Session 8: Stability and Dynamics

Part A- Newton Euler Dynamics (Core)
Part B: Recursive Newton Euler Dynamics
Part c: Whole Body Dynamics

Dr. Arshiya Sood

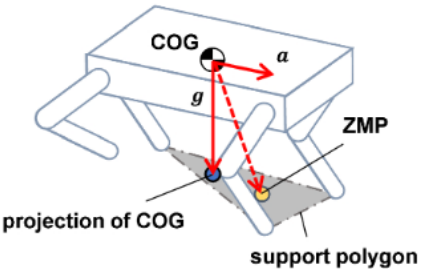
June 12, 2025

Table of Contents

Session 7 Recap & Reflect: What Did We Learn?

Support Polygon (SP):

- 2D area formed by projecting all ground contact points onto horizontal plane.
- Defines robot's mechanical base of support.
- Stability condition: CoG projection inside SP $\rightarrow$ Stable; outside SP $\rightarrow$ Tipping risk.
- Examples:
  - 2-legged: line segment.
  - 3-legged: triangle.
  - 4-legged: quadrilateral.
  - 6-legged: polygon with higher stability.



Session 7 Recap & Reflect (contd.)

Static vs Dynamic Stability

Static Stability:

- Maintains balance without motion or active control.
- Purely geometric and gravity-based.

Dynamic Stability:

- Actively maintained during movement.
- CoG may temporarily leave SP; balance restored via sensors, feedback, control.

Trade-off:

- Static: Simple but limits mobility.
- Dynamic: Enables agile motions but requires real-time control.

Session 7 Recap & Reflect (contd.)

Legged Locomotion: Stability Challenges

- **2-legged robots:** SP is a line; highly unstable without active control.
- **3-legged robots:** Triangular SP allows partial static stability.
- **4-legged robots:** Statically stable walking possible with careful gait sequencing.
- **6-legged robots:**
  - Use tripod gait (3 legs always supporting).
  - Highly stable even on uneven terrain.
- Humans: Rely on complex neuromuscular feedback for dynamic stability.

Session 7 Recap & Reflec (contd.)

Why Zero Moment Point (ZMP) is critical?

- Humanoids must control CoM, foot placement, and hand motion simultaneously.
- Ground contact point is unactuated $\rightarrow$ balance loss possible.
- **ZMP**: Provides dynamic stability criterion.
- Prevents rotational tipping during motion.
- Ensures foot maintains full contact with ground.

ZMP: Definition and Interpretation

Zero Moment Point (ZMP):

- Point where sum of horizontal moments due to ground reaction forces equals zero:

$$\sum M_x = 0, \quad \sum M_y = 0$$

- If ZMP lies inside SP $\rightarrow$ stable contact.
- Reflects combined effects of: Gravity, Inertial forces (accelerations) and External disturbances

CoM vs ZMP: Key Differences

	CoM	ZMP
Static	CoM projection $\approx$ ZMP	CoM projection $\approx$ ZMP
Dynamic	CoM may leave SP	ZMP must stay inside SP
Role	Mass distribution	Dynamic balance indicator
Control	Not directly used	Key for stability control

ZMP Equation: Control Formulation

ZMP location depends on CoM position, accelerations, and moments:

$$x_{ZMP} = x_{CM} - \frac{z_{CM}}{Mg}\left(\ddot{x}_{CM} + \frac{M_y}{Mz_{CM}}\right)$$

$$y_{ZMP} = y_{CM} - \frac{z_{CM}}{Mg}\left(\ddot{y}_{CM} - \frac{M_x}{Mz_{CM}}\right)$$

Where:

- x_{CM}, y_{CM}, z_{CM} : CoM positions
- $\ddot{x}_{CM}, \ddot{y}_{CM}$: CoM accelerations
- M_x, M_y : external moments about CoM
- M : robot mass
- g : gravitational acceleration

Recap Quiz

Quiz Questions (1/3)

Q1. A mobile robot carries a box while moving slowly. As the load shifts slightly forward, the projection of CoG moves closer to the front edge of the support polygon. What happens to stability margin?

A. Stability margin decreases because CoG projection approaches polygon boundary.
B. Stability margin increases because total mass increased.
C. Stability margin remains unchanged since polygon size is constant.
D. Stability margin depends only on robot speed, not CoG position.

Q2. In a 6-legged robot using tripod gait, which factor primarily maintains dynamic stability during leg swing?

A. Keeping CoG projection inside triangle formed by stance legs.
B. Equal leg lengths on both sides.
C. Synchronizing leg movement speeds.
D. The number of legs in contact over time.

Quiz Questions (2/3)

Q3. A humanoid robot accelerates forward while walking. Its CoG projection lies slightly ahead of the support polygon, but ZMP lies inside. What does this indicate?

A. Robot is statically stable since ZMP is inside support polygon.
B. Robot is dynamically stable due to ZMP position.
C. Robot will definitely tip forward.
D. Stability cannot be determined without mass.

Q4. A humanoid robot experiences rapid forward deceleration during walking (negative acceleration along X direction). According to the ZMP equation:

$$x_{ZMP} = x_{CM} - \frac{z_{CM}}{Mg}\left(\ddot{x}_{CM} + \frac{M_y}{Mz_{CM}}\right)$$

Which effect occurs to ZMP location if CoM deceleration $\ddot{x}_{CM}$ becomes large negative?

A. ZMP shifts forward (walking direction).
B. ZMP shifts backward.
C. ZMP remains unaffected.
D. ZMP shifts randomly depending on mass.

Quiz Questions (3/3)

Q5. You are designing ZMP-based walking for a humanoid crossing stepping stones. Which is the primary reason ZMP planning becomes challenging on uneven terrain?

A. Gravitational acceleration changes.

B. Support polygon changes shape and location.

C. CoM remains constant but leg mass changes.

D. Robot mass fluctuates with steps.

Q6. A mobile manipulator lifts a heavy object high above its torso while standing on a flat surface. The CoM shifts upward and slightly forward. Which stability risk increases most?

A. Tipping moment arm increases.

B. Friction decreases.

C. ZMP moves closer to geometric center.

D. Joint torques become irrelevant.

Answer Key with Explanation

- **Q1: A** — As CoG projection approaches the polygon edge, the distance to tipping boundary reduces $\rightarrow$ stability margin decreases.
- **Q2: A** — Tripod gait maintains stability by ensuring CoG projection always remains inside the triangle formed by the 3 stance legs.
- **Q3: B** — Although CoG projection may be outside, ZMP inside the support polygon indicates dynamic stability is still maintained.
- **Q4: A** — Large negative deceleration increases inertial forces backward, causing ZMP to shift forward to balance resulting moment.
- **Q5: B** — On uneven terrain, the shape and position of the support polygon continuously change due to varying ground contact points.
- **Q6: A** — Lifting load upward increases CoG height and shifts mass away from base $\rightarrow$ larger tipping moment arm $\rightarrow$ higher tipping risk.

Part A: Newton-Euler Dynamics (Core)

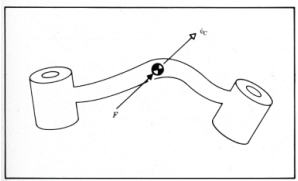
Newton-Euler Formulation: Core Idea

The Newton-Euler formulation describes the motion of a rigid body by combining translational and rotational dynamics.

Translational (Linear) Motion — Newton's 2nd Law:

$$\mathbf{F} = m\mathbf{a}_C$$

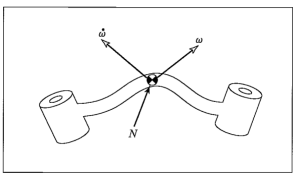
where $\mathbf{F}$ is the net external force, m is the mass, and $\mathbf{a}_C$ is the acceleration of the center of mass.



Rotational (Angular) Motion — Euler's Equation:

$$\boldsymbol{\tau} = I_C\boldsymbol{\alpha} + \boldsymbol{\omega} \times (I_C\boldsymbol{\omega})$$

where $\boldsymbol{\tau}$ is the net external torque, I_C is the inertia tensor about the CoM, $\boldsymbol{\alpha}$ is the angular acceleration, and $\boldsymbol{\omega}$ is the angular velocity.



Newton-Euler Formulation: Core Idea (contd.)

These two equations together fully describe rigid body motion in 3D.

The Newton-Euler formulation is the foundation for:

- Robot link dynamics
- Inverse dynamics
- Control design
- Whole-body dynamic models

Newton-Euler Equations in Terms of Linear and Angular Momentum

Newton's Equation for Linear Acceleration

$$\sum \mathbf{F} = \dot{\mathbf{G}} = m\frac{d\mathbf{v}}{dt} = m\mathbf{a}, \quad \mathbf{G} = m\mathbf{v}$$

The resultant of all forces acting on a system is equivalent to the system's time rate of change of linear momentum ($\mathbf{G}$).

Euler's Equation for Angular Acceleration

$$\sum \mathbf{M} = \dot{\mathbf{H}} = \frac{d(I\boldsymbol{\omega})}{dt}, \quad \mathbf{H} = I_c\boldsymbol{\omega}$$

$$\sum \mathbf{M} = I_c\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times (I_c\boldsymbol{\omega})$$

$$\sum \mathbf{M} = I_c\boldsymbol{\alpha} + \boldsymbol{\omega} \times (I_c\boldsymbol{\omega})$$

For planar systems, the gyroscopic term $\boldsymbol{\omega} \times (I_c\boldsymbol{\omega})$ vanishes.

Part B: Recursive Newton-Euler Dynamincs

Newton-Euler Formulation: Inverse Dynamics Approach

Main Objective:

Given joint positions, velocities, and accelerations:

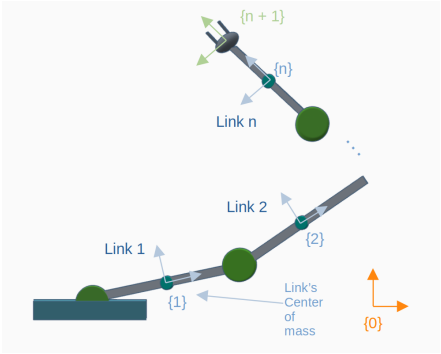
- Compute joint torques τ required to achieve this motion.

Robot Structure:

- Robot manipulator = chain of rigid links.
- Each link moves relative to its neighbors.

Approach:

- Analyze link-by-link.
- Newton-Euler breaks down full system into smaller subproblems.



Newton-Euler Formulation: Inverse Dynamics Approach (contd.)

What Inputs Do We Need?

Known quantities:

- Joint positions θ
- Joint velocities $\dot{\theta}$
- Joint accelerations $\ddot{\theta}$
- Robot kinematics: link lengths, joint types
- Mass properties: mass, center of mass, inertia tensor

Source of information:

- Control system commands
- Motion capture or sensors

Newton's Law (Translation):

$$F = ma$$

Euler's Law (Rotation):

$$N = I\alpha + \omega \times (I\omega)$$

Applied at each link:

- Separate equations for each link
- Avoids solving large system directly

Why Recursive?

We propagate information in two directions:

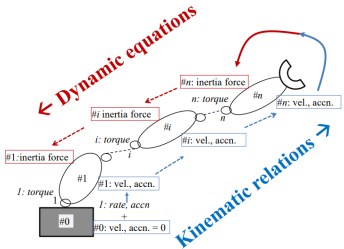
Forward Pass (Kinematics – Blue Arrows):

- Start at base #0: $\omega_0 = 0$, $\alpha_0 = 0$, a_0 includes gravity.
- At joint i: compute ω_i, α_i, a_i.
- Move link-by-link toward end-effector #n.

Backward Pass (Dynamics – Red Arrows):

- Start at end-effector #n (external forces/moments known or zero).
- Compute inertia forces/moments.
- Propagate joint forces f_i, moments n_i. and Extract joint torques τ_i.

What is recursive?



Main Benefit: Recursive evaluation is fast and efficient for real-time robot control.

Symbols and Notation

Frames and Indices:

- i, $i+1$: Link indices
- ${}^{i}(\cdot)$: Quantity expressed in frame i
- ${}^{i+1}R_i$: Rotation matrix from frame i to $i+1$
- ${}^{i}R_{i+1}$: Rotation matrix from frame $i+1$ to i

Joint Variables:

- θ_{i+1}: Joint angle
- $\dot{\theta}_{i+1}$: Joint angular velocity
- $\ddot{\theta}_{i+1}$: Joint angular acceleration

Velocities and Accelerations:

- ${}^{i}\omega_i$: Angular velocity
- ${}^{i}\alpha_i$: Angular acceleration
- ${}^{i}a_i$: Linear acceleration (origin)
- ${}^{i}a_{c_i}$: Linear acceleration at CoG
- ${}^{i}P_{i+1}$: Vector to next frame
- ${}^{i}P_{c_i}$: Vector to CoG

Symbols and Notation (cont.)

Forces and Moments:

- m_i: Mass of link i
- I_{c_i}: Inertia tensor at CoG
- ${}^{i}F_i$: Inertial force
- ${}^{i}N_i$: Inertial moment
- ${}^{i+1}f_{i+1}$: Force on link $i+1$ by link i
- ${}^{i}f_i$: Force on link i by link $i-1$
- ${}^{i+1}n_{i+1}$: Moment on link $i+1$ by link i
- ${}^{i}n_i$: Moment on link i by link $i-1$
- τ_i: Joint torque
- g: Gravitational acceleration

Newton-Euler Formulation: Overview

Two Recursive Phases:

- **Forward recursion:** propagation of joint velocities and accelerations.
- **Backward recursion:** propagation of forces and moments.

Key Assumptions:

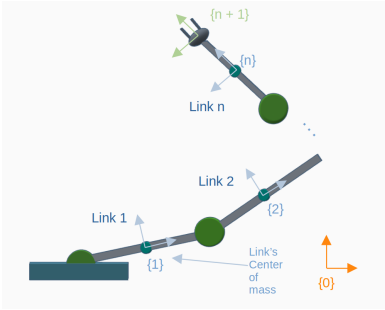
- Rigid links with mass and inertia concentrated at link centers of mass.
- Ideal (massless) joints.

Physical Laws Applied:

- Newton's law: $F = ma$
- Euler's law: $N = I\alpha + \omega \times (I\omega)$

Advantages:

- Efficient for serial manipulators.
- Suitable for real-time inverse dynamics.



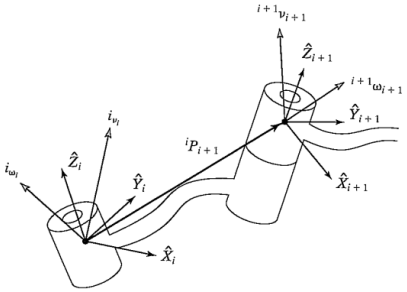
Forward Recursion: Velocities and Accelerations

Angular Velocity Propagation:

$$^{i+1}\omega_{i+1} = {}^{i+1}R_i \cdot {}^{i}\omega_i + \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_{i+1} \end{bmatrix}$$

Angular Acceleration Propagation:

$$^{i+1}\alpha_{i+1} = {}^{i+1}R_i \left({}^{i}\alpha_i + {}^{i}\omega_i \times \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_{i+1} \end{bmatrix} \right) + \begin{bmatrix} 0 \\ 0 \\ \ddot{\theta}_{i+1} \end{bmatrix}$$



Linear Acceleration at Frame Origin:

$$^{i+1}a_{i+1} = {}^{i+1}R_i \left({}^{i}a_i + {}^{i}\alpha_i \times {}^{i}P_{i+1} + {}^{i}\omega_i \times \left({}^{i}\omega_i \times {}^{i}P_{i+1} \right) \right)$$

Inertial Forces and Moments (at CoG)

Inertial Force:

$$^{i}F_i = m_i \cdot {}^{i}a_{c_i}$$

Inertial Moment:

$$^{i}N_i = I_{c_i} \cdot {}^{i}\alpha_i + {}^{i}\omega_i \times \left(I_{c_i} \cdot {}^{i}\omega_i\right)$$

- Inertial force is proportional to linear acceleration of center of mass. - Inertial moment includes both rotational acceleration and gyroscopic coupling.

Backward Recursion: Forces, Moments, and Torque Extraction

Force Recursion:

$$^{i}f_i = {}^{i}R_{i+1} \cdot {}^{i+1}f_{i+1} + {}^{i}F_i$$

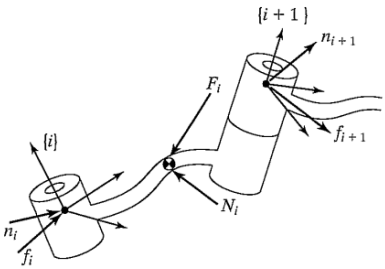
Moment Recursion:

$$^{i}n_i = {}^{i}R_{i+1} \cdot {}^{i+1}n_{i+1} + {}^{i}N_i + \left({}^{i}P_{c_i} \times {}^{i}F_i\right) + \left({}^{i}P_{i+1} \times \left({}^{i}R_{i+1}\right.\right.$$

Joint Torque Extraction:

$$\tau_i = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \cdot {}^{i}n_i$$

- Start recursion from end-effector. - Propagate forces and moments backward. - Extract joint torque at each joint axis.



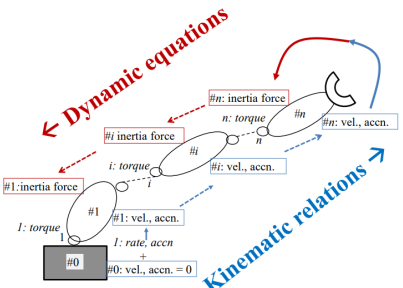
Full Newton-Euler Recursive Flow Summary

- **Initialization at base:**

$$ {}^0\omega_0 = {}^0\alpha_0 = 0, \quad {}^0a_0 = \begin{bmatrix} 0 \\ -g \\ 0 \end{bmatrix} $$

- **Forward Recursion (per link):**
  - Compute: $\omega_{i+1}, \alpha_{i+1}, a_{i+1}, a_{c_i}, F_i, N_i$
- **Backward Recursion (per link):**
  - Compute: f_i, n_i, τ_i
- **Result:**
  - Efficient inverse dynamics solution for serial manipulators.

What is recursive?



Part C: Whole Body dynamics and Momentum Formulation

Equation of motion: General form

Newton-Euler and Momentum Formulation

The equations of motion of a physical system describe its **motion** as a function of **time** and optional **control inputs**. In their general form, they are written:

$$F(q(t), \dot{q}(t), \ddot{q}(t), u(t), t) = 0$$

Where:

- t: the time variable,
- q: the vector of generalized coordinates (e.g., joint angles for a manipulator),
- $\dot{q}$: first time-derivative (velocity) of q,
- $\ddot{q}$: second time-derivative (acceleration) of q,
- u: vector of control inputs.

These equations provide a mapping between the **control space** (the actuator commands) and the **state space** of robot motions.

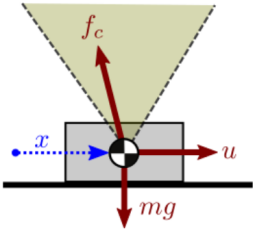
Example: Block Sliding on a Table

Imagine a rigid block sliding on a table, which we can see as a **1-DOF (one degree of freedom)** system with coordinate x. The operator applies a horizontal force u that pushes the block forward.

If the force is high enough to overcome friction, applying Newton's second law of motion (and Coulomb's model of sliding friction), we get:

$$m\ddot{x} = u - \mu_k mg$$

where μ_k is the **kinetic coefficient of friction**.



This expression is of the form:

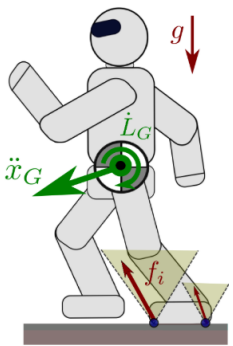
$$F(x, \dot{x}, \ddot{x}, u, t) = 0$$

Newton-Euler Equations

The Newton-Euler equations of motion correspond to the six unactuated coordinates in the equations of motion of our robots.

Newton's equation applies to (linear) translational motions:

$$\sum_{\text{link } i} m_i \ddot{p}_i = mg + \sum_{\text{contact } i} f_i$$



Euler's equation applies to angular motions:

$$\sum_{\text{link } i} (p_i - p_G) \times m_i \ddot{p}_i + I_i \dot{\omega}_i + \omega_i \times (I_i \omega_i) = \sum_{\text{contact } i} (p_i - p_G) \times f_i + \tau_i$$

Newton's Equation: Center of Mass

Let p_i denote the position in the world frame of the center of mass of the robot's i^{th} link. Let m_i denote the mass of link i, and $m = \sum_i m_i$ the total mass of the robot. The overall center of mass G is located at the position p_G in the world frame such that:

$$mp_G = \sum_{\text{link } i} m_i p_i$$

In other words, the robot's center of mass is a convex combination of the centers of mass of its links, weighted by their masses.

FORCES

The robot is subject to gravity and contact forces. Let g denote the gravity vector.
For a link i in contact with the environment, write f_i as the resultant force exerted on the link.

Newton's Equation: Forces and Internal Forces

Let h_{ij} denote the internal force exerted by link i on link j.
We take $h_{ij} = 0$ if links i and j are not connected (similarly, $f_i = 0$ if link i is not in contact). All force vectors are expressed in the world frame.

Newton's equation of motion links the resultant accelerations and forces:

$$\sum_{\text{link } i} m_i \ddot{p}_i = \sum_{\text{link } i} m_i g + f_i + \sum_{j \neq i} h_{ij}$$

By Newton's third law:

$$h_{ij} = -h_{ji} \implies \sum_i \sum_{j \neq i} h_{ij} = 0$$

Thus:

$$\sum_{\text{link } i} m_i \ddot{p}_i = \sum_{\text{link } i} m_i g + f_i$$

Momentum Form of Newton's Equation

The linear momentum of the robot is defined by:

$$P := m\dot{p}_G$$

Then, Newton's equation can be written as:

$$\dot{P} = m\ddot{p}_G = mg + \sum_{\text{contact } i} f_i$$

In other words, the rate of change of linear momentum equals the sum of external forces exerted on the robot.

Euler's Equation: Setup and Notation

Newton's equation is related to translational motions of the robot. Euler's equation provides a similar relation for angular motions.

Let R_i denote the rotation matrix from the i^{th} link frame to the world frame.

Let ω_i denote the spatial angular velocity of the link, that is, the angular velocity from the link frame to the world frame, expressed in the world frame.

Let I_i denote the inertia matrix of the i^{th} link, expressed in the world frame and taken at the center of mass p_i of the link.

Euler's Equation of Motion

For a link i in contact with the environment, we write τ_i as the resultant moment of contact forces exerted on the link at p_i. If the link is in point contact at p_i, the moment will be zero. In surface contact, both f_i and τ_i may be non-zero.

Euler's equation of motion links angular momentum and external moments:

$$\sum_{\text{link } i} (p_i - p_G) \times m_i \ddot{p}_i + I_i \dot{\omega}_i + \omega_i \times (I_i \omega_i) = \sum_{\text{link } i} (p_i - p_G) \times (f_i + m_i g) + \tau_i$$

Because:

$$\sum_{\text{link } i} (p_i - p_G) \times m_i g = 0$$

The simplified equation becomes:

$$\sum_{\text{link } i} (p_i - p_G) \times m_i \ddot{p}_i + I_i \dot{\omega}_i + \omega_i \times (I_i \omega_i) = \sum_{\text{contact } i} (p_i - p_G) \times f_i + \tau_i$$

Momentum Form of Euler's Equation

The angular momentum of the robot, taken at the center of mass G, is defined by:

$$L_G := \sum_{\text{link } i} (p_i - p_G) \times m_i \dot{p}_i + I_i \omega_i$$

Then, Euler's equation can be written in concise form as:

$$\dot{L}_G = \sum_{\text{contact } i} (p_i - p_G) \times f_i + \tau_i$$

In other words, the rate of change of the angular momentum equals the resultant moment of external forces exerted on the robot.