Unit Test Documentation for Four-Bar Linkage

October 13, 2024

Overview of the Test Collection

The unit tests for the Four-Bar Linkage system are designed to verify the correct classification of linkage motion types. Each test checks whether the linkage is correctly identified as a specific combination of crank, rocker, and intermediate states based on the values of T_1, T_2, and T_3. These tests ensure that for various configurations of the four-bar linkage, the input and output links are classified accurately. The classifications are important for understanding the behavior of the system and ensuring correct functionality in different scenarios.

The test cases systematically cover all possible combinations of positive, negative, and zero values for the parameters T_1, T_2, and T_3, ensuring that every possible motion configuration is tested. The expected outcomes are predefined according to the known behavior of four-bar linkages.

Explanation of Test Collection

Each collection of test cases verifies a specific motion classification of the four-bar linkage system. The classifications generally fall into the categories of Crank, Rocker, and intermediate states like π-rocker or 0-rocker.

Crank-Rocker Classifications

The tests in this group verify the combinations where one of the links acts as a crank while the other behaves as a rocker.

- **test_case_1**: Tests a scenario where all the parameters T_1, T_2, and T_3 are positive. The expected classification is crank for the input link and rocker for the output link. - **test_case_4**: Verifies the scenario where $T_1 > 0$, $T_2 = 0$, and $T_3 > 0$, leading to a crank on the input and 0-rocker on the output. - **test_case_10**: Tests the scenario where $T_1 > 0$, $T_2 > 0$, and $T_3 = 0$. The input link is classified as a crank and the output as π-rocker. article amsmath [utf8]inputenc

Comprehensive Documentation of the FourBarLinkage Class in Python OpenAI - ChatGPT October 13, 2024

1 Introduction

The `FourBarLinkage` class in Python simulates a four-bar linkage, which is widely used in kinematic mechanisms. This class allows for the modeling of a system consisting of four rigid bars connected by rotary joints. Depending on the lengths of the bars, different types of linkage mechanisms can be realized, such as crank-rocker, double-rocker, and others.

2 Class Attributes

The class contains the following attributes, which define the fundamental parameters and states of the four-bar linkage:

2.1 Geometrical Parameters

- `AB`, `BC`, `CD`, `DA`: The lengths of the bars, representing the distances between the linkage's connection points.

2.2 Angles and Angular Velocity

- `alpha`: The angle of the input link in degrees.
- `theta`: The angle of the fixed link in degrees.
- `alpha_rad`, `theta_rad`: The angles `alpha` and `theta` in radians (automatically converted).
- `alpha_velocity`: The angular velocity of the input link, in degrees per second.

2.3 Time Parameters and Mode Switches

- `t`: The time interval used for the simulation.
- `C_mode`: A control to switch between the two possible positions of point C (`'C1'` or `'C2'`).

3 Methods of the FourBarLinkage Class

This section provides detailed descriptions of the methods within the class. These methods are responsible for calculating the kinematics and dynamics of the four-bar linkage.

3.1 __init__() - Constructor

- **Location:** Defined at the beginning of the class.
- **Description:** Initializes all relevant parameters of the four-bar linkage (link lengths, angles, time intervals, etc.). It also converts angles from degrees to radians and sets the initial state of the linkage.
- **Parameters:**
  - `AB, BC, CD, DA`: The lengths of the four bars.
  - `alpha, theta`: The input and fixed link angles.
  - `coupler_position, coupler_offset`: The position and offset of the coupler link.
  - `timeinterval, alpha_velocity`: Time interval and angular velocity.
- **Return:** None.

3.2 init_default_values() - Initialize Default Values

- **Location:** Defined after the constructor.
- **Description:** Initializes default values such as geometric validity, mode switches, and input/output link types.
- **Return:** None.

3.3 run() - Start Simulation

- **Location:** Defined after `init_default_values`.
- **Description:** Starts the simulation by calculating all necessary parameters, classification values, and point positions. This method triggers the entire kinematic calculation process.
- **Return:** None.

3.4 calculate_Classification_Value() - Calculate Classification Values

- **Location:** Called within `run`.
- **Description:** This method computes the classification values T_1, T_2, and T_3, which determine the type of the four-bar linkage:

$$T_1 = AB + CD - BC - DA$$
$$T_2 = BC + AB - CD - DA$$
$$T_3 = CD + BC - AB - DA$$

- **Return:** None.

3.5 check_Parameter() - Check Parameters

- **Location:** Called within `run`.
- **Description:** Checks the geometric validity of the linkage, ensuring that the configuration satisfies the Grashof criterion and other geometry constraints.
- **Return:** None.

3.6 find_Linkage_Type() - Find Linkage Type

- **Location:** Called within `run`.
- **Description:** Classifies the type of four-bar linkage (e.g., crank-rocker or double-rocker) based on the values of T_1, T_2, and T_3.
- **Return:** None.

3.7 calculate_Point_Position() - Calculate Point Positions

- **Location:** Called within `run`.
- **Description:** Computes the positions of the points A, B, C, D, and P based on the input parameters (link lengths and angles).
- **Return:** None.

3.8 calculate_alpha_lims() - Calculate Alpha Limits

- **Location:** Called within `run`.
- **Description:** Calculates the limits for the angle `alpha` to ensure the linkage forms a valid closed configuration throughout its movement.
- **Return:** None.

3.9 calculate_C_Position() - Calculate Position of C

- **Location:** Called within `calculate_Point_Position`.
- **Description:** Computes the possible positions of point C (C1 and C2) based on the current configuration. It then selects the appropriate position based on the mode.
- **Return:** None.

3.10 calculate_P_Position() - Calculate Position of P

- **Location:** Called within `calculate_Point_Position`.
- **Description:** Calculates the position of the coupler point P using the coupler position and offset relative to link CD.
- **Return:** None.

3.11 animation_alpha() - Animate Alpha Angle

- **Location:** Defined near the end of the class.
- **Description:** Updates the angle `alpha` during each step of the animation. When the angle reaches its limits, the direction of motion is reversed for a continuous looping animation.
- **Return:** None.

3.12 switch_C2_C1() - Switch Between C1 and C2

- **Location:** Defined near the end of the class.
- **Description:** Switches between the two possible positions for point C (C1 and C2). This is necessary when the mechanism transitions between different configurations where either position is valid.
- **Return:** None.

4 GUI Implementation

In addition to the kinematic calculations, a graphical user interface (GUI) is implemented to visualize the four-bar linkage and allow user interaction.

4.1 GUI Class Constructor (__init__())

- **Location:** The `GUI` class is defined after the `FourBarLinkage` class.
- **Description:** Initializes the GUI components, including the canvas for drawing the linkage, sliders for adjusting the linkage's parameters, and buttons for starting or resetting the simulation.
- **Return:** None.

4.2 init_linkage_display()

- **Location:** Within the `GUI` class.
- **Description:** Initializes all graphical elements needed to display the linkage, such as the lines for the bars, the points A, B, C, D, and the coupler point P.
- **Return:** None.

4.3 refresh()

- **Location:** Within the `GUI` class.
- **Description:** Updates the display of the linkage on the canvas whenever a parameter is changed by the user. This method is called every time the sliders are adjusted or a new configuration is applied.
- **Return:** None.

4.4 run_animation()

- **Location:** Within the `GUI` class.
- **Description:** Runs the continuous animation of the four-bar linkage, updating the positions of the points and the angles as the linkage moves.
- **Return:** None.

5 Test Cases

To ensure the correctness of the `FourBarLinkage` class, several unit tests are provided.

5.1 Test Case: Crank-Rocker Mechanism

- **Location:** In the test class `TestFourBarLinkageCases`.
- **Description:** Tests the configuration where $T_1 > 0$, $T_2 > 0$, and $T_3 > 0$, which corresponds to a crank-rocker mechanism. The lengths are set to values that satisfy these conditions.
- **Expected Result:** The system should classify the linkage as a crank-rocker.

5.2 Test Case: Double Crank

- **Location:** In the test class `TestFourBarLinkageCases`.
- **Description:** Tests the configuration where $T_1 > 0$, $T_2 = 0$, and $T_3 > 0$, which corresponds to a double crank mechanism.
- **Expected Result:** The system should classify the linkage as a double crank.

5.3 Test Case: Double Rocker

- **Location:** In the test class `TestFourBarLinkageCases`.
- **Description:** Tests the configuration where $T_1 < 0$, $T_2 < 0$, and $T_3 < 0$, which corresponds to a double rocker mechanism.
- **Expected Result:** The system should classify the linkage as a double rocker.

6 Conclusion

The `FourBarLinkage` class, along with its accompanying GUI and test cases, provides a comprehensive simulation of a four-bar linkage mechanism. The class supports different types of mechanisms based on the lengths of the bars and allows for dynamic visualization through the GUI.