In [5]:

```
import modern_robotics as mr
import numpy as np

R = 0.1 # (const) wheel radius
L = 0.1 # (const) distance to wheel in x_b direction
W = 0.1 # (const) distance to wheel in y_b direction
```

NextState:

Args:

- curr_state
  Vector (12x1) containing the current angle of each joint
  - Chasis (0 - 2)
  - Arm Joints (3 - 7)
  - Wheel (8 - 11)
- velocities
  Vector (9x1) containing joint and wheel velocities
  - Arm Joints (0 - 4)
  - Wheel Velocities (5 - 8)
- dt
  (float) value of the timestep
- max_vel
  (float) maximum velocity for joint and wheel movements

In [6]:

```
def NextState(curr_state: np.ndarray, velocities: np.ndarray, dt: float, max_vel: float) -> np.ndarray:
    next_state = np.zeros(12)

    # determine the new joint angles
    next_state[3:8] = curr_state[3:8] + dt * velocities[:5]

    # determine the new wheel angles
    next_state[8:] = curr_state[8:] + dt * velocities[5:]

    # calculate the change in wheel angles
    d_wheel_angles = next_state[8:] - curr_state[8:]

    # eqn 13.10 (pg. 541)
    H_0 =  (1/R) * np.array([[-L-W, 1, -1], [L+W, 1, 1], [L+W, 1, -1], [-L-W, 1, 1]])

    # body twist is a 3x1 vector
    # eqn 13.33 (pg. 569)
    #   w_bz
    #   v_bx
    #   v_by
    body_twist = np.matmul(np.linalg.pinv(H_0), d_wheel_angles)

    # eqn 13.35 (pg. 570)
    dq_b = np.zeros(3)
    if body_twist[0] == 0.0:
        dq_b = body_twist
    else:
        dq_b = np.array([
            body_twist[0],
            ( body_twist[1] * np.sin(body_twist[0]) + body_twist[2] * (np.cos(body_twist[0]) - 1) ) / body_twist[0],
            ( body_twist[2] * np.sin(body_twist[0]) + body_twist[1] * (1 - np.cos(body_twist[0])) ) / body_twist[0]
        ])

    # eqn 13.36 (pg 570)
    dq = np.matmul(np.array([
        [1, 0, 0],
        [0, np.cos(curr_state[0]), -np.sin(curr_state[0])],
        [0, np.sin(curr_state[0]), np.cos(curr_state[0])],
    ]), dq_b)

    next_state[:3] = curr_state[:3] + dq

    return next_state
```

Testing the Code:

In [7]:

```
import csv

dt = 0.05
N = 100 * int(dt / 0.01)

curr_state = np.zeros(12)
velocities = np.ones(9)
velocities[:5] *= 0.125
velocities[5] *= 0.5
velocities[6] *= 0.25
velocities[7] *= 0.25
velocities[8] *= 0.5

with open("test.csv", "w+") as file:
    writer = csv.writer(file)
    write_data = np.zeros(13)
    for _ in range(N):
        write_data[:12] = curr_state
        writer.writerow(write_data.round(4))
        curr_state = NextState(curr_state, velocities, dt, 10)
```

Video Link

link