

0 Basics

Vector from point B to P in frame \mathcal{A} : ${}^{\mathcal{A}}r_{BP}$

Reference coordinate System \mathcal{A} : $(e_x^{\mathcal{A}}, e_y^{\mathcal{A}}, e_z^{\mathcal{A}})$

Cartesian Coordinates

Stacked parameters of Position: $\chi_{Pc} = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$

Position Vector: ${}^{\mathcal{A}}r = x e_x^{\mathcal{A}} + y e_y^{\mathcal{A}} + z e_z^{\mathcal{A}} = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$

Cylindric Coordinates

Spherical Coordinates

$\chi_{Pz} = \begin{pmatrix} \rho \\ \theta \\ z \end{pmatrix}$; ${}^{\mathcal{A}}r = \begin{pmatrix} \rho \cos \theta \\ \rho \sin \theta \\ z \end{pmatrix}$ $\chi_{Ps} = \begin{pmatrix} r \\ \theta \\ \phi \end{pmatrix}$; ${}^{\mathcal{A}}r = \begin{pmatrix} r \cos \theta \sin \phi \\ r \sin \theta \sin \phi \\ r \cos \phi \end{pmatrix}$

1 Kinematics

1.1 Linear Velocity

The Velocity of point B relative to point A is given by: \dot{r}_{AB} . There exists a linear mapping $E_p(\chi)$ between velocities \dot{r} and the derivatives of the representation $\dot{\chi}_P$:

$$\dot{r} = E_P(\chi_P) \dot{\chi}_P$$

$$\dot{\chi}_P = E_P^{-1}(\chi_P) \dot{r}$$

For **Cartesian** this is the identity: $E_{Pc}(\chi_{Pc}) = E_{Pc}^{-1}(\chi_{Pc}) = 1$
In **Cylindrical** we get:

$$E_{Pz} = \frac{\partial r(\chi)}{\partial \chi} = \begin{pmatrix} c\theta & -\rho s\theta & 0 \\ s\theta & \rho c\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}; \quad E_{Ps}^{-1} = \begin{pmatrix} -\frac{1}{\rho} s\theta & \frac{1}{\rho} c\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

For **Spherical**:

$$E_{Ps} = \begin{pmatrix} c\theta s\phi & -r s\theta s\phi & r c\theta c\phi \\ s\theta s\phi & r c\theta s\phi & r s\theta c\phi \\ c\phi & 0 & -r s\phi \end{pmatrix} \quad E_{Ps}^{-1} = \begin{pmatrix} \frac{c\theta s\phi}{r} & \frac{s\theta s\phi}{r} & \frac{c\phi}{r} \\ -s\theta/(r s\phi) & c\theta/(r s\phi) & 0 \\ \frac{1}{r} c\theta c\phi & \frac{1}{r} s\theta c\phi & -\frac{1}{r} s\phi \end{pmatrix}$$

1.2 Rotation

Orientation of frame \mathcal{B} with reference to frame \mathcal{A} :

$$\phi_{AB} \in SO(3)$$

Important: There is no numerical equivalent to a position such as "angular position". Instead the orientation can be parametrized in several ways.

Passive Rotation: Passive rotations are transformations between different coordinate frames. ${}^{\mathcal{A}}u = C_{AB} \cdot {}^{\mathcal{B}}u$

Active Rotations: Active rotations contain an operator (i.e. $R \in \mathbb{R}^{3 \times 3}$) to rotate a vector in the same frame. ${}^{\mathcal{A}}v = R \cdot {}^{\mathcal{A}}u$

1.2.1 Rotation Matrix

Mapping Coordinates P from frame \mathcal{B} to \mathcal{A} :

$${}^{\mathcal{A}}r_{AP} = \begin{pmatrix} A e_x^{\mathcal{B}} & A e_y^{\mathcal{B}} & A e_z^{\mathcal{B}} \end{pmatrix} \cdot {}^{\mathcal{B}}r_{AP} = C_{AB} \cdot {}^{\mathcal{B}}r_{AP}$$

C is orthogonal: $C_{BA} = C_{AB}^{-1} = C_{AB}^T \Rightarrow C_{AB}^T C_{AB} = I$

1.2.2 Elementary Rotations

Around X-Axis	Around Y-Axis	Around Z-Axis
$\begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos \phi & -\sin \phi \\ 0 & \sin \phi & \cos \phi \end{pmatrix}$	$\begin{pmatrix} \cos \phi & 0 & \sin \phi \\ 0 & 1 & 0 \\ -\sin \phi & 0 & \cos \phi \end{pmatrix}$	$\begin{pmatrix} \cos \phi & -\sin \phi & 0 \\ \sin \phi & \cos \phi & 0 \\ 0 & 0 & 1 \end{pmatrix}$

1.2.3 Homogeneous Transformation

Combined Translation and Rotation for frames with an offset. The vector ${}^{\mathcal{A}}r_{AB}$ points from the origin of frame \mathcal{A} to the origin of the \mathcal{B} in frame \mathcal{A} . T_{AB} transforms a **point P** from frame \mathcal{B} to \mathcal{A} .

$$\begin{pmatrix} {}^{\mathcal{A}}r_{AP} \\ 1 \end{pmatrix} = \underbrace{\begin{bmatrix} C_{AB} & {}^{\mathcal{A}}r_{AB} \\ 0_{1 \times 3} & 1 \end{bmatrix}}_{T_{AB}} \begin{pmatrix} {}^{\mathcal{B}}r_{BP} \\ 1 \end{pmatrix}$$

$$\text{With inverse: } T_{AB}^{-1} = \begin{bmatrix} \overbrace{C_{AB}^T}^{{}^{\mathcal{B}}r_{BA}} & 0 \\ 0_{1 \times 3} & 1 \end{bmatrix}$$

1.3 Representation of Rotations

1.3.1 Euler Angles

Rotation Matrix from Euler Angles:

$$C_{AD} = C_{AB}(z) C_{BC}(y) C_{CD}(x)$$

Euler Angles from Rotation Matrix:

$$\chi_{R, \text{eulerZYX}} = \begin{pmatrix} z \\ y \\ x \end{pmatrix} = \begin{pmatrix} \text{atan2}(c_{21}, c_{11}) \\ \text{atan2}(-c_{31}, \sqrt{c_{32}^2 + c_{33}^2}) \\ \text{atan2}(c_{32}, c_{33}) \end{pmatrix}$$

Proper Euler Angles: first and last around same axis
Tait Bryan = Roll-Pitch-Yaw = ZYX, Cardan = XYZ

1.3.2 Angle Axis

$$\chi_{R, \text{AngleAxis}} = \begin{pmatrix} \theta \\ \vec{n} \end{pmatrix} \quad \text{Rotation Vector: } \varphi = \theta \cdot \vec{n}$$

Rotation Matrix from Angle Axis:

$$C_{AB} = \cos(\theta) I_{3 \times 3} - \sin(\theta) [\vec{n}]_{\times} + (1 - \cos(\theta)) n n^T$$

Parameters from Rotation Matrix ($\|\vec{n}\| = 1$, singularity $= \pi/2$):

$$\theta = \cos^{-1} \left(\frac{c_{11} + c_{22} + c_{33} - 1}{2} \right); \quad n = \frac{1}{2 \sin(\theta)} \begin{pmatrix} c_{32} - c_{23} \\ c_{13} - c_{31} \\ c_{21} - c_{12} \end{pmatrix}$$

1.3.3 Unit Quaternion

A non-minimal representation which doesn't suffer from singularities
From rotation vector $\varphi \in \mathbb{R}^3$ to unit quaternion:

$$\chi_{R, \text{quat}} = \xi = \begin{pmatrix} \xi_0 \\ \xi \end{pmatrix} = \begin{pmatrix} \cos(\frac{\theta}{2}) \\ \sin(\frac{\theta}{2}) \vec{n} \end{pmatrix} = \begin{pmatrix} \xi_0 \\ \xi_1 \\ \xi_2 \\ \xi_3 \end{pmatrix}$$

Hamiltonian conventions:

$$\xi = \xi_0 + \xi_1 i + \xi_2 j + \xi_3 k \quad i^2 = j^2 = k^2 = ijk = -1;$$

$$ij = -ji = -ijk^2 = k \quad jk = -kj = i \quad ki = -ik = j$$

The first parameter ξ_0 is the real part of the quaternion, ξ the imaginary part. $\xi_0^2 + \xi_1^2 + \xi_2^2 + \xi_3^2 = 1$

Given a rotation matrix, the corresponding quaternion is:

$$\chi_{R, \text{quat}} = \xi_{AD} = \frac{1}{2} \begin{pmatrix} \sqrt{c_{11} + c_{22} + c_{33} + 1} \\ \frac{sgn(c_{32} - c_{23}) \sqrt{c_{11} - c_{22} - c_{33} + 1}}{2} \\ \frac{sgn(c_{13} - c_{31}) \sqrt{c_{22} - c_{33} - c_{11} + 1}}{2} \\ \frac{sgn(c_{21} - c_{12}) \sqrt{c_{33} - c_{11} - c_{22} + 1}}{2} \end{pmatrix}$$

with $sgn(x) = 1$ for $x \geq 0$ and $sgn(x) = -1$ for $x < 0$

The Corresponding Rotation Matrix can be calculated with:

$$\begin{aligned} C_{AD} &= I_{3 \times 3} + 2\xi_0 [\xi]_{\times} + 2[\xi]_{\times}^2 \\ &= (2\xi_0^2 - 1) I_{3 \times 3} + 2\xi_0 [\xi]_{\times} + 2\xi \xi^T \\ &= \begin{pmatrix} \xi_0^2 + \xi_1^2 - \xi_2^2 - \xi_3^2 & 2\xi_1 \xi_2 - 2\xi_0 \xi_3 & 2\xi_0 \xi_2 + 2\xi_1 \xi_3 \\ 2\xi_0 \xi_3 + 2\xi_1 \xi_2 & \xi_0^2 - \xi_1^2 + \xi_2^2 - \xi_3^2 & 2\xi_2 \xi_3 - 2\xi_0 \xi_1 \\ 2\xi_1 \xi_3 - 2\xi_0 \xi_2 & 2\xi_0 \xi_1 - 2\xi_2 \xi_3 & \xi_0^2 - \xi_1^2 - \xi_2^2 + \xi_3^2 \end{pmatrix} \end{aligned}$$

It holds that:

$$\xi = \begin{pmatrix} \xi_0 \\ \xi \end{pmatrix} \xrightarrow{\text{inverse}} \xi^{-1} = \xi^T = \begin{pmatrix} \xi_0 \\ -\xi \end{pmatrix}$$

Multiplication: $\xi_{AB} \otimes \xi_{BC} = M_l(\xi_{AB}) \xi_{BC}$

$$\underbrace{\begin{pmatrix} \xi_0 & -\xi_1 & -\xi_2 & -\xi_3 \\ \xi_1 & \xi_0 & -\xi_3 & \xi_2 \\ \xi_2 & \xi_3 & \xi_0 & -\xi_1 \\ \xi_3 & -\xi_2 & \xi_1 & \xi_0 \end{pmatrix}}_{M_l(\xi_{AB})} ; \underbrace{\begin{pmatrix} \xi_0 & -\xi_1 & -\xi_2 & -\xi_3 \\ \xi_1 & \xi_0 & \xi_3 & -\xi_2 \\ \xi_2 & -\xi_3 & \xi_0 & \xi_1 \\ \xi_3 & \xi_2 & -\xi_1 & \xi_0 \end{pmatrix}}_{M_r(\xi_{BC})}$$

Direct way of rotating a Vector: (with $p({}^{\mathcal{B}}r) = \begin{pmatrix} 0 \\ {}^{\mathcal{B}}r \end{pmatrix}$)

$$p({}^{\mathcal{A}}r) = \xi_{AB} \otimes p({}^{\mathcal{B}}r) \otimes \xi_{AB}^T = M_l(\xi_{AB}) M_r(\xi_{AB}^T) p({}^{\mathcal{B}}r)$$

1.4 Angular Velocity

Consider Frame \mathcal{B} which is moving with respect to fixed Frame \mathcal{A} . The angular velocity of the rotation of \mathcal{B} w.r.t. \mathcal{A} is ${}^{\mathcal{A}}\omega_{AB}$.

$$[{}^{\mathcal{A}}\omega_{AB}]_{\times} = \dot{C}_{AB} \cdot C_{AB}^T \quad \omega_{AB} = -\omega_{BA}$$

with $[{}^{\mathcal{A}}\omega_{AB}]_{\times} = \begin{pmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{pmatrix}$ and ${}^{\mathcal{A}}\omega_{AB} = \begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix}$

Angular Velocities can be transformed as normal vectors:

$${}^{\mathcal{B}}\omega_{AB} = C_{BA} \cdot {}^{\mathcal{A}}\omega_{AB}$$

the product matrix is transformed by:

$$[{}^{\mathcal{B}}\omega_{AB}]_{\times} = C_{BA} \cdot [{}^{\mathcal{A}}\omega_{AB}]_{\times} \cdot C_{AB}$$

Angular Velocities in the same Frame can be added:

$$\mathcal{I} \omega_{IE} = \mathcal{I} \omega_{I0} + \dots + \mathcal{I} \omega_{nE}$$

1.4.1 Time Derivative of Rotation Parameters

Similar to lin. vel. we can define: ${}^{\mathcal{A}}\omega_{AB} = E_R(\chi_R) \cdot \dot{\chi}_R$

quaternions: $E_R = 2H(\xi) = 2 \begin{bmatrix} -\xi & [\xi]_{\times} + \xi_0 I_{3 \times 3} \end{bmatrix}$
Angle Axis: $E_R = \begin{bmatrix} n & \sin \theta I_{3 \times 3} + (1 - \cos \theta) [\vec{n}]_{\times} \end{bmatrix}$

Rotation Vector: $E_R = \begin{bmatrix} I_{3 \times 3} + [\varphi]_{\times} \left(\frac{1 - \cos \|\varphi\|}{\|\varphi\|^2} \right) + [\varphi]_{\times}^2 \left(\frac{\|\varphi\| - \sin \|\varphi\|}{\|\varphi\|^3} \right) \end{bmatrix}$

1.5 Velocity in Moving Bodies

v_P	the absolute Velocity of point P (absolute) acceleration of P (absolute) angular velocity of body B (absolute) angular acceleration of body B
$a_P = \dot{v}_P$	
$\Omega_B = \omega_{AB}$	
$\Psi_B = \dot{\Omega}_B$	

We can write the Position P as:

$${}^{\mathcal{A}}r_{AP} = {}^{\mathcal{A}}r_{AB} + C_{AB} \cdot {}^{\mathcal{B}}r_{BP}$$

Which can be differentiated to (using $\dot{C}_{AB} = [{}^{\mathcal{A}}\omega_{AB}]_{\times} \cdot C_{AB}$):

$${}^{\mathcal{A}}\dot{r}_{AP} = {}^{\mathcal{A}}\dot{r}_{AB} + {}^{\mathcal{A}}\omega_{AB} \times {}^{\mathcal{A}}r_{BP}$$

This is the famous **Rigid Body Formulation** for velocities:

$$v_P = v_B + \Omega \times r_{BP}$$

For Accelerations: $a_P = a_B + \Psi \times r_{BP} + \Omega \times (\Omega \times r_{BP})$

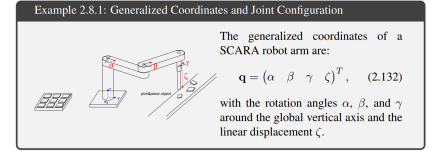
In case a moving system \mathcal{B} is used for representation, the **Euler Differentiation rule** must be applied (with a non moving system \mathcal{A}):

$$\begin{aligned} {}^{\mathcal{B}}v_P &= C_{BA} \cdot \frac{d}{dt} (C_{AB} \cdot {}^{\mathcal{B}}r_{AP}) \\ &= {}^{\mathcal{B}}\dot{r}_{AP} + {}^{\mathcal{B}}\omega_{AB} \times {}^{\mathcal{B}}r_{AP} \end{aligned}$$

1.6 Kinematics of Systems of Bodies

1.6.1 Generalized Coordinates and Joint Configurations

The configuration of a robot such as a manipulator can be described by the **generalized coordinate vector**: $q = (q_1 \dots q_n)^T$. The choice of q isn't unique, but it has to completely describe the configuration of the system (q const. \Rightarrow robot can't move).



1.6.2 Task-Space Coordinates

End-Effector Configuration Parameters

The position $r_e \in \mathbb{R}^3$ and rotation $\phi_e \in SO(3)$ of a frame w.r.t. a base can be parametrized by:

$$\chi_e = \begin{pmatrix} \chi_{eP} \\ \chi_{eR} \end{pmatrix} = \begin{pmatrix} \chi_1 \\ \vdots \\ \chi_m \end{pmatrix} \in \mathbb{R}^m$$

Operational Space Coordinates The end-effector operates in the operational space, which depends on the geometry and structure of the arm. It can be described with:

$$\chi_o = \begin{pmatrix} \chi_{oP} \\ \chi_{oR} \end{pmatrix} = \begin{pmatrix} \chi_1 \\ \vdots \\ \chi_{m_o} \end{pmatrix}$$

where $\chi_1 \dots \chi_{m_o}$ are *independent* operational space coordinates. It can be seen as a **minimal selection** of the above end-effector parameters.

1.6.3 Forward Kinematics

Forward kinematics map from joint coordinates q to the end-effector configuration χ_e : $\chi_e = \chi_e(q)$
This relation can be obtained through the transformations of each link:

$$T_{IE}(q) = T_{I0} \left(\prod_{k=1}^{n_j} T_{k-1,k} \right) T_{n_j E} = \begin{bmatrix} C_{IE}(q) & I^{rIE}(q) \\ 0_{1 \times 3} & 1 \end{bmatrix}$$

1.6.4 Differential Kinematics and Analytical Jacobian

To linearise the forward kinematics we use a first order approximation (m depends on parameterization):

$$\Delta \chi_e \approx J_{eA}(q) \Delta q, \quad J_{eA}(q) = \begin{bmatrix} \frac{\partial \chi_1}{\partial q_1} & \dots & \frac{\partial \chi_1}{\partial q_{n_j}} \\ \vdots & \ddots & \vdots \\ \frac{\partial \chi_m}{\partial q_1} & \dots & \frac{\partial \chi_m}{\partial q_{n_j}} \end{bmatrix}$$

It results in an exact relation between velocities:

$$\dot{\chi}_e = J_{eA}(q) \dot{q}$$

Literature often talk about **position** and **rotation** Jacobians:

$$J_{eA} = \begin{bmatrix} J_{eA_P} \\ J_{eA_R} \end{bmatrix} = \begin{bmatrix} \frac{\partial \chi_{eP}}{\partial q} \\ \frac{\partial \chi_{eR}}{\partial q} \end{bmatrix} \in \mathbb{R}^{m_e \times n_j}$$

1.6.5 Geometric or Basic Jacobian

The geometric or basic Jacobian relates the generalized velocity \dot{q} to the end-effector velocity (linear v_e and angular ω_e):

$$w_e = \begin{pmatrix} v_e \\ \omega_e \end{pmatrix} = J_{e0}(q) \cdot \dot{q} \quad J_{e0} \in \mathbb{R}^{6 \times n_j} (3D)$$

Note: In the most general cases J_{e0} has dimension $6 \times n_j$ and has Frame \mathcal{A} as a basis (like the velocity). From the velocities $w_C = w_B + w_{BC}$ we can derive, that geometric Jacobians can simply be added (in the same reference):

$${}^{\mathcal{A}}J_C = {}^{\mathcal{A}}J_B + {}^{\mathcal{A}}J_{BC}$$

Geometric Jacobian:

$${}^{\mathcal{I}}J_{e0} = \begin{bmatrix} {}^{\mathcal{I}}J_{e0P} \\ {}^{\mathcal{I}}J_{e0R} \end{bmatrix} = \begin{bmatrix} {}^{\mathcal{I}}n_1 \times \mathcal{I}r_1(n+1) & \dots & {}^{\mathcal{I}}n_n \times \mathcal{I}r_n(n+1) \\ {}^{\mathcal{I}}n_1 & \dots & {}^{\mathcal{I}}n_n \end{bmatrix}$$

where n_k represents the rotation axis of joint k such that:

$$\omega_{(k-1)k} = n_k \dot{q}_k \quad {}^{\mathcal{I}}n_k = C_{{}^{\mathcal{I}}(k-1)} * k-1 n_k$$

and $r_1(n+1) \dots r_n(n+1)$ represent the position vector from the joint $1 \dots n$ to the end-effector.

Don't forget to transform to inertial frame!

For **prismatic Joints** the Position part ($n \times r$) is an unit-vector ξ_k in joint direction. The Rotational part (n_i) is obviously zero.

Mapping from Analytic to Geometric Jacobian, it holds that:

$$J_{e0}(q) = E_e(\chi) J_{eA}(q)$$

$$\text{with } E_e(\chi) = \begin{bmatrix} E_p(\chi) & 0 \\ 0 & E_R(\chi) \end{bmatrix} \in \mathbb{R}^{6 \times m_e}$$

1.7 Kinematic Control Methods

1.7.1 Inverse Differential Kinematics

The Jacobian $J_{e0}(q)$ performs a simple mapping from joint space to end-effector velocity.

$$w_e = J_{e0} \dot{q}$$

To solve the inverse problem, we use take the **pseudo-inverse** J_{e0}^+ of the Jacobian.

$$\dot{q} = J_{e0}^+ \cdot w_e^*$$

By taking the Moore-Penrose pseudo inverse, the solution $\dot{q} = J_{e0}^+ \cdot w_e^*$ minimises the least square error $\|w_e^* - J_{e0} \dot{q}\|^2$

Moore-Penrose Inverse

$$A^+ = A^T (A A^T)^{-1} \text{ right inverse (full row rank) } \square$$

$$A^+ = (A^T A)^{-1} A^T \text{ left inverse (full col rank) } \square$$

Note: For close to singular configurations, J_{e0} becomes badly conditioned, what causes large joint velocities for just a small end-effector velocity. This can be handled by using a **damped solution**. $\dot{q} = J_{e0}^T (J_{e0} J_{e0}^T + \lambda^2 \mathbb{1})^{-1} w_e^*$

Redundancy

For a Robot that has more joints than DOF ($rank(J_{e0}) < n$), the configuration is called **redundant**. Like previous, we can take the pseudo inverse:

$$\dot{q} = J_{e0}^T (J_{e0} J_{e0}^T)^{-1} \cdot w_e^* = J_{e0}^+ \cdot w_e^*$$

redundancy implies, that there are infinite additional solutions:

$$\dot{q} = J_{e0}^+ \cdot w_e^* + N \dot{q}_0$$

with $N = \mathcal{N}(J_{e0})$ as null-space projection matrix, fulfilling $J_{e0} N = 0$. Thus, we can choose arbitrary \dot{q}_0 without changing the velocity w_e^* .

The simplest method for the **null-space projection** is:

$$N = \mathbb{1} - J_{e0}^+ J_{e0}$$

1.7.2 Multi-task Inverse Differential Kinematic Control

For multiple tasks (same priority) $task_i := \{J_i, w_i^*\}$ we can calculate the velocity (least square if row rank > col rank):

$$\dot{q} = \underbrace{\begin{bmatrix} J_1 \\ \vdots \\ J_{n_t} \end{bmatrix}}_{\tilde{J}} \cdot \underbrace{\begin{bmatrix} w_1^* \\ \vdots \\ w_{n_t}^* \end{bmatrix}}_{\tilde{w}}$$

For weighted tasks we could use a weighted pseudo inverse: $\tilde{J}^+ \tilde{W} = (\tilde{J}^T \tilde{W} \tilde{J})^{-1} \tilde{J}^T \tilde{W}$ with weight $W = diag(w_1, \dots, w_m)$

Multitask Prioritisation

An approach for prioritisation tasks (descending priority) is to use consecutive null-space projections.

Using the solution for task 1 $\dot{q} = J_1^+ w_1^* + N_1 \dot{q}_0$, we can derive a term for q_0 :

$$w_2 = J_2 \dot{q} = J_2 (J_1^+ w_1^* + N_1 \dot{q}_0)$$

$$\iff \dot{q}_0 = (J_2 N_1)^+ (w_2^* - J_2 J_1^+ w_1^*)$$

Substitution in the first solution for task 1 gives:

$$\dot{q} = J_1^+ w_1^* + N_1 (J_2 N_1)^+ (w_2^* - J_2 J_1^+ w_1^*)$$

For n_t tasks this can be written recursively:

$$\dot{q} = \sum_{i=1}^{n_t} \tilde{N}_i \dot{q}_i \text{ with } \dot{q}_i = (J_i \tilde{N}_i)^+ \left(w_i^* - J_i \sum_{k=1}^{i-1} \tilde{N}_k \dot{q}_k \right)$$

with \tilde{N}_i the null space projection of the stacked J , $\tilde{J}_i = [J_1^T \dots J_{i-1}^T]^T$

1.7.3 Inverse Kinematics (Numerical Solution)

The goal of inverse Kinematics is to find the joint configuration for a given end-effector configuration χ_e^* : $q = q(\chi_e^*)$

We can solve this problem iteratively by using: $\Delta \chi_e = J_{eA} \Delta q$

Algorithmus 1 : Numerical Inverse Kinematics

```

q ← q0; // Start Configuration
while ||χe* − χe(q)|| > tol do
    JeA ← JeA(q) = ∂χe/∂q(q); // Evaluate (local) Jacobian
    JeA+ ← (JeA)+; // Calculate Pseudo Inverse
    Δχe ← χe* − χe(q); // Find Error Vector
    q ← q + JeA+ Δχe; // Update generalized Coordinates
end
    
```

To $\Delta \chi_e^i$ the linearization (Jacobian) becomes inaccurate, scale: $q \leftarrow q + k J_{eA}^+ \Delta \chi_e$, $0 < k < 1$

For badly conditioned (singular) Jacobians, we use either the damped inverse or use the Jacobi-transposed method:

$$q \leftarrow q + \alpha J_{eA}^T \Delta \chi_e$$

For small enough α convergence can be guaranteed.

Shortest Path rotation

For a straight rotation along the "shortest path", we rotate along the rotation vector $\Delta \varphi$ (\rightarrow rot parameterization affects convergence, axis angle = shortest rot path).

The rotation Matrix is given by:

$$C_{AB}(\Delta \varphi) = C_{\mathcal{I}A}(\varphi^t)^T C_{\mathcal{I}B}(\varphi^*)$$

(Note that $\Delta \varphi \neq \varphi^* - \varphi^t$)

The rotation vector is the same in both frames A & B.

$${}^{\mathcal{A}}\Delta \varphi = {}^{\mathcal{B}}\Delta \varphi = \text{rotVec}(C_{AB}) \rightarrow \Delta \varphi = \theta * n$$

Instead of mapping this vector into \mathcal{I} , we can derive it directly:

$${}^{\mathcal{I}}\Delta \varphi = C_{\mathcal{I}A} \text{rotVec}(C_{\mathcal{I}B}) = \text{rotVec}(C_{\mathcal{I}B} C_{\mathcal{I}A}^T)$$

Now we can change the update step δ of the algorithm to:

$$q \leftarrow q + k_{PR} {}^{\mathcal{I}}J_{e0}^+ {}^{\mathcal{I}}\Delta \varphi$$

1.7.4 Trajectory Control

Pure inverse differential kinematics often drift away from the predefined path. Hence, we introduce a feedback.

For predefined position $r_e^*(t)$ and velocity $\dot{r}_e^*(t)$:

$$\dot{q}^* = J_{e0P}^+(q^t) \cdot (\dot{r}_e^*(t) + k_{PP} \Delta r_e^t) \quad \text{with } \Delta r_e^t = r_e^*(t) - r_e(q^t)$$

Similar for orientation $\chi_R^*(t)$ and angular velocity $\omega^*(t)$:

$$\dot{q}^* = J_{e0R}^+(q^t) \cdot (\omega_e^*(t) + k_{PR} \Delta \varphi) \quad \text{with } \Delta \varphi \text{ from above.}$$

2 Dynamics

We formulate multi-body dynamics as:

$$M(q) \ddot{q} + b(q, \dot{q}) + g(q) = \tau + J_c(q)^T F_c$$

consisting of the following elements:

$M(q)$	Generalized mass (or inertia) matrix (orthogonal)
$\dot{q}, \ddot{q}, \ddot{\ddot{q}}$	Generalized position, velocity and acceleration vectors
$b(q, \dot{q})$	Coriolis and centrifugal terms
$g(q)$	Gravitational terms
τ	External generalized forces
F_c	External cartesian forces (e.g. from contacts)
$J_c(q)$	Geometric Jacobian corresponding to external forces

2.1 Principle of virtual Work

$$\delta W \int_{\mathcal{B}} \delta r^T \cdot (\ddot{r} dm - dF_{ext}) = 0, \quad \forall \delta r$$

dm	infinitesimal mass element
dF_{ext}	external Forces acting on element dm
\ddot{r}	acceleration of element dm
δr	virtual displacement of dm
\mathcal{B}	Body System containing infinitesimal particles dm

2.2 Newton-Euler Method

$$m \cdot \ddot{x} = \sum F_i \quad \text{and} \quad \Theta \cdot \ddot{\varphi} = \sum T_i$$

For Multi-Body System we need to cut every joint free and introduce constraining forces for every piece. This results in a system of equations with additional kinematic constraints.

2.3 Lagrange Method

The **Lagrangian Function** for mech. sys. is exactly the difference between the total kinetic energy \mathcal{T} and total potential energy \mathcal{U} .

$$\mathcal{L} = \mathcal{T} - \mathcal{U}$$

Euler-Lagrange (of the second kind): $\frac{d}{dt} \left(\frac{\partial \mathcal{L}}{\partial \dot{q}} \right) - \left(\frac{\partial \mathcal{L}}{\partial q} \right) = \tau \Rightarrow$

$$\frac{d}{dt} \left(\frac{\partial \mathcal{T}}{\partial \dot{q}} \right) - \frac{\partial \mathcal{T}}{\partial q} + \frac{\partial \mathcal{U}}{\partial q} = \tau \text{ with total gen. ext. forces } \tau$$

The **Hamiltonian** states the total energy: $\mathcal{H} = \mathcal{T} + \mathcal{U}$

2.3.1 Kinetic Energy

The kinetic energy is defined as (recall the basic formulas $E = \frac{1}{2} m v^2 \& \frac{1}{2} J \omega^2$):

$$\mathcal{T} = \sum_{i=1}^{n_b} \left(\frac{1}{2} m_i {}^{\mathcal{I}}A_i^T S_i A_i \dot{r}_i + \frac{1}{2} {}^{\mathcal{B}}\Omega_{S_i}^T \cdot {}^{\mathcal{B}}\Theta_{S_i} \cdot {}^{\mathcal{B}}\Omega_{S_i} \right)$$

With the Jacobian relations $\dot{r}_{S_i} = J_{S_i} \dot{q}$, $\Omega_{S_i} = J_{R_i} \dot{q}$ we can rewrite this:

$$\mathcal{T}(q, \dot{q}) = \frac{1}{2} \dot{q}^T \underbrace{\left(\sum_{i=1}^{n_b} (J_{S_i}^T m_i J_{S_i} + J_{R_i}^T \Theta_{S_i} J_{R_i}) \right)}_{M(q)} \dot{q} \Rightarrow \frac{\partial \mathcal{T}}{\partial \dot{q}} = M \dot{q}$$

2.3.2 Potential Energy

r_{S_i} to CoM of each body (Zero energy level arbitrary):

$$F_{g_i} = m_i \cdot g \cdot \mathcal{I}e_g \Rightarrow \mathcal{U}_g = - \sum_{i=1}^{n_b} r_{S_i}^T F_{g_i}$$

Potential energy for elastic elements: $\mathcal{U}_{E_j} = \frac{1}{2} k_j \underbrace{(d(q) - d_0)}_{\text{deflection}}$

and $F_j = k_j (d(q) - d_0) \frac{d(q) - d_0}{\|d(q) - d_0\|}$, with the current length of the spring $d(q)$ & resting length d_0 .

2.4 Projected Euler Method

$$\begin{bmatrix} v_{\Omega}^s \\ \dot{\Omega} \end{bmatrix} = \begin{bmatrix} J_P \\ J_R \end{bmatrix} \dot{q} \quad \begin{bmatrix} a_{\Psi}^s \\ \ddot{\Psi} \end{bmatrix} = \begin{bmatrix} J_P \\ J_R \end{bmatrix} \ddot{q} + \begin{bmatrix} \dot{J}_P \\ \dot{J}_R \end{bmatrix} \dot{q}$$

Rotate Inertia Tensor: $\mathcal{I}\Theta = C_{\mathcal{I}B} \cdot {}^{\mathcal{B}}\Theta \cdot C_{\mathcal{I}B}^T$

$$M = \sum_{i=1}^{n_b} \left({}^{\mathcal{A}}J_{S_i}^T \cdot m \cdot {}^{\mathcal{A}}J_{S_i} + {}^{\mathcal{B}}J_{R_i}^T \cdot {}^{\mathcal{B}}\Theta_{S_i} \cdot {}^{\mathcal{B}}J_{R_i} \right) (\mathcal{R}^{n_q \times n_q})$$

$$b = \sum_{i=1}^{n_b} \left({}^{\mathcal{A}}J_{S_i}^T m {}^{\mathcal{A}}\dot{J}_{S_i} \dot{q} + {}^{\mathcal{B}}J_{R_i}^T ({}^{\mathcal{B}}\Theta_{S_i} \cdot {}^{\mathcal{B}}\dot{J}_{R_i} \cdot \dot{q} + \underbrace{{}^{\mathcal{B}}\Omega_{S_i} \times {}^{\mathcal{B}}\Theta_{S_i} \cdot {}^{\mathcal{B}}\Omega_{S_i}}_{J_{R_i} \dot{q} \times {}^{\mathcal{B}}\Theta_{S_i} J_{R_i} \dot{q} = 0 \text{ in 2D}}) \right)$$

$$g = \sum_{i=1}^{n_b} \left(-{}^{\mathcal{A}}J_{S_i}^T \cdot {}^{\mathcal{A}}F_{g,i} \right)$$

2.4.1 External Forces & Actuation

known Forces F_j on system \rightarrow generalized forces $\tau_{F,ext}$ (due external force)

$$\tau_{F,ext} = \sum J_{P,j}^T F_j$$

with the translational (geometric) Jacobian of Point j (i.e. J_e for end effector)

Similar for external Torques T_j : $\tau_{T,ext} = \sum J_{R,j}^T T_j$ Motor

Torque: $\tau_{a,k} = (J_{S_k} - J_{S_{k-1}})^T F_{a_k} + (J_{R_k} - J_{R_{k-1}})^T T_{a_k}$

2.5 Joint-Space Dynamic Control

Joint Impedance Regulation

In case of torque controlled actuators, we can get a simple PD control law for the desired(*) actuator torque:

$$\tau^* = k_p(q^* - q) + k_d(\dot{q}^* - \dot{q})$$

This ends in a steady state offset of: $g(q) = k_p(q^* - q) + k_d(\dot{q}^* - \dot{q})$

Gravity Compensation: To compensate for the gravity offset, we simply add an estimated value $\hat{g}(q)$ to the control law:

$$\tau^* = k_p(q^* - q) + k_d(\dot{q}^* - \dot{q}) + \hat{g}(q)$$

Note: k_d and k_p are constant for all configurations (q), which reduces the overall performance.

Inverse Dynamics Control: A simple way to get dynamic decoupling and motion control is to get estimates \hat{M} , \hat{b} and \hat{g} and select the torque with:

$$\tau^* = \hat{M}(q) \ddot{q}^* + \hat{b}(q, \dot{q}) + \hat{g}(q)$$

Then, a common approach selects the desired acceleration according to:

$$\ddot{q}^* = k_p(q^* - q) + k_d(\dot{q}^* - \dot{q})$$

which has eigenfrequency $\omega = \sqrt{k_p}$ and Damping $D = \frac{k_d}{2\sqrt{k_p}}$ ($D = 1$ critical-, $D < 1$ under-, $D > 1$ over-damped)

2.6 Task-Space Dynamic Control

To move to a specific point in Task-Space (Fixed Frame) we need the linear and rotational acceleration of the end-effector:

$$\dot{w}_e = \begin{pmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{pmatrix} = J_e \ddot{q} + \dot{J}_e \dot{q} \Rightarrow \ddot{q}^* = J_e^+ (w^* - \dot{J}_e \dot{q}) \rightarrow \tau = \dots$$

2.6.1 Multi-task

Similar to kinematics, we can fulfill multiple tasks:

$$\ddot{q} = \begin{bmatrix} J_1 \\ \vdots \\ J_{n_t} \end{bmatrix}^+ \left(\begin{bmatrix} \dot{w}_1 \\ \vdots \\ \dot{w}_{n_t} \end{bmatrix} - \begin{bmatrix} \dot{J}_1 \\ \vdots \\ \dot{J}_{n_t} \end{bmatrix} \dot{q} \right)$$

and the recursive algorithm (hierarchical):

$$\dot{q} = \sum_{i=1}^{n_t} \bar{N}_i \dot{q}_i \text{ with } \ddot{q}_i = (J_i \bar{N}_i)^+ \left(w_i^* - \dot{J}_i \dot{q} - J_i \sum_{k=1}^{i-1} \bar{N}_k \ddot{q}_k \right)$$

2.6.2 End-Effector Dynamics

With $\tau = J_e^T F_e$ we can formulate the end-effector Dynamics:

$$\Lambda_e \dot{w}_e + \mu + p = F_e$$

$$\text{with } \begin{cases} \Lambda_e = (J_e M^{-1} J_e^T)^{-1}, & \mu = \Lambda_e J_e M^{-1} b - \Lambda_e \dot{J}_e \dot{q}, \\ p = \Lambda_e J_e M^{-1} g \end{cases}$$

as the end-effector inertia, centrifugal and gravitational terms in task-space.

End-Effector Motion Control

From the above Dynamics, we can get an inversion motion control, like in the joint space:

$$\dot{w}_e = k_p \underbrace{E (\chi_e^* - \chi_e)}_{\text{for rotations}=\text{rotVec}(C_i^* C_i^T)} + k_d (w_e^* - w_e) + \dot{w}_e(t)$$

$$\tau^* = \hat{J}^T (\hat{\Lambda}_e \dot{w}_e^* + \hat{\mu} + \hat{p})$$

together with a control law:

$$\dot{w}_e^* = k_p \begin{pmatrix} r_e^* - r_e \\ \Delta \phi_e \end{pmatrix} + k_d (w_e^* - w_e)$$

For small errors we can approximate:

$$\begin{pmatrix} r_e^* - r_e \\ \Delta \phi_e \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & E_R \end{pmatrix} \begin{pmatrix} r^* - r \\ \chi_R^* - \chi_R \end{pmatrix}$$

2.6.3 Operational Space Control

Note: We need to extend the end effector dynamics with a contact Force F_c :

$$F_c + \Lambda_e \dot{w}_e + \mu + p = F_e$$

In some situations the robot has to either apply a force or move in a direction. This can be described by two specification matrices for position and orientation:

$$\Sigma_p = \begin{pmatrix} \sigma_{px} & 0 & 0 \\ 0 & \sigma_{py} & 0 \\ 0 & 0 & \sigma_{pz} \end{pmatrix} \quad \Sigma_r = \begin{pmatrix} \sigma_{rx} & 0 & 0 \\ 0 & \sigma_{ry} & 0 \\ 0 & 0 & \sigma_{rz} \end{pmatrix}$$

with σ_i either 1(move) or 0(don't).

$$\tau^* = \hat{J}^T (\hat{\Lambda}_e S_M \dot{w}_e^* + S_F F_c + \hat{\mu} + \hat{p})$$

$$S_M = \begin{pmatrix} C^T \Sigma_p C & 0 \\ 0 & C^T \Sigma_r C \end{pmatrix} \quad S_F = \begin{pmatrix} C^T (1 - \Sigma_p) C & 0 \\ 0 & C^T (1 - \Sigma_r) C \end{pmatrix}$$

2.7 Least Square Optimisation

I Slides 6.5

So far we only considered the optimisation of $\min \|\ddot{q}\|_2$ as result of the Pseudoinverse J^+ . To optimize another objective, we can formulate the problem in multiple tasks:

$$\left| \begin{array}{l} \tau = M \ddot{q} + b + g \\ \dot{w} = J \ddot{q} + \dot{J} \dot{q} \end{array} \right| \Rightarrow \left| \begin{array}{l} [M \quad -1] \begin{pmatrix} \ddot{q} \\ \tau \end{pmatrix} + b + g = 0 \\ [J_e \quad 0] \begin{pmatrix} \ddot{q} \\ \tau \end{pmatrix} + \dot{J}_e \dot{q} = \dot{w}^* \end{array} \right|$$

This has always to be fulfilled, and can be extended by additional objectives.

It can be solved as single (stacked) tasks:

$$\min_{\ddot{q}, \tau} \left\| \begin{bmatrix} M & -1 \\ J_e & 0 \end{bmatrix} \begin{pmatrix} \ddot{q} \\ \tau \end{pmatrix} - \begin{pmatrix} -b - g \\ \dot{w}_e^* - \dot{J}_e \dot{q} \end{pmatrix} \right\|_2$$

Or with different priorities:

$$\min_{\ddot{q}, \tau} \left\| [J_e \quad 0] \begin{pmatrix} \ddot{q} \\ \tau \end{pmatrix} - (\dot{w}_e^* - \dot{J}_e \dot{q}) \right\|_2$$

such that: $[M \quad -1] \begin{pmatrix} \ddot{q} \\ \tau \end{pmatrix} - (-b - g) = 0$

This will exploit the nullspace of the higher priority task to minimize the solution. \Rightarrow Solve with numeric solver

3 Floating Base Systems

3.1 FB Kinematics

Free floating robots are described by n_b **unactuated** base coordinates q_b and n_j actuated joint coordinates q_j .

$$q = \begin{pmatrix} q_b \\ q_j \end{pmatrix} \quad \text{with } q_b = \begin{pmatrix} q_{bP} \\ q_{bR} \end{pmatrix} \in \mathbb{R}^3 \times SO(3)$$

The minimal number of generalized coordinates for the base is $n_{b0} = 6$ (3D).

Generalized Velocity (often simply written as \dot{q})

$$u = \begin{pmatrix} \mathcal{I} v_B \\ \mathcal{B} \omega_{IB} \\ \dot{\varphi}_1 \\ \vdots \\ \dot{\varphi}_{n_j} \end{pmatrix} \quad \text{with mapping} \quad E_{fb} = \begin{bmatrix} 1_{3 \times 3} & 0 & 0 \\ 0 & E_{\chi_R} & 0 \\ 0 & 0 & 1_{n_j \times n_j} \end{bmatrix}$$

$$u = E_{fb} \cdot \dot{q}, \quad \text{where } \dot{q} = \begin{pmatrix} \dot{q}_b \\ \dot{q}_j \end{pmatrix}$$

3.1.1 Forward Kinematics

The position vector of point Q can be expressed via the Base B

$$\tau^r I_Q(q) = \tau^r I_B(q) + C_{IB}(q) \cdot \mathcal{B}^r B_Q(q)$$

3.1.2 Differential Kinematics

The spacial Jacobian maps u to v and ω :

$$\begin{pmatrix} \mathcal{I} v_Q \\ \mathcal{I} \omega_{IQ} \end{pmatrix} = \mathcal{I} J_Q(q) \cdot u$$

$$\mathcal{I} J_Q(q) = \begin{bmatrix} 1_{3 \times 3} & -C_{IB} \cdot [\mathcal{B}^r B_Q] \times & C_{IB} \cdot \mathcal{B} J_{P_{q_j}}(q_j) \\ 0_{3 \times 3} & C_{IB} & C_{IB} \cdot \mathcal{B} J_{R_{q_j}}(q_j) \end{bmatrix}$$

3.1.3 Contacts & Constraints

Every Point C_i in contact with the environment imposes **constant** position and **zero** velocity and acceleration. \rightarrow Contact Jac. J_{C_i}

$$\mathcal{I} J_{C_i} u = 0, \quad \mathcal{I} J_{C_i} \dot{u} + \mathcal{I} \dot{J}_{C_i} u = 0$$

where multiple J_{C_i} can be stacked for multiple contact points.

The $\text{rank}(J_c)$ indicates the number of independent contact constraints. The stacked J_c can be split in a body and joint part:

$$J_c = [J_{c,b} \quad J_{c,j}] = \left[\frac{\partial r_c}{\partial q_b} \quad \frac{\partial r_c}{\partial q_j} \right] \in \mathbb{R}^{n_c \times (n_b + n_j)}$$

If the rank of $J_{c,b}$ has full rank (≤ 6 in 3D), the joints can move the body in every direction. The difference $\text{rank}(J_c) - \text{rank}(J_{c,b})$ is the number of **internal kinematic constraints** (i.e. legs can move in respect to each other).

3.1.4 Inverse Kinematics

We apply inverse kinematics, where the ground contact $J_c u = 0$ has the highest priority:

$$J_c u = 0 \Rightarrow u = J_c^+ 0 + \mathcal{N}(J_c) u_0 = \mathcal{N}(J_c) u_0$$

Given a demanded motion w_t we can calculate the required velocity: $w_t = J_t u \Rightarrow u = \mathcal{N}(J_t N_c)^+ w_t$

3.2 FB Dynamics

We need to extend the known dynamics with a selection for the torques τ , since the body is unactuated:

$$M(q) \dot{u} + b(q, u) + g(q) = \mathbf{S}^T \tau + J_{ext}(q)^T F_{ext}$$

with $u_j = S u = S \begin{pmatrix} u_b \\ u_j \end{pmatrix} = [0_{n_j \times 6} \quad 1_{n_j \times n_j}] \begin{pmatrix} u_b \\ u_j \end{pmatrix}$

Note: If we have the forces that the robot exerts on its environment, we need them to switch the side in the equation:

$$M(q) \dot{u} + b(q, u) + g(q) + J_c(q)^T F_c = S^T \tau$$

Together with the contact constraints we can calculate the contact Force F_c :

$$F_c = (J_c M^{-1} J_c^T)^{-1} (J_c M^{-1} (S^T \tau - b - g) + \dot{J}_c u)$$

3.2.1 Constraint Dynamics

We can define a Null-space matrix for the contact constraints:

$$N_c = 1 - M^{-1} J_c^T (J_c M^{-1} J_c^T)^{-1} J_c$$

This gives the following equations of motion, which are reduced, but **consistent with the constraints** (contact forces):

$$N_c^T (M \dot{u} + b + g) = N_c^T S^T \tau \Leftrightarrow J_c N_c = 0$$

3.2.2 FB Inverse Dynamics

With a desired $\dot{u}_{consistent}^*$ We can invert the equation of motion of above:

$$\tau^* = (N_c^T S^T)^+ N_c^T (M \dot{u}^* + b + g) + \mathcal{N}(N_c^T S^T) \tau_0^*$$

When taking only the first part without a Nullspace ($\tau_0^* = 0$), the solution is the least square minimal torque τ^* that fulfills the EoM.

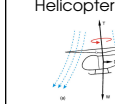
3.2.3 FB Quadratic Programming

Objective: $\min_x \|A_i x - b_i\|_2 \quad x = \begin{pmatrix} \dot{F}_c \\ \tau \end{pmatrix}$ for different tasks:

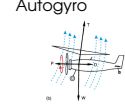
Motion Task: $J \dot{u} + \dot{J} u = \dot{w}^* \Rightarrow A = [J_i, 0, 0] \quad b = w^* - \dot{J} u$
 Force Task: $F_i = F_i^* \Rightarrow A = [0, I, 0] \quad b = F_i^*$
 Torque Task: $\min \|\tau\| \Rightarrow A = [0, 0, I] \quad b = 0$

4 Rotorcrafts

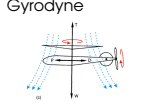
Overview



Has a power driven main rotor, which can be tilted



Passive main rotor and a forward facing active propeller. Can't hover.

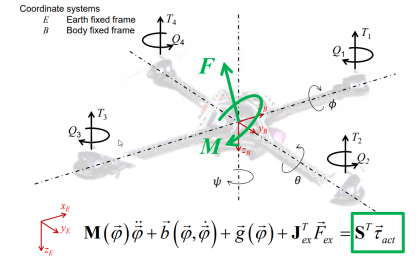


Active main rotor, but can't be tilted. Additional front facing active propeller.

Typical rotorcrafts are: Single Rotors, Multi rotors, Coaxial, Ducted Fan, Omnidirectional Multicopter (movable rotors).

4.1 Modelling of Quadrotor

Modelling and simulations are important, but they must be validated in reality.



Structural Properties:

Arm length l , Rotor height h , Mass m , Inertia $I = \begin{pmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{pmatrix}$

Hub force & rolling moments depend on flight regime and can be neglected in hovering.

Rotation:

$$\begin{pmatrix} \phi \\ \theta \\ \psi \end{pmatrix} = \begin{pmatrix} Roll \\ Pitch \\ Yaw \end{pmatrix} \rightarrow \begin{pmatrix} x \text{ Axis} \\ y \text{ Axis} \\ z \text{ Axis} \end{pmatrix} \in \begin{pmatrix} [-\pi, \pi] \\ [\pi/2, \pi/2] \\ [\pi, \pi] \end{pmatrix}$$

Equation for the rotational speed (singularity at $\theta = \pi/2$): $\mathcal{B} \omega = \mathcal{B} \omega_{roll} + \mathcal{B} \omega_{pitch} + \mathcal{B} \omega_{yaw}$

$$\mathcal{B} \omega = E_r \dot{\chi}_r = E_r \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}, \text{ with } E_r = \begin{bmatrix} 1 & 0 & -\sin \theta \\ 0 & \cos \phi & \sin \phi \cos \theta \\ 0 & -\sin \phi & \cos \phi \cos \theta \end{bmatrix}$$

Linearization small Roll & Pitch ($\phi, \theta \approx 0$) \rightarrow **unity matrix** $E_r = 1$

4.1.1 Body Dynamics

$$\begin{bmatrix} m 1_{3 \times 3} & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} \mathcal{B} \dot{v} \\ \mathcal{B} \dot{\omega} \end{bmatrix} + \begin{bmatrix} \mathcal{B} v \times m \mathcal{B} v \\ \mathcal{B} \omega \times I \mathcal{B} \omega \end{bmatrix} = \begin{bmatrix} \mathcal{B} F \\ \mathcal{B} M \end{bmatrix}$$

with Forces

$$\mathcal{B} F = \mathcal{B} F_G + \mathcal{B} F_{Aero} = C_{EB}^T \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} + \sum_{i=1}^4 \begin{pmatrix} 0 \\ 0 \\ -T_i \end{pmatrix}$$

and Hover Moments

$$\mathcal{B} M_{Aero} = \mathcal{B} \begin{pmatrix} l(T_4 - T_2) \\ l(T_1 - T_3) \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ \sum_{i=1}^4 Q_i (-1)^{(i-1)} \end{pmatrix}$$

with Thrust forces $T_i = b_i \omega_{p,i}^2$ and Drag Forces $Q_i = d_i \omega_{p,i}^2$

Rotational Dynamics (2nd row of Dynamics)

$$\begin{aligned} I_{xx} \dot{\omega}_x &= \omega_y \cdot \omega_z (I_{yy} - I_{zz}) + l \cdot b (\omega_{p,4}^2 - \omega_{p,2}^2) \\ I_{yy} \dot{\omega}_y &= \omega_z \cdot \omega_x (I_{zz} - I_{xx}) + l \cdot b (\omega_{p,1}^2 - \omega_{p,3}^2) \\ I_{zz} \dot{\omega}_z &= d (\omega_{p,1}^2 - \omega_{p,2}^2 + \omega_{p,3}^2 - \omega_{p,4}^2) \end{aligned}$$

with ω_{xyz} as entries of the body rotation \mathcal{B}

\Rightarrow We have full control over all rotational speeds (every equation depends on rotor speeds $\omega_{p,i}$). Note that $(I_{xx} - I_{yy}) = 0$ (x-y symmetry), which dropped out of the 3rd equation.

Translational Dynamics (first row of dynamics)

$$\begin{aligned} m \dot{v}_x &= m(\omega_z \cdot v_y - \omega_y \cdot v_z) - \sin \theta mg \\ m \dot{v}_y &= m(\omega_x \cdot v_z - \omega_z \cdot v_x) + \sin \phi \cos \theta mg \\ m \dot{v}_z &= m(\omega_y \cdot v_x - \omega_x \cdot v_y) + \cos \phi \cos \theta mg \\ &\quad - b(\omega_{p,1}^2 + \omega_{p,2}^2 + \omega_{p,3}^2 + \omega_{p,4}^2) \end{aligned}$$

with gravitational Terms in blue, $\mathcal{B}FG = C_{EB}^T \cdot E \vec{n}_z mg$

\Rightarrow Only z-Axis can be controlled directly with $\omega_{p,i}$

Note: To be consistent with the lecture notation: $(\omega_x, \omega_y, \omega_z) = (p, q, r)$ and $(v_x, v_y, v_z) = (u, v, w)$

4.2 Control of a Quadrotor

The system has 6 DoF, but only 4 Inputs (Motors) \rightarrow **Under-actuated!**
 \Rightarrow Forward Motion requires tipping around Roll and Pitch.

Define Virtual control inputs:

By defining a new set of inputs, we can decouple the Dynamic equations

Total Thrust Moments along axis

$$\begin{aligned} U_1 &= b(\omega_{p1}^2 + \omega_{p2}^2 + \omega_{p3}^2 + \omega_{p4}^2) \\ U_2 &= l \cdot b(\omega_{p4}^2 - \omega_{p2}^2) \\ U_3 &= l \cdot b(\omega_{p1}^2 - \omega_{p3}^2) \\ U_4 &= d(\omega_{p1}^2 - \omega_{p2}^2 + \omega_{p3}^2 - \omega_{p4}^2) \end{aligned}$$

which simplify the dynamics from above.

Linearize Attitude Dynamics

Linearization around the Equilibrium $(\omega_x, \omega_y, \omega_z = \phi = \theta = U_{2,3,4} = 0; U_1 = mg)$ gives further simplification (Recall: $E_r = \mathbb{1} \Rightarrow \dot{\chi}_r = [\ddot{\phi} \ \ddot{\theta} \ \ddot{\psi}]^T$): \Rightarrow Can use 3 individual controller

$$\dot{\omega}_x = \ddot{\phi} = \frac{1}{I_{xx}} U_2 \quad \dot{\omega}_y = \ddot{\theta} = \frac{1}{I_{yy}} U_3 \quad \dot{\omega}_z = \ddot{\psi} = \frac{1}{I_{zz}} U_4$$

Attitude Control:

Dynamics from above, but in *Inertial* Frame:

$$\ddot{v}_z = \ddot{z} = g - \cos \phi \cos \theta \frac{1}{m} U_1 =: g - \frac{1}{m} T_z$$

Now we can derive the input U_1 for a chosen controller T_z (representing the Thrust in the inertial frame):

$$T_z = -k_p(z_{des} - z) + k_d \dot{z} - mg \Rightarrow U_1 = \frac{T_z}{\cos \phi \cos \theta}$$

Position Control:

We can use 3 separate PD Controller to get the trust for x,y and z in the inertial frame.

$$\text{Dynamics} \Rightarrow [\ddot{x} \ \ddot{y} \ \ddot{z}]^T = \frac{1}{m} [T_x \ T_y \ T_z]^T + [0 \ 0 \ g]^T$$

This must be transformed to get the desired Total thrust as well as roll and pitch angles.

$$T = \sqrt{T_x^2 + T_y^2 + T_z^2} \quad \& \quad \frac{1}{T} C_{E1}^T(z, \psi) \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} = \begin{bmatrix} \sin \theta \cos \phi \\ -\sin \phi \\ \cos \theta \cos \phi \end{bmatrix}$$

Remark: $C_{E1}(z, \psi)$ is the rotation matrix around z with angle ψ .

Remark2: $\begin{bmatrix} \sin \theta \cos \phi \\ -\sin \phi \\ \cos \theta \cos \phi \end{bmatrix}$ describes the Thrust of the Quadrotor in the

Inertial frame (can only apply thrust perpendicular to the rotors). It's calculated with $C_{12}(y, \theta) \cdot C_{2B}(x, \phi) \cdot (0, 0, 1)^T$

4.3 Propeller Aerodynamics

There are 4 main forces generated by the rotor:

For a Rotor in Hover:

Thrust Force T

Aerodynamic force perpendicular to rotor plane

$$|T| = \frac{\rho}{2} A_P C_T (\omega_p R_P)^2$$

For a Rotor in forward flight:

Hub force H

Opposite to horizontal flight direction V_H

$$|H| = \frac{\rho}{2} A_P C_H (\omega_p R_P)^2$$

Drag Torque Q

Torque around rotor plane

$$|Q| = \frac{\rho}{2} A_P C_Q (\omega_p R_P)^2 R_P$$

Rolling Moment R

Around flight direction

$$|R| = \frac{\rho}{2} A_P C_R (\omega_p R_P)^2 R_P$$

4.3.1 Momentum Theory

ρ : fluid density, \vec{V} : flow speed, \vec{n} : surface normal,

dA : surface Area patch, p : surface pressure, E : Energy, P : Power

Conservation of fluid mass

Mass flow inside and outside (closed) control Volume must be equal

$$\iint \rho \vec{V} \cdot \vec{n} dA = 0$$

Conservation of fluid Momentum:

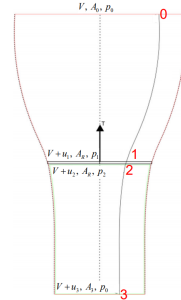
The net Force is the change of momentum of the fluid

$$\iint p \cdot \vec{n} dA + \iint (\rho \vec{V}) \vec{V} \cdot \vec{n} dA = \vec{F}$$

Conservation of energy:

Work done on the fluid results in a gain of kinetic energy

$$\iint \frac{1}{2} \rho V^2 \vec{V} \cdot \vec{n} dA = \frac{dE}{dt} = P$$



1D Analysis:

The formulas lead to the following results:
 $\rho A_0 V = \rho A_R (V + u_1)$
 $= \rho A_R (V + u_2) = \rho A_R (V + u_3)$
 $\Rightarrow u_1 = u_2$

$$\begin{aligned} F_{Thrust} &= \rho A_R (V + u_1) u_3 = 2 \rho A_R (V + u_1) u_1 \\ P_{Thrust} &= F_{Thrust} (V + u_1) \\ &= \frac{1}{2} \rho A_R (V + u_1) (2V + u_3) u_3 \\ &\Rightarrow u_3 = 2u_1 \end{aligned}$$

In the Hover case ($V = 0$):

Thrust Force: $F_{Thrust} = 2 \rho A_R u_1^2$

Slipstream Tube: $A_0 = \infty \quad A_3 = \frac{A_R}{2}$

Combining $P = F_{Thrust} (V + u_1)$ with $F_{Thrust} = 2 \rho A_R u_1^2$ gives the Ideal Power to Hover:

$$P = \frac{F_{Thrust}^{3/2}}{\sqrt{2 \rho A_R}} = \frac{(mg)^{3/2}}{\sqrt{2 \rho A_R}} \quad \text{with } F_{Thrust} = mg$$

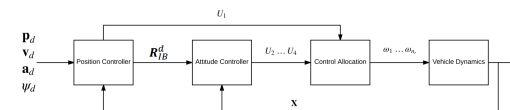
The Power depends on the **Disc Loading** := F_{Thrust} / A_R

Defining the rotor efficiency, Figure of Merit, FM

$$FM = \frac{\text{Ideal power to hover}}{\text{Actual power to hover}} < 1$$

\rightarrow compare different propellers with the same disc loading.

5 Case Study: Micro Aerial Vehicles



5.1 Control for MAVs

Virtual Control Input (allocation):

$$\begin{pmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{pmatrix} = A \begin{pmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{pmatrix}, \quad A \in \mathbb{R}^{4 \times n_r}$$

5.1.1 Trajectory Tracking Controller

Error definitions: Position $e_p = p - p_d$, Velocity $e_v = v - v_d$

$$z_B^d = \frac{-K_p e_p - K_v e_v - m(g - a_d)}{||-K_p e_p - K_v e_v - m(g - a_d)||}$$

$$I z_{temp}^d = \begin{pmatrix} \cos \psi_d \\ \sin \psi_d \\ 0 \end{pmatrix} \quad I y_B^d = \frac{I z_B^d \times I x_{temp}^d}{||I z_B^d \times I x_{temp}^d||}$$

From this we can construct the Rotation Matrix $R_{IB}^d = [I y_B^d \times I z_B^d, I y_B^d, I z_B^d]$

Note: $I x_{temp}^d$ is generally not perpendicular to $I z_B^d$, so we use some orthogonal projections for the first entry of R_{IB}^d

Attitude Control

$$\begin{pmatrix} U_2 \\ U_3 \\ U_4 \end{pmatrix} = -K_R e_R - K_\omega e_\omega + \omega \times J \omega$$

with errors:

$$e_R = \frac{1}{2} \left((R_{IB}^d)^T R_{IB} - R_{IB}^T R_{IB}^d \right)^\vee, \quad e_\omega = \omega - R_{IB}^T R_{IB}^d \omega_d$$

where $(\cdot)^\vee$ maps the cross product matrix to a vector.

Trajectory Tracking

Project Thrust onto Body z-Axis:

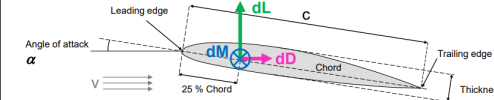
$$U_1 = (-K_p e_p - K_v e_v - m(g - a_d)) \cdot I \vec{n}_{z,B}$$

6 Fixed Wing UAVs

6.1 Basic Principles

For incompressible and non viscous fluids, we can use the Bernoulli Equation: $\frac{v^2}{2} + gh + \frac{p}{\rho} = \text{const.}$ This results in a **Lift** and a **Drag**

Force with an additional **Moment** on the Airfoil. Lift/ Drag is always perpendicular/parallel to the air velocity!



Stall Point: The angle of attack at which the maximum lift occurs. (Flow will separate for higher α)

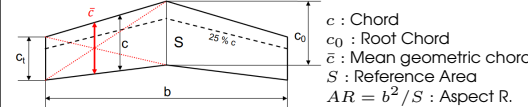
6.2 Fixed Wing Kinematics

Assumptions: Aerodynamics where we don't enter Stall, neglect side-forces and interference effects

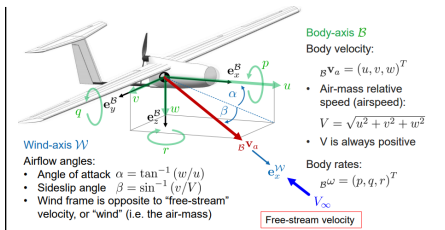
Inertial frame: X-Axis: North; Y-Axis: East; Z-Axis: Downwards

$\rightarrow \phi, \theta, \psi, \quad C_{TB} = C_{T1}(\psi) C_{T2}(\theta) C_{2B}(\phi)$

Wing Geometry



Control Surfaces: The standard Control surfaces are **Elevator** (pitch), **Aileron** (roll), **Rudder** (yaw).



Flight Path Angle: γ , defined from horizon to $I v_a$

Heading Angle: ξ , defined from North to $I v_a$

Course Angle: χ , defined from North to $I v = I v_a + I v_{wind}$

6.3 Fixed Wing Dynamics

6.3.1 Forces

Non Aerodynamics: Weight at CoG (mg), Propeller Thrust T .
Aerodynamic:

$$\text{Lift: } L = \frac{1}{2} \rho V^2 S c_L, \quad \text{Drag: } D = \frac{1}{2} \rho V^2 S c_D$$

with surface S . Note: c_L and c_D are dependent on α , Side-Forces: assumed zero

Moments: (Rolling, Pitching, Yawing)

$$L_m = \frac{1}{2} \rho V^2 S b c_{l_l} \quad M_m = \frac{1}{2} \rho V^2 S \bar{c} c_{m_m} \quad N_m = \frac{1}{2} \rho V^2 S b c_{n_n}$$

6.3.2 Equations of Motion, ($I x_{zz}$ is typically small)

constant wind: $\frac{d}{dt} (I v = I v_a + I v_w) = \frac{d}{dt} (I v_a)$; $\dot{\Theta} = E_r(\Theta)^{-1} \mathcal{B} \dot{\omega}$

$$\mathcal{B} \dot{v}_a = \frac{1}{m} \sum \mathcal{B} F - \mathcal{B} \omega \times \mathcal{B} v_a, \quad I \dot{r} = C_{TB} \mathcal{B} v_a + I v_w$$

$$\mathcal{B} \dot{\omega} = \mathcal{B} I^{-1} \left(\sum \mathcal{B} M - \mathcal{B} \omega \times (\mathcal{B} I \mathcal{B} \omega) \right); \quad \mathcal{B} I = \begin{bmatrix} I_{xx} & 0 & I_{xz} \\ 0 & I_{yy} & 0 \\ I_{xz} & 0 & I_{zz} \end{bmatrix}$$

6.4 Cascaded Control Loops

6.4.1 Steady Level Turning

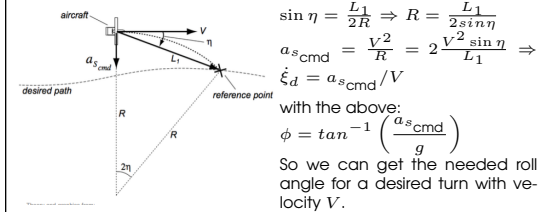
We need $\mathcal{B} \dot{v}_a = \mathcal{B} \dot{\omega} = 0$; $\theta = \alpha \rightarrow \gamma = 0$ and $\phi = \text{const} \neq 0$.

- L increases with $\frac{1}{\cos \phi} \leftarrow (L = \frac{mg}{\cos \phi})$

- Air speed V_{min} increases with $\sqrt{1/\cos \phi}$

The Heading rate $\dot{\xi}$ can be found with a force balance with the centripetal force (and $\psi \approx \xi$). Note: $\dot{\xi} = V/R \Leftrightarrow v = r \cdot \omega$
 $L \cos \phi = mg$; $D = T$; $F_{cent} = m \frac{V^2}{R} = L \sin \phi$; $\Rightarrow \dot{\psi} = \dot{\xi} = g \tan \phi / V$

Lateral-directional path following guidance



6.4.2 TECS - Total Energy Control System

$$\dot{E}_{spec} = \frac{\dot{V}}{g} + \sin \gamma \approx \frac{\dot{V}}{g} + \gamma, \quad \dot{E}_{dist} = \gamma - \frac{\dot{V}}{g}$$

'Potential energy rate minus kinetic energy rate'