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: $\chi_{Pc} = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$

Postion Vector:
$$_{\mathcal{A}}r=xe_{x}^{\mathcal{A}}+ye_{y}^{\mathcal{A}}+ze_{z}^{\mathcal{A}}=\begin{pmatrix}x\\y\\z\end{pmatrix}$$

Cylindric Coordinates

Spherical Coordinates

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

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: $E_{Pc}(\chi_{Pc})=E_{Pc}^{-1}(\chi_{Pc})=\mathbb{1}$ In Cylindrical we get:

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

For **Spherical**

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

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. $_{A}u=C_{AB}\cdot _{B}u$

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

1.2.1 Rotation Matrix

Mapping Coordinates P from frame $\mathcal B$ to $\mathcal A$:

$$A_{TAP} = \begin{pmatrix} A e_x^{\mathcal{B}} & A e_y^{\mathcal{B}} & A e_z^{\mathcal{B}} \end{pmatrix} \cdot B_{TAP}$$
$$= C_{AB} \cdot B_{TAP}$$

This matrix is orthogonal: $C_{\mathcal{B}\mathcal{A}} = C_{\mathcal{A}\mathcal{B}}^{-1} = C_{\mathcal{A}\mathcal{B}}^{T}$

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 a offset. The vector $_{\mathcal{A}^{T}AB}$ points from the origin of frame \mathcal{A} to the origin of the \mathcal{B} in frame \mathcal{A} . $T_{\mathcal{A}\mathcal{B}}$ transforms a **point P** from frame \mathcal{B} to \mathcal{A} .

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

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

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:

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

1.3.2 Angle Axis

$$\chi_{R, \mathsf{AngleAxis}} = \left(rac{ heta}{ec{n}}
ight)$$
 Rotation Vector: $arphi = heta \cdot \overrightarrow{n}$

Rotation Matrix from Angle Axis:

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

Parameters from Rotation Matrix:

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

1.3.3 Unit Quaternion

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

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

Hamiltonian conventions:

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

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

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

Given a rotation matrix, the corresponding quaternion is:

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

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

The Corresponding Rotation Matrix can be calculated with:

$$\begin{split} C_{\mathcal{A}\mathcal{D}} &= \mathbbm{1}_{3\times 3} + 2\xi_0 \, [\check{\xi}]_\times + 2[\check{\xi}]_\times^2 \\ &= (2\xi_0^2 - 1) \, \mathbbm{1}_{3\times 3} + 2\xi_0 [\check{\xi}]_\times + 2\check{\xi}\check{\xi}^T \\ &= \begin{pmatrix} \xi_0^2 + \xi_1^2 - \xi_2^2 - \xi_3^2 & 2\xi_1\xi_2 - 2\xi_0\xi_3 & 2\xi_0\xi_2 + 2\xi_1\xi_3 \\ 2\xi_0\xi_3 + 2\xi_1\xi_2 & \xi_0^2 - \xi_1^2 + \xi_2^2 - \xi_3^2 & 2\xi_2\xi_3 - 2\xi_0\xi_1 \\ 2\xi_1\xi_3 - 2\xi_0\xi_2 & 2\xi_0\xi_1 - 2\xi_2\xi_3 & \xi_0^2 - \xi_1^2 - \xi_2^2 + \xi_3^2 \end{pmatrix} \end{split}$$

It holds that:

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

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

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

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

$$p(Ar) = \xi_{AB} \otimes p(Br) \otimes \xi_{AB}^{T} = M_{l}(\xi_{AB}) M_{r}(\xi_{AB}^{T}) p(Br)$$

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 ${}_{\mathcal A}\omega_{\mathcal A}\mathcal B$. It can be shown that:

$$\begin{split} [_{\mathcal{A}}\omega_{\mathcal{A}\mathcal{B}}]_{\times} &= \dot{C}_{\mathcal{A}\mathcal{B}} \cdot C_{\mathcal{A}\mathcal{B}}^T \qquad \omega_{\mathcal{A}\mathcal{B}} = -\omega_{\mathcal{B}\mathcal{A}} \\ \text{with } [_{\mathcal{A}}\omega_{\mathcal{A}\mathcal{B}}]_{\times} &= \begin{pmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ \omega_x & 0 \end{pmatrix} \text{ and } {}_{\mathcal{A}}\omega_{\mathcal{A}\mathcal{B}} = \begin{pmatrix} \omega_x \\ \omega_y \\ \omega_y \end{pmatrix} \end{split}$$

Angular Velocities can be transformed as normal vectors:

$$_{\mathcal{B}}\omega_{\mathcal{A}\mathcal{B}}=C_{\mathcal{B}\mathcal{A}}\cdot_{\mathcal{A}}\omega_{\mathcal{A}\mathcal{B}}$$

the product matrix is transformed by:

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

Angular Velocities in the same Frame can be added:

$$\tau \omega_{\mathcal{I}\mathcal{E}} = \tau \omega_{\mathcal{I}0} + \dots + \tau \omega_{n\mathcal{E}}$$

1.4.1 Time Derivative of Rotation Parameters

Similar to lin. vel. we can define:
$${}_A\omega_{AB}=E_R(\chi_R)\cdot\dot{\chi}_R$$
 quaternions: $E_R=2H(\xi)=2\left[-\xi\left[\xi\right]_X+\xi_01_{3\times3}\right]$ Angle Axis: $E_R=\left[n\sin\theta I_{3\times3}+(1-\cos\theta)[n]_X)\right]$ Rotation Vector: $E_R=\left[I_{3\times3}+[\varphi]_X\left(\frac{1-\cos\theta}{\|\varphi\|^2}\right)+[\varphi]_X^2\left(\frac{\|\varphi\|-\sin\|\varphi\|}{\|\varphi\|^3}\right)\right]$

1.5 Velocity in Moving Bodies

$$\begin{array}{c|c} v_P & \text{the absolute Velocity of point P} \\ a_P = \dot{v}_P & \text{(absolute) acceleration of P} \\ \Omega_{\mathcal{B}} = \omega_{\mathcal{A}\mathcal{B}} & \text{(absolute) angular velocity of body B} \\ \Psi_{\mathcal{B}} = \dot{\Omega}_{\mathcal{B}} & \text{(absolute) angular acceleration of body B} \end{array}$$

We can write the Position P as:

$$Ar_{AP} = Ar_{AB} + C_{AB} \cdot Br_{BP}$$

Which can be differentiated to:

$$A\dot{r}_{AP} = A\dot{r}_{AB} + A\omega_{AB} \times Ar_{BP}$$

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

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

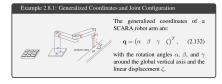
For Accelerations: $a_P=a_B+\Psi\times r_{BP}+\Omega\times (\Omega\times r_{BP})$ In case a moving system ${\cal B}$ is used for representation, the **Euler Differentiation rule** must be applied (with a non moving system ${\cal A}$):

$$\beta v_P = C_{\mathcal{B}\mathcal{A}} \cdot \frac{d}{dt} (C_{\mathcal{A}\mathcal{B}} \cdot \beta^r A_P)$$
$$= \beta \dot{r}_{AP} + \beta \omega_{\mathcal{A}\mathcal{B}} \times \beta^r A_P$$

1.6 Kinematics of Systems of Bodies

.6.1 Generalized Coordinates and Joint Configurations

The configuration of a root such as a manipulator can be described by the **generalized coordinate vector**: $q=\begin{pmatrix}q_1&\dots&q_n\end{pmatrix}^T$ 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 $\chi_1 \dots \chi_{m_0}$ are independent operational space coordinates. It can be seen as a **minimal selection** of the above endeffector parameters.

1.6.3 Forward Kinematics

Forward kinematics map from joint coordinates q to the end-effector configuration $\chi_e\colon\quad \chi_e=\chi_e(q)$

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

1.6.4 Differential Kinematics and Analytical Jacobian

To linearise the forward kinematics we use a first order approximation:

$$\Delta\chi_{e} \approx J_{eA}(q) \,\Delta q, \quad J_{eA}(q) = \begin{bmatrix} \frac{\partial\chi_{1}}{\partial q_{1}} & \cdots & \frac{\partial\chi_{1}}{\partial q_{n_{j}}} \\ \cdots & \cdots & \cdots \\ \frac{\partial\chi_{m}}{\partial q_{1}} & \cdots & \frac{\partial\chi_{m}}{\partial q_{n_{j}}} \end{bmatrix}$$

It results in an exact relation between velocities:

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

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

$$J_{eA} = \begin{bmatrix} J_{eA_p} \\ J_{eA_R} \end{bmatrix} = \begin{bmatrix} \frac{\partial \chi_{eP}}{\partial q} \\ \frac{\partial \chi_{eR}}{\partial q} \end{bmatrix}$$

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):

$$w_e = \begin{pmatrix} v_e \\ \omega_e \end{pmatrix} = J_{e0}(q) \cdot \dot{q}$$

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 $w_{\mathcal{C}} = w_{\mathcal{B}} + w_{\mathcal{B}\mathcal{C}}$ we can derive, that geometric Jacobians can simply be added (in the same reference):

$$_{A}J_{C} = _{A}J_{B} + _{A}J_{BC}$$

Geometric Jacobian

$${\scriptstyle \mathcal{I}J_{e0} = \begin{bmatrix} \mathcal{I}J_{e0}_P \\ \mathcal{I}J_{e0}_R \end{bmatrix} = \begin{bmatrix} \mathcal{I}^{n_1} \times \mathcal{I}^{r_1}(n+1) & \cdots & \mathcal{I}^{n_n} \times \mathcal{I}^{r_n}(n+1) \\ & \mathcal{I}^{n_1} & \cdots & \mathcal{I}^{n_n} \end{bmatrix} }$$

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

$$\omega_{(k-1)k} = n_k \dot{q}_k$$

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 \mathcal{E}_{L} in joint direction. The Rotational part (n_i) is obviously zero.

Mapping from Analytic to Geometric Jacobian, it holds that:

$$J_{e0}(q)=E_e(\chi)J_{eA}(q)$$
 with $E_e(\chi)=\left[egin{array}{cc} E_p(\chi)&0\\0&E_R(\chi) \end{array}\right]\in\mathbb{R}^{6\times m_e}$.

1.7 **Kinematic Control Methods**

171 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^+

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

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

$$A^{+} = A^{T} (AA^{T})^{-1}$$
 right inverse (full row rank) \square
 $A^{+} = (A^{T} A)^{-1} A^{T}$ 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 endeffector velocity. This can be handled by using a damped solution. $\dot{q} = J_{e0}^T (J_{e0} J_{e0}^T + \lambda^2 1)^{-1} w_s^*$

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:

$$\dot{q} = J_{a0}^{+} \cdot w_{a}^{*} + N\dot{q}_{0}$$

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

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

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

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:

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

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

Multitask Prioritisation

An approach for prioritisationing 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

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

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

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_{+} tasks this can be written recursively

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

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

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: $\Delta\chi_e = J_{eA}\Delta q$

Alaorithmus 1: Numerical Inverse Kinematics

$$\begin{array}{l} q \leftarrow q^0 : & \text{// Start Configuration} \\ \text{while} & ||\chi_e^* - \chi_e(q)|| > tol \ \text{do} \\ & J_{eA} \leftarrow J_{eA}(q) = \frac{\partial \chi_e}{\partial q}(q) : & \text{// Evaluate (local) Jacobian} \\ & J_{eA}^+ \leftarrow (J_{eA})^+ : & \text{// Calculate Pseudo Inverse} \\ & \Delta\chi_e \leftarrow \chi_e^* - \chi_e(q) : & \text{// Find Error Vector} \\ & q \leftarrow q + J_{eA}^+ \Delta\chi_e : & \text{// Update generalized Coordinates} \\ & \text{end} \end{array}$$

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

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 augranteed.

Shortest Path rotation

For a straight rotation along the "shortest path", we rotate along the rotation vector $\Delta \varphi$.

The rotation Matrix is given by:

$$C_{\mathcal{A}\mathcal{B}}(\Delta\varphi) = C_{\mathcal{I}\mathcal{A}}(\varphi^t)^T C_{\mathcal{I}\mathcal{B}}(\varphi^*)$$
(Note that $\Delta\varphi \neq \varphi^* - \varphi^t$

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

$$_{\mathcal{A}}\Delta\varphi = _{\mathcal{B}}\Delta\varphi = \text{rotVec}(C_{\mathcal{A}\mathcal{B}})$$

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

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

Now we can change the update step 6 of the algorithm to:

$$q \leftarrow q + k_{p_R \mathcal{I}} J_{e0_R \mathcal{I}}^+ \Delta \varphi$$

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 $\dot{r}_e^*(t)$:

$$\label{eq:decomposition} \dot{\boldsymbol{q}}^* = \boldsymbol{J}_{e0}^+_P(\boldsymbol{q}^t) \cdot (\dot{\boldsymbol{r}}_e^*(t) + k_{PP} \, \Delta \boldsymbol{r}_e^t) \quad \text{with } \Delta \boldsymbol{r}_e^t = \boldsymbol{r}_e^*(t) - \boldsymbol{r}_e(\boldsymbol{q}^t)$$

Similar for orientation $\chi_{R}^{*}(t)$ and angular velocity $\omega^{*}(t)$:

$$\dot{q}^* = J_{e0_R}^+(q^t) \cdot (\omega_e^*(t) + k_{p_R} \Delta \varphi) \quad \text{ with } \Delta \varphi \text{ from above}.$$

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) $\dot{q}, \ddot{q}, \ddot{q}$ Generalized position, velocity and acceleration vectors Coriolis and centrifugal terms $b(q, \dot{q})$ Gravitational terms

External aeneralized forces External cartesian forces (e.g., from contacts)

Geometric Jacobian corresponding to external forces

2.1 Principle of virtual Work

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

infinitesemal mass element external Forces acting on element dm dF_{ext} acceleration of element dm virtual displacement of dm Body System containing infinitesemal 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.

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): $rac{d}{dt}\left(rac{\partial {\cal L}}{\partial \dot a}
ight)-\left(rac{\partial {\cal L}}{\partial a}
ight)= au\Rightarrow$ $\frac{d}{dt}\left(\frac{\partial \mathcal{T}}{\partial \dot{q}}\right) - \frac{\partial \mathcal{T}}{\partial q} + \frac{\partial \mathcal{U}}{\partial q} = \tau$ with total gen. ext. forces τ 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 $\frac{1}{2}mv^2 \& \frac{1}{2}J\omega^2$):

$$\mathcal{T} = \sum_{i=1}^{n_b} \left(\frac{1}{2} m_{i,\mathcal{A}} \dot{r}_{S_i,\mathcal{A}}^T \dot{r}_{S_i} + \frac{1}{2} \mathbf{B} \boldsymbol{\Omega}_{S_i}^T \cdot \mathbf{B} \boldsymbol{\Theta}_{S_i} \cdot \mathbf{B} \boldsymbol{\Omega}_{S_i} \right)$$

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

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

2.3.2 Potential Energy

Knowing $r_{S_{\vec{s}}}$ to the Center of Mass of each body, we can calculate gravitational forces (Zero energy level can be choosen arbitrarily):

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

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

and $F_j=k_j(d(q)-d_0)rac{d(q)-d_0}{\|d(q)-d_0\|}$, with the current length of

Projected Euler Method

$$\left[\begin{smallmatrix} v_S \\ \Omega \end{smallmatrix}\right] = \left[\begin{smallmatrix} J_P \\ J_R \end{smallmatrix}\right] \dot{q} \qquad \left[\begin{smallmatrix} a_S \\ \Psi \end{smallmatrix}\right] = \left[\begin{smallmatrix} J_P \\ J_R \end{smallmatrix}\right] \ddot{q} + \left[\begin{smallmatrix} \dot{J}_P \\ \dot{J}_R \end{smallmatrix}\right] \dot{q}$$

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

$$M = \sum_{i=1}^{n_0} (_{\mathcal{A}} J_{S_i}^T \cdot m \cdot _{\mathcal{A}} J_{S_i} + _{\mathcal{B}} J_{R_i}^T \cdot _{\mathcal{B}} \Theta_{S_i} \cdot _{\mathcal{B}} J_{R_i})$$

$$b = \sum_{i=1}^{n_b} (\mathcal{A} J_{S_i}^T m_{\mathcal{A}} \dot{J}_{S_i} \dot{q}$$

$$+ \mathop{\mathcal{B}} J_{R_i}^T (\mathop{\mathcal{B}} \Theta_{S_i} \cdot \mathop{\mathcal{B}} \dot{J}_{R_i} \cdot \dot{q} + \underbrace{\mathop{\mathcal{B}} \Omega_{S_i} \times \mathop{\mathcal{B}} \Theta_{S_i} \cdot \mathop{\mathcal{B}} \Omega_{S_i}}_{J_{R_i} \dot{q} = 0 \text{ in 2D}}))$$

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

2.4.1 External Forces & Actuation

For known Forces F_i acting on the system, we can calculate the generalized forces $\tau_{F,ext}$ (due to the external force).

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

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

Similar for external Torques T_i : $\tau_{T,ext} = \sum_i J_{R,i}^T T_i$

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:

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

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

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

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

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:

$$\dot{w}_e = \begin{pmatrix} \ddot{r} \\ \dot{\omega} \end{pmatrix} = J_e \ddot{q} + \dot{J}_e \dot{q} \Rightarrow \ddot{q}^* = J_e^+ \left(w^* - \dot{J}_e \dot{q} \right)$$

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:

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

2.6.2 End-Effector Dynamics

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

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

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

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

End-Effector Motion Control

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

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

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

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

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

Operational Space Control

Note: We need to extend the end effector dynamics with a contact Force F_{α} :

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

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

$$\Sigma_{p} = \begin{pmatrix} \sigma_{px} & 0 & 0 \\ 0 & \sigma_{py} & 0 \\ 0 & 0 & \sigma_{pz} \end{pmatrix} \quad \Sigma_{r} = \begin{pmatrix} \sigma_{rx} & 0 & 0 \\ 0 & \sigma_{ry} & 0 \\ 0 & 0 & \sigma_{rz} \end{pmatrix}$$

with σ_i either 1(move) or 0(don't).

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

$$S_M = \begin{pmatrix} C^T \Sigma_p C & 0 \\ 0 & C^T \Sigma_T C \end{pmatrix} \quad S_F = \begin{pmatrix} C^T (1 - \Sigma_p) C & 0 \\ 0 & C^T (1 - \Sigma_p) C \end{pmatrix} \qquad \mathcal{I}_{Q}(q) = \begin{bmatrix} 1_{3 \times 3} & -C_{\mathcal{I}\mathcal{B}} \cdot [\mathbf{g}^T B_Q] \times & C_{\mathcal{I}\mathcal{B}} \cdot \mathbf{g} J_{Pq_j}(q_j) \\ 0_{3 \times 3} & C_{\mathcal{I}\mathcal{B}} & C_{\mathcal{I}\mathcal{B}} \cdot \mathbf{g} J_{Rq_j} \end{bmatrix}$$

Least Square Optimisation

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:

$$\left| \begin{array}{cc} \tau = M\ddot{q} + b + g \\ \dot{w} = J\ddot{q} + \dot{J}\dot{q} \end{array} \right| \Rightarrow \left| \begin{array}{cc} \left[M & -\mathbbm{1} \right] \left(\ddot{q} \\ \tau \right) + b + g = 0 \\ \left[J_e & 0 \right] \left(\ddot{q} \\ \tau \right) + \dot{J}_e \dot{q} = \dot{w}^* \end{array} \right|$$

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

It can be solves as single (stacked) tasks

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

Or with different priorities

$$\begin{split} \min_{\ddot{q},\tau} \left\| \begin{bmatrix} J_e & 0 \end{bmatrix} \begin{pmatrix} \ddot{q} \\ \tau \end{pmatrix} - (\dot{w}_e^* - \dot{J}_e \dot{q}) \right\|_2 \\ \text{such that: } \begin{bmatrix} M & -1 \end{bmatrix} \begin{pmatrix} \ddot{q} \\ \tau \end{pmatrix} - (-b - g) = 0 \end{split}$$

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

Floating Base Systems

FB Kinematics

Free floating robots are described by n_b unactuated base coordinates q_b and n_i actuated joint coordinates q_i .

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

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

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

$$u = \begin{pmatrix} \mathcal{I}^{v_B} \\ \mathcal{B}^{\omega} \mathcal{I} \mathcal{B} \\ \dot{\varphi}_1 \\ \vdots \\ \dot{\varphi}_{n_j} \end{pmatrix} \text{ with mapping } \begin{aligned} u &= E_{fb} \cdot \dot{q}, \\ u &= E_{fb} \cdot \dot{q}, \\$$

3.1.1 Forward Kinematics

The position vector of point Q can be expressed via the Base ${\cal B}$

$$_{\mathcal{I}}r_{IO}(q) = _{\mathcal{I}}r_{IB}(q) + C_{\mathcal{I}\mathcal{B}}(q) \cdot _{\mathcal{B}}r_{BO}(q)$$

3.1.2 Differential Kinematics

The spacial Jacobian maps u to v and ω :

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

$$_{\mathcal{I}}J_{Q}(q) = \begin{bmatrix} \mathbb{1}_{3\times3} & -C_{\mathcal{I}\mathcal{B}} \cdot [_{\mathcal{B}}r_{BQ}]_{\times} & C_{\mathcal{I}\mathcal{B}} \cdot _{\mathcal{B}}J_{Pq_{j}}(q_{j}) \\ 0_{3\times3} & C_{\mathcal{I}\mathcal{B}} & C_{\mathcal{I}\mathcal{B}} \cdot _{\mathcal{B}}J_{Rq_{z}} \end{bmatrix}$$

3.1.3 Contacts & Constraints

I Slides 6.5

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

$$_{\mathcal{I}}J_{C_{i}}u = 0$$
, $_{\mathcal{I}}J_{C_{i}}\dot{u} + _{\mathcal{I}}\dot{J}_{C_{i}}u = 0$

where multiple J_C , can be stacked for multiple contact points.

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

$$J_c = \begin{bmatrix} J_{c,b} \ J_{c,j} \end{bmatrix} = \begin{bmatrix} \frac{\partial r_c}{\partial q_b} \ \frac{\partial r_c}{\partial q_j} \end{bmatrix} \in \mathbb{R}^{n_c \times (n_b + n_j)}$$

If the rank of $J_{c,b}$ has full rank (=6 in 3D), the joints can move the body in every direction. The difference $rank(J_c) - rank(J_{c,b})$ is the number of internal kinematic constraints (i.e. leas 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:

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

Given a demanded motion w_t we can calculate the required velocity: $w_t = J_t u \implies u = N_c (J_t N_c)^+ w_t$

3.2 **FB Dynamics**

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

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

with
$$u_j = Su = S \begin{pmatrix} u_b \\ u_j \end{pmatrix} = \begin{bmatrix} 0_{n_j \times 6} & \mathbb{1}_{n_j \times n_j} \end{bmatrix} \begin{pmatrix} u_b \\ u_j \end{pmatrix}$$

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

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

Together with the contact constraints we can calculate the con-

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

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):

$$\boldsymbol{N}_{c}^{T}(\boldsymbol{M}\dot{\boldsymbol{u}}+\boldsymbol{b}+\boldsymbol{g})=\boldsymbol{N}_{c}^{T}\boldsymbol{S}^{T}\boldsymbol{\tau}$$

3.2.2 FB Inverse Dynamics

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

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

When taking only the first part without a Nullspace ($au_0^*=0$), the solution is the least square minimal torque au^* that fullfils the EoM.

Rotorcrafts

a power

driven main rotor,

which can be

Overview

Helicopter



facing

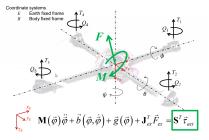
Gyrodyne

Active main ro and a forward tor, but can't be tilted. Additional active propeller. Can't front facing active propeller

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

Modelling of Quadrotor

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



Structural Properties:

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

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

 $\begin{pmatrix} \phi \\ \theta \\ \psi \end{pmatrix} = \begin{pmatrix} Roll \\ Pitch \\ Yaw \end{pmatrix} \rightarrow \begin{pmatrix} xAxis \\ yAxis \\ zAxis \end{pmatrix}$ with the known rot. matrices.

This can be used for a equation for the rotational speed:

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

Linearization for small Roll and Pitch ($\phi \approx \theta \approx 0$) results in a **unity** matrix $E_r = 1$

4.1.1 Body Dynamics

$$\begin{bmatrix} m\mathbbm{1}_{3\times3} & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} \mathbf{g}\dot{v} \\ \mathbf{g}\dot{\omega} \end{bmatrix} + \begin{bmatrix} \mathbf{g}\omega\times m\mathbf{g}v \\ \mathbf{g}\omega\times I\mathbf{g}\omega \end{bmatrix} = \begin{bmatrix} \mathbf{g}F \\ \mathbf{g}M \end{bmatrix}$$

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

and Hover Moments

$$_{\mathcal{B}}M_{\mathsf{Aero}} = \left(egin{matrix} l(T_4 - T_2) \\ l(T_1 - T_3) \\ 0 \end{pmatrix} + \left(egin{matrix} 0 \\ 0 \\ \sum\limits_{i=1}^4 Q_i(-1)^{(i-1)} \end{pmatrix}
ight)$$

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

Rotational Dynamics (fist row of Dynamics)

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

with ω_{xyz} as entries of the body rotation $\kappa\omega$

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

Translational Dynamics (2nd row of dynamics)

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

with gravitational Terms in blue, $_{B}F_{G}=C_{EB}^{T}\cdot _{E}\vec{n}_{z}mg$ \Rightarrow Only z-Axis can be controlled directly with $\omega_{n,i}$

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

Control of a Quadrotor

The system has 6 DoF, but only 4 Inputs (Motors) → Under-actuated! ⇒ 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

Moments alona axis

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

which simplify the dynamics from above

Linearize Attitude Dynamics

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

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

Altitude Control:

Dynamics from above, but in Inertial Frame:

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

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

Dynamics
$$\Rightarrow \begin{bmatrix} \ddot{x} & \ddot{y} & \ddot{z} \end{bmatrix}^T = \frac{1}{m} \begin{bmatrix} T_x & T_y & T_z \end{bmatrix}^T + \begin{bmatrix} 0 & 0 & g \end{bmatrix}^T$$

This must be transformed to get the desired Total thrust as well as roll

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

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

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

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

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

Drag Torque Q Torque around rotor plane $|Q| = \frac{\rho}{2} A_P C_Q (\omega_p R_P)^2 R_P$

For a Rotor in forward flight: Hub force H

Opposite to horizontal flight direction V_H

Rolling Moment R Around flight direction $|R| = \frac{\rho}{2} A_P C_R (\omega_p R_P)^2 R_P$

$|H| = \frac{\rho}{2} A_P C_H (\omega_p R_P)^2$ 4.3.1 Momentum Theory

ho: fluid density, $ec{V}$: flow speed, $ec{n}$: surface normal,

dA: surface Area patch, p: surface pressure, E: Energy, P: Power Conservation of fluid mass

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

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

Conservation of fluid Momentum:

Conservation of tiula Momentum: The net Force is the
$$\iint p \cdot \vec{n} dA + \iint (\rho \vec{V}) \vec{V} \cdot \vec{n} dA = \vec{F} \quad \text{the fluid}$$

Conservation of energy:

$$\iint \frac{1}{2} \rho V^2 \vec{V} \cdot \vec{n} dA = \frac{dE}{dt} = P$$
 Work done on the fluid results in a gain of kinetic energy

$V + u_1, A_R, p$ $V + u_{\gamma}$, A_{α} , D_{γ}

1D Analysis:

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

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

In the Hover case (V=0): Thrust Force: $F_{Thrust} = 2\rho A_R u_1^2$ Slipstream Tube: $A_0 = \infty$ $A_3 = \frac{A_R}{2}$

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

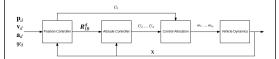
$$P = \frac{F_{\textit{Thrust}}^{3/2}}{\sqrt{2\rho A_R}} = \frac{(mg)^{3/2}}{\sqrt{2\rho A_R}} \; \text{with} \; F_{\textit{Thrust}} = mg$$

The Power depends on the **Disc Loading** F_{Thrust}/A_R Defining the rotor efficiency. Figure of Merit, FM

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

→ compare different propellers with the same disc loading

Case Study: Micro Aerial Vehicles



Control for MAVs

Virtual Control Input (allocation):

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

5.1.1 Trajectory Tracking Controller

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

$$z_{B}^{d} = \frac{-K_{p}e_{p} - K_{v}e_{v} - m(g - a_{d})}{|| - K_{p}e_{p} - K_{v}e_{v} - m(g - a_{d})||}$$

$${}_{I}x_{temp}^{d} = \begin{pmatrix} cos\psi_{d} \\ sin\psi_{d} \\ 0 \end{pmatrix} \quad {}_{I}y_{B}^{d} = \frac{{}_{I}z_{B}^{d} \times {}_{I}x_{temp}^{d}}{||{}_{I}z_{B}^{d} \times {}_{I}x_{temp}^{d}||}$$

From this we can construct the Rotation Matrix $\mathbf{R}_{\mathbf{I}\mathbf{B}}^{\mathbf{d}} = [\mathbf{I}\mathbf{y}_{\mathbf{B}}^{\mathbf{d}} \times \mathbf{I}\mathbf{z}_{\mathbf{B}}^{\mathbf{d}}, \ \mathbf{I}\mathbf{y}_{\mathbf{B}}^{\mathbf{d}}, \ \mathbf{I}\mathbf{z}_{\mathbf{B}}^{\mathbf{d}}]$

Note: ${}_{I}x^{d}_{temp}$ is generally not perpendicular to ${}_{I}z^{d}_{B}$, so we use some orthogonal projections for the first entry of $\mathbf{R}_{\mathbf{IP}}^{\mathbf{d}}$

Attitude Control

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

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

where (.) V maps the cross product matrix to a vector.

Trajectory Tracking

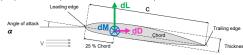
Project Thrust onto Body z-Axis:

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

Fixed Wina UAVs

Basic Principles

For incompressable and non viscous fluids, we can use the Bernoulli Equation: $\frac{v^2}{2} + gh + \frac{p}{a} = const.$ 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!



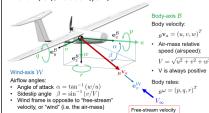
Stall Point: The angle of attack at which the maximum lift occurs. (Flow will separate for higher α)

Fixed Wina Kinematics

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

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

Body-fixed frame: X-Axis: out of the nose; Y-Axis: out of the right wing; Z-Axis: Downwards; The Orientation of the Body Frame is describes as usual in Euler Angles with roll, pitch and yaw Control Surfaces: The standard Control surfaces are Elevator (pitch), Aileron (roll), Rudder (vaw).



Flight Path Angle: γ , defined from horizon to τv_a **Heading Angle:** ξ , defined from North to τv_a

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

Fixed Wina Dynamics 6.3

6.3.1 Forces

Non Aerodynamics: Weight, Propeller Thrust Aerodynamic:

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

Note: c_L and c_D are dependent on α , Side-Forces: assumed zero

6.3.2 Equations of Motion, (I_{xz} is typically small)

$$\begin{split} \mathbf{\mathcal{B}}\dot{v}_{a} &= \frac{1}{m} \sum_{\mathcal{B}} \mathbf{\mathcal{F}} - \mathbf{\mathcal{B}}\omega \times \mathbf{\mathcal{B}}v_{a}, \quad \mathbf{\mathcal{I}}\dot{r} = C_{\mathcal{I}\mathcal{B}\mathcal{B}}v_{a} + \mathbf{\mathcal{I}}v_{w} \\ \mathbf{\mathcal{B}}\dot{\omega} &= \mathbf{\mathcal{B}}I^{-1} \left(\sum_{\mathcal{B}} \mathbf{\mathcal{M}} - \mathbf{\mathcal{B}}\omega \times (\mathbf{\mathcal{B}}I_{\mathcal{B}}\omega) \right), \ \mathbf{\mathcal{B}}I = \begin{bmatrix} I_{xx} & 0 & I_{xz} \\ 0 & I_{yy} & 0 \\ I_{xz} & 0 & I_{zz} \end{bmatrix} \end{split}$$

Cascaded Control Loops

Steady Level Turnina

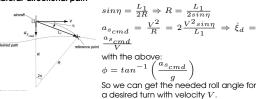
We need $\beta \dot{v}_a = \beta \dot{\omega} = 0$; $\theta = \alpha \rightarrow \gamma = 0$ and $\phi = const \neq 0$.

- The Lift L increases with $\frac{1}{\cos\phi}$ \leftarrow $(L = \frac{mg}{\cos\phi})$
- The air speed V_{min} increases with $\sqrt{1/cos\phi}$ \leftarrow

The Heading rate ξ can be found with a force balance with the centripetal force (and $\dot{\psi} \approx \dot{\xi}$). Note: $\dot{\xi} = V/R \Leftrightarrow v = r \cdot \omega$

$$F_{cent} = m \frac{V_G^2}{S} = L \cdot sin\phi \& L \cdot cos\phi = mg \Rightarrow \dot{\psi} = \dot{\xi} = \frac{g}{V} tan\phi$$

Lateral-directional path



6.4.2 TECS - Total Energy Control System

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

'Potential energy rate minus kinetic energy rate