Fundamentals of Robot Control

Three Link Manipulator (FANUC R-2000iC-165F)

Ilia Sevostianov

Innopolis University

December 5, 2019

Contents

Abstract

This project is dedicated to control of the positioning part of FANUC R-2000iC-165F. I needed to implement various control techniques on certain and "uncertain" system, analyze and compare results.

Project goals

- Obtain Dynamics equations of my system
- Implement PD control
- Implement Feedback Linearization control
- Implement Robust control
- Implement those 3 control techniques when there are some uncertainties
- Analyze results and compare them
  This also should be done on point-to-point movement and trajectory movement.

Robot Model

You can see my robot in the figure 1. The configuration of my robot are:

- Lengths:
  $L_{11} = 0.324m, L_{12} = 0.312m, L_2 = 1.075m, L_{31} = 0.225m$
- Radius: $c = 0.20m$

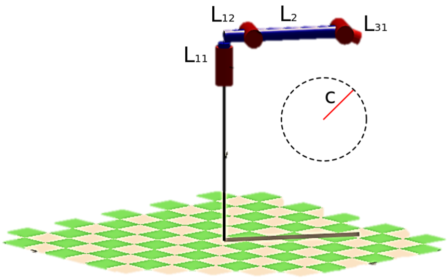

Figure: Robot Model

Dynamics

By defining the potential **P** and kinetic energies **K** of the system, and the Lagrangian **L** as $L = K - P$, the Eulear-Lagrange equation 1 below can be used to determine the dynamics model of the robot:

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}} - \frac{\partial L}{\partial q} = \tau \tag{1}$$

But for control purposes it's more practical to use matrix form as follows:

$$\boldsymbol{D}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{G}(\boldsymbol{q}) = \boldsymbol{\tau} \tag{2}$$

where

- C -Coriolis and Centrifugal term;
- D - Inertia matrix;
- G - Gravity term
- τ - torques(control) (**u** further)

So, for implementation I modified equation 2 to:

$$\boldsymbol{D}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{\beta}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{u} \tag{3}$$

where

$$\boldsymbol{\beta}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{G}(\boldsymbol{q}) \tag{4}$$

So, let's start with control!

The robot without control

Let's at first see how my robot behaves without control:

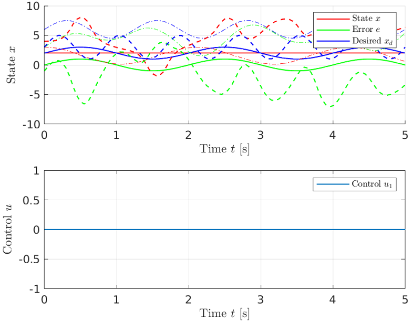

Figure: Not Controllable (Trajectory movement)

- non-stable
- big error

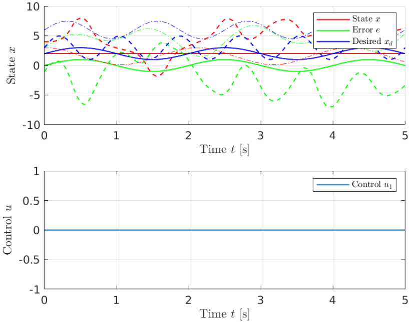

Figure: Not Controllable (PTP movement)

- non-stable
- big error

PD Control

The controller of the general form

$$\boldsymbol{u} = \boldsymbol{K_P}\boldsymbol{e} + \boldsymbol{K_D}\dot{\boldsymbol{e}} \tag{5}$$

It's called **proportional-derivative (PD)** controller.
It aims at:

- pulling the system towards the desired point
- gradually removing energy from the system

But its performance suffers significantly in the presence of nonlinearities.

PTP Movement

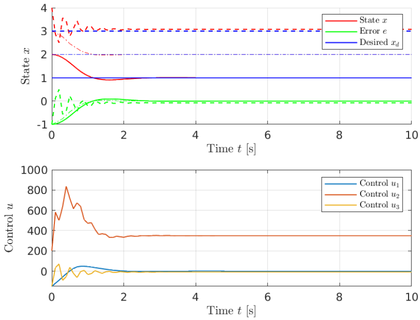

Figure: PD Control (PTP movement)

- Stable
- non "zero" error

PTP Movement with Uncertainties

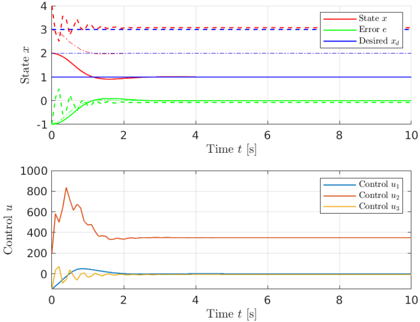

Figure: PD Control (PTP movement)

- Stable
- non "zero" error

Trajectory Movement

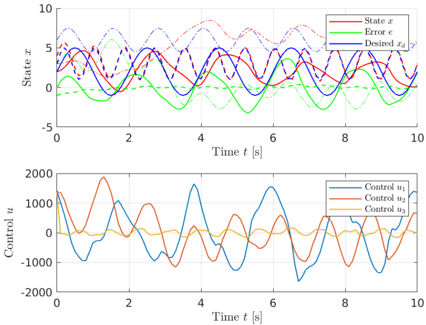

Figure: PD Control (Trajectory movement)

- Non-Stable
- Difficult to tune PD Controller

Trajectory Movement with Uncertainties

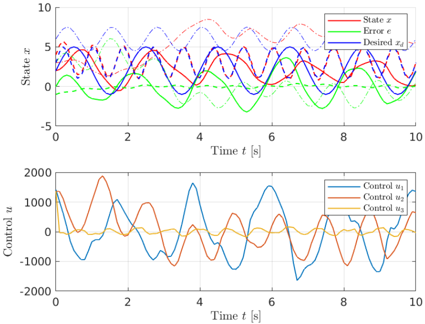

Figure: PD Control (Trajectory movement)

- Non-Stable
- PD-Controller is not capable for controlling it...

Feedback Linearization Control

I incorporate known system dynamics in PD controller in following form:

$$\boldsymbol{u} = \boldsymbol{D}(\boldsymbol{q})(\ddot{\boldsymbol{x}}_{\boldsymbol{d}} + \boldsymbol{K_P}\boldsymbol{e} + \boldsymbol{K_D}\dot{\boldsymbol{e}}) + \beta(\boldsymbol{q}, \dot{\boldsymbol{q}}) \tag{6}$$

PTP Movement

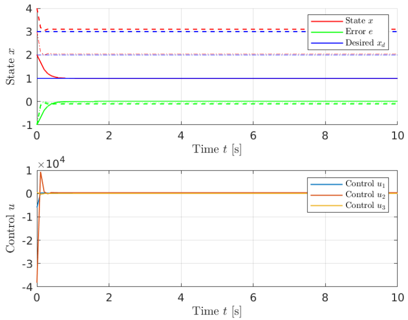

Figure: FBL Control (PTP movement)

- Stable
- High PD coefs

PTP Movement with Uncertainties

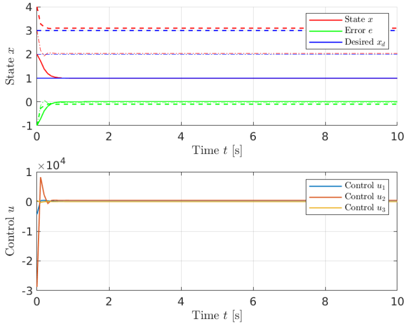

Figure: FBL Control (PTP movement)

- Stable, but still there is the error. It's needed to increase FBL coeffs, may be

Trajectory Movement

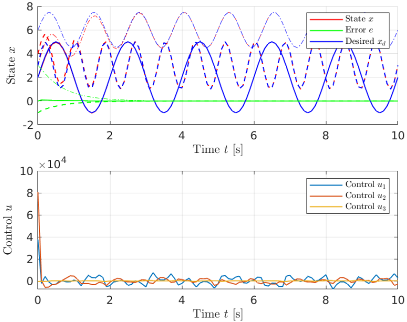

Figure: FBL Control (Trajectory movement)

- Stable
- It's very easy to tune FBL Controller

Trajectory Movement with Uncertainties

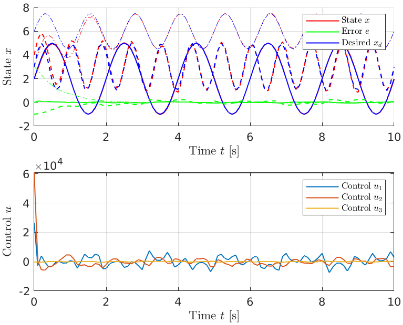

Figure: FBL Control (Trajectory movement)

- Stable
- It's very easy to tune FBL Controller
- It is capable for uncertainties

Robust Control

Let me define the sliding surface for n = 2 and the state $\boldsymbol{x} = [\boldsymbol{q}, \dot{\boldsymbol{q}}]^T$:

$$s(\boldsymbol{x}; t) = (\frac{d}{dt} + \lambda)\tilde{e} = \dot{\tilde{e}} + \lambda\tilde{e} \tag{7}$$

Now, I can choose the control law as

$$\boldsymbol{u}^\star = \boldsymbol{D}(\boldsymbol{q})(\ddot{\boldsymbol{x_d}} + \boldsymbol{K_P}\boldsymbol{e} + \boldsymbol{K_D}\dot{\boldsymbol{e}}) + \boldsymbol{w} \tag{8}$$

$$\boldsymbol{u} = \boldsymbol{D}(\boldsymbol{q})\boldsymbol{u}^\star + \beta(\boldsymbol{q}, \dot{\boldsymbol{q}}) \tag{9}$$

$$\boldsymbol{w} = \begin{cases} 0 & ||s|| = 0 \\ \frac{\rho s}{||s||} & ||s|| \neq 0 \end{cases} \tag{10}$$

PTP Movement

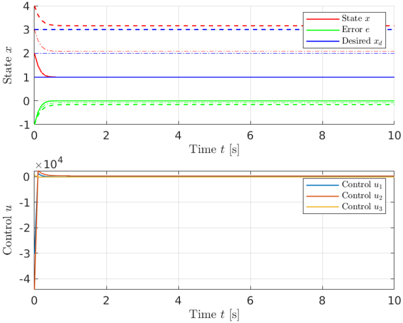

Figure: R Control (PTP movement)

- Stable, but still the error. It's needed to increase the P coeff
- Very high PD coeffs

PTP Movement with Uncertainties

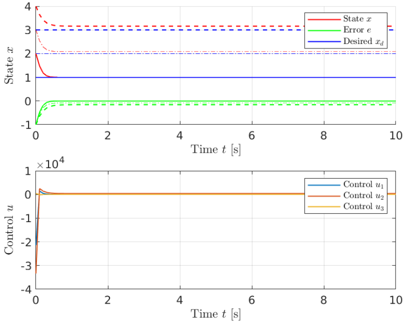

Figure: R Control (Trajectory movement)

- Stable
- Easy to tune coeffs
- Very robust technique

Trajectory Movement

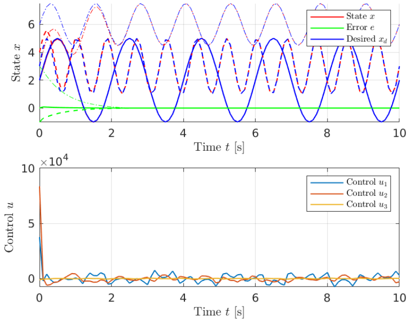

Figure: R Control (Trajectory movement)

- Stable
- Very low error

Trajectory Movement with Uncertainties

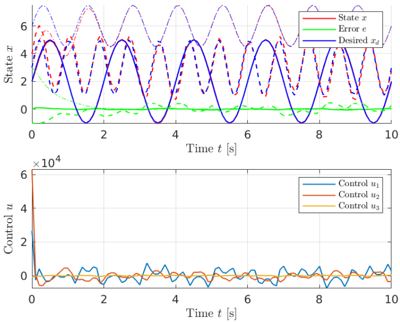

Figure: R Control (Trajectory movement)

- Stable
- Works good even with uncertainties
- Easy and understandable to tune coeffs

Conclusion

- PD controller gave the worst results. It's not applyable for the systems with nonlinearities. And it's also complicated to follow some trajectory law and tune PD coefficients.
- FBL Control worked enough accurate and good for my system. Also, for me, it wasn't difficult to tune PD coefficients to get accurate results, but The control amplitude is high.
- Robust Control worked for me as good as FBL Control but for the system with uncertainties it gave less accurate results. I think, it's because of non so properly parameters tuning.

Link to the Code