Saving Energy Catching A Ball: An Application of Shrinking Horizon Optimal Control to A Robot Arm

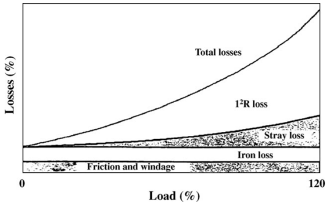
Xiyu Fu

28 June 2018

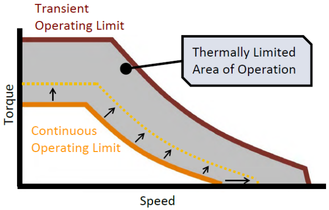
Outline

Introduction and Motivation

- About 8% of energy are eaten by robots in production sector and the copper loss I^2R accounts for 55% - 60% of the total energy loss in robots' operations
- Lowering heat generation could enlarge the continuous operating capacity of robots
- The heat generated during operation is also a threat to robot accuracy
- Minimizing the thermal energy generation is of vital importance in improving energy efficiency and robots performance



(a) Energy losses


Transient Operating Limit
Thermally Limited Area of Operation
Torque
Continuous Operating Limit
Speed

(b) Thermal limit

Introduction and Motivation

- Many different ways to reduce thermal energy generation, e.g., usage of permanent magnetics, increasing cross section area of windings, improving power electronics, etc.
- One interesting way is by using optimal motion planning
  - requires no or few hardware modification
  - could be adopted in a wide range of robot operations
- Ball catching is used as a benchmark task in this project
- Robot used for this task is the IRB 120

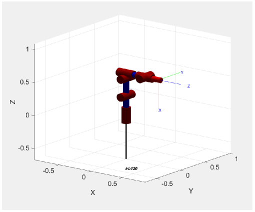


Figure: IRB 120 and its model

Problem Statement

Catching a flying a ball is a complex task. It could be divided into different subtasks:

- Plan:
  - Finding a point in the ball's trajectory to catch it
  - Planning a path that leads the robot to the catch point
- Perception:
  - Estimating the trajectory of the ball
  - Locating the robots states based on feedback data
- Control:
  - Computing an thermal energy optimal control policy
  - Calculating real-time optimal control inputs based on feedback

Problem Statement

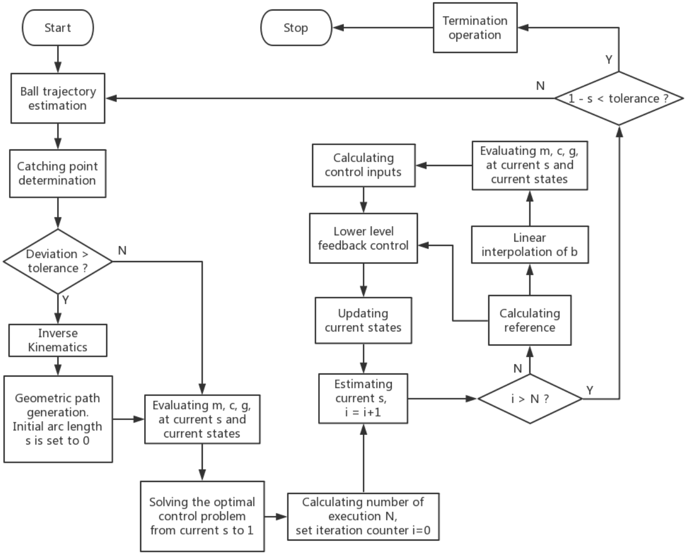
The thermal energy optimal control problem could be formulated as follows:

$$\begin{aligned} \min_{x,T} \quad & \int_0^T \sum_{i=1}^{6} \frac{\tau_i^2(t)}{\tau_{max,i}^2} dt \\ s.t.\ \tau \ &= \ M(q)\ddot{q} + C(q,\dot{q})\dot{q} + F_v\dot{q} + F_s sgn(\dot{q}) + g(q) \\ & 0 \leq T \leq T_{catch} \\ & \tau_{min,i} \leq \tau_i \leq \tau_{max,i} \quad i = 1, 2, ..., 6 \\ & q_i = I(t_i) \\ & x(0) = x_{init} \\ & x(T) = x_{end} \end{aligned}$$

The bridge that connects joint torques to thermal energy is the armature current:

$$\begin{aligned} \tau &= K_\phi I \\ Q &= RI^2 \end{aligned}$$

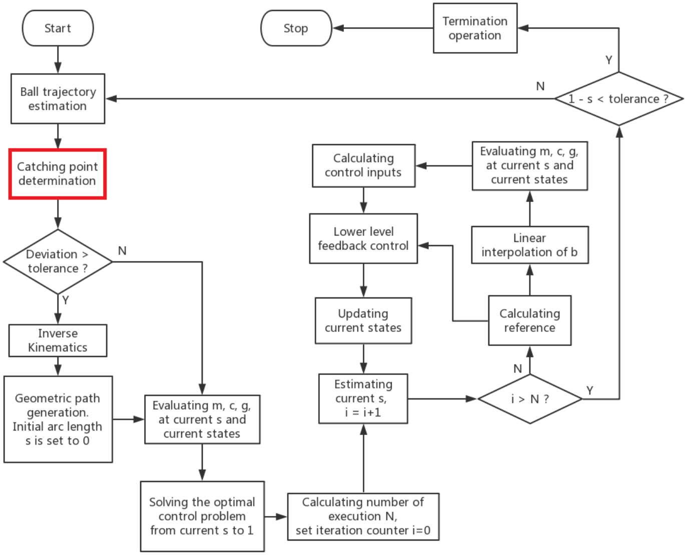
Overview of the Scheme



Start
Stop
Termination operation
Y
N
1 - s < tolerance ?
Ball trajectory estimation
Catching point determination
Calculating control inputs
Evaluating m, c, g, at current s and current states
Lower level feedback control
Linear interpolation of b
Deviation > tolerance ?
N
Y
Updating current states
Calculating reference
Inverse Kinematics
N
Geometric path generation. Initial arc length s is set to 0
Evaluating m, c, g, at current s and current states
Estimating current s, i = i+1
i > N ?
Y
Solving the optimal control problem from current s to 1
Calculating number of execution N, set iteration counter i=0

Overview of the Scheme

Catch Point Determination



Start
Stop
Termination operation
Y
N
1 - s < tolerance ?
Ball trajectory estimation
Catching point determination
Calculating control inputs
Evaluating m, c, g, at current s and current states
Lower level feedback control
Linear interpolation of b
Deviation > tolerance ?
N
Y
Updating current states
Calculating reference
Inverse Kinematics
N
Estimating current s, i = i+1
i > N ?
Y
Geometric path generation. Initial arc length s is set to 0
Evaluating m, c, g, at current s and current states
Solving the optimal control problem from current s to 1
Calculating number of execution N, set iteration counter i=0

Overview of the Scheme

Catch Point Determination

- In this project, a heuristic method is used to determine the catch point:
  - 1) Find the point at which the ball flies into the workspace. Let's call it point C
  - 2) The balls trajectory is linearized at point C
  - 3) Find the point that is closest to the robot's end effector, along the linearized trajectory, by the following equation:

$$t = \frac{x - x_C}{u} = \frac{y - y_C}{v} = \frac{z - z_C}{w}$$
$$(x - x_{ee},\ y - y_{ee},\ z - z_{ee}) \cdot (u, v, w) = 0$$

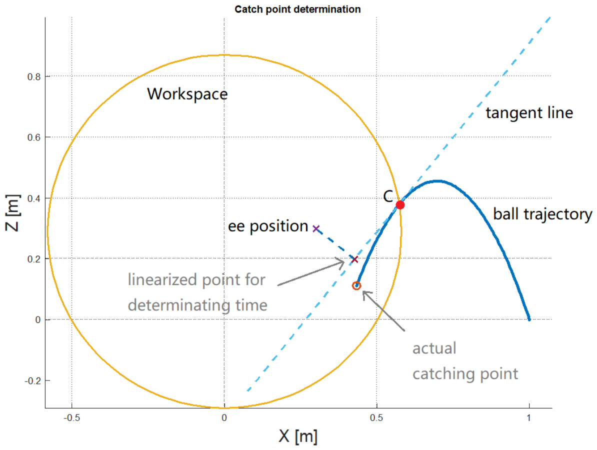
    Let's call it point N
  - Use the time stamp of point N to find the corresponding point in the real trajectory. And that point is the catch point

$$t_N = \frac{u(x_{ee} - x_C) + v(y_{ee} - y_C) + w(z_{ee} - z_C)}{u^2 + v^2 + w^2}$$

Overview of the Scheme

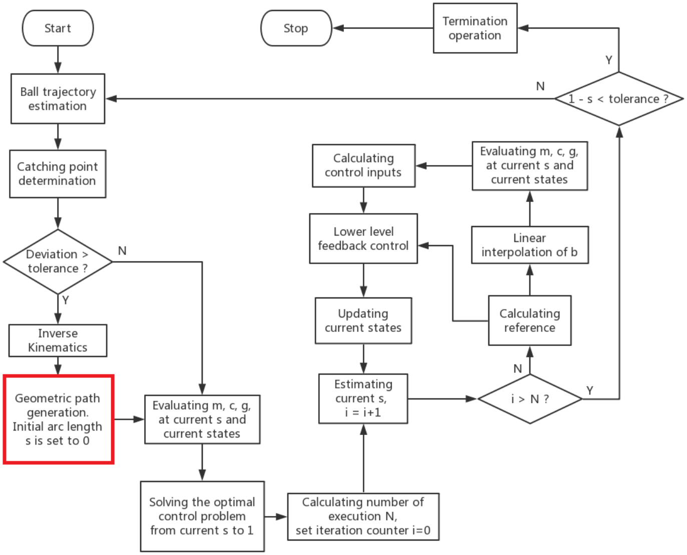
Catch Point Determination

This method is illustrated in the figure below:



Overview of the Scheme

Path Generation


Start
Stop
Termination operation
Y
N
1 - s < tolerance ?
Ball trajectory estimation
Catching point determination
Calculating control inputs
Evaluating m, c, g, at current s and current states
Lower level feedback control
Linear interpolation of b
Deviation > tolerance ?
N
Y
Updating current states
Calculating reference
Inverse Kinematics
Estimating current s, i = i+1
N
Y
i > N ?
Geometric path generation. Initial arc length s is set to 0
Evaluating m, c, g, at current s and current states
Solving the optimal control problem from current s to 1
Calculating number of execution N, set iteration counter i=0

Overview of the Scheme

Path Generation

To find a proper path, we first consider constraints needed:

- the end position has to be the catch point:

$$q(1) = q_{catch}$$

- and the robot starts from it initial position:

$$q(0) = q_{init}$$

- The robot should stop at the catch point, therefore: $\frac{dq}{dt}|_{end} = \frac{dq}{ds}\frac{ds}{dt} = 0$, but $\frac{dq}{ds}$ can't be 0 otherwise we get numerical difficulties, so:

$$\frac{ds}{dt}|_{end} = 0$$

- For the same reason, to have robot start from standstill:

$$\frac{ds}{dt}|_{init} = 0$$

Overview of the Scheme

Path Generation

There are 4 constraints needed to be imposed on the path. It should be a function that has at least 4 parameters.

- One simple path is a 3rd order polynomial:

$$q(s) = c_0 s^3 + c_1 s^2 + c_2 s + c_3$$

- But when the path is regenerated due to change of catch point, the new path is better to be close to the old one. We use a 4th order polynomial. The additional parameter is used to make the change in path small by solving the following optimization problem:

$$\begin{aligned} \min_{c} \quad & \sum_{i=0}^{N} ||q_{new}(s_i) - q_{old}(s_i)||_2^2 \\ s.t. \quad & q_{new}(0) = q_{previous} \\ & \frac{dq_{new}(0)}{ds} = \frac{dq_{previous}}{ds} \\ & q_{new}(1) = q_{end} \\ & \frac{dq_{new}(1)}{ds} = 0 \end{aligned}$$

Overview of the Scheme
Path Generation

The choice of path will definitely influence the total thermal energy. It even influences the feasibility of the problem. Is there a way to find a path that prevails other paths according to thermal energy standard? We could do it heuristically:

- The joint torque could be divided into 2 parts:
  - static part: $\tau_s = G(q)$
  - dynamic part: $\tau_d = M(q)\ddot{q} + c(q, \dot{q})\dot{q}$
- The following cost function is proposed:

$$f(\xi) = \frac{1}{2} w_g ||G(\xi)||_2^2 + \frac{1}{2} w_d ||K_d \xi||_2^2 + \frac{1}{2} w_a ||K_a \xi||_2^2$$

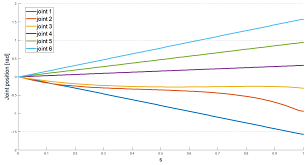
where $\xi = [q_1^1, q_1^2, ..., q_1^N, q_2^1, ..., q_6^N]^T$ is a vector that contains joint coordinates discretized within the joint limits. K_d and K_a are finite differencing matrices

Overview of the Scheme

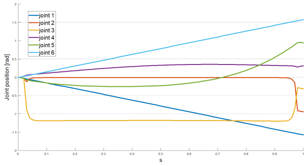
Path Generation

The resulting path is shown below.

- For a large w_d and w_a, only the 2nd joint (the elbow) tries to stay vertical because it is mostly influenced by the gravity
- For a large w_g, only wrist joints are still smooth
- The dimension of this problem is to high. Solving it costs a lot of time. The 4th order polynomial is used for faster path generation



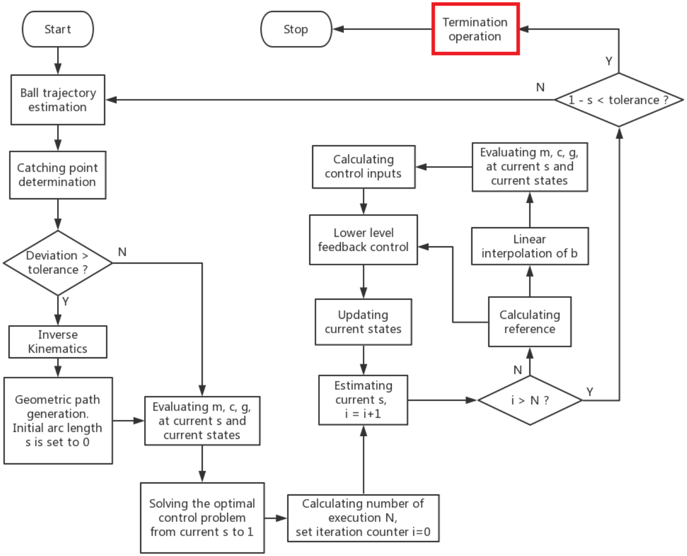
(a) Path solved with a large w_d and w_a



(b) Path solved with a larger w_g

Overview of the Scheme

Termination Operation



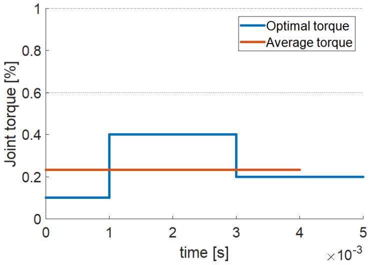
Start
Stop
Termination operation
Y
N
1 - s < tolerance ?
Ball trajectory estimation
Catching point determination
Calculating control inputs
Evaluating m, c, g, at current s and current states
Lower level feedback control
Linear interpolation of b
Deviation > tolerance ?
N
Y
Updating current states
Calculating reference
Inverse Kinematics
N
Estimating current s, i = i+1
i > N ?
Y
Geometric path generation. Initial arc length s is set to 0
Evaluating m, c, g, at current s and current states
Solving the optimal control problem from current s to 1
Calculating number of execution N, set iteration counter i=0

Overview of the Scheme

Termination Operation

As mentioned in the title of this project, a shrinking horizon is used. It means the distance between collocation points becomes smaller while the robot is approaching the catch point.

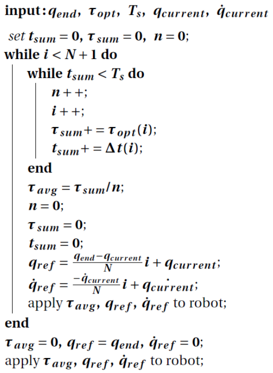
- The time interval will be shorter than the sample period around the catch point
- The termination operation is used to control the robot moving to the catch point blindly using control history computed in the last iteration
- The basic idea is to take the average optimal torque of adjacent time intervals



Overview of the Scheme

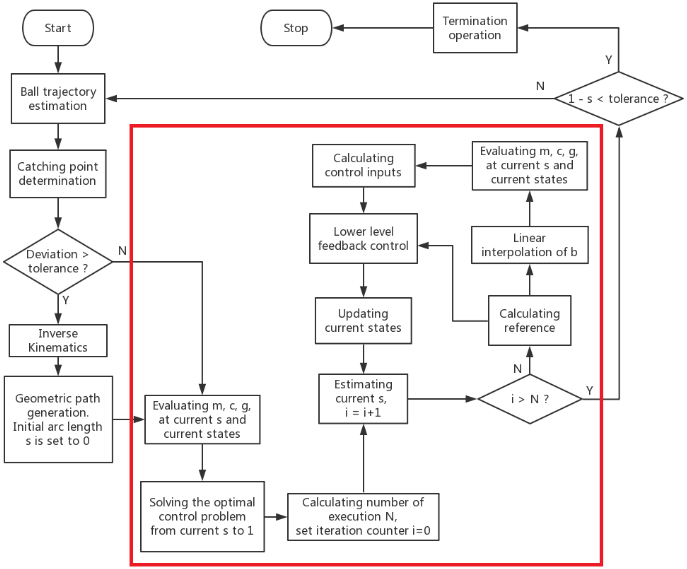
Termination Operation

The termination operation is described in the following algorithm:



```
input: $q_{end},\ \tau_{opt},\ T_s,\ q_{current},\ \dot{q}_{current}$
set $t_{sum} = 0,\ \tau_{sum} = 0,\ n = 0$;
while $i < N+1$ do
    while $t_{sum} < T_s$ do
        $n++$;
        $i++$;
        $\tau_{sum} += \tau_{opt}(i)$;
        $t_{sum} += \Delta t(i)$;
    end
    $\tau_{avg} = \tau_{sum}/n$;
    $n = 0$;
    $\tau_{sum} = 0$;
    $t_{sum} = 0$;
    $q_{ref} = \frac{q_{end} - q_{current}}{N} i + q_{current}$;
    $\dot{q}_{ref} = \frac{-\dot{q}_{current}}{N} i + \dot{q}_{current}$;
    apply $\tau_{avg},\ q_{ref},\ \dot{q}_{ref}$ to robot;
end
$\tau_{avg} = 0,\ q_{ref} = q_{end},\ \dot{q}_{ref} = 0$;
apply $\tau_{avg},\ q_{ref},\ \dot{q}_{ref}$ to robot;
```

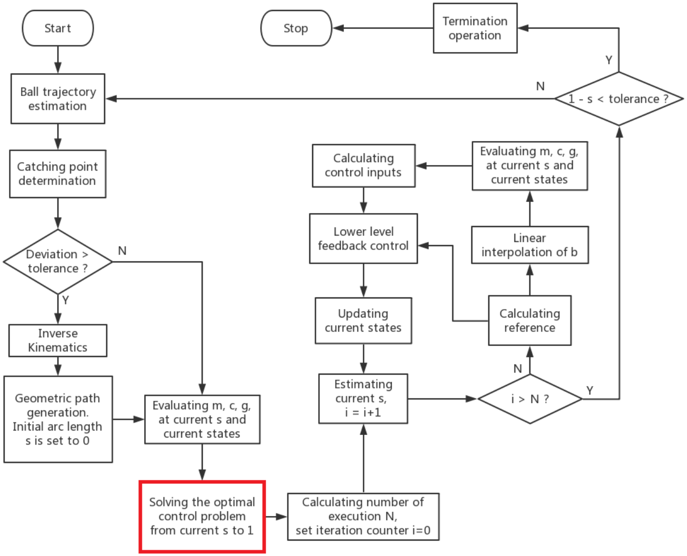
Solving the Optimal Control Problem



Start
Stop
Termination operation
Y
N
1 - s < tolerance ?
Ball trajectory estimation
Catching point determination
Calculating control inputs
Evaluating m, c, g, at current s and current states
Lower level feedback control
Linear interpolation of b
N
Deviation > tolerance ?
Y
Inverse Kinematics
Updating current states
Calculating reference
N
Estimating current s, i = i+1
i > N ?
Y
Geometric path generation. Initial arc length s is set to 0
Evaluating m, c, g, at current s and current states
Solving the optimal control problem from current s to 1
Calculating number of execution N, set iteration counter i=0

Solving the Optimal Control Problem

Reformulating the Problem



Start
Stop
Termination operation
Ball trajectory estimation
Y
N
1 - s < tolerance ?
Catching point determination
Calculating control inputs
Evaluating m, c, g, at current s and current states
Lower level feedback control
Linear interpolation of b
Deviation > tolerance ?
N
Y
Updating current states
Calculating reference
Inverse Kinematics
N
Geometric path generation. Initial arc length s is set to 0
Evaluating m, c, g, at current s and current states
Estimating current s, i = i+1
i > N ?
Y
Solving the optimal control problem from current s to 1
Calculating number of execution N, set iteration counter i=0

Solveing the Optimal Control Problem

Reformulating the Problem

Solving the problem in time domain directly has some disadvantages:

- The inertia matrix $M(q)$ in system dynamics has to be inverted
- When DOF of the robot increases, the dimension of the decision variable also increases
- It's hard to find the global optimizer

To remedy them,

- the optimal control problem is first reformulated into path parameter domain using decoupled approach
- It is then turned into a convex problem through a nonlinear transformation

Solving the Optimal Control Problem

Reformulating the Problem

Let $q(s)$ be the path to be followed by the robot, $s \in [0,1]$. Using the chain rule:

$$\begin{aligned}
\min_{\tau,s} \quad & \int_0^1 \sum_{i=1}^{6} \frac{\tau_i^2(s)}{\tau_{max,i}^2} \frac{1}{\dot{s}} ds \\
s.t.\ \tau \; = \; & m(s)\ddot{s} + c(s)\dot{s}^2 + g(s) \\
& 0 \leq T \leq T_{catch} \\
& \tau_{min,i} \leq \tau_i \leq \tau_{max,i} \qquad i = 1, 2, ..., 6 \\
& s(0) = 0 \qquad s(T) = 1 \\
& \dot{s}(0) = \dot{s}_0 \qquad \dot{s}(t) \geq 0
\end{aligned}$$

where

$$\begin{aligned}
m(s) &= M(q(s))q'(s) \\
c(s) &= M(q(s))q''(s) + C(q(s), q'(s))q'(s) \\
g(s) &= G(q(s)) + F_s sgn(q'(s)) \\
T &= \int_0^1 \frac{1}{\dot{s}} ds
\end{aligned}$$

Solving the Optimal Control Problem

Reformulating the Problem

By introducing the nonlinear transformation:

$$\dot{s}^2 = b$$
$$\ddot{s} = a$$

the above problem could be reformulated as:

$$\begin{aligned}
\min_{\tau,a,b} \quad & \int_0^1 \sum_{i=1}^{6} \frac{\tau_i^2(s)}{\tau_{max,i}^2} \frac{1}{\sqrt{b(s)}} ds \\
s.t.\ \tau \;=\; & m(s)a + c(s)b + g(s) \\
& 0 \leq T \leq T_{catch} \\
& \tau_{min,i} \leq \tau_i \leq \tau_{max,i} \qquad i = 1, 2, ..., 6 \\
& b(0) = b_{init} \\
& b'(s) = 2a'(s) \\
& b(s) \geq 0
\end{aligned}$$

This problem is convex because 1) the cost function is convex (integration preserves convexity) and 2) the constraints are linear

Solving the Optimal Control Problem

Reformulating the Problem

To solve the new problem, the collocation method is used.

- Collocation points are equally spaced: $s_i = \frac{i}{N} \quad i = 0, 1, 2, ..., N$
- $a(s)$ is the s-domain counterpart of control input in time domain
- $b(s)$ and $\tau(s)$ corresponds to system states.

After discretization, the NLP to be solved is:

$$\min_{\tau,a,b} \sum_{i=0}^{N-1} \sum_{j=1}^{6} \frac{\tau_j^2(s_{i+\frac{1}{2}})}{\tau_{j,max}^2} \frac{2}{\sqrt{b_{i+1}} + \sqrt{b_i}} \Delta s_i$$

$$s.t. \ \tau(s_{i+\frac{1}{2}}) = m((s_{i+\frac{1}{2}})a_i + c((s_{i+\frac{1}{2}})\frac{b_i + b_{i+1}}{2} + g((s_{i+\frac{1}{2}})$$

$$\tau_{min} \leq \tau \leq \tau_{max}$$

$$b_0 = b_{init}$$

$$b_{i+1} - b_i = 2a_i \Delta s$$

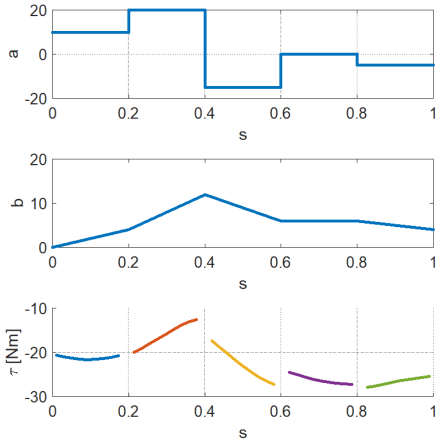
$$b(s) \geq 0$$

$$0 \leq T = \sum_{i=0}^{N} \frac{2}{\sqrt{b_{i+1}} + \sqrt{b_i}} \Delta s_i \leq T_{catch}$$

Solving the Optimal Control Problem

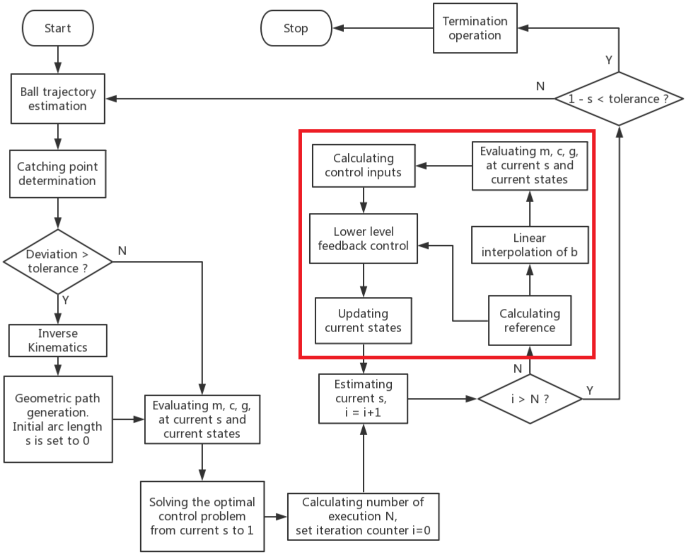
Reformulating the Problem

Since $a(s)$ is regarded as control input, it is piecewise constant. From the new system dynamics $b_{i+1} - b_i = 2a_i\Delta s$, $b(s)$ is piecewise constant. And $\tau(s)$ is piecewise nonlinear.



Solving the Optimal Control Problem

Online Feedback and Linear Interpolation



Start
Stop
Termination operation
Y
N
1 - s < tolerance ?
Ball trajectory estimation
Catching point determination
Calculating control inputs
Evaluating m, c, g, at current s and current states
Lower level feedback control
Linear interpolation of b
Deviation > tolerance ?
N
Y
Updating current states
Calculating reference
Inverse Kinematics
N
Estimating current s, i = i+1
i > N ?
Y
Geometric path generation. Initial arc length s is set to 0
Evaluating m, c, g, at current s and current states
Solving the optimal control problem from current s to 1
Calculating number of execution N, set iteration counter i=0

Solving the Optimal Control Problem

Online Feedback and Linear Interpolation

By solving the above problem, a precomputed control history could be found. However, applying it blindly (an open-loop approach) is not a good idea because:

- the catch point might change during the operation
- the later estimation of ball trajectory is better than the former one (in terms of uncertainty)
- errors in system model (model-plant-mismatch) can cause deviation from expected trajectory

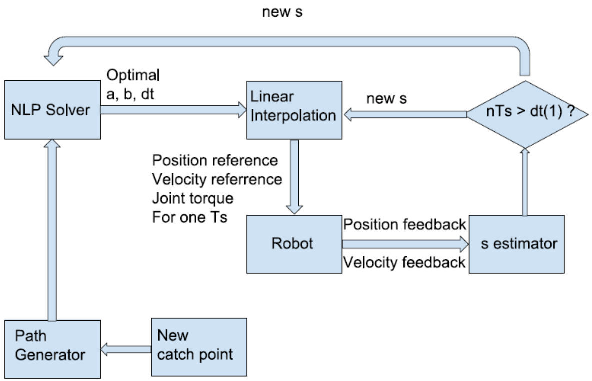
To remedy this problem, the NLP should be solved online.

- But the number of collocation points N is limited by the high sample frequency ($f_s = 250Hz$).
- A linear interpolation method is used to recover nonlinear optimal torques from coarse grid solution

Solving the Optimal Control Problem

Online Feedback and Linear Interpolation

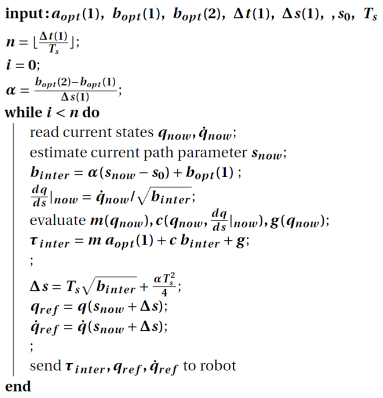
The online feedback scheme is shown below:



Solving the Optimal Control Problem

Online Feedback and Linear Interpolation

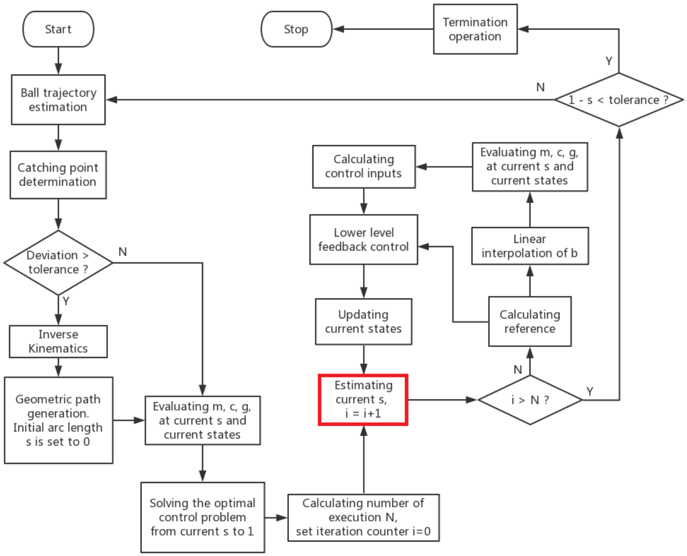
The linear interpolation is summarized in the following algorithm:



```
input: $a_{opt}(1),\ b_{opt}(1),\ b_{opt}(2),\ \Delta t(1),\ \Delta s(1),\ ,s_0,\ T_s$
$n = \lfloor \frac{\Delta t(1)}{T_s} \rfloor$;
$i = 0$;
$\alpha = \frac{b_{opt}(2) - b_{opt}(1)}{\Delta s(1)}$;
while $i < n$ do
    read current states $q_{now}, \dot{q}_{now}$;
    estimate current path parameter $s_{now}$;
    $b_{inter} = \alpha(s_{now} - s_0) + b_{opt}(1)$ ;
    $\frac{dq}{ds}|_{now} = \dot{q}_{now} / \sqrt{b_{inter}}$;
    evaluate $m(q_{now}), c(q_{now}, \frac{dq}{ds}|_{now}), g(q_{now})$;
    $\tau_{inter} = m\, a_{opt}(1) + c\, b_{inter} + g$;
    ;
    $\Delta s = T_s \sqrt{b_{inter}} + \frac{\alpha T_s^2}{4}$;
    $q_{ref} = q(s_{now} + \Delta s)$;
    $\dot{q}_{ref} = \dot{q}(s_{now} + \Delta s)$;
    ;
    send $\tau_{inter}, q_{ref}, \dot{q}_{ref}$ to robot
end
```

Solving the Optimal Control Problem

Path Parameter Estimation



Start
Stop
Termination operation
Y
N
1 - s < tolerance ?
Ball trajectory estimation
Catching point determination
Deviation > tolerance ?
N
Y
Inverse Kinematics
Geometric path generation. Initial arc length s is set to 0
Evaluating m, c, g, at current s and current states
Solving the optimal control problem from current s to 1
Calculating number of execution N, set iteration counter i=0
Estimating current s, i = i+1
i > N ?
N
Y
Calculating reference
Linear interpolation of b
Evaluating m, c, g, at current s and current states
Calculating control inputs
Lower level feedback control
Updating current states

Solving the Optimal Control Problem

Path Parameter Estimation

There are 2 feedback path of the robot states:

- by evaluating system dynamics $m(q)$, $c(q, q')$, $g(q)$ at current state
- by estimating the corresponding path parameter s_{now}

The second path requires a path parameter estimator. Two estimators, one for continuous path and the other for discrete path, are described below:

- For continuous path, solving the following problem:

$$\begin{aligned} \min_{s} \quad & ||q(s) - q_{now}||_2^2 + \gamma||q'(s) - \frac{dq}{ds}|_{now}||_2^2 \\ s.t. \quad & s_{old} - \delta s \leq s \leq s_{old} + \delta s \\ & 0 \leq s \leq 1 \end{aligned}$$

Solving the Optimal Control Problem

Path Parameter Estimation

- For discrete path, the problem is trickier because the robot moves inside the gap in the discrete trajectory during the linear interpolation. A 2-step strategy is developed:
  - First, find the closet points in the trajectory by solving

    $$\min_{s_i} \quad ||q_i - q_{now}||_2^2$$

    q_i and s_i denote the ith point in the trajectory and the corresponding path parameter respectively. The result is called q_i* and s_i*
  - Next, linearize the trajectory in the vicinity of (s_i*, q_i*) to get a continuous straight line $q_l(s)$ and conduct search on it

    $$\begin{aligned} q_l(s) = & \frac{q_{i+1} - qi - 1}{s_{i+1} - s_{i-1}}(s - s_i*) + q_i* \\ \min_{s} \quad & ||q_l(s) - q_{now}||_2^2 \\ s.t. \quad & s_{i-1} \leq s \leq s_{i+1} \\ & 1 \leq s \leq N_t \end{aligned}$$

    The result is called $s*$

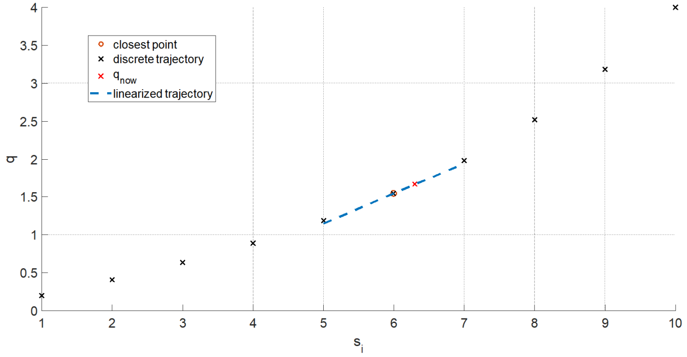
Solving the Optimal Control Problem

Path Parameter Estimation

- To use $s*$ in the feedback, the trajectory is linear interpolated around s_i*

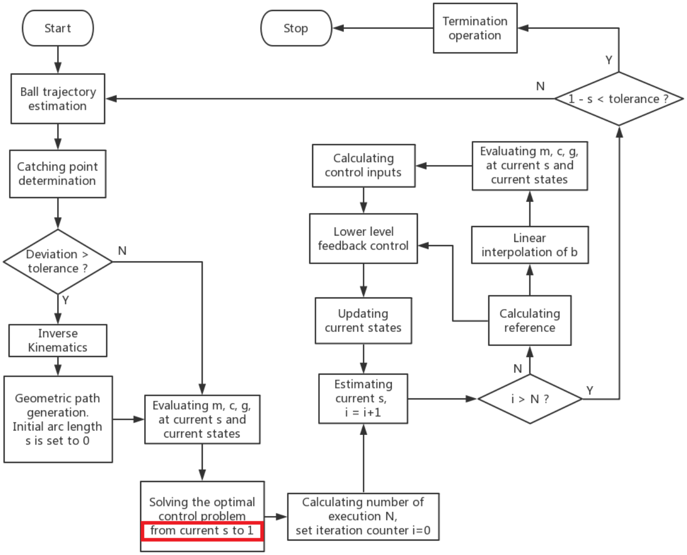
$$q* = \frac{q_{i+1} - qi - 1}{si + 1 - si - 1}(s * - s_i*) + q_i*$$

This scheme is illustrated in figure below:



Solving the Optimal Control Problem

Shrinking Horizon



Start
Stop
Termination operation
Y
N
1 - s < tolerance ?
Ball trajectory estimation
Catching point determination
Deviation > tolerance ?
N
Y
Inverse Kinematics
Geometric path generation. Initial arc length s is set to 0
Evaluating m, c, g, at current s and current states
Solving the optimal control problem from current s to 1
Calculating number of execution N, set iteration counter i=0
Estimating current s, i = i+1
i > N ?
N
Y
Calculating reference
Linear interpolation of b
Evaluating m, c, g, at current s and current states
Calculating control inputs
Lower level feedback control
Updating current states

Solving the Optimal Control Problem

Shrinking Horizon

- The robot's movement doesn't have to be very accurate at the beginning. High precision is only needed around the catch point
- A coarse grid is used at the beginning for faster computation
- The grid is refined while the robot approaching the catch point
- To have such a behaviour, modify the cost function as:

$$\int_{s_{now}}^{1} \sum_{i=1}^{6} \frac{\tau_i^2(s)}{\tau_{max,i}^2} \frac{1}{\sqrt{b(s)}} ds$$

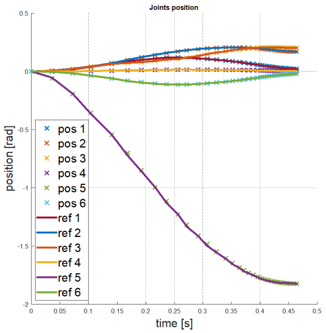
The lower limit of integration is changed into s_{now} instead of 0. And the grid size is:

$$\Delta s = \frac{1 - s_{now}}{N - 1}$$

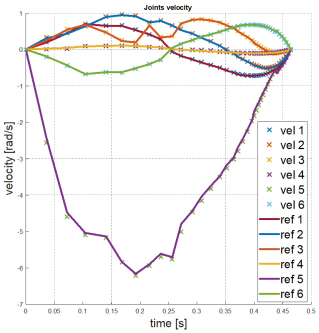
Simulation Results and Discussion

Results with a changing catch point

- The following results are obtained using CasADi
- The proposed scheme could track a changing catch point



(a) Result:position, changing catch point



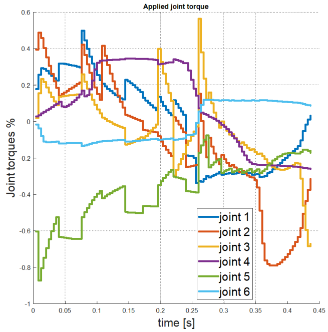
(b) Result:velocity, changing catch point

Figure: Simulation result with a changing catch point

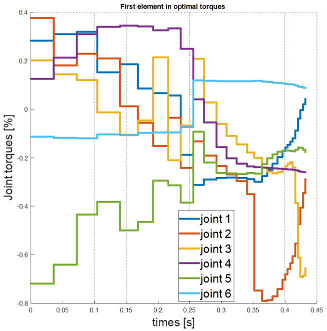
Simulation Results and Discussion

Results with a changing catch point

- The linear interpolation recovered the nonlinear optimal joint torque as expected
- Shrinking horizons could be seen in the figure



(a) Result:torque from linear interpolation



(b) Result:torque from NLP directly

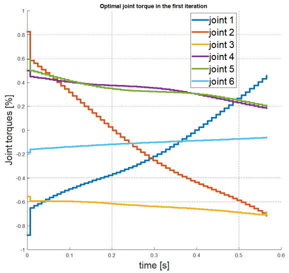
Figure: Simulation result with a changing catch point

Simulation Results and Discussion

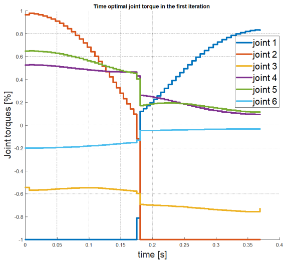
Comparison of Thermal Energy

Table: Comparison of two optimal control

	Time optimal	Energy optimal
Duration [s]	0.376	0.575
Energy (normalized) [s]	0.8216	0.5589



(a) Thermal energy optimal joint torques



(b) Time optimal joint torques

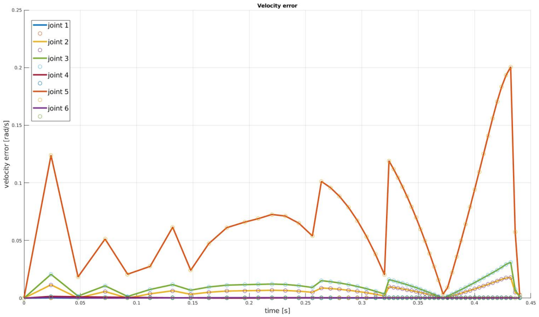
Figure: Comparison between two optimal control problems

Simulation Results and Discussion

Tracking Error

Because of the decreasing grid size Δs, we expect a decreasing tracking error. However, this is not the case, as shown in figure below. Noticeable features are:

- a large error at the beginning
- some decreasing 'period'
- monotonically increasing errors at the end



Simulation Results and Discussion

Tracking Error

This error pattern is caused by mixed mechanisms:

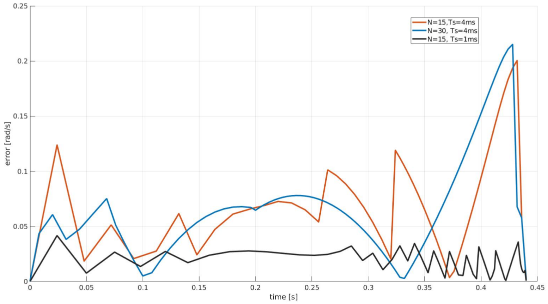
- coarse grid
- finite sample frequency
- termination operation
- numerical error

Simulation Results and Discussion

Tracking Error - Coarse grid & finite sample frequency

Figure below shows the velocity error of the 5th joint under three different settings: N=15, T_s=4ms; N=30, T_s=4ms and N=15, T_s=1ms

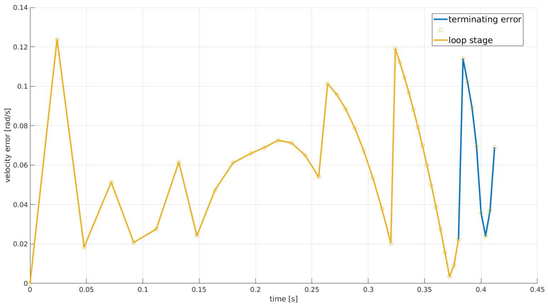
- A refined grid results in smaller tracking error
- The 'period' is caused by finite sample frequency. Because the time interval we applied, $\Delta t(1)$, is not always a integer multiple of sample period T_s. That's why a smaller sample period reduces the error 'period' significantly (black curve).



Simulation Results and Discussion

Tracking Error - termination operation

The monotonically increasing part is caused by late termination operation. In this part, the time interval $\Delta t(1)$ is smaller than sample period T_s and it's still shrinking. But we have to apply constant torque within one sample period. Therefore the error is increasing monotonically. Changing the tolerance that triggers the termination operation could reduce this error.

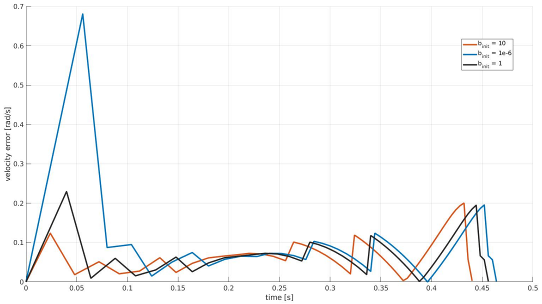


Simulation Results and Discussion

Tracking Error - numerical error

There is a large error at the first step.

- b_{init}, the initial value of b, has strong influence on this error
- A larger b_{init} results in smaller error



Simulation Results and Discussion

Tracking Error - numerical error

- One possible reason is related to how $\sqrt{b_{k+\frac{1}{2}}}$ is evaluated:

$$\sqrt{b_{k+\frac{1}{2}}} = \frac{\sqrt{b_k} + \sqrt{b_{k+1}}}{2}$$

- If we take a look at the Taylor expansion of $\sqrt{b}$ around b_k:

$$\sqrt{b} = \sqrt{b_k} + \frac{1}{2}\frac{1}{2\sqrt{b_k}}(b - b_k) - \frac{1}{6}\frac{1}{4\sqrt{b_k^3}}(b - b_k)^2 + o((b - b_k)^2)$$

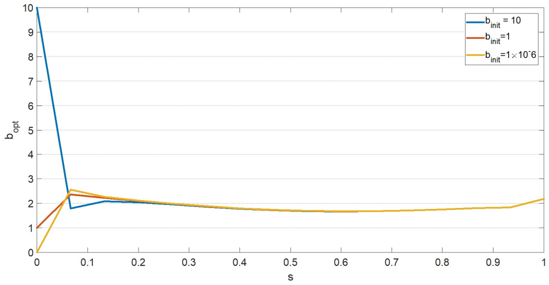
  it could be found that the nonlinear part (the numerical error) becomes smaller if b_k is larger
- How is this numerical error related to the tracking error?
  - The time interval $\Delta t(1)$, during which we apply the optimal torque, becomes longer than expected (it's longer because the square root is a concave function) and that results in the large overshoot.

$$\Delta t = \frac{\Delta s_k}{\sqrt{b_{k+\frac{1}{2}}}}$$

Simulation Results and Discussion

Tracking Error - numerical error

- But why does error only influence the first step?
  - Because the change of optimal $b(s)$ in the rest part is small. Since $\sqrt{b_{\frac{1}{2}}}$ must locates somewhere between $\sqrt{b_k}$ and $\sqrt{b_{k+1}}$, the error between $\frac{\sqrt{b_k}+\sqrt{b_{k+1}}}{2}$ and $\sqrt{b_{\frac{1}{2}}}$ is also very small
- $b_{opt}(s)$ converge to the same value because the NLP is convex

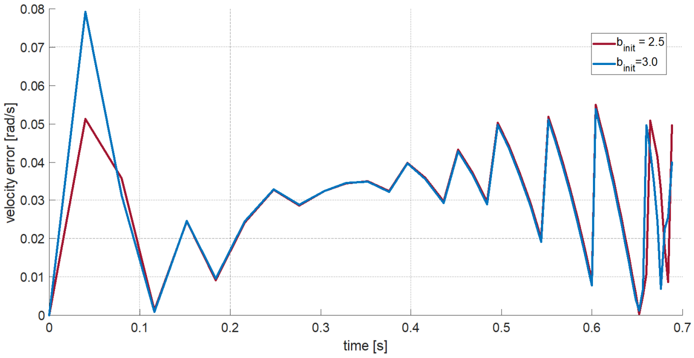


Simulation Results and Discussion

Tracking Error - numerical error

This argument could be verified by the following result:

- the $b_{opt}(2)$ is about 2. Therefore, when b_{init} is set to 2.5, we expect a smaller error than $b_{init} = 3$ because the difference between b_{init} and $b_{opt}(2)$ is smaller. And the result shown in figure below behaves as expected



Simulation Results and Discussion

Feasibility

There are 2 constraints that result in infeasible problems:

- Torque constraints:

$$\tau_{min} \leq \tau = m(s_{k+\frac{1}{2}})a_k + c(s_{k+\frac{1}{2}})b_{k+\frac{1}{2}} + g(s_{k+\frac{1}{2}}) \leq \tau_{max}$$

$$a_k = \frac{b_{k+1} - b_k}{2\Delta s} \ , \quad b_{k+\frac{1}{2}} = \frac{b_{k+1} + b_k}{2}$$

- Time constraint:

$$T = \sum_{k=1}^{N} \frac{2\Delta s}{\sqrt{b_{k+1}} + \sqrt{b_k}} \leq T_{remain}$$

Simulation Results and Discussion

Feasibility - Torque constraints

- For joint i, torque constraint could be written as:

$$\frac{\tau_{i,min} - g(s_{k+\frac{1}{2}}) - c(s_{k+\frac{1}{2}})b_{k+\frac{1}{2}}}{m_i(s_{k+\frac{1}{2}})} \leq a_k \leq \frac{\tau_{i,max} - g(s_{k+\frac{1}{2}}) - c(s_{k+\frac{1}{2}})b_{k+\frac{1}{2}}}{m_i(s_{k+\frac{1}{2}})}$$

- It seems that we could always find an a_k which falls in this range and the torque constraints won't be violated.
- But there is another constraint:

$$a_k = \frac{b_{k+1} - b_k}{2\Delta s}, \quad b_k \geq 0$$

- Since b_k can't be negative, the following inequality must hold:

$$\frac{-b_k}{2\Delta s} \leq \frac{\tau_{i,max} - g(s_{k+\frac{1}{2}}) - c(s_{k+\frac{1}{2}})b_{k+\frac{1}{2}}}{m_i(s_{k+\frac{1}{2}})}$$

Simulation Results and Discussion

Feasibility - Torque constraints

The above inequality is violated when:

- $\frac{dq}{ds}$ is too large. Because $c(q(s), q'(s))$ is a quadratic function of $\frac{dq}{ds}$, sharp corners in the path might cause infeasible problem
- the robot is around the outer edge of its workspace because the gravity term $g_{k+\frac{1}{2}}$ becomes larger due to larger radius
- the remaining time is too short or the remaining path is too long. b_k has to be large in these situations.

Simulation Results and Discussion

Feasibility - Torque constraints

- From the inequality $\frac{-b_k}{2\Delta s} \leq \frac{\tau_{i,max} - g(s_{k+\frac{1}{2}}) - c(s_{k+\frac{1}{2}}) b_{k+\frac{1}{2}}}{m_i(s_{k+\frac{1}{2}})}$, it seems a smaller Δs could make infeasible problem feasible.
  - This is true. For example, let the ball's initial states be $pos_{init} = [2, 0, 0]^T$, $vel_{init} = [-3.15, 0, 2.86]^T$. For N = 15, $\Delta s = \frac{1}{N+1} = 0.0625$, the problem is infeasible; but for N = 50, $\Delta s = \frac{1}{N+1} = 0.020$, the problem becomes feasible.
  - But reducing Δs is limited by computation cost and sample period
- b_{init} also has impact on feasibility. The above inequality could be rewritten as

$$\left(\frac{c_{1+\frac{1}{2}}}{2} + \frac{m_{1+\frac{1}{2}}}{2\Delta s}\right) b_2 \; + \; \left(\frac{c_{1+\frac{1}{2}}}{2} - \frac{m_{1+\frac{1}{2}}}{2\Delta s}\right) b_{init} \; \leq \; \tau_{max} - g_{1+\frac{1}{2}}$$

  - If b_{init} is too large, there won't be a positive b_2 that satisfies this inequality
  - For a smaller tracking error, a larger b_{init} is desired. But the torque constraints set an upper limit for it.

Simulation Results and Discussion

Feasibility - Time constraints

- From a batch test using 100 random catch point, the time constraint is usually inactive.
  - This result implies a slow trajectory doesn't necessarily have the smallest energy.
  - The cost function explains it:

$$E = \sum_{k=1}^{N} \sum_{i=0}^{6} \frac{\tau_i^2(k)}{\tau_{i,\ max}^2} \frac{2\Delta s}{\sqrt{b_{k+1}} + \sqrt{b_k}} = \sum_{k=1}^{N} \sum_{i=0}^{6} \frac{\tau_i^2(k)}{\tau_{i,\ max}^2} \Delta t_k$$

- b_{init} also plays a role here. For example: $pos_{init} = [2, 0, 0]^T$ and $vel_{init} = [-3.15, 0, 2.86]^T$. If $b_{init} = 1 \times 10^-6$, it's infeasible. If $b_{init} = 10$, it's feasible.
  - Therefore, if the torque constraint is the limiting factor, we should lower b_{init}; if the time constraint is active, we should elevate it.

Simulation Results and Discussion

Feasibility changes during operation

- Sometimes, with a fixed catch point, the NLP is feasible at the beginning but becomes infeasible after a few steps. This problem often happens in time optimal control.
- It is caused by torque constraints violation.
  - In collocation method, constraints are only imposed at collocation points.
  - But, as mentioned above, τ is piecewise nonlinear. When torque constraint is almost active at collocation points, there are already constraint violation. And the linear interpolation will recover this violation and send it to the robot.
  - The motor is saturated and the robot is slower than the reference. This in turn causes more torque violation. A positive feedback is established.
  - After a few steps, the problem becomes infeasible.

Conclusion and Future Work

- An energy optimal ball catching scheme is proposed. Simulation results showed different factors that influence the tracking accuracy and feasibility. The initial value of $b(s)$ has influence on both subjects. The collocation method might cause constraint violation and eventually result in infeasible problems
- A vision system needs to be developed for ball trajectory estimation. The proposed method is still too slow for real-time applications. Possible solutions are using more efficient solvers, designing a linear predictor and parallelizing the proposed scheme.