16-311-Q Introduction to Robotics

Lecture 10: Feedback-based Control 2

Instructor:
Gianni A. Di Caro

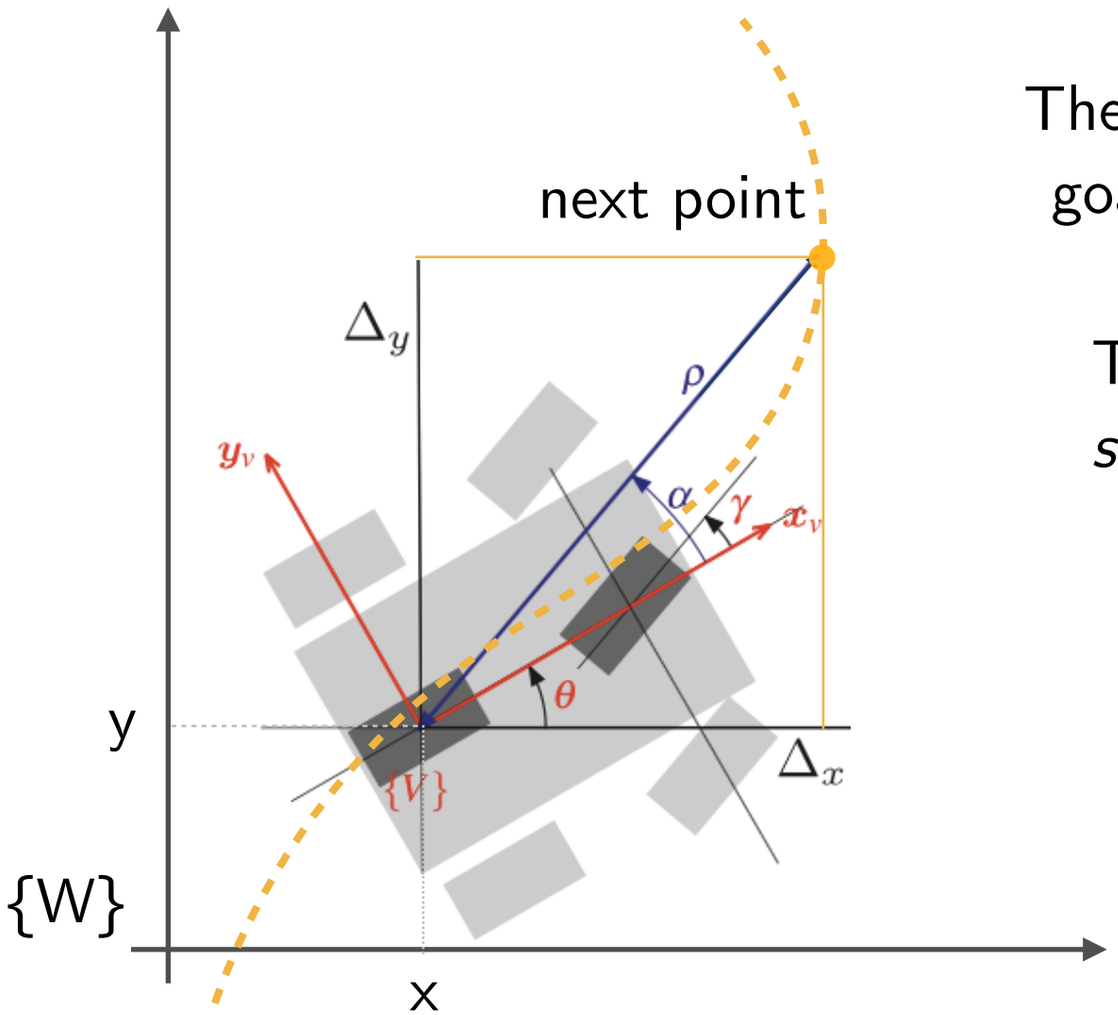
FOLLOWING A PATH (PI CONTROLLER)

Goal: Follow a general parametric curve (x*(t),y*(t)) on the plane
(e.g., generated by a path planner or resulting from sensors)

Control inputs: γ(t), v(t)

Current known state: Pose [x y θ](t) and the next point in the path (in the {W} frame)

Error vector: Distance and angle from next point in the path



Pursuit strategy:

The next point (x*(t),y*(t)) on the path defines the goals state and keeps moving along the goal path, with the robot keeping following it.

The problem becomes therefore equivalent to a *sequence of "moving to a goal point" problems, arriving with non-zero velocity*

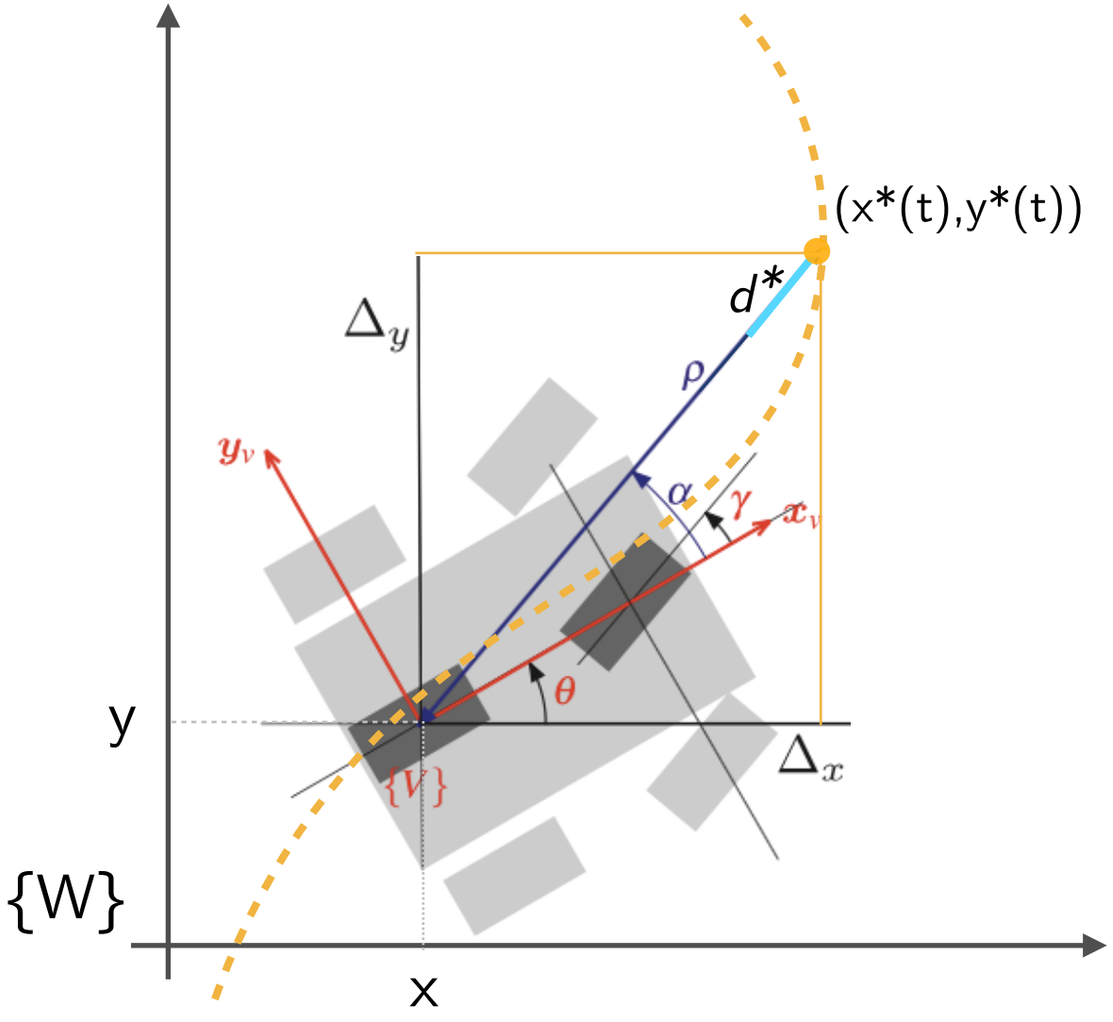
Carrot on a stick!

FOLLOWING A PATH (PI CONTROLLER)

Control inputs: γ(t), v(t)

Current known state: Pose [x,y,θ](t) and the next point in the path (in the {W} frame)

Error vector: Distance (with some offset) and angle from next point in the path



Velocity proportional to the linear distance ρ from the pursuit point, that should be maintained at a defined distance d*

$$\varepsilon_\rho(t) = \sqrt{(x^*(t) - x(t))^2 + (y^*(t) - y(t))^2} - d^*$$

The linear velocity PI controller aims to keep the robot at the distance d* from the next point. *The integral term guarantees a finite velocity when the error on distance goes to 0*

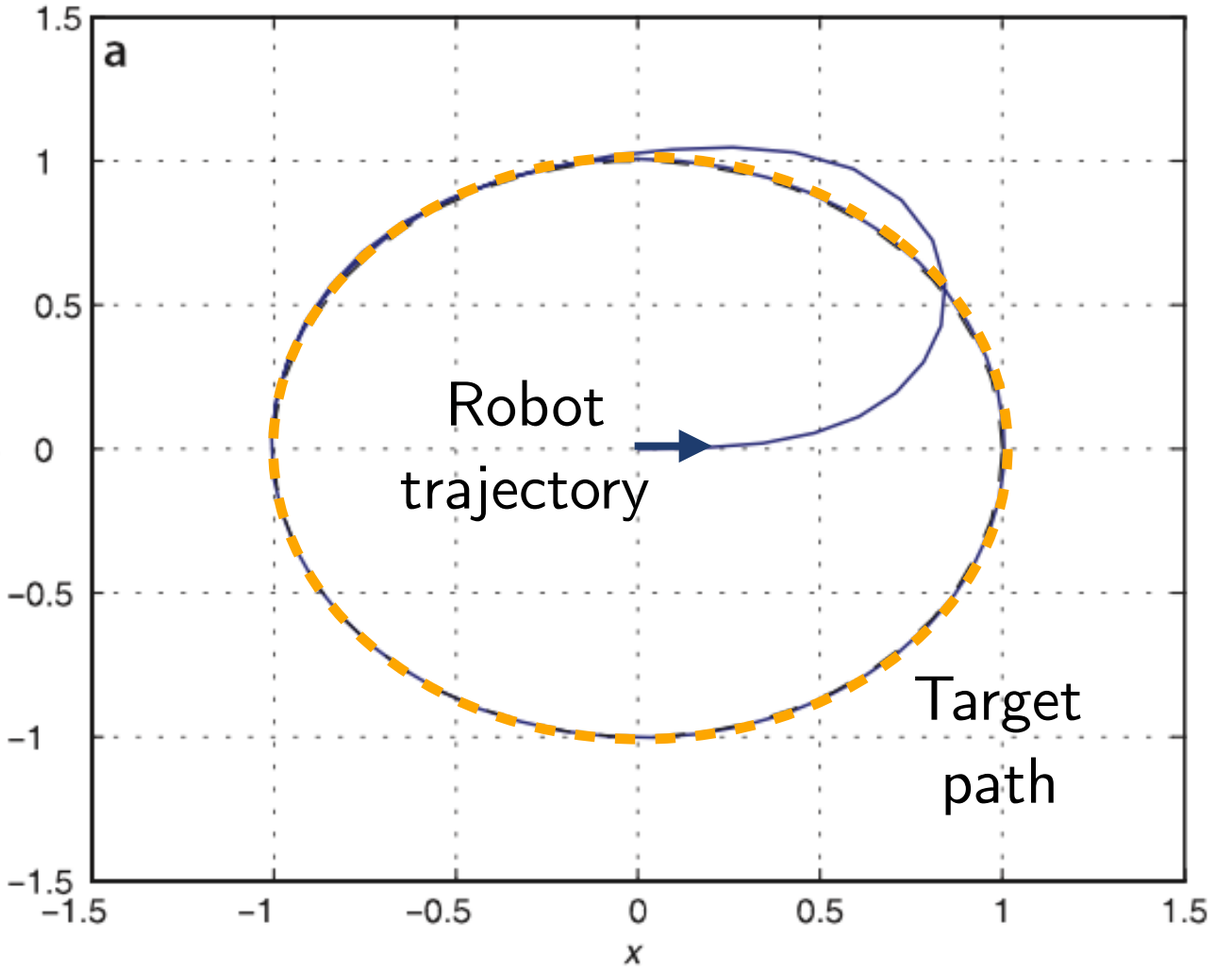
$$v^*(t) = K_\rho \varepsilon_\rho(t) + K_i \int \varepsilon_\rho(t) dt$$

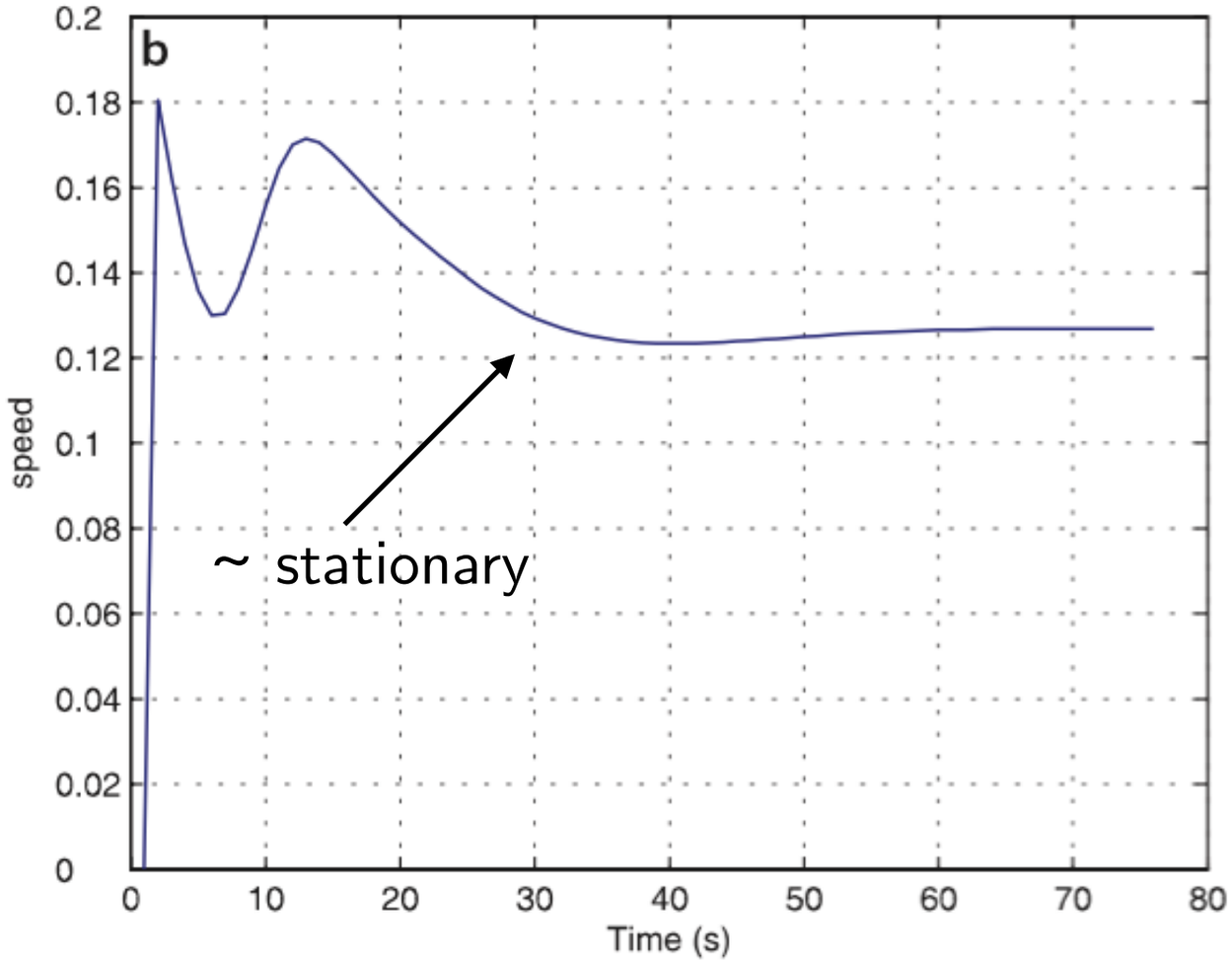
Steering proportional to the angular difference between robot and target point

$$\theta^* = \text{atan2}\left(\frac{y^* - y}{x^* - x}\right)$$

$$\gamma(t) = K_\theta(\theta^* \ominus \theta), \quad K_\theta > 0$$

FOLLOWING A PATH (PI CONTROLLER)





$d^* = 0.05,\ \ K_\rho = 4,\ \ K_i = 2,\ \ K_\theta = 5$

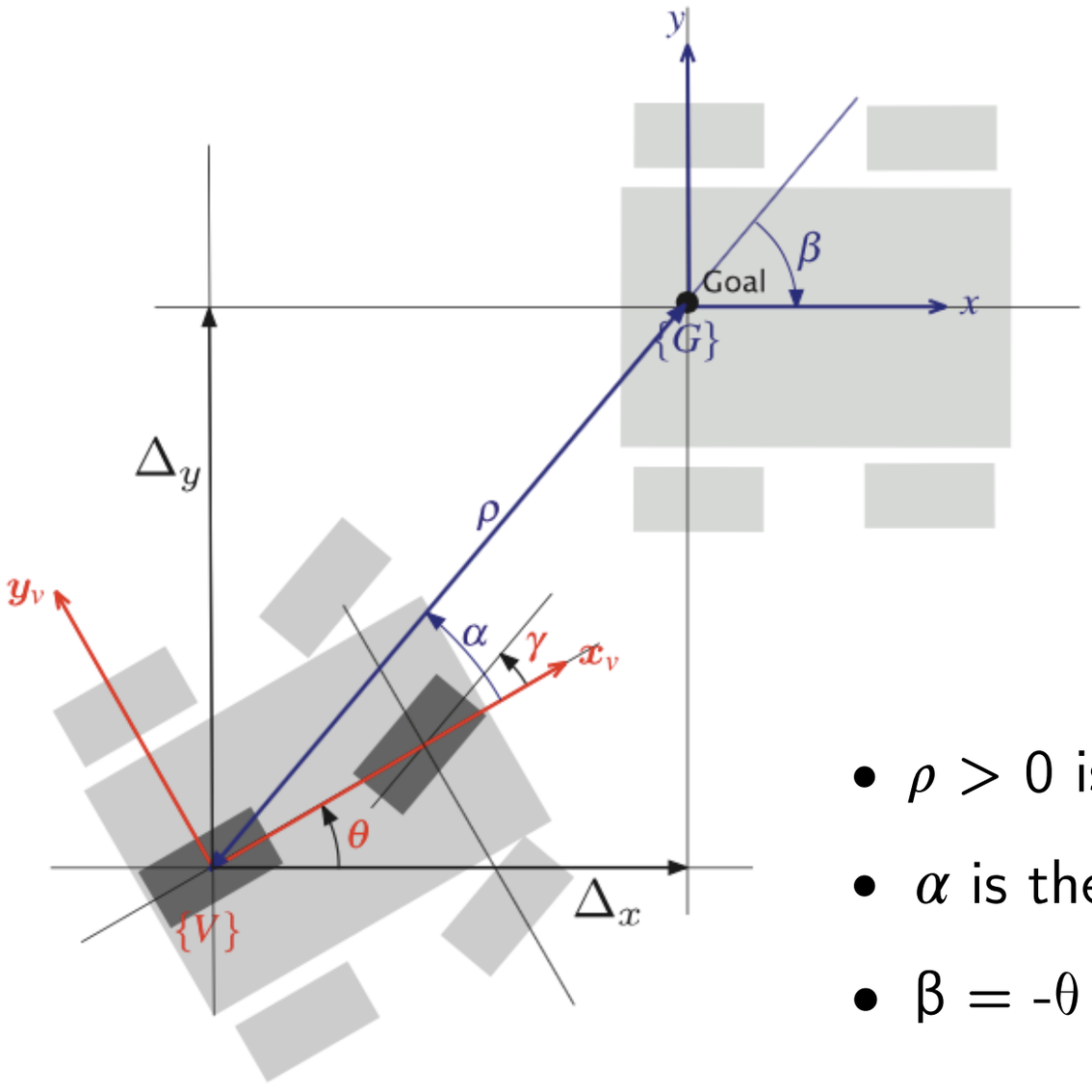
MOVING TO A POSE (P CONTROLLER)

Goal: Move to a specific *pose* $^{W}(x^*, y^*, \theta^*)$ in the plane

Control inputs: v(t) and γ(t)

Current known state: Current pose [x y θ](t) and goal pose (in the {W} frame)

Error vector: Distance from the goal, orientation with respect to the goal pose orientation



Without loss of generality, the goal pose {G} is assumed to be coinciding with the world inertial frame {W}. Therefore, moving to the goal pose is the same as to make the transformation from frame {V} to frame {G}

Polar coordinates are convenient to represent the transformation between {V} and {G}:

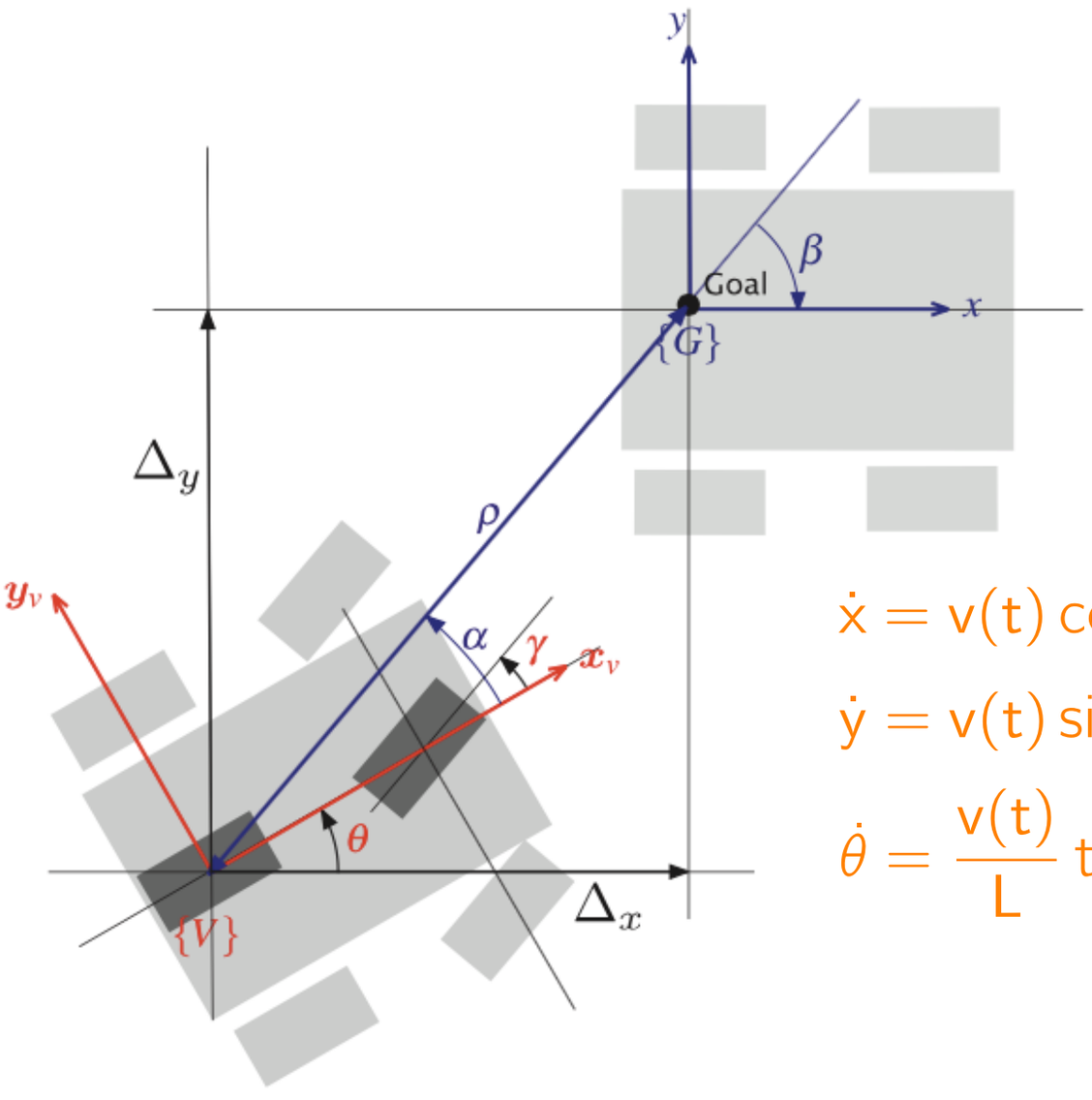
- $\rho > 0$ is the the error distance between robot and goal state
- α is the the orientation of the vector $\vec{\rho}$ wrt {V}
- $\beta = -\theta - \alpha$ is the orientation of the vector $\vec{\rho}$ wrt {G}

MOVING TO A POSE (P CONTROLLER)

Control inputs: v(t) and γ(t)

Current known state: Current pose [x y θ](t) and goal pose , {G}

Polar error vector: Distance ρ, orientation of $\vec{\rho}$ wrt to {V} and wrt {G} (by β,α)



Cartesian to Polar coordinates:

$$\begin{bmatrix} \rho \\ \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \sqrt{\Delta_x^2 + \Delta_y^2} \\ \operatorname{atan}\left(\frac{\Delta_y}{\Delta_x}\right) - \theta \\ -\alpha - \theta \end{bmatrix}$$

$$\dot{x} = v(t)\cos(\theta(t))$$

$$\dot{y} = v(t)\sin(\theta(t))$$

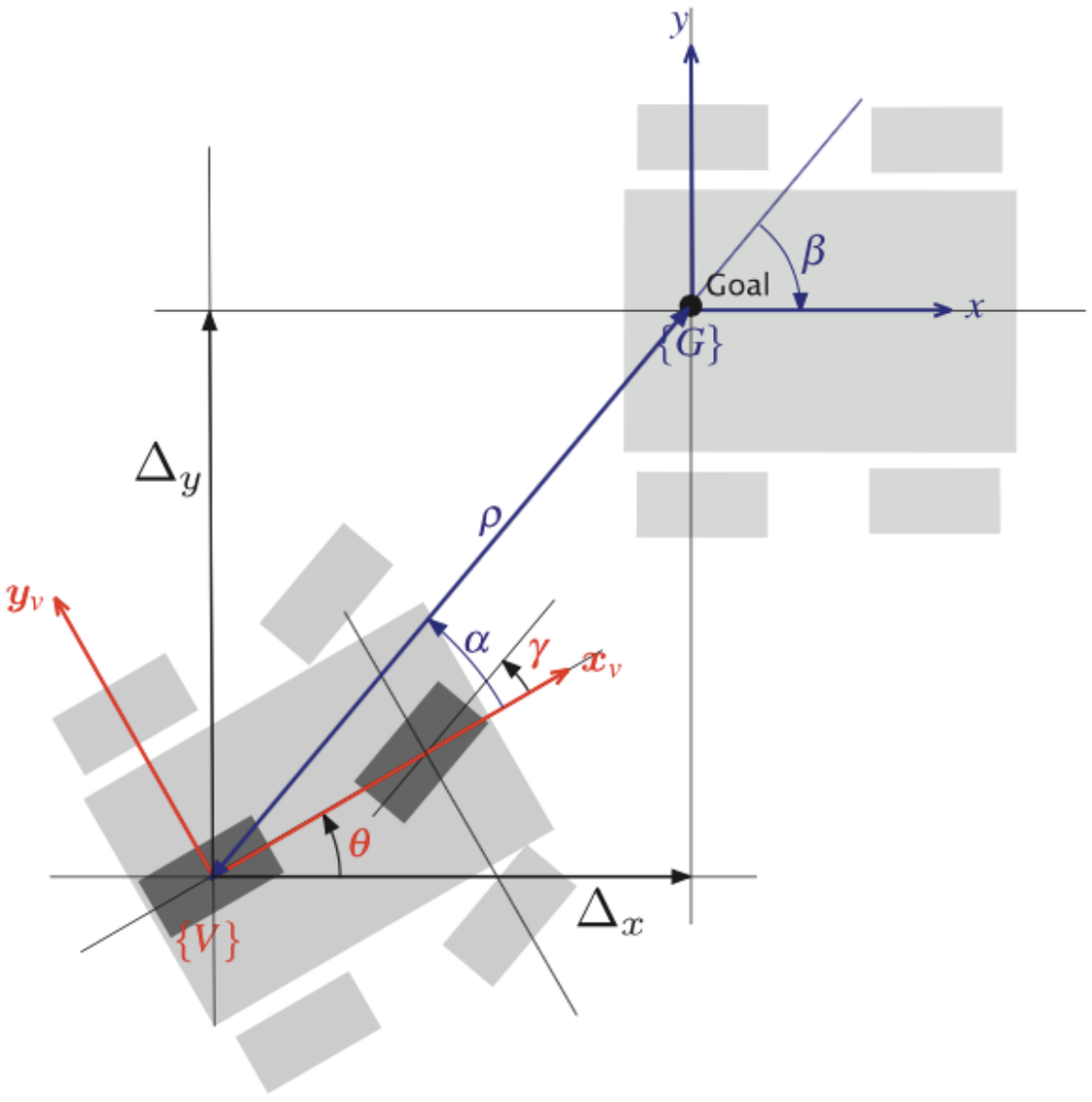
$$\dot{\theta} = \frac{v(t)}{L}\tan\gamma(t) = \omega$$

$$\begin{bmatrix} \dot{\rho} \\ \dot{\alpha} \\ \dot{\beta} \end{bmatrix} = \begin{bmatrix} -\cos(\alpha) & 0 \\ \frac{\sin(\alpha)}{\rho} & -1 \\ -\frac{\sin(\alpha)}{\rho} & 0 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}$$

for {G} in front of {V} $\leftrightarrow$ $\alpha \in \left(-\frac{\pi}{2}, \frac{\pi}{2}\right]$

for {G} not in front of {V} $\mapsto$ the forward direction is redefined by setting v = -v, ω = -ω, and all matrix coefficients change sign

MOVING TO A POSE (P CONTROLLER)



$$\begin{bmatrix} \dot{\rho} \\ \dot{\alpha} \\ \dot{\beta} \end{bmatrix} = \begin{bmatrix} -\cos(\alpha) & 0 \\ \frac{\sin(\alpha)}{\rho} & -1 \\ -\frac{\sin(\alpha)}{\rho} & 0 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}$$

for {G} in front of {V} $\leftrightarrow$ $\alpha \in \left(-\frac{\pi}{2}, \frac{\pi}{2}\right]$

Control laws on v and γ that aim to bring to zero the polar coordinates, meaning {V} $\equiv$ {G}

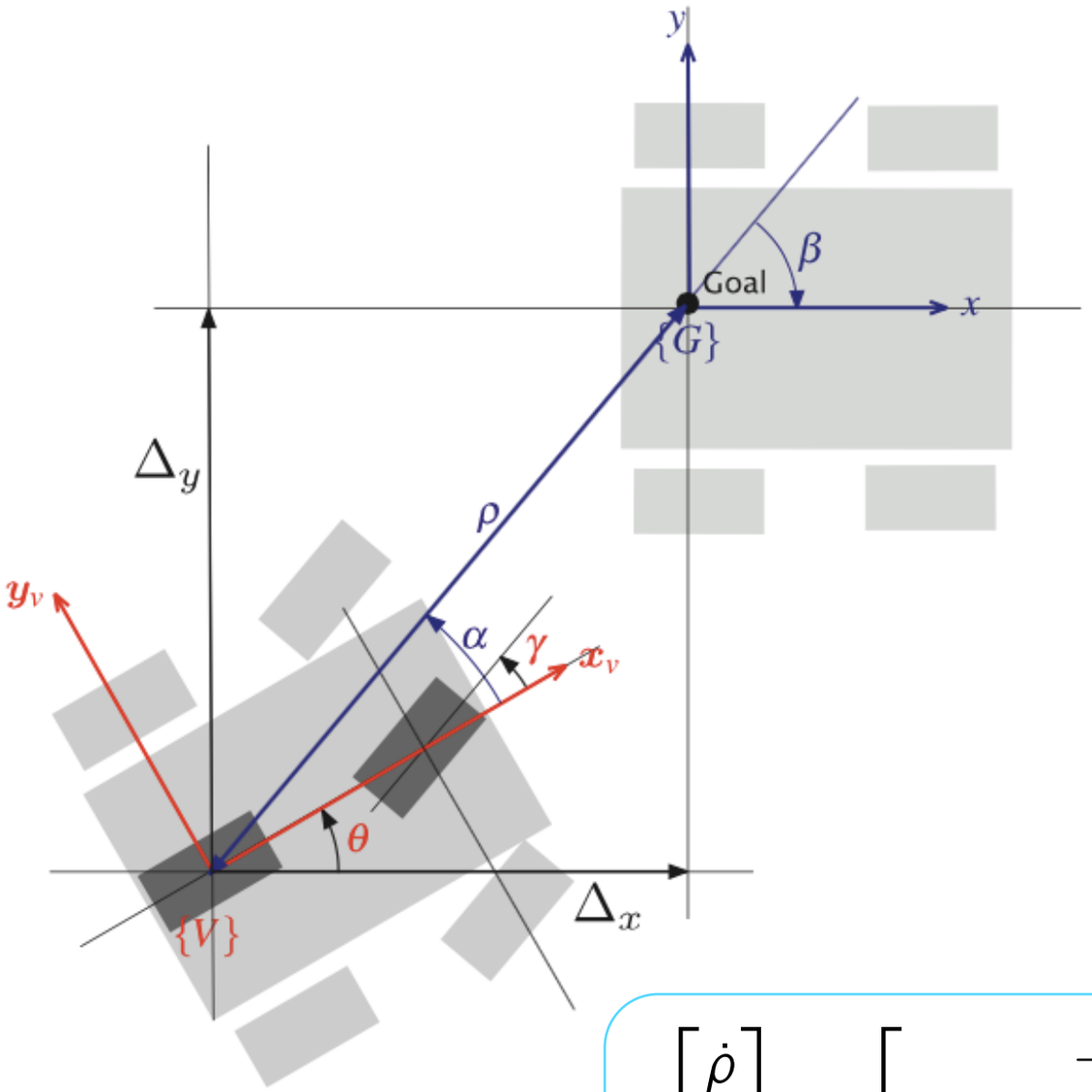
$$v(t) = K_\rho \rho(t)$$

$$\gamma(t) = K_\alpha \alpha(t) + K_\beta \beta(t)$$

For reaching a generic goal pose, the following transformation can be used:

$x' = x - x^*,\ y' = y - y^*,\ \theta' = \theta,\ \beta = \beta' + \theta^*$

MOVING TO A POSE (P CONTROLLER)



What are good values for the gains?

Is the dynamic system stable?

Does it converge?

$$v(t) = K_\rho \rho(t)$$

$$\gamma(t) = K_\alpha \alpha(t) + K_\beta \beta(t)$$

$$\begin{bmatrix} \dot{\rho} \\ \dot{\alpha} \\ \dot{\beta} \end{bmatrix} = \begin{bmatrix} -\cos(\alpha) & 0 \\ \frac{\sin(\alpha)}{\rho} & -1 \\ -\frac{\sin(\alpha)}{\rho} & 0 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}$$

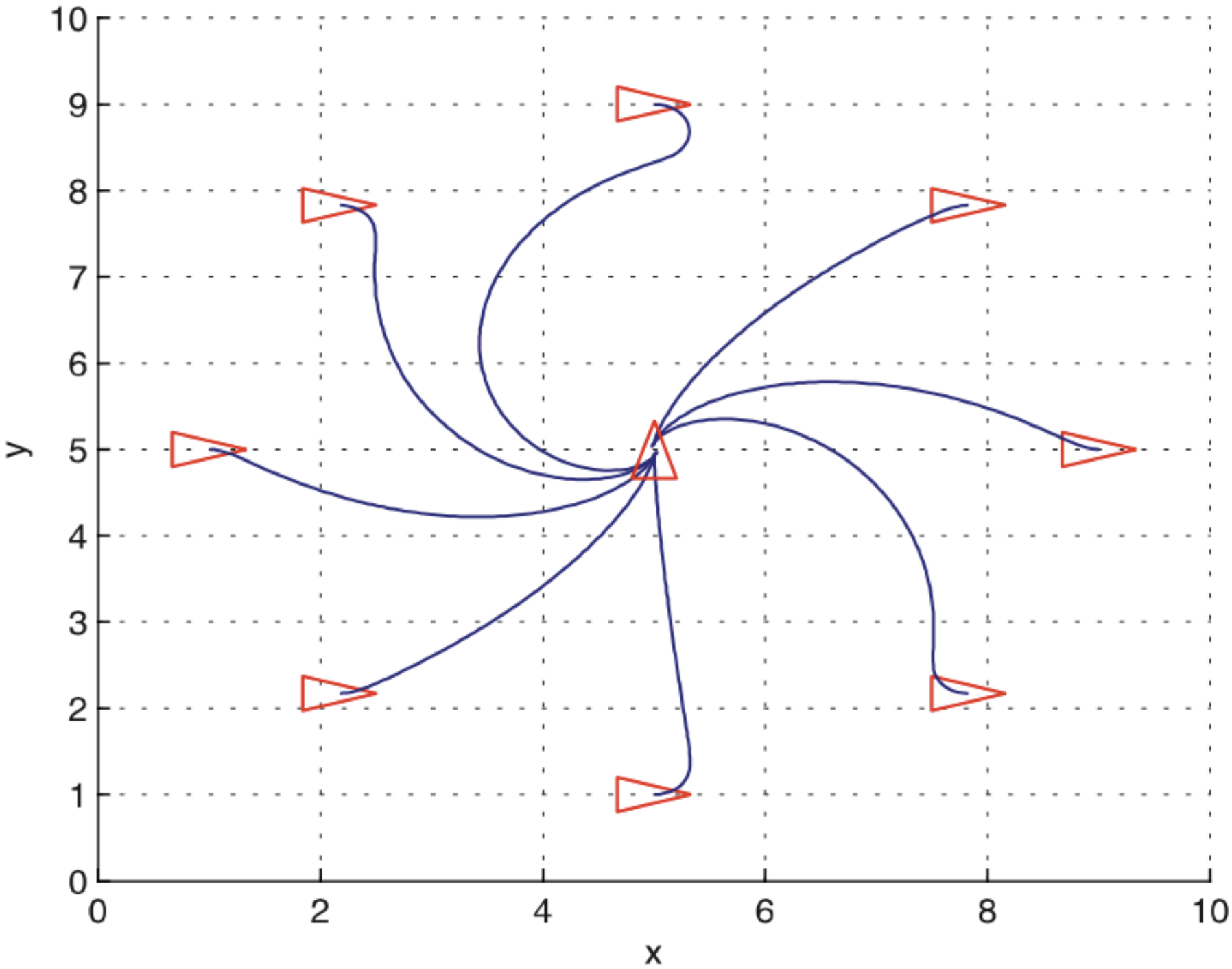
$$\begin{bmatrix} \dot{\rho} \\ \dot{\alpha} \\ \dot{\beta} \end{bmatrix} = \begin{bmatrix} -K_\rho \rho \cos(\alpha) \\ -K_\rho \sin(\alpha) - K_\alpha \alpha - K_\beta \beta \\ -K_\rho \sin(\alpha) \end{bmatrix}$$

Dynamic system

Asymptotically stable as long as:

$$K_\rho > 0, \quad K_\beta < 0, \quad K_\alpha - K_\rho > 0$$

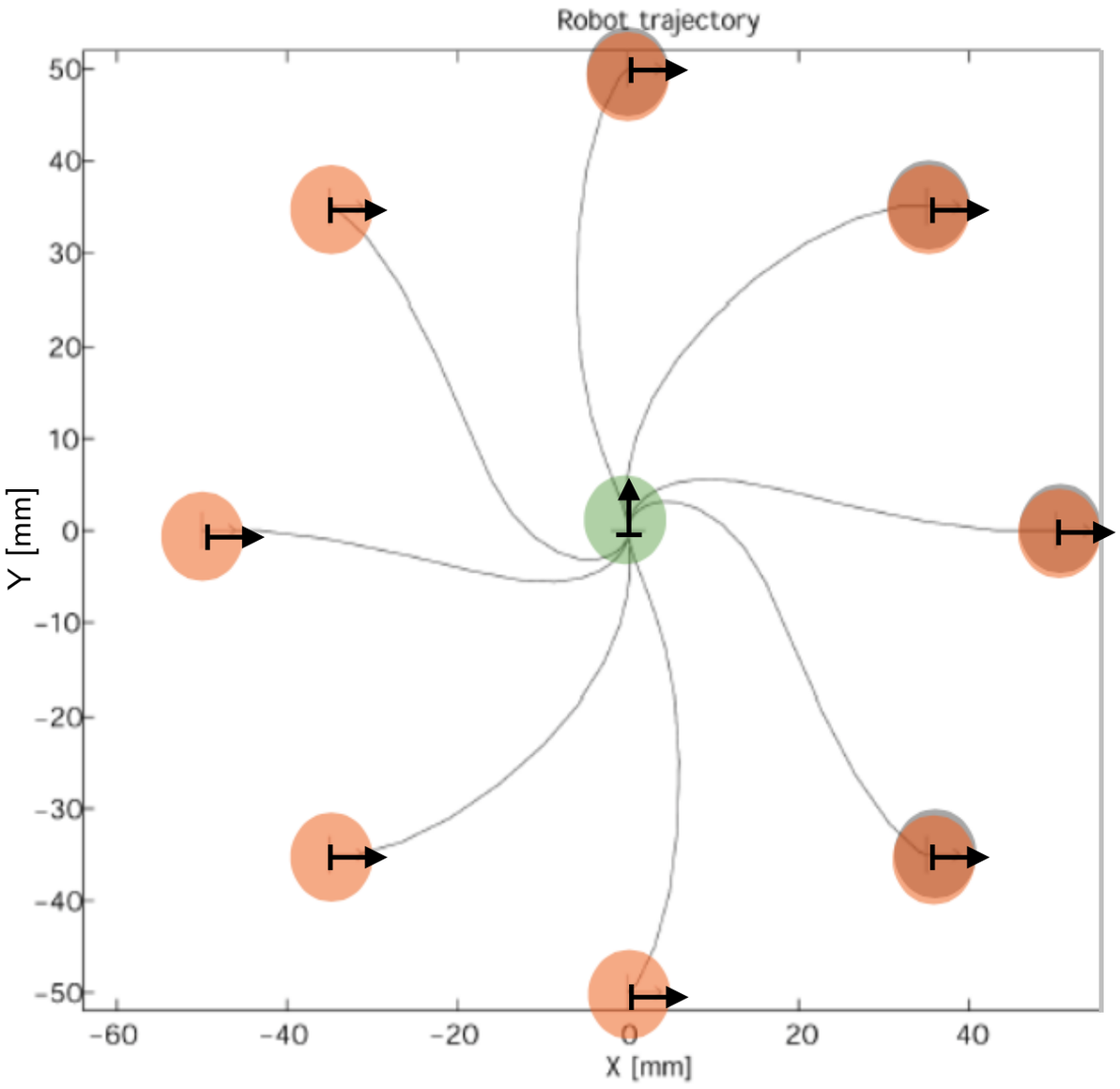
MOVING TO A POSE (P CONTROLLER)

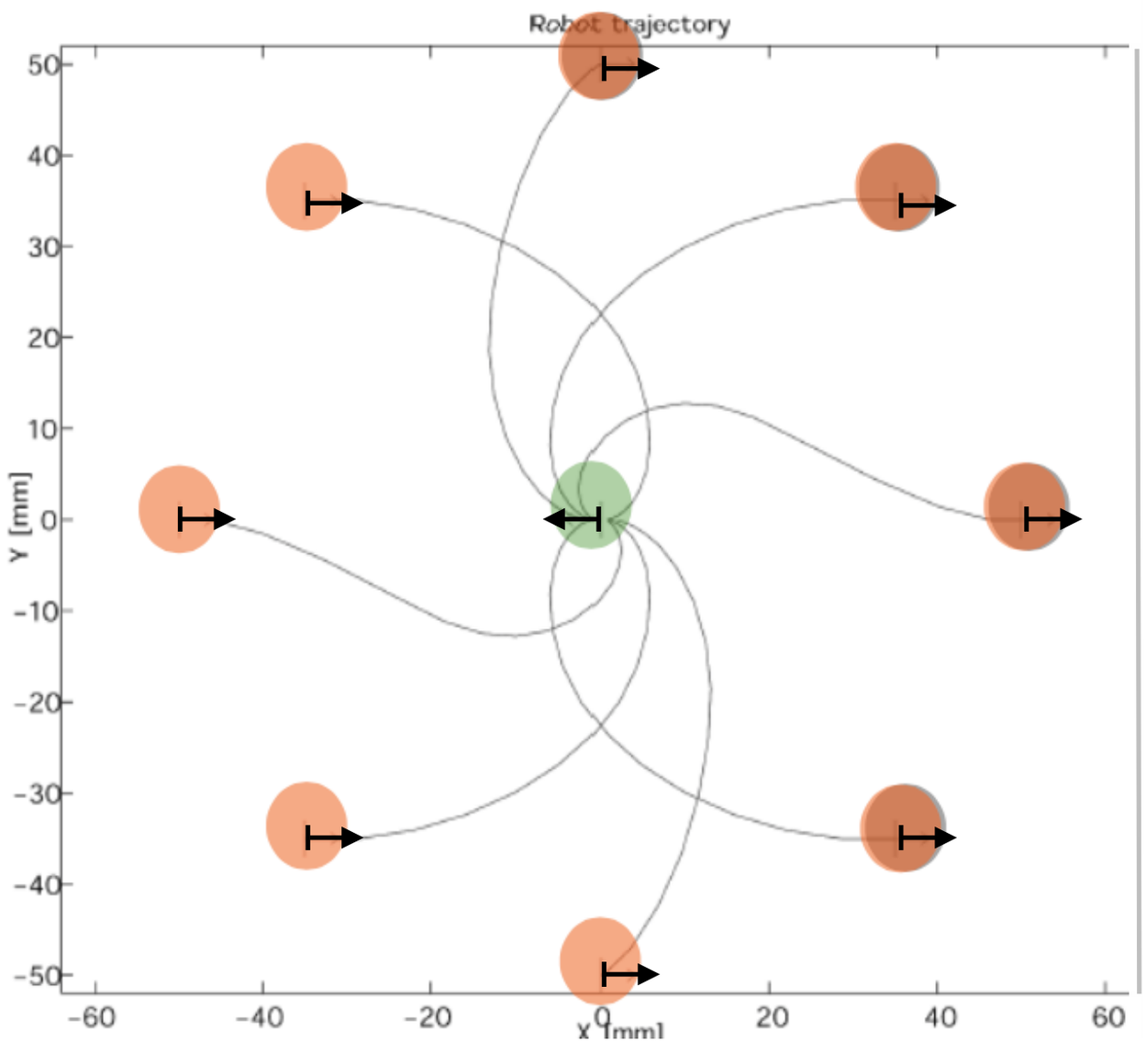


In some cases the robot has *backed into* the goal pose

$$K_\rho = 3, \quad K_\beta = -3, \quad K_\alpha = 8$$

MOVING TO A POSE (P CONTROLLER)





Different goal orientations produce
rather different trajectories

$$K_\rho = 3, \quad K_\beta = -1.5, \quad K_\alpha = 8$$

ISSUES AND NEXT STEPS....

- The systems describing robot and error dynamics used to solve inverse general kinematics problems (with feedback-based control) are *not linear*

- Feedback-based approaches require setting a number of gain parameters when adopting a PID controller: *how do we set the gain values?*

- Is the (linear) *PID approach appropriate*, in general, for the type of inverse kinematics problems we are dealing with?

- How do we get some *guarantees* about the trajectory followed by the robot: does the robot converge to the desired setpoint? Once reached, is the setpoint stable? Will convergence happen in finite time?

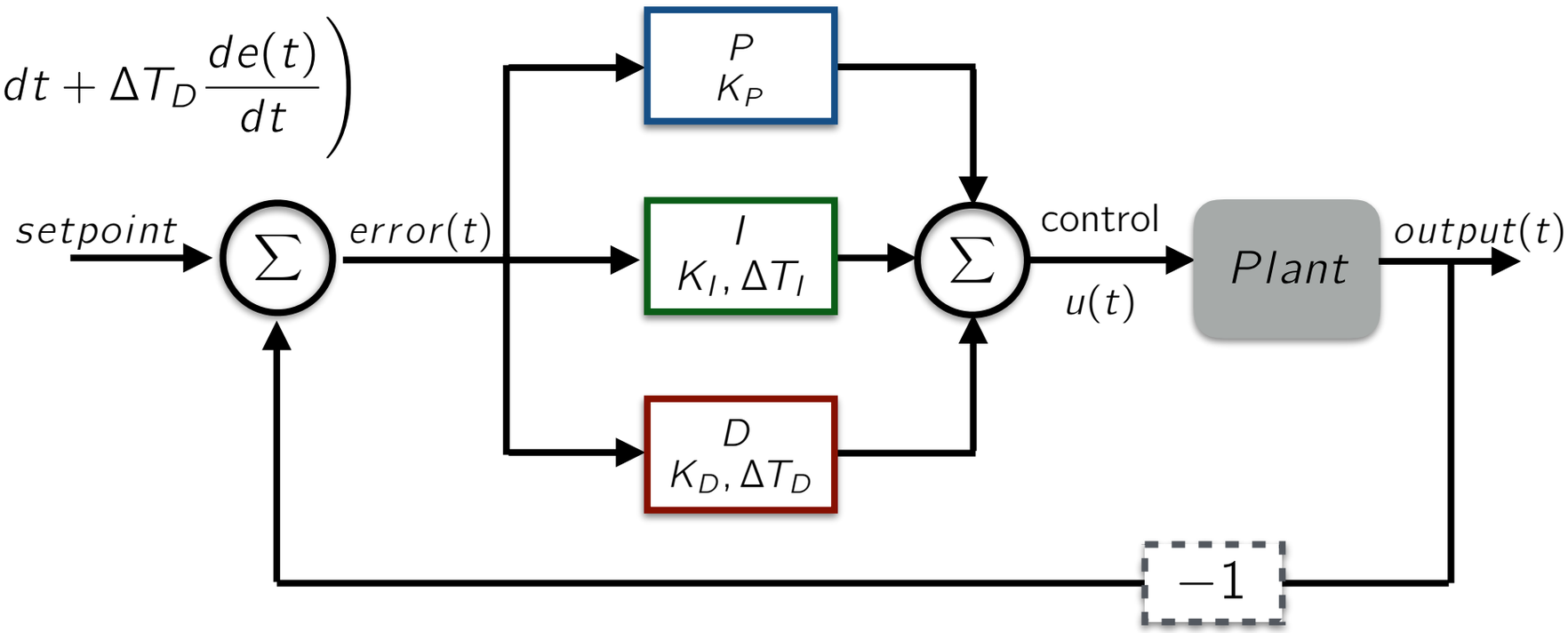
Next steps:

1. PIDs more in depth: properties and parameter tuning
2. Overview (brief) about control and equilibrium in dynamic systems
3. Linearization techniques

LIMITATIONS OF PID CONTROLLERS

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

$$u(t) = K_p \left(e(t) + \frac{1}{\Delta T_I} \int_0^t e(t)dt + \Delta T_D \frac{de(t)}{dt} \right)$$



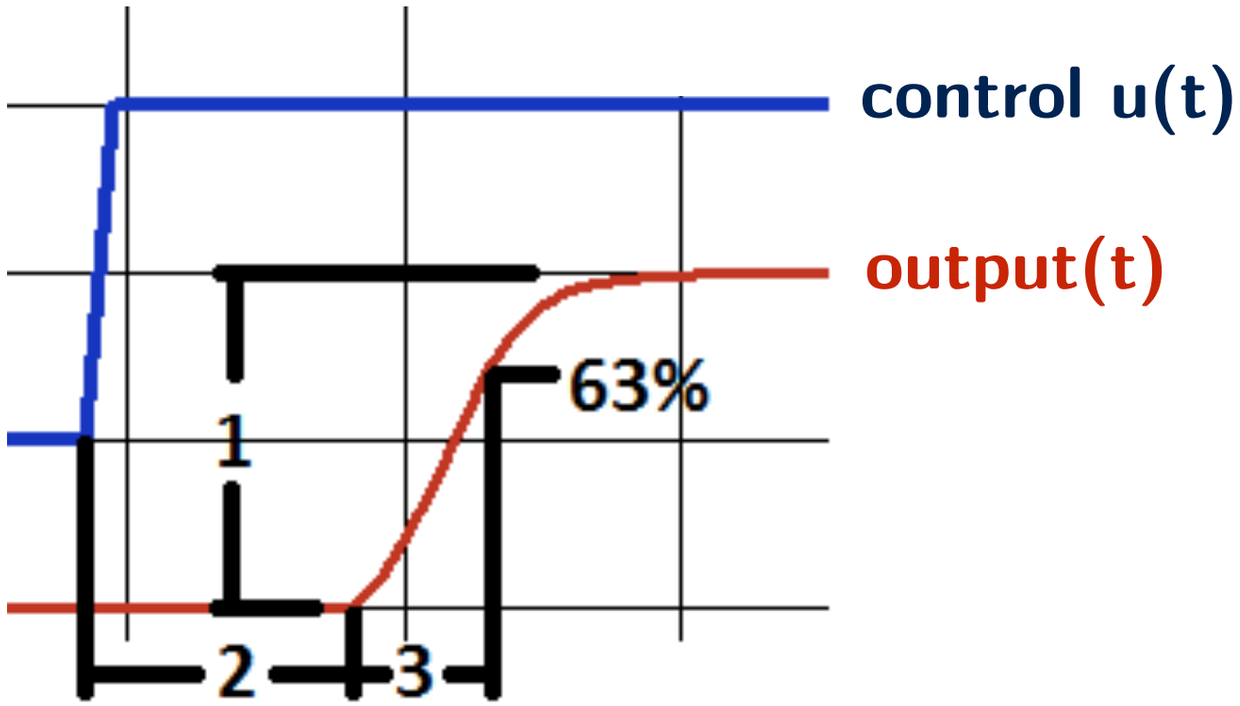
- Work without the help/need of a model of the dynamics of the system/plant being controlled → **Potentially (usually) sub-optimal**

- Provide a feedback-based linear control, since the control signal *u(t)* is linear in the error (and in linear operators of the error)
  → **In general, sub-optimal when plant's dynamics is *not* linear:**

$$o(t) = A(t)o(t-1) + u(t) + \nu(t) \quad \text{Goal:} \quad |s(t) - o(t)| \to 0$$

$$\Rightarrow u(t) \approx s(t) - A(t)o(t-1)$$

PID'S RESPONSE FIGURES



Response factors:

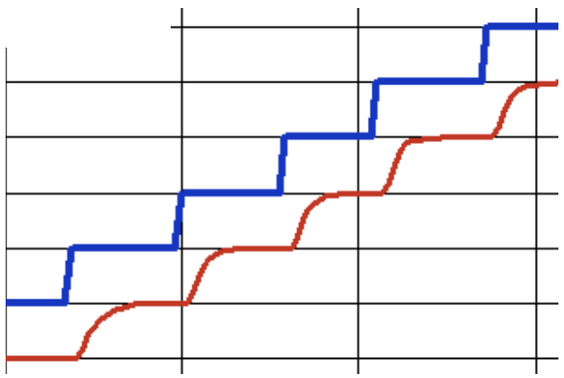
1. ***Step response gain:*** Amount of *output* change resulting from a step unit change in control *u(t)*

2. ***Step response time:*** Time from when *u(t)* changes until the change starts to be seen in *output(t)*

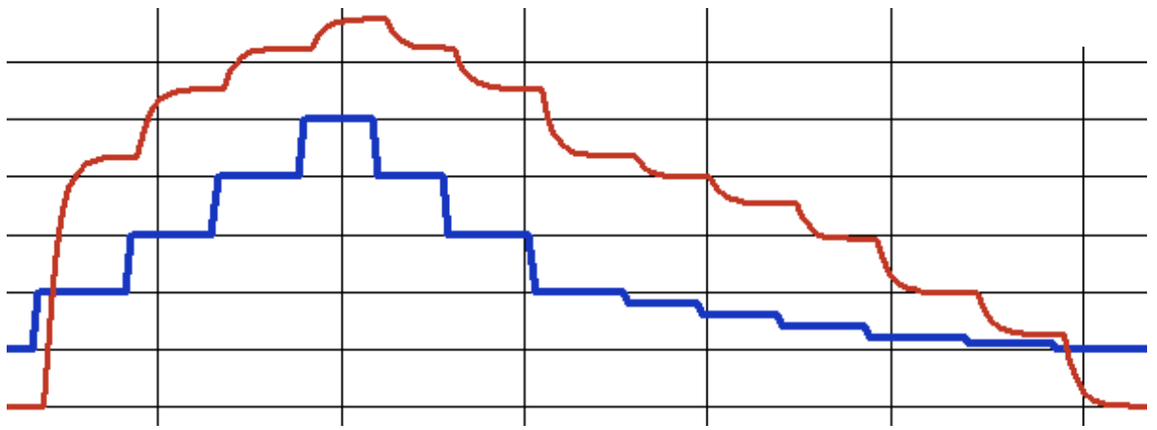
3. ***Step characteristic time:*** The time for *output(t)* to reach its new level. Because the output usually approaches its new level asymptotically, the *characteristic time constant* = time it takes the system's step response to reach (-1)1/e, or about 63% of the distance from the initial to its final value

PID'S RESPONSE FIGURES

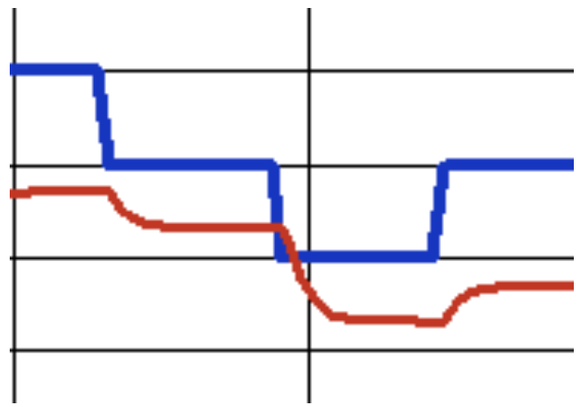
output(t) **control u(t)**



Linear



Non-Linear



With Hysteresis

Response type:

1. ***Linearity of response:*** The same change in *u(t)* through the whole scale results in a similar change in *output(t)* at each point

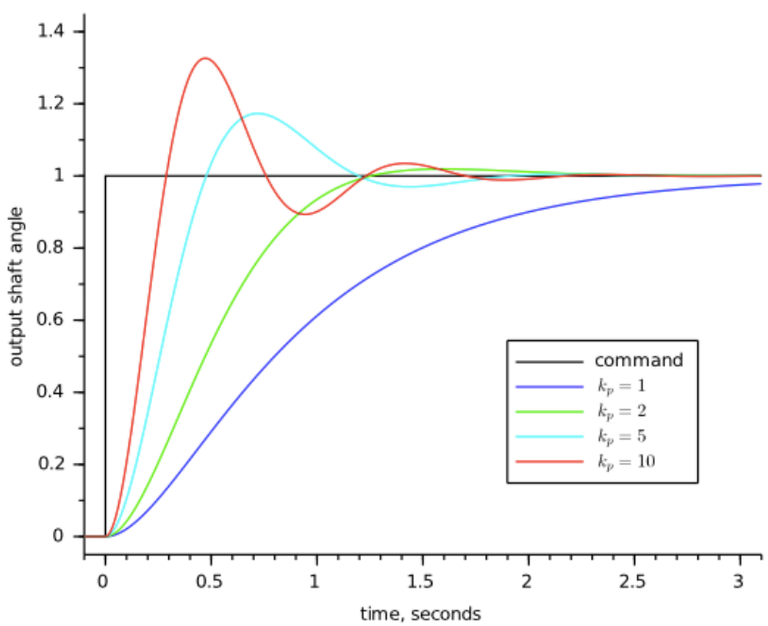
2. ***Non Linearity of response:*** A change on one part of the *u(t)* range results in more *output(t)* change than the same *u(t)* change in a different range

3. ***Hysteresis of response:*** A different *output(t)* is produced for the same control input *u(t)* depending on "history", whether the *u(t)* went up or down right before

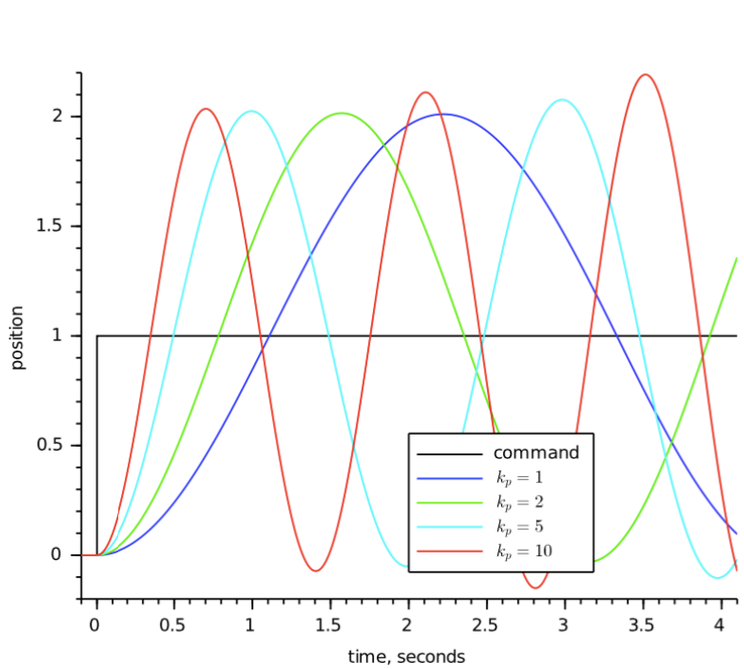
TUNING PID'S GAINS: P GAIN

- **Proportional gain:** Contributes to the control signal *u(t)* with a value proportional to *e(t)*, or, equivalently to *output(t)* (it's just a matter of scale given that the *(setpoint - output(t))* is linear). Corrects *u(t)* based on upsets as they happen.

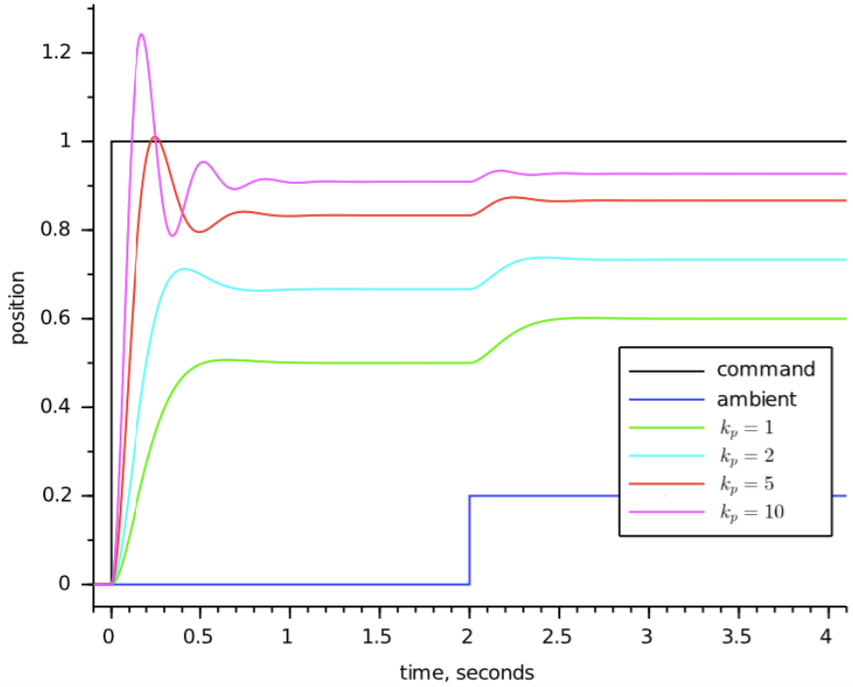
- **High gain:** Strong reaction to error, Lower steady-state error, Higher overshot
- **Low gain:** Less sensitivity, High steady-state error



Motor shaft control by DC
Short step response time



Precision actuator
Long step response time,
Non-linear dynamics

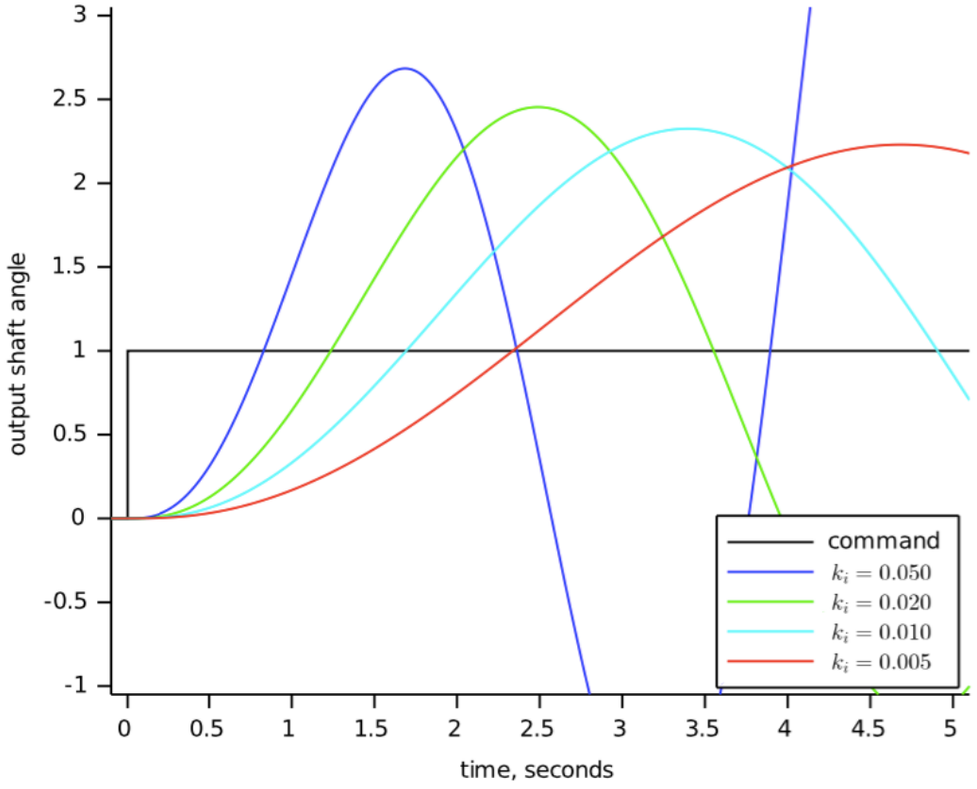


Device temperature control
Short step response time,
Steady-state error, u(t) not enough
External disturbance

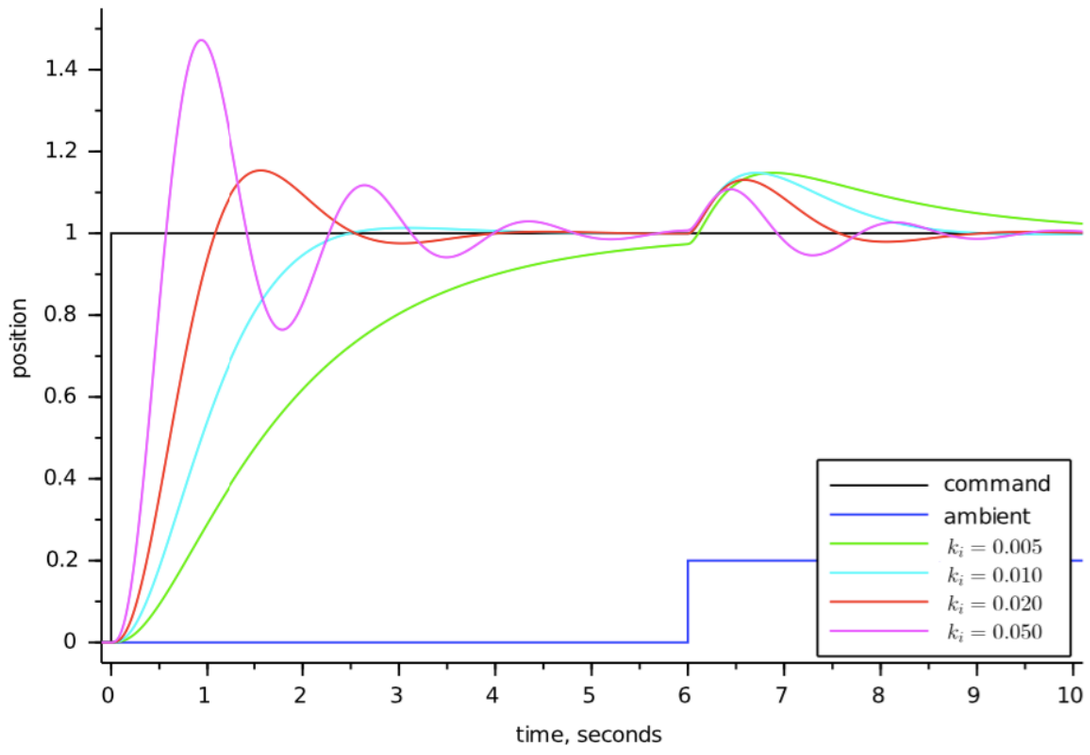
TUNING PID'S GAINS: I GAIN

- **Integral gain:** Removes steady-state error by integrating the error over time. It is used to add long-term precision to a control loop, and it is almost always used in conjunction with proportional control. Alone, it usually doesn't drive the system to stability since it brings "inertia"

- **High gain:** More oscillatory, Faster reduction of SS error
- **Low gain:** Less oscillatory, Slower reduction of SS error



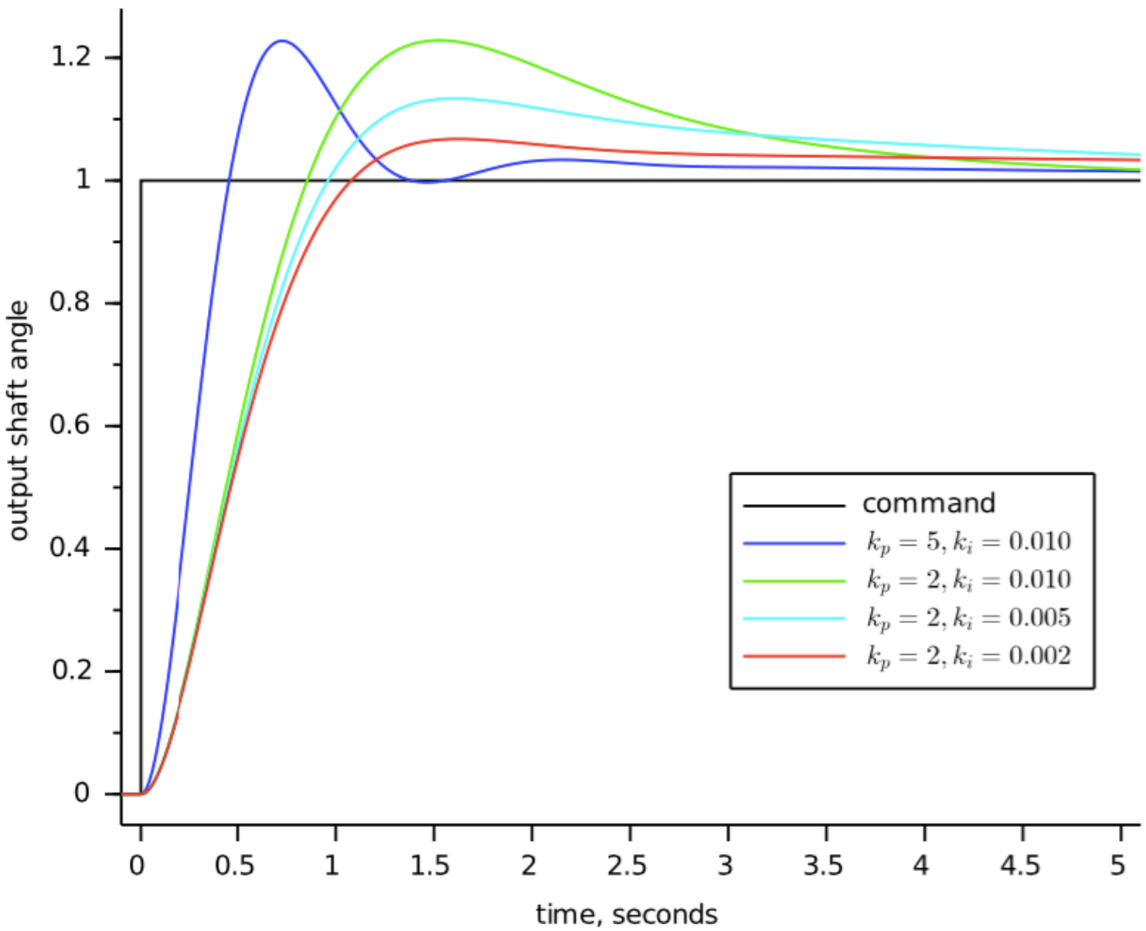
Motor shaft control by DC



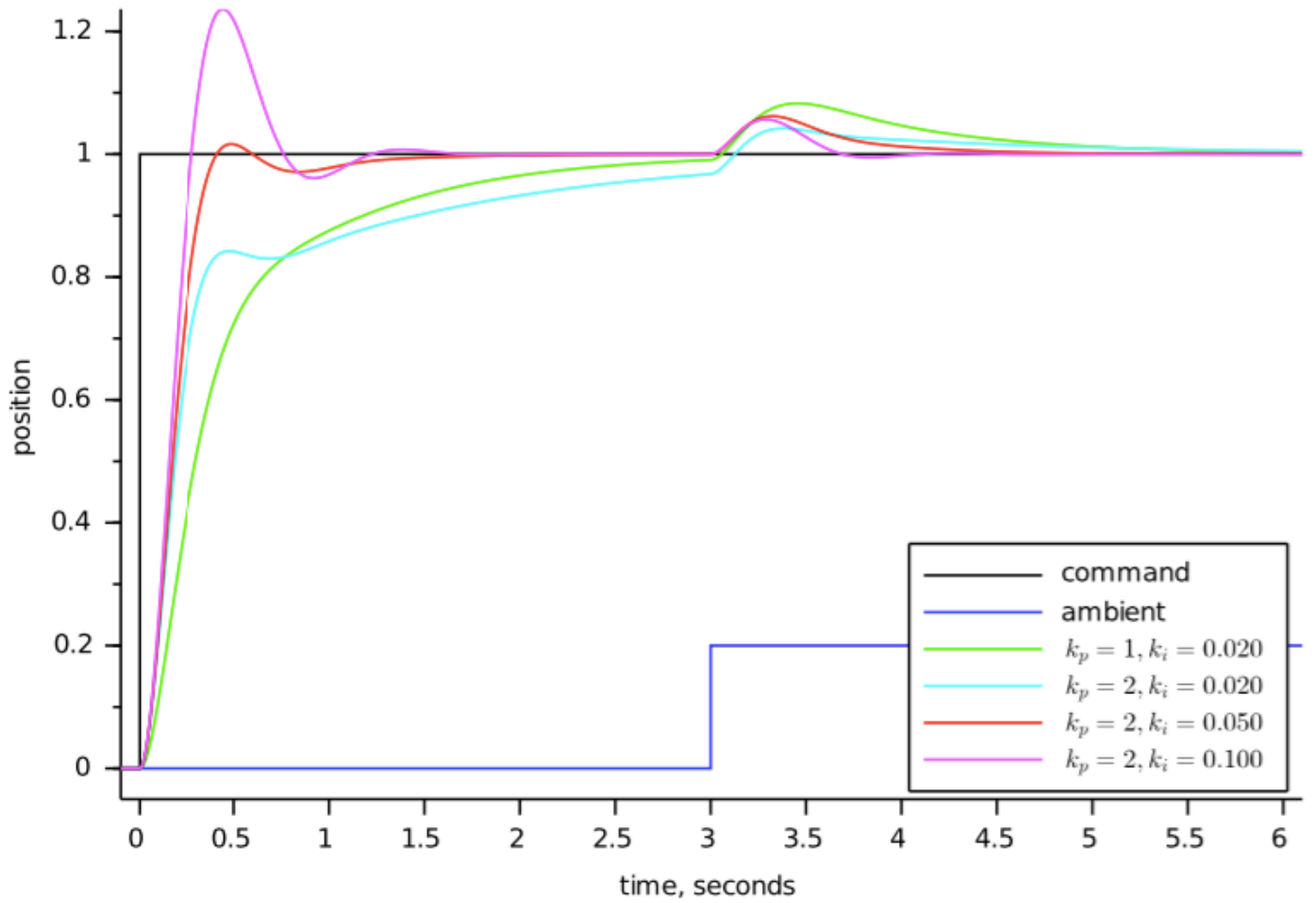
Device temperature control

TUNING PID'S GAINS: P-I GAIN

- **Integral gain + Proportional Gain:** React fast (P) and let the steady-state error going to zero (I), but overall hits the setpoint later than when using P



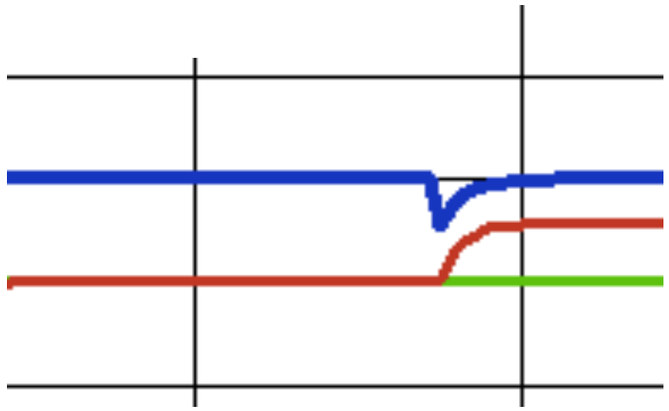
Motor shaft control by DC

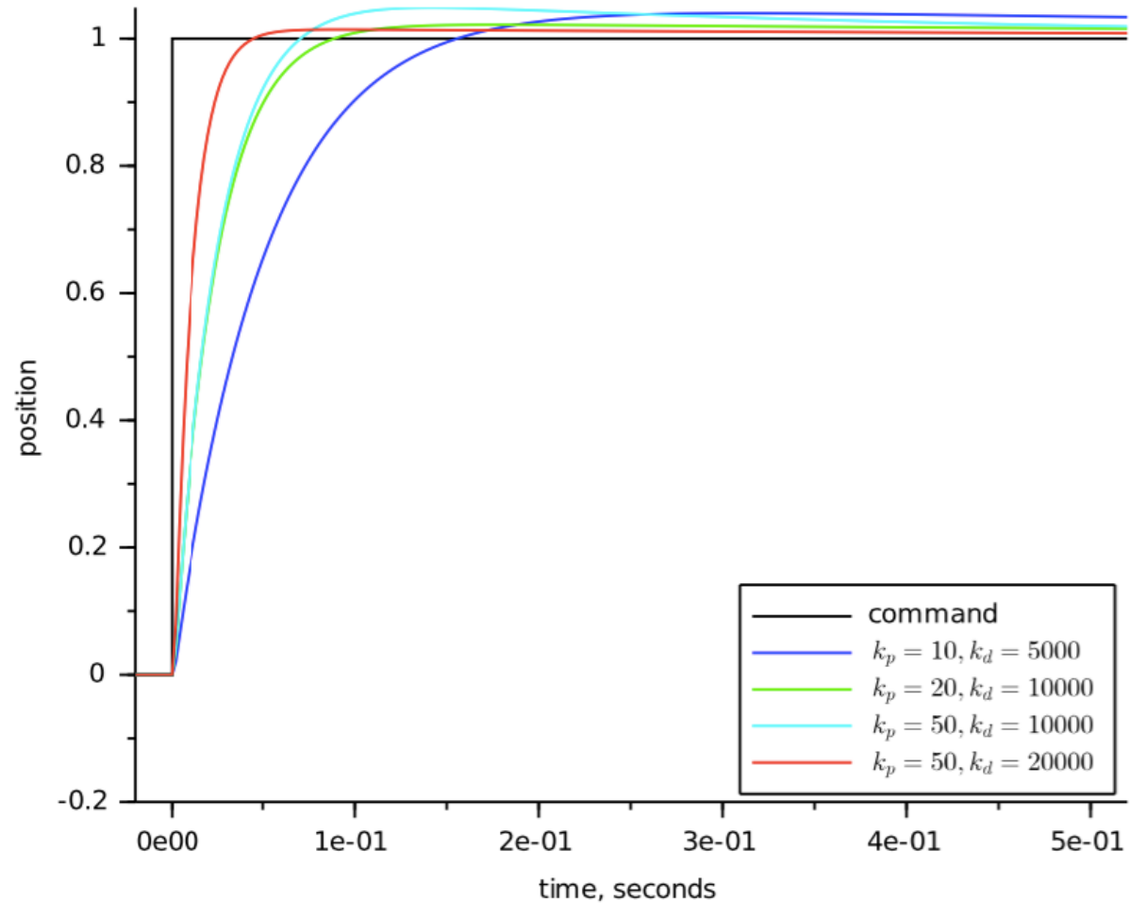


Device temperature control

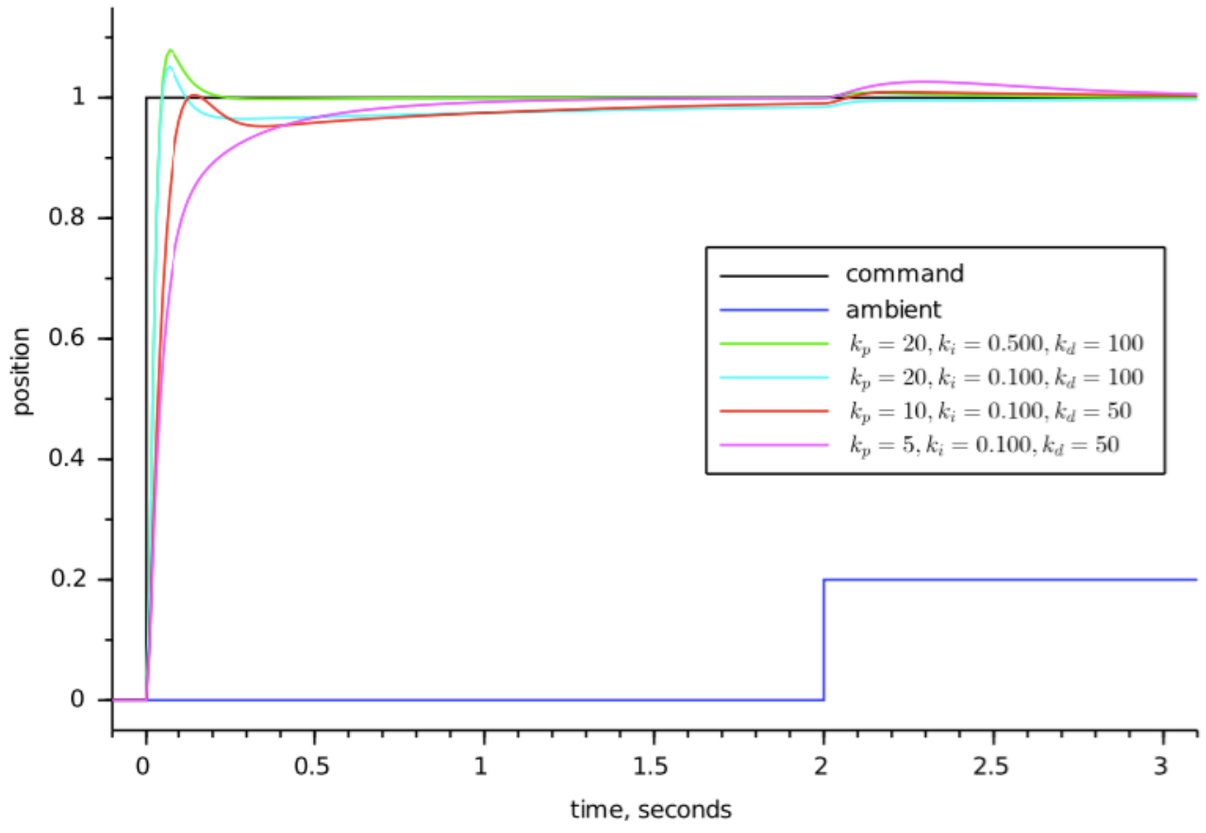
TUNING PID'S GAINS: D GAIN

- **Differential gain:** It is also called *preact* because it allows the loop to "anticipate" upsets as they begin to happen, spot by a rate of change in the error, and then react quickly.
- Reduces the oscillatory behavior and overshot of the response.





Precision actuator with PD



Temperature control with PID

HEURISTICS FOR P,I,D TUNING

TABLE 1 **Effects of independent P, I, and D tuning on closed-loop response.**
For example, while K_I and K_D are fixed, increasing K_P alone can decrease rise time, increase overshoot, slightly increase settling time, decrease the steady-state error, and decrease stability margins.

	Rise Time	Overshoot	Settling Time	Steady-State Error	Stability
Increasing K_P	Decrease	Increase	Small Increase	Decrease	Degrade
Increasing K_I	Small Decrease	Increase	Increase	Large Decrease	Degrade
Increasing K_D	Small Decrease	Decrease	Decrease	Minor Change	Improve

Challenge:

Their effects are intertwined, they need to be set altogether!

(One typical, manual) Heuristic recipe:

1. Start with a P controller, and set P gain to a value K_P that's low enough to prevent appearing of oscillations in system's response.
2. Add derivative D, that will help to damp oscillations when P gain will be increased (to get faster response). Start with $K_P = 100K_P$, if not oscillations appear, increase it, until oscillations happen, or, if oscillations are there, decrease it until they disappear.
3. Adjust P's gain to possibly increase it (by a factor 2 or 3), until oscillations appear.
4. Start the I gain setting it to about 1/100 of P's gain. With oscillations, decrease K_I; with no oscillations, increase it until oscillations happen.
5. ... stop! $\rightarrow$ Try it out ...

TWIDDLE HEURISTIC (~COORDINATE ASCENT)

```
# Choose an initialization parameter vector
p = [0, 0, 0]
# Define potential changes
dp = [1, 1, 1]
# Calculate the error
best_err = A(p)

threshold = 0.001

while sum(dp) > threshold:
    for i in range(len(p)):
        p[i] += dp[i]
        err = A(p)

        if err < best_err:  # There was some improvement
            best_err = err
            dp[i] *= 1.1
        else:  # There was no improvement
            p[i] -= 2*dp[i]  # Go into the other direction
            err = A(p)

            if err < best_err:  # There was an improvement
                best_err = err
                dp[i] *= 1.05
            else  # There was no improvement
                p[i] += dp[i]
                # As there was no improvement, the step size in either
                # direction, the step size might simply be too big.
                dp[i] *= 0.95
```

A is a black-box returning an error (e.g., a PID controller connected to plant)

The output of twiddle is a *local minimum* in PID parameter space

Any global heuristic optimization algorithm can be used. Good options include: *Genetic Algorithms*, *Particle Swarm Optimization*, *Tabu Search* ...

Gain tuning of position domain PID control using particle swarm optimization
Pano, V., Ouyang, P., Robotica, 09/2014, Volume 760, Issue 6

CONTROLLABILITY OF A DYNAMICAL SYSTEM

Time-invariant dynamical system with *m control inputs* **u**

$$\dot{\mathbf{x}} = f\big(\mathbf{x}(t), \mathbf{u}(t)\big), \quad \mathbf{x} \in \mathbb{R}^n, \ \mathbf{u} \in \mathbb{R}^m$$

Control inputs are defined according to a **feedback law**: $\boldsymbol{u}(t) = K\boldsymbol{x}(t)$, K is an $n \times m$ feedback Gain matrix

Controllability: Any initial state $\boldsymbol{x}(0)$ can be steered to any final state $\boldsymbol{x}^1$ at a finite time t_1 based on the inputs from the feedback law.

For a robot: All configurations can be achieved in finite time from a given initial configuration.

Note: *The trajectory between 0 and t_1 is not specified*

For linear dynamical systems

$$f\big(\mathbf{x}(t), \mathbf{u}(t)\big) = A\mathbf{x}(t) + B\mathbf{u}(t)$$

algebraic criteria for controllability are available:

$C = [\ B\ AB\ A^2B\ \cdots\ A^{n-1}B\ ]$, $\quad \text{rank}(C) = n$ (C has full rank)

For non-linear dynamical systems, general controllability criteria are not available!

Local (in space and time) notions of controllability are employed

STABILITY OF A DYNAMICAL SYSTEM

Equilibrium: A state $\boldsymbol{x}^e$ of is said to be an *equilibrium state* if and only if

$$\boldsymbol{x}^e = \boldsymbol{x}(t;\, \boldsymbol{x}^e,\, u(t)=0) \text{ for all } t \geq 0.$$

If a trajectory reaches an equilibrium state and if no input is applied the trajectory will stay at the equilibrium state forever (internal system's dynamics doesn't move the system away from the equilibrium point)

For a linear system the zero state is a always an equilibrium state

Stable equilibrium: An equilibrium state $\boldsymbol{x}^e$ is said to be *stable* if and only if for any positive ε, there exists a positive number $\delta(\varepsilon)$ such that the inequality

$$||\boldsymbol{x}(0) - \boldsymbol{x}^e|| \leq \delta$$

Lyapunov stability

implies that $||\boldsymbol{x}(t;\, \boldsymbol{x}(0),\, u(t)=0) - \boldsymbol{x}^e|| \leq \varepsilon$ for all $t \geq 0$.

An equilibrium state $\boldsymbol{x}^e$ is stable if the response following after starting at any initial state $x(0)$ that is sufficiently near to $\boldsymbol{x}^e$ will not move the state far away from $\boldsymbol{x}^e$.

Asymptotically stable equilibrium: If the equilibrium $\boldsymbol{x}^e$ is Lyapunov-stable and if every motion starting sufficiently near to $\boldsymbol{x}^e$ converges (go back) to $\boldsymbol{x}^e$ as $t \to \infty$.

TO BE CONTINUED ...

Stability properties of linear systems

Linearization of previous control systems

Stability domain for feedback-based gains

Other types of controllers?