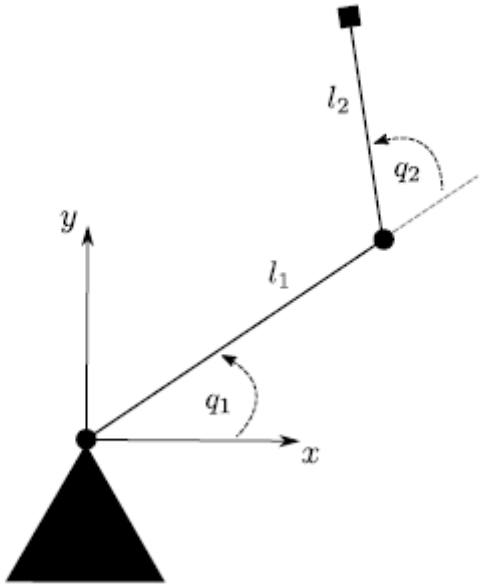
Nonlinear Model Predictive Control

Project: Rest-to-Rest Control of a 2-DoF Robot



Name: Xin Ye

Matric No: 2136390

Major: Mechatronics and Information Technology (Master)

Karlsruhe Institute of Technology

November 2018

Table of Contents

1. Background and Modeling

An often occurring control task in industry is to compute optimal input trajectories for robots to reach a given end position while consuming minimal energy or to reach the end position in minimal time. In this project we want to tackle this challenges by controlling a Two degree of Freedom (2-DoF) robot shown in Figure 1.

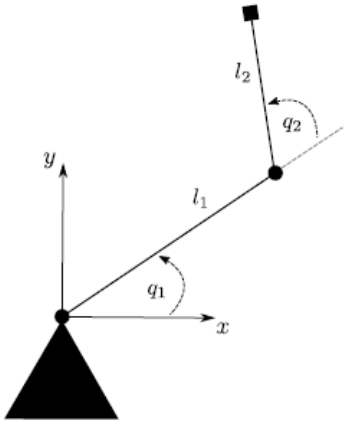


Figure 1: 2-DoF robot

a) Physical Model

We choose q_1 and q_2 as general coordinates because these two describe the states of the 2-DoF Robot completely. But if we choose x-y coordinates, we have 4 general coordinates, and they don't represent the two degrees of freedom directly, and we have to apply equations of mechanical constraints. In this case, many square root calculations come to present, and the sign need to be considered carefully, which enlarge the computational burden.

The system consists of two rigid bodies with mass m_1, m_2, moments of inertial to the z-axis J_1, J_2, and length l_1, l_2. When the robot is put into industrial applications, it is often loaded at the end effector, and we simplify it as a mass point m_3 at the end of arms. We denote the centers of gravity of arms and the end effector as P, Q and R respectively. We apply the principle of Jourdain to describe the kinematics:

$$\sum_{i=1}^{\nu}\left\{\left[\frac{^N d}{dt}(m_i\,{}^N\vec{v}^{S_i}) - \vec{F}_i^e\right]\cdot\frac{\partial\,{}^N\vec{v}^{S_i}}{\partial u_j} + \left[\frac{^N d}{dt}\vec{H}_{S_i} - \vec{M}_{S_i}^e\right]\cdot\frac{\partial\,{}^N\vec{\omega}^{K_i}}{\partial u_j}\right\} = 0, \qquad j = 1,\dots,p \qquad (1)$$

In which we evaluate the velocities as well as derivatives of momentums at the centers of gravity ($^N\vec{v}^{S_i}$, $\frac{^Nd}{dt}(m_i\,^N\vec{v}^{S_i})$, where $S_i = P, Q, R$), we evaluate the angular velocities as well as derivatives of angular momentums w.r.t. centers of gravity of the two arms ($^N\vec{\omega}^{K_i}$, $\frac{^Nd}{dt}\vec{H}_{S_i}$, where $S_i = P, Q$ and $K_i = B, C$), and calculate the partial derivatives of the general velocities ($\frac{\partial\,^N\vec{v}^{S_i}}{\partial u_j}$, $\frac{\partial\,^N\vec{\omega}^{K_i}}{\partial u_j}$, where $S_i = P, Q, R$ and $K_i = B, C$ and $u_j = \dot{q}_1, \dot{q}_2$). As simplification, we only include gravity forces as $\vec{F}_i^e = -m_i g\vec{a}_y$ $(i = 1, 2, 3)$ and input torques as $\vec{M}_1^e = u_1 - u_2$, $\vec{M}_2^e = u_2$ but ignore others such as frictions.

With help of Matlab symbolic toolbox, we are able to obtain the differential equations:

```
u1 =
(- (l1*l2*m2*s2)/2 - l1*l2*m3*s2)*q2_dot^2 - q1_dot*(l1*l2*m2*s2 + 2*l1*l2*m3*s2)*q2_dot
+ q2_ddot*(J2 + (l2^2*m2)/4 + l2^2*m3 + (c2*l1*l2*m2)/2 + c2*l1*l2*m3) + q1_ddot*(J1 + J2
+ (l1^2*m1)/4 + l1^2*m2 + l1^2*m3 + (l2^2*m2)/4 + l2^2*m3 + c2*l1*l2*m2 + 2*c2*l1*l2*m3)
+ (c1*g*l1*m1)/2 + c1*g*l1*m2 + c1*g*l1*m3 + (g*l2*m2*cos(q1(t) + q2(t)))/2 +
g*l2*m3*cos(q1(t) + q2(t))

u2 =
((l1*l2*m2*s2)/2 + l1*l2*m3*s2)*q1_dot^2 + q2_ddot*(J2 + (l2^2*m2)/4 + l2^2*m3) +
q1_ddot*(J2 + (l2^2*m2)/4 + l2^2*m3 + (c2*l1*l2*m2)/2 + c2*l1*l2*m3) + (g*l2*m2*cos(q1(t)
+ q2(t)))/2 + g*l2*m3*cos(q1(t) + q2(t))
```

We organize them in matrices:

$$B(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = u \tag{2}$$

where

$$B(q) = \begin{pmatrix} b_1 + b_2\cos q_2 & b_3 + \frac{1}{2}b_2\cos q_2 \\ b_3 + \frac{1}{2}b_2\cos q_2 & b_3 \end{pmatrix}$$

$$C(q,\dot{q}) = -c_1\sin q_2 \begin{pmatrix} 0 & 2\dot{q}_1 + \dot{q}_2 \\ -\dot{q}_1 & 0 \end{pmatrix}$$

$$g(q) = (g_1\cos q_1 + g_2\cos(q_1 + q_2)\,, \quad g_2\cos(q_1 + q_2))^T$$

$$b_1 = J_1 + J_2 + \frac{1}{4}l_1^2 m_1 + l_1^2 m_2 + l_1^2 m_3 + \frac{1}{4}l_2^2 m_2 + l_2^2 m_3$$

$$b_2 = l_1 l_2 m_2 + 2 l_1 l_2 m_3$$

$$b_3 = J_2 + \frac{1}{4}l_2^2 m_2 + l_2^2 m_3$$

$$c_1 = \frac{1}{2}b_2 = \frac{1}{2}l_1 l_2 m_2 + l_1 l_2 m_3$$

$$g_1 = \frac{1}{2} l_1 m_1 g + l_1 m_2 g + l_1 m_3 g$$

$$g_2 = \frac{1}{2} l_2 m_2 g + l_2 m_3 g$$

b) State Space Representation

Model as described by (2) can be reformulated in state space form.

$$x = [x_1\ x_2\ x_3\ x_4]^T = [q_1\ q_2\ \dot{q}_1\ \dot{q}_2]^T$$

$$u = [u_1\ u_2]^T$$

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} x_3 \\ x_4 \end{bmatrix}$$

$$\begin{bmatrix} \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = B^{-1}(-C\dot{q} - g + u)$$

2. Open-loop Optimal Control

a) Formulation of Fixed End-Time Problem

An optimal control problem can be designed as follows to drive the robot from the initial position to the end position in given time $t_f = 3s$ with input and state constraints. Moreover, the terminal state is given as constraints.

$$\min_{u(\cdot)} \int_0^{t_f} u^T(t) R u(t) dt \qquad (3)$$

subject to:

$$B(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = u$$

$$[q_1(0), q_2(0), \dot{q}_1(0), \dot{q}_2(0)]^T = [-5, -4, 0, 0]^T$$

$$[q_1(t_f), q_2(t_f), \dot{q}_1(t_f), \dot{q}_2(t_f)]^T = [\frac{\pi}{2}, 0, 0, 0]^T$$

$$u \in [-1000, 1000]$$

$$\dot{q} \in [-\frac{3\pi}{2}, \frac{3\pi}{2}]$$

Another design makes the terminal state error as a weighted part of objective function to minimize.

$$\min_{u(\cdot)} \int_0^{t_f} u^T(t) R u(t) dt + w\big(x(t_f) - x_f\big)^T P(x(t_f) - x_f) \qquad (4)$$

subject to:

$$B(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = u$$

$$[q_1(0), q_2(0), \dot{q}_1(0), \dot{q}_2(0)]^T = [-5, -4, 0, 0]^T$$

$$u \in [-1000, 1000]$$

$$\dot{q} \in [-\frac{3\pi}{2}, \frac{3\pi}{2}]$$

b) Numerical Solution of Fixed End-Time Problem

With an accurately given end state, the objective function is minimized to 6.5404e+6 when R=[10;10] N=30, and the angular velocity of the second joint reaches its maximum after the first joint reaches its maximum, as shown in Figure 2(a). It is natural to think that there is another possibility in which the angular velocity of the second joint reaches its maximum before the first joint, as shown in Figure 2(b). These two cases can be very different from each other, so it is reasonable to consider that when the solver get close to one of the cases, it may get stuck in the local optimization, but cannot find out that the other case is even more optimal. Therefore we add two additional constraints in order to control the solver on making the decision if the second joint accelerates earlier or later. When we want that the angular velocity of the second joint reaches its maximum after the first joint:

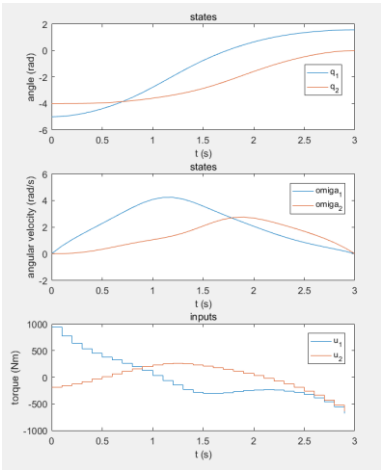
$$\dot{q}_1(t) \geq \dot{q}_2(t) \text{ for } t < \frac{1}{6} t_f$$

$$\dot{q}_1(t) \leq \dot{q}_2(t) \text{ for } t > \frac{5}{6} t_f$$

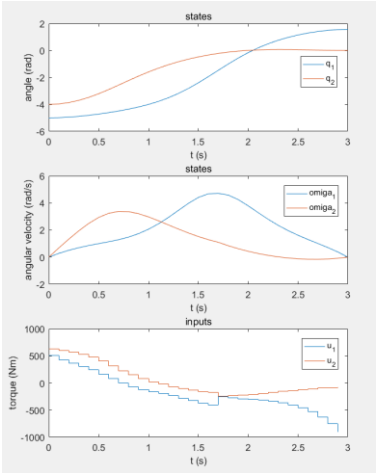
When we want that the angular velocity of the second joint reaches its maximum before the first joint:

$$\dot{q}_1(t) \leq \dot{q}_2(t) \text{ for } t < \frac{1}{6} t_f$$

$$\dot{q}_1(t) \geq \dot{q}_2(t) \text{ for } t > \frac{5}{6} t_f$$



(a)The angular velocity of the second joint reaches its maximum after the first joint reaches its maximum, cost 6.5404e+06



(b)The angular velocity of the second joint reaches its maximum before the first joint reaches its maximum, cost 6.8751e+06

Figure 2: Two different local optimal solutions

When we hold all other conditions identical, we get the cost of the second case at 6.8751e+06. The first case is repeated with the presence of the two additional constraints. Neither the cost (6.5404e+6), nor the solution changes. This proves that the two additional constraints are effective to sort out the two cases. As the cost of the first case is lower, we therefore choose it in the later

studies of open-loop OCP.

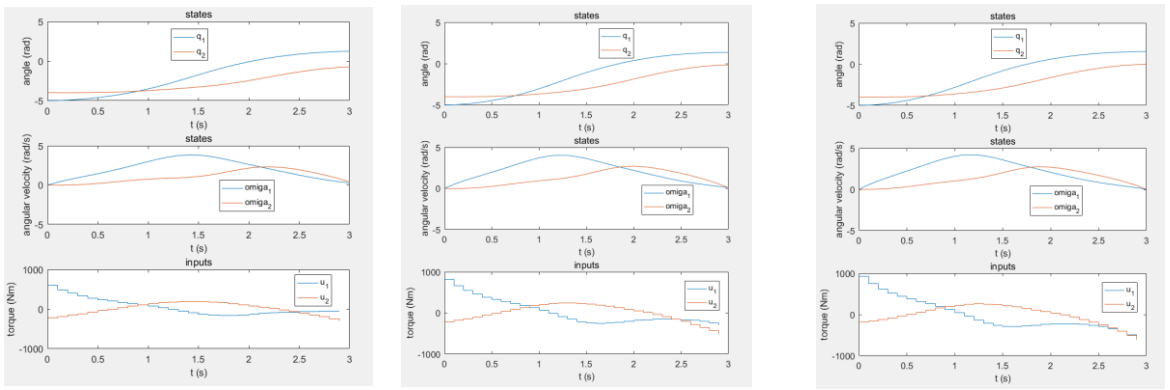
It is recommended to use terminal error as a part of minimized objective function, instead of giving the exact terminal conditions, because in real systems with modeling errors and disturbance, the OPC problem might be infeasible if the terminal condition is exact. This is especially true at the last few steps of closed-loop model predictive control process.

Here, we should further modify our two additional constraints because we don't want them to be active at the start point and the end point, where the terminal constraints are already not exact. Instead, two additional constraints serve as guidelines at the beginning of iteration.

$$\dot{q}_1(t) \geq \dot{q}_2(t) \text{ for } \frac{1}{12}t_f < t < \frac{1}{6}t_f$$

$$\dot{q}_1(t) \leq \dot{q}_2(t) \text{ for } \frac{5}{6}t_f < t < \frac{11}{12}t_f$$

Taking P as identity matrix in the penalty term in problem (4), we make trials changing the weight w of terminal term to find a compromise between accuracy and iteration time. The stability and region of attraction will be analyzed later. The reference is, that the method with exact terminal constraints takes 25 iterations and costs 6.5404e+06. With terminal error as a part of minimized objective function and with weight 1e+7 for each terminal error and rate error, the value function $V_T^0(x) = V_T^0 = \int_{t_k}^{t_k+T} u^{*T}(\tau)Ru^*(\tau)d\tau$ (stage cost, which can be interpreted as consumed energy) is minimized to 4. 0833e+06, and it takes 30 iterations. When given weight of 1e+8, minimized to 5.9909e+06, it takes 34 iterations but the terminal error is much smaller; when given weight of 1e+6, it takes 29 iterations, the cost is minimized to 1.9290e+06, but the terminal error of angle is too larger to be acceptable. (See Figure 3.)



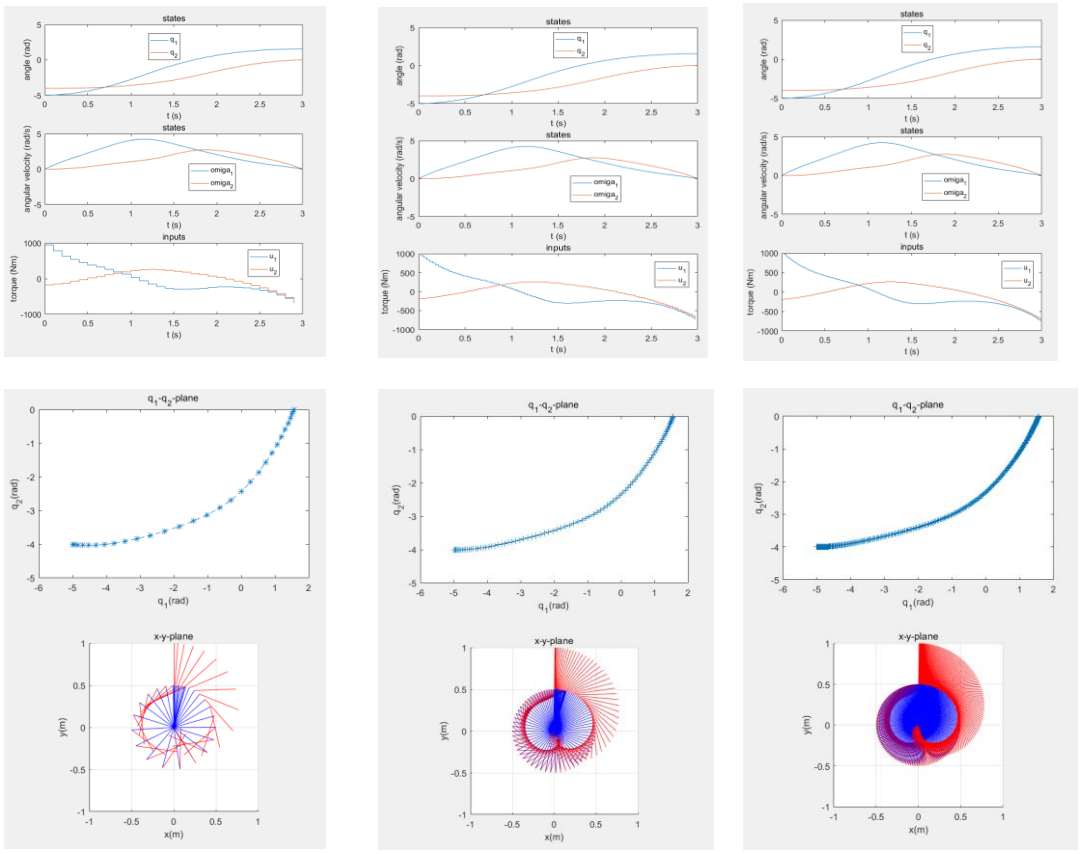
(a) R=diag([10;10]), weight 1e+6, cost 1.9290e+06, 29 iterations

(b) R=diag([10;10]), weight 1e+7, cost 4.0833e+06, 30 iterations

(c) R=diag([10;10]), weight 1e+8, cost 5.9909e+06, 34 iterations

Figure 3: Solution with different end-state penalty weight

We compare differences of cost and iteration times with different numbers of sampling points and different integrators. Here we only apply the method with given exact terminal constraints.

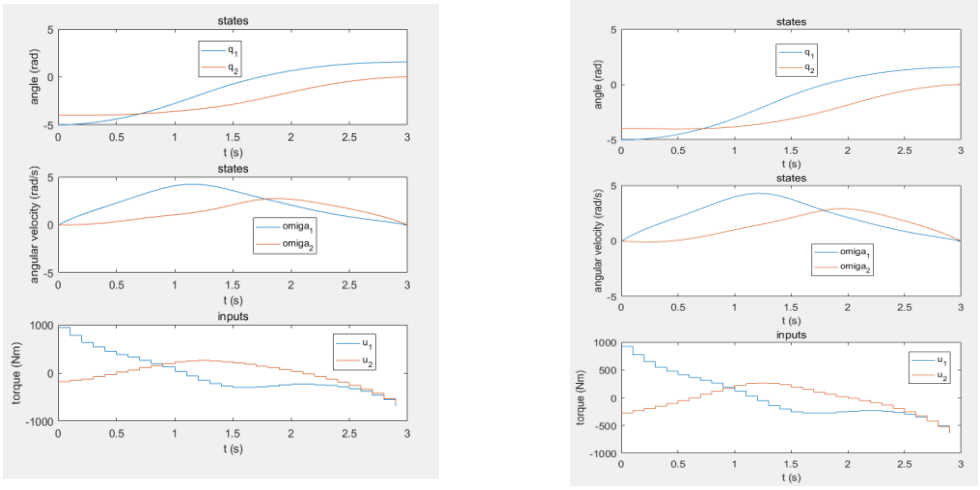


(a) N=30, cost 6.5404e+06, 25 iterations

(b) N=100, cost 6.5191e+06, 39 iterations

(c) N=300, cost 6.5174e+06, 52 iterations

Figure 4: Cost reduces in micro-scale and iteration increases slightly with the increase of sampling points



(a) Using rk4 integrator, N=30, cost 6.5404e+06, 25 iterations

(b) Using Euler integrator, N=30, cost 6.3705e+06, 32 iterations

Figure 5: Using different integrators

c) Formulation of Minimum Time Problem

We formulate a minimal time problem:

$$\min_{u(\cdot),t_f} t_f$$

subject to:

$$B(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = u$$
$$[q_1(0), q_2(0), \dot{q}_1(0), \dot{q}_2(0)]^T = [-5, -4, 0, 0]^T$$
$$[q_1(t_f), q_2(t_f), \dot{q}_1(t_f), \dot{q}_2(t_f)]^T = [\frac{\pi}{2}, 0, 0, 0]^T$$
$$u(\cdot) \in \hat{C}([0, t_f], [-1000, 1000])$$
$$\dot{q} \in [-\frac{3\pi}{2}, \frac{3\pi}{2}]$$

d) Time Transformation

To transform the problem above into a fixed end-time problem, we introduce:

$$t = \tau t_f$$
$$\tau \in [0, 1]$$

As we rewrite the dynamic function

$$B(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = u$$

in the form of state variables:

$$\frac{dx(t)}{dt} = f(t, x(t), u(t))$$

We can now substitute t with τ:

$$\frac{dx(\tau)}{d\tau} = t_f f(\tau, x(\tau), u(\tau))$$

So the original minimum time problem is reformulated as a fixed end-time problem with an extra decision variable tf:

$$\min_{u(\cdot),t_f} t_f$$

subject to:

$$\frac{dx(\tau)}{d\tau} = t_f f(\tau, x(\tau), u(\tau))$$
$$[q_1(0), q_2(0), \dot{q}_1(0), \dot{q}_2(0)]^T = [-5, -4, 0, 0]^T$$
$$[q_1(1), q_2(1), \dot{q}_1(1), \dot{q}_2(1)]^T = [\frac{\pi}{2}, 0, 0, 0]^T$$
$$u(\cdot) \in \hat{C}([0,1], [-1000, 1000])$$
$$\dot{q} \in [-\frac{3\pi}{2}, \frac{3\pi}{2}]$$

e) Numerical Solution of Minimum Time Problem

We solve the problem numerically. The minimal time is 2.303s and the plots are as shown in Figure 6.

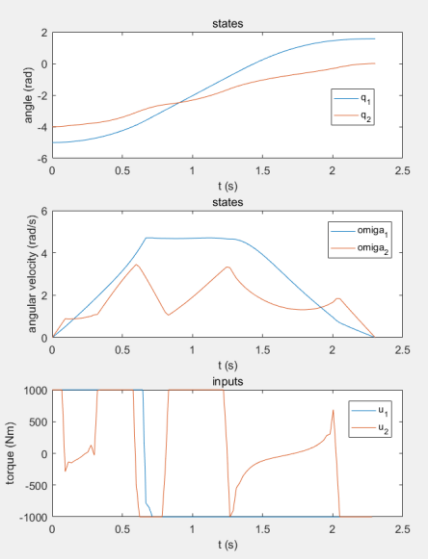


Figure 6: Numerical Solution of minimum time problem

The input of the first joint follows a Bang-Bang control phenomenon. The input value takes either the maximum or the minimum of the input constraint so as to let the states reach the target in minimum time. However, we find out that the input and states with respect to the second joint is badly controlled. The input often unnecessarily reaches the extreme value from one side to the other. This makes the states tremble and causes extra energy consumption. The reason behind that can be that the second joint is not critical for the minimum time. But with the final time as the only objective, the input for the second joint is lack of proper objective and therefore not optimized at all.

We reformulate the objective to take the control input energy of the second joint into consideration:

$$\min_{u(\cdot),t_f} t_f + \int_0^1 u^T(\tau) R u(\tau) d\tau$$

subject to:

$$\frac{dx(\tau)}{d\tau} = t_f f(\tau, x(\tau), u(\tau))$$

$$[q_1(0), q_2(0), \dot{q}_1(0), \dot{q}_2(0)]^T = [-5, -4, 0, 0]^T$$

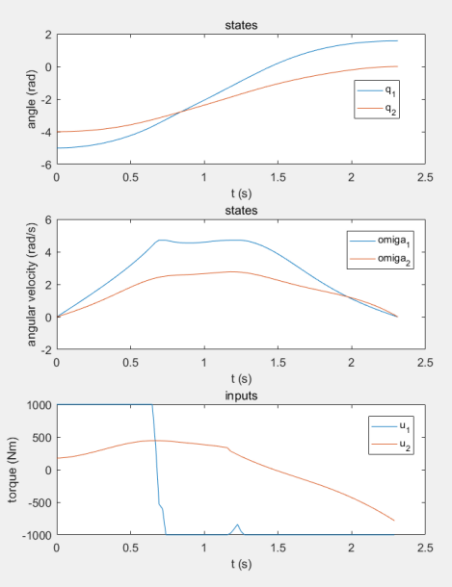
$$[q_1(1), q_2(1), \dot{q}_1(1), \dot{q}_2(1)]^T = [\frac{\pi}{2}, 0, 0, 0]^T$$

$$u(\cdot) \in \hat{C}([0,1], [-1000, 1000])$$

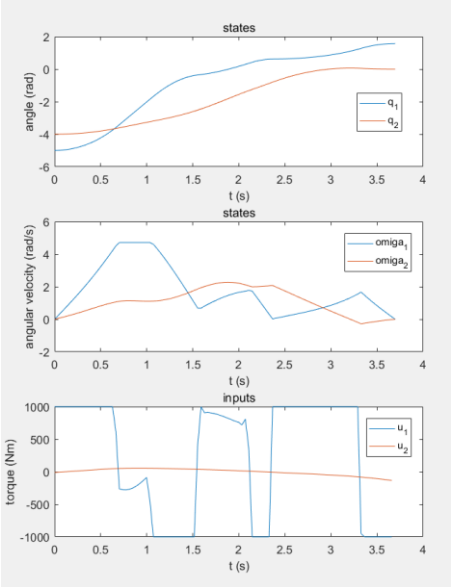
$$\dot{q} \in [-\frac{3\pi}{2}, \frac{3\pi}{2}]$$

We set R=diag([0, 1e-6]), so that we only consider the energy consumption (with small portion) of the second joint, and still let the first joint only be optimized to minimize the time. The final time is only 0.01s (0.43%) longer, but the behavior is much more regular, as shown in Figure 7(a). Important

is, that the weight for the second joint is small. If this is not the case, for example when R=diag([0, 1e-4]), it turns out that the input on the second joint will try to stay as close to zero as possible. This makes the problem interestingly approaching a double inverted pendulum swing-up problem. And of course, the final time will be very long, as shown in Figure 7(b).



(a) When R=diag([0, 1e-6]). The minimal time is 2.313s.



(b) When R=diag([0, 1e-4]). The minimal time is 3.696s.

Figure 7: Controls with different weights of input energy

3. Model Predictive Control

a) Uncertainties and Disturbances

The disturbances and uncertainties can be categorized. When a factor is already modelled in the differential equations of motions, there can be parameter errors. We call them parameter uncertainties. Some factors are either varying or too difficult to be described. So these factors are not modelled in the differential equations of motions due to simplicity. They are regarded as systematic modelling errors. Other factors are random disturbances. They include the noise at the sensors and the disturbance to the system.

Table 1: Classification of uncertainties and disturbances

parameter uncertainties	systematic modelling errors	random disturbances
manufacturing errors of length, mass, moment of inertial	moment of inertial of end-effector, moved center of mass of end-effector	measuring noise
measuring errors in modelling, especially difficult to determine moment of inertial	friction at joints	random disturbance such as wind
mass variation of end-effector	forces caused by cable deformation	collision with unknown objects or human
	deformation and vibration due to the stiffness of materials	
	extra loads due to the working condition of end-effector	
	dynamics of lower control units such as delay and inaccuracy of torque	

Some of the uncertainties and disturbances can be modelled mathematically and thus make the differential equation more complicated. For example, when the moment of inertial of end-effector is considered,

$$\sum_{i=1}^{\nu}\{[\frac{^{N}d}{dt}(m_i\,{}^{N}\vec{v}^{S_i})-\vec{F}_i^{e}]\cdot\frac{\partial^{N}\vec{v}^{S_i}}{\partial u_j}+[\frac{^{N}d}{dt}\vec{H}_{S_i}-\vec{M}_{S_i}^{e}]\cdot\frac{\partial^{N}\vec{\omega}^{K_i}}{\partial u_j}\}=0$$

the term $[\frac{^{N}d}{dt}\vec{H}_{S_i}-\vec{M}_{S_i}^{e}]\cdot\frac{\partial^{N}\vec{\omega}^{K_i}}{\partial u_j}$ for i=3 can no longer be ignored. When the viscos friction is considered, $\vec{M}_{S_i}^{e}$ (for i=1 and 2) not only include u1 and u2 but also $-c_1\dot{q}_1$ for joint 1 and $-c_2(\dot{q}_2-\dot{q}_1)$ for joint 2. When there are extra loads due to the working condition of end-effector, $\vec{F}_3^{e}$ and $\vec{M}_{S_3}^{e}$ are no more zero.

b) Controller Design

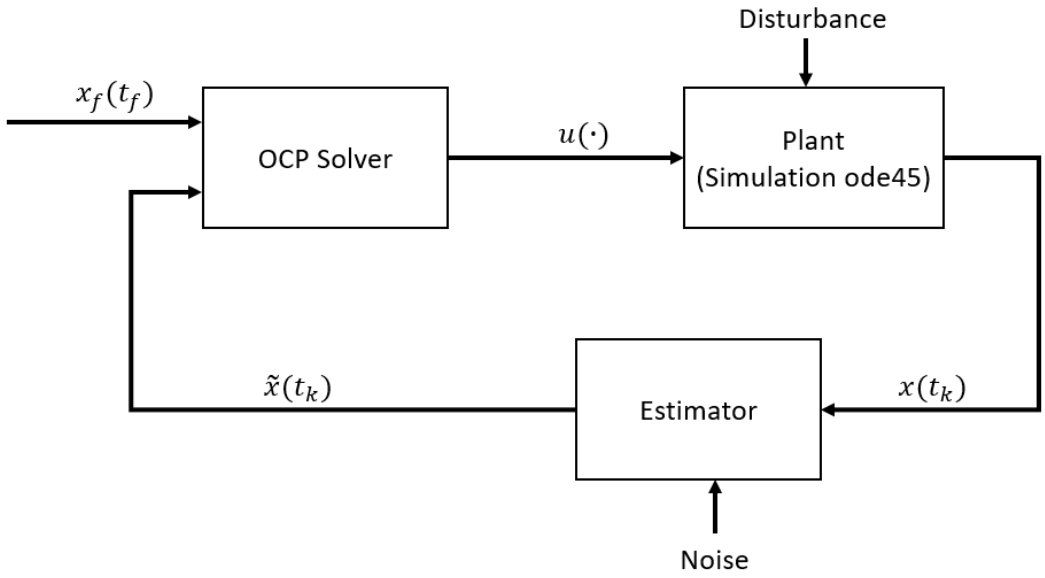


Figure 8: Block diagram of model predictive controller

Table 2: Scheme of model predictive control

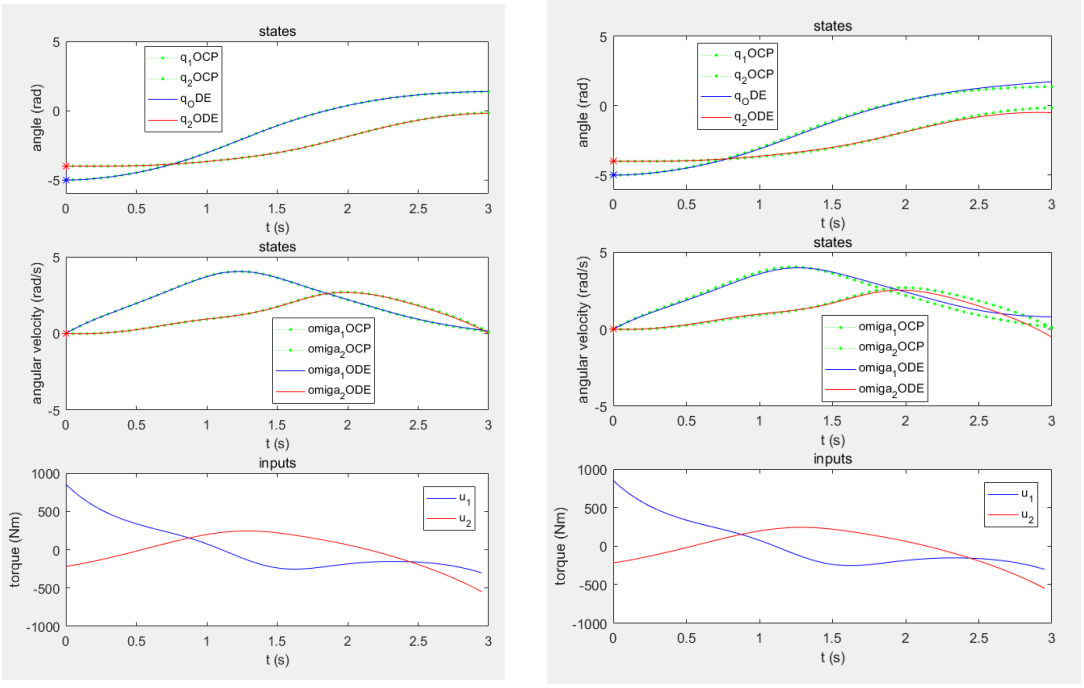
1. State measurement / estimate $x(t_k)$
2. Solve OCP: $$\min_{u(\cdot)} \int_{t_k}^{t_k+T} u^T(\tau)Ru(\tau)d\tau + w\big(x\big(t_f\big) - x_f\big)^T(x(t_f) - x_f)$$ $\forall\tau \in [t_k, t_k + T]$: $$B(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = u$$ $$[q_1(t_k), q_2(t_k), \dot{q}_1(t_k), \dot{q}_2(t_k)]^T = x(t_k)^T$$ $$u \in [-1000, 1000]$$ $$\dot{q} \in [-\frac{3\pi}{2}, \frac{3\pi}{2}]$$ $$T = t_f - t_k$$
3. Apply $\bar{u}^*(\tau)$ for $\tau \in [t_k, t_{k+1})$, go to Step 1.

Model predictive control (MPC) is repeated optimal control. This closed-loop MPC controller consists of 3 fundamental blocks: estimator, OCP solver, and plant, as shown in figure 8.

The estimator takes the sampled state $x(t_k)$ and noise parameters as inputs, and outputs the estimations $\tilde{x}(t_k)$ with certain period: $t_{k+1} - t_k = \delta$.

The fixed end–time OCP solver receives the estimations and the set-points $x_f(t_f)$, and then computes the optimal output $u(\cdot)$ with the period δ. To be noticed is that the prediction horizon T is so determined that the solver always compute the control process from present to the fixed end-time of 3 seconds. This means that T varies from 3s to 0 throughout the process. When the disturbance and uncertainties are taken into consideration, we need to adjust some controller settings. We should not use accurately given end state anymore because an inaccurate state estimation caused by noise might make the solver unable to find a feasible solution in the last few steps before the process reaches 3 seconds. Therefore we set the terminal error as a part of minimized objective function with weight 1e+7 for each of the terminal error of state variables.

The newest optimal output $u(\cdot)$ is exerted on the plant (possibly under disturbance) for a period of δ, before the next output is computed. The behavior of the plant under each optimal output in its exerting period is computed by ode45. Important is the method to deal with time-varying $u(\cdot)$ in the differential equation of ode45. In the period of δ, $u(\cdot)$ is discretized into a sequence of input vectors with the respective time points. When ode45 call the differential equation, the time may not be exactly at those time points where input vectors are given. Here we use interpolation with the method "previous" but not "linear", because the OCP is solved with stair-shaped input vectors u(.) in the solver. This is proven correct through the trial and plot of an open-loop control process in Figure 9.



(a) Open loop control comparing OCP-solver's states and ode45-solver's states with u(t) interpolation method "previous"

(b) Open loop control comparing OCP-solver's states and ode45-solver's states with u(t) interpolation method "linear"

Figure 9: Proof that the "previous" interpolation is the correct application of input variables

We design a closed-loop simulator for MPC controller. The important parameters include control (and measurement) interval, discretization interval of the OCP, and noise variation. The effectiveness of this simulator is proved at zero noise, as shown in Figure 10. The optimized inputs stays consistent so the curves in the third plot "optimized inputs for each step" coincide. Principle of Optimality (Lemma 3.17, B. Chachuat. Nonlinear and Dynamic Optimization – From Theory to Practice) ensures that the same control inputs is always optimal in any time interval from t1 to terminal time, as t1 varies from the initial time to the terminal time. So the control inputs computed at different steps should overlap with each other in the plot of closed-loop control if there are no disturbance or uncertainties.

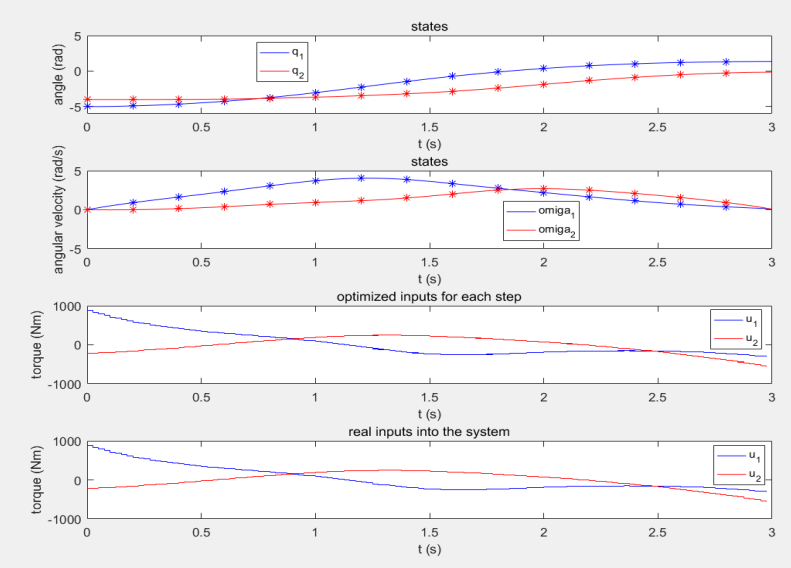


Figure 10: Overlapping of the third plot proves effectiveness of the controller with control interval 0.2s, noise variance 0

c) Modeling Uncertainties

If we set incorrect parameters in the model, the differential equation for OCP is changed with parameters b1=180, b2=45, but the differential equation for ODE is not changed.
According to the dynamics, we have:

$$b_1 = J_1 + J_2 + \frac{1}{4}l_1^2 m_1 + l_1^2 m_2 + l_1^2 m_3 + \frac{1}{4}l_2^2 m_2 + l_2^2 m_3$$

$$b_2 = l_1 l_2 m_2 + 2 l_1 l_2 m_3$$

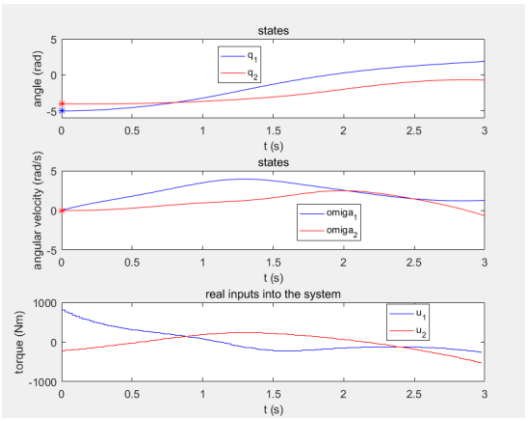
$$b_3 = J_2 + \frac{1}{4}l_2^2 m_2 + l_2^2 m_3$$

The incorrect parameter can happen in practice when angular momentum J1, mass m1 and arm length l1 are measured with error. In this case, b3 is still correct, but b1 and b2 suffer from errors.

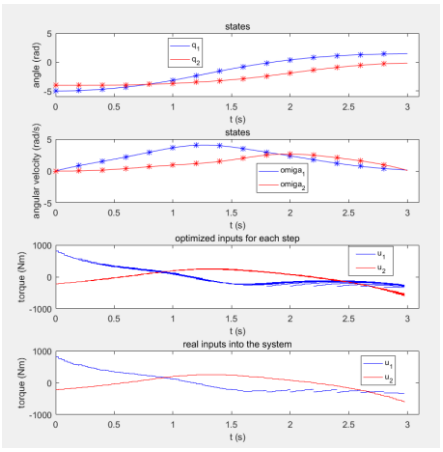
An open-loop system can be achieved simply by setting control (and measurement) interval as 3 seconds. The end-state errors are quite large. The angular errors are more than 27 degrees, and the angular velocities are larger than 50 degrees per second, as shown in Figure

11. The detailed data are shown in Table 3.

The end-state errors are much smaller in close-loop simulations. The real inputs into the system is adjusted in closed-loop. It is especially obvious at the end of the process, and the real output can be discontinuous at the beginning of each control interval due to the large adjustment. The closed-loop controllers with modelling error achieve better performance than the open-loop controller without modelling error. And when the control frequency is larger, the performance is enhanced.



(a) open-loop control with incorrect parameters



(b) closed-loop control with incorrect parameters

Figure 11: Model uncertainties and control accuracy

Table 3: Errors in cases of open and closed loop, correct and incorrect parameters

	open-loop with correct parameters	closed-loop with correct parameters, interval 0.2s	open-loop with incorrect parameters	closed-loop with incorrect parameters, interval 0.2s	closed-loop with incorrect parameters, interval 0.1s
error q_1(rad)	0.0462	0.0088	0.4742	0.0062	0.0012
error q_2(rad)	0.0674	0.0051	0.6235	0.0055	0.0043
error $\dot{q}_1$ (rad/s)	0.1091	0.0030	1.077	0.0341	0.01695
error of $\dot{q}_2$ (rad/s)	0.0859	0.0010	0.8646	0.0129	0.0026

d) Measurement Error

Now the additional Gaussian noise during state estimation is added to the system. The optimized solutions are different from each other, but similar in form in the first two seconds. Because that the most deviated part of the solutions are always the initial point of each solution, which will be directly sent to the actuators, the real inputs are very discontinuous at these points in many situations, especially when the error from Gaussian noise is large at this step. This might cause trembles of the robot. Also to be noticed is that in the last second, the optimized solutions no longer comply to the form as before, and the trembles are very serious. This problem is due to the change of component proportions of the objective function. If the final errors are directly added to the integrated "energy" cost of actuators, the final errors have larger and larger proportion as the process approaches the end time, because the integrated interval of the "energy" cost of actuators becomes smaller and smaller, and the controller is more and more sensative to the estimation error.

If we modify the weight of the final errors:

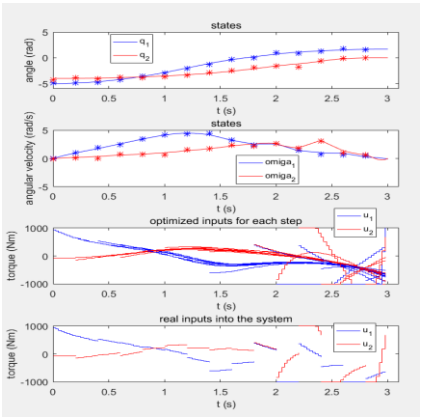
$$\min_{u(\cdot)} \int_{t_k}^{t_f} u^T(t)Ru(t)dt + w(t)\big(x\big(t_f\big) - x_f\big)^T(x(t_f) - x_f)$$

as:

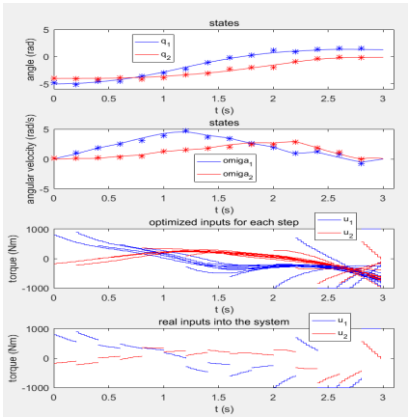
$$w(t) = \frac{t_f - t_k}{t_f} w$$

where tf is the final time (3s), and tk is the current time at which the integration part of the cost function starts. This means, the weight of final error becomes smaller as time approaches the final time, and the proportion between the integrated part and final error part of the cost function remain almost unchanged. This new design of cost funtion milds the abnormal behavior and serious trembles at the end of the control process, but it makes the final error worse.

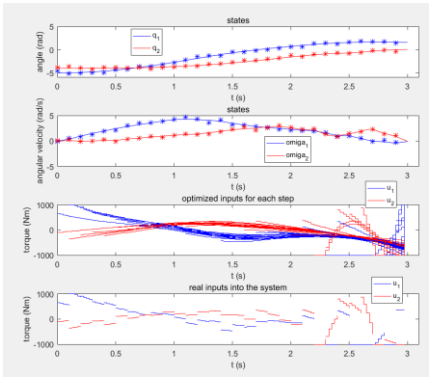
Simulation experiments are carried out with different cost functions and different control intervals in Figure 12. However, the result of each situation can vary greatly due to the random error.



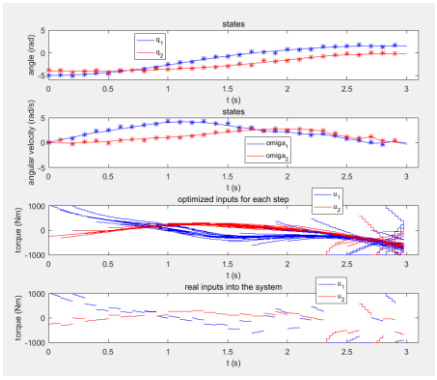
(a) control interval 0.2s, noise variation 0.2, fixed weight of final error, serious trembles at the end, errors [0.1092, 0.0024, -0.1122, -0.0196]



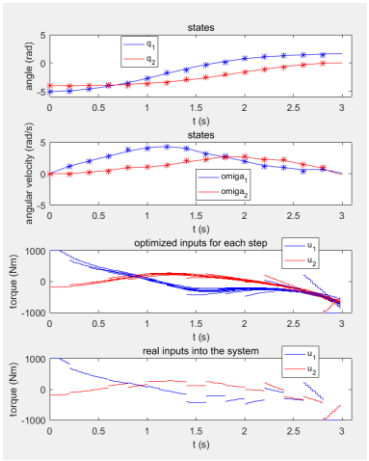
(b) control interval 0.2s, noise variation 0.2, adjustable weight of final error, large errors at the end, errors [-0.2969, -0.1291, 0.0160, 0.0417]



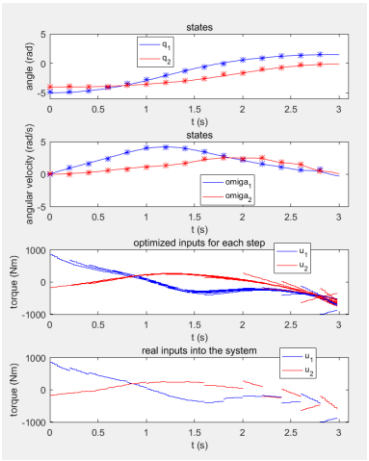
(c) control interval 0.1s, noise variation 0.2, fixed weight of final error, serious trembles at the end, errors [0.0092, 0.0250, -0.0814, -0.0552]



(d) control interval 0.1s, noise variation 0.2, adjustable weight of final error, serious trembles at the end, errors [-0.1042, -0.2038, -0.1002, -0.3449]



(e) control interval 0.2s, noise variation 0.1, fixed weight of final error, serious trembles at the end, errors [0.0837, 0.0061, 0.0754, -0.1243]



(f) control interval 0.2s, noise variation 0.1, fixed weight of final error, serious trembles at the end, errors [-0.0700, -0.1177, -0.3115, 0.0254]

Figure 12: Performance of model predictive controller with different cost functions and different control intervals

e) Stability Properties

This nonlinear model predictive control strategy follows the scheme of NMPC with replaced terminal constraint. It is not a trajectory tracking problem, because the desired trajectory r(t) is not explicitly given except for the final state. And the integrated part of objective is only dependent on input but not on tracking errors. So we still regard the problem as a set-point stabilization problem without terminal constraint in term of its form.

1. State measurement / estimate $x(t_k)$

2. Solve OCP:

$$\min_{u(\cdot)} \int_{t_k}^{t_k+T} F\big(\bar{x}(\tau), \bar{u}(\tau)\big)d\tau + \beta E(\bar{x}(t_k+T))$$
$$\forall \tau \in [t_k, t_k+T]: \quad \dot{\bar{x}} = f(\bar{x}, \bar{u}), \qquad \bar{x}(t_k) = x(t_k)$$
$$\bar{x}(\tau) \in \mathcal{X}, \bar{u}(\tau) \in \mathcal{U}$$

3. Apply $\bar{u}^*(\tau)$ for $\tau \in [t_k, t_{k+1})$, go to Step 1.

The horizon T spans from present to the fixed end-time of 3 seconds, so the terminal penalty is always computed with respect to the same terminal time point $t_k + T = t_f = 3$s.

The Value function is in this specific case:

$$V_T^{\beta}\big(x(t_k)\big) \coloneqq \int_{t_k}^{t_k+T} u^{*T}(\tau) R u^*(\tau) d\tau + \beta\big(x^*\big(t_f\big) - x_f\big)^T (x^*(t_f) - x_f)$$

We consider a nominal setting. a) There is no plant-model mismatch, i.e. the nonlinear differential equations for the OCP solver and for the ODE solver are identical without incorrect parameters. b) State feedback is exactly the plant state at the same time point, i.e. there is no measurement error or measurement delay. c) An optimal solution for each step of sampling and control exists and is attained. Since the terminal constraints are replaced by terminal penalty, there always exist feasible solutions and the optimal solution is computed by the OCP solver.

Now we verify if the three assumptions (Slides Page 19, Lecture Nonlinear Model Predictive Control – Theory and Applications SoSe 2018, Dr. Timm Faulwasser) hold.

A1 (Steady state): Given x_s, there exists $u_s \in \mathcal{U}$, such that $0 = f(x_s, u_s)$ and $(x_s, u_s) \in \text{int}(\mathcal{X} \times \mathcal{U})$. W.l.o.g. consider $(x_s, u_s) = (0, 0)$.

A2 (Lower boundedness of F): There exists a class-$\mathcal{K}$ function $\alpha : \mathbb{R}_0^+ \to \mathbb{R}_0^+$, such that $F(x, u) \geq \alpha(\|x - x_s\|)$, and $F(0, 0) = 0$.

A3 (Absolute continuity of ODE solutions): For all $x_0 \in \mathcal{X}$, and any $u(\cdot) \in \hat{\mathcal{C}}([0, T], \mathcal{U})$, the solution $x(\cdot, x_0, u(\cdot))$ exists and is absolutely continuous.

A1 requires that the robot arms can stay at steady state at the set-point while applying feasible input u. We consider the terminal pose $q_1 = \frac{\pi}{2}, q_2 = 0$ without the need to convert to $(x_s, u_s) = (0,0)$. The ODE degenerates to algebraic equations:

$$g(q) = u$$

and we get $u_s = (0,0)^T$. Both state and input variables are certainly feasible in this case. Considering A2, we get the F function

$$F(x, u) = F(u) = u^T(t)Ru(t)$$

from the objective function:

$$\min_{u(\cdot)} \int_0^{t_f} u^T(t)Ru(t)dt + w\big(x(t_f) - x_f\big)^T(x(t_f) - x_f)$$

where $R = \begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix}$

It is easy to prove that $F(\mathbf{0}, \mathbf{0}) = 0$. The goal is to find a class-K function α, such that $F(x, u) \geq \alpha(\|x - x_s\|)$. The function F is independent on x, so we only need to adjust u to let F be larger than the class-K function. When we take the extreme value 1000 of u, F can be adjusted to the maximum of 2e+7. As long as we find a class-K function α with the upper bound 2e+7, A2 is satisfied. An example can be of the type $\alpha(x) = k \arctan(x)$, $k \leq 2 \cdot 10^7$.

As for A3, there always exists the feasible solution as long as the required time from start to end is larger than or equal to the minimum time calculated above 2.303s. The restriction on input variables $u \in [-1000, 1000]$ and the structure of the nonlinear differential equation guarantee that the state variables are absolutely continuous.

The Assumptions A1-A3 hold, and then we check the additional assumptions for the theorem of convergence with replaced terminal constraint (Slides Page 51, Lecture Nonlinear Model Predictive Control – Theory and Applications SoSe 2018, Dr. Timm Faulwasser).

Assumption A4: For some $\gamma > 0$, let $\mathcal{E}_\gamma = \{x \in \mathcal{X} \mid E(x) \leq \gamma\}$ and E satisfy the quasi infinite-horizon NMPC stability conditions for specific values of $\delta > 0, T > 0$.

A5 The set Ω_T^0 is compact and $0 \in \text{int}(\Omega_T^0)$.

A6 There exists $\alpha \in \mathcal{K}_\infty$, such that $V_T^0(x) \leq \alpha(\|x\|)$.

Assumption A4 can be proved by Theorem 1 of Quasi-infinite horizon NMPC (Findeisen&Allgöwer, The quasi-infinite horizon approach to nonlinear model predictive

control, 2003). If we need an accuracy of angle at +/-0.5° , and angular rate at +/-0.25° /sec, we set γ=0.625.

$$\mathcal{E}_\gamma = \{x \in \mathcal{X} \,|\big(x(t_f) - x_f\big)^T \big(x(t_f) - x_f\big) < 0.625\}$$

To verify A5, we have $V_T^0(x) = V_T^0 = \int_{t_k}^{t_k+T} u^{*T}(\tau) R u^*(\tau) d\tau$, which is always smaller than positive infinity because T is smaller than or equal to 3 seconds and u is constrained. Therefore,

$$\Omega_T^0 = \{x \in \mathcal{X} | V_T^0(x) < \infty\} = \{x \in \mathcal{X}\}$$

is the same set as the set of constraints for state variables. That is to say, if the NMPC would be stable, any state satisfying the state constraints is in the region of attraction as long as the terminal set-point can be reached in 3s. 0 is an element in this set. Now is to prove that the set is compact. We have two state variables already constrained in compact set:

$$\dot{q}_i \in \left[-\frac{3\pi}{2}, \frac{3\pi}{2}\right], i = 1,2$$

Although the range of q1 and q2 are not given, they are constrained implicitly by the time of 3s. Therefore, there exists a compact set also for the states of angles.

A6 involves the upper bound of value function. We need to modify the problem setting to let the upper position be the origin.

$$x = [x_1\ x_2\ x_3\ x_4]^T = [q_1 - \frac{\pi}{2}, q_2, \dot{q}_1, \dot{q}_2]^T$$

When the robot arms reach the upper position, x=0, the value function will also be zero. As the state variables go further away from this position, the value function (with stage cost but without terminal cost) will also increase, but upper bounded. There always exists a K-function $\alpha \in K_\infty$, such that $V_T^0(x) \leq \alpha(\|x\|)$.

When all the above assumptions hold, there exist a weight value $\beta \in (0, \infty)$, such that the NMPC scheme achieves asymptotic stability $\lim_{t\to\infty} \|x(t)\| = 0$. But in the case of this project, the requirement on time of 3s is stricter. And the region of attraction is given only necessarily by $\Omega_T^0 = \{x \in \mathcal{X} | V_T^0(x) < \infty\} = \{x \in \mathcal{X}\}$.

A way to find the sufficient weight value $\beta \in (0, \infty)$ of terminal cost to achieve stability is given with the help of local linearization of the model and Lyapunov equation (Chen&Allgöwer, A Quasi-Infinite Horizon Nonlinear Model Predictive Control Scheme with Guaranteed Stability). The weight value is integrated in the matrix P, so that the terminal cost becomes:

$$\big(x(t_f) - x_f\big)^T P (x(t_f) - x_f)$$