Position Control

After looking at attitude control, we can cascade velocity and position controllers to provide the attitude setpoints. The position controller outputs velocity commands to the velocity controller, which in turn provide thrust vector command. The thrust command is transformed to attitude setpoint and fed to attitude controller. Finally the attitude controller outputs rates commands to the rates controller which feeds torques and thrust values to the mixer and motor controllers.

Position control

- P controller only
- Output is fed as velocity set point to velocity controllers
- Output is limited by maximum allowed velocity

$$V_{\text{sp}} = \begin{bmatrix} k_{\text{px}} \\ k_{\text{py}} \\ k_{\text{pz}} \end{bmatrix} \left(\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} - \begin{bmatrix} x \\ y \\ z \end{bmatrix} \right)$$

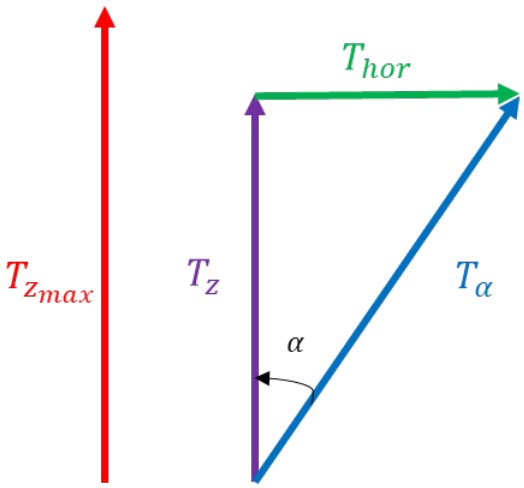
Velocity control

- Output is the thrust setpoint vector T_{sp}
- Hover thrust (-m*g) is sent as a Feed-Forward term, in order to allow hover when the position and velocity error are zero
- Desired acceleration (Acc_{sp}) is also a feed-forward term
- Vertical velocity controller gain is different from horizontal (xy) velocity controllers
- For practical implementations, the D term is negative-multipled by the acceleration rather than error in velocity
- For practical implementations, the I term is clipped to avoid winding.

$$T_{\text{sp}} = \text{PID}\left(\begin{bmatrix} v_{\text{xc}} \\ v_{\text{yc}} \\ v_{\text{zc}} \end{bmatrix}, \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} \right) + g * \text{Acc}_{\text{sp}} + \begin{bmatrix} 0 \\ 0 \\ -m.g \end{bmatrix}$$

Limiting tilt angle by saturating horizontal thrust

- The horizontal thrust must be limited according to the maximum allowed tilt angle α. Maximum tilt angle is user defined and depends on the drone hardware, weight and size.



- $T_{z_{\max}}$ is max thrust the drone can produce
- α is maximum allowed tilt angle
- Thrust setpoint from velocity control is (Tx,Ty, Tz)
- **Saturate the horizontalthrust setpoints as follow:**

- Calculate required horizontal thrust at maximum tilt

$$T_{hor} = |\mathrm{T_z}| * \tan(\alpha)$$

- Calculate maximum allowed horizontal thrust (maximum thrust -vertical commanded thrust)

$$T_{hor_{max}} = \sqrt{\left|\left(T_{\mathrm{z_{max}}}\right)^2 - (T_{\mathrm{z}})^2\right|}$$

- Take the minimum of the previous two as the allowed T_{hor}

$$T_{hor} = \min\left(T_{hor}, T_{hor_{max}}\right)$$

- Saturate set points of horizontal thrust $\mathrm{T_h} = \left[T_x, T_y\right]^T$ coming from velocity control if exceed allowed T_{hor} value

$$T_h = \begin{bmatrix} T_x \\ T_y \end{bmatrix} = \begin{cases} \dfrac{T_h}{\|T_h\|} * T_{hor} & if \quad T_h^T * T_h > (T_{hor})^2 \\ T_h & else \end{cases}$$

Thrust and yaw to attitude set point

The thrust vector set point coming from velocity controller along with yaw set point need to be converted to an attitude set point for the attitude controller.

- We will look into this conversion for quaternion-based attitude controller.
- This is adapted from PX4 implementation.
- The idea is to poject the thrust vector towards the target position and get the rotation matrix. The rotation matrix is then converted into a quaternion setpoint.

- This projection is for the horizontal components. Vertical thrust equals to the magnitude of the thrust vector.

the procedure is as follows:

- Thrust setpoint from velocity control after saturation is $T_c = [T_x, T_y, T_z]^T$
- ϕ_c is yaw setpoint
- **Calculate attitude set point as follows:**
  - Desired body_z axis direction

$$\vec{z}_b = \frac{T_c}{\|T_c\|}$$

  - Desired body x axis direction after projecting by desired yaw direction

$$\vec{x}_b = \begin{bmatrix} -\sin(\phi_c) \\ cos(\phi_c) \\ 0 \end{bmatrix} \times \vec{z}_b$$

$$\vec{x}_b = \frac{\vec{x}_b}{\|\vec{x}_b\|}$$

  - Desired body y axis direction

$$\vec{y}_b = \vec{z}_b \times \vec{x}_b$$

  - Form the desired rotation matrix

$$R = [\vec{x}_b \quad \vec{y}_b \quad \vec{z}_b]$$

  - Convert to quaternion setpoint

Simulations

Drone parameters

```
clear all;
run("quad_params.m")
```

Full control simulations

```
open_system("four_velocity_position_control.slx")
```