Adaptive Trajectory Control for a Robotic Arm Subject to Varying Load

Aishwary Jagetia, Animesh Nema, Nishant Shah

Abstract— **The focus of this paper is to implement two methods for an adaptive trajectory tracking control on a robotic arm manipulator such that the robot arm can maintain its desired trajectory while carrying objects of different masses. While the first technique relies on linear parameterization, the second method works on function approximation techniques. In both methods, the control law has to be updated in a way that it adjusts the gain such that the output does not change with changing parameters of the dynamic model (mass of the object,in this instance). Subsequently, the robot arm can be made to follow a given trajectory while also accommodating for the change in mass of the object it is carrying.**

I. INTRODUCTION

Choosing an appropriate control method is one of the most crucial tasks while performing any kind of manipulation. While a few tasks are subjected to dealing with fixed set of parameters and a fixed dynamic model, this may not always be the case. In fact most real life applications concerning robot manipulation such as UAV's, spot welding, spray painting, pick and place applications and tasks related to humanoid robots etc have parameters that are either not known or change unpredictably with time. The former set of tasks can be accomplished with a simple control scheme such as feed forward control or robust control. The latter tasks however require a slightly advanced control scheme that can accommodate for the unpredictable changes in the parameters of the system.

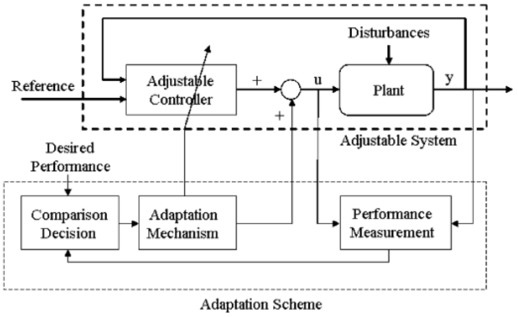


Fig. 1. Basic Configuration for an adaptive Control System

Hence, a method called Adaptive control can be used in which the controller is capable of automatically adjusting to the change in the parameters under a certain range in order to maintain the system performance to a desired level. Without such a controller, the system's performance may drastically reduce and may even become unsatisfactory depending on the degree of changes. An adaptive control system measures a certain performance index (IP) of the control system using the inputs, the states, the outputs and the known disturbances. From the comparison of the measured performance index and a set of given ones, the adaptation mechanism modifies the parameters of the adjustable controller and/or generates an auxiliary control in order to maintain the performance index of the control system close to the set of given ones.

With advancements in the field of robotics, concepts such as Unmanned aerial vehicles, human-robot interaction, shared autonomy etc are becoming popular. In all these applications there is a high chance of encountering unforeseen changes in the environment of the robot. Also, in many applications it is quite natural for the robot to be subject to varying loads or change in mass of the robot itself e.g., a pick and place robot carrying objects of different masses or a change of mass of airplane due to fuel consumption during it's flight. In such situations it is not feasible to design a simple feed forward system. In order to achieve the desired target, and to increase the reliability of automation, we are motivated to opt for an adaptive controller that can account for such changes and reduce the need for human intervention while ensuring a more efficient functioning.

II. LITERATURE REVIEW

In [1], the 2-DOF dynamic equation was derived and solved and 2-segment compound pendulum model is considered which mimics as a human upper arm and forearm. Modeling humanlike properties is achieved using two types of controller, first is PD controller and combining it with a second controller known as feed forward controller. A particular control strategy that humans exert when carrying out a task is known as the Feedback Error Learning (FEL) strategy. This can be achieved by expanding the dynamic equations with a set of controllers and introducing a task into the algorithm to test them. PD feedback controllers performance is enhanced by adding a feedforward controller in which the controller gradually learns the necessary torque that is required to perform a task for any repetitive task. This paper shows a computational model of human motor control based on Feedback Error Learning (FEL) and an adaptive algorithm for a nonlinear 2-DOF arm model which was successfully developed, simulated and tested in MATLAB software. Advantage of adaptive control method is that it manually adjust and updates the controller gain parameters and the algorithm used has the ability to learn and adapt its trajectory in order to achieve the desired task objective.

In [2] they have discussed how the dynamic model and the robots properties varies with the time-varying payloads, which is different compared to the robots dynamic model with constant payload. They have developed a passivity based adaptive control law for a two-link robotic arm and tested it on the experimental platform which mimics pouring or filling operations from a cylindrical vessel. Advantages of using adaptive control law over other controller is that we can separate the time-varying load parameters from the Inertia matrix and make the dynamic model linear in the unknown parameters, also account for significant changes in the dynamic model. They have introduced the Time-Scaling to account for the magnification of the adaptive law running for a long time due to small error. Based on this time-scaling factor they have modified their control law which did not affect the stability. They have concluded that the proposed adaptive controller with time scaling resulted better while comparing the results of passivity-based controller without adaptation, proposed adaptive controller without and proposed adaptive controller with time scaling.

The paper [3] presented a new adaptive control scheme for a 2 degree of freedom robot arm carrying uncertain time varying payload based on Function Approximation technique(FAT).The advantage of this method was that the authors didn't have to perform linear parametrization. The dynamic model of the arm was a summation of all the known and uncertain terms. Since the mass is attached to the second link, its mass is regarded as mass of link 2. The authors tested their approach using MATLAB simulink and the results proved that their control scheme was successful. The model could account for mass that varied with time in all the cases. The interesting aspect was that only one control gain was required to adjust the model.

In [4], they talk about implementing an adaptive controller on a rigid link manipulator holding objects of different masses. But the unique thing in [4] is that they have devised an control algorithm which consists of PD feedback part and a full dynamics feedforward compensation part, with the unknown manipulator and payload parameters being estimated online. Also, they did not need to measure the joint acceleration or to invert the estimated inertia matrix as mentioned as a prerequisite they read in the previous papers. In this paper the H(inertia matrix) and C(Coriolis matrix) are not independent and another thing they mentioned is that the dynamic structure is linear in terms of suitably selected set of robot and load parameters. To derive the control algorithm and adaptation law, they considered Lyapunov function candidate. In this paper they considered the unknown mass as a part of the last link as an augmented link, with four unknown parameters, namely mass, moment of inertia, the distance of its mass center to the last joint, and the angle relative to the original last link.

Paper[5] focuses on adaptive neural network control with full-state feedback proposed for an uncertain robot with constraints, which can guarantee the performance and improve the robustness of closed-loop system effectively. In this paper, they propose the MoorePenrose inverse term and design the adaptive neural network control to address the tracking problem of a Multiple Input Multiple Output (MIMO) robotic system. They simulated their approach on a robot with two revolute joints in the vertical plane, during which they compared the proposed adaptive NN control approach with the PD control and from the comparison they observed that the errors of PD control were larger than the proposed adaptive NN control when the parameters of the system are unknown, because of the NN learning ability.

III. PROBLEM STATEMENT

The challenge of implementing such an adaptive controller is designing a suitable control law that accommodates for the change in the mass of the object while also satisfying the requirements of being a Lyapunov function so that its stability can be established.

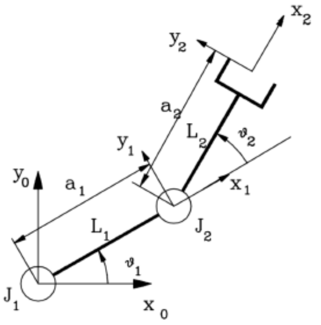


Fig. 2. Adaptive Control of a one link arm

The physical constraints of the robotic arm such as its load carrying capacity, its working range and singularities have to be taken into consideration while designing its trajectory and appropriately choosing the objects as we would not want the robot arm to get stuck due to singularities during the task or not be able to perform the task due to hardware constraints such as its load carrying capacity.

IV. METHODS

We have implemented adaptive control on a two link arm using 2 different approaches.

1) Adaptive control of a Robot Carrying a Time varying Payload using Linear Parameterization.
2) Adaptive control on time varying load using Function Approximation techniques.

Both the approaches have been tested at different payloads and the results of the analysis have been plotted.At the end of the paper, a comparison between both the approaches has been presented.

V. IMPLEMENTATION

A. Adaptive control of a Robot Carrying a Time varying Payload using Linear Parameterization

We have implemented the method described in [2]. This method is based on the linear parametrization. The advantages of this method is to have a time varying mass attached to end effector which can be considered as an individual mass attached separately, along with its geometric configuration. They have used mass = K * time (K is a parameter) to approximated the mass variation, which they have also

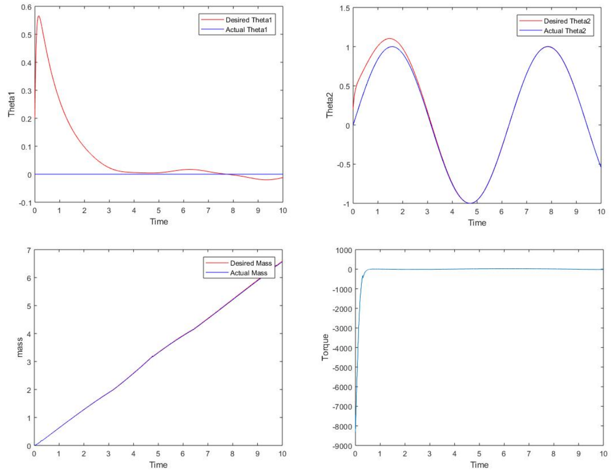


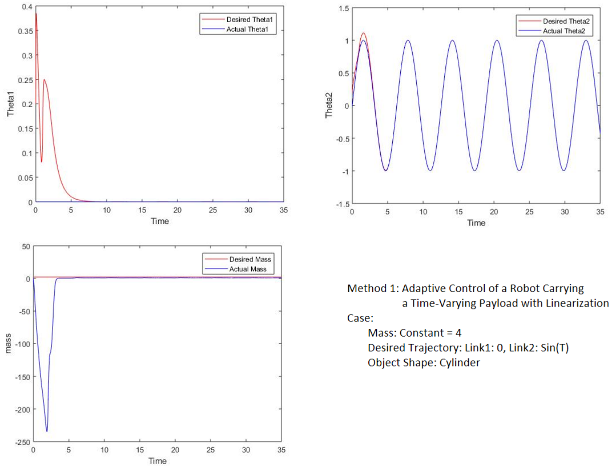
Fig. 3. Reproducing the results discussed in paper

incorporated in the dynamic model. We have reproduced the same results described in the paper as shown in figure 3.

We have further experimented with different time varying mass functions which is to be approximated by the same mass function stated above as shown in figure 4,5,6,7,8,9 and 10. We found that the system converges to a stability but along with small errors. Also, if we observe during the system convergence there is a high amplitude peak at the initial stage.

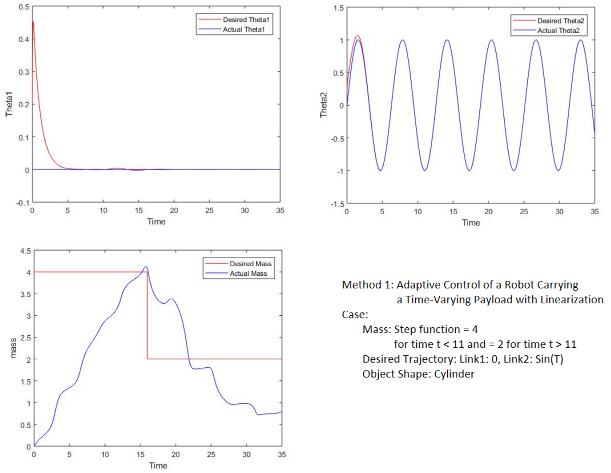
The following graphs were plotted.

- The actual and desired angle of link 1.
- The actual and desired angle of link 2.
- The difference between actual and estimated mass.



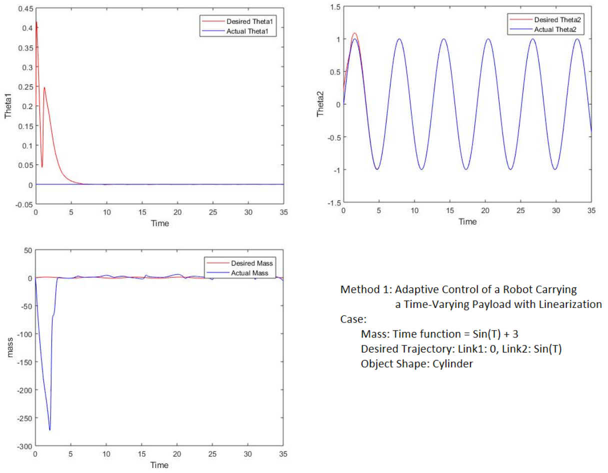
Method 1: Adaptive Control of a Robot Carrying
a Time-Varying Payload with Linearization
Case:
Mass: Constant = 4
Desired Trajectory: Link1: 0, Link2: Sin(T)
Object Shape: Cylinder

Fig. 4. Experiment 1



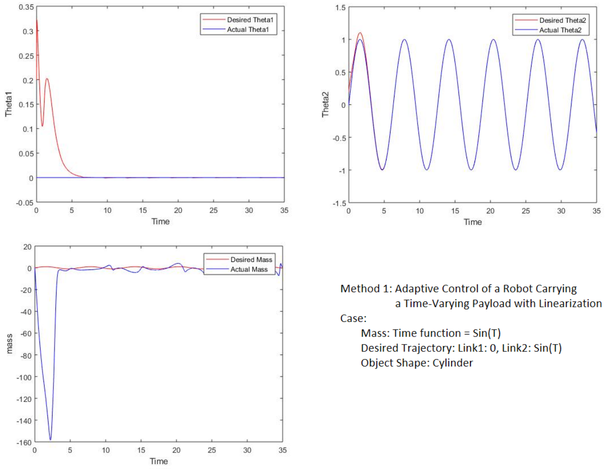
Method 1: Adaptive Control of a Robot Carrying
a Time-Varying Payload with Linearization
Case:
Mass: Step function = 4
for time t < 11 and = 2 for time t > 11
Desired Trajectory: Link1: 0, Link2: Sin(T)
Object Shape: Cylinder

Fig. 5. Experiment 2



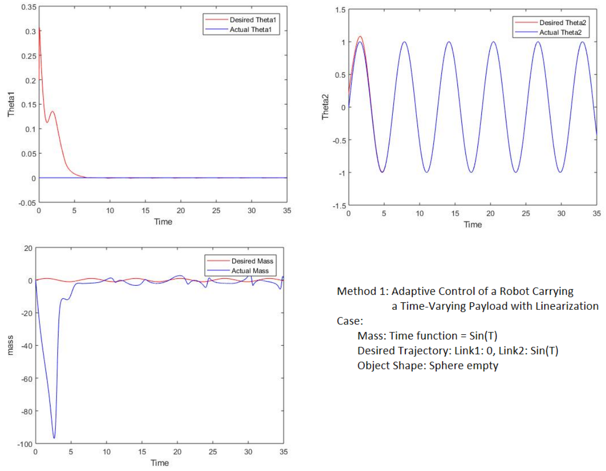
Method 1: Adaptive Control of a Robot Carrying
a Time-Varying Payload with Linearization
Case:
Mass: Time function = Sin(T) + 3
Desired Trajectory: Link1: 0, Link2: Sin(T)
Object Shape: Cylinder

Fig. 6. Experiment 3



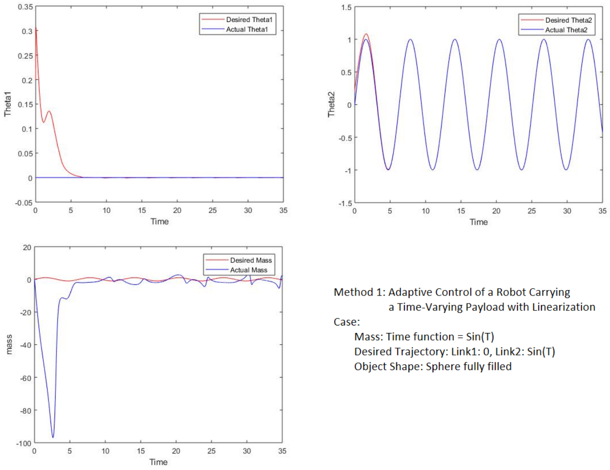
Method 1: Adaptive Control of a Robot Carrying
a Time-Varying Payload with Linearization
Case:
Mass: Time function = Sin(T)
Desired Trajectory: Link1: 0, Link2: Sin(T)
Object Shape: Cylinder

Fig. 7. Experiment 4



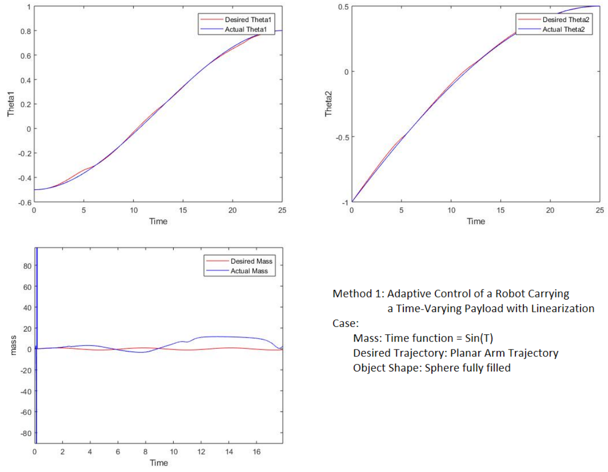
Method 1: Adaptive Control of a Robot Carrying
a Time-Varying Payload with Linearization
Case:
Mass: Time function = Sin(T)
Desired Trajectory: Link1: 0, Link2: Sin(T)
Object Shape: Sphere empty

Fig. 8. Experiment 5



Method 1: Adaptive Control of a Robot Carrying
a Time-Varying Payload with Linearization
Case:
Mass: Time function = Sin(T)
Desired Trajectory: Link1: 0, Link2: Sin(T)
Object Shape: Sphere fully filled

Fig. 9. Experiment 6



Method 1: Adaptive Control of a Robot Carrying
a Time-Varying Payload with Linearization
Case:
Mass: Time function = Sin(T)
Desired Trajectory: Planar Arm Trajectory
Object Shape: Sphere fully filled

Fig. 10. Experiment 7

B. Adaptive control on time varying load using Function Approximation techniques

The highlight of this method is that the need of linear parameterization is eliminated. The method rather makes use of Function approximation techniques(FAT) such as fourier series, taylor series expansion etc, to approximate the unknown terms (mass of the object in this case).

In this method, since the mass is on the second link, we assume that the total mass of link 2 is the sum of the masses of link 2 and the mass of object. In other words, the object is considered to be a mass of link 2 itself.

The dynamic model, i.e., the Inertia Matrix(M) , the force matrix(C) and the gravity matrix(G) are written such that the unknown mass (m2) is factorized out of the matrices. The estimated model comprises of the estimated M,C and G matrices and also the estimated mass which is factorized out of the matrices.
The uncertain mass can be written as a FAT expression such as a fourier series expansion. This approximates the mass in terms of sine and cosine functions, which makes it easier to work on a differential equation. The control law is a normal adaptive law control law that utilizes the estimated models and reference errors.

It can be noted that in a fourier series, each term is multiplied to a certain constant value. Hence, on updating these constants, the fourier series expression approximates the original value.
The update law is chosen such that the matrix of the constants(W) are updated at each time step. The constant matrix W is updated by the product of a positive definite matrix and the estimated model multiplied by the error term. The error term is in the form of lambda, where lambda is a positive matrix. The function is also proved to be stable in terms of Lyapunov.

The approach was tested under the following conditions:-

- Mass2 = sint(t) + 3
- Mass2 = 2
- Mass2 = 4 at t <7 and 2 at t >7

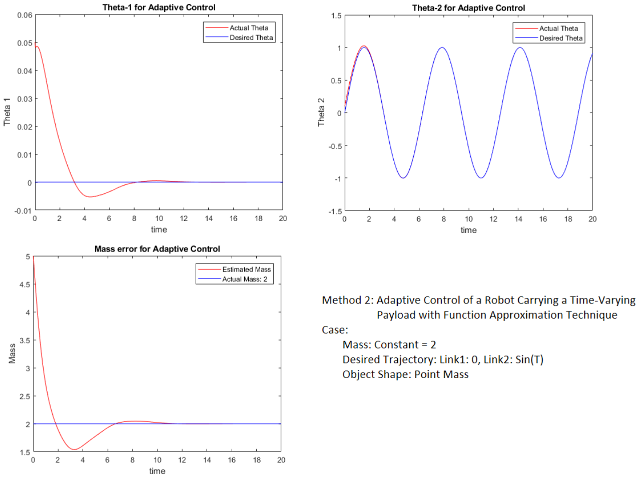
The following graphs were plotted.

- The actual and desired angle of link 1.
- The actual and desired angle of link 2.
- The difference between actual and estimated mass.

Advantages:

- The computation cost for the method is reduced as compared to the conventional process that uses linear parametrization.
- Only one control gain is required to tune the process.
- The nature of dynamic modeling is such that the system is simpler to implement and tune.

From the results obtained, it can be seen that the estimated mass converged fairly well with the true mass. Graphs for the results can be seen in figures 11, 12 and 13 respectively.

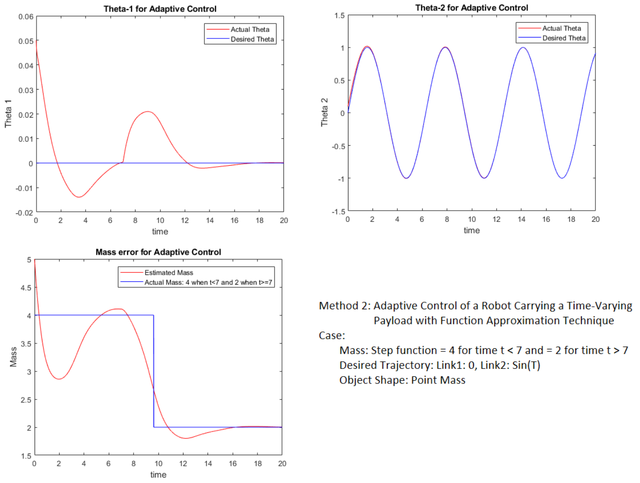


Method 2: Adaptive Control of a Robot Carrying a Time-Varying Payload with Function Approximation Technique

Case:

Mass: Constant = 2
Desired Trajectory: Link1: 0, Link2: Sin(T)
Object Shape: Point Mass

Fig. 11. Experiment 8

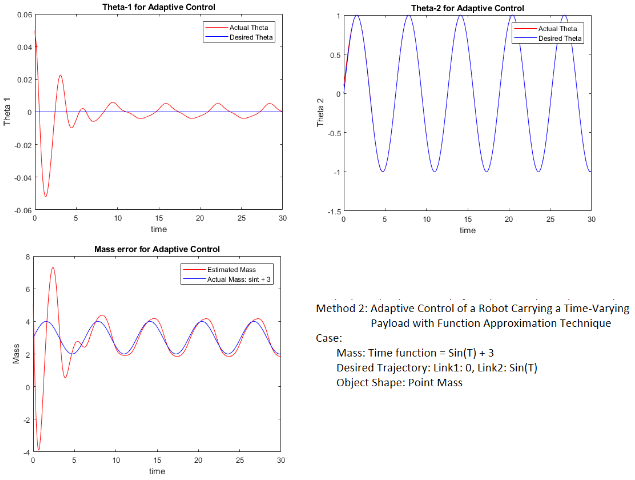


Method 2: Adaptive Control of a Robot Carrying a Time-Varying Payload with Function Approximation Technique

Case:

Mass: Step function = 4 for time t < 7 and = 2 for time t > 7
Desired Trajectory: Link1: 0, Link2: Sin(T)
Object Shape: Point Mass

Fig. 12. Experiment 9



Method 2: Adaptive Control of a Robot Carrying a Time-Varying Payload with Function Approximation Technique

Case:

Mass: Time function = Sin(T) + 3
Desired Trajectory: Link1: 0, Link2: Sin(T)
Object Shape: Point Mass

Fig. 13. Experiment 10

VI. Simulation Setup

To visualize the adaptive control technique implemented above, two link arm model was built from scratch in VREP and the environment was set up to simulate the two link arm model. Adaptive control using Function approximation technique was implemented on the model. Vrep remote api was used to establish a communication network between matlab and vrep. In vrep, every joint is treated as a handle and commands has to be sent on those handles whether position or velocity commands. Vrep does not support force control of a robot/model and thus a model can be controlled on a position or velocity level only. The dynamic model properties were defined in the Vrep environment itself and all those parameters like mass of the link, length of the link were then used in the control law we defined. The mass of the second link was updated at every time step in the control law defined for the two link arm model.

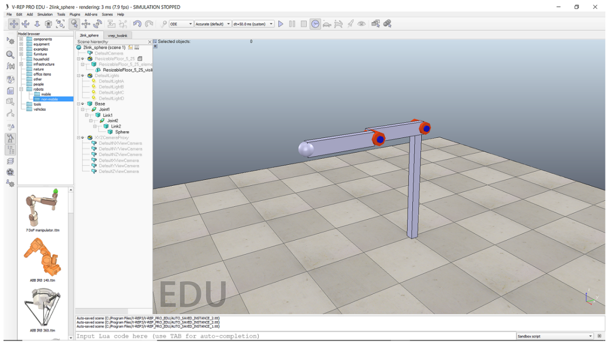
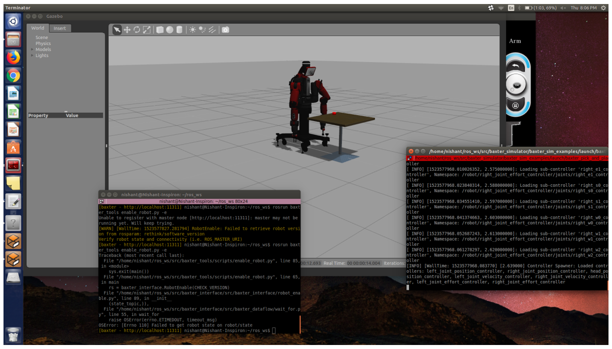


Fig. 14. Two link arm in VREP Simulator

We faced many difficulties while simulating the model because of the synchronous and asynchronous execution behavior in Vrep. Synchronization issues were faced because of the different time stamp in matlab and Vrep and hence When we used to send command from matlab it used to skip some of the commands and the model behaved inappropriately.

Our dream goal was to implement the same controller on Baxter simulated in Gazebo. As it is clear from the method discussed above, that the adaptive controller requires dynamic model of the robot so as to update the parameters in the dynamic model to compensate for the change in mass. We referred a paper[9] related to Baxter dynamic model and tried to derive the dynamic model of the robot to use it for our simulation. We were successful in deriving the Inertia and Gravity matrix but the formula to derive the Coriolis matrix was difficult to interpret and also because of the shortage of time, we were not able to derive the dynamic model completely. But we tried to play with the Gazebo model of the Baxter and tried tracik library as an Inverse Kinematics solver which shows that it solves the IK of Baxter 98 percent of the time accurately. We also tried publishing the joint positions on the joint topic and tried to move the arm explicitly. We also played around with Baxter interface package [7] to interface the python executable with the Gazebo model without the need of publishing any message on any topic which is handled by the Baxter interface package itself.



```
nishant@Nishant-Inspiron: ~/ros_ws
nishant@Nishant-Inspiron: ~/ros_ws 80x24
[baxter - http://localhost:11311] nishant@Nishant-Inspiron:~/ros_ws$ rosrun baxt
er_tools enable_robot.py -e
Unable to register with master node [http://localhost:11311]: master may not be
running yet. Will keep trying.
[WARN] [WallTime: 1523577827.281794] RobotEnable: Failed to retrieve robot versi
on from rosparam: rethink/software_version
Verify robot state and connectivity (i.e. ROS_MASTER_URI)
[baxter - http://localhost:11311] nishant@Nishant-Inspiron:~/ros_ws$ rosrun baxt
er_tools enable_robot.py -e
Traceback (most recent call last):
  File "/home/nishant/ros_ws/src/baxter_tools/scripts/enable_robot.py", line 85,
 in <module>
    sys.exit(main())
  File "/home/nishant/ros_ws/src/baxter_tools/scripts/enable_robot.py", line 65,
 in main
    rs = baxter_interface.RobotEnable(CHECK_VERSION)
  File "/home/nishant/ros_ws/src/baxter_interface/src/baxter_interface/robot_ena
ble.py", line 89, in __init__
    (state_topic,)),
  File "/home/nishant/ros_ws/src/baxter_interface/src/baxter_dataflow/wait_for.p
y", line 55, in wait_for
    raise OSError(errno.ETIMEDOUT, timeout_msg)
OSError: [Errno 110] Failed to get robot state on robot/state
[baxter - http://localhost:11311] nishant@Nishant-Inspiron:~/ros_ws$
```

```
/home/nishant/ros_ws/src/baxter_simulator/baxter_sim_examples/launch/bax
/home/nishant/ros_ws/src/baxter_simulator/baxter_sim_examples/launch/baxter_pick_and_pla
oller
[ INFO] [1523577968.010026352, 2.575000000]: Loading sub-controller 'right_e1_co
ntroller', Namespace: /robot/right_joint_effort_controller/joints/right_e1_contr
oller
[ INFO] [1523577968.023840314, 2.588000000]: Loading sub-controller 'right_s0_co
ntroller', Namespace: /robot/right_joint_effort_controller/joints/right_s0_contr
oller
[ INFO] [1523577968.034551410, 2.597000000]: Loading sub-controller 'right_s1_co
ntroller', Namespace: /robot/right_joint_effort_controller/joints/right_s1_contr
oller
[ INFO] [1523577968.041374663, 2.603000000]: Loading sub-controller 'right_w0_co
ntroller', Namespace: /robot/right_joint_effort_controller/joints/right_w0_contr
oller
[ INFO] [1523577968.052687243, 2.613000000]: Loading sub-controller 'right_w1_co
ntroller', Namespace: /robot/right_joint_effort_controller/joints/right_w1_contr
oller
[ INFO] [1523577968.061278297, 2.620000000]: Loading sub-controller 'right_w2_co
ntroller', Namespace: /robot/right_joint_effort_controller/joints/right_w2_contr
oller
[INFO] [WallTime: 1523577968.083770] [2.639000] Controller Spawner: Loaded contr
ollers: left_joint_position_controller, right_joint_position_controller, head_po
sition_controller, left_joint_velocity_controller, right_joint_velocity_controll
er, left_joint_effort_controller, right_joint_effort_controller
```

Fig. 15. Baxter in Gazebo Simulator

VII. Comparing the Results

After executing the passivity base adaptive control for the two-link arm with time scaling we have observed that the method 2 worked better when compared to method 1. Method 2 also takes care of high amplitude peaks generated at the initial stage whereas method 1 generates high initial errors.

VIII. CONCLUSION

We have successfully implemented an passivity based adaptive trajectory tracking control algorithm on a 2-link robot arm subject to varying loads and to be able to view the performance of the system for different techniques.

As an extension to our project, we would like to be able to implement the same on Baxter arm, which is a 7-DOF manipulator arm with reference to [8] and [7] and test it on the actual hardware. The Baxter is available at the CIBR lab at WPI. Also, we would try to implement Dynamic Movement Primitives on the controller so that the robotic arm can learn over time, to efficiently maintain its trajectory and implement Neural Network (a machine learning technique) to learn the dependencies of trajectories on a particular load over repeated iterations, and these stored dependencies can be later used while generating trajectory for a similar load.

References

[1] Tomoki Saiio, Masaki Okada, Koichiro Shida, Hideji Fujikawa and Shin-ichi Yamada, Adaptive Controller algorithm for 2-DOF humanoid robot arm. Department of E and E Eng, Musashi Institute of Technology. Tamazutsumi 1-28-1,Setagaya-ku,Tokyo 158, Japan.

[2] Pagilla, P.R. and Yu, B., 2000. Adaptive control of a robot carrying a time-varying payload. In Control Applications, 2000. Proceedings of the 2000 IEEE International Conference on (pp. 68-73). IEEE.

[3] Azlan, Norsinnira Zainul, and Syarifah Nurul Syuhada Sayed Jaafar. ”Adaptive control of robotic arm carrying uncertain time-varying payload based on function approximation technique.” International Conference on Social Robotics. Springer, Cham, 2014.

[4] Slotine JJ, Li W. On the adaptive control of robot manipulators. The international journal of robotics research. 1987 Sep;6(3):49-59.

[5] He, Wei, Yuhao Chen, and Zhao Yin. ”Adaptive neural network control of an uncertain robot with full-state constraints.” IEEE transactions on cybernetics 46.3 (2016): 620-629.

[6] Yang, Chenguang, Hongbin Ma, and Mengyin Fu. ”Robot kinematics and dynamics modeling.” Advanced Technologies in Modern Robotic Applications. Springer, Singapore, 2016. 27-48.

[7] https://github.com/RethinkRobotics

[8] Smith, A., Yang, C., Li, C., Ma, H. and Zhao, L., 2016, August. Development of a dynamics model for the baxter robot. In Mechatronics and Automation (ICMA), 2016 IEEE International Conference on (pp. 1244-1249). IEEE.

[9] Chenguang Yang, Hongbin Ma, Mengyin Fu, Robot Kinematics and Dynamics Modeling