In [147…
```
%matplotlib widget
import numpy as np
import scipy as sp
import matplotlib.pyplot as plt
import so101
from math_utils import hat_twist, unhat_twist, transformation_adjoint, tange
```

Generate the Path that we want to follow

In [148…
```
# Basic parameters
# increasing the number of points makes it more accurate however I am leavir
num_points_target = 25
# Interpolating in between IK points to see how the continuous function beha
joint_space_interpolation_factor = 100
```

In [149…
```
# Straight line path in task space
start_point = np.array([0.05, -0.05, 0.05])
end_point = np.array([0.05, 0.05, 0.05])
rotation = np.array([
    [0, 1, 0],
    [0, 0, -1],
    [-1, 0, 0]
])
displacements = np.linspace(start_point, end_point, num_points_target)
task_space_path = [transformation_matrix(rotation, disp) for disp in displac

displacements_interpolated = np.linspace(start_point, end_point, ((num_point
task_space_path_interpolated = [transformation_matrix(rotation, disp) for di
```

Joint space calculations

In [150…
```
def joint_space_trajectory_interpolation(joint_space_path, factor):
    interpolated_path = []
    for i in range(len(joint_space_path) - 1):
        start_angles = joint_space_path[i]
        end_angles = joint_space_path[i + 1]
        for t in np.linspace(0, 1, factor, endpoint=False):
            interp_angles = (1 - t) * start_angles + t * end_angles
            interpolated_path.append(interp_angles)
    interpolated_path.append(joint_space_path[-1])  # Append the last point
    return interpolated_path
```

In [151…
```
def plot_trajectories(joint_space_path, end_effector_path, displacements, me
    # Plot joint angles and end-effector trajectory side-by-side
    joint_space_path = np.array(joint_space_path)
    end_effector_path = np.array(end_effector_path)

    fig = plt.figure(figsize=(12, 5))

    # Left subplot: joint angles over time
    ax1 = fig.add_subplot(1, 2, 1)
    for i in range(joint_space_path.shape[1]):
```

```
        ax1.plot(joint_space_path[:, i], label=f'Joint {i+1}')
    ax1.set_title(f'Joint Angles over Time ({method})')
    ax1.set_xlabel('Time Step')
    ax1.set_ylabel('Joint Angle (radians)')
    ax1.legend()
    ax1.grid(True)

    # Right subplot: 3D target vs end-effector trajectory
    ax2 = fig.add_subplot(1, 2, 2, projection='3d')
    ax2.plot(displacements[:, 0], displacements[:, 1], displacements[:, 2],
    ax2.plot(end_effector_path[:, 0], end_effector_path[:, 1], end_effector_
    ax2.set_title(f'End-Effector Trajectory vs Target Trajectory ({method})'
    ax2.set_xlabel('X (m)')
    ax2.set_ylabel('Y (m)')
    ax2.set_zlabel('Z (m)')
    ax2.legend()
    # Set axes limits to -1..1 for consistent scale
    ax2.set_xlim([0.04, 0.06])
    ax2.set_ylim([-0.1, 0.1])
    ax2.set_zlim([0.04, 0.06])
    # Try to enforce equal aspect ratio (requires matplotlib >= 3.3). If una
    plt.tight_layout()
    plt.show()
```

starting with the inverse kinematics

In [152…

```
joint_space_path_ik = []
joint_angles = None
for g_target in task_space_path:
    joint_angles = so101.SO101._inverse_kinematics(g_target, joint_angles)
    joint_space_path_ik.append(joint_angles)

joint_space_path_ik = joint_space_trajectory_interpolation(joint_space_path_
end_effector_path_ik = []
for joint_angles in joint_space_path_ik:
    gwe = so101.SO101._forward_kinematics(joint_angles)
    end_effector_path_ik.append(gwe[:3, 3])
```

```
================ Inverse Kinematics Result ================
IK computation time: 0.0317 seconds
IK optimization success: True, message: Optimization terminated successfull
y.
Final IK error norm: 2.2194673112137136e-11
===========================================================
================ Inverse Kinematics Result ================
IK computation time: 0.0209 seconds
IK optimization success: True, message: Optimization terminated successfull
y.
Final IK error norm: 3.7197250990929035e-11
===========================================================
================ Inverse Kinematics Result ================
IK computation time: 0.0225 seconds
IK optimization success: True, message: Optimization terminated successfull
y.
Final IK error norm: 3.876626672827686e-12
===========================================================
================ Inverse Kinematics Result ================
IK computation time: 0.0203 seconds
IK optimization success: True, message: Optimization terminated successfull
y.
Final IK error norm: 2.6192549581140127e-11
===========================================================
================ Inverse Kinematics Result ================
IK computation time: 0.0214 seconds
IK optimization success: True, message: Optimization terminated successfull
y.
Final IK error norm: 1.1177306874704765e-10
===========================================================
================ Inverse Kinematics Result ================
IK computation time: 0.0218 seconds
IK optimization success: True, message: Optimization terminated successfull
y.
Final IK error norm: 3.9613292096423403e-11
===========================================================
================ Inverse Kinematics Result ================
IK computation time: 0.0219 seconds
IK optimization success: True, message: Optimization terminated successfull
y.
Final IK error norm: 9.293172197781805e-11
===========================================================
================ Inverse Kinematics Result ================
IK computation time: 0.0219 seconds
IK optimization success: True, message: Optimization terminated successfull
y.
Final IK error norm: 2.210220932725235e-11
===========================================================
================ Inverse Kinematics Result ================
IK computation time: 0.0225 seconds
IK optimization success: True, message: Optimization terminated successfull
y.
Final IK error norm: 8.523622681705257e-12
===========================================================
================ Inverse Kinematics Result ================
IK computation time: 0.0226 seconds
```

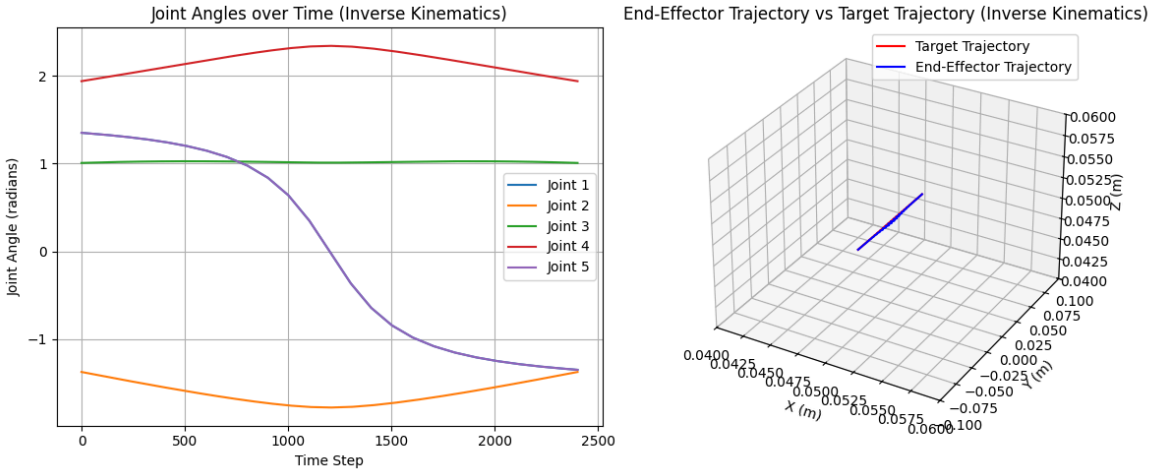
```
IK optimization success: True, message: Optimization terminated successfull
y.
Final IK error norm: 1.480696101572585e-11
============================================================
================ Inverse Kinematics Result ================
IK computation time: 0.0236 seconds
IK optimization success: True, message: Optimization terminated successfull
y.
Final IK error norm: 3.1072826183723656e-11
============================================================
================ Inverse Kinematics Result ================
IK computation time: 0.0228 seconds
IK optimization success: True, message: Optimization terminated successfull
y.
Final IK error norm: 1.6763671282161853e-12
============================================================
================ Inverse Kinematics Result ================
IK computation time: 0.0201 seconds
IK optimization success: True, message: Optimization terminated successfull
y.
Final IK error norm: 4.967125050229074e-11
============================================================
================ Inverse Kinematics Result ================
IK computation time: 0.0212 seconds
IK optimization success: True, message: Optimization terminated successfull
y.
Final IK error norm: 1.4196337284359576e-11
============================================================
================ Inverse Kinematics Result ================
IK computation time: 0.0208 seconds
IK optimization success: True, message: Optimization terminated successfull
y.
Final IK error norm: 1.6012366300697958e-10
============================================================
================ Inverse Kinematics Result ================
IK computation time: 0.0217 seconds
IK optimization success: True, message: Optimization terminated successfull
y.
Final IK error norm: 4.0133118399449595e-12
============================================================
================ Inverse Kinematics Result ================
IK computation time: 0.0208 seconds
IK optimization success: True, message: Optimization terminated successfull
y.
Final IK error norm: 2.5579083477678774e-10
============================================================
================ Inverse Kinematics Result ================
IK computation time: 0.0213 seconds
IK optimization success: True, message: Optimization terminated successfull
y.
Final IK error norm: 3.774013070911835e-11
============================================================
================ Inverse Kinematics Result ================
IK computation time: 0.0216 seconds
IK optimization success: True, message: Optimization terminated successfull
y.
```

```
Final IK error norm: 6.491200030811849e-12
============================================================
================ Inverse Kinematics Result ================
IK computation time: 0.0225 seconds
IK optimization success: True, message: Optimization terminated successfull
y.
Final IK error norm: 8.702690659236205e-12
============================================================
================ Inverse Kinematics Result ================
IK computation time: 0.0211 seconds
IK optimization success: True, message: Optimization terminated successfull
y.
Final IK error norm: 1.0483981238042061e-10
============================================================
================ Inverse Kinematics Result ================
IK computation time: 0.0216 seconds
IK optimization success: True, message: Optimization terminated successfull
y.
Final IK error norm: 2.3454346986492873e-11
============================================================
================ Inverse Kinematics Result ================
IK computation time: 0.0212 seconds
IK optimization success: True, message: Optimization terminated successfull
y.
Final IK error norm: 3.1992894562256635e-11
============================================================
================ Inverse Kinematics Result ================
IK computation time: 0.0206 seconds
IK optimization success: True, message: Optimization terminated successfull
y.
Final IK error norm: 1.0619879497382942e-10
============================================================
================ Inverse Kinematics Result ================
IK computation time: 0.0218 seconds
IK optimization success: True, message: Optimization terminated successfull
y.
Final IK error norm: 1.631722369750201e-11
============================================================
```

In [153…

```
plot_trajectories(joint_space_path_ik, end_effector_path_ik, displacements,
```

Figure



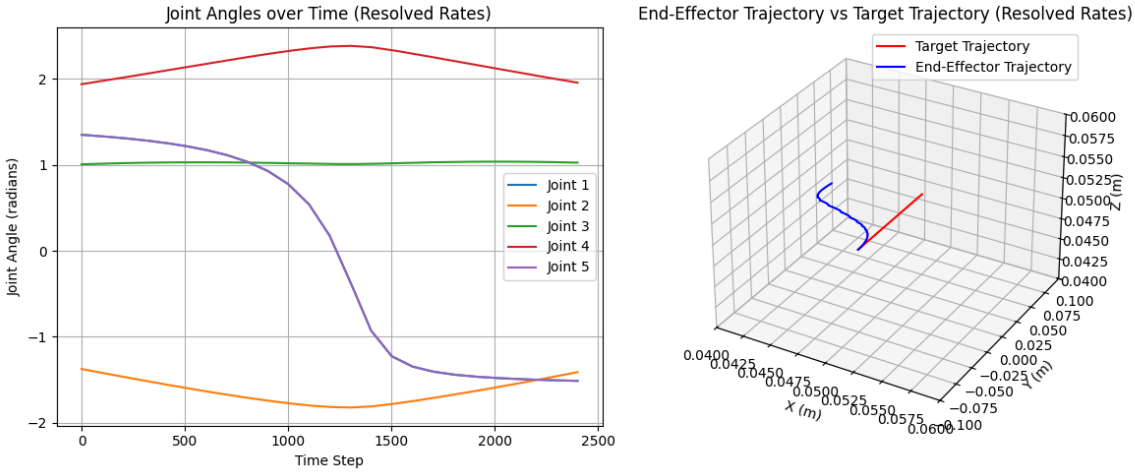
Resolved Rate

In [154…
```
joint_space_path_rr = np.zeros((len(task_space_path), so101.SO101.num_joints
joint_space_path_rr[0] = so101.SO101._inverse_kinematics(task_space_path[0])

for i in range(1, len(task_space_path)):
    gwe = task_space_path[i - 1]
    g_w_ep1 = task_space_path[i]
    g_e_ep1 = np.linalg.inv(gwe) @ g_w_ep1
    twist = unhat_twist(sp.linalg.logm(g_e_ep1))
    Jb = so101.SO101._body_jacobian(joint_space_path_rr[i - 1])
    Jb_dagger = np.linalg.pinv(Jb)
    joint_space_path_rr[i] = joint_space_path_rr[i - 1] + Jb_dagger @ twist

joint_space_path_rr = joint_space_trajectory_interpolation(joint_space_path_
end_effector_path_rr = []
for joint_angles in joint_space_path_rr:
    gwe = so101.SO101._forward_kinematics(joint_angles)
    end_effector_path_rr.append(gwe[:3, 3])
```

```
================= Inverse Kinematics Result ================
IK computation time: 0.0563 seconds
IK optimization success: True, message: Optimization terminated successfull
y.
Final IK error norm: 2.2194673112137136e-11
============================================================
```

In [155…
```
plot_trajectories(joint_space_path_rr, end_effector_path_rr, displacements,
```

Figure



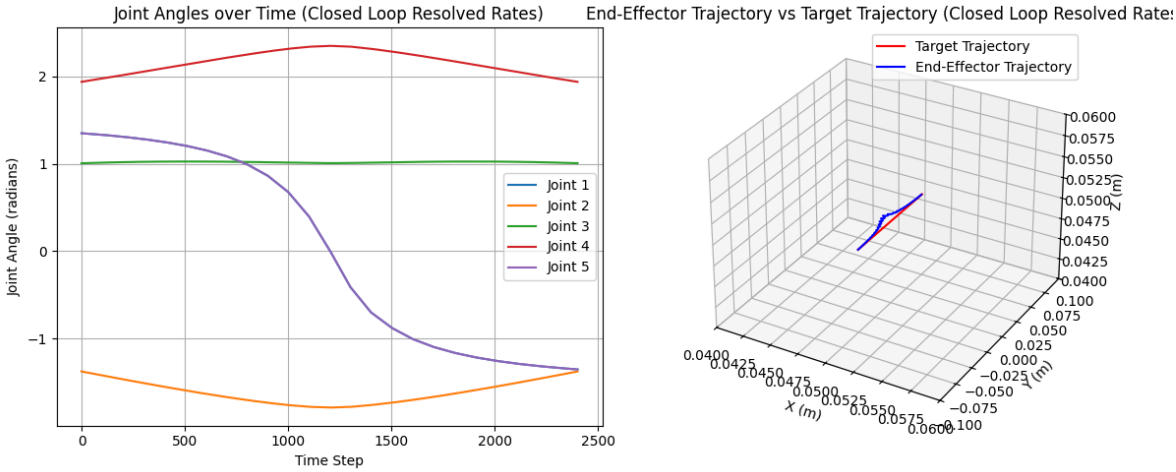
"Closed Loop" Resolved Rate

In [156…
```
joint_space_path_rrcl = np.zeros((len(task_space_path), so101.SO101.num_joir
joint_space_path_rrcl[0] = so101.SO101._inverse_kinematics(task_space_path[0

for i in range(1, len(task_space_path)):
    gwe = so101.SO101._forward_kinematics(joint_space_path_rrcl[i - 1])
    g_w_ep1 = task_space_path[i]
    g_e_ep1 = np.linalg.inv(gwe) @ g_w_ep1
    twist = unhat_twist(sp.linalg.logm(g_e_ep1))
    Jb = so101.SO101._body_jacobian(joint_space_path_rrcl[i - 1])
    Jb_dagger = np.linalg.pinv(Jb)
    joint_space_path_rrcl[i] = joint_space_path_rrcl[i - 1] + Jb_dagger @ tw

joint_space_path_rrcl = joint_space_trajectory_interpolation(joint_space_pat
end_effector_path_rrcl = []
for joint_angles in joint_space_path_rrcl:
    gwe = so101.SO101._forward_kinematics(joint_angles)
    end_effector_path_rrcl.append(gwe[:3, 3])
```

```
================= Inverse Kinematics Result ================
IK computation time: 0.0319 seconds
IK optimization success: True, message: Optimization terminated successfull
y.
Final IK error norm: 2.2194673112137136e-11
============================================================
```

In [157…
```
plot_trajectories(joint_space_path_rrcl, end_effector_path_rrcl, displacemer
```

Figure



Error Comparison

In [158…

```
# Plot the closed loop position errors for the resolved rate control with cl
errors_rrcl = []
errors_rr = []
errors_ik = []
print(len(task_space_path_interpolated))
print(len(joint_space_path_rrcl))
for i in range(len(task_space_path_interpolated)):
    gwe_target = task_space_path_interpolated[i]
    pos_target = gwe_target[:3, 3]

    gwe_rrcl = so101.SO101._forward_kinematics(joint_space_path_rrcl[i])
    pos_rrcl = gwe_rrcl[:3, 3]
    error_rrcl = np.linalg.norm(pos_target - pos_rrcl)
    errors_rrcl.append(error_rrcl)

    gwe_rr = so101.SO101._forward_kinematics(joint_space_path_rr[i])
    pos_rr = gwe_rr[:3, 3]
    error_rr = np.linalg.norm(pos_target - pos_rr)
    errors_rr.append(error_rr)

    gwe_ik = so101.SO101._forward_kinematics(joint_space_path_ik[i])
    pos_ik = gwe_ik[:3, 3]
    error_ik = np.linalg.norm(pos_target - pos_ik)
    errors_ik.append(error_ik)

# Plotting the errors
plt.figure(figsize=(8, 5))
plt.plot(errors_ik, label='Inverse Kinematics Error', color='g')
plt.plot(errors_rr, label='Resolved Rate Control Error', color='b')
plt.plot(errors_rrcl, label='Resolved Rate Control with Closed Loop Error',
plt.yscale('log')
plt.title('End-Effector Position Error Comparison')
plt.xlabel('Time Step')
plt.ylabel('Position Error (m)')
plt.legend()
```

```
plt.grid(True)
plt.show()
```

```
2401
2401
```

Figure