

1.8 Closed-Loop Control

The primary control problem in this assignment is to make steering corrections toward a desired *heading*, or direction of travel. Every one-tenth of a second, your component compares the *actual* heading read from the compass, to the *desired* heading held in the component state. The difference between the desired and actual headings is the *steering error*.

`Square` computes a *turning radius* based on the steering error, in units of degrees, and issues a command through `jd driver.s` to the steering actuator. On ERTS, the steering unit is the *inverse of the desired turning radius* for the vehicle. That is, steering control is in units of meters⁻¹.

If all you are worried about is accurate steering this choice may seem strange, at first, but it is actually more intuitive and convenient:

- For higher-level navigation, turning radius is more important than steering angle.
- Going straight translates to an infinite turning radius. Hence, there is a mathematical divergence at the most “interesting” steering angle, and consequently, instability in digital computations around that point.
- When the turning error is zero, so is the correction, making the correction function simpler to compute, as we shall see.

1.8.1 The PID function

The PID is a basic function for closed-loop control. It is defined as

$$C = P \cdot \epsilon + D \cdot \Delta\epsilon + I \cdot \Sigma\epsilon$$

where ϵ is the *error*, $\Delta\epsilon$ is the *change in error* and $\Sigma\epsilon$ is the cumulative error.

P specifies a correction that is *proportional* to the error. The greater the error the greater the correction. This is what is needed for the steering correction. Think about driving an automobile.

D is the derivative of the error. In the discrete case, if ϵ_t is the error at one cycle and $\epsilon_{t'}$ at the next, then the change in error at cycle t' is

$$\Delta\epsilon = \epsilon_{t'} - \epsilon_t$$

The D coefficient comes into play when a sudden change in the system state requires an extra correction. This is not the case for steering, but it will be later on, when we consider speed control.

I is the *integral* of the error over some time period. In the discrete case,

$$\Sigma\epsilon = \sum_{t-n}^t$$

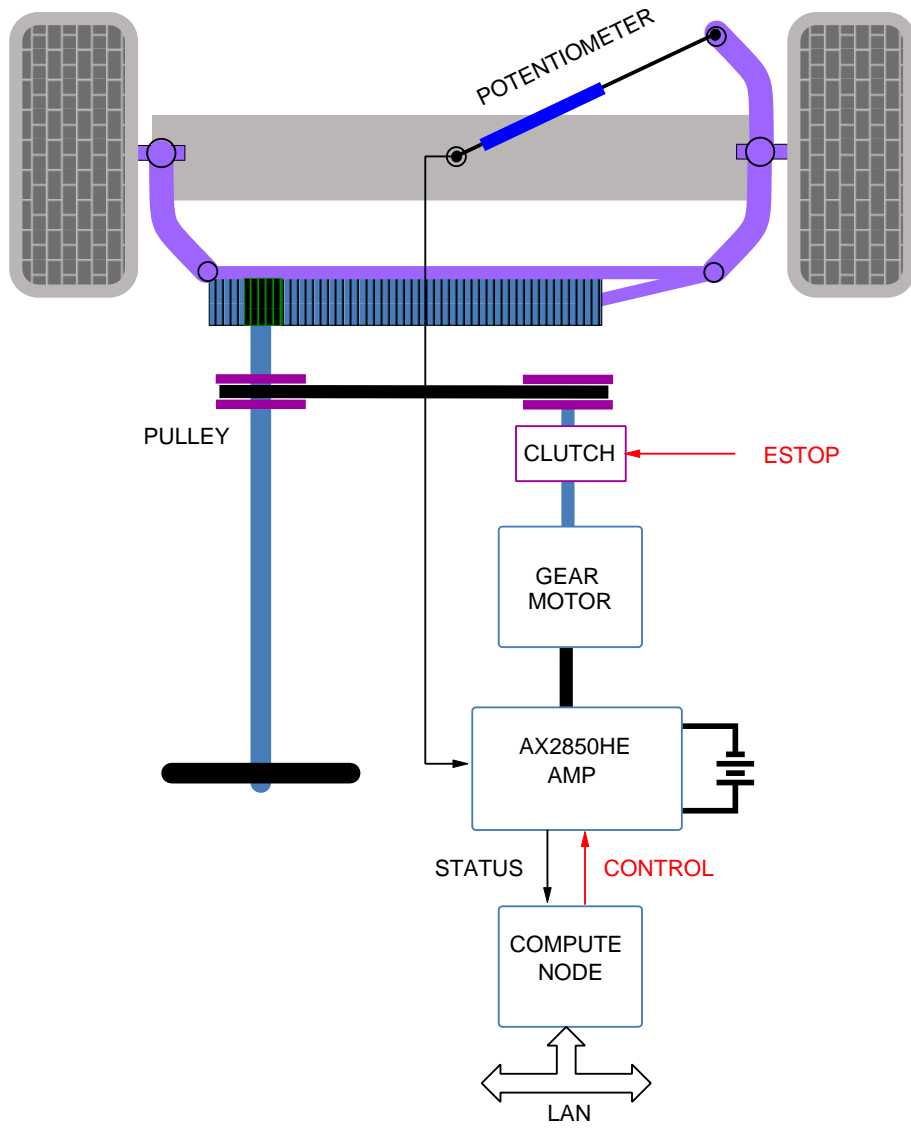


Figure 1.1: Steering Control

for the current cycle, t . The I coefficient comes into play when the control function fails to converge on a zero error. A thermostat that doesn't quite bring the temperature to its setting is an example. Over time, the cumulative error causes in a "boost" in the correction. The I coefficient will not be needed in any of the control calculations for ERTS.

So a good steering correction for ERTS is

$$C_s = P \cdot \epsilon + 0 \cdot \Delta\epsilon + 0 \cdot \Sigma\epsilon = P \cdot \epsilon$$

As ϵ approaches 0, C_s does too, so that steering is straight ahead, as desired.

It remains to determine the value for P . We should try to come up with a reasonable estimate for P and then "tune" it on the test field.

- *As you are tuning P , experiment with some bad values and observe their effects. Discuss these observations in your lab report.*

1.8.2 Estimating P

The steering correction for this assignment is

$$s' = P \cdot e$$

where e is the current steering error and P is a proportionality constant. Error e is in units of degrees and s' is in units of m^{-1} (*inverse* turning radius (ITR) in meters). Although s' is called a "correction," it is not an adjustment to the current steering, but instead, a *re-calculation* of what the ITR value should.

A first estimate for P is based on looking at the extreme case. Disregarding the initial error, the nominal worse-case error arises when the vehicle reaches a turning point. There, the error will suddenly jump to $\pm 90^\circ$. The assignment specifies that the turning radius should be at least 3 meters, so the *inverse* turning radius, *ITR*, should be no greater than one-third. In the most extreme case,

$$\frac{1}{3} \geq P \cdot 90$$

Solving for P yields

$$P \leq \frac{1}{3 \cdot 90} = 0.003\overline{7}$$

This function procedure illustrates a few other important things to do (Fig. 1.2)

- Actual heading, desired heading, and *difference* (error) are translated to the range $(-180, 180]$. It is certainly reasonable to have a heading of, say, 270° , and perhaps even 540° . However, headings that differ by 360 are just aliases of the same heading. Furthermore, the steering correction calculation is based on the heading range $(-90, 90)$ and we should not diverge too far from that range.

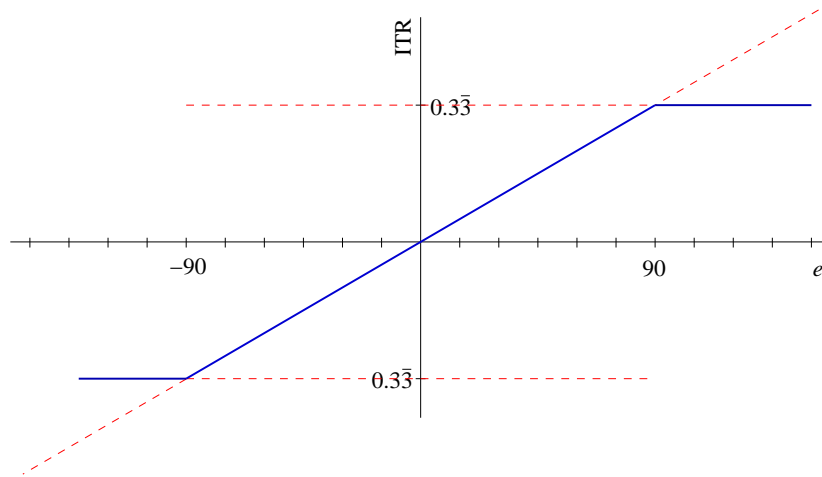


Figure 1.2: Estimating P

- (b) Before returning, `calc_inv_turn()` restricts the result to $s' \leq |M|$, where M is a limiting ITR (`max_turn` in the listing). Such a restriction is called *clipping*. It keeps the value returned in a “safe” range with respect to the vehicle’s mechanical capabilities. For example, if a programming error were to compute an ITR of 10, the vehicle could not possibly execute a turn with radius $\frac{1}{10}$ m (10 cm).
- (c) The estimate of P above is based on a maximal error of 90° , but we know the error can be twice that. And a larger correction results in a tighter turning radius, which we do not want. However, clipping fixes that problem.

It is probably no surprise to you that the more primitive steering mechanism also do clipping. In fact, ERTS has mechanical limit switches that disconnect the actuator if the wheels are turned to far. Extreme motion of the steering rack can cause damage to the parts! Thus simply by inspecting the statement sequence

```
# Constrain (clip) our turn to (-max_turn, max_turn)
if t > max_turn: t = max_turn
elif t < -max_turn: t = -max_turn
return t, diff
```

one may verify that the result is in a “safe” range.