Chapter 11: Dynamics of Robot Motion

Introduction

Dynamics is a core component of robotics that deals with the *forces and torques* acting on a robot and the resulting *motions*. Unlike kinematics, which only considers motion without regard to the causes, dynamics accounts for the *mass, inertia, external forces,* and *internal actuators* that influence how a robot moves. For civil engineers involved in construction robotics, automated machinery, or inspection drones, understanding the dynamics of robot motion is essential for safe, efficient, and precise operation.

Robot dynamics is fundamentally split into two major domains:

- **Forward Dynamics (Direct Dynamics)**: Calculates accelerations given the torques and forces.
- **Inverse Dynamics**: Computes the required torques or forces to produce a desired motion (acceleration, velocity).

This chapter dives into both these aspects while covering mathematical modeling, Newton-Euler and Lagrangian formulations, dynamic equations of motion, and applications in control.

11.1 Difference Between Kinematics and Dynamics

- **Kinematics** describes motion in terms of position, velocity, and acceleration, without considering the causes (i.e., forces and torques).
- **Dynamics** explains *why* the motion occurs, by considering forces, torques, and masses.

  Kinematics → What happens Dynamics → Why it happens

In robotics, both are used: kinematics for planning, and dynamics for actuation and control.

11.2 Newton-Euler Formulation

11.2.1 Basic Principles

The **Newton-Euler formulation** combines Newton's second law (for translational motion) and Euler's equations (for rotational motion).

- **Newton's Law**: $\mathbf{F} = m \cdot \mathbf{a}$
- **Euler's Law**: $\tau = \mathbf{I} \cdot \alpha + \omega \times (\mathbf{I} \cdot \omega)$

Where:

- **F**: Force vector
- m: Mass
- **a**: Linear acceleration
- τ: Torque
- **I**: Inertia tensor
- ω: Angular velocity
- α: Angular acceleration

11.2.2 Recursive Newton-Euler Algorithm

For an *n-link manipulator*, the recursive Newton-Euler algorithm operates in two phases:

1. **Forward recursion**: Compute linear and angular velocities and accelerations from base to end-effector.
2. **Backward recursion**: Compute forces and torques from end-effector to base.

11.2.3 Advantages

- Computationally efficient
- Suitable for real-time control applications
- Good for serial manipulators

11.3 Lagrangian Formulation

11.3.1 Lagrangian Mechanics Basics

The **Lagrangian (L)** is defined as the difference between kinetic and potential energy:

$$L = T - V$$

Where:

- T: Total kinetic energy
- V: Total potential energy

The **Euler-Lagrange Equation** is used to derive dynamic equations:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} = \tau_i$$

Where:

- q_i: Generalized coordinate (joint angle, position, etc.)
- $\dot{q}_i$: Generalized velocity
- τ_i: Generalized force or torque

11.3.2 Application to Robotics

For an **n-DOF manipulator**, we write the total kinetic and potential energies as functions of joint coordinates, velocities, and link properties. Applying the Euler-Lagrange equation to each DOF gives us a set of *n coupled nonlinear second-order differential equations.*

11.4 Dynamic Equation of Motion (EoM)

The general form of robot dynamic equations (in joint space) is:

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q}) = \tau$$

Where:

- $\mathbf{M}(\mathbf{q})$: Mass/Inertia matrix (n × n)
- $\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})$: Coriolis and centrifugal force matrix
- $\mathbf{G}(\mathbf{q})$: Gravity torque vector
- τ: Vector of joint torques
- $\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}$: Joint position, velocity, and acceleration vectors

11.4.1 Components Explanation

- **Mass Matrix $\mathbf{M}(\mathbf{q})$**: Symmetric and positive definite. Depends on link masses and geometry.
- **Coriolis/Centrifugal Matrix $\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})$**: Causes complex non-linear interactions in multi-DOF systems.
- **Gravity Vector $\mathbf{G}(\mathbf{q})$**: Models the effect of gravity on each joint.

11.5 Forward and Inverse Dynamics

11.5.1 Forward Dynamics

Given:

- Joint torques τ Find:
- Accelerations $\ddot{\mathbf{q}}$

Solve:

$$\ddot{\mathbf{q}} = \mathbf{M}^{-1}(\tau - \mathbf{C}\dot{\mathbf{q}} - \mathbf{G})$$

Applications:

- Simulation
- Trajectory verification
- Motion prediction

11.5.2 Inverse Dynamics

Given:

- Desired $\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}$ Find:
- Required τ

Used in:

- Control systems (e.g., computed torque control)
- Real-time torque computation

11.6 Dynamic Modeling of Manipulators

Each robotic link is modeled using:

- Link mass
- Length
- Center of mass
- Inertia tensor

Two modeling approaches:

1. **Symbolic Modeling**: Using software tools (e.g., MATLAB Symbolic Toolbox)
2. **Numerical Modeling**: Real-time computations using parameters

---

11.7 Friction and Actuator Dynamics

11.7.1 Friction Models

- **Static Friction (Coulomb)**: Constant opposing force at low velocity
- **Viscous Friction**: Proportional to velocity
- **Stribeck Effect**: Drop in friction near zero velocity

Total torque due to friction:

$$\tau_f = \tau_c \cdot \text{sign}(\dot{q}) + b \cdot \dot{q}$$

11.7.2 Actuator Dynamics

Actuators (like electric motors or hydraulic cylinders) have dynamics that affect the system. Common elements:

- Motor torque-speed curve
- Time delay
- Gear ratios

In real robots, actuator dynamics are modeled as a first-order or second-order system.

11.8 External Forces and Contact Dynamics

11.8.1 External Disturbances

- Forces from human interaction
- Obstacle collisions
- Environmental disturbances (wind, vibration)

11.8.2 Contact Dynamics

Key for:

- Walking robots
- Grasping and manipulation

Must consider:

- Normal and tangential forces
- Slipping and rolling
- Impact dynamics

Methods:

- Penalty-based models
- Constraint-based models (using Lagrange multipliers)

11.9 Dynamics for Parallel and Mobile Robots

11.9.1 Parallel Robots

- Dynamics are more complex due to closed-loop chains.
- Requires constraint handling in equations of motion.

11.9.2 Mobile Robots

Involves:

- Rolling constraints
- Skid/slip modeling
- Dynamic control of non-holonomic systems

Equations derived using Lagrangian or Kane's method.

11.10 Simulation and Control Applications

11.10.1 Simulation Tools

- MATLAB/Simulink
- Gazebo
- ROS + RViz
- MSC Adams

These tools are used to test dynamic models, implement controllers, and simulate robot-environment interaction.

11.10.2 Dynamic Control Strategies

- **Computed Torque Control**
- **Model Predictive Control**
- **Force Control**
- **Adaptive and Robust Control**

Each uses dynamic models for real-time control of motion and interaction.

11.10.3 Computed Torque Control (CTC)

Computed Torque Control is a model-based nonlinear control technique used in robotic systems to achieve accurate trajectory tracking. It uses the inverse dynamics model of the robot to linearize and decouple the joint dynamics.

Control Law:

$$\tau = \mathbf{M}(\mathbf{q})\mathbf{v} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q})$$

Where:

$$\mathbf{v} = \ddot{\mathbf{q}}_d + \mathbf{K}_v(\dot{\mathbf{q}}_d - \dot{\mathbf{q}}) + \mathbf{K}_p(\mathbf{q}_d - \mathbf{q})$$

- $\mathbf{q}_d$: Desired position
- $\dot{\mathbf{q}}_d$: Desired velocity
- $\ddot{\mathbf{q}}_d$: Desired acceleration
- $\mathbf{K}_v, \mathbf{K}_p$: Velocity and position gain matrices

Advantages:

- Linearizes the system
- Good tracking performance
- Effective for set-point regulation

Challenges:

- Requires precise modeling
- Sensitive to parameter variations
- Not robust to external disturbances

11.10.4 Adaptive Control

Adaptive control is used when there are uncertainties in robot parameters such as mass, inertia, or friction. It modifies control parameters in real time based on observed data.

Approaches:

- **Model Reference Adaptive Control (MRAC)**
- **Self-Tuning Regulators (STR)**
- **Adaptive Computed Torque Control**

These methods estimate unknown parameters and adjust control laws accordingly.

Applications:

- Industrial manipulators with unknown payloads
- Collaborative robots (cobots) adapting to human force
- Autonomous systems handling unstructured environments

11.10.5 Robust Control

Robust control is designed to function correctly despite model uncertainties and external disturbances. It guarantees performance within a specified bound.

Techniques:

- **H-infinity Control**
- **Sliding Mode Control (SMC)**
- **Bounded Input-Bounded Output (BIBO) stability**

Sliding Mode Control Example:

A robust and simple method using a sliding surface $s(t)$:

$$s(t) = \dot{e}(t) + \lambda e(t)$$

Where $e(t) = q_d - q$

Control Law:

$$\tau = \tau_{eq} - k \cdot \text{sign}(s)$$

Provides high robustness but may suffer from *chattering*, which can be mitigated using smoothing functions.

11.11 Force and Impedance Control

Robots interacting with their environment need to manage both position and force. This is where **force control** and **impedance control** come into play.

11.11.1 Force Control

Ensures that the force applied by the robot on the environment stays within desired bounds.

- **Hybrid Position/Force Control**: Applies position control along some directions and force control along others.
- **Force Feedback**: Uses force/torque sensors to regulate contact forces in real-time.

11.11.2 Impedance Control

Introduced by Hogan (1985), impedance control regulates the mechanical impedance (relationship between force and motion):

$$F = M_d\ddot{x} + B_d\dot{x} + K_d x$$

Where:

- M_d: Desired mass
- B_d: Desired damping
- K_d: Desired stiffness

This control mode is compliant and widely used in human-robot interaction, rehabilitation robots, and contact-rich tasks.

11.12 Modeling and Control of Flexible Links

Many lightweight or long-reach robots have **flexible links** or **compliant joints**. In such cases, the dynamics involve not just rigid body motion but also vibration and elastic deformation.

11.12.1 Flexible Link Dynamics

Modeled using:

- Euler-Bernoulli Beam Theory
- Timoshenko Beam Model (if shear deformation is significant)
- Partial Differential Equations (PDEs) for distributed parameters

11.12.2 Control of Flexible Robots

Approaches include:

- **Modal Control**: Control each mode of vibration separately
- **Observer-Based Control**: Estimates flexible states and compensates accordingly
- **Feedforward and Feedback Compensation**

These systems are complex but necessary for lightweight space robots, aerial manipulators, and long-arm construction machines.

11.13 Dynamics in Legged and Wheeled Robots

11.13.1 Legged Robot Dynamics

Legged robots (e.g., bipeds, quadrupeds) require complex modeling of:

- Multi-body limb systems
- Ground reaction forces
- ZMP (Zero Moment Point) stability

Walking Dynamics are phase-based: stance and swing phases with discontinuities at foot impacts.

Control requires:

- Whole-body inverse dynamics
- Trajectory optimization
- Nonlinear predictive control

11.13.2 Wheeled Robot Dynamics

Wheeled mobile robots must satisfy *non-holonomic constraints* (no side slip for standard wheels):

Dynamic equations include:

- Rolling constraints
- Mass and inertia distribution
- Terrain interaction models (e.g., rough ground, slopes)

Control often uses:

- Lyapunov-based methods
- Differential drive or Ackermann steering kinematics + dynamics

---

11.14 Dynamics-Aware Path Planning

While traditional path planning uses kinematic constraints, **dynamics-aware path planning** considers acceleration, velocity, and force limits.

11.14.1 Time-Optimal Path Parameterization (TOPP)

Given a geometric path, determine time-optimal velocity profile under:

- Torque limits
- Velocity and acceleration bounds
- Dynamic constraints

11.14.2 Kinodynamic Planning

Combines kinematic and dynamic feasibility:

- Uses state-space search (position, velocity)
- Sampling-based planners: RRT*, Kinodynamic A*
- Applications: drones, self-driving vehicles, humanoids

11.15 Experimental Validation and Calibration

Robotic dynamics models need experimental validation to ensure accuracy.

11.15.1 System Identification

Techniques used to estimate physical parameters:

- Least Squares Estimation
- Recursive Estimation
- Frequency Domain Analysis

Sensors needed: encoders, IMUs, F/T sensors

11.15.2 Model Calibration

Adjusting model parameters (mass, center of mass, inertia) by comparing model predictions with real sensor data.

Calibration is crucial in:

- Aerospace and surgical robots
- Industrial manipulators

- Mobile platforms
