1. Motivation 4

2. Forward Kinematics 4

Spatial descriptions 4

Pose of an object in space 4

Configuration Space 5

Manipulator kinematics 5

Denavit-Hartenberg Rules 6

Yaw-Pitch-Roll representation for orientation 8

3. Inverse Kinematics 9

Workspace 9

Multiplicity of solution 9

Analytic solutions 9

Numeric solutions 11

Numeric computation of the Jacobian 12

propagation of force and torques 12

4. Manipulator dynamics 13

General formulas 13

Euler-Lagrange Equations 13

Force and torque relation 14

Newton-Euler Equations 14

State-space equation 16

Cartesian state-space equation 16

Forces and torques for static manipulators 16

5. Jacobians: velocities and static forces 18

Compute the Jacobian 18

Characteristics 18

Velocity propagation from link to link 19

Velocities 19

Jacobian for approximating very small movements 19

Changing a Jacobian’s frame of reference 19

6. Trajectory Generation 20

Path Generation 20

7. Linear control of manipulators 21

Second-order linear systems 21

Pole-Zero Plot 22

Laplace Transformation 22

Control of second-order systems 22

Control-law partitioning 22

Multi-dimensional systems 23

Trajectory-following control 23

Disturbance Rejection 23

Other systems 24

Torque control 24

Effective inertia 24

8. Nonlinear control of manipulators 26

Control problem for manipulators 26

Cartesian-based control systems 26

9. Force control of manipulators 27

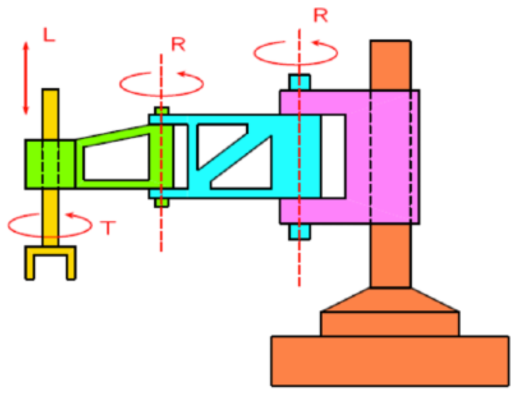
Force Control of a mass-spring system 27

Hybrid position/force Controller 27

10. Task requirements 28

1. Motivation

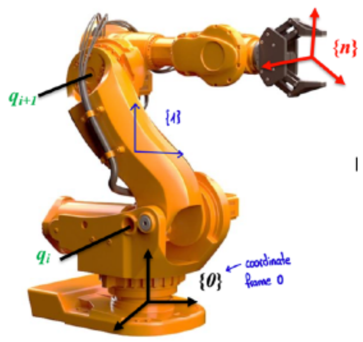
- Current Trend: Use Deep Learning to avoid explicit programming ⚡ limits, not enough data for robotics
- Not only need to know what is in the area around the robot but also
  - how big is the confidence in the correctness of the observation
  - how much of the object was visible
  - how certain is the system to see a specific object
  - where is it relative to the robot
  - …
- Complex physical models don't have to run several times
- Origin: from the word „robota" (work)
- Definition of the Robot Institute of America (1980): A robot is a reprogrammable multifunctional manipulator designed to move material, parts, tools or specialised devices through variable programmed motions for the performance of a variety of tasks. (does not need to be human like)
-



.SCARA = selective compliance assembly robot arm

2. Forward Kinematics

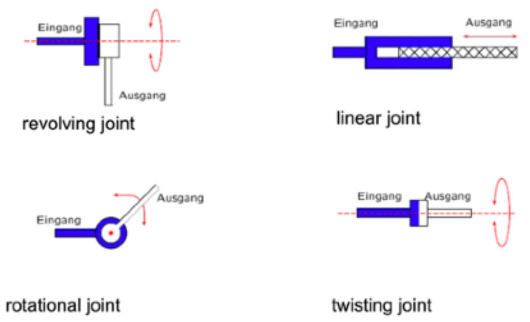
Find the end effector position & orientation Y = (x, y, z,α, β, ɣ) given the joint variables q



SPATIAL DESCRIPTIONS

- Coordinate system/frame is attached to every rigid object —> {n}
- Manipulators: rigid links which are connected by joints
- **Kinematics**: Science of motion, relation between joints (q_i) and the pose (position/orientation) of some point
- **Primary Workspace (reachable) WS_1:**
  - Positions that can be reached with at least on orientation
  - Each point can be reached, orientation doesn't matter
  - Out of WS_1 there is no solution to the problem —> for all points in WS_1, there is at least one solution
- **Secondary Workspace (dexterous) WS_2:**
  - Positions can be reached with any orientation
  - For all points in WS_2, there is at least one solution for every orientation
- Relation: $WS_2 \subseteq WS_1$
- Degrees of Freedom: number of independent motion parameters of a body in space

Type of joints



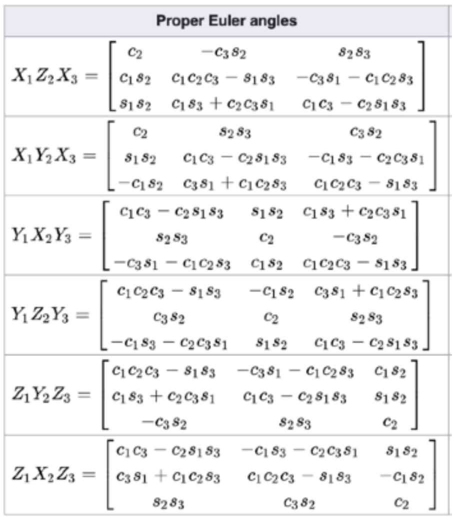
POSE OF AN OBJECT IN SPACE

- q = (position, orientation) = (x, y ,z, α, β, ɣ) —> parametrisation by 3x3 rotation matrix $R = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix}$
  - $r_{1i}^2 + {r_{2i}}^2 + {r_{3i}}^2 = 1$ for all i
  - $r_{1i}r_{1j} + r_{2i}r_{2j} + r_{3i}r_{3j} = 0$ for all $i \neq j$,
  - $\det(R) = +1$ —> length stays the same
- Not all elements are independent, representation by fewer elements —> three possibilities:

Euler angles

- Correspond to three successive rotations around z-axis, y-axis and lastly z-axis:

$$R = R_{z,\phi} R_{y,\theta} R_{z,\psi}$$
$$= \begin{bmatrix} c_\phi & -s_\phi & 0 \\ s_\phi & c_\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_\theta & 0 & s_\theta \\ 0 & 1 & 0 \\ -s_\theta & 0 & c_\theta \end{bmatrix} \begin{bmatrix} c_\psi & -s_\psi & 0 \\ s_\psi & c_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$
$$= \begin{bmatrix} c_\phi c_\theta c_\psi - s_\phi s_\psi & -c_\phi c_\theta s_\psi - s_\phi c_\psi & c_\phi s_\theta \\ s_\phi c_\theta c_\psi + c_\phi s_\psi & -s_\phi c_\theta s_\psi + c_\phi c_\psi & s_\phi s_\theta \\ -s_\theta c_\psi & s_\theta s_\psi & c_\theta \end{bmatrix}.$$



Proper Euler angles
$X_1 Z_2 X_3 = \begin{bmatrix} c_2 & -c_3 s_2 & s_2 s_3 \\ c_1 s_2 & c_1 c_2 c_3 - s_1 s_3 & -c_3 s_1 - c_1 c_2 s_3 \\ s_1 s_2 & c_1 s_3 + c_2 c_3 s_1 & c_1 c_3 - c_2 s_1 s_3 \end{bmatrix}$
$X_1 Y_2 X_3 = \begin{bmatrix} c_2 & s_2 s_3 & c_3 s_2 \\ s_1 s_2 & c_1 c_3 - c_2 s_1 s_3 & -c_1 s_3 - c_2 c_3 s_1 \\ -c_1 s_2 & c_3 s_1 + c_1 c_2 s_3 & c_1 c_2 c_3 - s_1 s_3 \end{bmatrix}$
$Y_1 X_2 Y_3 = \begin{bmatrix} c_1 c_3 - c_2 s_1 s_3 & s_1 s_2 & c_1 s_3 + c_2 c_3 s_1 \\ s_2 s_3 & c_2 & -c_3 s_2 \\ -c_3 s_1 - c_1 c_2 s_3 & c_1 s_2 & c_1 c_2 c_3 - s_1 s_3 \end{bmatrix}$
$Y_1 Z_2 Y_3 = \begin{bmatrix} c_1 c_2 c_3 - s_1 s_3 & -c_1 s_2 & c_3 s_1 + c_1 c_2 s_3 \\ c_3 s_2 & c_2 & s_2 s_3 \\ -c_1 s_3 - c_2 c_3 s_1 & s_1 s_2 & c_1 c_3 - c_2 s_1 s_3 \end{bmatrix}$
$Z_1 Y_2 Z_3 = \begin{bmatrix} c_1 c_2 c_3 - s_1 s_3 & -c_3 s_1 - c_1 c_2 s_3 & c_1 s_2 \\ c_1 s_3 + c_2 c_3 s_1 & c_1 c_3 - c_2 s_1 s_3 & s_1 s_2 \\ -c_3 s_2 & s_2 s_3 & c_2 \end{bmatrix}$
$Z_1 X_2 Z_3 = \begin{bmatrix} c_1 c_3 - c_2 s_1 s_3 & -c_1 s_3 - c_2 c_3 s_1 & s_1 s_2 \\ c_3 s_1 + c_1 c_2 s_3 & c_1 c_2 c_3 - s_1 s_3 & -c_1 s_2 \\ s_2 s_3 & c_3 s_2 & c_2 \end{bmatrix}$

- no global parametrization because there are some singularities —> define functions like atan and use other Euler angles
- Each matrix has the singularity in a different place —> choose right presentation

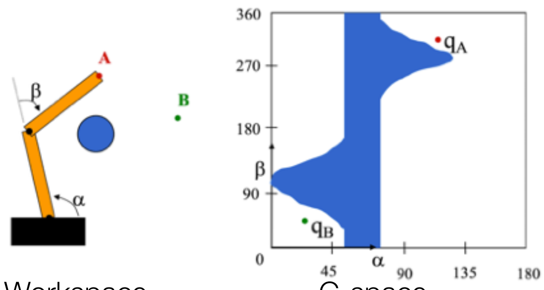
<u>Axis-angle representation</u>
- efficient algorithm for rotating a vector in space, given an axis and angle of rotation
- Coordinate with vector X, where k is the unit vector representing the axis of rotation, x is the result of rotating X by an angle theta about k:
  - $\vec{x} = R\vec{X}$
  - Rodrigues formula: $R = \left(I + (1 - \cos\theta)K^2 + \sin\theta K\right)$ with $K = \begin{bmatrix} 0 & -k_3 & k_2 \\ k_3 & 0 & -k_1 \\ -k_2 & k_1 & 0 \end{bmatrix}$
- Inverse: Find back k and theta (Apple Rodrigues formula for both sides)
  - $\vec{k} = \dfrac{1}{2\sin\theta}\operatorname{vect}(K)$ and theta by solving $2\sin\theta = \left\| \operatorname{vect}\left(R - R^T\right) \right\|$

<u>Unit quaternion</u>
- The axis-angle parameterization described above parameterizes a rotation matrix by three parameters. Quaternions, which are closely related to the axis-angle parameterization, can be used to define a rotation by four numbers.
- Quaternions are a popular choice for the representation of rotations in three dimensions because compact, no singularity and naturally reflect the topology to the space of orientations

$$u = \left(u_1, u_2, u_3, u_4\right) \text{ with } u_1^2 + u_2^2 + u_3^2 + u_4^2 = 1$$
$$\left(u_1, u_2, u_3, u_4\right) = \left(\cos\theta/2, n_x \sin\theta/2, n_y \sin\theta/2, n_z \sin\theta/2\right) \text{ with } n_x^2 + n_y^2 + n_z^2 = 1$$

- # DOF = 6
- Topology = $\mathbb{R}^3 \times SO(3)$

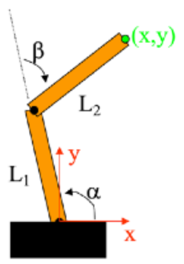
CONFIGURATION SPACE
- Configuration of a moving object is a specification of the position of every point on the object, usually express as a vector of position & orientation $q = \left(q_1, q_2, \ldots, q_n\right)$
- Configuration space C is the set of all possible configurations (a configuration is a point in C)



Workspace C-space

<u>Dimension</u>
- Minimum number of parameters needed to specify the configuration of the object completely = DOFS
- Important to keep minimum number because otherwise coupling effects and many tests would be required

MANIPULATOR KINEMATICS
- Cartesian variables: x = [x, y]
- Joint variable q = [α, β]



$c_\alpha = \cos(\alpha)$, $s_\alpha = \sin(\alpha)$
$c_\beta = \cos(\beta)$, $s_\beta = \sin(\beta)$
$c_+ = \cos(\alpha+\beta)$, $s_+ = \sin(\alpha+\beta)$

- Position of end effector: $\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} L_1 c_\alpha \\ L_1\ s_\alpha \end{pmatrix} + \begin{pmatrix} L_2 c_+ \\ L_2\ s_+ \end{pmatrix}$
- A point can be transformed by a rotation and a translation
  - 2D: $\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \text{Cos}(\theta) & -\text{Sin}(\theta) \\ \text{Sin}(\theta) & \text{Cos}(\theta) \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} tx \\ ty \end{pmatrix}$
  - 3D: $p' = Rp + T$ or $p = R^t\left(p' - T\right)$
- Homogeneous transforms: $p' = \begin{pmatrix} & R & & T \\ 0 & 0 & 0 & 1 \end{pmatrix} p$
  - Advantage: Matrix multiplications instead of addition of translation vector and matrix transpose instead of inverse

DENAVIT-HARTENBERG RULES

- convention to describe any robot kinematically by giving four quantities for each link i:

Name	Symbol	Description	In link frames
link length	**a_{i-1}**	mutual perpendicular defined for link i -1 (shortest distance)	distance from Z_{i-1} to Z_i measured along X_{i-1}
link twist	**α_{i-1}**	angle between axis i-1 and i about axis i-1	angle from Z_{i-1} to Z_i measured about X_{i-1}
link offset	**d_i**	signed distance measured along the axis of joint i from the point where a_{i-1} intersects to the point where ai intersects (variable is joint i is prismatic)	distance from X_{i-1} to X_i measured along Z_i
joint angle	**θ_i**	angle between a_{i-1} and a_i measured about the joint axis i	angle from X_{i-1} to X_i measured about Z_i

- First and last links: $a_0 = a_n = 0, \quad \alpha_0 = \alpha_n = 0, \quad d_1 = d_n = 0, \quad \theta_1 = \theta_n = 0$
- Revolute joint: θ_i is joint variable and other three are fixed
- Prismatic joint: d_i is joint variable and other three are fixed

Attach a frame {i} to link i

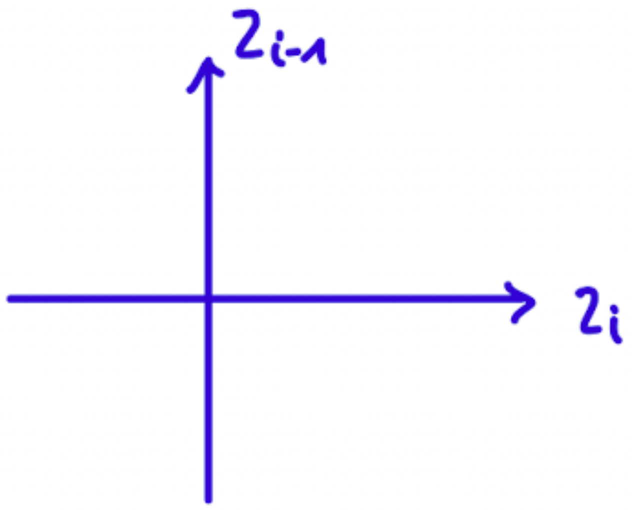
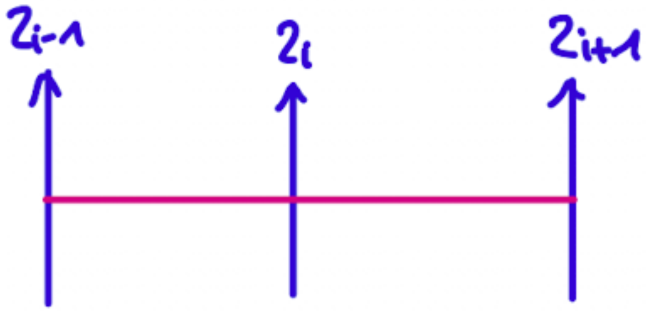
- Z-axis along joint axis i with origin where the a_i perpendicular intersects the joint axis i
- X-axis along ai in the direction from joint i to joint i+1
- Y-axis by right hand rule

<u>Link-frame attachment procedure</u>

1. Determine z-Axis for each joint
   - If the joint is prismatic: z-Axis is along the direction of movement
   - If the joint is rotational: z-Axis is along the direction of rotation
2. Determine origin and x-Axis for each joint/coordinate frame. Look at the relation between z_{i-1} and z_i:
   - If z_{i-1} and z_i are a pair of skew lines: determine the line perpendicular on both skew lines (line with shortest distance). The intersection of this line and z_{i-1} is the origin of the coordinate frame $i-1$ and x_{i-1} is on this line towards z_i.
   - If z_{i-1} and z_i intersect in only one point: the point is the origin of the coordinate frame $i-1$. x_{i-1} is the cross-product between z_{i-1} and z_i. Choose the direction of x_{i-1} such that the α_{i-1} parameter of the DH parameters is > 0.
   - If z_{i-1} and zi are parallel: we choose the origin of $i-1$ such that the d_i parameter of the DH parameters is 0. x_{i-1} is along the line from z_{i-1} to z_i.
   - If z_{i-1} and z_i coincide: the origin of $i-1$ is the origin of i and x_{i-1} is arbitrary.

3. Assign the Y_i-axis to complete a right-hand coordinate system.
4. Assign {0} to match {1} when the first joint variable is zero. For {N}, choose an origin location and X_N direction freely, but generally so as to cause as many linkage parameters as possible to become zero.

Exceptions

Z-Axis intersects	**Z-Axis parallel**
	
$\vec{x}_{i-1} = \pm \frac{\vec{z}_{i-1} \times \vec{z}_i}{\lVert \vec{z}_{i-1} \times \vec{z}_i \rVert}$	Choose origin sucht that $d_i = 0$

Example
- All joint axes are perpendicular to the plane —> Z shows into plane —> parallel —> d = 0

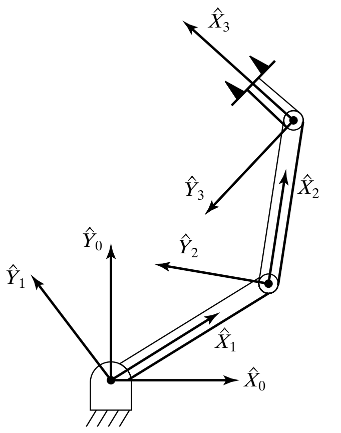


FIGURE 3.7: Link-frame assignments.

	Angle around x	Transl. along x	Transl. along z	Angle around z
i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	0	θ_1
2	0	L_1	0	θ_2
3	0	L_2	0	θ_3

FIGURE 3.8: Link parameters of the three-link planar manipulator.

$$ {}^4_0T = \begin{pmatrix} \cos(\theta_3+\theta_2+\theta_1) & -\sin(\theta_3+\theta_2+\theta_1) & 0 & l_3\cos(\theta_3+\theta_2+\theta_1) + l_2\cos(\theta_2+\theta_1) + l_1\cos\theta_1 \\ \sin(\theta_3+\theta_2+\theta_1) & \cos(\theta_3+\theta_2+\theta_1) & 0 & l_3\sin(\theta_3+\theta_2+\theta_1) + l_2\sin(\theta_2+\theta_1) + l_1\sin\theta_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \Rightarrow {}^0p(\Theta_1,\Theta_2,\Theta_3) = {}^0p(\Theta) = \begin{pmatrix} l_3\cos_{123} + l_2\cos_{12} + l_1\cos_1 \\ l_3\sin_{123} + l_2\sin_{12} + l_1\sin_1 \\ \Theta_3+\Theta_2+\Theta_1 \end{pmatrix} $$

Link transformations
1. **α**: Rotation along current x-Axis (x_{i-1}) to make z_{i-1} and z_i parallel/aligned
2. **a**: Translation along current x-Axis (x_{i-1})
3. **d**: Translation along new z-Axis (z_i; obtained from z_{i-1} by the rotation in a)
4. **θ**: Rotation along new z-Axis (z_i)

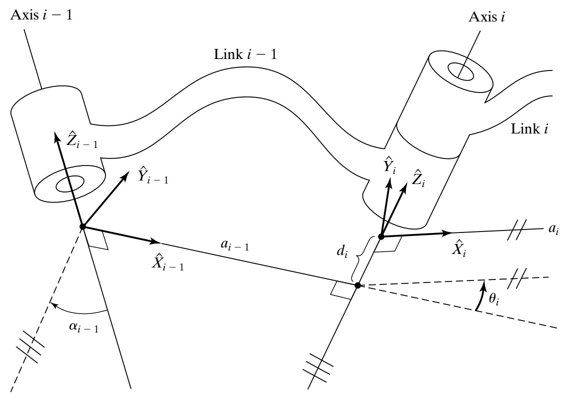


FIGURE 3.5: Link frames are attached so that frame $\{i\}$ is attached rigidly to link i.

5. $^{i}_{i-1}\boldsymbol{T} = \boldsymbol{T}(0,0,d_i)\cdot\boldsymbol{R}(z,\theta_i)\cdot\boldsymbol{T}(a_i,0,0)\cdot\boldsymbol{R}(x,\alpha_i) = \begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&1&d_i\\0&0&0&1\end{pmatrix}\cdot\begin{pmatrix}C\theta_i&-S\theta_i&0&0\\S\theta_i&C\theta_i&0&0\\0&0&1&0\\0&0&0&1\end{pmatrix}\cdot\begin{pmatrix}1&0&0&a_i\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{pmatrix}\cdot\begin{pmatrix}1&0&0&0\\0&C\alpha_i&-S\alpha_i&0\\0&S\alpha_i&C\alpha_i&0\\0&0&0&1\end{pmatrix} = \begin{pmatrix}C\theta_i&-C\alpha_i S\theta_i&S\alpha_i S\theta_i&a_i C\theta_i\\S\theta_i&C\alpha_i C\theta_i&-C\theta_i S\alpha_i&a_i S\theta_i\\0&S_{\alpha_i}&C_{\alpha_i}&d_i\\0&0&0&1\end{pmatrix}$

- Inverse for determining the forward kinematics of the robot: $^{i-1}_{i}T = \begin{bmatrix}c\theta_i&-s\theta_i&0&a_{i-1}\\s\theta_i c\alpha_{i-1}&c\theta_i c\alpha_{i-1}&-s\alpha_{i-1}&-s\alpha_{i-1}d_i\\s\theta_i s\alpha_{i-1}&c\theta_i s\alpha_{i-1}&c\alpha_{i-1}&c\alpha_{i-1}d_i\\0&0&0&1\end{bmatrix}$
- $^{A}_{B}$ T: transformations from coordinates that are relative to frame B to frame A: $^{A}_{B}T^{B}p = {}^{A}p$
- Link transformations can be multiplied to find a single transformation that relates frame {N} to frame {0}: $^{0}_{N}T = {}^{0}_{1}T^{1}_{2}T^{2}_{3}T\ldots N-1_{N}T$

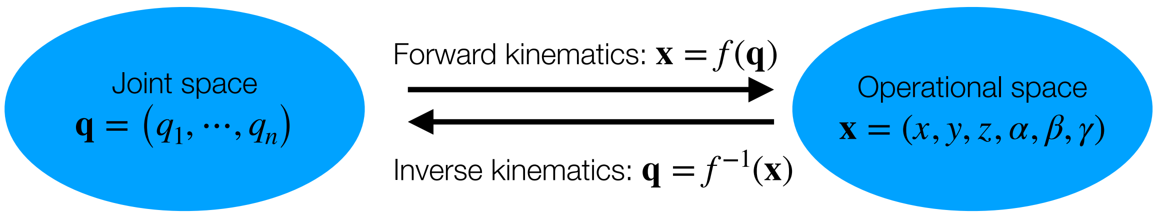
<u>Distorted version of DH</u>

- x-axis goes from previous link is x-axis for next link

YAW-PITCH-ROLL REPRESENTATION FOR ORIENTATION

- $^{n}_{0}T == \begin{bmatrix}^{n}_{0}R&^{n}_{0}P\\0&1\end{bmatrix} = \begin{bmatrix}n&s&a&^{n}_{0}P\\0&0&0&1\end{bmatrix} == \begin{bmatrix}C\phi C\vartheta & C\phi S\theta S\psi - S\phi C\psi & C\phi S\theta C\psi + S\phi S\psi & p_x\\ S\phi C\theta & S\phi S\theta S\psi + C\phi C\psi & S\phi S\theta C\psi - C\phi S\psi & p_y\\ -S\theta & C\theta S\psi & C\theta C\psi & p_z\\ 0&0&0&1\end{bmatrix}$
- with $\theta = \sin^{-1}(-n_z)$, $\psi = \cos^{-1}\left(\frac{a_z}{\cos\theta}\right)$, $\phi = \cos^{-1}\left(\frac{n_x}{\cos\theta}\right)$

3. Inverse Kinematics


Joint space
$\mathbf{q} = (q_1, \cdots, q_n)$
Forward kinematics: $\mathbf{x} = f(\mathbf{q})$
Inverse kinematics: $\mathbf{q} = f^{-1}(\mathbf{x})$
Operational space
$\mathbf{x} = (x, y, z, \alpha, \beta, \gamma)$

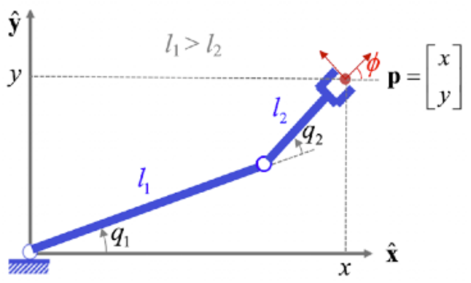
- Forward kinematics: Given a joint configuration, find the pose (position/orientation) of some part of the robot given q —> find x = f(q)
- Inverse kinematics: Calculate required position of joints based on desired position of end effector —> given T or x = f(q), find q
- For exam: notice <u>Trigonometric functions</u>
- Direct inversion of forward kinematics —> System of nonlinear trigonometric equations
- Nonlinear problem: Is there a solution? Unique or multiple solutions? How to solve it?

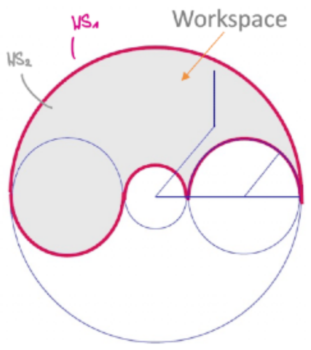
WORKSPACE

- Example for RR-Robot (Two rotations)
- full circle without limit
- with joint limits workspace is reduced (analytic expression more complicated)

<u>Homotopic configuration</u>

- Adjust 2 DOF independently —> more accurate, high precision because velocities don't influence themselves





MULTIPLICITY OF SOLUTION

- What solution to choose?
  - Shortest joint distance between configuration is preferred
  - In general: If there are N possible configurations for desired position q_b from the initial configuration q_A:

$$\mathbf{q}_b = \underset{\mathbf{q}}{\arg\min} \left\| \mathbf{q} - \mathbf{q}_A \right\| \quad \text{for} \quad \mathbf{q} \in \{\mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_N\}$$

- Redundancy
  - n joints: $\mathbf{q} = (q_1, \cdots, q_n)$ —> n DoF (expect for linear joints, only true for non-redundant joints)
  - m Dimension of task space: $\mathbf{x} = (x_1, x_2, \cdots, x_m)$
  - —> Robot is redundant with respect to this task if n>m
- Complexity
  - Equations are nonlinear
  - There can be one, multiple, infinite (when there is redundancy) or no admissible solution (outside of workspace)
  - Existence of a solution for the **position** is guaranteed when the position (of the end effector) belongs to the **reachable workspace**
  - Existence of a solution for the **pose** is guaranteed when the position (of the end effector) belongs to the **dexterous workspace**

ANALYTIC SOLUTIONS

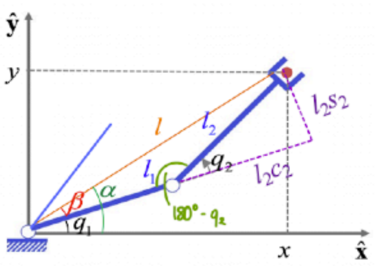
- Exact, preferred, when it can be found

<u>Geometric ad-hoch approach</u>

- Applicable to robots with few DOF (3 or less) or to first 3 DOF

- Not a generic solution —> depend on the robot
  - **Example:**

Find the inverse kinematics for the position of the R-R robot using a geometric approach



Solution

For q_2:

- Using the law of cosines:

$$l^2 = l_1^2 + l_2^2 - 2l_1l_2\cos(\underline{180 - q_2})$$
$$x^2 + y^2 = l_1^2 + l_2^2 + 2l_1l_2\cos(q_2)$$
$$c_2 = \frac{x^2+y^2-(l_1^2+l_2^2)}{2l_1l_2}$$

- Using a trigonometric identity:

$$s_2^2 + c_2^2 = 1$$
$$s_2 = \pm\sqrt{1-c_2^2}$$
$$q_2 = \text{atan2}(s_2, c_2) \rightarrow \text{second solution } + -$$

↳ from 0 to 360°

For q_1 (using the geometry of the figure)

$$q_1 = \alpha - \beta$$
$$\alpha = \text{atan2}(y, x)$$
$$\beta = \text{atan2}(l_2s_2, l_1 + l_2c_2)$$

Inverse kinematics:

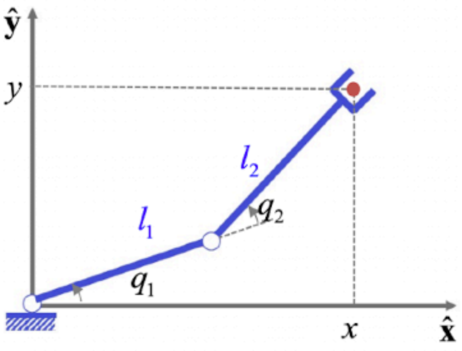
$$q_1 = \text{atan2}(y,x) - \text{atan2}(l_2s_2, l_1 + l_2c_2)$$
$$q_2 = \text{atan2}(s_2, c_2)$$

Algebraic approach (solution of polynomial equations)

Solution

- **Forward kinematics:**

$$x = l_1 c_1 + l_2 c_{12} \qquad y = l_1 s_1 + l_2 s_{12}$$



- **For q_2:**

From F.K.
$$\begin{cases} x^2 = l_1^2 c_1^2 + l_2^2 c_{12}^2 + 2l_1c_1l_2c_{12} \\ y^2 = l_1^2 s_1^2 + l_2^2 s_{12}^2 + 2l_1s_1l_2s_{12} \end{cases}$$

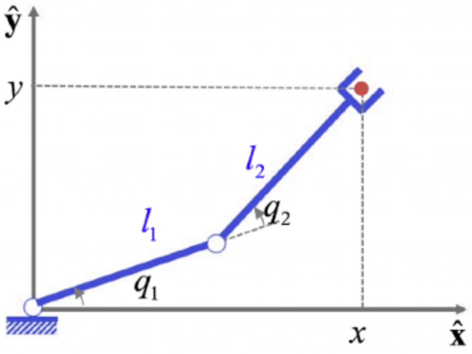
$$x^2 + y^2 = l_1^2 + l_2^2 + 2l_1l_2(\underbrace{c_1c_{12} + s_1s_{12}}_{c(\theta_1 - \theta_1 + \theta_2) = c_2})$$
$$x^2 + y^2 = l_1^2 + l_2^2 + 2l_1l_2c_2$$
$$c_2 = \frac{x^2+y^2-(l_1^2+l_2^2)}{2l_1l_2}$$

$$s_2^2 + c_2^2 = 1$$
$$s_2 = \pm\sqrt{1-c_2^2}$$
$$q_2 = \text{atan2}(s_2, c_2)$$

Solution

For q_1 (expanding terms from forward kinematics):

$$x = l_1c_1 + l_2(c_1c_2 - s_1s_2)$$
$$y = l_1s_1 + l_2(s_1c_2 + c_1s_2)$$



Equations are linear in c_1, s_1: solve for them:

$$x = c_1(l_1 + l_2c_2) - s_1(l_2s_2)$$
$$y = s_1(l_1 + l_2c_1) + c_1(l_2s_2)$$

In matrix form:

$$\begin{bmatrix} l_1 + l_2c_2 & -l_2s_2 \\ l_2s_2 & l_1 + l_2c_2 \end{bmatrix}\begin{bmatrix} c_1 \\ s_1 \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} \xrightarrow{\text{solving}} \begin{cases} s_1 = \dfrac{y(l_1 + l_2c_2) - xl_2s_2}{\det} \\ c_1 = \dfrac{x(l_1 + l_2c_2) + yl_2s_2}{\det} \\ \det = l_1^2 + l_2^2 + 2l_1l_2c_2 \end{cases}$$

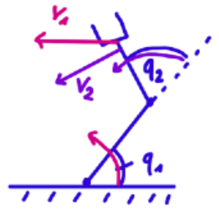
$$q_1 = \text{atan2}(s_1, c_1)$$

Systematic reduction approach (obtain a reduced set of equations)
Kinematic decpuoling (Pieper) robots with 6 DOF
- When the last 3 axes are revolute and they intersect each other (spherical twist)

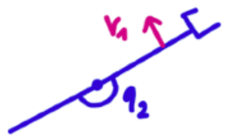
NUMERIC SOLUTIONS

- Iterative, needed when there is redundancy n > m, no analytic solution or too complicated
- Easter to obtain, but slower because of iterations
- Use of the Jacobian matrix of the forward kinematics $J(\mathbf{q}) = \dfrac{\partial f(\mathbf{q})}{\partial \mathbf{q}}$ $\quad J(\mathbf{q}) \in {}^{\sim n \times m}$
  - $$\mathbf{J} = \begin{bmatrix} \dfrac{\partial \mathbf{f}}{\partial x_1} & \cdots & \dfrac{\partial \mathbf{f}}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \nabla^{\mathrm{T}} f_1 \\ \vdots \\ \nabla^{\mathrm{T}} f_m \end{bmatrix} = \begin{bmatrix} \dfrac{\partial f_1}{\partial x_1} & \cdots & \dfrac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \dfrac{\partial f_m}{\partial x_1} & \cdots & \dfrac{\partial f_m}{\partial x_n} \end{bmatrix}$$
  - $\delta Y = J(X)\delta X$
  - n: size of q (number of joints)
  - m: size of x (size of the task space)
  - If more than 6 joints, joints are independent.
- Idea: x - f(q) = 0 —> find q using iterations such that the difference is 0

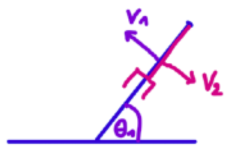
Newtons methods
- Problem: given a x_d, find q such that $\mathbf{x}_d - f(\mathbf{q}) = 0$
- Procedure: first order Taylor approximation
- Solution: $\mathbf{q}_{k+1} = \mathbf{q}_k + J^{-1}\left(\mathbf{q}_k\right)\left(\mathbf{x}_d - f\left(\mathbf{q}_k\right)\right)$
- Algorithm:
  - Start with initial q_0
  - Iteratively update q_{k+1}
  - Stop when $\underbrace{\| \mathbf{x}_d - f\left(\mathbf{q}_k\right) \| < \varepsilon}_{\text{Small Cartesian error}}$ or $\underbrace{\| \mathbf{q}_{k+1} - \mathbf{q}_k \| < \varepsilon}_{\text{Small joint increment}}$ $\quad \varepsilon$: small value
- Comments
  - Convergence if we start close to the solution (Result depends on the initial value)
  - When redundant, J is not square, use pseudo-inverse
  - - Computation time of the inverse
  - - Problems near singularities of J: det(J) = 0 —> no inverse, appears if arm is straight or folded back, then q1 and q2 cause a colinear velocity, 1 DOF is los —> high joint rates/ vector p can't be reached anymore
  - + Its fast



Normal case



Arm is straight



Arm folded back

Gradient descent method
- Objective: Minimize the generic function g(q)
- Idea:
  - Start with an initial value q_0
  - Move in the negative direction of the gradient:
    $\mathbf{q}_{k+1} = \mathbf{q}_k - \alpha \nabla g\left(\mathbf{q}_k\right) \quad \alpha \in {\sim}^{\sim}$: size of the step
  - Step size must guarantee a maximum descent of q(q) in every iteration
    - α very high: divergence (minimum is not found)
    - α very small: slow convergence
- Procedure:
  - Define a scalar error function: $g(\mathbf{q}) = \dfrac{1}{2} \left\| \mathbf{x}_d - f(\mathbf{q}) \right\|^2 \longleftarrow g : {\sim}^n \rightarrow \sim$
  - Objective: Minimize the error: $\min\limits_{\mathbf{q}} g(\mathbf{q})$

- Compute the gradient of q and apply gradient descent: $\mathbf{q}_{k+1} = \mathbf{q}_k + \alpha J^T(\mathbf{q}_k)\left(\mathbf{x}_d - f(\mathbf{q}_k)\right)$
- Computationally simpler

Comparison

- Newton: quadratic convergence rate (fast) ⚡ Problems near singularity
- GDM: linear convergence rate (slow), no singularity problems, step size must be chosen carefully
- Efficient algorithm: start with GDM (safe but slow)

NUMERIC COMPUTATION OF THE JACOBIAN

- Jacobian: mapping from the joint velocities to the end effector linear and angular velocities
- Very tedious to manually compute the Jacobian —> Compute numerically by numeric differentiation
- The Jacobian (considering only position): $J(\boldsymbol{q}) = \begin{bmatrix} \frac{\partial \boldsymbol{x}_p}{\partial q_1} & \frac{\partial \boldsymbol{x}_p}{\partial q_2} & \cdots & \frac{\partial \boldsymbol{x}_p}{\partial q_n} \end{bmatrix}$ with $\mathbf{x}_p = (x, y, z) = \mathbf{f}(q_1, \cdots, q_n)$
- Approximation of the derivative of the position x_p with respect to the joint q_i:

$$\frac{\partial \mathbf{x}_p}{\partial q_i} \approx \frac{\Delta \mathbf{x}_p}{\Delta q_i} = \frac{\mathbf{f}(q_1, \cdots, q_i + \Delta q_i, \cdots, q_n) - \mathbf{f}(q_1, \cdots, q_n)}{\Delta q_i}$$

PROPAGATION OF FORCE AND TORQUES

${}^i f_i = {}^{i}_{i+1}R\ {}^{i+1}f_{i+1}$	${}^{i+1}f_{i+1} = {}^{i+1}_{i}R\ {}^i f_i$
${}^i n_i = {}^{i}_{i+1}R\ {}^{i+1}n_{i+1} + {}^{i}_{i+1}P \times \left({}^{i}_{i+1}R\ {}^{i+1}f_{i+1}\right)$	${}^{i+1}n_{i+1} = {}^{i+1}_{i}R\left({}^i n_i - {}^{i}_{i+1}P \times \left({}^{i}_{i+1}R\ {}^{i+1}f_{i+1}\right)\right)$

4. Manipulator dynamics

- Goal: equations of motion for any n DOF system —> n coupled second order ODEs
- relationship between motion of bodies and its causes, namely the forces acting on the bodies and the properties of the bodies (particularly mass and moment of inertia) influencing that movement<—> kinematics, where we were not concerned with the physical phenomena causing robot movement, but only with the movement itself.
- System may be linear or nonlinear, conservative or non-conservative (looses energy)
- Develop expression of form $\dot{q} = f(q, t)$
- Afterwards: choose appropriate controller that will put our dynamical system in a desired state (configuration)

GENERAL FORMULAS

	Linear	**Angular**
Momentum	Newton equation: p = mv	Angular momentum: $\vec{H} = \vec{p} \times \vec{r}$
Force	$\vec{F} = \frac{d}{dt}\vec{p} = m\frac{d}{dt}\vec{v} = m\vec{a}$	Torque: $\vec{N} = \frac{d}{dt}\vec{H}$

- Energy: $K = \frac{1}{2}mv^2$ and $P = mgh$ (if not perpendicular to g: $\vec{p} = m\vec{g}^T\vec{x}$)

<u>Acceleration of a rigid body</u>

- Linear acceleration
  - Velocity of a vector ${}^BQ$ as seen from frame {A} when the origins are coincident (${}^A\Omega_B$: rotational velocity of {B} relative to {A}): ${}^AV_Q = \underbrace{{}^A_BR\,{}^BV_Q}_{\text{Q is changing}} + \underbrace{{}^A\Omega_B \times {}^A_BR\,{}^BQ}_{\text{Frames are rotating}}$
  - Acceleration if ${}^BV_Q = {}^B\dot{V}_Q = 0$: ${}^A\dot{V}_Q = \underbrace{{}^A\dot{V}_{BORG}}_{\text{Acc. of origin of B}} + {}^A\Omega_B \times \left({}^A\Omega_B \times {}^A_BR\,{}^BQ\right) + {}^A\dot{\Omega}_B \times {}^A_BR\,{}^BQ$
- Angular acceleration:
  - {B} is rotating relative to A with ${}^A\Omega_B$ and {C} is rotating relative to {B} with ${}^B\Omega_C$: ${}^A\Omega_C = {}^A\Omega_B + {}^A_BR\,{}^B\Omega_C$
  - Differentiating: ${}^A\dot{\Omega}_C = {}^A\dot{\Omega}_B + {}^A_BR\,{}^B\dot{\Omega}_C + {}^A\Omega_B \times {}^A_BR\,{}^B\Omega_C$

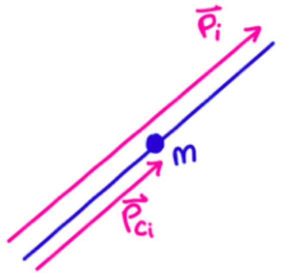
EULER-LAGRANGE EQUATIONS

- Derive equations of motion by using energy methods —> need to know the conservative (kinetic and potential) energy and non-conservative (dissipative) terms
- Lagrange equations: $L = K - P$
- Tau needs to be provided to keep robot in position:

$$\tau_i = \sum_\mu F_\mu \frac{\partial x_\mu}{\partial q_i} = \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} \qquad x_\mu = x_\mu\left(q_1 \cdots q_N, t\right)$$

<u>Algorithm</u>

1. Determine angular velocities ${}^i\omega_i$ and center of masses ${}^0P_{C_i}$ —> v_{C_i}
2. Compute kinetic and potential energies (for every link)
   - Kinetic energy
     - Kinetic energy for each link i (left: due to linear velocity, right: due to angular velocity):
       $k_i = \frac{1}{2}m_i v_{C_i}^{\mathrm{T}} \cdot v_{C_i} + \frac{1}{2}{}^i\omega_i^{\mathrm{T}} \cdot {}^{C_i}I_i \cdot {}^i\omega_i$
     - Another way: $k(\Theta, \dot{\Theta}) = \frac{1}{2}\dot{\Theta}^{\mathrm{T}}M(\Theta)\dot{\Theta}$ where M is the n x n matrix from the MVG equation
   - Potential energy
     - Potential energy for each link i: $u_i = -m_i \cdot {}^0g^{\mathrm{T}} \cdot {}^0P_{C_i} + u_{\mathrm{ref}_i}$



- ${}^0P_{C_i}$: center o f mass of link i
- u_{ref}: constant

3. Compute the derivatives (For $\dot{\Theta}$, also differentiate terms that contain θ)
4. Compute the joint torque vector
    - Computation of tau: $\tau = \frac{d}{dt}\frac{\partial k}{\partial \dot{\Theta}} - \frac{\partial k}{\partial \Theta} + \frac{\partial u}{\partial \Theta}$
    - Or per joint: $\tau_i = \frac{d}{dt}\frac{\partial k}{\partial \dot{\Theta}_i} - \frac{\partial k}{\partial \Theta_i} + \frac{\partial u}{\partial \Theta_i}$

- Example: 1DOF system
  - Consider a particle of mass *m*
  - Using Newton's second law:

    $$m\ddot{y} = f - mg$$

  - Now define the kinetic and potential energies:

    $$K = \frac{1}{2}m\dot{y}^2 \qquad P = mgy$$

  - Rewrite the above differential equation
    - Left side:

    $$m\ddot{y} = \frac{d}{dt}(m\dot{y}) = \frac{d}{dt}\frac{\partial}{\partial \dot{y}}\left(\frac{1}{2}m\dot{y}^2\right) = \frac{d}{dt}\frac{\partial K}{\partial \dot{y}}$$

    - Right side:

    $$mg = \frac{\partial}{\partial y}(mgy) = \frac{\partial P}{\partial y}$$

  - Thus we can rewrite the initial equation:

    $$\frac{d}{dt}\frac{\partial K}{\partial \dot{y}} = f - \frac{\partial P}{\partial y}$$

  - Now we make the following definition:

    $$L = K - P$$

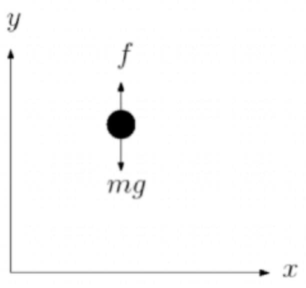
  - *L* is called the *Lagrangian*
    - We can rewrite our equation of motion again:

    $$\frac{d}{dt}\frac{\partial L}{\partial \dot{y}} - \frac{\partial L}{\partial y} = f$$

    take derivative of L to get result

    linear or rotational torque

  - Thus, to define the equation of motion for this system, all we need is a description of the potential and kinetic energies



- In general: calculate kinetic and potentiale Energy of every link and sum: $P = \sum P_i$, $K = \sum K_i$
- Write equations of motion for any nDOF system as $\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_i} - \frac{\partial L}{\partial q_i} = \tau_i$
  - If $\tau_i = 0$: Switch between kinetic and potential energy —> pendulum
  - Left side contains conservative terms, right side contains non-conservative terms
  - —> set of n coupled 2nd order differential equations

FORCE AND TORQUE RELATION

- $F \cdot \delta x = \tau \cdot \delta\Theta$
  - F: Cartesian force-moment vector acting at the end-effector
  - δx: Infinitesimal cartesian displacement of the end-effector
  - τ: vector of torques at the joints
  - $\delta\Theta$: Vector of infinitesimal joint displacements
- Replace with Jacboian $\delta X = J\delta\Theta$ —> $F^TJ = \tau^T$ / $\tau = J^TF$

NEWTON-EULER EQUATIONS

Rigid body dynamics: Get joint torques τ out of joint trajectory θ

- Newton's equation: Forces: $F = m\dot{v}_C$
- Euler's equation: Torque: $N = {}^CI\dot{\omega} + \omega \times {}^CI\omega$
  - ᶜI: inertia matrix given by $\mathrm{I}_c = \int \mathrm{sk(r)}^T\mathrm{sk(r)}dm$
  - Example: Body of uniform density

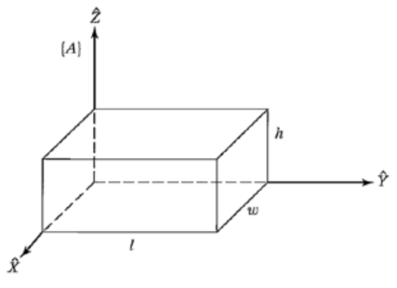


FIGURE 6.2: A body of uniform density.

$${}^A\mathrm{I} = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{xy} & I_{yy} & -I_{yz} \\ -I_{xz} & -I_{yz} & I_{zz} \end{bmatrix} = \begin{bmatrix} \frac{m}{3}\left(l^2+h^2\right) & -\frac{m}{4}\omega l & -\frac{m}{4}h\omega \\ -\frac{m}{4}\omega l & \frac{m}{3}\left(\omega^2+h^2\right) & -\frac{m}{4}hl \\ -\frac{m}{4}h\omega & -\frac{m}{4}hl & \frac{m}{3}\left(l^2+\omega^2\right) \end{bmatrix}$$

$$I_{xx} = \int_B \left(y^2+z^2\right)dm \quad I_{xy} = \int_B xy\,dm$$
$$I_{yy} = \int_B \left(x^2+z^2\right)dm \quad I_{xz} = \int_B xz\,dm$$
$$I_{zz} = \int_B \left(x^2+y^2\right)dm \quad I_{yz} = \int_B yz\,dm$$

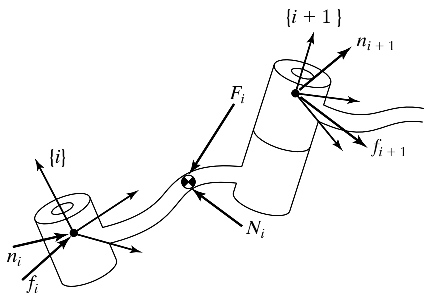
- Facts: I is a positive-definite symmetric matrix, not in general constant but frame dependent ($\mathbf{I}_c^i = R_i^T \mathbf{I}_c R_i$), any rigid body has a set of principal directions with respect to which the inertia matrix is diagonal
- If xy is the plane of symmetry, then $I_{XZ} = I_{YZ} = 0$
- If body is axis-symmetric (e.g. about Z) then I is diagonal and 2 of the moment are equal ($I_{XX} = I_{YY}$)

<u>Parallel-axis Theorem</u>

- If a body is made to rotate about a new axis which is parallel to the first axis (trough the center of mass) and displaced from it by a distance d, then the moment of inertia ${}^{A}I$ with respect to the new axis is related to ${}^{C}I$:

$$^{A}I = ^{C}I + m\ \text{sk}(d)^T \text{sk}(d) = \begin{bmatrix} I_{xx} + md_x^2 & -\left(I_{xy} + md_xd_y\right) & -\left(I_{xz} + md_xd_z\right) \\ * & I_{yy} + md_y^2 & -\left(I_{yz} + md_yd_z\right) \\ * & * & I_{zz} + md_z^2 \end{bmatrix}$$

- Example: Same body but describe in a coordinate system with the origin at the body's center of mass:
  - $\begin{bmatrix} d_x \\ d_y \\ z_z \end{bmatrix} = \frac{1}{2}\begin{bmatrix} \omega \\ l \\ h \end{bmatrix} \rightarrow {}^{C}I = \begin{bmatrix} \frac{m}{12}\left(h^2 + l^2\right) & 0 & 0 \\ 0 & \frac{m}{12}\left(\omega^2 + h^2\right) & 0 \\ 0 & 0 & \frac{m}{12}\left(l^2 + \omega^2\right) \end{bmatrix}$
  - —> Result is diagonal, so frame {C} must represent the principal axes of this body



<u>Algorithm: Forward phase and backward phase</u>

- Compute velocities and accelerations (both rotational and linear) for each joint:

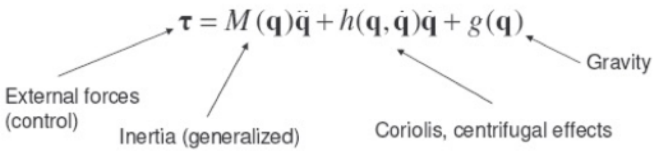
	rotational joint i+1	prismatic/translational joint i+1
rotational velocity and acceleration	${}^{i+1}\omega_{i+1} = {}^{i+1}_{i}R \cdot {}^{i}\omega_i + \dot{\Theta}_{i+1} \cdot {}^{i+1}Z_{i+1}$ ${}^{i+1}\dot{\omega}_{i+1} = {}^{i+1}_{i}R \cdot {}^{i}\dot{\omega}_i + {}^{i+1}_{i}R \cdot {}^{i}\omega_i \times \dot{\Theta}_{i+1} \cdot {}^{i+1}Z_{i+1} + \ddot{\Theta}_{i+1}{}^{i+1}Z_{i+1}$	$\omega_{i+1} = {}^{i+1}_{i}R \cdot {}^{i}\omega_i$ ${}^{i+1}\dot{\omega}_{i+1} = {}^{i+1}_{i}R \cdot {}^{i}\dot{\omega}_i$
linear velocity and acceleration	${}^{i+1}\dot{v}_{i+1} = {}^{i+1}_{i}R\left({}^{i}\dot{\omega}_i \times {}^{i}P_{i+1} + {}^{i}\omega_i \times \left({}^{i}\omega_i \times {}^{i}P_{i+1}\right) + {}^{i}\dot{v}_i\right)$	${}^{i+1}\dot{v}_{i+1} = {}^{i+1}_{i}R\left({}^{i}\dot{\omega}_i \times {}^{i}P_{i+1} + {}^{i}\omega_i \times \left({}^{i}\omega_i \times {}^{i}P_{i+1}\right) + {}^{i}\dot{v}_i\right) + 2 \cdot {}^{i+1}\omega_{i+1} \times \dot{d}_{i+1}{}^{i+1}Z_{i+1} + \ddot{d}_{i+1}{}^{i+1}Z_{i+1}$
linear acceleration of the center of mass	${}^{i}\dot{v}_{C_i} = {}^{i}\dot{\omega}_i \times {}^{i}P_{C_i} + {}^{i}\omega_i \times \left({}^{i}\omega_i \times {}^{i}P_{C_i}\right) + {}^{i}\dot{v}_i$	
Forces and Torques that that apply to the center of mass of each link	${}^{i}F_i = m \cdot {}^{i}\dot{v}_{C_i}$ —> are required to act on the center of mass of link i ${}^{i}N_i = {}^{C_i}I_i \cdot {}^{i}\dot{\omega}_i + {}^{i}\omega_i \times {}^{C_i}I_i \cdot {}^{i}\omega_i$ in order to make the robot move as desired.	
Compute f and n for the joints (Backward phase)	Force-Balance relationship: ${}^{i}f_i = {}^{i}_{i+1}R \cdot {}^{i+1}f_{i+1} + {}^{i}F_i$ (should be equal while resting) Torque-Balance relationship: ${}^{i}n_i = {}^{i}N_i + {}^{i}_{i+1}R \cdot {}^{i+1}n_{i+1} + \underbrace{{}^{i}P_{C_i} \times {}^{i}F_i}_{\text{Force on one site}} + \underbrace{{}^{i}P_{i+1} \times {}^{i}_{i+1}R^{i+1}f_{i+1}}_{\text{Force on other site}}$ fi: Force exerted on link i by link i-1 ni: Torque exerted on link i by link i-1	
Torque	$\tau_i = {}^{i}n_i^{\mathrm{T}} \cdot {}^{i}Z_i$	$\tau_i = {}^{i}f_i^{\mathrm{T}} \cdot {}^{i}Z_i$

- Consider gravity by setting: ${}^{0}\dot{v}_0 = -G$

- Equations of movement that have been computed this way can be rearranged into the so-called state-space equation (or M-V-G-form): $\tau = M(\Theta)\ddot{\Theta} + V(\Theta, \dot{\Theta}) + G(\Theta)$

STATE-SPACE EQUATION

- Express the dynamic equations of a manipulator in a single equation that hides some of the details but shows some of the structure of the equations
- When Newton-Euler equations are evaluated symbolically for any manipulator, they yield a dynamic equation that can the written in the form of $\tau = M(\Theta)\ddot{\Theta} + V(\Theta, \dot{\Theta}) + G(\Theta)$


$\tau = M(\mathbf{q})\ddot{\mathbf{q}} + h(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + g(\mathbf{q})$
External forces (control)
Inertia (generalized)
Coriolis, centrifugal effects
Gravity

  - $M(\Theta)$: nxnmass matrix of the manipulator —> factoring all summands that contain $\ddot{\Theta}$
  - $V(\Theta, \dot{\Theta})$: nx1 vector of centrifugal and coriolis terms
  - $G(\Theta)$: nx1 vector of gravity terms
- V can be further decomposed into components B and C, yielding the configuration-space equation (or M-B-C-G-form): $\tau = M(\Theta)\ddot{\Theta} + B(\Theta)[\dot{\Theta}_i\dot{\Theta}_j] + C(\Theta)\left[\dot{\Theta}^2\right] + G(\Theta)$
  - B: n x n (n-1)/2 matrix
  - $[\dot{\Theta}\dot{\Theta}] = \left(\dot{\Theta}_1\dot{\Theta}_2, \dot{\Theta}_1\dot{\Theta}_3, \ldots, \dot{\Theta}_{n-1}\dot{\Theta}_n\right)^{\mathrm{T}}$ with length n(n-1)/2
  - C: n x n matrix
  - $\left[\dot{\Theta}^2\right] = \left(\dot{\Theta}_1^2, \dot{\Theta}_2^2, \ldots, \dot{\Theta}_n^2\right)^{\mathrm{T}}$ with length n
- Total force: $\boldsymbol{F} \underbrace{-m\frac{\mathrm{d}\boldsymbol{\omega}}{\mathrm{d}t} \times \boldsymbol{r}'}_{\text{Euler force}} \underbrace{-2m\boldsymbol{\omega} \times \boldsymbol{v}'}_{\text{Coriolis force}} \underbrace{-m\boldsymbol{\omega} \times (\boldsymbol{\omega} \times \boldsymbol{r}')}_{\text{centrifugal force}} = m\boldsymbol{a}'$
  - F: vector sum of the physical forces acting on the object
  - ω: angular velocity of the rotating reference frame relative to the inertial frame
  - v': velocity relative to the rotating reference frame
  - r': position vector of the object relative to the rotating reference frame
  - a': acceleration relative to the rotating reference frame

CARTESIAN STATE-SPACE EQUATION

- Relationship between doing space and Cartesian acceleration
- Start with definition of Jacobian: $\dot{\mathcal{X}} = J\dot{\Theta}$
- Differentiate, solve for $\ddot{\Theta}$, get the MVG equation: $F = M_x\ddot{x} + V_x\dot{x} + G_x$
- Expressions for the terms in Cartesian dynamics:

$$\begin{aligned} M_x(\Theta) &= J^{-T}(\Theta)M(\Theta)J^{-1}(\Theta) \\ V_x(\Theta, \dot{\Theta}) &= J^{-T}(\Theta)\left(V(\Theta, \dot{\Theta}) - M(\Theta)J^{-1}(\Theta)\dot{J}(\Theta)\dot{\Theta}\right) \qquad (6.99) \\ G_x(\Theta) &= J^{-T}(\Theta)G(\Theta) \end{aligned}$$

FORCES AND TORQUES FOR STATIC MANIPULATORS

- propagation of forces and torques in a non-moving manipulator
- Force that affects a link: ${}^{i}f_i = {}^{i}_{i+1}R \cdot {}^{i+1}f_{i+1}$
- Torque that affects a link: ${}^{i}n_i = {}^{i}_{i+1}R \cdot {}^{i+1}n_{i+1} + {}^{i}P_{i+1} \times {}^{i}f_i$
- Some parts of the forces and torques apply directly to the corresponding joint, some parts are absorbed by the mechanics of the robot:

- Rotational joints: $\tau_i = {}^in_i^{\mathrm{T}\,i}Z_i = {}^in_i^{\mathrm{T}}\begin{pmatrix}0\\0\\1\end{pmatrix}$
- Prismatic joints: $\tau_i = {}^if_i^{\mathrm{T}\,i}Z_i = {}^if_i^{\mathrm{T}}\begin{pmatrix}0\\0\\1\end{pmatrix}$
- τ_i: amount of torque resp. force that is affecting the joint ant thus the amount torque resp. force that the robot should counteract in order to remain static

- Jacobian relates joint torques/forces τ_i to end effector forces and torques f and n:
$$\begin{pmatrix}\tau_1\\\tau_2\\\vdots\\\tau_n\end{pmatrix} = \tau = {}^AJ^{T\,A}F = {}^AJ^T\begin{pmatrix}{}^Af\\{}^An\end{pmatrix}$$

5. Jacobians: velocities and static forces

COMPUTE THE JACOBIAN

a) Compute velocities and derive Jacobian: $\begin{pmatrix} {}^i v_{EE} \\ {}^i \omega_{EE} \end{pmatrix} = {}^iJ \cdot \dot{\theta}$, EE is the robot's endeffector

b) Compute force-torque relation and derive Jacobian: $\tau = {}^iJ^T \cdot {}^i\mathscr{F} = {}^iJ^T \cdot \begin{pmatrix} {}^i f_{EE} \\ {}^i n_{EE} \end{pmatrix}$

c) Geometric observations: ${}^0J = \begin{pmatrix} {}^0J_v \\ {}^0J_\omega \end{pmatrix}$, ${}^iJ = \begin{pmatrix} {}^i_0R & 0_3 \\ 0_3 & {}^i_0R \end{pmatrix} {}^0$ J

- Jacobian with arbitrary rotations
- Let $p : \mathbb{R}^n \rightarrow \mathbb{R}^3$ a function that computes the coordinates of the origin of the end effector with respect to system {0}, then the full Jacobian looks like: ${}^0J = \begin{pmatrix} \frac{\partial p_1}{\partial x_1} & \frac{\partial p_1}{\partial x_2} & \cdots & \frac{\partial p_1}{\partial x_n} \\ \frac{\partial p_2}{\partial x_1} & \frac{\partial p_2}{\partial x_2} & \cdots & \frac{\partial p_2}{\partial x_n} \\ \frac{\partial p_3}{\partial x_1} & \frac{\partial p_3}{\partial x_2} & \cdots & \frac{\partial p_3}{\partial x_n} \\ {}^0\hat{Z}_1 & {}^0\hat{Z}_2 & \dots & {}^0\hat{Z}_n \end{pmatrix}$
  - upper part Jv: ${}^0\dot{p}_{EE} = \text{fkm}(\theta) = {}^0J_v \cdot \dot{\theta}$
  - lower part Jw:
    - For an n-jointed robot: ${}^iJ_\omega = \left({}^i\hat{Z}_1 \ \vdots \ {}^i\hat{Z}_2 \ \vdots \ \cdots \ \vdots \ {}^i\hat{Z}_n\right) \in \mathbb{R}^{3\times n}$
    - ${}^i\hat{Z}_j \in \mathbb{R}^3$ is the rotation axis of the j-th joint expressed in the coordinate frame i
    - If the joint is rotational: ${}^i\hat{Z}_j = {}^i_jR\,{}^jZ_j = {}^i_jR \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$
    - If the joint is prismatic, then there is no rotation axis: ${}^i\hat{Z}_j = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$
    - Example: If all joints are parallel (planar 3R-manipulator): $J = \begin{pmatrix} {}^0\hat{Z}_1 & {}^0\hat{Z}_2 & {}^0\hat{Z}_3 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}$

CHARACTERISTICS

- **Isotropic** configurations where the columns of the Jacobian become orthogonal and of equal magnitude: $J^TJ = \delta I$
  - Consider in J of end effector: One can see now that Θ_1 maps directly to a velocity in x_3 direction, and Θ_2 maps directly to a velocity in y_3 direction.
- **Singular** if det(J) = 0
  - **Workspace-boundary singularities (row of J = 0 —> no motion)** occur when the manipulator is fully stretched out or folded back on it self in such a way that the end-effector is at or very near the boundary of the workspace.
  - **Workspace-interior singularities** occur away from the workspace boundary; they generaly are caused by alining up of two or more joint axes —> endeffector cannot be chosen freely
  - —> one or more degrees of freedom are lost: some direction (or subspace) along which it is impossible to move the robot (Robot loses degree of freedom **if rows are linearly dependent —> coupled DOF, rows-axis velocity**)
  - Same columns in J —> Joints are the same, can be replaced by one

- Problem: Incremental inverse kinematics: Joint rates might become extremely large, the torques can approach infinity.
- Joint speeds and accelerations are limited, thus limiting the speeds on the next link

VELOCITY PROPAGATION FROM LINK TO LINK

- Angular velocity: ${}^{i+1}\omega_{i+1} = {}^{i+1}_{i}R \cdot {}^{i}\omega_i + \begin{pmatrix} 0 \\ 0 \\ \dot{\Theta}_{i+1} \end{pmatrix}$ $\dot{\Theta}_{i+1} = 0$ for a prismatic joint i+1
- Linear velocity: ${}^{i+1}v_{i+1} = {}^{i+1}_{i}R \left({}^{i}v_i + {}^{i}\omega_i \times {}^{i}P_{i+1}\right) + \begin{pmatrix} 0 \\ 0 \\ \dot{d}_{i+1} \end{pmatrix}$ $\dot{d}_{i+1} = 0$ for a rotational joint i+1
- Can be used to determine Jacobian indirectly: ${}^{n}J\dot{\Theta} = \begin{pmatrix} {}^{n}v_n \\ {}^{n}\omega_n \end{pmatrix}$

VELOCITIES

- $\frac{\partial f}{\partial t} = \frac{\partial f}{\partial\left(x_1, \ldots, x_n\right)} \frac{\partial \Theta}{\partial t}$ with $\frac{\partial \Theta}{\partial t} = \left(\frac{\partial \Theta_1}{\partial t}, \frac{\partial \Theta_2}{\partial t}, \ldots, \frac{\partial \Theta_n}{\partial t}\right)^T$
- short: $\dot{f} = J \cdot \dot{\Theta}$

JACOBIAN FOR APPROXIMATING VERY SMALL MOVEMENTS

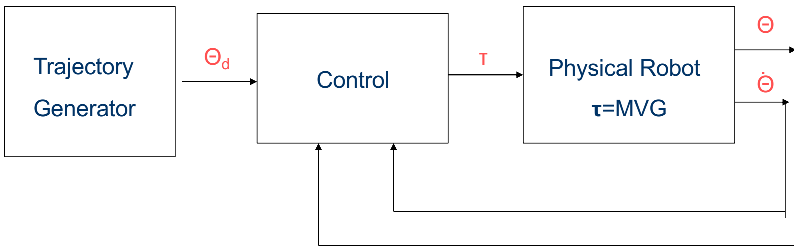
- Taylor: $f(x + \delta x) \approx f(x) + \frac{\partial f}{\partial x} \cdot \delta x \;{-}{>} f(x + \delta x) - f(x) \approx \frac{\partial f}{\partial x} \cdot \delta x$
- δx denotes a small change in x or a small change in the robot parameters
- Now possible: relate small change in δx in joint parameters to a small change $f(x + \delta x) - f(x)$ in the position or other direction: given a desired small step $f(x + \delta x) - f(x)$, we can compute a small change δx in joint parameters that leads to the desired change: $\delta x \approx J^{-1}(f(x + \delta x) - f(x))$

CHANGING A JACOBIAN'S FRAME OF REFERENCE

- Jacobian depends on the choice of reference coordinate system
- Given a Jacobian in {B}: $\begin{bmatrix} {}^{B}v \\ {}^{B}\omega \end{bmatrix} = {}^{B}\nu = {}^{B}J(\Theta)\dot{\Theta}$
- Changing the reference by: ${}^{A}J(\Theta) = \begin{pmatrix} {}^{A}_{B}R & \mathbf{0} \\ \mathbf{0} & {}^{A}_{B}R \end{pmatrix} {}^{B}J(\Theta)$

6. Trajectory Generation

- compute a trajectory (time history of position velocity and acceleration for each degree of freedom) that describe the desired motion of a manipulator
- Trajectory Generator: takes preplanned task (sequence of points) and generates motion profile

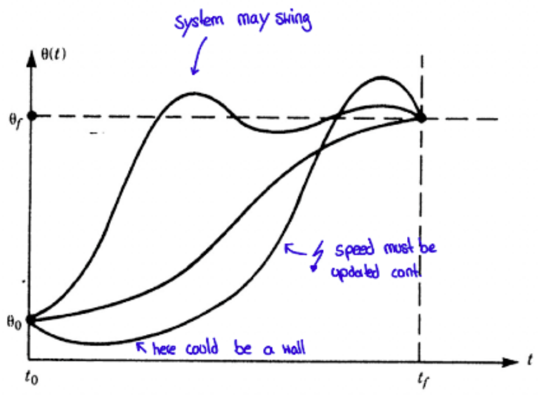


PATH GENERATION

- Smooth function for each n joints must be found that passes trough the points to the goal point: function for each joint whose value at t₀ is the initial position and at t_f its at the desired goal position —> many possible smooth functions

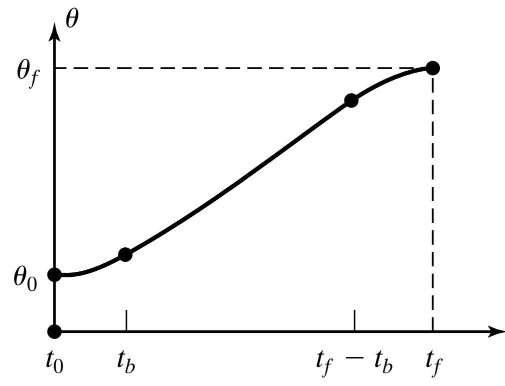
<u>Cubic polynomials</u>

- four constraints for θ(t) are evident:
  $\theta(0)=\theta_0 \quad \theta(t_f)=\theta_f \quad \dot\theta(0)=0 \quad \dot\theta(t_f)=0$
- Need at least a polynomial of 3rd degree (linear is not sufficient because immediate jumps in velocity)
- $-> \theta(t)=a_0+a_1t+a_2t^2+a_3t^3$
- Find coefficients by:
  $a_0=\theta_0 \quad a_1=0 \quad a_2=\frac{3}{t_f^2}\left(\theta_f-\theta_0\right) \quad a_3=-\frac{2}{t_f^3}\left(\theta_f-\theta_0\right)$
- Also possible with desired end and start velocities > 0
- ⚡ System may swing, Speed must be updated continuously, underswinging could hit a wall



Linear function with parabolic blends

- start with linear function but add parabolic blend region to create a smooth path
- Velocity at the end of the blend region bus equal the velocity of the linear section: $\ddot\theta t_b=\frac{\theta_h-\theta_b}{t_h-t_b}$
- $t_b=\frac{t}{2}-\frac{\sqrt{\ddot\theta^2t^2-4\ddot\theta\left(\theta_f-\theta_0\right)}}{2\ddot\theta}$ with the acceleration constraint $\ddot\theta\geq\frac{4\left(\theta_f-\theta_0\right)}{t^2}$ (if equal: first half accelerating, second half slowing down)



- For a general segment connecting points n-1 and n with the desired duration t_d:

$$\ddot\theta_n=SGN\left(\theta_{n-1}-\theta_n\right)\left|\ddot\theta_n\right| \qquad \dot\theta_{(n-1)n}=\frac{\theta_n-\theta_{n-1}}{t_{d(n-1)n}-\frac{1}{2}t_n}$$

$$t_n=t_{d(n-1)n}-\sqrt{t_{d(n-1)n}^2+\frac{2\left(\theta_n-\theta_{n-1}\right)}{\ddot\theta_n}} \qquad t_{(n-1)n}=t_{d(n-1)n}-t_n-\frac{1}{2}t_{n-1}$$

7. Linear control of manipulators

- How to cause the manipulator to actually perform these desired motions
- Control system: compute appropriate actuator commands that will realise desired motion (e.g. by MVG equation), these torques are determined by using feedback from the joint sensors to compute the torque required

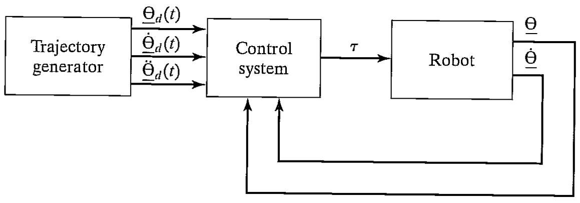


FIGURE 9.1: High-level block diagram of a robot-control system.

Open-loop system —> no use made of feedback from the sensor

SECOND-ORDER LINEAR SYSTEMS

- start with simple systems
- Equation of motion: $m\ddot{x} + b\dot{x} + kx = 0$
- From Laplace Transformation, compute characteristic equation: $ms^2 + bs + k = 0$
- Compute back to time domain by partial fraction expansion $G(s) = \dfrac{1}{(s+2)(s+3)} = \dfrac{K_1}{(s+2)} + \dfrac{K_2}{(s+3)}$
- $g(t) = K_1e^{-2t} + K_2e^{-3t} \quad g^{(0)} \quad g^{(0)} \rightarrow$ compute K1 u. K $_2$
- Roots of characteristic equation (poles): $s_{1,2} = -\dfrac{b}{2m} \pm \dfrac{\sqrt{b^2 - 4mk}}{2m}$
- Can be characterised in terms of
  - Natural frequency: frequency of oscillation without damping $\omega_n = \sqrt{\dfrac{k}{m}} = \sqrt{\dfrac{k + k_p}{m}}$
  - Damping ratio: exponential decay frequency/natural frequency $\zeta = \dfrac{b}{2\sqrt{km}} = \dfrac{b}{2\omega_n m}$
- resonant frequency: components of the robot deform minimally —> resonance: deformations adding up until finally components may be damaged —> $\omega_n \leq \dfrac{1}{2}\omega_{\text{res}}$

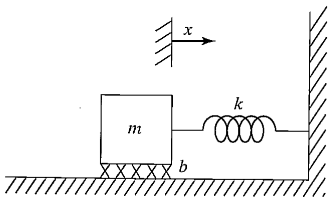
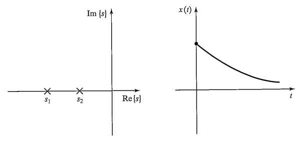
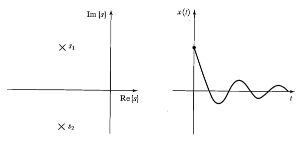
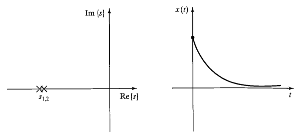


FIGURE 9.2: Spring–mass system with friction.

Real unequal roots: b² > 4mk —> overdamped	• non-oscillatory exponential decay • friction dominates, sluggish behaviour • solution: $x(t) = c_1e^{s_1t} + c_2e^{s_2t}$ (c₁ and c₂ can be computed by given initial conditions) • Much larger pole can be neglected because term will decay to 0 readily in comparison to dominant pole	

Complex roots: b² < 4mk —> underdamped
 	
b = k$_v$
mk = k$_p$
	• stiffness dominates, imaginary component, then the systems oscillates
	• Purely imaginary roots cause the system to oscillate forever
	• Complex roots: $s_{1,2} = \lambda \pm \mu i$
	• Solution: same formula as above but with Eulers formula ($e^{ix} = \cos x + i \sin x$): $x(t) = c_1e^{\lambda t}\cos(\mu t) + c_2e^{\lambda t}\sin(\mu t)$
	• With $\begin{matrix} c_1 = r\cos\delta \\ c_2 = r\sin\delta \end{matrix}$: $x(t) = re^{\lambda t}\cos(\mu t - \delta)$
		

Real equal roots: b² = 4mk —> critically damped	• fiction and stiffness are balanced —> „fastest“ non-oscillatory exponential decay possible in a second order system (desired) • Solution: $x(t) = \left(c_1 + c_2t\right)e^{-\frac{b}{2m}t}$	

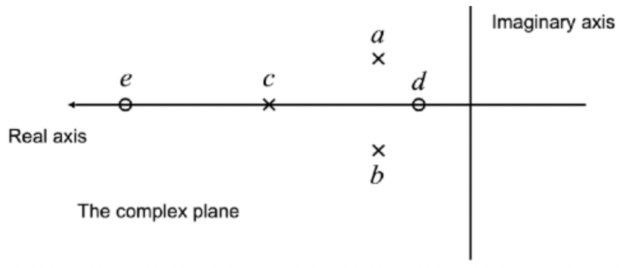
Positive real poles	• positive real poles correspond to non-oscillatory exponential increase, (not BIBO stable)	

POLE-ZERO PLOT

- How to characterise the transient response:

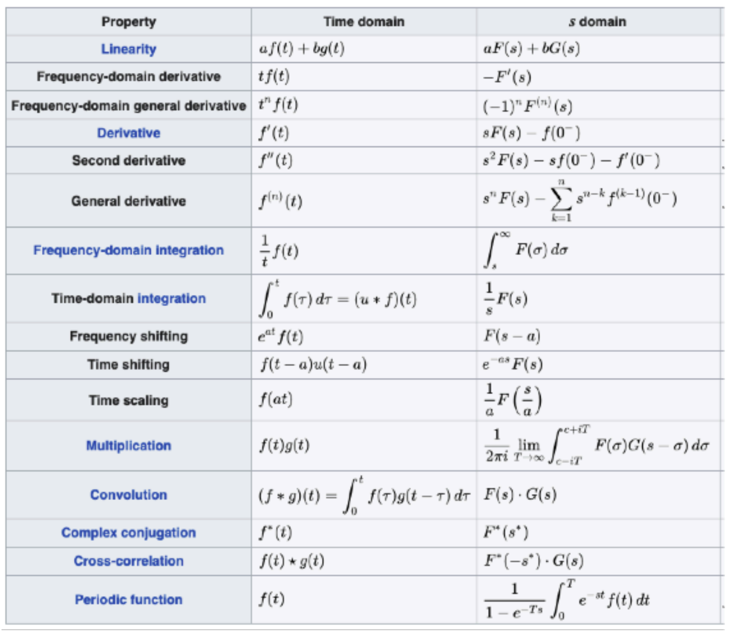
$$G(s) = \frac{(s-d)(s-e)}{(s-a)(s-b)(s-c)}$$

- The poles of G(s) are the roots of the denominator
- The zeros of G(s) are the roots of the numerator



LAPLACE TRANSFORMATION

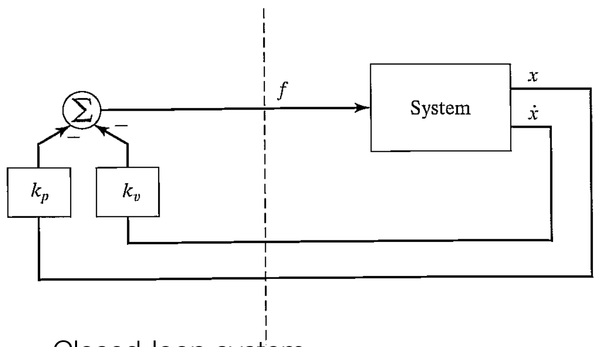
- Forward: $L[f(t)] = F(s) = \int_{0-}^{\infty} f(t)e^{-st}dt$
- Inverse: $L^{-1}[F(s)] = f(t)u(t)$
- $L[u(t)] = \int_{0-}^{\infty} e^{-st}dt = -\frac{1}{s}e^{-st}\Big|_{0-}^{\infty} = \frac{1}{s}$
- $L[tu(t)] = \int_{0-}^{\infty} te^{-st}dt = \frac{1}{s^2}$



Property	Time domain	s domain
Linearity	$af(t) + bg(t)$	$aF(s) + bG(s)$
Frequency-domain derivative	$tf(t)$	$-F'(s)$
Frequency-domain general derivative	$t^n f(t)$	$(-1)^n F^{(n)}(s)$
Derivative	$f'(t)$	$sF(s) - f(0^-)$
Second derivative	$f''(t)$	$s^2F(s) - sf(0^-) - f'(0^-)$
General derivative	$f^{(n)}(t)$	$s^nF(s) - \sum_{k=1}^{n} s^{n-k} f^{(k-1)}(0^-)$
Frequency-domain integration	$\frac{1}{t}f(t)$	$\int_s^{\infty} F(\sigma)\,d\sigma$
Time-domain integration	$\int_0^t f(\tau)\,d\tau = (u * f)(t)$	$\frac{1}{s}F(s)$
Frequency shifting	$e^{at}f(t)$	$F(s-a)$
Time shifting	$f(t-a)u(t-a)$	$e^{-as}F(s)$
Time scaling	$f(at)$	$\frac{1}{a}F\left(\frac{s}{a}\right)$
Multiplication	$f(t)g(t)$	$\frac{1}{2\pi i}\lim_{T\to\infty}\int_{c-iT}^{c+iT} F(\sigma)G(s-\sigma)\,d\sigma$
Convolution	$(f * g)(t) = \int_0^t f(\tau)g(t-\tau)\,d\tau$	$F(s)\cdot G(s)$
Complex conjugation	$f^*(t)$	$F^*(s^*)$
Cross-correlation	$f(t) \star g(t)$	$F^*(-s^*)\cdot G(s)$
Periodic function	$f(t)$	$\frac{1}{1-e^{-Ts}}\int_0^T e^{-st}f(t)\,dt$

CONTROL OF SECOND-ORDER SYSTEMS

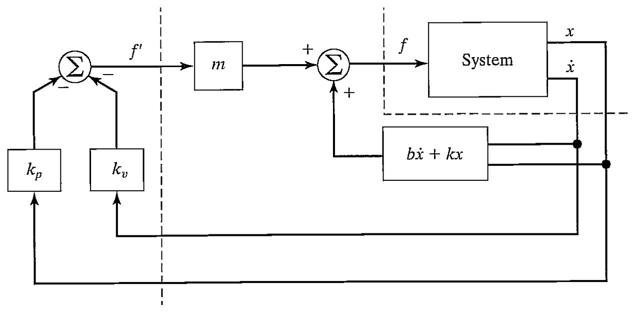
- Natural response is not as desired
- Apply force to the mass: $m\ddot{x} + b\dot{x} + kx = f$
- Control law: compute force that should be applied by the actuator as a function of the feedback $f = -k_p x - k_v \dot{x}$
- Close-loop equation: $m\ddot{x} + b'\dot{x} + k'x = 0$
  where $b' = b + k_v$ and $k' = k + k_p$
- b' and k' always positive, negative would lead to an unstable system
- Control gains k_p and k_v —> obtain every desired behaviour
- We always want a "stiff" system ⇒ $k_p + k$ and $k_v + b$ should be as large as possible



Closed-loop system
Left: Control system (usually in computer)
Right: physical system

CONTROL-LAW PARTITIONING

- Choice of b' depends on m ⚡ very complicated for complex systems —> decouple mass-dependent part from the equation



Closed-loop system
Left: observer part
Middle: Partitioned controller

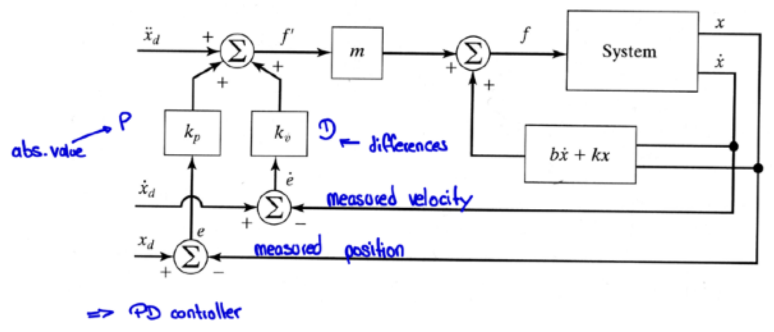
- Partition the controller into a model-based portion (parameters) and a servo portion (independent of parameters)
- Model-based portion: $f = \alpha f' + \beta$ such that f' is taken as the new input to the system, appears as unit mass. With $\begin{matrix} \alpha = m \\ \beta = b\dot{x} + kx \end{matrix}$ $\ddot{x} = f'$
- Now same as open-loop: $f' = -k_v\dot{x} - k_p x \;{-}{>}\; \ddot{x} + k_v\dot{x} + k_p x = 0$
- Setting of control gain for critical damping: $k_v = 2\sqrt{k_p}$

MULTI-DIMENSIONAL SYSTEMS

- Partitioning greatly simplifies control problem for multi-dimensional problems
- x is a multi-dimensional quantity: $M\ddot{x} + \left(B + K_v\right)\dot{x} + \left(K + K_p\right)x = 0$
- Quite difficult to handle —> Determining K$_v$ and K$_p$ to achieve critical damping becomes a major problem
- Use partitioning
  - Set $f = \alpha f' + \beta$ with $\alpha = M$, and $\beta = B\dot{x} + Kx$
  - $M\ddot{x} + B\dot{x} + Kx = Mf' + B\dot{x} + Kx$
  - If M is invertible —> $\ddot{x} = f'$
  - With setting $f' = -K_v\dot{x} - K_p x \;{-}{>}\; \ddot{x} + K_v\dot{x} + K_p x = 0$
  - Choose K$_p$ and K$_v$ as diagonal matrices with entries k$_{pi}$, k$_{vi}$, this becomes a series of decoupled differential equations. Avoid critical damping with $k_{vi} = 2\sqrt{k_{pi}}$
- $$\begin{pmatrix} \ddot{x}_1 \\ \ddot{x}_2 \\ \vdots \\ \ddot{x}_n \end{pmatrix} + \begin{pmatrix} k_{v_1} & 0 & \dots & 0 \\ 0 & k_{v_2} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & k_{v_n} \end{pmatrix}\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \vdots \\ \dot{x}_n \end{pmatrix} + \begin{pmatrix} k_{p_1} & 0 & \dots & 0 \\ 0 & k_{p_2} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & k_{p_n} \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \Leftrightarrow \begin{pmatrix} \ddot{x}_1 \\ \ddot{x}_2 \\ \vdots \\ \ddot{x}_n \end{pmatrix} + \begin{pmatrix} k_{v_1}\dot{x}_1 \\ k_{v_2}\dot{x}_2 \\ \vdots \\ k_{v_n}\dot{x}_n \end{pmatrix} + \begin{pmatrix} k_{p_1}x_1 \\ k_{p_2}x_2 \\ \vdots \\ k_{p_n}x_n \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

TRAJECTORY-FOLLOWING CONTROL

- Rather than maintaining a desired location —> follow a trajectory x$_d$(t)
- Servo error (between actual and desired position): $e = x_d - x$
- Servo control law that will follow trajectory: $f' = \ddot{x}_d + k_v\dot{e} + k_p e = \ddot{x}$ (with unit mass) or $\ddot{e} + k_v\dot{e} + k_p e = 0$ in error space —> chose coefficients so we can design any response we wish



- Error will tend towards 0, and it will approach 0 with a speed depending on k$_v$ and k$_p$ - critical damping will thus provide the fastest possible convergence of the error towards 0.

DISTURBANCE REJECTION

- Minimize errors in presence of external disturbances or noise force f$_{dist}$
- Dynamics: $\ddot{e} + k_v\dot{e} + k_p e = f_{\text{dist}}$
- If f$_{dist}$ is bounded, then e(t) is bounded —> BIBO stable

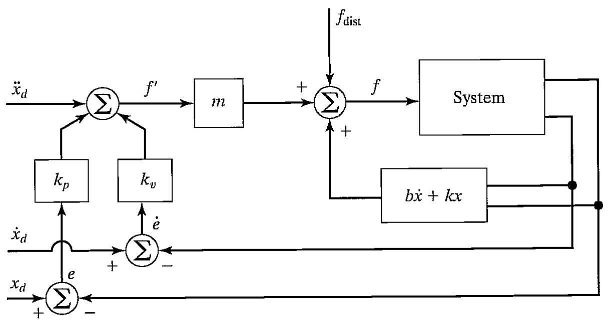


FIGURE 9.10: A trajectory-following control system with a disturbance acting.

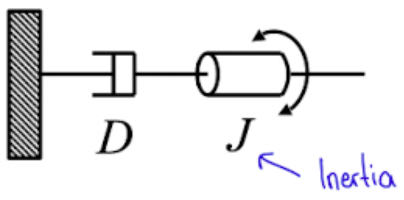
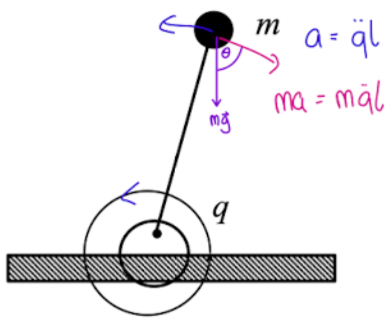
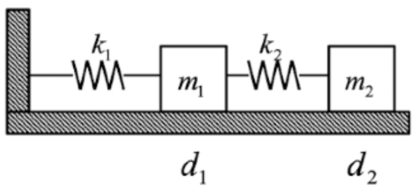
Steady-State error

- f$_{dist}$ is constant —> analyse the system at rest
- Setting derivatives to 0: $e = f_{\text{dist}}/k_p$ (steady-state error)
- To eliminate steady-state error, add integral term to

control law: $\ddot{e} + k_v\dot{e} + k_p e + k_i \int e dt = f_{\text{dist}}$

- —> Third-order system, PID control law

OTHER SYSTEMS

Torsional damped inertia		• Torque due to inertia $\tau = -J\ddot{\theta}$ • Torque due to damping: $\tau = -D\dot{\theta}$ • Torque applied from outside: $\tau = J\ddot{\theta} + D\dot{\theta}$
Frictionless joint		$0 = -ml^2\ddot{q} - mgl\cos(q)$
Coupled springs		$0 = -m_1\ddot{x}_1 - d_1\dot{x}_1 - k_1x_1 + k_2\left(x_2 - x_1\right)$ $0 = -m_2\ddot{x}_2 - d_2\dot{x}_2 + k_2\left(x_1 - x_2\right)$

TORQUE CONTROL

- Underlying physical phenomenon that causes a motor to generate a torque when current passes through the windings: $F = qV \times B$, where charge q moving with velocity V trough a magnetic field B, experiences a force F
- Torque-producing ability of a motor is stated by a motor torque contestant, which relates armature current to the output torque: $\tau_m = k_m i_a$
- Circuit: $l_a \dot{i}_a + r_a i_a = v_a - k_e \dot{\theta}_m$
  - v_a: voltage source
  - l_a: inductance of the armature windings
  - r_a: resistance of the armature windings
  - v: generated back electromotive force
  - Sense the current trough the armature and continuously adjust the voltage source v_a so that a desired current i_a flows through the armature

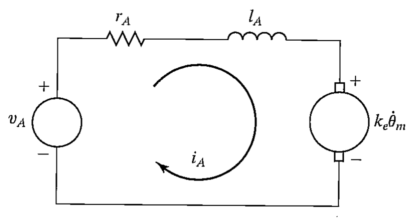


FIGURE 9.11: The armature circuit of a DC torque motor.

EFFECTIVE INERTIA

- The gear ration η causes an increase in the toque seen at the load and a reduction in the speed of the load given by: $\begin{aligned}\tau &= \eta\tau_m \\ \dot{\theta} &= (1/\eta)\dot{\theta}_m\end{aligned}$
- Torque balance: $\tau_m = I_m\ddot{\theta}_m + b_m\dot{\theta}_m + (1/\eta)(I\ddot{\theta} + b\dot{\theta})$
  - I_m and I are the inertias of the motor rotor and the load
  - b_m and b are the viscous friction coefficients for the rotor and load bearings
- $\tau_m = \left(I_m + \frac{I}{\eta^2}\right)\ddot{\theta}_m + \left(b_m + \frac{b}{\eta^2}\right)\dot{\theta}_m$ OR $\tau = \left(I + \eta^2 I_m\right)\ddot{\theta} + \left(b + \eta^2 b_m\right)\dot{\theta}$

- $I + \eta^2 I_m$ is called the effective inertia seen at the output of the gearing
- $b + \eta^2 b_m$ is called effective damping

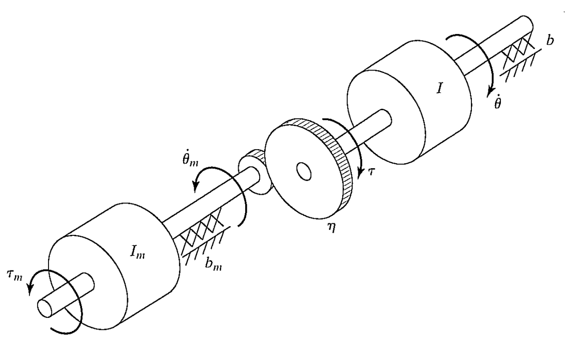


FIGURE 9.12: Mechanical model of a DC torque motor connected through gearing to an inertial load.

8. Nonlinear control of manipulators

- Consider non-linear spring: f = qx³ —> Construct a control law to keep the system critically damped with a stiffness of k_{CL}
- Open-loop equation: $m\ddot{x} + b\dot{x} + qx^3 = f$
- With $f = \alpha f' + \beta, \alpha = m$ and $\beta = b\dot{x} + qx^3 \rightarrow f' = \ddot{x}_d + k_v\dot{e} + k_p e$

CONTROL PROBLEM FOR MANIPULATORS

- Rigid-body dynamics and add friction with term F:
  $\tau = M(\Theta)\ddot{\Theta} + V(\Theta, \dot{\Theta}) + G(\Theta) + F(\Theta, \dot{\Theta})$
- Handle problem with partitioned controller: $\tau = \alpha\tau' + \beta$ where tau is a nx1 vector if joint torques
- Choose $\alpha = M(\Theta)$, $\beta = V(\Theta, \dot{\Theta}) + G(\Theta) + F(\Theta, \dot{\Theta})$ with the servo law $\tau' = \ddot{\Theta}_d + K_v\dot{E} + K_p E$ where $E = \Theta_d - \Theta$
- Error equation: $\ddot{E} + K_v\dot{E} + K_p E = 0$
- Equation is decoupled (Kv and Kp are diagonal) so that it can be written on a joint-by-joint basis $\ddot{e}_i + k_{vi}\dot{e} + k_{pi}e = 0$ where $\omega_{ni} = \sqrt{k_{pi}}$ and $k_{vi} = 2\sqrt{k_{pi}}$ holds

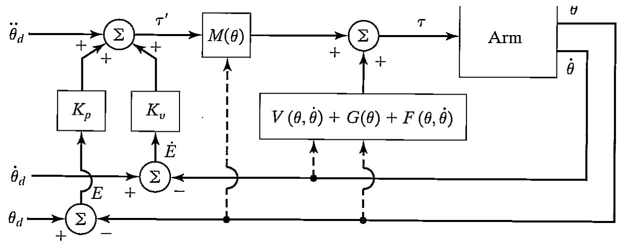


FIGURE 10.5: A model-based manipulator-control system.

CARTESIAN-BASED CONTROL SYSTEMS

- In case of a strictly Cartesian manipulator: $J^T = J^{-1}$
- Both schemes will work, but not well

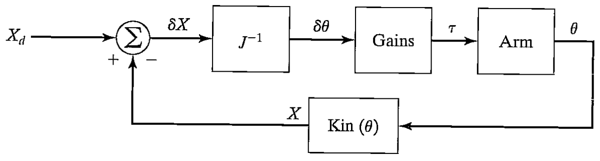


FIGURE 10.12: The inverse-Jacobian Cartesian-control scheme.

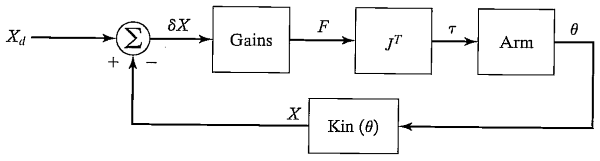


FIGURE 10.13: The transpose-Jacobian Cartesian-control scheme.

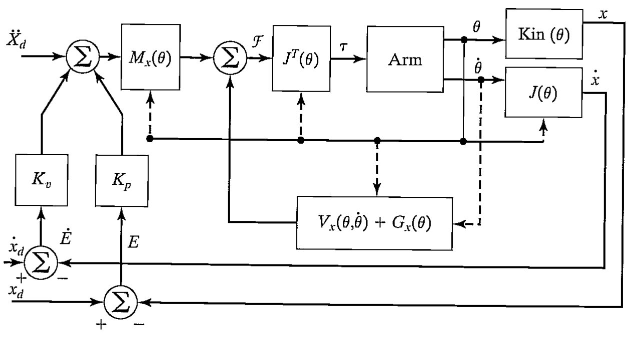


FIGURE 10.14: The Cartesian model-based control scheme.

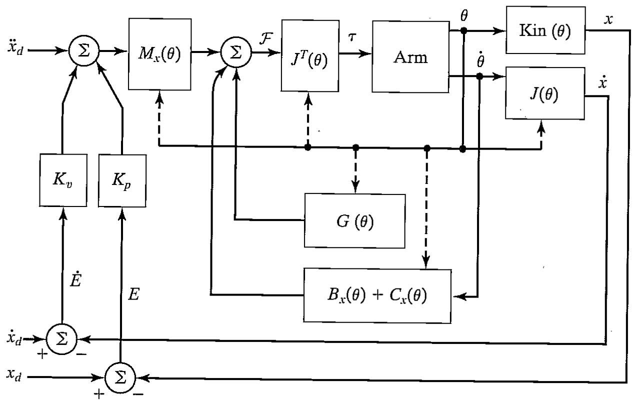


FIGURE 10.15: An implementation of the Cartesian model-based control scheme.

9. Force control of manipulators

- Specify a force that should be maintained instead of a position (e.g. pressing a sponge on a window)

FORCE CONTROL OF A MASS-SPRING SYSTEM

- model contact with environment as a spring, system is rigid and environment has some stiffness k_e (very high for surfaces)
- Force acting in the spring: $f_e = k_e x$
- Physical system: $f = m\ddot{x} + k_e x + f_{\text{dist}} = m k_e^{-1} \ddot{f}_e + f_e + f_{\text{dist}}$
- With $\alpha = m k_e^{-1}$ and $\beta = f_e + f_{\text{dist}}$ —> control law: $f = m k_e^{-1}\left[\ddot{f}_d + k_{vf}\dot{e}_f + k_{pf} e_f\right] + f_e + f_{\text{dist}}$
- Closed-loop system: $\ddot{e}_f + k_{vf}\dot{e}_f + k_{pf} e_f = 0$ ⚡ Don't know f_{dist}, so leave term out and do steady state analysis
- Effective force-feedback gain: $e_f = \dfrac{f_{\text{dist}}}{\alpha}$
- Use f_d instead of $f_e + f_{\text{dist}}$ in the control law: $e_f = \dfrac{f_{\text{dist}}}{1+\alpha}$. As α is small the second error is better, and the control law should be $f = m k_e^{-1}\left[\ddot{f}_d + k_{vf}\dot{e}_f + k_{pf} e_f\right] + f_d$
- ⚡ $\dot{f}_d$ and $\ddot{f}_d$ inputs are often set to 0, sensed forces are quite noisy —> numerical differentiation even more noisy: obtain derivative of force by derivative of x —> control law: $f = m\left[k_{pf} k_e^{-1} e_f - k_{uf}\dot{x}\right] + f_d$

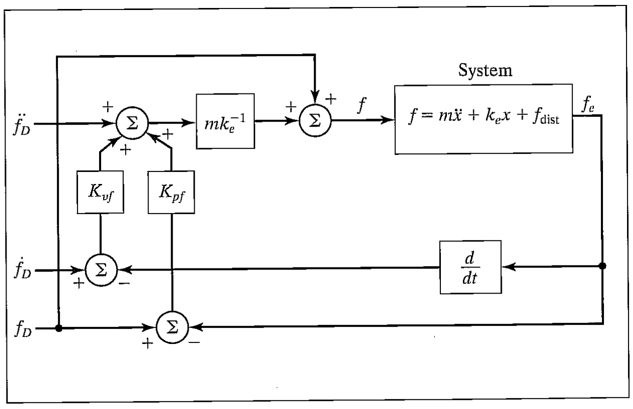


FIGURE 11.6: A force control system for the spring–mass system.

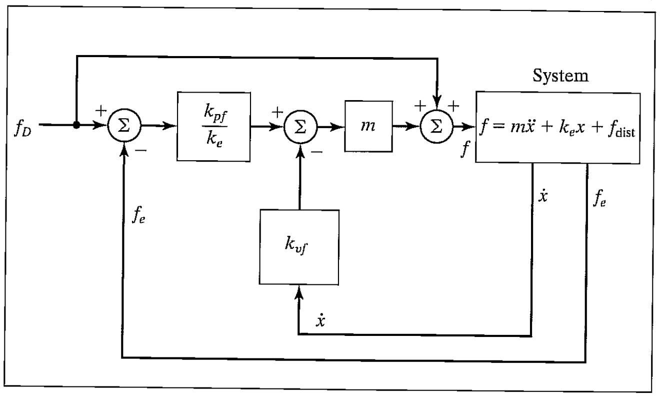


FIGURE 11.7: A practical force-control system for the spring–mass system.

HYBRID POSITION/FORCE CONTROLLER

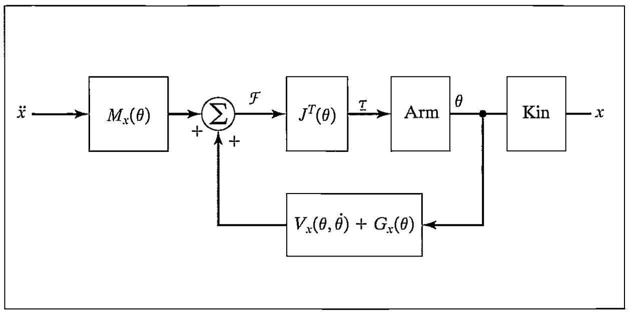


FIGURE 11.11: The Cartesian decoupling scheme introduced in Chapter 10.

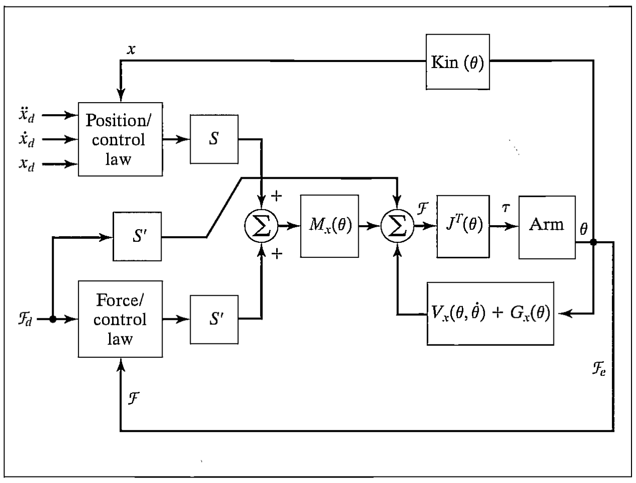


FIGURE 11.12: The hybrid position/force controller for a general manipulator. For simplicity, the velocity-feedback loop has not been shown.

10. Task requirements

- Robots usually don't fit the ideal of „universally programmable“ devices
- DOF should match the task —> minimises cost (hardware, computing power and power consumption), minimises size/weight
- Examples
  - Haptic devices: 3DOF actuation, torque not important for many virtual environment, however 6DOF positioning is important
  - Circuit Assembly: Placement of components on a circuit board (4DOF, x, y, z, rotation)

- Other task requirements:
  - Workspace: scale and precise shape
  - Load capacity: sizing of structural members
  - Speed
  - Accuracy and precision

- Kinematic configurations
  - Decide DOF first
  - Then close kinematic configuration to obtain the best workspace, dynamic properties, use of actuators and sensor, accuracy
  - A general, 6DOF manipulator is usually classified by the first 3 DOF plus a wrist

- Workspace attributes
  - Design efficiency: How much material is need to build different designs with the same workspace?
  - Length sum: $L = \sum_{i=1}^{N} \left(a_{i-1} + d_i\right)$
  - Structural length index: $Q_L = \frac{L}{\sqrt[3]{W}}$

- Condition of workspace
  - When the manipulator is near a singular point, actions of the manipulator are said to be poorly conditioned
  - Singular conditions are given by det(J) = 0 —> Use Jacobian as a measure of manipulator dexterity

- Manipulability Measure (vel)
  - Defined as $w = \sqrt{\det\left(J(\theta)J^T(\theta)\right)}$
  - For a non-redundant manipulator $w = |\det(J(\theta))|$
  - A good manipulator has a high w over large area of its workspace

- Redundant structures
  - can be useful for avoiding collisions while operating in cluttered work environments
  - Gears produce a large reduction in a compact configuration ⚡ backlash and friction
  - Gear ration: relationship between input and output speeds and torques $\eta > 1$ $\dot{\theta}_o = \frac{\dot{\theta}_i}{\eta}$ $\tau_o = \eta \tau_i$

- Actuator types
  - Electric motors: DC, brushed, permanent magnetic
  - Pneumatic actuator

- Potentiometers
  - produce a voltage proportional to shaft position
  - voltage divider

- ⚡ friction, noise, resolution linearity

- Optical encoder
  - focused beam of light aimed at a matched photodetector is interrupted periodically by a coded pattern on a disk
  - produces a number of pulses per revolution (lots of pulses = high cost)
  - Quantisation problems a low speeds