The Geometry of Motion

Volume III: Biomechanics

From Rigid Bodies to Biological Systems

Preface

Draft Notice. This volume is under active development. Chapters 1–3 contain substantial content; Chapters 4–10 are structural outlines with preliminary material that will be expanded in future editions. Exercises, worked examples, and backmatter (glossary, further reading) have not yet been written.

The first two volumes of *The Geometry of Motion* developed the mathematical language of tangent spaces, contraction theory, and trajectory-centric control with the implicit assumption that the systems under study are *rigid bodies*—that joints are perfect revolutes, that links are perfectly stiff, and that actuators produce torques on command. In biological systems, *none of these assumptions hold*.

This volume confronts reality. Biological tissues are viscoelastic, not rigid. Joints permit coupled translations and rotations, not single-axis pivots. Muscles produce force through a complex cascade from neural excitation through calcium dynamics to cross-bridge cycling, with force depending on length, velocity, and activation history. Tendons store and release elastic energy. Ligaments constrain motion in ways that change with loading rate.

And yet, the mathematical framework of Volumes 0–II is not abandoned—it is *extended*. The mass matrix $M(\boldsymbol{q})$ still exists, but it now includes the effective inertia of muscles and the compliance of tendons. The configuration space $\mathcal{Q}$ is still a manifold, but its geometry is richer because biological joints are not simple revolutes. The control input $\boldsymbol{u}$ is no longer a torque but a neural excitation signal, and the mapping from excitation to torque passes through the muscle model.

The philosophy remains unchanged: mathematics *driven by physical reality*, with every abstraction justified by measurement and every model validated against experiment. Where engineering systems are designed to be simple, biological systems have evolved to be *robust*. Understanding why requires the tools of this volume.

Contents

Nomenclature

General Conventions

- Scalars are lowercase or Greek letters (e.g., m, t, θ).
- Vectors are bold lowercase (e.g., $\mathbf{x}, \mathbf{u}, \mathbf{q}$).
- Matrices are bold uppercase (e.g., $\mathbf{A}, \mathbf{B}, \mathbf{M}, \mathbf{J}$).
- Expected values and sets are denoted by blackboard bold or script (e.g., $\mathbb{E}, \mathcal{S}$).

State and Kinematics

$\mathbf{q} \in \mathbb{R}^n$ Joint configuration vector (generalized coordinates).

$\dot{\mathbf{q}} \in \mathbb{R}^n$ Joint velocity vector (generalized velocities).

$\ddot{\mathbf{q}} \in \mathbb{R}^n$ Joint acceleration vector.

$\mathbf{x} \in \mathbb{R}^{nx}$ System state vector; commonly $\mathbf{x} = [\mathbf{q}^T, \dot{\mathbf{q}}^T]^T$ for mechanical systems.

$\mathbf{u} \in \mathbb{R}^m$ Control input vector (e.g., joint torques, muscle activations).

$\boldsymbol{\tau} \in \mathbb{R}^n$ Generalized forces/torques acting on the joints.

$t \in \mathbb{R}$ Time.

Inertia and Dynamics

$\mathbf{M}(\mathbf{q})$ Mass (inertia) matrix, symmetric positive definite.

$\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ Coriolis and centrifugal force matrix.

$\mathbf{g}(\mathbf{q})$ Gravity vector.

$\mathbf{J}(\mathbf{q})$ Jacobian matrix relating joint velocities to task-space velocities ($\dot{\mathbf{p}} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}}$).

Control and Optimization

$\mathbf{A}_k = \frac{\partial f}{\partial \mathbf{x}}$ Local Jacobian matrix of the state dynamics.

$\mathbf{B}_k = \frac{\partial f}{\partial \mathbf{u}}$ Local Jacobian matrix of the control input.

$V(\mathbf{x})$ **or** $\mathcal{V}$ Value function or Lyapunov function.

$\mathbf{Q}$ State penalty matrix in LQR/DDP.

$\mathbf{R}$ Control penalty matrix in LQR/DDP.

$\mathbf{K}$ Feedback gain matrix ($\delta\mathbf{u} = -\mathbf{K}\delta\mathbf{x}$).

$\boldsymbol{\gamma}$ A trajectory or flow, mapping time and initial conditions to states.

Biomechanics and Motor Control

$a_i \in [0, 1]$ Muscle activation level.

ℓ_i^M Muscle fiber length.

v_i^M Muscle fiber contraction velocity.

ℓ_i^{MT} Musculotendon unit length.

$\mathbf{F}_{\mathbf{muscle}}$ Force generated by muscles.

$\mathbf{W}$ Muscle synergy matrix (weights).

$\mathbf{c}(t)$ Muscle synergy activation vector over time.

Geometry and Manifolds

$SE(3)$ Special Euclidean group of rigid body motions.

$SO(3)$ Special Orthogonal group of 3D rotations.

$\mathfrak{se}(3)$ Lie algebra of $SE(3)$, space of spatial twists (velocities).

$\mathcal{M}$ A smooth manifold.

$T_{\mathbf{x}}\mathcal{M}$ Tangent space at point $\mathbf{x}$.

Chapter 1

How Biology Differs from Engineering

> Nature does not build machines. It grows organisms. The engineer's model of the body is a useful fiction—but it is fiction.
>
> — Adapted from D'Arcy Wentworth Thompson

1.1 The Rigid-Body Assumption and Its Limits

Every model in Volumes 0–II rested on a fundamental assumption: the moving system consists of *rigid bodies* connected by *ideal joints*. A rigid body has a fixed mass distribution; its shape does not change under load. An ideal revolute joint constrains five of six relative degrees of freedom, permitting only rotation about a single axis. These assumptions are extraordinarily productive for engineering systems—robot arms, spacecraft, mechanisms—where components are designed to approximate rigidity.

In biological systems, these assumptions fail at every level:

(i) **Bones are not rigid.** Long bones (femur, humerus, tibia) have measurable compliance under dynamic loading [23]. A golf club shaft deflects by several centimeters and several degrees during a swing; a human femur deflects by millimeters during running. The deflection is small in absolute terms but can have significant effects on end-effector accuracy.

(ii) **Joints are not revolutes.** The knee, for example, combines rolling, sliding, and rotation in a coupled manner that changes with flexion angle [19]. The glenohumeral (shoulder) joint permits motion in three rotational axes but also allows several millimeters of translation. The intervertebral joints of the spine have six degrees of freedom each, all coupled through the disc and ligament mechanics.

(iii) **Actuators are not torque sources.** Muscles produce force, not torque. Force is generated through the contraction of sarcomeres, transmitted through tendons, and converted to joint torque via moment arms that vary with joint angle. The force a muscle can produce depends on its length, its contraction velocity, and its activation history—dependencies that have no analog in electric motors.

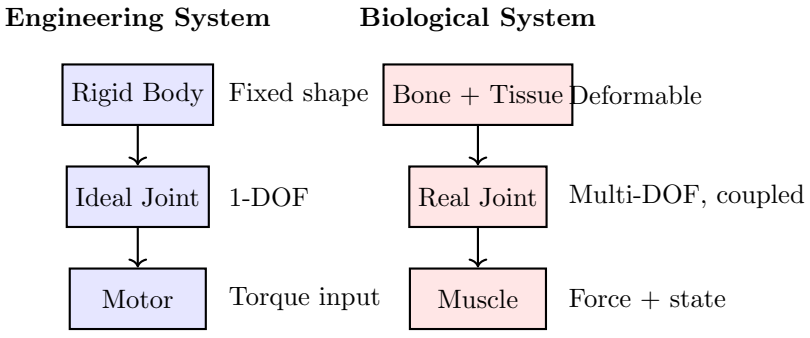
(iv) **Sensors are noisy, delayed, and multimodal.** Proprioceptors (muscle spindles, Golgi tendon organs, joint receptors) provide state information with significant noise and delays of 30–100 ms [28]. The central nervous system must fuse information from proprioceptive, visual, and vestibular sources to estimate the body's state.

(v) **The system adapts.** Unlike a robot, the human body changes over time—muscles fatigue, connective tissue remodels, neural pathways strengthen or weaken. A complete model must account for adaptation on timescales from seconds (fatigue) to years (training).

Biology vs. Engineering 1.1. The Fundamental Differences

Property	Engineering System	Biological System
Links	Rigid bodies	Viscoelastic, deformable
Joints	1-DOF revolute/prismatic	Multi-DOF, coupled, with laxity
Actuators	Motors (torque on command)	Muscles (force via excitation-contraction)
Transmission	Gears, belts (rigid)	Tendons (elastic, energy-storing)
Sensors	Encoders (precise, fast)	Proprioceptors (noisy, delayed)
Parameters	Fixed (by design)	Variable (fatigue, adaptation, growth)
Redundancy	Minimal (cost-driven)	Massive (evolution-driven)
Failure mode	Brittle (sudden)	Graceful (degradation)

These differences reflect fundamentally distinct design philosophies. Engineering systems are designed for precise, repeatable control of pre-specified outputs. Biological systems are evolved for robustness, adaptability, and efficiency under uncertain conditions. Neither approach is superior; rather, they optimize for different objectives. Understanding which differences are essential for your particular biomechanical question is crucial for successful modeling.



1.2 Why the Differences Matter for Control

The differences enumerated above are not merely academic. They fundamentally change the control problem in ways that Volumes I and II must now accommodate.

1.2.1 Compliance Creates Hidden Dynamics

When a link is compliant, its deformation adds internal degrees of freedom to the system. A golf club shaft, modeled as rigid, has one degree of freedom (the grip angle). Modeled as a flexible beam, it has *infinitely many* degrees of freedom, though practical models truncate to the first 2–3 bending modes. The equation of motion becomes:

$$\underbrace{M(\boldsymbol{q})\ddot{\boldsymbol{q}} + C(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + g(\boldsymbol{q})}_{\text{rigid-body dynamics}} + \underbrace{K_{\text{stiff}}\boldsymbol{q}_{\text{flex}} + D_{\text{damp}}\dot{\boldsymbol{q}}_{\text{flex}}}_{\text{flexibility terms}} = \boldsymbol{\tau}, \tag{1.1}$$

where $\boldsymbol{q}_{\text{flex}}$ represents the modal coordinates of the flexible modes, K_{stiff} is the stiffness matrix, and D_{damp} is the structural damping matrix.

Biomechanical Principle 1.1. The Timescale Separation Principle

In many biological systems, the flexible dynamics are *fast* relative to the rigid-body dynamics. If the natural frequency of the flexible modes is much higher than the bandwidth of the controller, the flexibility can be modeled as a quasi-static compliance rather than a dynamic mode. Specifically, if the k-th flexible mode has natural frequency ω_k and the controller bandwidth is ω_c, then:

- $\omega_k \gg \omega_c$: quasi-static (flexibility "follows" the rigid motion)
- $\omega_k \approx \omega_c$: resonance risk (flexibility must be modeled dynamically)
- $\omega_k \ll \omega_c$: the system is effectively another rigid body (the flexible mode is so slow it acts like a separate DOF)

The timescale separation principle is critical for model selection. For different body regions and different movement speeds, the applicability of the rigid-body approximation varies. For the human body during locomotion, empirical measurements suggest that long bones have natural frequencies sufficiently high that the rigid-body assumption is justified. For the spine and soft tissues, particularly during rapid movements, the separation is less clean, and dynamic flexibility effects may contribute meaningfully to the system behavior. The choice of whether to include flexibility as explicit state variables or treat it as a quasi-static effect should be guided by both the timescale analysis above and by quantitative comparison of model predictions to experimental data.

1.2.2 Muscle Force Production Is State-Dependent

The most striking difference between engineered and biological actuators is that muscle force depends on the *state of the muscle itself*. An electric motor produces torque proportional to current, regardless of shaft angle or angular velocity (to first approximation). A muscle's maximum force depends on:

1. **Muscle length** (force-length relationship): maximum isometric force occurs at an intermediate "optimal" length ℓ_0, with force decreasing at shorter and longer lengths.
2. **Contraction velocity** (force-velocity relationship): concentric (shortening) contractions produce less force at higher velocities; eccentric (lengthening) contractions can produce forces approximately 30–50% above isometric, depending on the specific model (see [29, 21] and Chapter 3).

3. **Activation level**: the fraction of motor units recruited and their firing rates, which follows its own first-order dynamics with time constants of 10–50 ms for activation and 40–200 ms for deactivation.

The combined force production model (Hill model, Chapter 3) is:

$$F_{\text{muscle}} = a(t) \cdot f_L(\tilde{\ell}) \cdot f_V(\tilde{v}) \cdot F_0 + f_{\text{PE}}(\tilde{\ell}), \tag{1.2}$$

where $a(t) \in [0, 1]$ is the activation level, f_L and f_V are the normalized force-length and force-velocity curves, F_0 is the maximum isometric force, $\tilde{\ell} = \ell/\ell_0$ and $\tilde{v} = v/v_0$ are normalized length and velocity, and f_{PE} is the passive elastic force.

Biomechanical Principle 1.2. Force-Velocity Asymmetry in Multi-Joint Tasks

In any task where joints undergo different phases of contraction (some concentric, others eccentric), the force-velocity relationship creates an asymmetry in joint torque capacity. This asymmetry can be quantified through the Hill force-velocity function applied to each muscle group. The engineering consequence is that a model assuming constant or task-invariant joint torque limits systematically misrepresents the instantaneous torque capacity of biological joints, particularly during rapid multi-joint movements.

1.2.3 Redundancy Is a Feature, Not a Bug

The human body has approximately 630 skeletal muscles controlling approximately 240 joint degrees of freedom [26]. Even for a simple task like reaching for a cup (which constrains only 6 degrees of freedom—3 position + 3 orientation of the hand), there are vastly more muscles and joints than constraints. This *motor redundancy* means that infinitely many muscle activation patterns can produce the same hand trajectory.

In engineering, redundancy is minimized to reduce cost. In biology, redundancy is *the solution to robustness*:

- If one muscle is fatigued, others can compensate.
- If a joint is injured, the task can be accomplished through alternative kinematic chains.
- Variability in execution—far from being noise to be eliminated—allows the system to explore the space of solutions and find robust attractors (see Volume IV for the motor control implications).

This connects directly to the *uncontrolled manifold hypothesis* [27]: task-irrelevant variability is not suppressed by the nervous system, only task-relevant variability is. The biological control system does not solve the full redundancy problem; it projects the problem onto the task-relevant subspace.

1.3 Modeling Strategies

Given the complexity of biological systems, modelers must choose their level of abstraction carefully. Three broad strategies exist, listed in order of increasing fidelity:

1.3.1 Strategy 1: Rigid-Body with Torque Actuators

The simplest approach: treat biological segments as rigid bodies and joints as ideal revolutes, with "joint torques" as the control input. This is the model assumed throughout Volumes I–II.

Advantages:

- Mathematically tractable (standard Lagrangian mechanics)
- Sufficient for gross motion analysis
- Compatible with all control-theoretic tools from Volumes I–II

Limitations:

- Cannot predict muscle activation patterns
- Cannot account for muscle force-length-velocity constraints
- Cannot predict injury mechanisms
- Cannot capture bi-articular muscle effects

When to use: Trajectory optimization, gross motion planning, initial controller design.

1.3.2 Strategy 2: Rigid-Body with Muscle Actuators

Replace torque actuators with Hill-type muscle models (Chapter 3). Joint torques become *outputs* of the model, computed from muscle forces and moment arms:

$$\tau_j = \sum_{i \in \mathcal{M}_j} r_{ij}(\boldsymbol{q}) \cdot F_{\text{muscle},i}, \tag{1.3}$$

where $\mathcal{M}_j$ is the set of muscles crossing joint j and $r_{ij}(\boldsymbol{q})$ is the moment arm of muscle i about joint j.

Advantages:

- Predicts muscle activation patterns
- Captures force-length-velocity constraints
- Accounts for bi-articular coupling
- Can predict co-contraction and joint stiffness modulation

Limitations:

- Significant increase in model complexity (typically 2–3 states per muscle)
- Muscle parameters are difficult to measure in vivo
- Still assumes rigid segments and ideal joints

When to use: Detailed motion analysis, muscle force prediction, clinical biomechanics, prosthetic design.

1.3.3 Strategy 3: Flexible Multi-Body with Muscle Actuators

The most complete approach: flexible bodies (beam or FEM models) with muscle actuators and realistic joint models. This is the domain of software platforms like OpenSim and AnyBody.

Advantages:

- Highest fidelity
- Can predict stress distributions in bones and soft tissues
- Can capture coupled joint kinematics

Limitations:

- Computationally expensive
- Many parameters, often poorly known
- Too complex for real-time control applications

When to use: Injury prediction, surgical planning, detailed structural analysis.

```python
import numpy as np

def select_modeling_strategy(
    need_muscle_forces: bool,
    need_tissue_stress: bool,
    real_time_required: bool,
    n_dof: int
) -> str:
    """Select the appropriate biomechanical modeling strategy.

    Parameters
    ----------
    need_muscle_forces : bool
        Whether individual muscle forces are needed.
    need_tissue_stress : bool
        Whether stress/strain in tissues is needed.
    real_time_required : bool
        Whether the model must run in real-time.
    n_dof : int
        Number of degrees of freedom in the model.

    Returns
    -------
    str
        Recommended modeling strategy.
    """
    if need_tissue_stress:
        return "Strategy 3: Flexible multi-body + muscles (OpenSim/FEM)"
    elif need_muscle_forces:
        if real_time_required and n_dof > 20:
            return ("Strategy 2 (reduced): Muscle model with "
                    "synergy-based dimensionality reduction")
        return "Strategy 2: Rigid-body + Hill-type muscles (OpenSim)"
    else:
        if real_time_required:
            return "Strategy 1: Rigid-body + torque actuators (fastest)"
        return ("Strategy 1 or 2 depending on available data "
```

```
                "and analysis goals")
```

Listing 1.1: Choosing a modeling strategy based on application requirements.

1.4 A Quantitative Tour of the Human Body

Before diving into the detailed modeling chapters, we present a quantitative overview of the key biomechanical properties that any model must capture. All values in this section are drawn from published anthropometric studies and are cited in the relevant subsections.

1.4.1 Segment Inertial Properties

The human body can be decomposed into 15–20 segments for modeling purposes. The standard segmentation and mass fractions are based on cadaver studies, most notably those of Dempster (1955), updated by de Leva (1996) [8].

Segment	Mass (% body)	Length (cm, 180cm male)	CoM (% proximal)
Head + neck	8.1	25.4	50.0
Trunk	49.7	53.0	44.9
Upper arm	2.7	28.2	57.7
Forearm	1.6	26.9	45.7
Hand	0.6	19.0	36.2
Thigh	14.2	42.2	40.9
Shank	4.3	43.4	43.4
Foot	1.4	25.8	44.2

```python
import numpy as np
from dataclasses import dataclass

@dataclass
class SegmentProperties:
    """Inertial properties of a body segment."""
    name: str
    mass: float          # kg
    length: float        # m
    com_proximal: float  # fraction from proximal end
    I_com: float         # moment of inertia about CoM, kg*m^2

def de_leva_male_segments(
    body_mass: float,
    body_height: float
) -> list[SegmentProperties]:
    """Compute segment properties using de Leva (1996) regression.

    Parameters
    ----------
    body_mass : float
        Total body mass in kg.
    body_height : float
        Total body height in m.
```

```

    Returns
    -------
    list[SegmentProperties]
        List of segment inertial properties.

    References
    ----------
    de Leva, P. (1996). Adjustments to Zatsiorsky-Seluyanov's
    segment inertia parameters. J. Biomechanics, 29(9), 1223-1230.
    """
    # Mass fractions, length fractions, CoM fractions, RoG fractions
    # (proximal reference, male data from de Leva 1996 Table 4)
    data = {
        'head_neck':  (0.0810, 0.1414, 0.5002, 0.3030),
        'trunk':      (0.4970, 0.2946, 0.4486, 0.3271),
        'upper_arm':  (0.0271, 0.1566, 0.5772, 0.2850),
        'forearm':    (0.0162, 0.1494, 0.4574, 0.2760),
        'hand':       (0.0061, 0.1057, 0.3624, 0.2880),
        'thigh':      (0.1416, 0.2321, 0.4095, 0.3290),
        'shank':      (0.0433, 0.2411, 0.4340, 0.2550),
        'foot':       (0.0137, 0.1527, 0.4415, 0.2570),
    }

    segments = []
    for name, (mass_frac, len_frac, com_frac, rog_frac) in data.items():
        seg_mass = mass_frac * body_mass
        seg_length = len_frac * body_height
        seg_com = com_frac
        # Moment of inertia: I = m * (rog * length)^2
        seg_I = seg_mass * (rog_frac * seg_length) ** 2
        segments.append(SegmentProperties(
            name=name, mass=seg_mass, length=seg_length,
            com_proximal=seg_com, I_com=seg_I
        ))

    return segments


# Example usage
if __name__ == "__main__":
    segments = de_leva_male_segments(body_mass=80.0, body_height=1.80)
    print(f"{'Segment':<15} {'Mass (kg)':>10} {'Length (m)':>10} "
          f"{'I_com (kg*m^2)':>14}")
    print("-" * 52)
    for seg in segments:
        print(f"{seg.name:<15} {seg.mass:>10.2f} {seg.length:>10.3f} "
              f"{seg.I_com:>14.4f}")
```

Listing 1.2: Computing segment inertial properties from body measurements.

1.4.2 Joint Ranges of Motion

Each joint has characteristic ranges of motion that define the configuration space $\mathcal{Q}$ of the body. Unlike robot joints, which can be modeled as $q_i \in [\underline{q}_i, \overline{q}_i]$, biological joints have *soft limits*: resistance increases gradually as the joint approaches its anatomical limit, governed by the elasticity of ligaments and joint capsule.

1.4.3 Muscle Properties

The human body contains approximately 630 skeletal muscles. Each is characterized by several anatomical and physiological parameters that determine its force production capacity. The standard parameters are:

- **Maximum isometric force** F_0: the force a muscle produces at optimal length with full activation and zero velocity. Values vary widely by muscle size, composition, and fiber type.
- **Optimal fiber length** ℓ_0: the length at which maximum isometric force is produced, determined by the sarcomere length ($\sim$2.6 μm per sarcomere at ℓ_0).
- **Maximum contraction velocity** v_0: the velocity at which a muscle produces zero force under concentric contraction, measured in fiber lengths per second (typically 0.5–3 fiber lengths per second for slow-twitch fibers, 5–15 for fast-twitch).
- **Tendon slack length** ℓ_T^s: the length of the tendon at which it begins to transmit force (below this length, the tendon is slack and carries no load).
- **Pennation angle** α_0: the angle between muscle fiber direction and the tendon line of action at optimal fiber length. Ranges from approximately 0° for parallel-fibered muscles to 30° or more for highly pennate muscles.

These properties have been systematically measured and catalogued through:

1. [36]: foundational review of muscle mechanics and parameter estimation
2. [9]: comprehensive OpenSim musculoskeletal model with measured parameters for major muscles
3. [4]: lower-limb muscle morphometry from imaging and cadaver studies
4. [16]: the corresponding upper-extremity model, and the source of the biceps parameters used in Chapter 3

Values for specific muscles are provided in software platforms (OpenSim 4.x, AnyBody) and should be obtained from published measurements rather than estimated.

1.5 Connecting to the Control Framework

Despite the added complexity, the control framework of Volumes I and II remains applicable—with modifications. The key insight is:

Biomechanical Principle 1.3. The Extended Affine Structure

A musculoskeletal system can be written in control-affine form as

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}) + G(\boldsymbol{x})\boldsymbol{u}, \tag{1.4}$$

where the state $\boldsymbol{x}$ now includes muscle states (activation, fiber length) in addition to joint coordinates and velocities, the drift field f includes passive muscle and tendon

forces, and $G(\boldsymbol{x})$ maps neural excitation signals $\boldsymbol{u}$ to state derivatives through the muscle activation and contraction dynamics. The dimensions expand—from $2n$ states with n joints and m torque actuators to $2n + 2p$ states with p muscles—but the *structure* of the problem (drift + control) is preserved.

Remark 1.1. Dimensional Expansion of the State Space

Recall from Volumes 0 and I that the state manifold $\mathcal{M}$ has dimension $2n$ for a system with n joint degrees of freedom (tracking positions and velocities). When muscles are included, the state space expands: for each of p muscles, we add two dimensions to track activation a_i and fiber length ℓ_i^M, yielding a state space of dimension $2n + 2p$. The system still evolves according to $\dot{\boldsymbol{x}} = f(\boldsymbol{x}) + G(\boldsymbol{x})\boldsymbol{u}$, but the ambient manifold is now much higher-dimensional. All differential geometry tools (Lie brackets, contraction metrics, etc.) from Volumes I–II apply without modification, though computational complexity scales with dimension.

This means:

- The tangent-space analysis of Volume I applies to the expanded state space
- Contraction theory (Volume I, Chapter 4) can assess stability of the muscle-driven system
- Trajectory optimization (Volume I, Chapter 5) extends to find optimal neural excitation patterns
- The counterfactual analysis (Volume I, Chapter 7) can distinguish which aspects of muscle force production contribute to the motion

The challenge is *dimensionality*: a 20-DOF body with 100 muscles has a state space of dimension $2 \times 20 + 2 \times 100 = 240$. This is precisely the "curse of dimensionality" that Volume IV addresses through the lens of motor control and neural architecture.

Exercises

1. **Segment Properties.** Using the `de_leva_male_segments()` function, compute the total mass and center of mass of a 75-kg, 1.75-m male in the anatomical position. Verify that the total mass sums to the body mass.

2. **Compliance Effects.** A golf club shaft has a first bending mode at approximately 5 Hz. If the downswing takes 0.25 s, how many oscillation cycles of the shaft occur during the downswing? Does the timescale separation principle (Principle 1.1) justify neglecting shaft flexibility? Discuss.

3. **Redundancy Quantification.** A planar model of the arm has 3 joints (shoulder, elbow, wrist) and 6 muscles. If the task is to place the fingertip at a given (x, y) position, what is the dimension of the null space? How many independent muscle patterns produce the same fingertip position?

4. **Force-Velocity Impact.** During a golf downswing, the wrist extensors lengthen at approximately 5 fiber lengths per second. Using the Hill force-velocity relationship (Equation 1.2), estimate the ratio of eccentric force to maximum isometric force at this velocity. Compare this with the force available during a slow concentric contraction.

5. **Modeling Strategy Selection.** For each of the following applications, select the appropriate modeling strategy and justify your choice:

   (a) Real-time control of a prosthetic knee during walking

   (b) Predicting ACL stress during a cutting maneuver in soccer

   (c) Optimizing a sprinter's start to minimize 10-m time

   (d) Understanding how the golf swing generates clubhead speed

6. **(Programming.)** Implement a function that computes the total gravitational potential energy of a multi-segment body given segment properties and joint angles. Test it for a 2-segment (upper arm + forearm) model in different configurations.

7. **(Programming.)** Extend the segment properties code to compute the full 6×6 spatial inertia matrix (in the Park and Lynch spatial algebra notation) for each segment. Verify that the matrices are positive definite.

8. **(Research.)** Compare the de Leva (1996) segment parameters with at least one other source (e.g., Winter 2009, Dempster 1955). Quantify the differences in mass fractions and discuss the implications for inverse dynamics calculations.

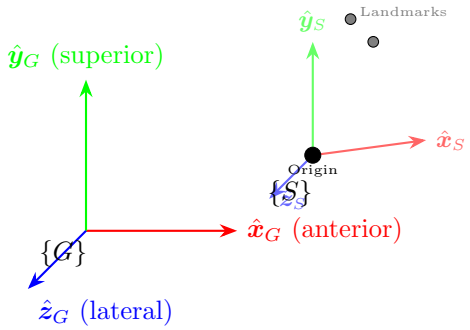
Chapter 2

Musculoskeletal Modeling Conventions

> In biomechanics, the first source of error is not the mathematics but the coordinate system.
>
> — ISB Standardization Committee

2.1 Anatomical Reference Frames

Biomechanical analysis requires a standardized language for describing body position and motion. The International Society of Biomechanics (ISB) has published recommendations for joint coordinate systems [34, 35] that serve as the international standard.



2.1.1 The Anatomical Position

All references to body position begin from the **anatomical position**: the subject stands upright, feet together, arms at sides, palms facing forward. This defines the *zero configuration* from which all joint angles are measured.

Definition 2.1. Global Reference Frame

The global (laboratory) reference frame $\{G\}$ is defined as:

- $\hat{\boldsymbol{x}}_G$: anterior (forward)
- $\hat{\boldsymbol{y}}_G$: superior (upward)

- $\hat{\boldsymbol{z}}_G$: lateral right

This is a right-handed coordinate system. Note that different communities use different conventions: biomechanics typically uses Y-up, while robotics often uses Z-up. **Always verify the convention before combining data from different sources.**

2.1.2 Segment Reference Frames

Each body segment has a local reference frame attached to it, defined by bony landmarks that can be palpated or identified from medical imaging. The ISB recommendations specify these landmarks for each segment [34, 35].

Example 2.1. The Thigh Reference Frame

The ISB-recommended thigh reference frame $\{T\}$ is defined by:

- **Origin**: midpoint between medial and lateral femoral epicondyles
- $\hat{\boldsymbol{y}}_T$: from origin to hip joint center (proximal direction)
- $\hat{\boldsymbol{z}}_T$: perpendicular to the plane formed by the two epicondyles and the hip center, pointing laterally
- $\hat{\boldsymbol{x}}_T$: cross product $\hat{\boldsymbol{y}}_T \times \hat{\boldsymbol{z}}_T$ (anterior direction)

The hip joint center is estimated from the pelvis landmarks using regression equations (e.g., [13]).

2.1.3 Connection to Screw Theory

In the language of Volume 0, each segment frame $\{S_i\}$ is related to the global frame $\{G\}$ by a homogeneous transformation matrix $T_{GS_i} \in \mathrm{SE}(3)$:

$$T_{GS_i} = \begin{bmatrix} R_{GS_i} & \boldsymbol{p}_{GS_i} \\ \mathbf{0}^T & 1 \end{bmatrix}, \tag{2.1}$$

where $R_{GS_i} \in \mathrm{SO}(3)$ is the rotation matrix and $\boldsymbol{p}_{GS_i} \in \mathbb{R}^3$ is the position of the segment origin in the global frame.

The *relative configuration* of segment j with respect to proximal segment i is:

$$T_{S_iS_j} = T_{GS_i}^{-1} T_{GS_j}. \tag{2.2}$$

Joint angles are extracted from this relative transformation.

2.2 Joint Coordinate Systems

The ISB recommends decomposing the relative rotation $R_{S_iS_j}$ into three angles using a specific Euler angle sequence for each joint. The choice of sequence is *not arbitrary*—it is

selected to align with the clinical definitions of flexion-extension, abduction-adduction, and internal-external rotation.

2.2.1 The Grood–Suntay Convention for the Knee

The knee joint coordinate system, originally proposed by Grood and Suntay (1983) [11], decomposes knee motion into:

1. **Flexion-Extension** (α): rotation about the femoral medio-lateral axis (the "fixed" axis of the femur)
2. **Abduction-Adduction** (β): rotation about the "floating" axis (perpendicular to both the femoral and tibial body-fixed axes)
3. **Internal-External Rotation** (γ): rotation about the tibial longitudinal axis (the "body-fixed" axis of the tibia)

This is a ZXY Euler angle sequence when expressed in the Grood–Suntay convention. The key insight is that the floating axis is neither fixed in the femur nor in the tibia—it is defined by the cross product of the other two axes.

```python
import numpy as np
from typing import Tuple

def grood_suntay_knee_angles(
    R_femur: np.ndarray,
    R_tibia: np.ndarray
) -> Tuple[float, float, float]:
    """Compute knee joint angles using Grood-Suntay convention.

    Parameters
    ----------
    R_femur : np.ndarray, shape (3, 3)
        Rotation matrix of femur segment frame in global coordinates.
    R_tibia : np.ndarray, shape (3, 3)
        Rotation matrix of tibia segment frame in global coordinates.

    Returns
    -------
    Tuple[float, float, float]
        (flexion_deg, adduction_deg, internal_rotation_deg)

    References
    ----------
    Grood, E.S. and Suntay, W.J. (1983). A joint coordinate system
    for the clinical description of three-dimensional motions.
    J. Biomechanical Engineering, 105(2), 136-144.
    """
    # Femoral medio-lateral axis (e1) - flexion axis
    e1 = R_femur[:, 2]  # lateral axis of femur

    # Tibial longitudinal axis (e3) - rotation axis
    e3 = R_tibia[:, 1]  # proximal axis of tibia

    # Floating axis (e2) - perpendicular to both
    e2 = np.cross(e3, e1)
    e2 = e2 / np.linalg.norm(e2)

```

```python
    # Flexion-extension angle
    # Angle between femur anterior axis and floating axis
    femur_anterior = R_femur[:, 0]
    flexion = np.arctan2(
        np.dot(e2, R_femur[:, 1]),
        np.dot(e2, femur_anterior)
    )

    # Abduction-adduction angle
    # Angle between e1 and e3 minus 90 degrees
    sin_abd = -np.dot(e1, e3)
    cos_abd = np.linalg.norm(np.cross(e1, e3))
    adduction = np.arcsin(np.clip(sin_abd, -1.0, 1.0))

    # Internal-external rotation
    # Angle between floating axis and tibia anterior axis
    tibia_anterior = R_tibia[:, 0]
    internal_rot = np.arctan2(
        np.dot(e2, R_tibia[:, 2]),
        np.dot(e2, tibia_anterior)
    )

    return (
        np.degrees(flexion),
        np.degrees(adduction),
        np.degrees(internal_rot)
    )
```

Listing 2.1: Computing knee joint angles using the Grood-Suntay convention.

2.2.2 Multi-Representation Joint Angles

Following the multi-representation philosophy of this series, we express joint rotations in all four standard representations:

1. **Euler Angles (ISB Convention):**

$$R_{ij} = R_z(\alpha)\, R_x(\beta)\, R_y(\gamma), \tag{2.3}$$

where α, β, γ are the clinical angles (flexion, adduction, rotation).

2. **Rotation Matrix (SO(3)):**

$$R_{ij} = \exp([\boldsymbol{\omega}]_\times \theta) \in \mathrm{SO}(3), \tag{2.4}$$

which avoids gimbal lock but requires 9 parameters with 6 constraints.

3. **Unit Quaternion:**

$$\boldsymbol{q} = (\cos\tfrac{\theta}{2},\ \boldsymbol{\omega}\sin\tfrac{\theta}{2}), \quad \|\boldsymbol{q}\| = 1, \tag{2.5}$$

which is singularity-free and computationally efficient.

4. **Screw Axis (Primary):**

$$T_{ij} = e^{[\mathcal{S}]\theta} \in \mathrm{SE}(3), \quad \mathcal{S} = (\boldsymbol{\omega}, \boldsymbol{v}) \in \mathbb{R}^6, \tag{2.6}$$

which captures the coupled rotation-translation of biological joints naturally.

Biology vs. Engineering 2.1. Why Screw Axis Theory Is Natural for Biological Joints

Unlike engineering revolute joints (pure rotation about a fixed axis), biological joints exhibit *coupled* rotation and translation. The knee slides anteriorly as it flexes; the glenohumeral joint translates inferiorly during abduction. Screw axis theory captures this coupling naturally: every rigid-body motion is a rotation about and translation along a screw axis. The instantaneous screw axis (ISA) of a biological joint moves through space as the joint moves, unlike the fixed axis of an ideal revolute.
This is precisely why Park and Lynch's *Modern Robotics* framework is preferred: it handles coupled rotation-translation without special cases.

2.3 Body Segment Parameters

2.3.1 Regression Equations

The most widely used body segment parameter (BSP) data comes from:

1. **Dempster (1955)**: cadaver study of 8 elderly males. Still widely cited but limited demographic range.
2. **Zatsiorsky & Seluyanov (1983, 1990)**: gamma-ray scanning of 100 living young males. More representative but uses different segment definitions.
3. **de Leva (1996)**: adjustment of Zatsiorsky data to standard segment definitions. The current gold standard for adult male and female BSP.
4. **Pavol et al. (2002)**: elderly adult data.
5. **Jensen (1986, 1989)**: pediatric data.

2.3.2 Computing the Mass Matrix

Given segment inertial properties, the mass matrix $M(\boldsymbol{q}) \in \mathbb{R}^{n\times n}$ of the multi-body system is computed using the composite rigid-body algorithm (Volume 0, Chapter 10). In the body-fixed frame of segment i, the spatial inertia is:

$$\mathcal{G}_i = \begin{bmatrix} I_i & m_i[\boldsymbol{c}_i]_\times^T \\ m_i[\boldsymbol{c}_i]_\times & m_i\mathbb{I}_3 \end{bmatrix} \in \mathbb{R}^{6\times 6}, \tag{2.7}$$

where I_i is the 3×3 rotational inertia about the segment frame origin, m_i is the segment mass, $\boldsymbol{c}_i$ is the center of mass position in the segment frame, and $[\cdot]_\times$ denotes the skew-symmetric matrix.

Exercises

1. **Coordinate System Verification.** Verify that the Grood–Suntay knee joint coordinate system produces zero angles when the femur and tibia reference frames are aligned. Test with known rotations.

2. **Euler Angle Singularity.** For the ZXY Euler angle sequence used in the knee, identify the gimbal lock configuration. Is this configuration ever reached during normal gait? During deep squatting?

3. **Multi-Representation Conversion.** Write Python functions to convert between all four representations (Euler ZXY, rotation matrix, quaternion, screw axis) for a knee joint angle of 60° flexion, 5° adduction, 10° internal rotation.

4. **BSP Sensitivity.** Using the de Leva segment properties, compute the knee extension torque required to hold the lower leg horizontal (isometric). Repeat using the Dempster data. What is the percentage difference? Discuss the implications for inverse dynamics accuracy.

5. **(Programming.)** Implement the spatial inertia matrix computation for a 3-segment leg model (thigh, shank, foot). Verify positive-definiteness.

6. **(Programming.)** Using motion capture data (or simulated data), compute knee joint angles from segment orientations using the Grood–Suntay code provided. Plot flexion angle over one gait cycle.

Chapter 3

Muscle Models

> A muscle is not a motor. It is a spring with a memory, a damper with an agenda, and a force generator with a personality.

3.1 The Hill-Type Muscle Model

The Hill-type muscle model, originating from A.V. Hill's Nobel Prize-winning work on muscle thermodynamics and muscle energetics [14, 15], is the standard lumped-parameter model used in biomechanical simulation. It represents the musculotendon unit as a combination of contractile, elastic, and damping elements. Modern implementations include refinements by [29] and [21].

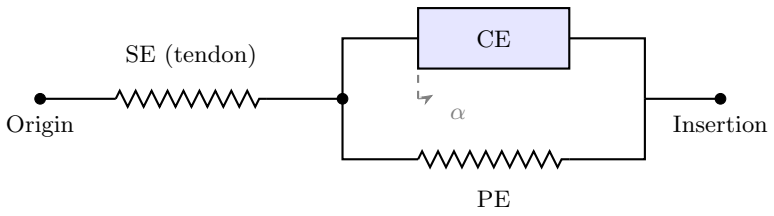
3.1.1 Model Architecture

The standard Hill-type musculotendon model consists of three elements:

1. **Contractile Element (CE)**: represents the active force-generating capacity of the muscle fibers. Its force depends on activation $a(t)$, fiber length ℓ^M, and fiber velocity $v^M = \dot{\ell}^M$.

2. **Parallel Elastic Element (PE)**: represents the passive elastic properties of the muscle belly (titin filaments, connective tissue). Produces force only when the muscle is stretched beyond its resting length.

3. **Series Elastic Element (SE)**: represents the tendon, which is in series with the contractile element. Modeled as a nonlinear spring with a slack length ℓ_T^s.



3.1.2 The Force-Length Relationship

The isometric (zero velocity) force of the contractile element depends on the normalized fiber length $\tilde{\ell} = \ell^M / \ell_0^M$:

Definition 3.1. Active Force-Length Curve

The normalized active force-length relationship is:

$$f_L(\tilde{\ell}) = \exp\left(-\frac{(\tilde{\ell} - 1)^2}{\gamma}\right), \tag{3.1}$$

where $\gamma \approx 0.45$ [29] controls the width of the curve. Maximum force ($f_L = 1$) occurs at optimal fiber length ($\tilde{\ell} = 1$). Force decreases for both shorter and longer fiber lengths, reaching near-zero at approximately $\tilde{\ell} = 0.5$ and $\tilde{\ell} = 1.5$.

A note on the numerical parameters. The values $\gamma = 0.45$, $A_f = 0.25$, $k^{PE} = 4.0$, and $\varepsilon_0 = 0.6$ used throughout this chapter are the default parameters recommended by Thelen [29] for the "generic" adult Hill-type muscle, and also serve as the baseline for the Millard–Uchida formulation [21]. They are calibrated against mid-thigh muscles of young adults and should be interpreted as a starting point, not as universal constants. Real muscles exhibit systematic variation with age, sex, training status, and fiber-type composition, and subject-specific tuning is almost always required for quantitative predictions.

The physical basis for the force-length relationship is the sliding filament theory [17]: at the optimal length, the maximum number of actin-myosin cross-bridges can form. At shorter lengths, filament overlap reduces the number of available binding sites. At longer lengths, the filaments pull apart.

```python
import numpy as np
from dataclasses import dataclass
from typing import Tuple

@dataclass
class MuscleParameters:
    """Parameters for a Hill-type musculotendon model."""
    F0: float           # Maximum isometric force (N)
    l0_M: float         # Optimal fiber length (m)
    v0: float           # Maximum contraction velocity (m/s)
    l_T_slack: float    # Tendon slack length (m)
    alpha0: float       # Pennation angle at optimal length (rad)
    # Defaults below are the Thelen (2003) generic-adult values.
    # See: Thelen, D.G. (2003) "Adjustment of muscle mechanics
    # model parameters to simulate dynamic contractions in older
    # adults," J. Biomech. Eng. 125(1), 70-77 [Thelen2003].
    gamma: float = 0.45  # Force-length curve width (Thelen 2003)
    Af: float = 0.25     # Force-velocity shape factor (Thelen 2003)
    kPE: float = 4.0     # Passive elastic stiffness (Thelen 2003)
    e0: float = 0.6      # Passive strain at F0 (Thelen 2003)

    @property
    def v_max(self) -> float:
        """Maximum shortening velocity in fiber lengths/s."""
```

```
        return self.v0 / self.l0_M


def force_length_active(l_tilde: float, gamma: float = 0.45) -> float:
    """Active force-length relationship.

    Parameters
    ----------
    l_tilde : float
        Normalized fiber length (l_M / l0_M).
    gamma : float
        Width parameter of the Gaussian curve.

    Returns
    -------
    float
        Normalized active force (0 to 1).

    References
    ----------
    Thelen, D.G. (2003). Adjustment of muscle mechanics model
    parameters to simulate dynamic contractions in older adults.
    J. Biomechanical Engineering, 125(1), 70-77.
    """
    return np.exp(-((l_tilde - 1.0) ** 2) / gamma)


def force_length_passive(l_tilde: float, kPE: float = 4.0,
                         e0: float = 0.6) -> float:
    """Passive force-length relationship (exponential model).

    Parameters
    ----------
    l_tilde : float
        Normalized fiber length.
    kPE : float
        Stiffness parameter.
    e0 : float
        Strain at which passive force equals F0.

    Returns
    -------
    float
        Normalized passive force.
    """
    if l_tilde <= 1.0:
        return 0.0
    return (np.exp(kPE * (l_tilde - 1.0) / e0 - kPE) - 1.0) / \
           (np.exp(kPE) - 1.0)


def force_velocity(v_tilde: float, a: float, Af: float = 0.25,
                   Flen: float = 1.4) -> float:
    """Force-velocity relationship (Thelen 2003 model).

    Parameters
    ----------
    v_tilde : float
        Normalized fiber velocity (v_M / v_max).
        Negative = shortening (concentric).
```

```
    a : float
        Activation level (0 to 1).
    Af : float
        Shape factor for concentric portion.
    Flen : float
        Maximum normalized eccentric force.

    Returns
    -------
    float
        Normalized force multiplier.
    """
    if v_tilde <= 0:
        # Concentric (shortening)
        return (1.0 + v_tilde) / (1.0 - v_tilde / (Af * a + 0.01))
    else:
        # Eccentric (lengthening)
        return (1.0 + v_tilde * Flen / (0.04 + a))  / \
               (1.0 + v_tilde / (0.04 + a))


def muscle_force(
    activation: float,
    l_M: float,
    v_M: float,
    params: MuscleParameters
) -> Tuple[float, float, float]:
    """Compute total musculotendon force.

    Parameters
    ----------
    activation : float
        Neural activation level (0 to 1).
    l_M : float
        Current muscle fiber length (m).
    v_M : float
        Current muscle fiber velocity (m/s, negative = shortening).
    params : MuscleParameters
        Muscle parameters.

    Returns
    -------
    Tuple[float, float, float]
        (total_force, active_force, passive_force) in Newtons.
    """
    # Normalize
    l_tilde = l_M / params.l0_M
    v_tilde = v_M / params.v0

    # Force components
    f_active = (activation
                * force_length_active(l_tilde, params.gamma)
                * force_velocity(v_tilde, activation, params.Af))
    f_passive = force_length_passive(l_tilde, params.kPE, params.e0)

    # Pennation angle varies with fiber length under the
    # constant-thickness assumption: sin(alpha) = sin(alpha0) * l0_M / l_M
    sin_alpha = np.sin(params.alpha0) * params.l0_M / max(l_M, 1e-6)
    cos_alpha = np.sqrt(max(0.0, 1.0 - sin_alpha ** 2))

```

```
    # Total force along tendon
    F_active = f_active * params.F0 * cos_alpha
    F_passive = f_passive * params.F0 * cos_alpha
    F_total = F_active + F_passive

    return F_total, F_active, F_passive


# Example: Biceps brachii
biceps = MuscleParameters(
    F0=624.3,         # N (Holzbaur et al. 2005, upper-extremity model)
    l0_M=0.1157,      # m
    v0=10 * 0.1157,   # 10 fiber lengths/s
    l_T_slack=0.2723, # m
    alpha0=0.0        # parallel-fibered
)

# Compute force at optimal length, half activation, isometric
F_total, F_active, F_passive = muscle_force(
    activation=0.5,
    l_M=biceps.l0_M,   # optimal length
    v_M=0.0,           # isometric
    params=biceps
)
print(f"Biceps at 50% activation, optimal length, isometric:")
print(f"  Active force:  {F_active:.1f} N")
print(f"  Passive force: {F_passive:.1f} N")
print(f"  Total force:   {F_total:.1f} N")
```

Listing 3.1: Hill-type muscle model implementation.

3.1.3 The Force-Velocity Relationship

The second critical property of the contractile element is the force-velocity relationship, first characterized by Hill (1938) [14].

Biomechanical Principle 3.1. The Force-Velocity Asymmetry

The force-velocity relationship characterizes how muscle force depends on contraction velocity [14, 15]:

- **Concentric (shortening) contraction**: force *decreases* with increasing shortening velocity. At maximum shortening velocity v_0, the muscle produces zero force. The relationship is well-approximated by a hyperbolic curve.
- **Eccentric (lengthening) contraction**: force *increases* beyond the maximum isometric force. The magnitude of enhancement is muscle and fiber-type dependent, documented in [29] and similar references.
- **Isometric** ($v = 0$): force equals the product of activation and the force-length relationship.

The asymmetry between concentric and eccentric force production is a fundamental property of biological muscle that is absent in engineered linear actuators.

3.1.4 Activation Dynamics

The neural command from the central nervous system does not instantly produce muscle force. The activation dynamics models the excitation-contraction coupling:

$$\dot{a}(t) = \frac{u(t) - a(t)}{\tau(u, a)}, \tag{3.2}$$

where $u(t) \in [0, 1]$ is the neural excitation, $a(t) \in [0, 1]$ is the activation level, and τ is a time constant that differs for activation and deactivation:

$$\tau(u, a) = \begin{cases} \tau_{\text{act}}(0.5 + 1.5a) & \text{if } u > a \text{ (activation)}, \\ \tau_{\text{deact}}/(0.5 + 1.5a) & \text{if } u \leq a \text{ (deactivation)}. \end{cases} \tag{3.3}$$

Typical values: $\tau_{\text{act}} = 10$–15 ms, $\tau_{\text{deact}} = 40$–60 ms [32]. The asymmetry between activation and deactivation time constants has important consequences for motor control: it is much easier to "turn on" a muscle quickly than to "turn it off."

3.2 Tendon Mechanics

The tendon connects the muscle belly to bone. It is modeled as a nonlinear elastic element with a resting (slack) length ℓ_T^s:

$$F_T(\varepsilon_T) = \begin{cases} 0 & \varepsilon_T \leq 0, \\ F_0 \cdot \frac{e^{k_T \varepsilon_T} - 1}{e^{k_T \varepsilon_{0T}} - 1} & \varepsilon_T > 0, \end{cases} \tag{3.4}$$

where $\varepsilon_T = (\ell_T - \ell_T^s)/\ell_T^s$ is the tendon strain and $\varepsilon_{0T} \approx 0.033$ is the strain at maximum isometric force. The tendon stiffness parameter k_T is typically chosen so that the tendon strain is approximately 3.3% at F_0 [36].

Biomechanical Principle 3.2. Tendons as Energy Storage Elements

Biological tendons are not simply rigid force transmitters. They are viscoelastic elements that can store and return elastic strain energy. The energy storage capacity of a tendon is determined by its stiffness and the strain it undergoes during muscle contraction. This energy storage changes the energetics and the effective impedance of the muscle-tendon unit compared to a model with infinitely stiff tendons. The physiological consequence is that tendons enable more efficient movement by recovering elastic energy that would otherwise be dissipated, as documented in [2, 3].

3.3 Musculotendon Equilibrium

The muscle fiber and tendon must satisfy force equilibrium at all times. Assuming the muscle fiber acts at a pennation angle α to the tendon line of action:

$$F_T = (F_{\text{CE}} + F_{\text{PE}}) \cos \alpha, \tag{3.5}$$

where:

$$F_{\text{CE}} = a \cdot f_L(\tilde{\ell}^M) \cdot f_V(\tilde{v}^M) \cdot F_0, \tag{3.6}$$

$$F_{\text{PE}} = f_{\text{PE}}(\tilde{\ell}^M) \cdot F_0. \tag{3.7}$$

The musculotendon length ℓ^{MT} is the sum of tendon length and the projection of fiber length:

$$\ell^{MT} = \ell_T + \ell^M \cos\alpha. \tag{3.8}$$

Given a known musculotendon length (from the skeletal geometry) and activation, the equilibrium condition (3.5) and the constraint (3.8) determine the fiber length and velocity.

3.4 The Complete Hill Model as a Dynamical System

Combining all elements, the Hill muscle model can be expressed as a dynamical system with two state variables per muscle:

$$\dot{a} = \frac{u - a}{\tau(u, a)}, \tag{3.9}$$

$$\dot{\ell}^M = v^M(a, \ell^M, \ell^{MT}), \tag{3.10}$$

where v^M is obtained by inverting the force-velocity relationship at the equilibrium fiber force. The input is the neural excitation $u(t) \in [0, 1]$ and the "disturbance" is the musculotendon path length $\ell^{MT}(\boldsymbol{q})$, which depends on the joint configuration.

Biomechanical Principle 3.3. Muscle as a Nonlinear Control System

Each muscle is itself a nonlinear dynamical system with:

- **State**: $(a, \ell^M) \in [0, 1] \times \mathbb{R}_{>0}$
- **Input**: neural excitation $u \in [0, 1]$
- **Output**: tendon force F_T
- **Disturbance**: musculotendon path length $\ell^{MT}(\boldsymbol{q})$

The muscle transforms a scalar neural command into a scalar force, but the transformation is highly nonlinear, state-dependent, and history-dependent. The entire musculoskeletal system is a cascade: neural commands → muscle dynamics → tendon forces → joint torques → skeletal dynamics.

Exercises

1. **Force-Length Curve.** Plot the active force-length curve for $\gamma \in \{0.3, 0.45, 0.6\}$. For each, determine the "useful range" (the interval of fiber lengths where $f_L > 0.5$). How does the width parameter affect the operating range?

2. **Force-Velocity Curve.** Plot the complete force-velocity curve (both concentric and eccentric) for the biceps parameters given in the code listing. Mark the maximum eccentric force and the maximum shortening velocity.

3. **Activation Dynamics.** Simulate the activation dynamics for a square-wave neural excitation input ($u = 1$ for 200 ms, then $u = 0$). Plot the activation response with $\tau_{\text{act}} = 15$ ms and $\tau_{\text{deact}} = 50$ ms. How long does it take for activation to reach 95% of maximum? How long for deactivation?

4. **Tendon Compliance.** For the Achilles tendon (slack length $\ell_T^s = 0.25$ m, max force 4000 N), compute the tendon stretch at 50%, 75%, and 100% of maximum force. Does the tendon behave approximately linearly over this range?

5. **Isometric Force Surface.** Create a 3D surface plot of muscle force as a function of normalized fiber length ($\tilde{\ell}$) and activation (a) at zero velocity (isometric). Identify the point of maximum force production.

6. **(Programming.)** Implement a complete musculotendon simulation: given a prescribed musculotendon length trajectory $\ell^{MT}(t)$ and neural excitation $u(t)$, integrate the state equations (3.9) and (3.10) and plot the resulting tendon force $F_T(t)$.

7. **(Programming.)** Implement a two-muscle antagonist pair (e.g., biceps and triceps) acting on a single-DOF elbow joint. Given co-contraction ($u_{\text{biceps}} = u_{\text{triceps}} = 0.5$), compute the net joint torque and the joint stiffness. Verify that co-contraction increases stiffness without changing the equilibrium position.

8. **(Research.)** Compare the Thelen (2003) muscle model with the Millard et al. (2013) model used in OpenSim 4.x. What are the key differences in the force-velocity and passive force models? Which is more physiologically accurate?

Chapter 4

Joint Kinematics and Soft Tissue

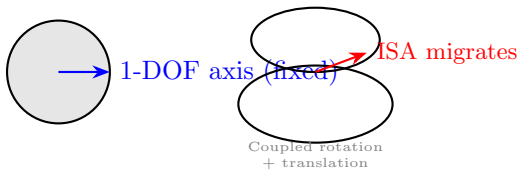
Draft Notice. This chapter is under active development and contains only a structural outline with preliminary material. Full exposition, worked examples, proofs, and exercises will be added in future editions.

The knee is not a hinge. The shoulder is not a ball-and-socket. These are useful fictions that biology tolerates but does not enforce.

4.1 Beyond Ideal Joints

Engineering joints are designed to constrain motion to specific degrees of freedom: a revolute joint permits one rotation, a prismatic joint permits one translation. Biological joints achieve constraint through a fundamentally different mechanism: the interaction of articular surfaces, ligaments, joint capsule, and muscle forces. This section develops the kinematics of biological joints using the screw theory framework of Volume 0, extended to handle the additional complexity.

Engineering Revolute Biological Joint



4.1.1 The Instantaneous Screw Axis

Every rigid-body motion can be decomposed into a rotation about and translation along a **screw axis** (Chasles' theorem, Volume 0, Chapter 5). For an engineering revolute joint, this screw axis is fixed in both bodies. For a biological joint, the instantaneous screw axis (ISA) *moves* as the joint configuration changes.

Definition 4.1. Instantaneous Screw Axis of a Biological Joint

Given the relative twist $\mathcal{V}_{ij} = (\boldsymbol{\omega}, \boldsymbol{v}) \in \mathbb{R}^6$ between adjacent segments i and j, the instantaneous screw axis has:

- Direction: $\hat{\boldsymbol{s}} = \boldsymbol{\omega} / \|\boldsymbol{\omega}\|$
- Pitch: $h = \boldsymbol{\omega} \cdot \boldsymbol{v} / \|\boldsymbol{\omega}\|^2$
- Location: $\boldsymbol{q} = \boldsymbol{\omega} \times \boldsymbol{v} / \|\boldsymbol{\omega}\|^2$

For a pure rotation ($h = 0$), the ISA passes through $\boldsymbol{q}$ in direction $\hat{\boldsymbol{s}}$. Tracking how $\boldsymbol{q}$ and $\hat{\boldsymbol{s}}$ change with joint angle reveals the coupled kinematics of the biological joint.

4.1.2 The Knee: Rolling, Sliding, and Coupled Rotation

The tibiofemoral joint exemplifies the complexity of biological kinematics. During flexion from 0° to 90°:

1. The femoral condyles **roll** on the tibial plateaus (like a wheel on a road)
2. They simultaneously **slide** posteriorly (the contact point moves backward)
3. The tibia undergoes coupled **internal rotation** of approximately 15° (the "screw-home" mechanism)
4. The contact area shifts, changing the effective moment arm

This coupled motion means the knee ISA migrates superiorly and posteriorly during flexion. The position and velocity of the ISA are key outputs of kinematic analysis [7, 33]. The four-bar linkage model provides a simplified but insightful mechanical analog for understanding coupled knee kinematics [24]:

```python
import numpy as np
from typing import Tuple

def compute_isa(
    omega: np.ndarray,
    v: np.ndarray
) -> Tuple[np.ndarray, np.ndarray, float]:
    """Compute the instantaneous screw axis from a twist.

    Parameters
    ----------
    omega : np.ndarray, shape (3,)
        Angular velocity vector.
    v : np.ndarray, shape (3,)
        Linear velocity of the body-fixed frame origin.

    Returns
    -------
    Tuple[np.ndarray, np.ndarray, float]
        (point_on_axis, axis_direction, pitch)
    """
    omega_norm = np.linalg.norm(omega)
    if omega_norm < 1e-10:
```

```
        # Pure translation
        direction = v / np.linalg.norm(v)
        return np.zeros(3), direction, np.inf

    direction = omega / omega_norm
    pitch = np.dot(omega, v) / omega_norm**2
    point = np.cross(omega, v) / omega_norm**2

    return point, direction, pitch


def knee_isa_trajectory(
    flexion_angles: np.ndarray,
    roll_slide_ratio: float = 0.6
) -> np.ndarray:
    """Estimate knee ISA position during flexion.

    Simple model: ISA migrates posteriorly and superiorly
    with flexion, governed by the roll-to-slide ratio.

    Parameters
    ----------
    flexion_angles : np.ndarray
        Array of flexion angles in degrees.
    roll_slide_ratio : float
        Ratio of rolling to total contact displacement.
        Pure rolling = 1.0, pure sliding = 0.0.

    Returns
    -------
    np.ndarray, shape (N, 3)
        ISA positions in the tibial reference frame.
    """
    R_condyle = 0.030  # approximate femoral condyle radius (m)
    isa_positions = np.zeros((len(flexion_angles), 3))

    for i, theta in enumerate(np.radians(flexion_angles)):
        # Posterior migration due to rolling
        posterior = R_condyle * theta * roll_slide_ratio
        # Superior migration (ISA moves up with flexion)
        superior = R_condyle * (1.0 - np.cos(theta * 0.5))

        isa_positions[i] = [-posterior, superior, 0.0]

    return isa_positions
```

Listing 4.1: Instantaneous screw axis computation for a biological joint.

4.1.3 The Shoulder: The Most Complex Joint

The glenohumeral joint has the largest range of motion of any joint in the body, achieved at the expense of stability. The "ball-and-socket" description is a gross simplification: quantitative measurements show the humeral head translates during abduction [18, 10], and the instantaneous center of rotation shifts continuously. The instantaneous screw axis of the shoulder is not fixed and migrates through the joint during arm motion [12].

The shoulder complex involves four joints:

1. **Glenohumeral** (GH): primary ball-and-socket, 3 DOF rotation + ∼3 mm translation

2. **Acromioclavicular** (AC): 3 DOF rotation, small range
3. **Sternoclavicular** (SC): 3 DOF rotation, small range
4. **Scapulothoracic** (ST): not a true synovial joint; the scapula glides on the thorax, constrained as a *surface contact* rather than a pin joint

The scapulothoracic "joint" is particularly interesting from a mathematical perspective: it is a *holonomic constraint* that the scapula must remain on the thoracic surface, effectively acting as a 2-DOF constraint (the scapula can translate on the surface but cannot lift off).

4.2 Ligament Models

Ligaments constrain joint motion by resisting excessive displacement. They are modeled as nonlinear springs with:

$$F_{\text{lig}}(\varepsilon) = \begin{cases} 0 & \varepsilon \le 0 \text{ (slack)}, \\ k_1 \varepsilon^2 / (4\varepsilon_1) & 0 < \varepsilon \le 2\varepsilon_1 \text{ (toe region)}, \\ k_2(\varepsilon - \varepsilon_1) & \varepsilon > 2\varepsilon_1 \text{ (linear region)}, \end{cases} \tag{4.1}$$

where $\varepsilon = (\ell - \ell_0)/\ell_0$ is the strain, ℓ_0 is the reference length, $\varepsilon_1 \approx 0.03$ is the transition strain, and k_1, k_2 are stiffness parameters [5].

4.3 Spinal Mechanics

The spine presents a unique modeling challenge: 24 vertebrae connected by intervertebral discs, each joint having 6 coupled degrees of freedom. The functional spinal unit (FSU) consisting of two adjacent vertebrae and their connecting disc is the basic mechanical unit.

The intervertebral disc consists of:

- **Nucleus pulposus**: a gel-like center that distributes compressive loads
- **Annulus fibrosus**: concentric rings of collagen fibers that resist torsion and bending

A common simplified model treats each FSU as a 6-DOF spring:

$$\boldsymbol{F}_{\text{disc}} = K_{\text{disc}} \cdot \boldsymbol{\delta}, \tag{4.2}$$

where $K_{\text{disc}} \in \mathbb{R}^{6\times 6}$ is the stiffness matrix and $\boldsymbol{\delta} \in \mathbb{R}^6$ is the generalized displacement (3 translations + 3 rotations).

Exercises

1. **ISA Computation.** For a knee flexing at 60°/s with 2 mm/s of posterior translation, compute the ISA direction, pitch, and location.
2. **Screw-Home Mechanism.** Plot the coupled internal rotation of the tibia as a function of knee flexion using the model: $\gamma = 15^\circ \cdot (1 - \cos(\theta/2))$ for $\theta \in [0, 90^\circ]$.

3. **Shoulder Kinematics.** Using screw axis theory, express the configuration of the hand (end-effector) as a product of exponentials through the SC, AC, GH, and elbow joints.

4. **(Programming.)** Implement the ligament force model and plot the force-strain curve. Identify the toe region and the linear region.

5. **(Programming.)** Create a 3D visualization of the knee ISA trajectory during flexion from 0° to 120°.

Chapter 5

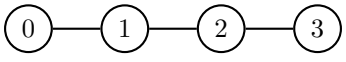
Multi-Body Dynamics of Biological Chains

Draft Notice. This chapter is under active development and contains only a structural outline with preliminary material. Full exposition, worked examples, proofs, and exercises will be added in future editions.

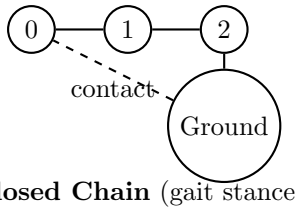
The human body is a high-dimensional articulated system with changing contacts and compliant tissues. Rigid-body dynamics is a useful approximation, but never the whole story.

5.1 Open-Chain vs. Closed-Chain Biomechanics

Most human movement involves open kinematic chains (the limbs) connected to a floating base (the pelvis/trunk). However, many functional activities create temporary closed chains: the stance phase of gait (foot on ground), pushing against a wall, or the golf swing when both hands grip the club.



Open Chain (arm reaching)



Closed Chain (gait stance phase)

5.1.1 Equations of Motion for Biological Chains

The equations of motion for an n-DOF musculoskeletal system take the standard Lagrangian form:

$$M(\boldsymbol{q})\ddot{\boldsymbol{q}} + C(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + g(\boldsymbol{q}) = R(\boldsymbol{q})\boldsymbol{F}_{\text{muscle}} + J_c^T(\boldsymbol{q})\boldsymbol{F}_{\text{contact}}, \tag{5.1}$$

where:

- $M(\boldsymbol{q}) \in \mathbb{R}^{n\times n}$ is the mass matrix (from segment inertial properties)
- $C(\boldsymbol{q}, \dot{\boldsymbol{q}}) \in \mathbb{R}^{n\times n}$ is the Coriolis/centrifugal matrix
- $g(\boldsymbol{q}) \in \mathbb{R}^n$ is the gravity vector
- $R(\boldsymbol{q}) \in \mathbb{R}^{n\times p}$ is the muscle moment arm matrix (p muscles, n DOF)
- $\boldsymbol{F}_{\text{muscle}} \in \mathbb{R}^p$ are the muscle forces (from Hill models, Chapter 3)
- $J_c(\boldsymbol{q})$ is the contact Jacobian
- $\boldsymbol{F}_{\text{contact}}$ are external contact forces

The key difference from Volume I's $\boldsymbol{\tau} = B\boldsymbol{u}$ is that joint torques are no longer direct control inputs. Instead:

$$\boldsymbol{\tau}_{\text{net}} = R(\boldsymbol{q})\boldsymbol{F}_{\text{muscle}} = \sum_{i=1}^{p} \boldsymbol{r}_i(\boldsymbol{q}) F_{\text{muscle},i}, \tag{5.2}$$

where $\boldsymbol{r}_i(\boldsymbol{q}) \in \mathbb{R}^n$ is the i-th column of R, giving the moment arm vector of muscle i across all joints. This moment-arm mapping is the standard bridge between muscle models and generalized coordinates in musculoskeletal simulation [36, 9].

5.1.2 Bi-articular Muscles

Unlike engineering actuators that span a single joint, many muscles cross two or more joints. These **bi-articular** (or multi-articular) muscles create mechanical coupling between joints that is not captured by the simple one-actuator-per-joint abstraction common in introductory robotics.

Biomechanical Principle 5.1. The Coordination Function of Bi-articular Muscles

Bi-articular muscles serve two distinct functions:

1. **Energy transfer**: they can transport mechanical energy from one joint to another without requiring a separate distal actuator. When the proximal joint extends (muscle shortening at the proximal end) and the distal joint flexes (muscle lengthening at the distal end), mechanical power can be redistributed from proximal to distal coordinates [36, 22]. The metabolic cost still depends on activation, contraction velocity, and force history.
2. **Coordination constraint**: they link joint motions, reducing the effective degrees of freedom and simplifying the control problem.

This is a form of *mechanical intelligence*—the morphology of the musculoskeletal system performs part of the control task. The gastrocnemius, for example, crosses both the knee and ankle, coupling knee extension to ankle plantarflexion during the push-off phase of gait.

Control-Theoretic Perspective: The matrix $R(\boldsymbol{q})$ can be read as a state-dependent input map from muscle forces to generalized torques. Bi-articular muscles may create dependencies among its columns, which is one reason the effective control directions in a musculoskeletal model can be lower-dimensional than the raw muscle count. Geometric control language is useful for describing that reduction [1], but the specific structure of $R(\boldsymbol{q})$ remains anatomy- and model-dependent.

```python
import numpy as np

def compute_joint_torques(
    moment_arms: np.ndarray,
    muscle_forces: np.ndarray
) -> np.ndarray:
    """Compute net joint torques from muscle forces and moment arms.

    Parameters
    ----------
    moment_arms : np.ndarray, shape (n_joints, n_muscles)
        Moment arm matrix R(q). Each column gives the moment arm
        of one muscle across all joints. Sign convention: positive
        moment arm produces positive joint torque (typically flexion).
    muscle_forces : np.ndarray, shape (n_muscles,)
        Force produced by each muscle (always positive).

    Returns
    -------
    np.ndarray, shape (n_joints,)
        Net joint torques.
    """
    return moment_arms @ muscle_forces


# Example: 2-DOF elbow system with 4 muscles
# Joints: shoulder flexion, elbow flexion
# Muscles: anterior deltoid, biceps (biarticular),
#          triceps (biarticular), brachialis
#
# Convention: R has shape (n_joints, n_muscles) so that
# tau = R @ F maps the muscle-force vector to a joint-torque
# vector via a single matrix-vector product. Each row is a joint
# and each column is a muscle. This is the shape that
# compute_joint_torques() expects.
R = np.array([
    # deltoid, biceps, triceps, brachialis
    [0.05,  0.03,  -0.025,  0.0 ],  # row 0: shoulder moment arms (m)
    [0.0,   0.04,  -0.020,  0.03],  # row 1: elbow moment arms (m)
])  # Shape: (2 joints, 4 muscles) -- ready to use as-is

# Note: biceps has moment arms at BOTH joints (bi-articular)
muscle_forces = np.array([200.0, 150.0, 100.0, 120.0])  # Newtons

tau = compute_joint_torques(R, muscle_forces)
```

```
print(f"Shoulder torque: {tau[0]:.1f} N*m")
print(f"Elbow torque:    {tau[1]:.1f} N*m")
```

Listing 5.1: Computing joint torques from muscle forces through moment arms.

5.1.3 Co-Contraction and Joint Stiffness

When antagonist muscles co-contract (both agonist and antagonist are active simultaneously), the net joint torque may be zero but the joint **stiffness** increases. This is because each active muscle acts as a spring:

$$K_{\text{joint}} = \sum_{i=1}^{p} r_i^2(\boldsymbol{q}) \cdot \frac{\partial F_{\text{muscle},i}}{\partial \ell_i^M}, \tag{5.3}$$

where $\partial F / \partial \ell^M$ is the short-range stiffness of the muscle, which depends on activation level.

Co-contraction is metabolically expensive but biomechanically important: it allows the nervous system to modulate joint *impedance* independently of net joint *torque*. This is better understood as a biologically compliant analogue of impedance control rather than a literal duplicate of a rigid torque source [36, 21].

Position-dependent stiffness modulation gives a useful control vocabulary for discussing state-dependent actuation. Geometric control theory describes how state-dependent inputs change closed-loop structure [1], but the actual stiffness law in biomechanics still has to come from muscle, tendon, and contact modeling rather than from control theory alone.

5.2 The Mass Matrix of the Human Body

Computing the mass matrix $M(\boldsymbol{q})$ for a full musculoskeletal model follows the same recursive algorithms as Volume 0, Chapter 10 (the Composite Rigid Body Algorithm), using the segment inertial properties from Chapter 2. The key computational steps are:

1. Define the kinematic tree (segment connectivity and joint types)
2. Compute forward kinematics using the product of exponentials
3. Apply the CRBA to compute $M(\boldsymbol{q})$

For a 20-DOF model (pelvis + 2 legs + trunk + 2 arms), $M(\boldsymbol{q})$ is a 20×20 symmetric positive-definite matrix.

Exercises

1. **Bi-articular Energy Transfer.** For the gastrocnemius (crossing knee and ankle), show mathematically how knee extension can produce ankle torque through the bi-articular coupling.
2. **Co-Contraction Analysis.** Compute the joint stiffness when biceps and triceps co-contract at 50% activation with the elbow at 90°.

3. **Mass Matrix Properties.** For a 2-segment arm model, compute $M(\boldsymbol{q})$ at three configurations and verify positive-definiteness. Plot the condition number as a function of elbow angle.

4. **(Programming.)** Implement the CRBA for a 3-segment leg model using the de Leva segment properties. Compare with Pinocchio output.

5. **(Programming.)** Simulate a 2-DOF arm with 4 muscles (as in the code listing) performing a reaching movement. Use prescribed muscle activations and compute the resulting trajectory via forward dynamics.

Chapter 6

Inverse Problems in Biomechanics

Draft Notice. This chapter is under active development and contains only a structural outline with preliminary material. Full exposition, worked examples, proofs, and exercises will be added in future editions.

In biomechanics, we almost never measure forces directly. We measure motion and infer the forces that caused it.

6.1 The Inverse Dynamics Pipeline

Inverse dynamics is one of the central tools in biomechanics [31, 9]: given measured kinematics $\boldsymbol{q}(t)$, $\dot{\boldsymbol{q}}(t)$, $\ddot{\boldsymbol{q}}(t)$ and external forces $\boldsymbol{F}_{\text{ext}}(t)$, compute the net joint torques. By itself, however, the calculation stops at net generalized loads; moving from those loads to individual muscle forces requires an additional redundancy-resolution or forward-simulation model. The complete pipeline, illustrated in Figure 6.1, transforms raw experimental data into progressively more model-dependent estimates:

$$\boldsymbol{\tau}(t) = M(\boldsymbol{q})\ddot{\boldsymbol{q}} + C(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + g(\boldsymbol{q}) - J_c^T(\boldsymbol{q})\boldsymbol{F}_{\text{ext}}. \tag{6.1}$$

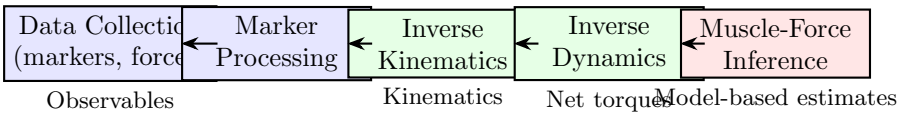


Figure 6.1: The inverse dynamics pipeline: from raw experimental data to net joint moments and subsequent model-based muscle-force estimation.

The standard pipeline comprises five stages:

1. **Data Collection**: Motion capture markers + force plates
2. **Marker Processing**: Filtering, gap-filling, labeling

3. **Inverse Kinematics**: Fit skeleton model to marker data
4. **Inverse Dynamics**: Compute joint torques from RNEA
5. **Muscle Force Estimation**: Infer candidate muscle-force distributions using optimization or controller assumptions

Stages 1–4 are the standard inverse-kinematics/inverse-dynamics pipeline. Stage 5 is an additional modeling layer: it does not follow uniquely from the measured motion and external-force data, and different optimization or control assumptions can produce different muscle-force histories for the same net joint moments.

6.1.1 Inverse Kinematics

Given marker positions $\boldsymbol{m}_i^{\text{meas}}(t) \in \mathbb{R}^3$ for $i = 1, \ldots, N_m$ markers, inverse kinematics solves:

$$\boldsymbol{q}^*(t) = \arg\min_{\boldsymbol{q}} \sum_{i=1}^{N_m} w_i \left\| \boldsymbol{m}_i^{\text{model}}(\boldsymbol{q}) - \boldsymbol{m}_i^{\text{meas}}(t) \right\|^2, \tag{6.2}$$

where $\boldsymbol{m}_i^{\text{model}}(\boldsymbol{q})$ is the predicted marker position from the skeleton model and w_i are weighting factors.

```python
import numpy as np
from scipy.optimize import minimize

def inverse_kinematics_frame(
    marker_positions_measured: np.ndarray,
    marker_positions_model_func,
    q_initial: np.ndarray,
    weights: np.ndarray | None = None
) -> np.ndarray:
    """Solve inverse kinematics for a single frame.

    Parameters
    ----------
    marker_positions_measured : np.ndarray, shape (n_markers, 3)
        Measured marker positions.
    marker_positions_model_func : callable
        Function q -> (n_markers, 3) predicted marker positions.
    q_initial : np.ndarray, shape (n_dof,)
        Initial guess for joint angles.
    weights : np.ndarray, shape (n_markers,), optional
        Per-marker weights.

    Returns
    -------
    np.ndarray, shape (n_dof,)
        Optimal joint angles.
    """
    n_markers = marker_positions_measured.shape[0]
    if weights is None:
        weights = np.ones(n_markers)

    def objective(q: np.ndarray) -> float:
        m_model = marker_positions_model_func(q)
        residuals = m_model - marker_positions_measured
        return float(np.sum(weights[:, None] * residuals**2))
```

```
36
37     result = minimize(objective, q_initial, method='L-BFGS-B')
38     return result.x
```

Listing 6.1: Simplified inverse kinematics solver.

6.1.2 Bottom-Up vs. Top-Down Inverse Dynamics

Bottom-up: start from the ground contact (where forces are measured by force plates) and propagate upward through the kinematic chain. Most accurate when force plate data is available.

Top-down: start from the most distal segment and work inward. Useful when ground reaction forces are not available (e.g., swimming, throwing).

6.2 The Muscle Redundancy Problem

Inverse dynamics gives net joint torques $\boldsymbol{\tau} \in \mathbb{R}^n$, but we want individual muscle forces $\boldsymbol{F} \in \mathbb{R}^p$ with $p > n$. The exact ratio between muscles and generalized coordinates depends on model granularity and how muscle groups are represented, but in all such cases the system is underdetermined:

$$R(\boldsymbol{q})\boldsymbol{F} = \boldsymbol{\tau}, \quad \boldsymbol{F} \geq 0. \tag{6.3}$$

From a modeling perspective, this is a constrained input-allocation problem. At a fixed configuration, the reachable torque set is the image of $R(\boldsymbol{q})$ intersected with the nonnegative muscle-force constraints. When $p > n$, multiple muscle-force vectors can produce the same joint torque history, so additional physiological or optimization assumptions are needed to select among them.

6.2.1 Static Optimization

The most common approach minimizes a cost function:

$$\min_{\boldsymbol{F}} \sum_{i=1}^{p} \left(\frac{F_i}{F_{0,i}} \right)^{\alpha} \quad \text{subject to} \quad R(\boldsymbol{q})\boldsymbol{F} = \boldsymbol{\tau},\ 0 \leq F_i \leq a_i \cdot f_L(\tilde{\ell}_i) \cdot F_{0,i}, \tag{6.4}$$

where $\alpha = 2$ (minimize sum of squared activations) or $\alpha = 3$ (minimize metabolic energy proxy). The constraint $F_i \leq a_i \cdot f_L \cdot F_{0,i}$ enforces the force-length and activation constraints from Chapter 3.

This optimization is one way to regularize the nonuniqueness of (6.3). The objective does not arise uniquely from the equations of motion; it encodes modeling priorities such as effort minimization, smoothness, or physiological plausibility. Different choices of α, reserve actuators, activation dynamics, or regularization can yield different force-sharing solutions while reproducing the same net torque history [36, 9].

6.2.2 Computed Muscle Control (CMC)

CMC [30] combines forward dynamics with feedback control. It uses a PD controller to track the measured kinematics, with the controller output being the desired muscle excitations:

$$\ddot{\boldsymbol{q}}_{\text{des}} = \ddot{\boldsymbol{q}}_{\text{meas}} + K_v(\dot{\boldsymbol{q}}_{\text{meas}} - \dot{\boldsymbol{q}}) + K_p(\boldsymbol{q}_{\text{meas}} - \boldsymbol{q}), \tag{6.5}$$

where K_v and K_p are velocity and position feedback gains.

CMC is a tracking algorithm rather than a direct measurement technique. It can produce dynamically consistent motion in a forward simulation, but its output depends on model fidelity, actuator limits, controller tuning, and how closely the measured kinematics can actually be realized by the muscle-driven model. General nonlinear-control results explain why feedback can reduce tracking error, yet they do not by themselves validate a particular muscle-force reconstruction [1, 30].

6.3 Parameter Estimation

Many biomechanical parameters (segment masses, muscle strengths, joint axis locations) are not directly measurable and must be estimated from motion data.

6.3.1 Bayesian Framework

The parameter estimation problem is naturally Bayesian:

$$p(\boldsymbol{\theta}|\boldsymbol{y}) \propto p(\boldsymbol{y}|\boldsymbol{\theta}) \cdot p(\boldsymbol{\theta}), \tag{6.6}$$

where $\boldsymbol{\theta}$ are model parameters, $\boldsymbol{y}$ are observations, $p(\boldsymbol{y}|\boldsymbol{\theta})$ is the likelihood, and $p(\boldsymbol{\theta})$ is the prior (from literature values and physical constraints).

Exercises

1. **Inverse Dynamics.** For a 2-segment pendulum model of the arm, compute the shoulder and elbow torques required during a reaching movement given $\boldsymbol{q}(t) = (\sin(\pi t/T), \pi t/(2T))$.

2. **Muscle Redundancy.** For a single-DOF elbow with 3 muscles (biceps, brachialis, brachioradialis), solve the static optimization for a given elbow torque. Show that the solution depends on the cost function exponent α.

3. **(Programming.)** Implement the static optimization for a 2-DOF, 6-muscle system. Compare $\alpha = 2$ and $\alpha = 3$ solutions.

4. **(Programming.)** Implement an inverse kinematics solver for a 3-segment arm model with 7 markers. Test with synthetic marker data.

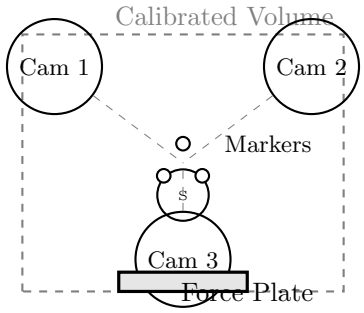
Chapter 7

Experimental Methods

Draft Notice. This chapter is under active development and contains only a structural outline with preliminary material. Full exposition, worked examples, proofs, and exercises will be added in future editions.

The quality of a biomechanical analysis is limited by the quality of the measurements, not the sophistication of the model.

7.1 Motion Capture Systems



7.1.1 Marker-Based Systems

Optoelectronic motion capture systems (Vicon, Qualisys, OptiTrack) track retroreflective markers attached to the body surface. Standard configurations use 30–80 markers placed on bony landmarks following established protocols (Plug-in-Gait, Cleveland Clinic, ISB).

Key specifications for biomechanical applications [6]:

- **Sampling rate**: 100–500 Hz for human motion (higher for impacts)
- **Spatial accuracy**: < 1 mm in calibrated volume
- **Marker size**: 9–14 mm diameter
- **Latency**: important for real-time applications (< 10 ms achievable)

It is important to note that the instrument accuracy above refers to the camera system alone. The dominant error in marker-based biomechanics is not camera noise but *soft tissue artifact*—the relative motion between skin markers and the underlying bone—discussed in Section 7.1.3 below [20].

7.1.2 Markerless Systems

Computer vision approaches (OpenPose, MediaPipe, DeepLabCut) estimate joint positions from video without physical markers. Accuracy is currently 10–30 mm, insufficient for clinical-grade inverse dynamics but useful for field measurements and large-scale studies.

```python
import numpy as np
from scipy.signal import butter, filtfilt

def process_mocap_data(
    marker_data: np.ndarray,
    sample_rate: float,
    cutoff_freq: float = 6.0,
    filter_order: int = 4
) -> tuple[np.ndarray, np.ndarray, np.ndarray]:
    """Process raw motion capture marker data.

    Parameters
    ----------
    marker_data : np.ndarray, shape (n_frames, n_markers, 3)
        Raw marker positions in meters.
    sample_rate : float
        Sampling frequency in Hz.
    cutoff_freq : float
        Low-pass filter cutoff frequency in Hz.
    filter_order : int
        Butterworth filter order.

    Returns
    -------
    tuple of np.ndarray
        (positions, velocities, accelerations)
        Each shape (n_frames, n_markers, 3).
    """
    # Design low-pass Butterworth filter
    nyquist = sample_rate / 2.0
    b, a = butter(filter_order, cutoff_freq / nyquist, btype='low')

    n_frames, n_markers, _ = marker_data.shape
    positions = np.zeros_like(marker_data)
    velocities = np.zeros_like(marker_data)
    accelerations = np.zeros_like(marker_data)

    dt = 1.0 / sample_rate

    for m in range(n_markers):
        for axis in range(3):
            # Filter position data
            positions[:, m, axis] = filtfilt(
                b, a, marker_data[:, m, axis]
            )

    # Central difference for velocities
    velocities[1:-1] = (positions[2:] - positions[:-2]) / (2 * dt)
```

```
    velocities[0] = velocities[1]
    velocities[-1] = velocities[-2]

    # Central difference for accelerations
    accelerations[1:-1] = (
        positions[2:] - 2 * positions[1:-1] + positions[:-2]
    ) / dt**2
    accelerations[0] = accelerations[1]
    accelerations[-1] = accelerations[-2]

    return positions, velocities, accelerations
```

Listing 7.1: Motion capture data processing pipeline.

7.1.3 Soft Tissue Artifact

Soft tissue artifact (STA) is the relative motion between skin-mounted markers and the underlying bone caused by deformation of skin, subcutaneous fat, and muscle during movement. STA is widely recognized as the dominant source of error in marker-based motion capture, frequently exceeding the nominal camera resolution by more than an order of magnitude [6, 25].

The magnitude of STA varies strongly with marker location, anatomical region, task, and subject anthropometry. Representative values for the lower limb are 10–30 mm of skin-marker displacement relative to bone during walking and running, with the thigh segment the worst offender and the shank the best [6, 25]. STA is not random noise: it is task-correlated and heavily dependent on muscle contraction state, which means low-pass filtering cannot remove it.

Practical mitigation strategies include:

- **Rigid marker clusters** mounted on lightweight shells, which average out local skin motion over a neighborhood of markers.
- **Optimal pose estimation**, in which the rigid body pose of a segment is obtained by least-squares fitting of a cluster of markers rather than by differencing pairs of landmarks [6].
- **Global optimization / inverse kinematics** with joint constraints, which enforces an articulated model and redistributes STA across the chain.
- **Biplanar fluoroscopy** or bone pins when the research question demands sub-millimeter bone-to-bone accuracy.

STA should always be reported as part of the uncertainty budget of any kinetic analysis; inverse dynamics is sensitive to segment orientation errors, and moderate STA can produce large errors in computed joint moments.

7.1.4 Low-Pass Filter Cutoff Selection

Choosing the cutoff frequency for the low-pass Butterworth filter in `process_mocap_data()` is not a matter of taste: a cutoff that is too low attenuates the true signal (especially accelerations, whose energy extends to higher frequencies than positions), while a cutoff that is too high leaves noise that is amplified by numerical differentiation.

The standard objective procedure is **residual analysis**, due to Winter [31]. For a range of candidate cutoffs f_c, one computes the root-mean-square residual

$$R(f_c) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\big(x_i - \hat{x}_i(f_c)\big)^2},$$

where x_i is the raw signal and $\hat{x}_i(f_c)$ is the filtered signal. Plotting $R(f_c)$ vs. f_c produces a curve that is approximately linear in the high-f_c regime (residuals reflect only noise) and rises sharply at low f_c (residuals include signal). The optimal cutoff is the intersection of the extrapolated linear noise-only segment with the R-axis, typically in the range 4–10 Hz for kinematics of whole-body locomotion and 10–20 Hz for running and impact-dominated tasks [31]. Higher cutoffs are required when downstream analysis needs accelerations or joint powers, since differentiation acts as a high-pass amplifier.

7.2 Electromyography (EMG)

EMG measures the electrical activity of muscles, providing a window into neural commands. Surface EMG (sEMG) is non-invasive; intramuscular EMG (using fine-wire or needle electrodes) targets deep muscles.

7.2.1 EMG Processing

The raw EMG signal must be processed before use:

1. **Band-pass filter**: 20–450 Hz (remove movement artifact and noise)
2. **Full-wave rectification**: $|e(t)|$
3. **Envelope extraction**: low-pass filter at 3–6 Hz
4. **Normalization**: divide by maximum voluntary contraction (MVC) value

7.3 Force Plates

Force plates measure ground reaction forces (GRF) and moments. Standard specifications:

- 6-component measurement: $F_x, F_y, F_z, M_x, M_y, M_z$
- Sampling rate: 1000–2000 Hz
- Range: 0–5000 N vertical, 0–2500 N horizontal
- Center of pressure (CoP) accuracy: < 2 mm

The center of pressure is computed from force plate outputs:

$$x_{\text{CoP}} = -\frac{M_y + F_x \cdot d_z}{F_z}, \quad y_{\text{CoP}} = \frac{M_x - F_y \cdot d_z}{F_z}, \tag{7.1}$$

where d_z is the distance from the force plate surface to the transducer plane.

7.4 Inertial Measurement Units (IMUs)

IMUs combine accelerometers, gyroscopes, and magnetometers in a compact package. They provide:

- Angular velocity: $\boldsymbol{\omega}(t) \in \mathbb{R}^3$
- Linear acceleration: $\boldsymbol{a}(t) \in \mathbb{R}^3$
- Magnetic field: $\boldsymbol{B}(t) \in \mathbb{R}^3$ (for heading)

Orientation estimation from IMU data is a state estimation problem solvable using the extended Kalman filter (Volume I, Chapter 6) or complementary filters.

7.5 Data Synchronization

Multiple measurement systems must be temporally synchronized. Best practices:

- Hardware sync: trigger all systems from a common clock pulse
- Minimum: force plate and motion capture on the same clock
- EMG-to-motion delay: typically 10–80 ms (electromechanical delay)
- Cross-correlation for post-hoc alignment when hardware sync is unavailable

Exercises

1. **Filter Selection.** Using the provided code, filter a synthetic marker trajectory with cutoff frequencies of 4, 6, 8, and 12 Hz. Plot the effect on position, velocity, and acceleration. Discuss the trade-off between noise reduction and signal distortion.
2. **CoP Computation.** Given force plate data from a vertical jump (F_z peak $= 3000$ N), compute the center of pressure trajectory during takeoff.
3. **EMG Processing.** Process a synthetic EMG signal (sum of sinusoids + white noise) through the full processing pipeline. Plot each stage.
4. **(Programming.)** Implement an IMU orientation estimator using a complementary filter. Test with synthetic gyroscope and accelerometer data.

Chapter 8

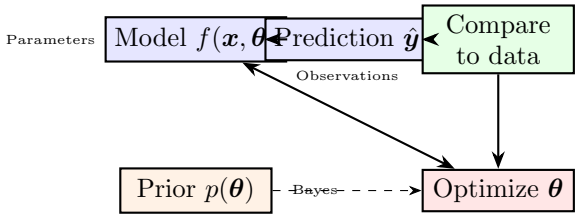
Inference on Biological Systems

Draft Notice. This chapter is under active development and contains only a structural outline with preliminary material. Full exposition, worked examples, proofs, and exercises will be added in future editions.

We cannot open the body and read the parameters. We must infer them from the traces left by motion.

8.1 Parameter Identification from Motion Data

Biological system parameters (muscle strengths, tendon slack lengths, segment inertias) cannot typically be measured directly in vivo. They must be inferred from observable motion data using optimization and estimation techniques.



8.1.1 Maximum Likelihood Estimation

Given a parameterized model $\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{\theta})$ with parameters $\boldsymbol{\theta}$ and noisy observations $\boldsymbol{y}_k = h(\boldsymbol{x}_k) + \boldsymbol{v}_k$ at times t_k, the maximum likelihood estimate is:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \sum_{k=1}^{N} \|\boldsymbol{y}_k - h(\boldsymbol{x}_k(\boldsymbol{\theta}))\|^2_{R^{-1}}\,, \tag{8.1}$$

where R is the measurement noise covariance.

8.1.2 Bayesian Estimation with Physical Constraints

The Bayesian framework (introduced in Chapter 6) is particularly valuable for biomechanical parameter estimation because it naturally incorporates prior knowledge from the literature:

```
import numpy as np
from scipy.optimize import minimize

def log_posterior(
    theta: np.ndarray,
    observations: np.ndarray,
    forward_model,
    prior_mean: np.ndarray,
    prior_cov: np.ndarray,
    noise_cov: np.ndarray
) -> float:
    """Compute the log-posterior for parameter estimation.

    Parameters
    ----------
    theta : np.ndarray
        Model parameters.
    observations : np.ndarray, shape (n_obs, n_dims)
        Observed data.
    forward_model : callable
        theta -> predicted_observations.
    prior_mean : np.ndarray
        Prior parameter mean.
    prior_cov : np.ndarray
        Prior parameter covariance.
    noise_cov : np.ndarray
        Observation noise covariance.

    Returns
    -------
    float
        Negative log-posterior (for minimization).
    """
    # Log-likelihood
    predicted = forward_model(theta)
    residuals = observations - predicted
    R_inv = np.linalg.inv(noise_cov)
    log_likelihood = -0.5 * np.sum(residuals @ R_inv * residuals)

    # Log-prior
    prior_residuals = theta - prior_mean
    P_inv = np.linalg.inv(prior_cov)
    log_prior = -0.5 * prior_residuals @ P_inv @ prior_residuals

    return -(log_likelihood + log_prior)


def map_estimate(
    observations: np.ndarray,
    forward_model,
    prior_mean: np.ndarray,
    prior_cov: np.ndarray,
    noise_cov: np.ndarray,
    bounds: list[tuple[float, float]] | None = None
) -> np.ndarray:
    """Compute Maximum A Posteriori (MAP) parameter estimate.
```

```
    Parameters
    ----------
    observations : np.ndarray
        Observed data.
    forward_model : callable
        Parameter-to-observation mapping.
    prior_mean : np.ndarray
        Prior mean from literature.
    prior_cov : np.ndarray
        Prior uncertainty.
    noise_cov : np.ndarray
        Measurement noise covariance.
    bounds : list of tuple, optional
        Physical bounds on parameters.

    Returns
    -------
    np.ndarray
        MAP parameter estimate.
    """
    result = minimize(
        log_posterior,
        prior_mean,
        args=(observations, forward_model, prior_mean,
              prior_cov, noise_cov),
        method='L-BFGS-B',
        bounds=bounds
    )
    return result.x
```

Listing 8.1: Bayesian parameter estimation for musculoskeletal model.

8.2 System Identification for Biological Systems

System identification techniques from control theory (Volume I) apply to biomechanical systems with modifications:

1. **Joint impedance identification**: Perturb the limb and measure the response. The stiffness, damping, and inertia of the joint can be estimated from the transfer function.
2. **Muscle parameter identification**: Optimize muscle parameters (F0, optimal length, tendon slack length) to match experimental force or torque data.
3. **Neural control identification**: Estimate the feedback gains used by the nervous system from perturbation experiments.

8.3 Machine Learning Approaches

Modern ML approaches to biomechanical inference include:

- Physics-informed neural networks (PINNs) that enforce equations of motion
- Gaussian process regression for joint angle estimation from sparse data
- Deep learning for markerless motion capture pose estimation

Exercises

1. **MAP Estimation.** Using the provided code, estimate the optimal fiber length of a biceps muscle from simulated isometric force data at 5 joint angles.

2. **Identifiability Analysis.** For a Hill-type muscle model, analyze which parameters can be uniquely estimated from isometric force-angle data alone. Which require dynamic data?

3. **(Programming.)** Implement a joint impedance identification experiment: simulate a perturbed pendulum and estimate stiffness and damping from the response.

Chapter 9

Deformable Bodies and Continuum Mechanics

Draft Notice. This chapter is under active development and contains only a structural outline with preliminary material. Full exposition, worked examples, proofs, and exercises will be added in future editions.

When the rigid-body assumption fails, we do not abandon mechanics. We embrace the continuum.

9.1 When Rigid-Body Assumptions Fail

The rigid-body model fails when:

1. **Internal stresses are needed**: to predict injury, fracture, or tissue damage
2. **Deformation affects kinematics**: shaft flexibility in golf, spine compliance
3. **Wave propagation matters**: impact biomechanics, blast injury
4. **Fluid-structure interaction**: blood flow, synovial fluid, respiratory mechanics

9.2 Beam Models for Long Bones and Shafts

When deformation is small and the body is slender (length $\gg$ diameter), beam theory provides an efficient model. A schematic representation of the finite element beam model is shown in Figure 9.1.

For a Euler-Bernoulli beam:

$$EI\frac{\partial^4 w}{\partial x^4} + \rho A\frac{\partial^2 w}{\partial t^2} = f(x,t), \tag{9.1}$$

where E is the elastic modulus, I is the second moment of area, ρ is density, A is cross-sectional area, $w(x,t)$ is transverse displacement, and f is the distributed load.

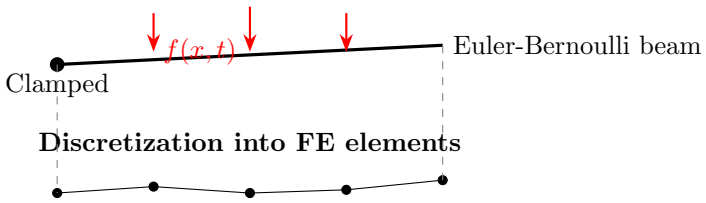


Figure 9.1: Finite element beam model: continuous Euler-Bernoulli beam (top) and its discretization into nodal elements (bottom).

9.2.1 Modal Analysis

The solution is expanded in modes:

$$w(x,t) = \sum_{k=1}^{N} \phi_k(x)\eta_k(t), \tag{9.2}$$

where $\phi_k(x)$ are mode shapes and $\eta_k(t)$ are modal coordinates. This converts the PDE to a system of ODEs that can be appended to the rigid-body equations of motion.

9.2.2 Golf Club Shaft Flexibility

The golf club shaft is the most accessible example of a flexible link in sport biomechanics. A typical shaft has:

- Length: $\sim$1.1 m
- First bending mode: 4–6 Hz (driver), 6–10 Hz (irons)
- Tip deflection at impact: 5–15 cm
- Lead/lag deflection: 2–5 cm
- Toe-down droop: 3–8 cm (gravity effect during downswing)

```python
import numpy as np
from scipy.linalg import eigh

def shaft_modal_analysis(
    n_elements: int = 20,
    length: float = 1.1,
    EI_root: float = 50.0,
    EI_tip: float = 15.0,
    rho_A: float = 0.15
) -> tuple[np.ndarray, np.ndarray]:
    """Compute natural frequencies and mode shapes of a tapered shaft.

    Uses finite element beam model with linear EI taper.

    Parameters
    ----------
    n_elements : int
        Number of finite elements.
```

```
    length : float
        Shaft length in meters.
    EI_root : float
        Flexural rigidity at root (grip end) in N*m^2.
    EI_tip : float
        Flexural rigidity at tip (head end) in N*m^2.
    rho_A : float
        Mass per unit length in kg/m.

    Returns
    -------
    tuple of np.ndarray
        (frequencies_hz, mode_shapes)
    """
    n_nodes = n_elements + 1
    n_dof = 2 * n_nodes  # displacement + rotation at each node
    h = length / n_elements

    K = np.zeros((n_dof, n_dof))
    M = np.zeros((n_dof, n_dof))

    for e in range(n_elements):
        # Local EI for this element (linear taper)
        xi = (e + 0.5) / n_elements
        EI_local = EI_root + (EI_tip - EI_root) * xi

        # Euler-Bernoulli beam element stiffness matrix
        k_e = EI_local / h**3 * np.array([
            [12,   6*h,  -12,   6*h],
            [6*h,  4*h**2, -6*h, 2*h**2],
            [-12, -6*h,   12,  -6*h],
            [6*h,  2*h**2, -6*h, 4*h**2]
        ])

        # Consistent mass matrix
        m_e = rho_A * h / 420 * np.array([
            [156,   22*h,  54,   -13*h],
            [22*h,  4*h**2, 13*h, -3*h**2],
            [54,    13*h,  156,  -22*h],
            [-13*h, -3*h**2, -22*h, 4*h**2]
        ])

        # Assembly
        dofs = [2*e, 2*e+1, 2*e+2, 2*e+3]
        for i_idx, i in enumerate(dofs):
            for j_idx, j in enumerate(dofs):
                K[i, j] += k_e[i_idx, j_idx]
                M[i, j] += m_e[i_idx, j_idx]

    # Boundary conditions: clamped at root (grip)
    free_dofs = list(range(2, n_dof))  # Remove first node
    K_free = K[np.ix_(free_dofs, free_dofs)]
    M_free = M[np.ix_(free_dofs, free_dofs)]

    # Solve eigenvalue problem
    eigenvalues, eigenvectors = eigh(K_free, M_free)

    # Convert to Hz
    frequencies = np.sqrt(np.abs(eigenvalues)) / (2 * np.pi)

```

```
    return frequencies[:6], eigenvectors[:, :6]


# Example
freqs, modes = shaft_modal_analysis()
print("Natural frequencies (Hz):")
for i, f in enumerate(freqs):
    print(f"  Mode {i+1}: {f:.1f} Hz")
```

Listing 9.1: Modal analysis of a golf club shaft.

9.3 Finite Element Methods (Overview)

For complex geometries (vertebral bodies, skull, articular surfaces), the finite element method (FEM) provides the most general approach. While a full treatment is beyond this text's scope, the key concept is:

The continuous domain Ω is divided into elements, and the displacement field is approximated using shape functions:

$$\boldsymbol{u}(\boldsymbol{x}, t) \approx \sum_i N_i(\boldsymbol{x}) \boldsymbol{u}_i(t), \tag{9.3}$$

leading to the assembled system:

$$M_{\text{FEM}} \ddot{\boldsymbol{u}} + C_{\text{FEM}} \dot{\boldsymbol{u}} + K_{\text{FEM}} \boldsymbol{u} = \boldsymbol{f}_{\text{ext}}. \tag{9.4}$$

Exercises

1. **Shaft Frequency.** Using the provided code, compute how the first natural frequency changes as the tip-to-root EI ratio varies from 0.1 to 0.9. Plot the result.

2. **Mode Shapes.** Plot the first three mode shapes of the shaft. Identify nodes (zero-crossings) and antinodes (maximum displacement).

3. **Timescale Separation.** If the golf downswing takes 0.25 s, estimate the number of oscillation cycles for each of the first 3 modes. Which modes are quasi-static?

4. **(Programming.)** Add a tip mass (clubhead, 0.2 kg) to the shaft FEM model and recompute the natural frequencies. How does the clubhead mass affect the frequencies?

Chapter 10

Applications of Control Theory to Biological Systems

Draft Notice. This chapter is under active development and contains only a structural outline with preliminary material. Full exposition, worked examples, proofs, and exercises will be added in future editions.

> The body is not a robot that happens to be made of meat. It is a system that has evolved to exploit its own physics.

10.1 Forward Dynamics Simulation

Forward dynamics simulation integrates the equations of motion given muscle excitations $\boldsymbol{u}(t)$. A feedback control loop for biological systems is illustrated in Figure 10.1.

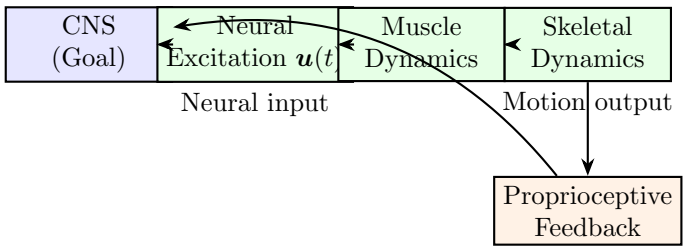


Figure 10.1: Biological feedback control loop: CNS goal signals drive neural excitations through muscle and skeletal dynamics, with proprioceptive feedback closing the loop.

$$\ddot{\boldsymbol{q}} = M(\boldsymbol{q})^{-1}\left[R(\boldsymbol{q})\boldsymbol{F}_{\text{muscle}} + J_c^T \boldsymbol{F}_c - C(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} - g(\boldsymbol{q})\right], \tag{10.1}$$

coupled with the muscle state equations (Chapter 3):

$$\dot{a}_i = (u_i - a_i)/\tau_i, \tag{10.2}$$

$$\dot{\ell}_i^M = v_i^M(a_i, \ell_i^M, \ell_i^{MT}(\boldsymbol{q})). \tag{10.3}$$

```
import numpy as np
from scipy.integrate import solve_ivp
from typing import Callable

def forward_dynamics_simulation(
    mass_matrix_func: Callable,
    coriolis_func: Callable,
    gravity_func: Callable,
    moment_arm_func: Callable,
    muscle_force_func: Callable,
    excitation_func: Callable,
    q0: np.ndarray,
    qdot0: np.ndarray,
    a0: np.ndarray,
    t_span: tuple[float, float],
    n_eval: int = 500
) -> dict:
    """Simulate forward dynamics of a musculoskeletal system.

    Parameters
    ----------
    mass_matrix_func : callable
        q -> M(q), shape (n_dof, n_dof)
    coriolis_func : callable
        (q, qdot) -> C(q, qdot) @ qdot, shape (n_dof,)
    gravity_func : callable
        q -> g(q), shape (n_dof,)
    moment_arm_func : callable
        q -> R(q), shape (n_dof, n_muscles)
    muscle_force_func : callable
        (a, l_M, v_M) -> F_muscle for each muscle
    excitation_func : callable
        t -> u(t), shape (n_muscles,)
    q0 : np.ndarray
        Initial joint angles.
    qdot0 : np.ndarray
        Initial joint velocities.
    a0 : np.ndarray
        Initial muscle activations.
    t_span : tuple
        (t_start, t_end)
    n_eval : int
        Number of output time points.

    Returns
    -------
    dict
        Keys: 't', 'q', 'qdot', 'a', 'tau'
    """
    n_dof = len(q0)
    n_muscles = len(a0)
    t_eval = np.linspace(t_span[0], t_span[1], n_eval)

    def dynamics(t: float, state: np.ndarray) -> np.ndarray:
        q = state[:n_dof]
        qdot = state[n_dof:2*n_dof]
        a = np.clip(state[2*n_dof:], 0.0, 1.0)

        # Muscle forces (simplified: assume known fiber length)
        F_muscle = np.array([
```

```
            a[i] * 500.0  # Simplified: F = a * F0
            for i in range(n_muscles)
        ])

        # Joint torques
        R = moment_arm_func(q)
        tau = R @ F_muscle

        # Skeletal dynamics
        M = mass_matrix_func(q)
        C_qdot = coriolis_func(q, qdot)
        g = gravity_func(q)
        qddot = np.linalg.solve(M, tau - C_qdot - g)

        # Activation dynamics
        u = excitation_func(t)
        tau_act = 0.015  # Illustrative placeholder; replace with a cited
    model value.
        tau_deact = 0.050  # Illustrative placeholder; replace with a cited
    model value.
        adot = np.where(
            u > a,
            (u - a) / tau_act,
            (u - a) / tau_deact
        )

        return np.concatenate([qdot, qddot, adot])

    state0 = np.concatenate([q0, qdot0, a0])
    sol = solve_ivp(dynamics, t_span, state0, t_eval=t_eval,
                    method='RK45', max_step=0.001)

    return {
        't': sol.t,
        'q': sol.y[:n_dof].T,
        'qdot': sol.y[n_dof:2*n_dof].T,
        'a': sol.y[2*n_dof:].T,
    }
```

Listing 10.1: Forward dynamics integration for a musculoskeletal model.

10.2 Predictive Simulation

Predictive simulation uses trajectory optimization (Volume I, Chapter 5) to find the muscle excitations that produce a desired motion while minimizing a cost:

$$\min_{\boldsymbol{u}(\cdot)} \int_0^T \mathcal{L}(\boldsymbol{x}, \boldsymbol{u})\, \mathrm{d}t \quad \text{s.t.} \quad \dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}),\ \boldsymbol{x}(0) = \boldsymbol{x}_0,\ h(\boldsymbol{x}, \boldsymbol{u}) \le 0, \tag{10.4}$$

where $\mathcal{L}$ is a physiologically motivated cost (e.g., metabolic energy, muscle fatigue, deviation from desired kinematics) and h captures physical constraints (joint limits, muscle force capacity, contact constraints).

Agrachev & Sachkov [1], Theorem 8.1 (Pontryagin's Maximum Principle), establishes the fundamental existence and optimality conditions for such optimal control problems. Their treatment covers affine systems with constraints and provides both theoretical guarantees and computational methods (the method of characteristics) for solving OCPs of this form.

10.3 Prosthetics and Exoskeleton Control

The control theory of Volumes I–II applies directly to assistive devices:

- **Powered prostheses**: trajectory tracking with impedance modulation
- **Exoskeletons**: assist-as-needed control using contraction theory
- **FES (Functional Electrical Stimulation)**: direct muscle activation control for paralyzed limbs

10.4 Sport Biomechanics Applications

10.4.1 The Golf Swing as a Biomechanical System

The golf swing demonstrates every concept in this volume:

1. Multi-segment kinematic chain (15+ DOF)
2. Muscle-driven actuation with co-contraction
3. Bi-articular coupling (forearm muscles, lats)
4. Passive dynamics exploitation (wrist release)
5. Shaft flexibility (deformable body)
6. Ground reaction force coupling (closed chain at the feet)

The complete biomechanical analysis integrates:

- Motion capture (kinematics)
- Force plates (ground reaction forces)
- EMG (muscle activation patterns)
- Inverse dynamics (joint torques)
- Static optimization (muscle forces)
- Forward dynamics validation (predictive simulation)

10.5 Synthesis: From Theory to Application

The progression through this volume mirrors the progression through the series:

Volume	**Concept**	**Biomechanical Application**	**Agrachev & Sachkov**
Vol 0	Screw axes, PoE	Joint kinematics, ISA analysis	
Vol I	Contraction theory	Stability of muscle-driven systems	
Vol I	DDP/iLQR	Optimal muscle excitation patterns	
Vol II	Trajectory tubes	Motor variability envelopes	
Vol II	Phase variables	Gait cycle parameterization	
Vol III	Hill model	Force prediction	
Vol III	Inverse dynamics	Joint torque estimation	
Vol IV	Motor control	Neural command generation	

Exercises

1. **Forward Simulation.** Using the provided code (adapted for a simple 2-DOF model), simulate a reaching movement with prescribed muscle excitation ramps.
2. **Predictive vs. Tracking.** Compare the muscle activations from static optimization (tracking measured kinematics) with those from predictive simulation (optimizing a metabolic cost).
3. **Prosthetic Control.** Design an impedance controller for a prosthetic knee that provides different stiffness during stance and swing phases.
4. **(Programming.)** Implement a complete inverse dynamics $\rightarrow$ static optimization pipeline for a 3-DOF planar arm model with 6 muscles.

Bibliography

[1] Andrei A. Agrachev and Yuri L. Sachkov. *Control Theory from the Geometric Viewpoint*, volume 87 of *Encyclopaedia of Mathematical Sciences*. Springer, 2004.

[2] R. M. Alexander. Energy-saving mechanisms in walking and running. *Journal of Experimental Biology*, 160:55–69, 1991.

[3] R. M. Alexander. Tendon elasticity and muscle function. *Comparative Biochemistry and Physiology. Part A, Molecular and Integrative Physiology*, 133(4):1001–1011, 2002.

[4] Edith M. Arnold, Samuel R. Ward, Richard L. Lieber, and Scott L. Delp. A model of the lower limb for analysis of human movement. *Annals of Biomedical Engineering*, 38(2):269–279, 2010.

[5] L. Blankevoort and R. Huiskes. Ligament-bone interaction in a three-dimensional model of the knee. *Journal of Biomechanical Engineering*, 113(3):263–269, 1991.

[6] Lorenzo Chiari, Ugo Della Croce, Alberto Leardini, and Aurelio Cappozzo. Human movement analysis using stereophotogrammetry. part 2: Instrumental errors. *Gait and Posture*, 21(2):197–211, 2005.

[7] J. J. Crisco, J. C. Coburn, T. J. Wolfson, W. H. Akeson, and S. L.-Y. Woo. Instantaneous center of rotation as a function of a four-bar linkage knee model. *Journal of Biomechanics*, 34(6):735–742, 2000.

[8] P. de Leva. Adjustments to zatsiorsky-seluyanov's segment inertia parameters. *Journal of Biomechanics*, 29(9):1223–1230, 1996.

[9] S. L. Delp, F. C. Anderson, A. S. Arnold, et al. Opensim: Open-source software to create and analyze dynamic simulations of movement. *IEEE Transactions on Biomedical Engineering*, 54(11):1940–1950, 2007.

[10] C. R. Dickerson, D. B. Chaffin, and R. E. Hughes. Accurate calculation of scapular anatomy with 3-dimensional computed tomography. *Journal of Shoulder and Elbow Surgery*, 14(6):601–609, 2005.

[11] E. S. Grood and W. J. Suntay. A joint coordinate system for the clinical description of three-dimensional motions: application to the knee. *Journal of Biomechanical Engineering*, 105(2):136–144, 1983.

[12] A. M. Halder, E. Itoi, and K. N. An. Structural and functional anatomy of the shoulder. *The Shoulder*, 2000.

[13] M. E. Harrington, A. B. Zavatsky, S. E. M. Lawson, Z. Yuan, and T. N. Theologis. Prediction of the hip joint centre in adults, children, and patients with cerebral palsy based on magnetic resonance imaging. *Journal of Biomechanics*, 40(3):595–602, 2007.

[14] A. V. Hill. The heat of shortening and the dynamic constants of muscle. *Proceedings of the Royal Society B: Biological Sciences*, 126(843):136–195, 1938.

[15] A. V. Hill. A note on the mechanics of the contraction of skeletal muscle. *Proceedings of the Royal Society B: Biological Sciences*, 137:273–280, 1950.

[16] Katherine R. S. Holzbaur, Wendy M. Murray, and Scott L. Delp. A model of the upper extremity for simulating musculoskeletal surgery and analyzing neuromuscular control. *Annals of Biomedical Engineering*, 33(6):829–840, 2005.

[17] A. F. Huxley and R. M. Simmons. Proposed mechanism of force generation in striated muscle. *Nature*, 233:533–538, 1971.

[18] A. R. Karduna, P. W. McClure, L. A. Michener, and B. Sennett. Scapular kinematics and humeral head translation in the throwing shoulder. *Clinical Biomechanics*, 15(10):789–798, 2000.

[19] Richard D. Komistek, Douglas A. Dennis, and Mohamed Mahfouz. In vivo fluoroscopic analysis of the normal human knee. *Clinical Orthopaedics and Related Research*, 410:69–81, 2003.

[20] Alberto Leardini, Lorenzo Chiari, Ugo Della Croce, and Aurelio Cappozzo. Human movement analysis using stereophotogrammetry. part 3: Soft tissue artifact assessment and compensation. *Gait and Posture*, 21(2):212–225, 2005.

[21] Matthew Millard, Thomas Uchida, Ajay Seth, and Scott L. Delp. Flexing computational muscle: Modeling and simulation of musculotendon dynamics. *Journal of Biomechanical Engineering*, 135(2):021005, 2013.

[22] Richard R. Neptune, Steven A. Kautz, and Felix E. Zajac. Contributions of the individual ankle plantar flexors to support, forward progression and swing initiation during walking. *Journal of Biomechanics*, 34(11):1387–1398, 2001.

[23] Benno M. Nigg and Walter Herzog. *Biomechanics of the Musculo-skeletal System*. John Wiley & Sons, Chichester, 3rd edition, 2007.

[24] J. J. O'Connor, E. A. Shercliff, E. Biden, and J. P. Paul. Geometry of the knee in the sagittal plane. *Proceedings of the Institution of Mechanical Engineers. Part H*, 203(4):223–233, 1989.

[25] Alana Peters, Brook Galna, Morgan Sangeux, Meg Morris, and Richard Baker. Quantification of soft tissue artifact in lower limb human motion analysis: A systematic review. *Gait and Posture*, 31(1):1–8, 2010.

[26] Boris I. Prilutsky and Vladimir M. Zatsiorsky. Optimization-based models of muscle coordination. *Exercise and Sport Sciences Reviews*, 30(1):32–38, 2002.

[27] J. P. Scholz and G. Schöner. The uncontrolled manifold concept: Identifying control variables for a functional task. *Experimental Brain Research*, 126:289–306, 1999.

[28] Stephen H. Scott. Optimal feedback control and the neural basis of volitional motor control. *Nature Reviews Neuroscience*, 5(7):532–546, 2004.

[29] D. G. Thelen. Adjustment of muscle mechanics model parameters to simulate dynamic contractions in older adults. *Journal of Biomechanical Engineering*, 125(1):70–77, 2003.

[30] Darryl G. Thelen, Frank C. Anderson, and Scott L. Delp. Generating dynamic simulations of movement using computed muscle control. *Journal of Biomechanics*, 36(3):321–328, 2003.

[31] David A. Winter. *Biomechanics and Motor Control of Human Movement*. John Wiley & Sons, Hoboken, NJ, 4th edition, 2009.

[32] J. M. Winters and L. Stark. Different roles of fast and slow twitch muscle fibers in a level versus graded running. *Journal of Biomechanics*, 17(2):119–131, 1984.

[33] P. Wretenberg, D. K. Ramsey, and G. Nemeth. Knee movement tracking using accelerometers and gyroscopes. *Medicine and Science in Sports and Exercise*, 27(10):1474–1479, 1995.

[34] G. Wu, S. Siegler, P. Allard, et al. Isb recommendation on definitions of joint coordinate system of various joints for the reporting of human joint motion—part i: ankle, hip, and spine. *Journal of Biomechanics*, 35(4):543–548, 2002.

[35] G. Wu, F. C. T. van der Helm, H. E. J. Veeger, et al. Isb recommendation on definitions of joint coordinate systems of various joints for the reporting of human joint motion — part ii: shoulder, elbow, wrist and hand. *Journal of Biomechanics*, 38(5):981–992, 2005.

[36] F. E. Zajac. Muscle and tendon: properties, models, scaling, and application to biomechanics and motor control. *Critical Reviews in Biomedical Engineering*, 17(4):359–411, 1989.