Static and Dynamic Stability

Stability

- Stability is a necessary property of mobile robots
- Stability can be
  - static (standing w/o falling over)
  - dynamic (moving w/o falling over)

Static stability is achieved through the mechanical design of the robot

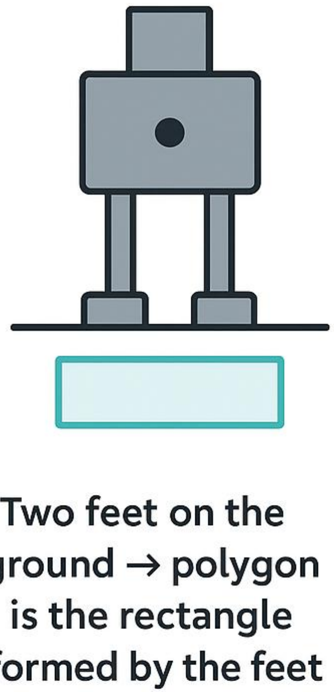
Dynamic stability is achieved through control mechanism

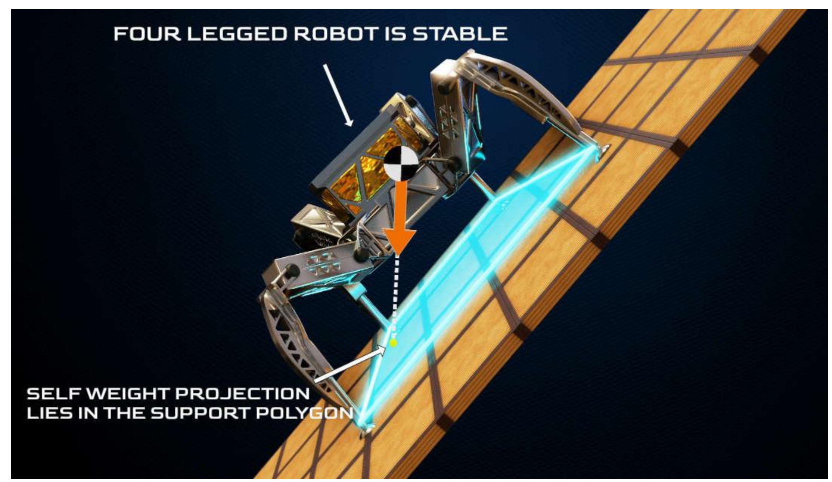
Static Stability

- Static stability refers to a robot’s ability to remain balanced when it is at rest, i.e., not moving
- A robot is statically stable if it can resist tipping over under gravity or external forces without needing to move its joints.
- A robot is statically stable if its Center of Gravity (CoG) projects within its support polygon (area enclosed by its feet or contact points with the ground).

Support Polygon

- Defined by contact points between the robot and the ground.
- Support polygon = convex polygon formed by the robot’s contact points (feet, wheels, legs).
- If the robot has:
  - Two feet on the ground → polygon is the rectangle formed by the feet.
  - Three legs (tripod robot) → polygon is the triangle formed by leg tips.
  - Four legs (quadruped) → polygon is the convex polygon connecting all four feet.





- 2 feet → rectangle (line segment in 2D).
- 3 legs → triangle.
- 4 legs → quadrilateral.
- 6 legs (hexapod) → hexagon.

Stability Criterion

- **Center of Mass (CoM)**:

  A robot is statically stable if it does not tip over when stationary.

  Requirement: The projection of the CoM onto the ground must lie inside the support polygon (the area covered by the robot's feet).

Role of Each Coordinate

- **x, y (ground plane):**
- Decide whether the projection falls inside or outside the support polygon.
- If outside → robot tips.
- **z (height):**
- Determines *how easy or hard* it is for the CoM to move outside when the robot tilts.
- Higher z → less stable, lower z → more stable.

Mathematical Criterion

Let the projection of the CoG on the ground plane = (x_{CoG}, y_{CoG}).

The robot is **statically stable** if:

$$(x_{CoG}, y_{CoG}) \in \text{Support Polygon}$$

- **Inside polygon → Stable**
- **On boundary → Marginally stable (about to tip)**
- **Outside polygon → Unstable (will tip over)**

•CoG (Center of Gravity) is essentially CoM under gravity.
•On Earth, with normal gravity, CoG height ≈ CoM height.

Dynamic Stability

Dynamic stability considers motion (walking, running, pushing).

- A robot is dynamically stable if it does not fall while moving.
- Requires control of momentum, forces, and timing.
- Example: Walking robots use 'Zero Moment Point (ZMP)' criterion.

Center of Mass (CoM)

The point where the robot’s mass is “concentrated” or balanced.

- Why it matters:
  - The position of the CoM relative to the feet determines whether the robot is stable.
  - If CoM projection (on the ground) is within the support polygon → robot resists tipping.
  - In dynamic walking, the CoM moves, so stability depends on how we control its trajectory.

Example:

- Think of a humanoid robot leaning forward to walk: the CoM moves ahead of the support foot. To prevent falling, the next foot placement must compensate.

Ground Reaction Forces (GRF)

- Whenever the robot’s foot touches the ground, the ground exerts an equal and opposite force: this is the GRF.
- Components of GRF:
  - Vertical (Fz): supports weight
  - Forward/backward (Fx): accelerates/decelerates robot
  - Sideways (Fy): maintains lateral balance
- GRF produces moments (torques) around the CoM and feet.
- Dynamic stability requires that the resulting moment does not tip the robot, which is why we track ZMP.

Example:

- Walking on a slope: GRF is not vertical → robot must adjust posture to keep ZMP inside support area.

Inertia and Accelerations

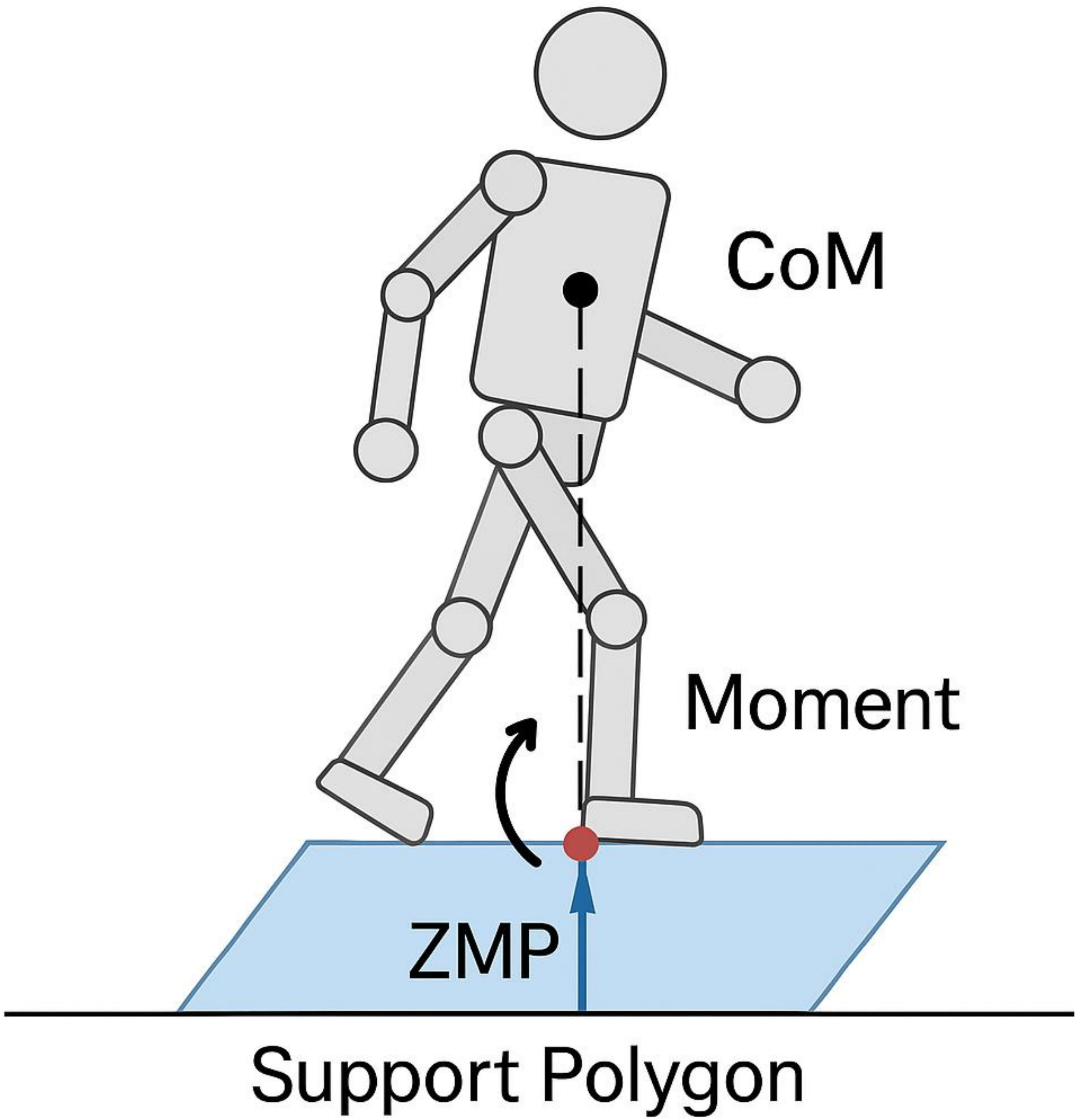
- Inertia: resistance of a robot’s mass to change in motion.
- When robot moves, CoM accelerates → creates dynamic forces.
- These forces produce extra torques, which can tip the robot if not controlled.
- Dynamic stability accounts for these inertial forces, unlike static stability.
- Example:
- While a robot jumps or runs, it may momentarily be “off balance,” but inertia can be managed via controlled leg motion to recover.

Zero Moment Point (ZMP) Criterion

- The point on the ground where the total effect of gravity + inertia forces has no tipping moment.
- Rule:
  If ZMP lies inside the support polygon → stable.
  If ZMP moves outside → unstable (robot tips).

Zero Moment Point (ZMP)

- ZMP: point on ground where net moment = 0.
- Dynamic stability criterion: ZMP must remain inside support polygon.
- ZMP shifts due to accelerations → control system adjusts foot placement or ankle torque.


CoM
Moment
ZMP
Support Polygon

Momentum control

Momentum = mass × velocity.

While moving, the robot has linear and angular momentum.

The robot must adjust its momentum (e.g., by swinging arms or shifting legs) to prevent tipping.

When a humanoid robot walks or runs:

- Each leg swing generates angular momentum about the robot’s center of mass (CoM).
- Without compensation, this causes torques on the torso, making it tilt or rotate, which can lead to loss of balance.
- In robotics, we can actively control arm joints to swing opposite to the legs:
- Right leg forward → left arm forward, right arm backward (or some variation depending on robot design).
- The angular momentum of the arms cancels the angular momentum of the legs.

Force control

- Ground Reaction Forces (GRF) are the forces the ground exerts on the robot's feet.
- Controlling these forces through joint torques and foot placement helps maintain balance.
- Example: If a robot starts to tip forward, it can push harder with the trailing foot to generate corrective force.

Scenario

- A humanoid robot is walking or standing, and its CoM projection moves toward the edge of the support polygon (e.g., tipping forward).
- If nothing is done, it will fall.

- Trailing foot push:
  - The robot can apply more ground reaction force (GRF) with the foot that is still on the ground (trailing foot).
  - This creates a corrective torque that rotates the robot backward, resisting the fall.
- Adjust foot placement:
  - The robot can move the next foot forward to widen the support polygon in the direction of tipping.
  - This helps catch the CoM projection, restoring stability.
- Body adjustments:
  - Arms or torso can swing or rotate to shift angular momentum, helping balance.

Timing control

- Timing control is about coordinating the robot’s movements in time so that corrective actions happen exactly when needed.
- In dynamic stability, the robot is constantly moving and tipping slightly.
- Sensors detect these deviations, but corrective actions must be applied at precise moments.
- If timing is too early or too late, the robot may overcompensate or fall.

Timing control

Foot placement timing

- When a humanoid starts to tip forward, the robot must lift and place the next foot quickly enough to catch its CoM projection.
- Too slow → robot falls forward.
- Too fast → inefficient or unstable gait.

Joint actuation timing

- Swinging arms to counter leg momentum must happen synchronized with the leg swing.
- Mismatch → torso twists or balance is lost.

Push-off timing

- Trailing foot push (to generate corrective force) must happen exactly when tipping begins.
- Early or late push → ineffective torque correction.

How it works together

- Dynamic stability is basically controlled falling:
- The robot’s CoM moves outside the support area.
- Sensors detect this movement.
- The controller calculates necessary forces, torques, and timing.
- The robot adjusts its joints and foot placement to recover.
- Think of it like how humans run: we are almost always tipping forward, but we catch ourselves with each step.

Dynamic Stability

Walking pattern generation

- The robot's gait is planned so that the calculated ZMP always lies within the support polygon during motion.
- If it goes outside → robot tips over.

Feedback control

- Sensors measure forces at the feet (force-torque sensors).
- Real-time ZMP is computed.
- The robot adjusts joint torques/posture so that ZMP shifts back inside the support area.

Relation with CoM

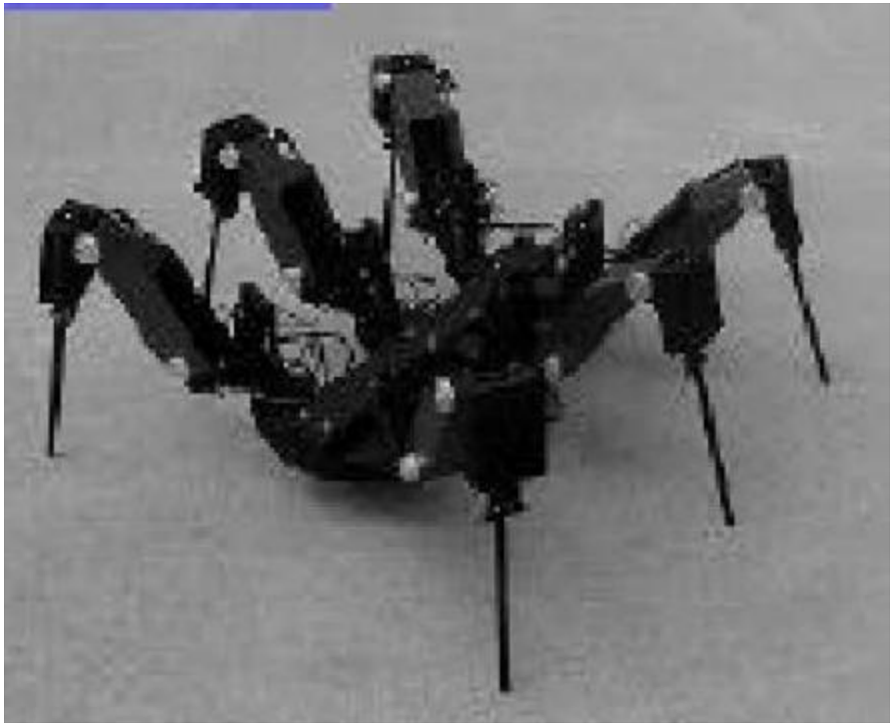
- For slow/static walking: keeping the CoM projection inside support polygon is enough.
- For fast/dynamic walking: ZMP must be considered, since inertia and accelerations affect stability.

Difference Between Static and Dynamic

- Static Stability: Balance while standing still.
  - Condition: CoG projection $\in$ support polygon.
- Dynamic Stability: Balance while moving.
  - Condition: ZMP $\in$ support polygon.

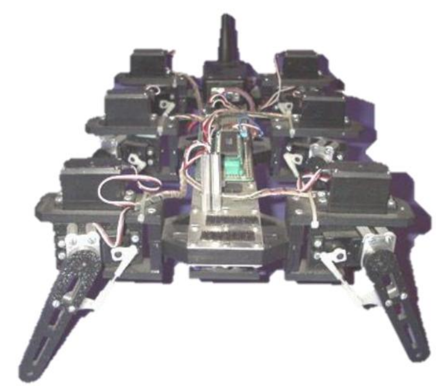
Hexapod and Tripod Gaits

- Hexapod: six-legged robot (like an insect).
- Tripod gait: three legs move at a time while the other three support the body.
- Example: legs 1-3-5 move together, legs 2-4-6 stay on the ground.
- This alternating tripod gait allows the robot to move continuously without fully losing support.
- Unlike static stability, where Center of Mass (CoM) must always be inside the support polygon, dynamic stability allows temporary “tip” as long as motion is controlled.
- Key principle: controlled tipping and recovery.



Hexapod

The Tripod Gait



Wheels vs Legs Stability

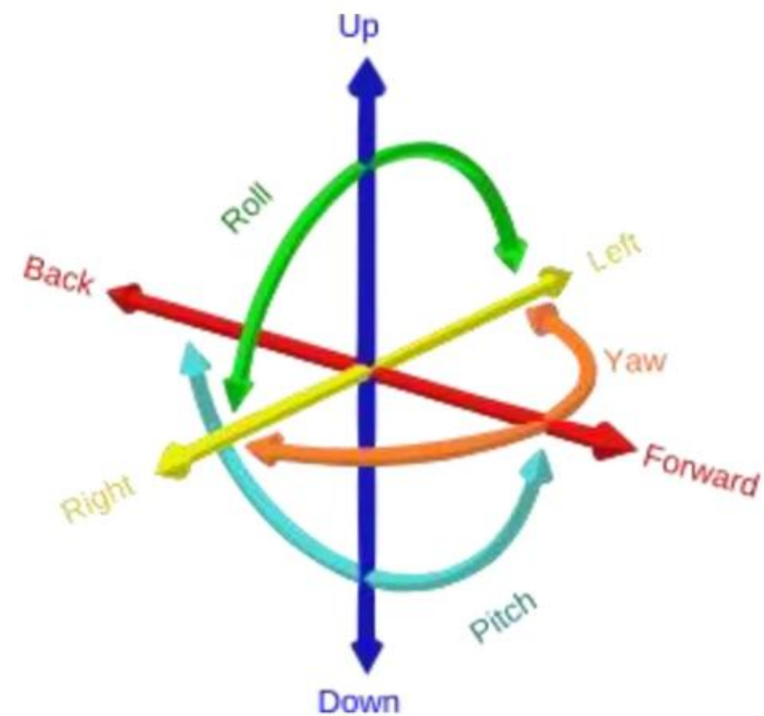
- Legs: stability depends on support polygon; need ZMP & foot placement during motion.
- Wheels: 3+ wheels usually statically stable; 2 wheels (bike/Segway) need dynamic balance.
- Dynamic effects in wheels: tipping in high-speed turns (depends on track width & zCoG).
- Cars → wide base, low CoG → stable. Bicycles → narrow base → need motion for balance.

Comparison Table: Legs vs Wheels

- Legs → adaptable to terrain, can achieve both static & dynamic stability.
- Wheels → efficient, more stable at rest (3+ wheels).
- Legs require advanced control (ZMP, gaits).
- Wheels require active control in 2-wheeled systems or during fast maneuvers.

Degree of Freedom

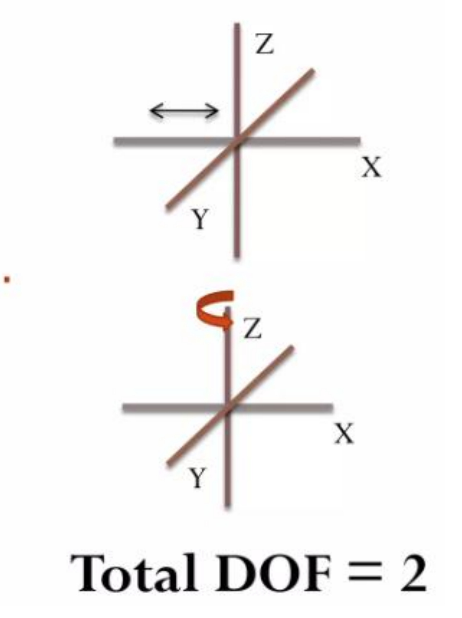
- The degrees of freedom (DOF) of a rigid body is defined as the number of independent movements it has.
- To determine DoF of a rigid body, we must consider how many distinct ways it can be moved.
- DoF is needed to uniquely define position of a system in space at any instant of time.



The six degrees of freedom: forward/back, up/down, left/right, yaw, pitch, roll

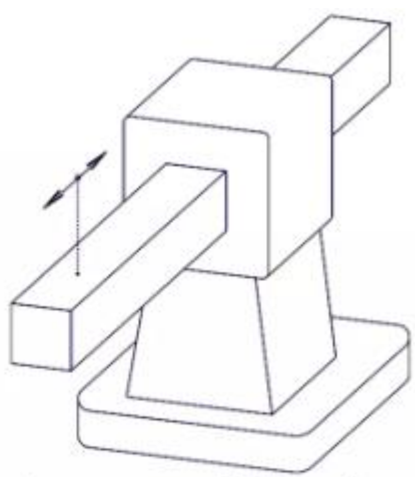
DOF in rigid bodies

- In 3D: 6 DOF (3 translations + 3 rotations).
- In 2D: 3 DOF (x, y + rotation).

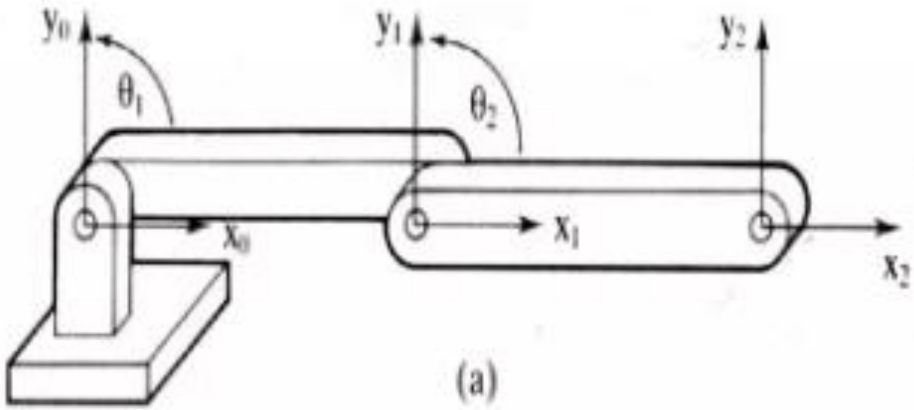


DOF in Mechanisms/Robots

- Each joint adds a DOF.
- **Prismatic joint** → 1 DOF (sliding).
- **Revolute joint** → 1 DOF (rotation).



Example: A 2-link planar arm has 2 DOF



- **Base joint (shoulder)**: Revolute joint → angle θ_1.
- **Second joint (elbow)**: Revolute joint → angle θ_2.
- Each revolute joint in a plane = **1 DOF**.

Counting DOF

- Joint 1 (revolute) → 1 DOF
- Joint 2 (revolute) → 1 DOF

$$\text{Total DOF} = 2$$

Calculating DOF in Mechanisms

Gruebler's/Kutzbach's formula

$$DOF = 3(n-1) - 2j - h \quad \text{(in 2D)}$$

$$DOF = 6(n-1) - 5j - h \quad \text{(in 3D)}$$

n = number of links
j = joints
h = higher pairs - Usually allows rolling or sliding motion such as
  Gear tooth contact (line contact along teeth).
  Ball bearing (point contact)

slider-crank

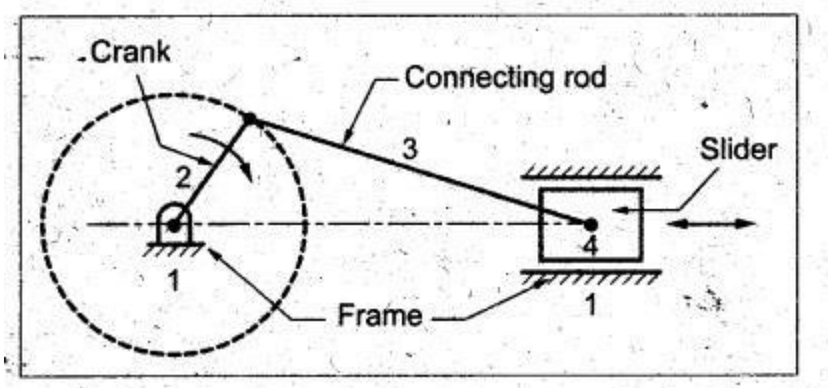


Fig. 1.33. Single slider-crank chain

In a slider-crank, the crank, connecting rod, slider, and frame are all links.

The hinge (revolute) joints and slider (prismatic) joint are joints.
4 links (ground, crank, connecting rod, slider).
4 joints (3 hinges + 1 slider).

$$DOF = 3(n-1) - 2j - h$$

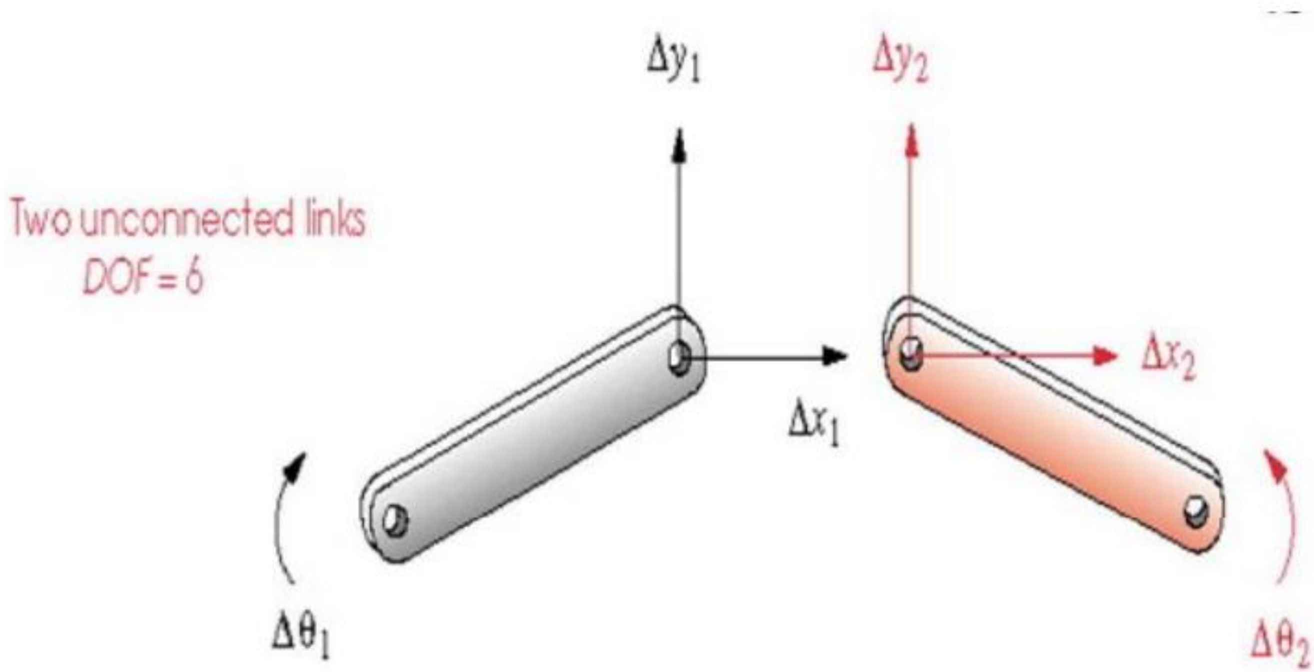
where:

- n = number of links = 4
- j = number of joints = 4
- h = higher pairs (cam, gear, etc.) = 0

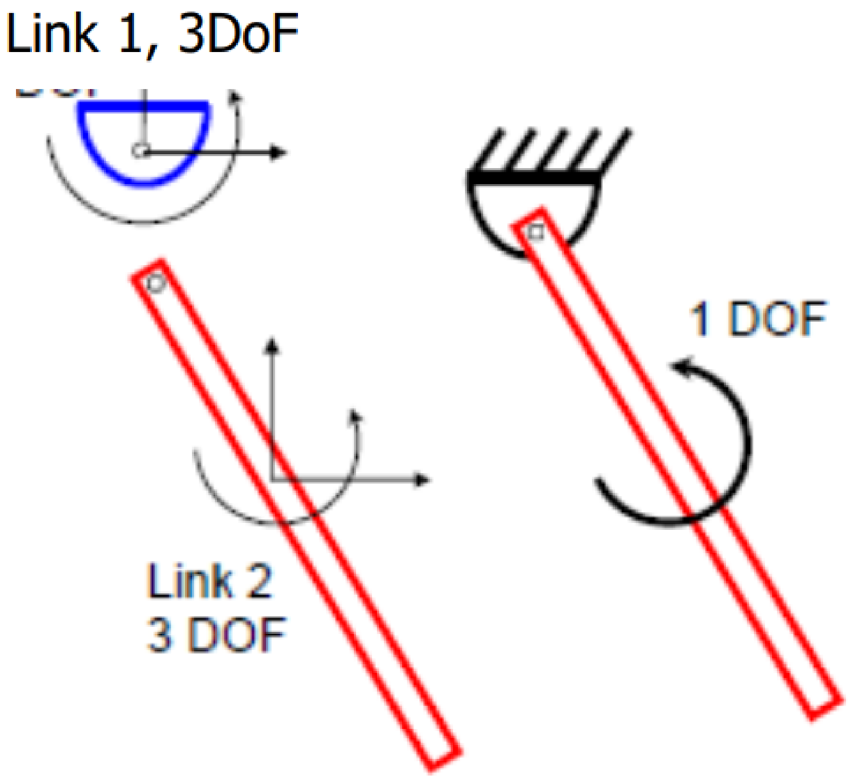
$$DOF = 3(4-1) - 2(4) - 0$$

$$DOF = 9 - 8 = 1$$

DOF



- Therefore, for two unconnected links: 6 DoF (each link has 3 DoF)



- Kutzbach's/Gruebler's equation for planar mechanisms:

$$m = 3(L\text{-}1)\text{-}2j_1$$

where, L: number of links, (=2); and j_1: number of full joints (=1).

- Therefore, this mechanism has: 1 DoF

Holonomic Robots

- The robot can control all the possible independent movements it theoretically has.
- Controllable DOF = System DOF.
- Example:
- Drone (quadcopter):
  - In 3D space → 6 DOF (x, y, z + roll, pitch, yaw).
  - Drone can move independently in all 6 directions → fully holonomic.
- 6-axis industrial robotic arm:
  - Has 6 joints (6 DOF).
  - Can position its end effector anywhere in 3D space with full orientation.
- Holonomic robots = maximum freedom, maximum control.

Non-Holonomic Robots

- The robot cannot control all DOF directly.
- Controllable DOF < System DOF.
- Robot may need a sequence of movements to reach a position it cannot move to directly.
- Example:
- Car:
  - A car on flat ground has 3 DOF (x, y position + orientation angle).
  - But you can only control 2 DOF: forward/backward speed + steering angle.
  - You cannot move sideways directly → need to steer and maneuver.
- Differential drive robot (two wheels):
  - Can move forward/backward and rotate,
  - But cannot directly move sideways.
- Non-holonomic = restricted movements.

Redundant Holonomic Robots

- The robot has more controllable DOF than needed for a task.
- Extra DOF = flexibility, better obstacle avoidance, multiple ways to reach the same target.
- Example:
- Human Arm:
  - Needs only 6 DOF to position and orient the hand in space.
  - But the human arm has 7 DOF (shoulder 3, elbow 1, wrist 3).
  - That extra DOF makes the arm redundant.
- 7-axis robotic arm:
  - Can reach the same target in multiple postures.
- Redundant holonomic robots = more options, more adaptability.

Type	Relation between DOF	Example	Key Idea
Holonomic	Controllable DOF = System DOF	Drone (6 DOF), 6-axis robot arm	Can move in all possible ways
Non-Holonomic	Controllable DOF < System DOF	Car, differential drive robot	Some motions not possible directly
Redundant Holonomic	Controllable DOF > Task DOF	Human arm, 7-axis robot arm	Extra DOF → flexibility