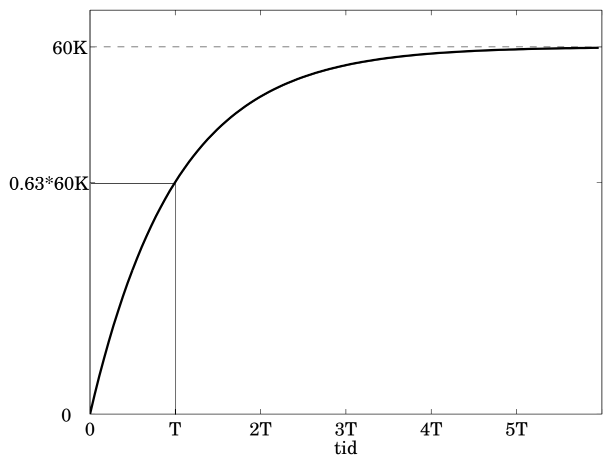

Figure 4: Output signal $y(t)$ when the input signal $\Delta u(t)$ is a unit step

Task: Run the m-file identification.m by typing **identification** in Matlabs command prompt and hit enter. This m-file will apply a step of $u = 60\%$ and measure the angular velocity $\dot{\phi}$ and then plot the step response. Study the obtained plot and determine the parameters K and T.

Question What values of K and T have you obtained?

You have now identified the system and the obtained model can be used to construct a model-based controller. This will be done in the Lab 3. In this lab however you will continue with designing a PID regulator, which can be implemented without a model.

4 PID control

You will now implement a PID controller to control the vehicle. You will here see the practical behavior of the PID that were studied in simulation in Computer Lab 1. The PID controller is by far the most used controller in the industry because of its simplicity and capability of achieving good control performance without having a detailed model of the system.

The PID controller consists of three parts: the proportional part (P), the integrating part (I), and the differentiating part (D). You will first implement only the proportional part and then add the integrating and differentiating parts. The PID controller will be implemented in state space form and is readily provided in the file **lineTrackerPID.m** that you will use for the implementation of the controller.

First the light sensor must be calibrated, in the same way as in Lab 1. i.e.

- Call [y l] = calibrateLS(y), with the input $y = 10^{-3}[-10, -5, 0, 5, 10]$
- Place the sensor at the specified values of y and hit enter.
- Determine a_0 and a_1 by calling $polyfit(y, l, 1)$
- Check your result using $plotCalCurve()$ function

4.1 Proportional control (P-control)

P-control uses the control law

$u(t) = K_P e(t)$

where $u(t)$ is the output signal of the controller and $e(t) = r(t) - y(t)$ is the control error with respect to the reference signal $r(t)$. In this application the reference signal is zero, $r(t) = 0$, since the vehicle should be kept at the center of the track) which thus gives $e(t) = -y(t)$.

Task: Use the m-file **lineTrackerPID.m**. Open it by typing **open lineTrackerPID** in Matlabs command prompt. Implement a proportional controller and try out some values of K_P to see the system behavior. To disable the integrating and differentiating parts, let $K_I = 0$ and $K_D = 0$.

Question: How does the value of K_P affect the behavior of the controlled system?

Question: How does the value of K_P affect the magnitude of the input signal $|u(t)|$?

Task: Make the vehicle run on the track. Does the output y seems to be centered?

4.2 Proportional control with integration (PI-control)

When using only P-control, there is thus a possibility of a static error, i.e. $\lim_{t \to \infty} e(t) \neq 0$. To eliminate the static error, an integrating part is usually added to the controller, yielding the control law

$$u(t) = K_P e(t) + K_I \int_0^t e(\tau) d\tau.$$

This control law will increase the magnitude of the control signal as long as $e(t) \neq 0$ forcing it to eliminate static error. However, when the $e(t) = 0$, the integrated error will not be zero which will cause the vehicle to somewhat overshoot the center of the track. Introducing the integration thus eliminates the static error but makes the system more oscillatory.

Task: Use the m-file **lineTrackerPID.m**. Open it by typing `open lineTrackerPID` in Matlab's command prompt. Implement a PI controller and try out some values of K_P and K_I to see the system behavior. Use $K_D = 0$ to disable the differentiating part.

Task: Run the vehicle on the track, is there still a static error?

4.3 Proportional control with integration and differentiation (PID control)

To dampen the oscillations in the closed-loop system, a differentiating part can be added to the control law, i.e.

$$u(t) = K_P e(t) + K_I \int_0^t e(\tau) d\tau + K_D \frac{de(t)}{dt}$$

Assume that $e(t) > 0$. Turning away from the center of the track results in $\frac{de(t)}{dt} > 0$. This will increase the control signal and make the vehicle to turn back on track faster. When the vehicle start to turn towards the track, the sign of the derivative will change $\frac{de(t)}{dt} < 0$ and the control signal will decrease, resulting in a smaller overshoot. The differentiating part will thus make the control more damped but also slower.

Note: There is not a good idea to use pure differentiation of the error signal since it is often corrupted by high frequency noise. Differentiating a noisy signal will enhance the noise and the system will get a jerky behavior. To remedy this problem, the signal $e(t)$ is typically low-pass filtered before differentiating it. This is taken care of in the function `pid`.

Task: Implement a PID controller using **lineTracker.m** and try out some values of K_D to see the system behavior.

4.4 Tuning a PID

It can be difficult to get good performance when tuning a PID manually as you did in the previous tasks. There are many tuning algorithms based on experimental data available for the PID controller, providing feasible values for the parameters K_P K_I and K_D. You will here use a method called relay tuning. Relay tuning is not part of this course but the interested reader can find more about it in e.g. Åström & Hägglund 1994. Here you will only follow the given "recipe" for the tuning procedure.

The experiment for relay tuning is to use a relay with amplitude h as control law, i.e.

$$u(t) = \begin{cases} h & \text{if } y(t) \geq 0 \\ -h & \text{if } y(t) < 0 \end{cases}$$

The system is then regulated with this control law which will cause it to oscillate with a certain frequency ω_0 and amplitude C. The obtained values of ω_0 and C give information used to specify the parameters for the PID controller.

Task:

- Put the vehicle on the straight part of the track.
- Call the function **relayTuning(a0,a1)** by typing `relayTuning(a0,a1)` in Matlab's command prompt and hit enter, where a_0 and $a_1$is the light sensor calibration parameters. This m-file will run the system with the relay control, where the relay has an amplitude of $h = 30\%$ of the max voltage, and also plot the result of the execution.
- Study the obtained plot and determine the frequency ω_0 and the amplitude C of the oscillation.
- Use the function `[Kp, Ki, Kd] = pidRelayParam(w_0,C)` to get the values for the PID parameters.
- Use the obtained parameters in **lineTrackerPID.m** and run the code to see the regulation of the system with these parameters.

Techniques for auto tuning of PID regulators are far from "fool proof", but they give a good initial guess of the parameters. Manual adjustments of the obtained parameters are therefore often performed afterward.

Task: Try to manually tune the parameters K_P K_I and K_D based on your knowledge about how the parameters will effect the behavior of the regulated system. Demonstrate your line tracker for the lab assistant when you have come up with a result that you are satisfied with.