2025-11-01

# E003: Plant Models and Dynamics

## The Physics Behind the Double Broomstick

**Part 1 · Duration: 30-35 minutes**

***Beginner-Friendly Visual Study Guide***

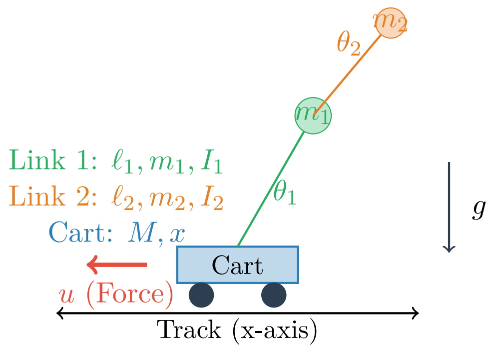
**Learning Objective:** Understand DIP physics, Lagrangian mechanics approach, and the three model variants for different speed/accuracy tradeoffs

# What is the Plant?

**Key Concept**

The "plant" is the physical system you're trying to control - the body that control decisions act upon.
**Key Question:** "If I push the cart with force $F$ right now, how will angles and velocities change in the next millisecond?"

## Physical System Description




**Components:**

- **Cart**: 1.5 kg (like a laptop), slides on track with friction
- **Pendulum 1**: 0.2 kg, 40 cm (wooden ruler weight/length)
- **Pendulum 2**: 0.15 kg, 30 cm (slightly lighter/shorter)

**Three Coordinates:**

enumi$x$ - Cart position (left/right)

0. enumi$\theta_1$ - First pendulum angle (from vertical)
0. enumi$\theta_2$ - Second pendulum angle (from vertical)

**Pro Tip**

Based on actual control lab rigs - not arbitrary!

# Lagrangian Mechanics: The Elegant Shortcut

## Why NOT Use Newton's $F = ma$ Directly?

**Common Pitfall**

Newton's approach would require:

0. enumiFree-body diagrams for cart + both pendulums
0. enumiAll internal forces (pin forces, constraint forces)
0. enumiCoupled force balance equations
0. enumiSolving for dozen+ variables you don't care about!

**Result:** Nightmare of algebra with unknown internal forces

## The Lagrangian Shortcut: Ignore Internal Glue

**Key Concept**

**Key Insight (1700s):** You don't need internal constraint forces if you focus on energy!
**Recipe:**

0. enumiCalculate total kinetic energy (motion)
0. enumiCalculate total potential energy (gravity)
0. enumiForm Lagrangian: $\mathcal{L} = T - V$
0. enumiApply Euler-Lagrange equations (systematic calculus)
0. enumiGet equations of motion - NO internal forces!

## The Beautiful Result

**Matrix Equation of Motion**

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

**Physical Meaning:**

- **M**$(q)$ - Mass matrix: "How mass/inertia distributed at these angles?"
- **C**$(q, \dot{q})$ - Spinning forces: Coriolis & centrifugal effects
- **G**$(q)$ - Gravity: "How hard does gravity pull at these angles?"
- **B** - Input distribution: "Force only pushes cart directly"
- $u$ - Control force, $d$ - Disturbances

# Mass Matrix Structure: Action and Reaction

## Diagonal Elements

**"Self-inertia":**

- $M_{11}$ = Total mass (cart + pendulums)
- $M_{22}$ = Rotational inertia of Pendulum 1
- $M_{33}$ = Rotational inertia of Pendulum 2

## Off-Diagonal Elements

**Coupling terms:**

- $M_{12}$ = "Cart acceleration → Pendulum 1 torque"
- Depend on $\cos(\theta_i)$
- Strongest when vertical

**Key Concept**

**Key Property - Symmetry:** $M_{12} = M_{21}$, $M_{13} = M_{31}$, $M_{23} = M_{32}$
This is Newton's Third Law: **Action = Reaction**
Effect of Link 1 on Link 2 = Effect of Link 2 on Link 1 (opposite directions)

## Coriolis & Centrifugal Terms

**Example**

**Coriolis Force:** Apparent force due to rotation

- Example: Walk straight on spinning merry-go-round ⇒ path curves
- Term: $-c_{12}\sin(\theta_1 - \theta_2)\dot{\theta}_2$ couples pendulum velocities

**Centrifugal Force:** Outward push from rotation

- Example: Car turn ⇒ pushed against door
- Term: $-c_1\sin(\theta_1)\dot{\theta}_1^2$ pushes cart sideways

## Gravity Vector

$$\mathbf{G}(q) = \begin{bmatrix} 0 \\ -g_1 g \sin(\theta_1) \\ -g_2 g \sin(\theta_2) \end{bmatrix}$$

- No gravity on cart (horizontal motion)
- Pendulum torques: $g \approx 9.81$ m/s²
- **Sign:** Upright ($\theta = 0$) is unstable - gravity torque pushes away!

## Singularities: When Math Locks Up

**Physical Analogy:**

Your arm fully extended with locked elbow - can't push further. Geometry "locks up."

Same happens with pendulums in specific configurations.

**Condition Number ($\kappa$):**
Health score for mass matrix:

- $\kappa$ = 1-100: Healthy
- $\kappa > 10^6$: Sick
- $\kappa \to \infty$: Singular

**Common Pitfall**

**Singularity Occurs:** Both pendulums horizontal (lying flat)
**How We Handle:**

enumiCheck $\kappa$ before inverting $\mathbf{M}$

0. enumiIf $\kappa > 10^8$: Switch to pseudoinverse with regularization

0. enumiPrevents division-by-near-zero crashes

**In Practice:** Near upright, $\kappa = 10$-$100$ (safe). Controller avoids dangerous configs.

## Three Model Variants: The Team

**Key Concept**

Trade-off: **Speed** vs. **Accuracy**
Different tasks need different balances. Meet the three teammates!

### Model 1: The Sprinter (Simplified Linear)

**Personality:** Fast, agile, assumes near-perfect
**Assumptions:**

- $\sin(\theta) \approx \theta$ (small angles)
- $\cos(\theta) \approx 1$
- Constant mass matrix
- Ignores coupled effects

**Superpowers:**

- 10-100x faster than full model
- Perfect for PSO (1500 sims)
- Great for teaching

**Kryptonite:**

- Can't handle angles > 10ř
- Lies about nonlinear effects
- Swing-up? Garbage results!

**When to Use:**

- Initial prototyping
- PSO gain optimization
- Educational demos
- Near-upright operation

### Model 2: The Simulator (Full Nonlinear)

**Personality:** Slow, heavy, brutally honest
**Reality Check:**

- Every trig term computed exactly
- All Coriolis effects
- All coupling captured
- Angle-dependent matrices

**Superpowers:**

- Truth: Full operating range
- Accurate: Hanging → upright
- Publishable benchmark results

**Kryptonite:**

- 10-100x slower
- Overkill for simple tasks
- Sledgehammer to crack a nut

**When to Use:**

- Final validation
- Swing-up control
- Research benchmarks
- Hardware deployment prep

### Model 3: The Efficient Pro (Low-Rank POD)

**Personality:** Smart compromise
**Clever Trick:** Proper Orthogonal Decomposition

enumiRun Simulator 1000x times

0. enumiCollect data snapshots

0. enumiSVD: Find important patterns

0. enumiKeep signal, discard noise

**Superpowers:**

- 10-50x speedup
- Preserves key dynamics
- Real-time capable (HIL)

**Kryptonite:**

- Needs training (run Simulator first)
- Can miss rare edge cases
- Not drop-in replacement

**When to Use:**

- Monte Carlo (1000 sims)
- Parameter sweeps
- Sensitivity analysis
- HIL testing

## Model Accuracy Comparison (MT-6 Benchmark)

### Validation Test Results

**Setup:** $\theta_1 = 10$ř, $\theta_2 = 5$ř, Classical SMC, 10 seconds

| Model | Settling [s] | Overshoot [°] | RMS Error [°] | Speed [sims/s] |
|---|---|---|---|---|
| Simplified | 2.31 | 4.2 | 0.12 | 450 |
| Full Nonlinear | 2.58 | 5.1 | 0.15 | 8 |
| Low-Rank (k=10) | 2.54 | 4.9 | 0.14 | 95 |

**Observations:**

- Simplified: Optimistic (underestimates settling time)
- Full: Conservative (most realistic)
- Low-Rank: Sweet spot (2% error, 12x speedup)

### Pro Tip

**The Bottom Line:**

- Quick prototyping? **Sprinter**
- Final validation? **Simulator**
- Massive data crunching? **Efficient Pro**

Like having hammer, precision screwdriver, and power drill - use the right tool!

# Numerical Integration Methods

## Three Integrators Available

### Euler (1st Order)

**Algorithm:**

```
lstnumberstate_dot = f(state, u)
lstnumberstate_new = state + state_dot * dt
```

**Pros:** Simplest, fastest **Cons:** Inaccurate (large errors)
**Use:** Educational only, NOT for research

### RK4 (4th Order)

**Algorithm:** Four slope evaluations per step

```
lstnumberk1 = f(state, u)
lstnumberk2 = f(state + 0.5*dt*k1, u)
lstnumberk3 = f(state + 0.5*dt*k2, u)
lstnumberk4 = f(state + dt*k3, u)
lstnumberstate_new = state + (dt/6) * (k1 + 2*k2 + 2*k3 + k4)
```

**Pros:** Good balance **Cons:** Fixed step size
**Use:** Standard choice with $dt = 0.001s$ (1 kHz)

### RK45 (Adaptive)

**Algorithm:** SciPy's `solve_ivp` with automatic step sizing
**Error Control:** `rtol=1e-6`, `atol=1e-9`
**Pros:** Most accurate, adaptive **Cons:** Slower
**Use:** High-accuracy validation, research benchmarks

## Quick Reference: Key Equations

### Equation of Motion

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

Solve for acceleration: $\ddot{q} = \mathbf{M}^{-1}\left[\mathbf{B}u - \mathbf{C}\dot{q} - \mathbf{G}\right]$

### State Vector

$$x = [x, \theta_1, \theta_2, \dot{x}, \dot{\theta}_1, \dot{\theta}_2]^T$$

Derivative: $\dot{x} = [\dot{x}, \dot{\theta}_1, \dot{\theta}_2, \ddot{x}, \ddot{\theta}_1, \ddot{\theta}_2]^T$

### Condition Number Check

$$\kappa(\mathbf{M}) = \frac{\sigma_{\max}}{\sigma_{\min}}$$

If $\kappa > 10^8$: Use pseudoinverse with regularization

## Implementation Workflow

**Quick Summary**

**Full Nonlinear Model (Python):**

enumiUnpack state: Extract $q = [x, \theta_1, \theta_2]$ and $\dot{q} = [\dot{x}, \dot{\theta}_1, \dot{\theta}_2]$

0. enumiBuild mass matrix: $\mathbf{M}(q)$ based on current angles (trig functions)
0. enumiCalculate Coriolis: $\mathbf{C}(q, \dot{q})$ with velocity coupling
0. enumiCalculate gravity: $\mathbf{G}(q)$ angle-dependent torques
0. enumiApply control: Force $u$ enters through $\mathbf{B}$
0. enumiSolve for $\ddot{q}$: Invert $\mathbf{M}$ (check $\kappa$ first!)
0. enumiReturn derivative: Pack $[\dot{q}, \ddot{q}]$ for integrator

**NumPy makes it simple:** `accel = np.linalg.solve(M, B*u - C@vel - G)`

## Configuration Parameters

**Physical:**

- Cart mass: $M = 1.5$ kg
- Pendulum 1: $m_1 = 0.2$ kg, $L_1 = 0.4$ m
- Pendulum 2: $m_2 = 0.15$ kg, $L_2 = 0.3$ m
- Gravity: $g = 9.81$ m/s²

**Numerical:**

- Time step: $dt = 0.001$ s (1 kHz)
- Integrator: RK4 (standard)
- Singularity threshold: $\kappa > 10^8$
- Duration: 10 s (typical)

## What's Next?

**Key Concept**

**E004: PSO Optimization** - How to automatically tune controller gains using particle swarm intelligence
**E005: Simulation Engine** - The runner, vectorized simulators, and how we achieve 100x speedups
**Remember:** Now you understand the physics. Next, we optimize the control to make it work perfectly!