

学号		成绩	
----	--	----	--



沈阳建筑大学

课程设计说明书

设计名称 《机械原理与设计 1》课程设计

设计题目 插齿机主运动机构计算机分析

学 院 机械工程学院

专 业 机械设计制造及其自动化

班 级 机械 2203 班

姓 名 孟睿豪

指导教师 杨谢柳

《机械原理与设计 1》课程设计任务书

设计时间	二周	
设计题目	插齿机主运动机构计算机分析	
设计条件	(见附页)	
设计任务	(见附页)	
设计要求	(见附页)	
进度计划	时间	设计内容
	第一周 工作要求	<p>1. <u>解析分析方面</u> 构件运动关系分析：应用解析方法，按指定插齿机主运动机构传动方案，推导相关构件、执行构件的位移、速度、加速度表达式；（速度，加速度可以采用差商法实现）</p> <p>2. <u>可视化编程：</u> （1）实现构件尺寸、位置参数、转速等条件输入，编程判断结构组成的合理性、技术指标满足条件。 （2）形成初始状态位型，通过控制，使机构按时间变化形成运动动画，可以快、慢切换。 （3）编程计算相关构件、执行构件的位移（角位移）、速度（角速度）和加速度（角加速度）、形成数据文件，曲线图，并描述动态变化。</p>
	第二周 工作要求	<p>3. <u>解析分析</u> 基于运动分析的推导结果（包括相关构件、执行构件的广义加速度），按能量原理推导驱动构件需要的驱动力偶（驱动力）随曲柄转角的函数关系（考虑阻力和惯性力成分）。折算曲柄驱动力偶。按盈亏功求解思路，确定飞轮，并在考虑效率下确定电机。</p> <p>4. <u>可视化编程：</u> 编程计算并绘制不同曲柄转角位置的驱动力偶函数的变化规律。在次基础上，按原理确定盈亏功，完成飞轮设计计算。 （提高版，按构件体系平衡关系，分析有摩擦条件下的驱动构件力偶随曲柄转角的变化规律，在次基础上，按原理确定盈亏功，完成飞轮设计计算。）</p>

	验证 图解法结果	5. 程序结果分析与验证：能够通过程序验证图解法运动分析结果和飞轮转动惯量计算结果对比分析。
		课设答辩材料：编写整理课程设计说明书（包括 插齿机主运动机构 原理，机构运动符号分析，机构运动程序代码，实例软件分析结果描述），打印即可，准备答辩。
成绩评 定办法	1) 答辩 ：叙述清楚、内容完整，回答问题正确。（30 分） 2) 程序内容（40 分） （1）程序代码；（2）机构运动分析表达情况； （3）典型实例分析结果，数据、图形等。 3) 设计说明书 ：（30 分） 说明书格式规范、内容完整、分析正确。	
参考资料	教材：机械原理课程设计指导书（本校编） 参考书： 1) 孙桓等. 机械原理 (第 8 版). 北京：高等教育出版社，2013 2) 邹慧君等. 机械原理课程设计手册.北京：高等教育出版社，2010 3) 木林森工作室， Visual Basic 入门与技巧， 清华大学出版社，2000 4) 杨钰等，Mathematica 应用指南，人民邮电出版社，2000	

《机械原理与设计 1》课程设计任务书（附页）

一、设计条件与要求

条件：图 1 为插齿机主运动机构，曲柄 1 连续转动，带动执行构件 6 往复移动。

使用寿命：5 年，连续运转，三班制工作，大修期三年，工作中，载荷变化较大。机构的传动效率按 0.95 计算，按小批量生产规模设计。动力来源：电力，三相交流（220/380V）**要求：**(1) 机构的最小传动角不得小于 40° ，以提高传力性能；(2) 加速度变化范围大，以适应功能要求。

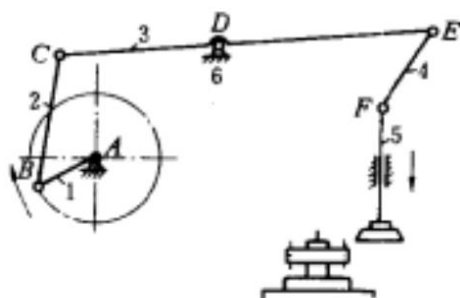


图 1 插齿机主运动机构

三. 计算分析任务

1. 明确此插齿机主运动机构的组成特点、工作原理、基本工作要求。

2. 针对此插齿机主运动机构图例，按解析法推导执行构件位置（角度）、速度（角速度）、加速度（角加速度）的表达式。

3. 在可视化程序环境下

(1) 针对给定的参数，可以判断机构是否满足技术要求；

(2) 针对给定的合理参数，完成机构运动分析动画；输出不同位置状态的执行构件 6 的位置、速度和加速度曲线。

(3) 针对插齿机主运动机构，按能量法，完成曲柄的阻力力偶与曲柄角度关系计算，并折算常驱动力偶，配置飞轮，选择电机。

(4) 按动态静力分析在考虑阻尼条件下计算效率。（此项根据情况完成）

编写设计说明书一份。完成准备《机械原理与设计 1》课程设计答辩。

《机械原理与设计 1》计算机分析课程设计说明书

目 录

1. 工作原理描述.....
2. 机构运动分析解析推导
3. 机构运动可视化程序设计
4. 能量法驱动力偶求解及飞轮计算.....
5. 实例分析检验.....
6. （考虑摩擦时的驱动力偶求解）
7. 设计体会.....

《机械原理与设计 1》计算机分析课程设计说明书

1. 工作原理描述

插齿机的运动由构件 1 驱动由构件 2、3、4 组成的连杆机构，再由构件 4 驱动构件 5 将运动转化为往复运动，实现设计的运动规律。

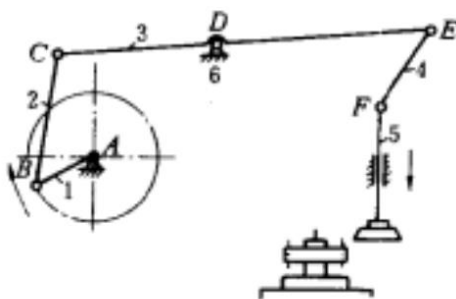


图 1 插齿机主运动机构

2. 机构运动分析解析推导

我们可将此五杆运动机构从图中 D 点分为两个部分：由杆 1、杆 2 和杆 3 左半部分（CD）组成的四连杆机构 I（ABCD），和由杆 3 右半部分（DE）、杆 4 和杆 5 所组成的摇杆导轨机构 II（DEFG）。我们在计算时可以将杆 5 替换为一个通过铰链 F 铰接在杆 4 上的滑块，这样机构 II 在运动规律不变的前提下就可等为摇杆滑块机构。因为机构 I 和机构 II 之间通过杆 3 链接，可以求出两构件的运动关系，我们不妨对机构 I 和机构 II 分开分析。

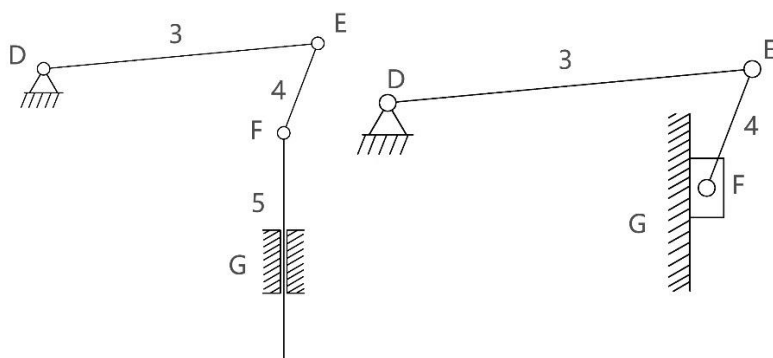


图 2 机构 II（DEFG）及其同运动规律等效机构

2.1. 四连杆机构 I (ABCD)

2.1.1. 求解机构 I 角位移

我们设构件 1、构件 2、构件 3 左半部分、机架 AD 的长度分别为 l_1 、 l_2 、 l_3^L 、 l_4 。可以得到如下封闭矢量方程：

$$\vec{l}_1 + \vec{l}_2 - \vec{l}_3 - \vec{l}_4 = 0 \quad (1)$$

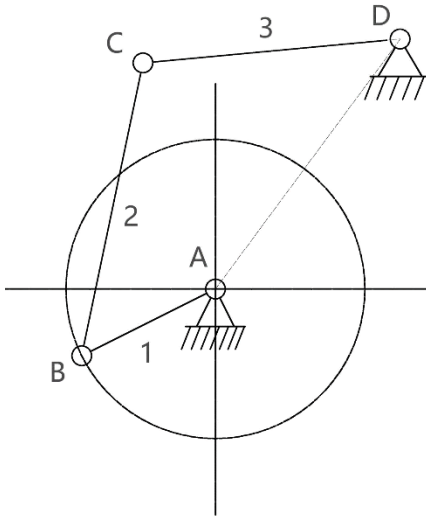


图 3 机构 I (ABCD)

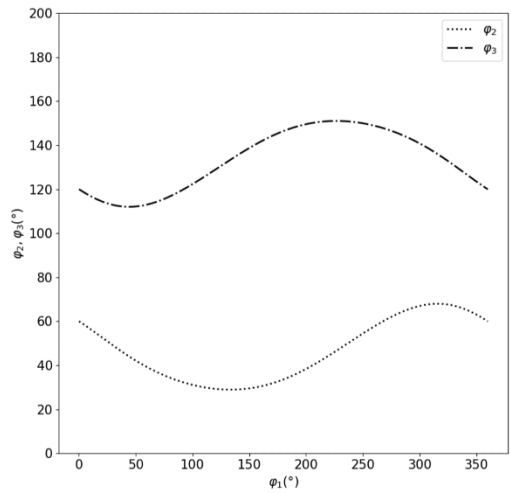


图 4 φ_1 和 φ_2 的角位移图

向 x, y 轴投影得：

$$\left. \begin{aligned} l_1 \cos \varphi_1 + l_2 \cos \varphi_2 - l_3^L \cos \varphi_3 - s_1 \cos \theta_1 &= 0 \\ l_1 \sin \varphi_1 + l_2 \sin \varphi_2 - l_3^L \sin \varphi_3 - s_1 \sin \theta_1 &= 0 \end{aligned} \right\} \quad (2)$$

其中 l_1 、 l_2 、 l_3^L 、 s_1 、 φ_1 、 θ_1 为设计机构的已知量， φ_2 和 φ_3 是未知量。我们把未知量放到一侧，已知量放到另一侧，化简得：

$$\left. \begin{aligned} l_2 \cos \varphi_2 - l_3^L \cos \varphi_3 &= s_1 \cos \theta_1 + l_1 \cos \varphi_1 = A \\ l_2 \sin \varphi_2 - l_3^L \sin \varphi_3 &= s_1 \sin \theta_1 + l_1 \sin \varphi_1 = B \end{aligned} \right\} \quad (3)$$

其中构件 1 的转角为 φ_1 ，构件 2 的转角为 φ_2 ，构件 3 左半部分的转角为 φ_3 。将矢量方程表示为复数适量形式，并用欧拉公式 $e^{i\varphi} = \cos \varphi + i\sin \varphi$ 将实部和虚部分离，得：

$$z_2 = l_2(\cos \varphi_2 + i\sin \varphi_2) \quad (4)$$

$$z_3 = l_3^L(\cos \varphi_3 + i\sin \varphi_3) \quad (5)$$

用 (1)–(2) 得：

$$z_2 - z_3 = A + iB = l_2 e^{i\varphi_2} - l_3 e^{i\varphi_3} \quad (6)$$

化简得：

$$e^{i\varphi_2} = \frac{A + iB + l_3 e^{i\varphi_3}}{l_2} \quad (7)$$

设 $l_2 e^{i\varphi_3} = u + iv$ ， $u = \cos \varphi_3$ ， $v = \sin \varphi_3$ 代入(5)，分离实部得：

$$\cos \varphi_2 = \frac{A + l_3 u}{l_2} \quad (8)$$

$$\sin \varphi_2 = \frac{B + l_3 v}{l_2} \quad (9)$$

其中 $u^2 + v^2 = 1$ ，这样，我们就可求出 φ_2 和 φ_3 。要注意我们需要根据实际运动规律对多解做取舍（见附件 1），经过对解的观察，我们可以确定解的范围为 [104,208]。

2.2.2. 求解机构 I 角速度和速度

每 1° 对角位移/位移进行差商求解角速度/速度。曲柄每转动 1° 所需时间为 $t = \frac{60}{360n_1}$ ，式中 n_1 为转速 [r/min]。

差商法求解角速度为：

$$\omega = \frac{\phi_{(i+1)} - \phi_{(i)}}{t} \quad (10)$$

差商法求解速度为：

$$v = \frac{x_{(i+1)} - x_{(i)}}{t} \quad (11)$$

2.2.3. 求解机构 I 角加速度和加速度

我们采用差商法求解速度与加速度，每 1° 对角位移/位移进行差商求解角速度/速度。曲柄每转动 1° 所需时间为 $t = \frac{60}{360n_1}$ ，式中 n_1 为转速 [r/min]。

差商法求解角加速度：

$$\omega = \frac{\phi_{(i+1)} - \phi_{(i)}}{t} \quad (12)$$

差商法求解加速度：

$$a = \frac{v_{(i+1)} - v_{(i)}}{t} \quad (13)$$

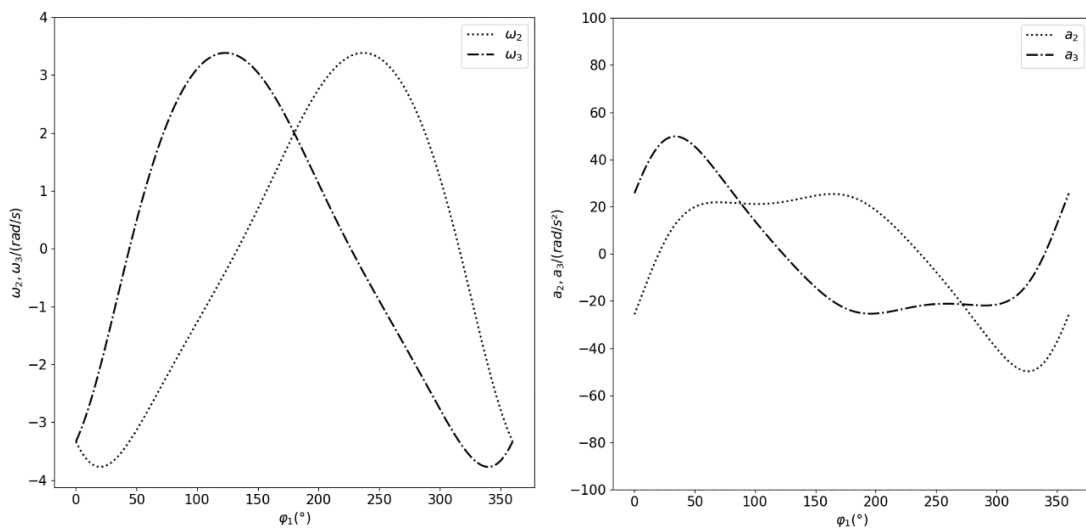


图 5 ϕ_1 和 ϕ_2 角速度与角加速度曲线图

2.2. 摇杆机构 II (DEFG)

2.2.1. 求解机构 II 角位移

我们设构件 3 右半部分长度、构件 4 的长度、支座到滑块运动面的距离、支座到滑块中心的距离分别为 l_3^R 、 l_4 、 e 、 s_2 。可以得到如下封闭矢量方程：

$$\vec{l}_3^R - \vec{s}_2 - \vec{e} - \vec{l}_4 = 0 \quad (12)$$

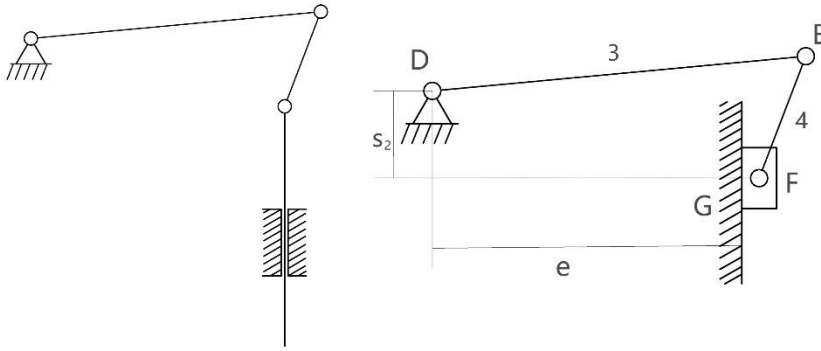


图 6 机构 II (DEFG) 及其同运动规律等效机构

向 x , y 轴投影得：

$$\left. \begin{aligned} l_3^R \cos \varphi_3^s - e - l_4 \cos \varphi_4 &= 0 \\ l_3^R \sin \varphi_3^s - s_2 \sin \theta_2 - l_4 \sin \varphi_4 &= 0 \end{aligned} \right\} \quad (13)$$

其中 l_3^R 、 e 、 l_4 、 θ_2 为设计机构的已知量，由几何关系可知 φ_3^s 数值上等于 φ_3 ， s_2 和 φ_4 是未知量。我们按照 2.1.1 求解机构 I 角位移 中所用方法，把未知量放到一侧，已知量放到另一侧，化简得：

$$\left. \begin{aligned} l_4 \cos \varphi_4 &= l_3^R \cos \varphi_3^s - e \\ l_4 \sin \varphi_4 + s_2 \sin \theta_2 &= l_3^R \sin \varphi_3^s \end{aligned} \right\} \quad (14)$$

这里要注意，我们并不需要使用复数去求解。我们通过观察可看出，(14) 中第一个方程仅有一个未知数，因此可以直接求出 φ_4 的解。之后再将 φ_4 的解带入 (14) 中第二个方程，求出 s_2 的解。

3. 机构运动可视化程序设计

3.1. 求解机构 I 的角位移

```
# 计算 phi3 角度的函数
def calculate_phi(l1, l2, l3, s1, phi1_deg, theta1_deg):
    phi1 = np.deg2rad(phi1_deg) # 将角度转换为弧度
    theta1 = np.deg2rad(theta1_deg) # 将角度转换为弧度

    phi2, phi3 = sp.symbols('phi2 phi3') # 定义符号变量

    # 建立方程
    eq1 = l1 * sp.cos(phi1) + l2 * sp.cos(phi2) - l3 * sp.cos(phi3) - s1 * sp.cos(theta1)
    eq2 = l1 * sp.sin(phi1) + l2 * sp.sin(phi2) - l3 * sp.sin(phi3) - s1 * sp.sin(theta1)

    # 解方程
    solutions = sp.solve([eq1, eq2], (phi2, phi3))
    phi2_solutions = [sol[0] for sol in solutions]
    phi3_solutions = [sol[1] for sol in solutions]

    # 计算角度值
    phi2_solutions_deg = [(float(sol.evalf()) * 180 / np.pi) % 360 for sol in phi2_solutions]
    phi3_solutions_deg = [(float(sol.evalf()) * 180 / np.pi) % 360 for sol in phi3_solutions]

    # 筛选合适的角度范围
    filtered_phi3_solutions = [sol for sol in phi3_solutions_deg if 103 <= sol <= 208]

    return phi2_solutions_deg, filtered_phi3_solutions
```

3.2. 求解机构 II 的角位移

```
def calculate_phi4_and_s2(l3r, l4, phi3_degrees, e):
    # Convert degrees to radians
    phi3_radians = math.radians(phi3_degrees)

    cos_phi4 = (l3r * math.cos(phi3_radians) - e) / l4
    sin_phi4_squared = 1 - cos_phi4 ** 2
    results = []

    if sin_phi4_squared >= 0:
        sin_phi4_positive = math.sqrt(sin_phi4_squared)
        # Calculate phi4 in radians for both positive and negative sin(phi4)
        phi4_positive = math.atan2(sin_phi4_positive, cos_phi4)
```

```

# Calculate s2 for both solutions
s2_positive = 14 * sin_phi4_positive - 13r * math.sin(phi3_radians)
# Append results in degrees for clarity
results.append(("Positive Solution", math.degrees(phi4_positive), s2_positive))
else:
    results.append(("No Real Solution", "N/A", "N/A"))

return results

```

3.3. 用差商法求角速度与角加速

```

# 运动学计算
def compute_kinematics(data_matrix, set_speed):
    # 计算角速度
    for i in range(1, data_matrix.shape[0]): # 从第二个时间点开始计算，因为第一个没有前一个状态
        for j in range(0, data_matrix.shape[2]): # 从第二列开始计算，第一列存放原始角度
            # 当前状态和前一个状态的角度差除以时间步长得到角速度
            data_matrix[i, 1, j] = (data_matrix[i, 0, j] - data_matrix[i - 1, 0, j]) / set_speed

    # 计算角加速度
    for i in range(1, data_matrix.shape[0]): # 从第二个时间点开始计算，因为第一个没有前一个状态
        for j in range(0, data_matrix.shape[2]): # 从第二列开始计算，第一列存放原始角度
            # 当前状态和前一个状态的角速度差除以时间步长得到角加速度
            data_matrix[i, 2, j] = (data_matrix[i, 1, j] - data_matrix[i - 1, 1, j]) / set_speed

```

3.4. 用求解出的角度计算出机构各点的坐标

```

# 计算信息矩阵 InfoMat
def InfoMat_calculation(data_matrix, l1, l3_left, l3_right, l4, l5, s1, theta1_deg):
    """
    计算机械连杆系统中各点的位置，并将每个 phi1 角度的结果存入一个三维矩阵,我们称之为信息矩阵 InfoMat。

    参数:
    - phi1_vals, phi3_vals: 角度数组（度），其中 phi1_vals[frame]是一个数组。
    - data_matrix: 包含额外角度数据的矩阵。
    - frame: 当前帧的索引。
    - l1, l3_left, l3_right, l4, l5, s1: 各杆的长度。
    - theta1_deg: 初始位置的固定角度。
    """

```

返回:

一个三维矩阵, 每个二维子矩阵包含对应 ϕ_1 角度计算得到的所有点坐标。

矩阵:

$-(\phi_1) * (x, y) * (A, B, C, D, E, F, G)$

"""

points_matrixs = [] # 初始化三维矩阵来存储所有点的坐标

frames = list(range(data_matrix.shape[0]))

遍历 ϕ_1_vals 中的每个 ϕ_1 角度

for frame in frames:

 # TotalAngleData = data_matrix[frame, 0, :]

$\phi_1 = \text{np.deg2rad}(\text{data_matrix}[\text{frame}, 0, 0])$

$\phi_3 = \text{np.deg2rad}(\text{data_matrix}[\text{frame}, 0, 2])$

 # $\phi_4 = \text{np.deg2rad}(\text{data_matrix}[\text{frame}, 0, 4])$

$\phi_{4_deg} = \text{data_matrix}[\text{frame}, 0, 4]$

$\phi_4 = \text{np.deg2rad}(180 - \phi_{4_deg})$

 # 计算各点坐标

$x_b, y_b = l_1 * \text{np.cos}(\phi_1), l_1 * \text{np.sin}(\phi_1)$

$x_d, y_d = s_1 * \text{np.cos}(\text{np.deg2rad}(\theta_{1_deg})), s_1 * \text{np.sin}(\text{np.deg2rad}(\theta_{1_deg}))$

$x_c, y_c = x_d + l_{3_left} * \text{np.cos}(\phi_3), y_d + l_{3_left} * \text{np.sin}(\phi_3)$

$\text{direction_x} = x_d - x_c$

$\text{direction_y} = y_d - y_c$

$\text{length_CD} = \text{np.sqrt}(\text{direction_x}^2 + \text{direction_y}^2)$

$\text{unit_x} = \text{direction_x} / \text{length_CD}$

$\text{unit_y} = \text{direction_y} / \text{length_CD}$

$x_e, y_e = x_d + l_{3_right} * \text{unit_x}, y_d + l_{3_right} * \text{unit_y}$

$x_f = x_e + l_4 * \text{np.cos}(\phi_4)$

$y_f = y_e - l_4 * \text{np.sin}(\phi_4)$

$x_g = x_f$

$y_g = y_f - l_5$

 # 构建当前 ϕ_1 角度的 points_matrix

 points_matrix = np.array([

 [0, 0],

$[x_b, y_b]$,

$[x_c, y_c]$,

$[x_d, y_d]$,

$[x_e, y_e]$,

$[x_f, y_f]$,

$[x_g, y_g]$

)

```
# 将当前 points_matrix 添加到三维数组中,再变化为矩阵
points_matrixs.append(points_matrix)
InfoMat = np.array(points_matrixs)
```

```
# 将列表转换为信息矩阵
return InfoMat
```

3.5. 根据各点位置画出插齿机运动仿真动画与执行机构的速度加速度曲线

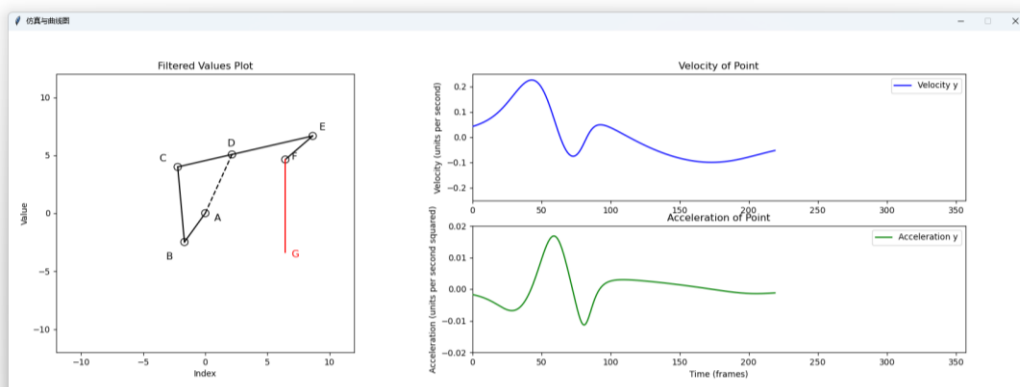


图 7 插齿机运动仿真动画与执行机构速度加速度曲线

设置绘图和动画

```
fig, ax = plt.subplots()
ax.set_xlim(-12, 12)
ax.set_ylim(-12, 12)

link1, = ax.plot([], [], 'k-')
link2, = ax.plot([], [], 'k-')
link3, = ax.plot([], [], 'k-')
link4, = ax.plot([], [], 'k--')
link5, = ax.plot([], [], 'k-') # 使用绿色线表示 CE 线段
link6, = ax.plot([], [], 'k-')
link7, = ax.plot([], [], 'r-')
```

创建文本标签元素

```
textA = ax.text(0, 0, "", fontsize=12, color='black')
textB = ax.text(0, 0, "", fontsize=12, color='black')
```

```

textC = ax.text(0, 0, " ", fontsize=12, color='black')
textD = ax.text(0, 0, " ", fontsize=12, color='black')
textE = ax.text(0, 0, " ", fontsize=12, color='black')
textF = ax.text(0, 0, " ", fontsize=12, color='black')
textG = ax.text(0, 0, " ", fontsize=12, color='red')

joint_radius = 0.3 # 铰链
joints = [plt.Circle((0, 0), joint_radius, fill=False, color='k') for _ in range(6)]
for joint in joints:
    ax.add_patch(joint)

# 初始化动画
def init():
    link1.set_data([], [])
    link2.set_data([], [])
    link3.set_data([], [])
    link4.set_data([], [])
    for joint in joints:
        joint.set_center((0, 0))
    # 初始化标签位置, 但不显示任何文本
    textA.set_text("")
    textB.set_text("")
    textC.set_text("")
    textD.set_text("")
    return link1, link2, link3, link4, *joints, textA, textB, textC, textD

# 更新动画
def update(frame):
    # 在函数内部可以直接使用 update.InfoMat 来引用 InfoMat 数据
    # InfoMat = update.InfoMat
    # 分别提取各点坐标
    xa, ya = InfoMat[frame, 0, :]
    xb, yb = InfoMat[frame, 1, :]
    xc, yc = InfoMat[frame, 2, :]
    xd, yd = InfoMat[frame, 3, :]
    xe, ye = InfoMat[frame, 4, :]
    xf, yf = InfoMat[frame, 5, :]
    xg, yg = InfoMat[frame, 6, :]

    link1.set_data([xa, xb], [ya, yb])
    link2.set_data([xb, xc], [yb, yc])
    link3.set_data([xc, xd], [yc, yd])
    link4.set_data([xd, xa], [yd, ya])
    link5.set_data([xd, xe], [yd, ye]) # 绘制 DE 线段
    link6.set_data([xe, xf], [ye, yf]) # 绘制 EF 线段
    link7.set_data([xf, xg], [yf, yg])

```

```

joints[0].set_center((xa, ya))
joints[1].set_center((xb, yb))
joints[2].set_center((xc, yc))
joints[3].set_center((xd, yd))
joints[4].set_center((xe, ye))
joints[5].set_center((xf, yf))

# 更新文本标签的位置和文本
textA.set_text('A')
textA.set_position((xa + 0.7, ya - 0.7))
textB.set_text('B')
textB.set_position((xb - 1.5, yb - 1.5))
textC.set_text('C')
textC.set_position((xc - 1.5, yc + 0.5))
textD.set_text('D')
textD.set_position((xd - 0.4, yd + 0.7))
textE.set_text('E')
textE.set_position((xe + 0.5, ye + 0.5))
textF.set_text('F')
textF.set_position((xf + 0.5, yf))
textG.set_text('G')
textG.set_position((xg + 0.5, yg - 0.4))

return link1, link2, link3, link4, link5, link6, link7, *joints, textA, textB, textC,
textD, textE, textF, textG

ani = animation.FuncAnimation(fig, update, frames=len(phi_vals), init_func=init,
                              blit=True, repeat=True, interval=set_speed)

va_fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(12, 8))

# 初始化两条线，一条用于速度，一条用于加速度
line1, = ax1.plot([], [], 'b-', label='Velocity y')
line2, = ax2.plot([], [], 'g-', label='Acceleration y')

# 设置图表标题和轴标签
ax1.set_title('Velocity of Point')
# ax1.set_xlabel('Time (frames)')
ax1.set_ylabel('Velocity (units per second)')
ax1.legend()
ax1.set_xlim(0, len(velocities) - 4) # 设置 x 轴范围
ax1.set_ylim(-0.25, 0.25) # 设置 y 轴范围

ax2.set_title('Acceleration of Point')
ax2.set_xlabel('Time (frames)')
ax2.set_ylabel('Acceleration (units per second squared)')
ax2.legend()

```



```

ax2.set_xlim(0, len(accelerations) - 4) # 设置 x 轴范围
ax2.set_ylim(-0.02, 0.02) # 设置 y 轴范围

def va_init():
    line1.set_data([], [])
    line2.set_data([], [])
    return line1, line2

def va_animate(frame):
    x_data = np.arange(0, frame + 1)
    y_data1 = velocities[:frame + 1, 1]
    y_data2 = accelerations[:frame + 1, 1]
    line1.set_data(x_data, y_data1)
    line2.set_data(x_data, y_data2)
    return line1, line2

va_ani = animation.FuncAnimation(va_fig, va_animate, frames=len(velocities),
init_func=va_init, blit=True, repeat=True, interval=set_speed)

return ani, va_ani, fig, va_fig

```

3.6. UI 设计与实现



图6 参数设置 UI

```

import tkinter as tk
import webbrowser
from tkinter import PhotoImage
from tkinter import messagebox
from PIL import Image, ImageTk
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from PIL import Image, ImageTk, ImageSequence
from simulation import *

def about():
    # 创建自定义的关于窗口
    about_window = tk.Toplevel(root)
    about_window.title("关于")
    about_window.geometry("400x300")
    about_window.resizable(False, False) # 禁止调整窗口大小

    # 创建标签显示关于信息
    about_label = tk.Label(about_window, text="这是一个插齿机运动仿真程序\n\n 指导老师：杨谢柳\n 作者：孟睿豪", padx=10, pady=10)
    about_label.pack()

    # 创建可点击的链接
    link = tk.Label(about_window, text="作者个人主页", fg="blue", cursor="hand2")
    link.pack()
    link.bind("<Button-1>", lambda e:
webbrowser.open_new("https://mengruihao.github.io/"))

    # 正确处理 GIF 图像序列
    gif_image = Image.open("彩虹小蓝猫.gif")
    gif_frames = [ImageTk.PhotoImage(image=img) for img in
ImageSequence.Iterator(gif_image)]

    gif_label = tk.Label(about_window)
    gif_label.pack()
    gif_index = 0

    def update_gif():
        nonlocal gif_index
        frame = gif_frames[gif_index]
        gif_label.configure(image=frame)
        gif_index = (gif_index + 1) % len(gif_frames)
        about_window.after(100, update_gif)

    update_gif()

def run_simulation():

```

```

l1 = float(entry_l1.get())
l2 = float(entry_l2.get())
l3_left = float(entry_l3_left.get())
l3_right = float(entry_l3_right.get())
l4 = float(entry_l4.get())
l5 = float(entry_l5.get())
theta = float(entry_theta.get())
v = float(entry_v.get())
s1 = float(entry_s1.get())
e = float(entry_e.get())

ani, va_ani, fig, va_fig = simulation(l1, l2, l3_left, l3_right, l4, l5, e, s1, theta, v)

# 创建一个顶层窗口来展示结果
result_window = tk.Toplevel(root)
result_window.title("仿真与曲线图")
result_window.geometry("1700x600") # 设置窗口的固定尺寸
result_window.resizable(False, False) # 禁止调整窗口大小

# 创建一个 Frame 用于仿真结果的显示
simulation_frame = tk.Frame(result_window)
simulation_frame.pack(side=tk.LEFT, fill=tk.BOTH, expand=True)

# 将仿真结果图形加入到 Frame 中
canvas1 = FigureCanvasTkAgg(fig, master=simulation_frame)
canvas1.draw()
canvas1.get_tk_widget().pack(side=tk.LEFT, fill=tk.BOTH, expand=True)

# 创建一个 Frame 用于曲线图的显示
curve_frame = tk.Frame(result_window)
curve_frame.pack(side=tk.RIGHT, fill=tk.BOTH, expand=True)

# 将曲线图加入到 Frame 中
canvas2 = FigureCanvasTkAgg(va_fig, master=curve_frame)
canvas2.draw()
canvas2.get_tk_widget().pack(side=tk.RIGHT, fill=tk.BOTH, expand=True)

# def ui_start():
# 创建主窗口
root = tk.Tk()
root.title("插齿机运动仿真")
root.geometry("720x300")
root.resizable(False, False) # 禁止调整窗口大小

# 创建菜单栏
menu_bar = tk.Menu(root)

```

```
# 创建“文件”菜单
file_menu = tk.Menu(menu_bar, tearoff=0)
file_menu.add_command(label="退出", command=root.quit)
menu_bar.add_cascade(label="文件", menu=file_menu)

# 创建“帮助”菜单
help_menu = tk.Menu(menu_bar, tearoff=0)
help_menu.add_command(label="关于", command=about)
menu_bar.add_cascade(label="帮助", menu=help_menu)

# 将菜单栏添加到主窗口
root.config(menu=menu_bar)

# 创建框架一来放置 l1 和 l2 的标签与输入框
frame_1 = tk.Frame(root)
frame_1.pack(anchor="w", pady=5)

# 创建并放置标签和输入框在同一行
label_l1 = tk.Label(frame_1, text="杆 l1 长度", width=10)
label_l1.pack(side="left")
entry_l1 = tk.Entry(frame_1, width=10)
entry_l1.pack(side="left")

label_l2 = tk.Label(frame_1, text="杆 l2 长度", width=10)
label_l2.pack(side="left", padx=(20, 0)) # 添加一些水平间距
entry_l2 = tk.Entry(frame_1, width=10)
entry_l2.pack(side="left")

# 创建框架二来放置 l3 的标签与输入框
frame_2 = tk.Frame(root)
frame_2.pack(anchor="w", pady=5)

# 创建并放置标签和输入框在同一行
label_l3_left = tk.Label(frame_2, text="杆 l3 左侧长度", width=10)
label_l3_left.pack(side="left")
entry_l3_left = tk.Entry(frame_2, width=10)
entry_l3_left.pack(side="left")

label_l3_right = tk.Label(frame_2, text="杆 l3 右侧长度", width=10)
label_l3_right.pack(side="left", padx=(20, 0)) # 添加一些水平间距
entry_l3_right = tk.Entry(frame_2, width=10)
entry_l3_right.pack(side="left")

# 创建框架三来放置 l4 和 l5 的标签与输入框
```

```
frame_3 = tk.Frame(root)
frame_3.pack(anchor="w", pady=5)

# 创建并放置标签和输入框在同一行
label_l4 = tk.Label(frame_3, text="杆 l4 长度", width=10)
label_l4.pack(side="left")
entry_l4 = tk.Entry(frame_3, width=10)
entry_l4.pack(side="left")

label_l5 = tk.Label(frame_3, text="杆 l5 长度", width=10)
label_l5.pack(side="left", padx=(20, 0)) # 添加一些水平间距
entry_l5 = tk.Entry(frame_3, width=10)
entry_l5.pack(side="left")

# 创建框架四来放置 l4 和 l5 的标签与输入框
frame_4 = tk.Frame(root)
frame_4.pack(anchor="w", pady=5)

# 创建并放置标签和输入框在同一行
label_s1 = tk.Label(frame_4, text="距离 S1", width=10)
label_s1.pack(side="left")
entry_s1 = tk.Entry(frame_4, width=10)
entry_s1.pack(side="left")

label_e = tk.Label(frame_4, text="距离 e", width=10)
label_e.pack(side="left", padx=(20, 0)) # 添加一些水平间距
entry_e = tk.Entry(frame_4, width=10)
entry_e.pack(side="left")

# 创建框架五来放置 l4 和 l5 的标签与输入框
frame_5 = tk.Frame(root)
frame_5.pack(anchor="w", pady=5)

# 创建并放置标签和输入框在同一行
label_theta = tk.Label(frame_5, text="角度  $\theta$ ", width=10)
label_theta.pack(side="left")
entry_theta = tk.Entry(frame_5, width=10)
entry_theta.pack(side="left")

label_v = tk.Label(frame_5, text="角速度 w", width=10)
label_v.pack(side="left", padx=(20, 0)) # 添加一些水平间距
entry_v = tk.Entry(frame_5, width=10)
entry_v.pack(side="left")

# 创建并放置按钮
button_login = tk.Button(root, text="运行仿真", command=run_simulation)
```

```
button_login.pack(pady=20)




# 等待一段时间再进入 root.mainloop()
root.after(1000000, lambda: None) # 等待 2 秒

# 运行主循环
root.mainloop()
print("结束哩！")
```

5. 设计体会

本次课设对我的机械和编程水平有极大的提升。在本次课设中在编程前我利用机械原理课上所学到的知识，使用解析法对机构的运动状态进行计算分析。然后利用所得数据去求解速度加速度，做平衡力矩分析。这个过程使我对机械原理有了更深的理解与掌握。此外，因为我在简锋实验室视觉组里学习了两年，有一定的编程基础，所以我决定摒弃 VB 转而使用 Python 去完成解析任务。在这个过程中我使用了 numpy、sympy、matplotlib、thinker 和 tqdm 等库以及其依赖库。虽然之前都听说过且简单使用过这些库，但从来没有完成如此大的工程。虽然我编程过程中遇到了很多困难与问题，但通过不懈努力大体都得到了解决。

6. 附件说明

每个 φ_1 对应的 φ_3 值 (含多解及其取舍)	每个 φ_1 对应的 φ_4 和 s_3 的值	每个 φ_1 对应的 φ_2 、 φ_3 值
 phi3_solutions.xls x	 results_phi4.xlsx	 informat.xlsx