Regeltechnik

Beispiel: Geradeaus Fahren

→ beide Räder drehen parallel.

Problem: Ungleiche Motoren,
Unebener Grund,
Entladene Batterien
....

Gleiche Power auf beide Motoren löst das Problem nicht.

Einstellungen experimentell bestimmen
ist nur für einen Roboter gültig.

Ziel: Zustand messen
→ Einstellung anpassen.

z.B. ticks links $\overset{\text{soll}}{=}$ ticks rechts

Ansatz:
- Soll (gegeben) (was man möchte)
- Ist (gemessen) (was man hat) oder glaubt zu haben.
- Fehler: Ist − Soll
- Einstellung aus Fehler berechnen.

z.B. lineare Funktion!

```
while (true) {  // Endlos-Schleife
  int fehler = ticks(1) - ticks(0);  // robot.motors.getTicks(C)....
  int powL = 700 + 20*fehler;  // lineare Funktion
  int powR = 700 - 20*fehler;  // Ausprobieren
  robot.motors.setPowers(powL, powR);  // Einstellung
  if (Abbruchbedingung) {  // Button, US, Zeit, ticks,...
    break;  // innerste Schleife verlassen
  } // if fertig
} // while fertig
// evtl. Aufräumen. z.B Motoren stop.
```

Soll ist 0
Messung

```
if (us.measure()
```

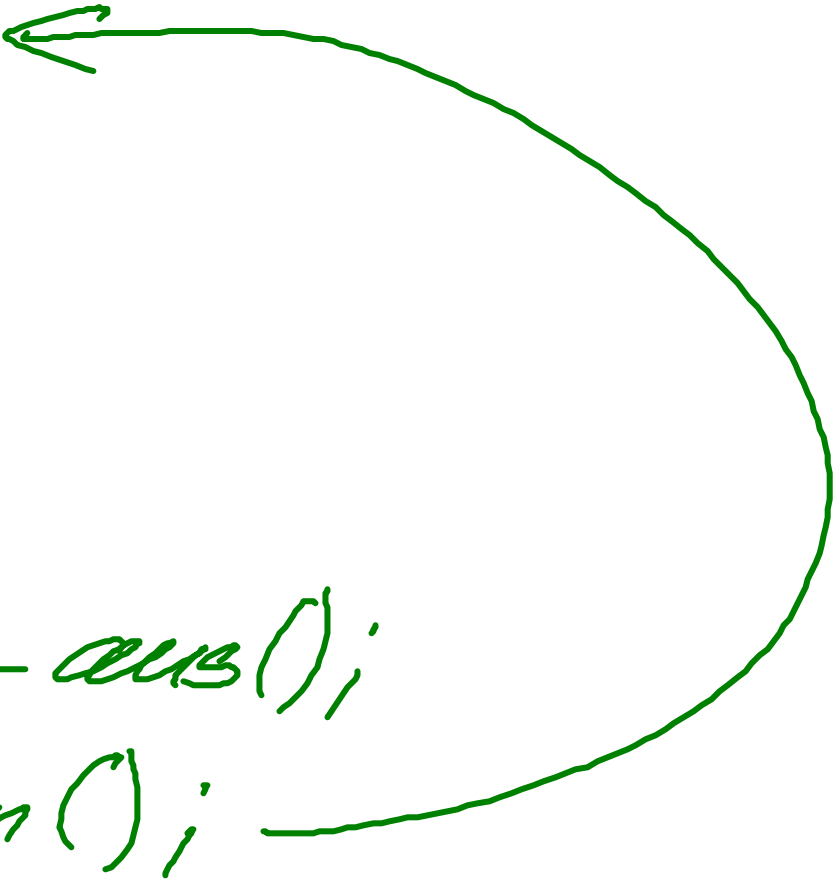
---

int : Ganzzahl ca. -32'000 bis 32'000

float : Kommazahl long : Ganzzahl bis $\pm 2 \cdot 10^9$

```
float d = us.measure();
if (d > 0.0 && d < 20.0) {
    break;
}

loop() {
    geradeaus();
    drehen();
}
```



P.I.D. - Regler

Proportional: Einstellung lineare Funktion vom Fehler

Integral: Einstellung proportional zur Summe der Fehler

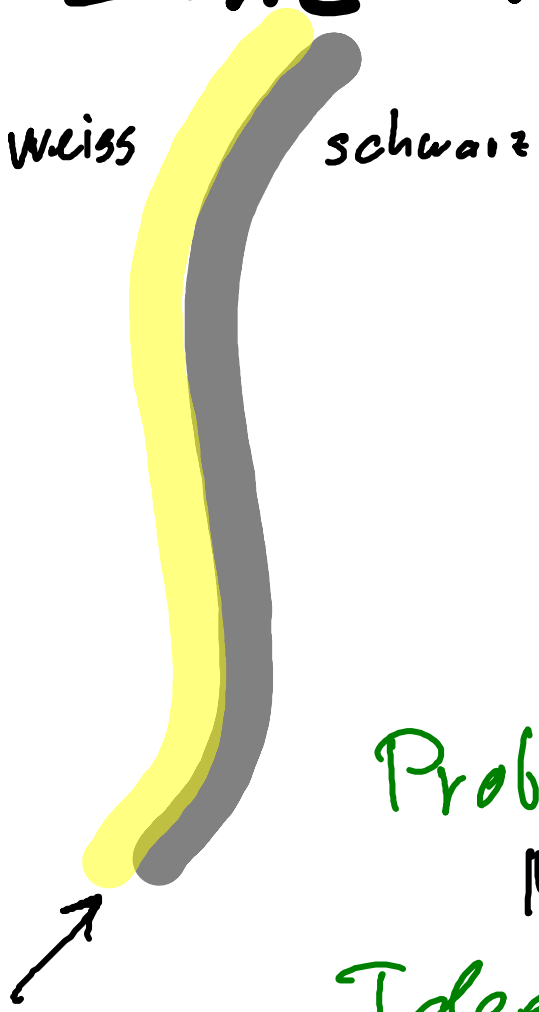
Differential: Einstellung proportional zur Änderung des Fehlers

Effektive Einstellung: Summe der drei.

Beispiele:
- P-Regler: Geradeaus Fahren
- P.I.-Regler: Heizung
- P.I.D-Regler: Tempomat
- P.D.-Regler: Line-Follower

3 Faktoren (Gewichte): $p, i, d \in \mathbb{R}$

Line - Follower



Soll: grau ← Wert experimentell Bestimmen.

Ist: Messung IR-Sensor:

```
int   wert = ir.measure();
float v = .... // Fehler auf Intervall [-1,1]
float diff = (v-lastv)/zeit;
```

Problem mit P-Regler:

Motoren sind träge ⇒ Überschiessen

Idee P.D.-Regler:

Je schneller die Helligkeit ändert,
desto mehr Gegensteuer.

Idee P.I.D.-Regler

in Kurve immer leicht daneben

→ Summe der Fehler wächst

→ zusätzliche Korrektur

→ Problem: nach Kurve leicht daneben.

Weitere Anwendung

Einer Wand folgen (z.B. Abstand 30 cm).

Anwendung

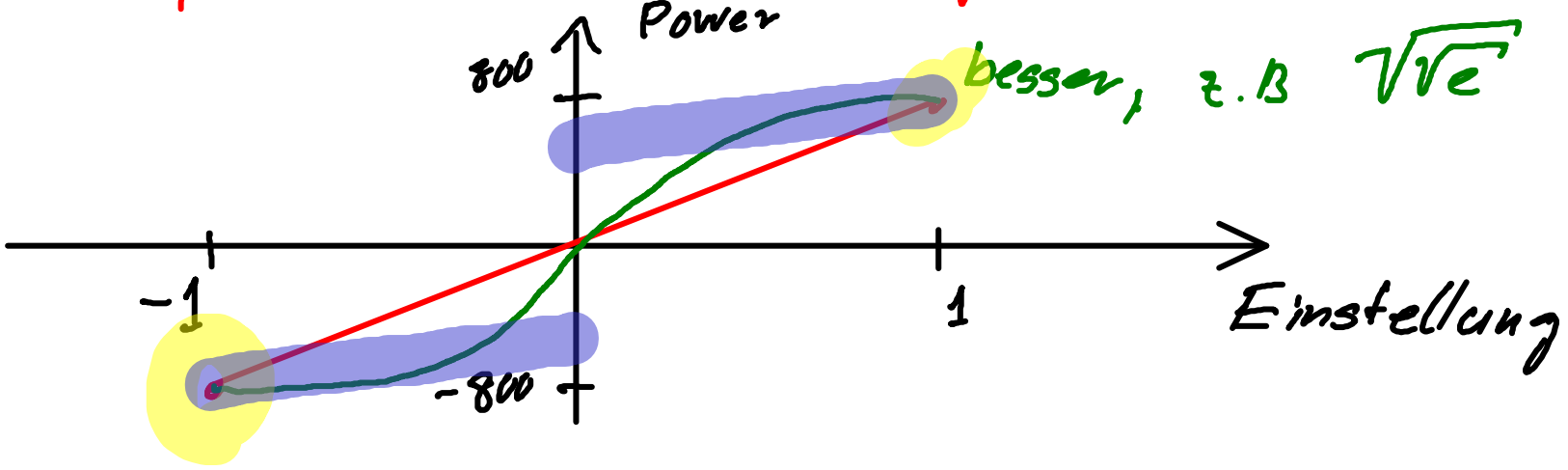
1.) Faktoren $i, d = 0$

p experimentell bestimmen.

2.) d anpassen

3.) evtl. i

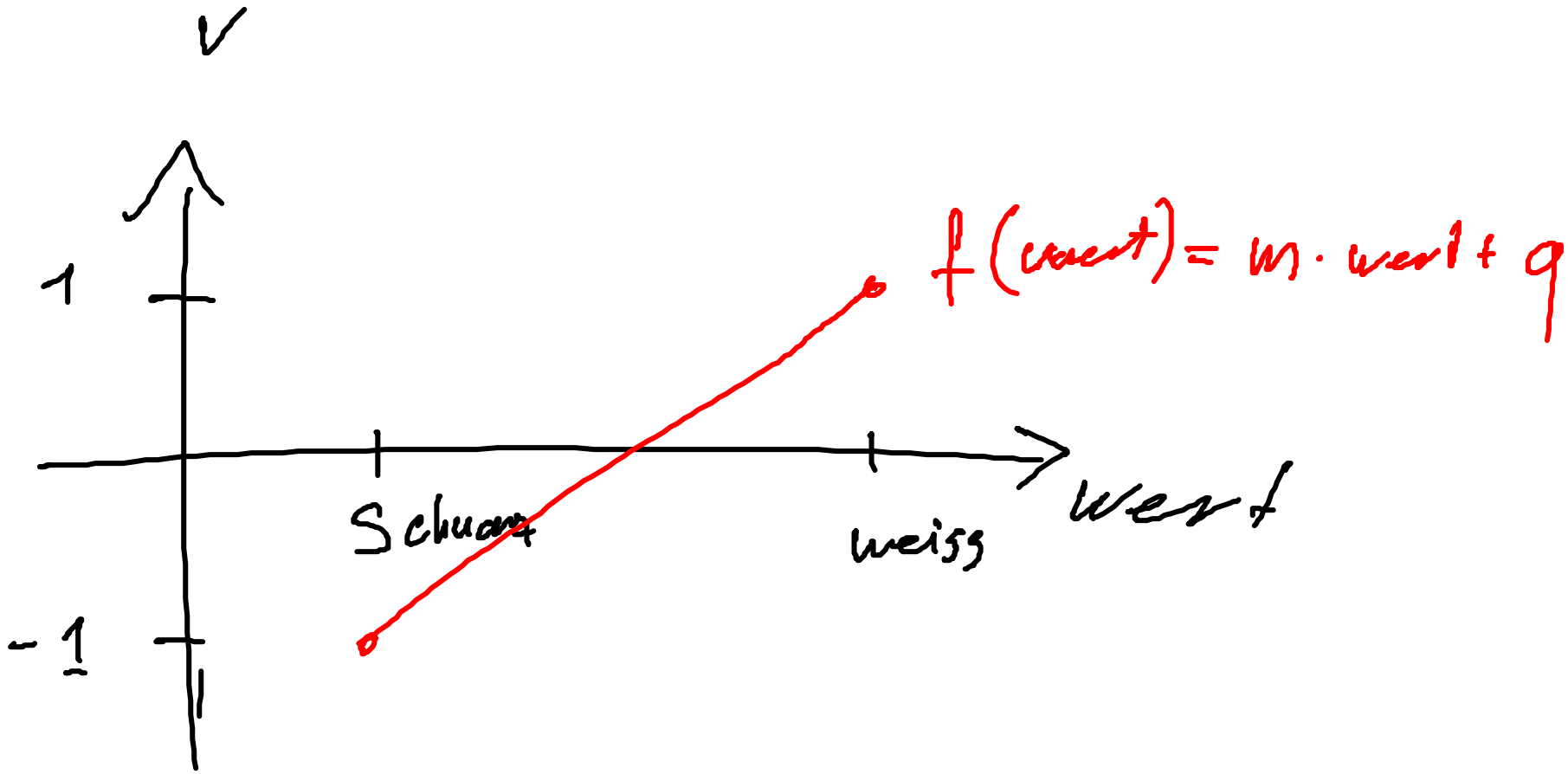
Umrechnen auf $[-1, 1]$ macht Zahlen einfach fassbar und manipulierbar



besser, z.B $\sqrt{\sqrt{e}}$

```
// Die Umrechnung braucht nicht linear zu sein! Die Motoren d
rehen schliesslich erst ab ca. Power 500
void setFloatPower(float links, float rechts) {
  // TODO: bessere Umrechnung (z.B. [0,1]->[500,799] und [-1,
0]->[-799,-500])
  int l = links*799;
  int r = rechts*799;
  robot.motors.setPowers(l,r);
}

// Funktion, die den Werte zwischen -1.0 und 1.0 beschränkt
// z.B. ist clip(4.2)=1.0, clip(-0.42)=-0.42 und clip(-42)=-
1.0
float clip(float f) {
  if (f>1.0) {
    return 1.0;
```


v
1
-1
Schwarz
weiss
wert
f(wert) = m · wert + q