Robotics 1
Exercise Solver 2

Martina Doku

February 13, 2024

Contents

1 DC motors

1.1 DC motor equations: electrical and mechanical

$$V_a = R_a I_a(t) + L_a \frac{dI_a(t)}{dt} + v_{emf}(t) \tag{1}$$

$$v_{emf}(t) = k_v \omega(t) \tag{2}$$

in control domain:

$$V_a = (R_a + L_a s) I_a + V_{emf} \tag{3}$$

$$V_{emf} = k_v \omega \tag{4}$$

where V_a is the voltage applied to the motor, R_a is the armature resistance, La i_s the armature inductance, i_a is the armature current, v_{emf} is the back emf, k_v is the back emf constant and ω is the angular velocity of the motor.

1.2 Reduction ratio

The reduction ratio of a the ransmission chain is the product of the reduction ratios of the single elements of the chain:

$$\eta = \prod_{i=1}^{n} \eta_i \tag{5}$$

1.2.1 Harmonic drive

The reduction ratio of a harmonic drive is given by:

$$\eta = \frac{\#theet_{flexspline}}{\#teeth_{CS} - \#teeth_{flexspline}} = \frac{\#teeth_{flexspline}}{2} \tag{6}$$

$$\#teeth_{flexspline} = \#teeth_{CS} - 2 \tag{7}$$

1.2.2 Standard gear

The reduction ratio of a standard gear is given by:

$$\eta = \frac{r1}{r2} \tag{8}$$

1.3 Optimal reduction ratio

The optimal reduction ratio is given by:

$$\eta = \sqrt{\frac{I_l}{I_m}} \tag{9}$$

where I_l is the link inertia and I_m is the motor inertia.

1.4 Motor optimal torque

Giventhe realation between accelerations:

$$\ddot{\theta}_l = \eta \ddot{\theta}_m \tag{10}$$

The motor optimal torque is given by:

$$\tau_m = I_m \ddot{\theta}_m + \frac{1}{\eta}(I_l * \ddot{\theta}_l) \tag{11}$$

where τ_m is the motor torque, I_m is the motor inertia, $\ddot{\theta}_m$ is the motor acceleration, η is the reduction ratio, I_l is the link inertia and $\ddot{\theta}_l$ is the link acceleration.
Note: if we have additional transimission elements we need to add them to the equation like:

$$\tau_m = I_m \ddot{\theta}_m + \frac{1}{\eta_1}(I_1 * \ddot{\theta}_1) + \frac{1}{\eta_2}(I_2 * \ddot{\theta}_2) \tag{12}$$

2 Encoders

2.1 Resolution relations

The resolution of a chain of transmission elements is given by:

$$\text{res}_l = \frac{\text{res}_{\text{motor}}}{\eta} \tag{13}$$

2.2 Absolute encoder

An absolute encoder is an encoder that provides the absolute position of the shaft. The resolution of an absolute encoder is given by:

$$\text{res} = \frac{2\pi}{2^{N_t}} \tag{14}$$

where N_t is the number of bits of the encoder.
Note: the resolution changes from base to link end!

$$\text{res}_{\text{base}} = \frac{\text{res}_{\text{link}}}{L} \tag{15}$$

where L is the length of the link.

2.3 Incremental encoder

An incremental encoder is an encoder that provides the relative position of the shaft. The resolution of an incremental encoder is given by:

$$\text{res} = \frac{2\pi}{N_p} \tag{16}$$

where N_p is the number of pulses per revolution of the encoder.

2.4 Multiturn encoder

A multiturn encoder is an encoder that provides the absolute position of the shaft and the number of revolutions of the shaft. The number of bits to count the turns in a multi-turn encoder is given by:

$$N_t = \log_2(N_r) \tag{17}$$

where N_r is the number of revolutions of the encoder.
The number of turns of a multiturn encoder is given by:

$$N_r = \frac{res_{max} * \eta_r}{2\pi} \tag{18}$$

where res_{max} is the maximum resolution of the encoder and η_r is the reduction ratio of the encoder.
The angular resolution of a multiturn absolute encoder is given by:

$$\text{res} = \frac{2\pi}{2^{N_t}} \tag{19}$$

where N_t is the number of bits of the encoder (only the bits for the single turn).

2.4.1 ex: from graycode to the measured angle

To convert the graycode to the measured angle we need to:

1. convert the graycode to decimal
2. multiply the decimal result for the resolution of the encoder

3 DH frames

3.1 Assign axis

- z_i along the direction of joint i-1
- x_i along the common normal between z_{i-1} and z_i
- y_i completing the right-handed frame

Also, for finding out frame origin O_i:

- If the axis of joint *i+1* and joint *i*, intersect, then the frame origin O_i is the intersection point
- If they do not intersect, the frame origin O_i is the intersection point between the axis of joint *i+1* and the common normal relative to joint *i*.

3.2 DH table

- α_i is the angle from z_i to z_{i-1} about x_i
- a_i is the distance from z_{i-1} to z_i along x_i
- d_i is the distance from x_{i-1} to x_i along z_{i-1}
- θ_i is the angle from x_i to x_{i-1} about z_{i-1}

4 Numerical Method Inverse Kinematics

4.1 Newton-Raphson

$$r_d = f(q) + J(q^k)(q - q^k) \tag{20}$$

$$q^{k+1} = q^k + J^{-1}(q^k)[r_d - f(q^k)] \tag{21}$$

Note: J^{-1} is the inverse of the Jacobian matrix. When the Jacobian is not square, the pseudo-inverse is used.

4.1.1 pseudo-inverse

$$J^+ = J^T(JJ^T)^{-1} \tag{22}$$

4.2 Gradient

$$q^{k+1} = q^k - \alpha J^T(q^k)H(q^k) \tag{23}$$

where $H(q^k)$ is the error vector and α is the learning rate. **Note**: the method does not converge when:

1. The Jacobian matrix is not full rank.
2. The error is in the null space of the Jacobian.

5 Building the Jacobian

We hae two kinds of Jacobian matrices:

1. **Analytic Jacobian**
2. **Geometric Jacobian**

Unless specified we usally refer to the analytic Jacobian.

5.1 Analytic Jacobian

The analytic Jacobian is computed by taking the **derivative of the forward kinematics**.

$$J(q) = \frac{\partial r(q)}{\partial q} \tag{24}$$

In matlab:

```
J = jacobian(r, q)
```

Note: sometimes it might be convenient to compute the Jacobian with respect to a frame different from the base frame (see section 5.3).

5.2 Geometric Jacobian

The geometric Jacobian is composed of two parts;

$$J(q) = \begin{bmatrix} J_L(q) \\ J_A(q) \end{bmatrix} \tag{25}$$

1. **Linear part** (same as the analytic Jacobian)
2. **Angular part**

5.2.1 Linear Jacobian

To build the linear Jacobian we have options:

1. **Analytic approach**: we compute the derivative of the forward kinematics.

   $$J_L(q) = \frac{\partial r(q)}{\partial q} \tag{26}$$

   Note: instead of the forward kinematics we could also use the position of the end-effector with respect to the base frame (computed from the DH as: ${}^0p_n = {}^0T_n[1:3,4]$).

2. **Geometric approach**: we compute the cross product of the axis of rotation and the vector from the origin of the base frame to the origin of the end-effector frame.

   $$J_L(q) = \begin{bmatrix} z_0 \times {}^0p_E & z_1 \times {}^1p_E & \dots & z_{n-1} \times {}^Ep_E \end{bmatrix} \tag{27}$$

   From the DH parameters we can:

- obtain the homogeneous transformation matrix ${}^0T_n$
- extract the position of the end-effector frame with respect to the base frame as ${}^0p_E = {}^0T_n[1:3,4]$.
- We can then computee ${}^ip_E$ as ${}^ip_E = {}^0p_E - {}^0p_i$ and ${}^0p_i$ as ${}^0p_i = {}^0T_i[1:3,4]$. z_0 is the defined as $[0,0,1]^T$.

Note: sometimes it might be convenient to compute the Jacobian with respect to a frame different from the base frame (see section 5.3).

5.2.2 Angular Jacobian

To angular Jacobian is defined as follows:

$$J_A(q) = \begin{bmatrix} {}^0z_0 & {}^0R_1 \cdot {}^1z_1 & \dots & {}^0R_{n-1} \cdot {}^{n-1}z_{n-1} \end{bmatrix} \tag{28}$$

where ${}^iz_i$ is the defined as $[0,0,1]^T$.
How to compute ${}^0R_i$?
We can obtain the rotation matrix ${}^0R_i$ by using the DH parameters, computing the homogeneous transformation matrix ${}^0T_i$ and extracting the rotation matrix from it.

5.3 Expressing Jacobian in a Rotated Frame

To obtain the rotated jacobian we do:

$$J_{rot}(q) = {}^0R_{frame} J(q) \tag{29}$$

When do we need to rotate the Jacobian? -no exact rule
Intuitivelywe can decide to rotate the matrix when we have a complex jacobian (ex: a matrix with entries in the form $sin(q_i) * (...)$. We can also decide to rotate the matrix if we figure that the determinant of the matrix is too complex to compute.
What R should I use to rotate the Jacobian?
Ususally ${}^0R_1$ that can be computed from the DH parameters.

6 Jacobian Analysis and Mobility Analysis

6.1 Jacobian Analysis

The first step is to compute the Jacobian (analytic if not differently specified) of the robot if not given, as in section 5.
Once we have the jacobian we can compute:

- **Singularities**: the singularities are the values of q that make the determinant of the Jacobian equal to zero.
  Note: if the Jacobian is not square we have two methods to compute the determinant:
  - find the determinant of $J^T(q)J(q)$
  - find the determinant of all the square submatrix of the Jacobian
- **Range**: the range of the Jacobian is the set of basis vector of the jacobian matrix.
- **Nullspace**: the nullspace of the Jacobian is the set of all the vector that multiplied by the Jacobian produce zero. It can be computed by the matlab command: null(J)
- **Complementary space**: the complementary space of the Jacobian is the set of all the vector that multiplied by the transpose of the Jacobian produce zero. It can be computed by the matlab command: null(J')
- **Rank**: the rank of the Jacobian is the number of linearly independent column of the Jacobian.

6.2 Mobility Analysis

- To find **singularities** we need to find the values of q that make the determinant of the Jacobian equal to zero.
- the R(J) -range- is the subspace of **all velocities that can be instantaneously achieved by the robot** varying the joint velocities.
- if $N(J) \neq 0$ there are **non zero velocites that produce zero end-effector velocity**.
- if $\rho(J) <$ the robot is not fully mobile.
- To find an **acceleration $\ddot{q}$ that makes $\ddot{r} = 0$ when $\dot{q} = 0$** means to find the null space of the Jacobian (try the siingular one if the standard one has nullspace 0) since $\ddot{r} = J\ddot{q} + \dot{J}\dot{q}$ and we know that $\dot{q} = 0$ so $\ddot{r} = J\ddot{q}$.
- the **direction(s) along which no task velocity can be realized** is the complementary space of the range of the Jacobian, which is the null space of the transpose of the Jacobian.

7 Differential Kinematics

7.1 Differential relations

$$\dot{p} = J(q)\dot{q} \tag{30}$$
$$\dot{p} = v \tag{31}$$
$$\omega = T(\phi)\dot{\phi} \tag{32}$$
$$\dot{R} = S(\omega)R \tag{33}$$
$$\ddot{p} = J(q)\ddot{q} + \dot{J}(q)\dot{q} \tag{34}$$
$$\tag{35}$$

7.2 Derivative of a rotation matrix

$$\frac{d}{dt}R = \dot{R} = S(\omega)R = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} R \tag{36}$$

Note: since $R = [xyz]$, where n is the normal vector, s is the sliding vector and a is the approach vector, we can also deduce that:

$$^0\dot{x_i} = S(\omega)^0x_i = \omega \times^0 x_i \tag{37}$$

More in general for axis-angle representation:

Also, $\theta = |\omega| * T$

$$R(r,\theta) \to \dot{R}(r,\theta)R^T(r,\theta) = S(\omega) \tag{38}$$
$$\omega = \dot{\theta}r \tag{39}$$

7.3 Derivative of a RPY angles

given $R_{RPY}(\alpha_x, \beta_y, \gamma_z) = R_{ZY'x''}(\gamma_z, \beta_y, \alpha_x)$ we have:

$$w = T_{RPY}(\beta,\gamma) \begin{bmatrix} \dot{\alpha_x} \\ \dot{\beta_y} \\ \dot{\gamma_z} \end{bmatrix} \tag{40}$$

where

$$T_{RPY}(\beta,\gamma) = \begin{bmatrix} cos\beta cos\gamma & -sin\gamma & 0 \\ cos\beta sin\gamma & cos\gamma & 0 \\ -sin\beta & 0 & 1 \end{bmatrix} \tag{41}$$

The first column of T_{RPY} is the first column of $R_Z(\gamma)R'_Y(\beta)$ and the second column of T_{RPY} is the second column of $R_Z(\gamma)$.
Note: we have det $T_{RPY} = 0$.

7.4 ex: given an euler rotation sequence over angles ϕ find the mapping with angular velocity

1. We have to use the relation:

$$S(\omega) = \dot{R}R^T \tag{42}$$

2. We compute R as the product of the rotation matrices of the euler sequence.

3. We extract the angular velocity from the Skew symmetric matrix.

$$S(\omega) = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \tag{43}$$

4. We now have the angular velocity, but we have to put it in the form of:

$$\omega = T(\phi)\dot{\phi} \tag{44}$$

8 Inverse Differential Kinematics

8.1 Incremental Solution

$$\Delta r = J(q)\Delta q \tag{45}$$

$$r \to r + \Delta r \tag{46}$$

$$q = f_r^{-1}(r + \Delta r) \tag{47}$$

$$q = q + \Delta q = q + J^{-1}(q)\Delta r \tag{48}$$

8.2 Damped Least Squares

$$\Delta q = J^T(JJ^T + \lambda I)^{-1}v \tag{49}$$

using pseudo-inverse:

$$\Delta q = J^+(q)v \tag{50}$$

If the Robot needs to produce a Force/Torque at its EE, J is positive.
If the Robot needs to resist or balance an external Force/Torque applied to its EE, J is negative.

8.3 Force Torque relations

$$\tau = J^T(q)F \tag{51}$$

Given a force F we can compute the torques τ that the joints need to apply to the end-effector by:

1. computing the Jacobian matrix $J(q)$
2. computing the transpose of the Jacobian matrix $J^T(q)$
3. multiplying the transpose of the Jacobian matrix by the force F
4. the result is the vector of torques τ

velocity manipulability ellipsoid:

$$\dot{q}^T\dot{q} = 1 \rightarrow v^T J^{\#} J^{\#T} v = 1 \tag{52}$$

torque manipulability ellipsoid:

$$\tau^T\tau = 1 \rightarrow \tau^T J^T J\tau = 1 \tag{53}$$

9 Joint Trajectory Planning

When we are required to plan a trajectory we need to identify the type of trajectory we need to plan. List of things to check:

- **continuity of acceleration**: if we need to plan a trajectory that is continuous in acceleration we can exclude the bang-to-bang trajectory and the bang-coast-bang (trapezoidal) trajectory and opt for a smooth trajectory.
- **smoothness**: if we need to plan a trajectory that is smooth we can exclude the bang-to-bang trajectory and the bang-coast-bang (trapezoidal) trajectory.
- **minimum time**: if we need to plan a trajectory that is minimum time and none of the previous conditions are required we can opt for a not smooth trajectory.

9.1 Smooth Trajectories

There are mainly two types of smooth trajectories, quintic and cubic.
How do we choose between the two?

- We choose the quintic trajectory when we need to impose limits on both **velocity** and an **acceleration**.
- We choose the cubic trajectory when we need to impose limits only on the **velocity**.

9.1.1 Cubic Polynomial

The cubic polynomial is defined as:

$$q(\tau) = a\tau^3 + b\tau^2 + c\tau + d \tag{54}$$

We further define the **doubly normalized polyinomial** as:

$$q_N(\tau) = q_0 + \Delta(q)(a\tau^3 + b\tau^2 + c\tau + d) \tag{55}$$

Special case if both initial and final velocity are zero (rest-to-rest) we have:

$$q(\tau) = q_0 + \Delta(q)(-2\tau^3 + 3\tau^2) \tag{56}$$

9.1.2 Quintic Polynomial

The quintic polynomial is defined as:

$$q(\tau) = a\tau^5 + b\tau^4 + c\tau^3 + d\tau^2 + e\tau + f \tag{57}$$

We further define the **doubly normalized polyinomial** as:

$$q_N(\tau) = q_0 + \Delta(q)(a\tau^5 + b\tau^4 + c\tau^3 + d\tau^2 + e\tau + f) \tag{58}$$

Special case if $v_{in} = v_{fin} = 0$ and $a_{in} = a_{fin} = 0$ (rest-to-rest) we have:

$$q(\tau) = q0 + \Delta(q)(6\tau^5 - 15\tau^4 + 10\tau^3) \tag{59}$$

9.1.3 Finding the coefficients

To find the coefficients of the polynomial we need to impose the conditions in the following way:
algorithm for **cubic spline**:

1. identify $q_N(\tau) = q0 + \Delta(q)(a\tau^3 + b\tau^2 + c\tau + d)$
2. compute the derivative of $q_N(\tau)$
3. find the coefficients a, b, c, d by solving the system of equations:
   - $q_N(0) = q_{in}$
   - $q_N(1) = q_f$
   - $q'_N(0) = v_{fin}$
   - $q'_N(1) = v_f$
4. we obtain:
5. d = 0
6. $c = \frac{v_0 * T}{\Delta(q)}$
7. a+b+c = 1
8. $3a + 2b + c = \frac{v_f * T}{\Delta(q)}$

algorithm for **quintic spline**:

1. identify $q_N(\tau) = q0 + \Delta(q)(a\tau^5 + b\tau^4 + c\tau^3 + d\tau^2 + e\tau + f)$
2. impose the constraints of the initial and final position, velocity and acceleration
3. find the coefficients a, b, c, d, e, f by solving the system of equations
4. $q(0) = q_0$ and $q(1) = q_1$
5. $q'(0) = v_0$ and $q'(1) = v_1$
6. $q''(0) = a_0 T$ and $q''(1) = a_1 T$

9.1.4 ex: finding the optimal time T given max velocity and max acceleration

To find the optimal time given max velocity and max acceleration we need to:

1. Compute the optimal tau that is the the tau where the acceleration is at it's maximum that can be computed as:

$$\dddot{q}(\tau) = 0 \tag{60}$$

2. Substitute the optimal tau in the velocity equation and find the optimal time.

3. Substitute the optimal time in the acceleration equation and find the optimal acceleration.

4. Select the maximum value between those times.

9.2 Non Smooth Trajectories

If not specified we can identify the kind of non smooth trajectory by checking if there is a coast phase. We check if:

$$L \geq \frac{v_{max}^2}{a_{max}} \tag{61}$$

then we have a coast phase.

9.2.1 Bang-Bang Trajectory

The bang-bang trajectory is defined as:

$$q(t) = \begin{cases} \frac{a_{max}t^2}{2} & t \in [0, t_s) \\ -\frac{a_{max}(T-t_s)^2}{2} + v_{max}T - \frac{v_{max}^2}{a_{max}} & t \in [T-t_s, T] \end{cases} \tag{62}$$

$$\dot{q}(t) = \begin{cases} a_{max}t & t \in [0, t_s) \\ -a_{max}(T-t_s) + v_{max} & t \in [T-t_s, T] \end{cases} \tag{63}$$

$$\ddot{q}(t) = \begin{cases} a_{max} & t \in [0, t_s) \\ -a_{max} & t \in [T-t_s, T] \end{cases} \tag{64}$$

9.2.2 ex: finding minimum time

To find the minimum time given max velocity and max acceleration we need to find the max of all th possible time computation for the bang-bang trajectory.

1. compute the total distance or length (L or $\Delta(q)$) fro each joint

2. Compute the possible values of time (for each joint) as: $T_{min} = \sqrt{\frac{4L}{a_{max}}}$

3. note: we can do that if we have initial and final velocity equal to zero

2. **Formulas:**

- Peak Velocity (V_1): $V_1 = \frac{2\Delta q_1}{T}$
- Constant Acceleration (A_1): $A_1 = \frac{V_1^2}{\Delta q_1}$

Simply if we want to slow down the faster
joint to align with the slower joint.
We slow it down by these 2 quantities.

4. (we obtain that from $v^2 = v_0^2 + 2a\Delta(q)$)
5. find the maximum value of the possible values of time

9.2.3 Bang-Coast-Bang Trajectory

The bang-coast-bang trajectory is defined as:

$$q(t) = \begin{cases} \frac{a_{max}t^2}{2} & t \in [0, t_s) \\ v_{max}t - \frac{v_{max}^2}{2a_{max}} & t \in [t_s, T - t_s) \\ -\frac{a_{max}(T-t_s)^2}{2} + v_{max}T - \frac{v_{max}^2}{a_{max}} & t \in [T - t_s, T] \end{cases} \tag{65}$$

$$\dot{q}(t) = \begin{cases} a_{max}t & t \in [0, t_s) \\ v_{max} & t \in [t_s, T - t_s) \\ -a_{max}(T - t_s) + v_{max} & t \in [T - t_s, T] \end{cases} \tag{66}$$

$$\ddot{q}(t) = \begin{cases} a_{max} & t \in [0, t_s) \\ 0 & t \in [t_s, T - t_s) \\ -a_{max} & t \in [T - t_s, T] \end{cases} \tag{67}$$

9.2.4 ex: finding t_s and t_c

To find the values of t_s and t_c we need to impose the following constraints:

$$t_s = \frac{v_{max}}{a_{max}} \tag{68}$$

$$t_c = \frac{La_{max} + v_{max}^2}{a_{max}v_{max}} - 2 * t_s \tag{69}$$

The final time is given by $t_f = t_c + 2t_s$

9.2.5 ex: finding minimum time

To find the minimum time given max velocity and max acceleration we need to find the max of all th possible time computation for the bang-coast-bang trajectory.

1. compute the total distance or length (L or $\Delta(q)$) fro each joint
2. Compute the possible values of time (for each joint) as:
   - $T_{min} = \frac{La_{max}+v_{max}^2}{a_{max}v_{max}}$
   - $T_{min} = \frac{L}{v_{max}}$
3. find the maximum value of the possible values of time

$$q = q_i + \dot{q}_\iota \cdot t + \frac{a \cdot t^2}{2}$$

check exam 22.09.09 for usage of this formula

10 Cartesian Trajectory Planning

10.1 Relation between path and s

The relation between the path and the parameter s is given by the following equation:

$$\dot{p}(t) = \frac{dp(s)}{ds}\dot{s} \tag{70}$$

$$\ddot{p}(t) = \frac{d^2p(s)}{ds^2}\dot{s}^2 + \frac{dp(s)}{ds}\ddot{s} \tag{71}$$

10.2 Cartesian Trajectories

we have different classes of Cartesian trajectories:

- **point-to-point**: we need to move the end-effector from a point to another
- **line**: we need to move the end-effector along a line
- **circle**: we need to move the end-effector along a circle
- **ellipse**: we need to move the end-effector along an ellipse
- **helix**: we need to move the end-effector along a helix

10.2.1 Point-to-point

The parametrized point-to-point trajectory is defined as:

$$p(s) = p_0 + (p_f - p_0)s \tag{72}$$

whith $s \in [0, 1]$.
Note: in this case we are not considering the orientation of the end-effector, if we want to arrive at the final point with a specific orientation we need to use the **orientation** aswell as the position.
Note: If we want with a specfic velocity we need to ensure that the orientation of the velocity is parallel to the line connecting the initial and final point, if not we cannot use the straight line trajectory.

10.2.2 Line

The parametrized line trajectory is defined as:

$$p(s) = p_0 + (p_f - p_0)s \tag{73}$$

whith $s \in [0, 1]$.

10.2.3 Circle

The parametrized circle trajectory is defined as:

$$p(s) = R \begin{bmatrix} cos(\frac{s}{R+\phi}) \\ sin(\frac{s}{R+\phi}) \end{bmatrix} \tag{74}$$

whith $s \in [0, 2\pi(R+\phi)]$.

10.2.4 Ellipse

The parametrized ellipse trajectory is defined as:

$$p(s) = \begin{bmatrix} a\cos(s) \\ b\sin(s) \end{bmatrix} \tag{75}$$

where a and b are the semi-axes.
If there is a phase shift we have:

$$p(s) = \begin{bmatrix} a\cos(s+\phi) \\ b\sin(s+\phi) \end{bmatrix} \tag{76}$$

10.3 Find a timing law s(t)

compute s depending on the type of law:

1. if the law is a trapezoidal $s(t) = \begin{cases} \frac{a_{max}t^2}{2} & t \in [0, t_s] \\ v_{max}t & t \in [t_s, t_f - t_s] \\ -\frac{a_{max}(t_f - t_s)^2}{2} + v_{max}t_f - \frac{v_m^2 ax}{a_m ax} & t \in [t_f - t_s, t_f] \end{cases}$
2. if the law is a quintic spline $s(\tau) = a\tau^5 + b\tau^4 + c\tau^3 + d\tau^2 + e\tau + f$
3. if the law is a rest to rest quintic spline $s(\tau) = 6\tau^5 - 15\tau^4 + 10\tau^3$
4. if the law is a cubic spline $s(\tau) = a\tau^3 + b\tau^2 + c\tau + d$
5. if the law is a rest to rest cubic spline $s(\tau) = -2\tau^3 + 3\tau^2$
6. remember that s is a function of t so we need to substitute τ with $\frac{t}{T}$

10.3.1 ex: find minimum time given max acceleration

To find the minimum time we have to:

1. compute the equation of the path in function of s
2. compute the equation of s in function of time
3. compute $\ddot{p}(t)$ as $\ddot{p}(t) = \frac{d^2p(s)}{ds^2}\dot{s}^2 + \frac{dp(s)}{ds}\ddot{s}$
4. impose the constraint $\ddot{p}(t) \leq a_{max}$
5. solve the inequality for T

10.4 Orientation trajectory (rotation matrices)

When asked to find the timing law for the orientation we can adobt mainly two strategies:

- **Euler angles**: we can find the timing law for each of the three angles
- **Axis-angle**: we can find the timing law for the angle θ and the axis r
- **Note**: if the desired final orientation velocity is not zero, the axis angle method cannot be used, you have to find an euler representation xyz

10.4.1 Euler angles

Given two rotation matrices R_1 and R_2, find the timing law of the euler angles.

1. find the euler angles in the initial and final position
2. wriyte the timing law for an angle theta
3. obtain the coefficient of theta (symbolic) and solve for each of the three angles
4. Note: if you have more than one trait the timing law of theta has to be defined for each trait

10.4.2 Axis-angle

Given starting and ending point A and B, and start and end rotation matrices R_A and R_B find the max velocity, max acceleration , max angular velocity and max angular acceleration.

1. write the position timing law as: $p(s) = p_A + s(p_B - p_A)$
2. compute the velocity as $\dot{p}(s) = (p_b - p_A)\dot{s}$ (s symbolic)
3. compute the acceleration as $\ddot{p}(s) = (p_b - p_A)\ddot{s}$ (s symbolic)
4. compute the rotation matrix $R_{AB} = R_A^T R_B$
5. find the axis-angle representation of R_{AB} as $[r, \theta]$
6. write the angular timing law as $\theta(s) = \theta * s$
7. compute the angular velocity as $\dot{\theta}(s) = \theta * \dot{s}r$
8. compute the angular acceleration as $\ddot{\theta}(s) = \theta * \ddot{s}r$
9. substitute s with the appropriate timing law and solve for the maxi-mum velocity, acceleration, angular velocity and angular acceleration

11 Kinematic Control

11.1 Joint space control

equation for a single joint:

$$\tau = M(q)\ddot{q} + mg_0 d\sin(q_1) \tag{77}$$

where $M(q) = I_l + I_m + m_1 d^2$ and g_0 is the gravity constant.

11.2 Error dynamics

$$\dot{e} = \dot{q}_d - \dot{q} = \dot{q}_d - (\dot{P}_d + K(p_d - p)) = -Ke \tag{78}$$

Note: from this equation we can obtian the following if we have a regulation problem (i.e. we want to reach a desired position):

$$\dot{q} = k(q_d - q) \tag{79}$$

error in task space

$$\dot{e_p} = \dot{p}_d - \dot{p} = -J(q)J(q)^{-1}(\dot{P}_d + K(p_d - p)) = -K_p e_p \tag{80}$$

where K_p is the proportional gain that is a diagonal matrix with the gains on the diagonal.

$$K_p = \begin{bmatrix} r & 0 & 0 \\ 0 & r & 0 \\ 0 & 0 & r \end{bmatrix} \tag{81}$$

Kinematic control law:

$$\dot{q} = J(q)^{-1}(\dot{p}_d + K_P(p_d - p(q))) \tag{82}$$

$$\ddot{q} = J(q)^{-1}(\ddot{p}_d + K_P(\dot{p}_d - \dot{p}(q)) + K_D(p_d - p(q)) - \dot{J}\dot{q}) \tag{83}$$

the equation of $P_d(t)$ is given by:

$$p_d(t) = p_0 + \Delta p(t) \tag{84}$$

where $\Delta p(t)$ is the trajectory in task space. ex: for a given costant velocity v_d:

$$p_d(t) = p_0 + v_d t(\Delta p(t)) \tag{85}$$

11.3 Tracking error

$\tau_p = 1/K_p$ is the time constant of the exponential decrease of tracking error. The exponential decrease of the tracking error is given by:

$$e_p(t) = e_p(0) * exp^{-tK_p} \tag{86}$$

condition on the control gain:
if we want the normal component of the tracking error to be different from the tangential component we can impose the following condition on the control gain:

$$K = \begin{bmatrix} K_t & 0 \\ 0 & K_n \end{bmatrix} \tag{87}$$

11.4 Basic Formula

$$\dot{e} = \dot{p}_d - \dot{p} \rightarrow \dot{p} = \dot{p}_d - \dot{e} \tag{88}$$

$$\dot{e} = -K_p e \tag{89}$$

$$e = p_d - p \rightarrow \dot{p} = \dot{p}_d + K(p_d - p) \tag{90}$$

11.5 Rotated error

When we express the standar control law we have:

$$\dot{q} = J(q)^{-1}(\dot{p}_d - \dot{e}_{base}) \tag{91}$$

We can rewrite the control law by substituting the $\dot{e}_{base}$ with $({}^0R_i)\dot{e}_{task}$:

$$\dot{q} = J(q)^{-1}(\dot{p}_d - ({}^0R_i)\dot{e}_{task}) \tag{92}$$

and $\dot{e}_{task}$ as:

$$\dot{e}_{task} = -K_{task}e_{task} \tag{93}$$

$$e_{task} = ({}^0R_i)^T e_{base} \tag{94}$$

Finally we can express $\dot{e}_{base}$ (that is the one we need to use in the control law) as:

$$\dot{e}_{base} = -({}^0R_i)K_{task}({}^0R_i)^T e_{base} \tag{95}$$

so the control law becomes:

$$\dot{q} = J(q)^{-1}(\dot{p}_d + ({}^0R_i)K_{task}({}^0R_i)^T e_{base}) \tag{96}$$

11.6 derivative of rotated error

The derivative of the rotated error is given by:

$$\dot{e} = R\dot{e} + \dot{R}e = R(\dot{e} + S(\omega)e) = R(\dot{p}_d - J(q)\dot{q} + S(\omega)e) \tag{97}$$

11.7 ex: how to correct an error in the trajectory

To correct an error in the trajectory we can add a proportional-derivative action along the positional trajectory i.e:

$$\ddot{p}_d = \ddot{p}_d + K_P(p_d - p) + K_D(\dot{p}_d - \dot{p}) \tag{98}$$

11.7.1 ex: design a velocity control law that lets e converge with exponential decaying rate r

To design a velocity control law that lets e converge with exponential decaying rate r we can use the following equation: $\dot{q} = J(q)^{-1}(\dot{p}_d + K_P(p_d - p(q)))$ and impose the condition on the control gain $K_P = rI$.
This guarantees exponential convergence of the tracking error with rate r.

12 Uniform Time Scaling

We can apply uniform time scaling when we have a trajectory that is not feasible in terms of velocity and acceleration.
To apply it we do the following:

- compute the maximum velocity and acceleration (peak)
- compute the **time scaling factor** as:

$$K_{vel} = \max \frac{|v_{peak}|}{v_{max}}, 1 \tag{99}$$

$$K_{acc} = \max \frac{|a_{peak}|}{a_{max}}, 1 \tag{100}$$

- find the maximum of the two factors

$$K = \max K_{vel}, K_{acc} \tag{101}$$

- we have **two cases**:
  if we **overcome the velocity or acc limit** we scale as:
  - scale the time as:

$$T_{new} = K * T_{old} \tag{102}$$

  - scale the velocity and acceleration as:

$$v_{new} = \frac{v_{old}}{K} \tag{103}$$

$$a_{new} = \frac{a_{old}}{K^2} \tag{104}$$

  - scale the intervals as:

$$t_{scaled} = t_1 + K(t_i - t_1) \tag{105}$$

    for i in [2,n] while t_1 stays the same.

- if we **don't reach the velocity or acc limit** we scale as:
  - scale the time as:

$$T_{new} = \frac{T_{old}}{K} \tag{106}$$

– scale the velocity and acceleration as:

$$v_{new} = v_{old} * K \tag{107}$$

$$a_{new} = a_{old} * K^2 \tag{108}$$

– scale the intervals as:

$$t_{scaled} = t_1 + \frac{t_i - t_1}{k} \tag{109}$$

for i in [2,n] while t_1 stays the same.

13 APPENDIX

13.1 Trigonmetry

$$\sin^2(\theta) + \cos^2(\theta) = 1 \tag{110}$$

$$\tan(\theta) = \frac{\sin(\theta)}{\cos(\theta)} \tag{111}$$

$$\sin(2\theta) = 2\sin(\theta)\cos(\theta) \tag{112}$$

$$\cos(2\theta) = \cos^2(\theta) - \sin^2(\theta) \tag{113}$$

$$\cos(2\theta) = 2\cos^2(\theta) - 1 \tag{114}$$

$$\cos(2\theta) = 1 - 2\sin^2(\theta) \tag{115}$$

$$\sin(\theta + \phi) = \sin(\theta)\cos(\phi) + \cos(\theta)\sin(\phi) \tag{116}$$

$$\cos(\theta + \phi) = \cos(\theta)\cos(\phi) - \sin(\theta)\sin(\phi) \tag{117}$$

$$\sin(\theta - \phi) = \sin(\theta)\cos(\phi) - \cos(\theta)\sin(\phi) \tag{118}$$

$$\cos(\theta - \phi) = \cos(\theta)\cos(\phi) + \sin(\theta)\sin(\phi) \tag{119}$$

$$\sin(\theta)\sin(\phi) = \frac{1}{2}(\cos(\theta - \phi) - \cos(\theta + \phi)) \tag{120}$$

13.2 Trascendental functions

given a trascendental equation with two angles θ_1 and θ_2 we can use the following indentity to find values of θ_2 that make θ_1 have real solutions:

$$a^2 + b^2 \geq c^2 \tag{121}$$

with a, b, c being the coefficients of the trascendental equation.

$$a\sin(\theta_1) + b\cos(\theta_1) = c \tag{122}$$

13.3 To remember

- when you have time continuous scalar values you can integrate them, ex for v_{max}:

$$v_{max} = \int_0^{t_f} \ddot{q} dt \tag{123}$$

- to see if the ful trajectory is in the workspace we can construct a bounding box for the workspace and check if the trajectory is inside the bounding box.