# 平面二连杆机械臂建模推导及py代码

## 1. 机器人概述

研究对象为**平面二连杆机器人**，其结构如图所示：

- 连杆长度均为 $l$
- 质量均匀分布，质量为 $M$
- 关节变量定义：
  - $\theta_1$：连杆1与 $y$ 轴负半轴夹角（逆时针为正）
  - $\theta_2$：连杆2与连杆1延长线夹角（逆时针为正）
- 关节力矩：$\tau_1, \tau_2$
- 末端执行器坐标：$P(x, y)$

代码块

```
          y
          ↑
          |
          |       τ₁
          O────────→ x
          |   θ₁
          |  /
          | /
          |/θ₂
          •
         / \
        /   \
      P(x,y)
```

## 2. 运动学建模

### 2.1 正运动学模型（基于旋量指数积法）

#### 旋量表示

关节轴线方向向量

$$\omega_1 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \quad \omega_2 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

关节轴上参考点

$$r_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \quad r_2 = \begin{bmatrix} 0 \\ -l \\ 0 \end{bmatrix}$$

对应的旋量指数形式为

$$e^{[V_1]\theta_1} = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 & l\sin\theta_1 \\ \sin\theta_1 & \cos\theta_1 & 0 & -l(1-\cos\theta_1) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad e^{[V_2]\theta_2} = \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & 0 & -l\sin\theta_2 \\ \sin\theta_2 & \cos\theta_2 & 0 & -l(1-\cos\theta_2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

初始位姿（theta = [0, 0]）：

$$^B_T(0) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -2l \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

最终位姿

$$^B_T(\theta) = e^{[V_1]\theta_1} e^{[V_2]\theta_2} \, ^B_T(0)$$

展开后可得末端位置：

$$x = l\sin\theta_1 + l\sin(\theta_1 + \theta_2)$$
$$y = -l\cos\theta_1 - l\cos(\theta_1 + \theta_2)$$

## 2.2 逆运动学模型（几何法）

设末端坐标为 $ (x, y) $，则总长度：

$$L^2 = x^2 + y^2$$

由余弦定理得：

$$\theta_2 = \arccos\left(\frac{x^2 + y^2}{2l^2} - 1\right)$$

令：

$$\psi = \arccos\left(\frac{x^2+y^2}{2l}\right), \quad \beta = \left|\arctan\left(\frac{y}{x}\right)\right|$$

则：

$$\theta_1 = \frac{\pi}{2} - (\beta \pm \psi)$$

取值规则如下表

| 条件 | $\theta_1$ 取值 |
|---|---|
| $\theta_2 > 0$，位形一 | $\frac{\pi}{2} - (\beta + \psi)$ |
| $\theta_2 > 0$，位形二 | $\frac{\pi}{2} - (\beta - \psi)$ |
| $\theta_2 < 0$，位形一 | $\frac{3\pi}{2} - (\beta + \psi)$ |
| $\theta_2 < 0$，位形二 | $\frac{3\pi}{2} - (\beta - \psi)$ |

## 2.3 机器人运动学仿真（Python Robotics 实现）

> 使用 `numpy` 和 `matplotlib` 实现正逆运动学验证。

### ✅ 安装依赖

代码块

```
pip install numpy matplotlib scipy
```

### 🔧 Python 代码（正逆运动学与可视化）

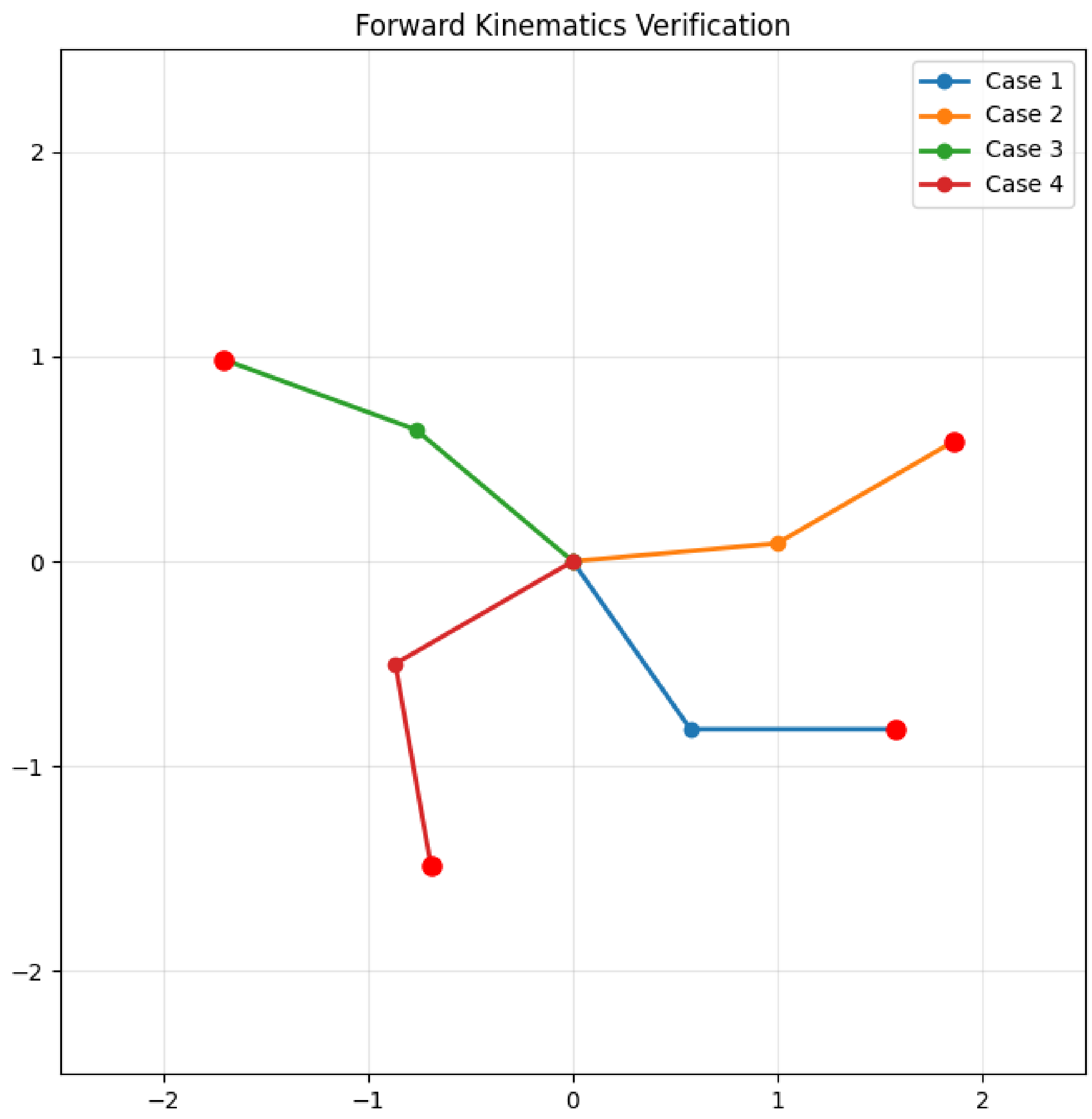
代码块

```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import fsolve

# 参数设置
l = 1.0  # 连杆长度

def forward_kinematics(theta1, theta2):
    """正运动学：计算末端坐标"""
```

```python
    x = l * np.sin(theta1) + l * np.sin(theta1 + theta2)
    y = -l * np.cos(theta1) - l * np.cos(theta1 + theta2)
    return x, y

def inverse_kinematics(x, y):
    """逆运动学：求解 θ1, θ2"""
    L_sq = x**2 + y**2
    if L_sq > (2*l)**2 or L_sq < 0:
        raise ValueError("Target point out of workspace!")

    cos_theta2 = (L_sq / (2*l**2)) - 1
    theta2 = np.arccos(cos_theta2)

    beta = np.abs(np.arctan2(y, x))
    phi = np.arccos(L_sq / (2*l))

    solutions = []
    for sign in [+1, -1]:
        theta1 = np.pi/2 - (beta + sign*phi)
        solutions.append((theta1, theta2))
    return solutions

def plot_arm(ax, theta1, theta2, color='blue', label=""):
    """绘制机械臂"""
    base = (0, 0)
    joint1 = (l * np.sin(theta1), -l * np.cos(theta1))
    end = forward_kinematics(theta1, theta2)
    xs = [base[0], joint1[0], end[0]]
    ys = [base[1], joint1[1], end[1]]
    ax.plot(xs, ys, marker='o', color=color, linewidth=2, markersize=6, label=label)

# 示例验证
test_cases = [
    (np.deg2rad(35), np.deg2rad(55)),
    (np.deg2rad(95), np.deg2rad(25)),
    (np.deg2rad(230), np.deg2rad(20)),
    (np.deg2rad(300), np.deg2rad(70))
]

fig, ax = plt.subplots(figsize=(8, 8))
ax.set_aspect('equal')
ax.grid(True, alpha=0.3)
ax.set_xlim(-2.5, 2.5)
ax.set_ylim(-2.5, 2.5)

for i, (t1, t2) in enumerate(test_cases):
```

```
        x, y = forward_kinematics(t1, t2)
        plot_arm(ax, t1, t2, color=f'C{i}', label=f'Case {i+1}')
        ax.plot(x, y, 'ro', markersize=8)

plt.title("Forward Kinematics Verification")
plt.legend()
plt.show()
```

代码运行结果图1



# 3. 动力学建模

## 3.1 动能计算

对质量微元 dm ，其速度平方为：

- **连杆 I**：

$$v_1^2 = r^2\dot{\theta}_1^2$$

- **连杆 II**：

$$v_2^2 = l^2\dot{\theta}_1^2 + r^2(\dot{\theta}_1 + \dot{\theta}_2)^2 + 2lr\dot{\theta}_1(\dot{\theta}_1 + \dot{\theta}_2)\cos\theta_2$$

积分得动能：

$$E_k = \frac{1}{2}ml^2\left[\left(\frac{5}{3} + \cos\theta_2\right)\dot{\theta}_1^2 + \left(\frac{2}{3} + \cos\theta_2\right)\dot{\theta}_1\dot{\theta}_2 + \frac{1}{3}\dot{\theta}_2^2\right]$$

## 3.2 势能与拉格朗日方程

以 x 轴为零势能面：

$$E_p = -mgl\left[\frac{3}{2}\cos\theta_1 + \frac{1}{2}\cos(\theta_1 + \theta_2)\right]$$

拉格朗日函数：

$$L = E_k - E_p$$

应用拉格朗日方程得动力学模型：

$$\tau = D_1(\theta)\ddot{\theta} + D_2(\theta)\dot{\theta}^2 + D_3(\theta)\dot{\theta}_i\dot{\theta}_j + D_4(\theta)$$

其中：

$$D_1(\theta) = ml^2\begin{bmatrix}\frac{5}{3} + \cos\theta_2 & \frac{1}{3} + \frac{1}{2}\cos\theta_2 \\ \frac{1}{3} + \frac{1}{2}\cos\theta_2 & \frac{1}{3}\end{bmatrix}, \quad D_2(\theta) = ml^2\begin{bmatrix}0 & -\frac{1}{2}\sin\theta_2 \\ -\frac{1}{2}\sin\theta_2 & 0\end{bmatrix}$$

$$D_3(\theta) = ml^2\begin{bmatrix}0 & 0 \\ 0 & 0\end{bmatrix}, \quad D_4(\theta) = mgl\begin{bmatrix}\frac{3}{2}\sin\theta_1 + \frac{1}{2}\sin(\theta_1 + \theta_2) \\ \frac{1}{2}\sin(\theta_1 + \theta_2)\end{bmatrix}$$

## 3.3 动力学仿真（Python 实现）

以下代码实现了**高精度的自由摆动仿真**，包含状态演化与轨迹可视化

代码块

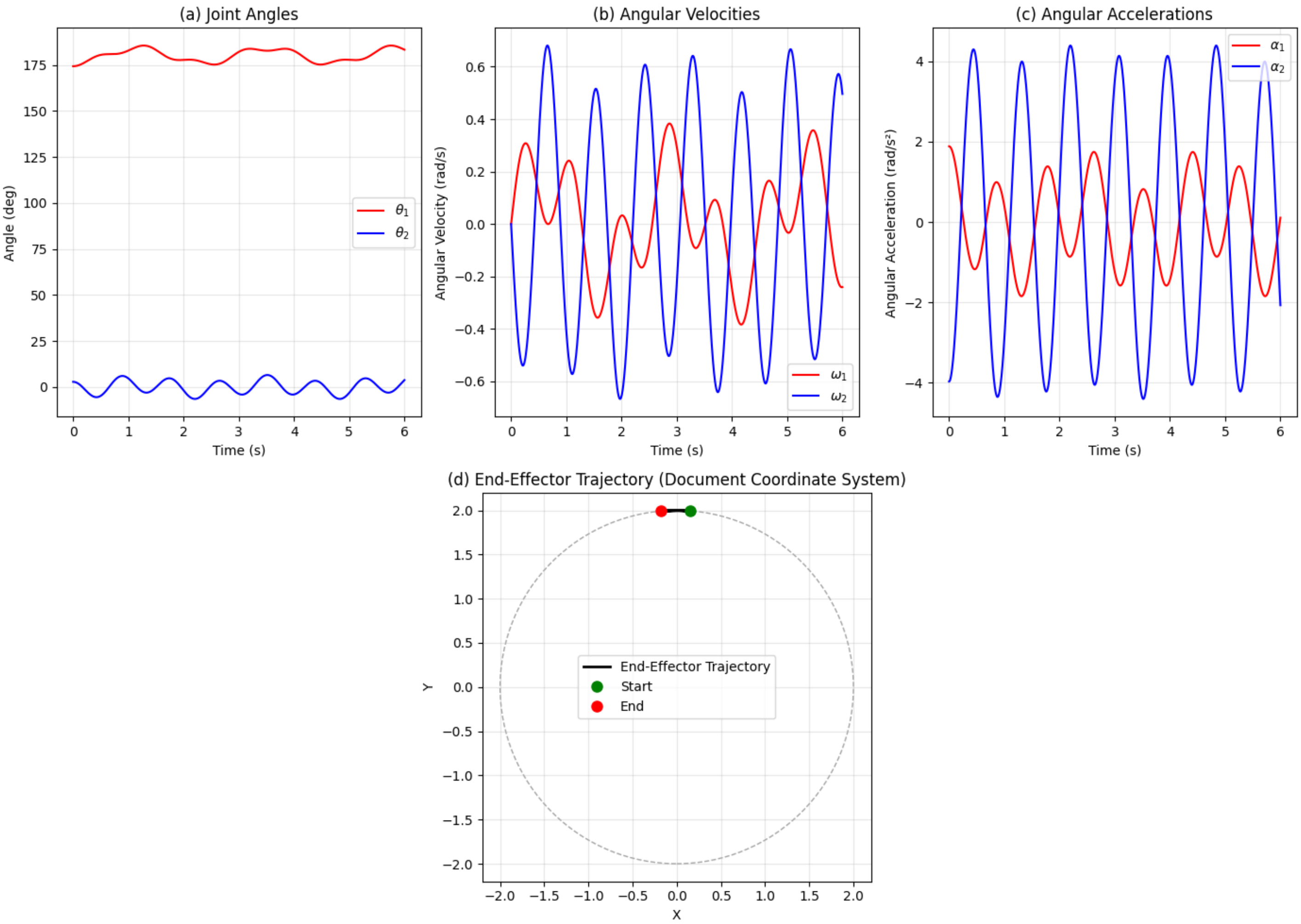
```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import solve_ivp

# =====================
```

```python
# 参数设置（与文档一致）
# ======================
m = 1.0      # 连杆质量 (kg)
l = 1.0      # 连杆长度 (m)
g = 9.81     # 重力加速度 (m/s²)

def dynamics(t, state, m, l, g):
    """
    平面2R机器人动力学方程（无外力矩）
    状态: [theta1, theta2, omega1, omega2]
    """
    theta1, theta2, dtheta1, dtheta2 = state

    c2 = np.cos(theta2)
    s2 = np.sin(theta2)

    # 惯性矩阵 M(q)
    M = m * l**2 * np.array([
        [5/3 + c2,       1/3 + 0.5*c2],
        [1/3 + 0.5*c2,   1/3         ]
    ])

    # 科里奥利和向心力 C(q, q )q
    C = m * l**2 * np.array([
        -s2 * dtheta2 * (dtheta1 + 0.5 * dtheta2),
        -0.5 * s2 * dtheta1**2
    ])

    # 重力项 G(q)
    G = m * g * l * np.array([
        1.5 * np.sin(theta1) + 0.5 * np.sin(theta1 + theta2),
        0.5 * np.sin(theta1 + theta2)
    ])

    # 动力学: M(q) * ddq = G(q) - C(q, q )q
    ddq = np.linalg.solve(M, G - C)
    return [dtheta1, dtheta2, ddq[0], ddq[1]]

# ======================
# 仿真设置（加入微小扰动以打破平衡）
# ======================
initial_state = [np.pi - 0.1, 0.05, 0.0, 0.0]  # [θ1, θ2, ω1, ω2]
t_span = (0, 6)
t_eval = np.linspace(0, 6, 1000)

sol = solve_ivp(
    dynamics,
```

```python
    t_span,
    initial_state,
    args=(m, l, g),
    t_eval=t_eval,
    method='RK45',
    rtol=1e-8,
    atol=1e-10
)

# 提取状态
theta1 = sol.y[0]
theta2 = sol.y[1]
omega1 = sol.y[2]
omega2 = sol.y[3]

# 计算加速度用于绘图
alpha1, alpha2 = [], []
for i in range(len(sol.t)):
    _, _, a1, a2 = dynamics(sol.t[i], [theta1[i], theta2[i], omega1[i], omega2[i]], m, l, g)
    alpha1.append(a1)
    alpha2.append(a2)
alpha1, alpha2 = np.array(alpha1), np.array(alpha2)

# ======================
# 可视化
# ======================
fig = plt.figure(figsize=(14, 10))

# (a) 关节角度
ax1 = plt.subplot(2, 3, 1)
ax1.plot(sol.t, np.rad2deg(theta1), 'r-', label=r'$\theta_1$')
ax1.plot(sol.t, np.rad2deg(theta2), 'b-', label=r'$\theta_2$')
ax1.set_xlabel('Time (s)')
ax1.set_ylabel('Angle (deg)')
ax1.set_title('(a) Joint Angles')
ax1.grid(True, alpha=0.3)
ax1.legend()

# (b) 角速度
ax2 = plt.subplot(2, 3, 2)
ax2.plot(sol.t, omega1, 'r-', label=r'$\omega_1$')
ax2.plot(sol.t, omega2, 'b-', label=r'$\omega_2$')
ax2.set_xlabel('Time (s)')
ax2.set_ylabel('Angular Velocity (rad/s)')
ax2.set_title('(b) Angular Velocities')
ax2.grid(True, alpha=0.3)
```

```
ax2.legend()

# (c) 角加速度
ax3 = plt.subplot(2, 3, 3)
ax3.plot(sol.t, alpha1, 'r-', label=r'$\alpha_1$')
ax3.plot(sol.t, alpha2, 'b-', label=r'$\alpha_2$')
ax3.set_xlabel('Time (s)')
ax3.set_ylabel('Angular Acceleration (rad/s²)')
ax3.set_title('(c) Angular Accelerations')
ax3.grid(True, alpha=0.3)
ax3.legend()

# (d) 末端轨迹（使用文档坐标系）
ax4 = plt.subplot(2, 3, (4, 6))
x_end = l * np.sin(theta1) + l * np.sin(theta1 + theta2)
y_end = -l * np.cos(theta1) - l * np.cos(theta1 + theta2)

ax4.plot(x_end, y_end, 'k-', linewidth=2, label='End-Effector Trajectory')
ax4.plot(x_end[0], y_end[0], 'go', markersize=8, label='Start')
ax4.plot(x_end[-1], y_end[-1], 'ro', markersize=8, label='End')

# 工作空间边界
max_r = 2 * l
circle = plt.Circle((0, 0), max_r, color='gray', fill=False, linestyle='--',
alpha=0.7)
ax4.add_patch(circle)
ax4.set_xlim(-max_r*1.1, max_r*1.1)
ax4.set_ylim(-max_r*1.1, max_r*1.1)
ax4.set_aspect('equal')
ax4.set_xlabel('X')
ax4.set_ylabel('Y')
ax4.set_title('(d) End-Effector Trajectory (Document Coordinate System)')
ax4.grid(True, alpha=0.3)
ax4.legend()

plt.tight_layout()
plt.show()

# 打印初始加速度验证
print(f"Initial α1 = {alpha1[0]:.4f} rad/s², α2 = {alpha2[0]:.4f} rad/s²")
```

代码运行结果图2


(a) Joint Angles
Angle (deg)
Time (s)
θ1
θ2
(b) Angular Velocities
Angular Velocity (rad/s)
Time (s)
ω1
ω2
(c) Angular Accelerations
Angular Acceleration (rad/s²)
Time (s)
α1
α2
(d) End-Effector Trajectory (Document Coordinate System)
Y
X
End-Effector Trajectory
Start
End