18.086 Project Report: A linear programming approach for dynamic system control with inequality constraints

Lei Zhou

May 15, 2014

1 Introduction

The presence of inequality constraint on the state variables and control effort is a distinctive feature of many practical control problems. In control of mechanical servo systems, the limitation of the control effort may comes from the physical limitation as the actuator's maximum ability, and the limitation of state variables may comes from the control objectives. For example, when controlling the motion of a machine tool, the position accuracy of the tool should be limited in certain range to avoid damage the part being cutting. Another example will be when a robot manipulating, the robot arm should be remained in certain range to avoid collision. Further more, these constraints can change with time in an unpredictable manner due to the change of the environment, equipment availability, or a shift in control objectives. Thus control in the presence of constraints is important and interesting from a practical point of view.

Linear programming (LP) has proved to be a powerful method to solve optimal control problems for linear systems with linear equality and inequality constraints. Significant attention has been paid to the classical time-optimal and energy-consumption-optimal problems. [1] have proposed resolving the time-optimal control problem on-line at each sample instant to generate a feedback control strategy. [2] gives a comprehensive introduction on the linear programming applying on optimal control.

In this project, a feedback control strategy is presented for multi-variable control problem with inequality constraints on the states and input control variables. At each sample instant, a linear programming problem is solved in real-time to determine the optimal value of the control input with minimize a linear performance index while satisfying the inequality constraints. The potential advantages of this linear programming based control strategy are:

- It provide a systematic, flexible way for control problems with large numbers of variables and constraints.

- Efficient computational techniques, such as the Simplex method, are available for solving linear programming problems.
- The system model, constraints and objective function can be easily modified in real time control if needed, thus it has the potential to deal with some nonlinear dynamic systems.

On the other hand, this linear programming based optimal control strategy have the potential disadvantages:

- The system model, performance measuring index, and constraints must be linear in order to be able to change into a standard linear programming problem.
- Since an LP problem will be solved at every time step, thus the computer requirements must be carefully evaluated. Also, this time of computation limits the sampling rate of the digital controller, thus will not be suitable for the systems with fast dynamics.

2 Problem formulation and control strategy

This project considers a linear, time-invariant (LTI) dynamic system state space model. In continuous time, the system can be written as:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} + \mathbf{G}\mathbf{d} \tag{1}$$

where $\mathbf{x}$ is the $n \times 1$ state vector. $\mathbf{u}$ is the $r \times 1$ control input vector, where $r \leq n$. $\mathbf{d}$ is the vector of the unknown disturbance.

In order to use a digital controller, the system needed to be transformed into a discrete-time state-space model by means of forward Euler approximation. By keeping the sample time as a constant dt, the discrete-time model of the system is

$$\mathbf{x}(k+1) = \mathbf{\Phi}\mathbf{x}(k) + \mathbf{\Delta}\mathbf{u}(k) + \mathbf{\Psi}\mathbf{d}(k) \tag{2}$$

where $\mathbf{\Phi} = \mathbf{I} + \mathbf{A}dt$, $\mathbf{\Delta} = \mathbf{B}dt$, and $\mathbf{\Psi} = \mathbf{G}dt$.

In this problem, the system is subject to inequality constraints. The system's input vector is subjected to the constraint of

$$\alpha \leq \mathbf{u} \leq \beta \tag{3}$$

and the systems state vector is subject to the constraint of

$$\theta \leq \mathbf{x} \leq \lambda \tag{4}$$

where α, β, θ and λ are constant vectors with the proper dimension, and they are specified accordingly to control requirements and the system constraints.

We want to design a control law for the system to make the state vector $\mathbf{x}$ to track a desired trajectory $\mathbf{x_d}$. In the next section, the control policy based on linear programming is presented in detail.

3 LP based control policy

In this section we will use the linear programming approach to determine the control signal $\mathbf{u}(k)$ based on the measurement at the certain sample instant. Here we assume that all the state variables can be measured. If this is not the case, then a state estimator such as Kalman filter [3] can be used to generate an estimation of the full state measurement.

In the linear programming based control approach, the dynamic model in Equation 2 and the current measurement are used to predict the state variables in the next sample step:

$$\hat{\mathbf{x}}(k+1) = \mathbf{\Phi}\mathbf{x}(k) + \mathbf{\Delta}\mathbf{u}(k) \tag{5}$$

where $\hat{\mathbf{x}}(k+1)$ is the predicted values of $\mathbf{x}(k+1)$. Note that in the state estimation there is no disturbance vector, since it is assumed to be unknown.

In order to use linear programming method to determine the control effort, a linear performance index must be select. A reasonable choice is a weighted sum of the l^1 norm of the tracking errors:

$$J = \sum_{i=1,\ldots,n} w_i |\hat{e}_i(k+1)| = \sum_{i=1,\ldots,n} w_i |\hat{x}_i(k+1) - x_{di}(k)| \tag{6}$$

where w_i is the non-negative weighting factor for the i-th state variable. These weighting factors are used to penalize prediction errors in the more important state variables.

We want to use linear programming to minimize the performance index in Equation 8 subject to all the constraints in every time step. In the following of this section, the major task is to transform this problem into a standard linear programming problem.

The standard linear programming formulation requires the unknown variables to be non-negative. In order to transform the objective function 8 to a linear form and to unsure the unknown variables are non-negative, we introduce new variables $x_i^+(k+1)$ and $x_i^-(k+1)$, and their definition are:

$$\hat{e}_i(k+1) = \hat{x}_i(k+1) - x_{di}(k) = x_i^+(k+1) - x_i^-(k+1) \tag{7}$$

By introducing these new variables, the equivalent form of the objective function 8 becomes

$$J = \sum_{i=1,\ldots,n} w_i \hat{e}_i(k+1) = \sum_{i=1,\ldots,n} w_i \left(x_i^+(k+1) - x_i^-(k+1)\right) \tag{8}$$

This approach is commonly used in minimizing l^1 norm of the variables, and is introduced in the textbook [4]. In this way the l^1 norm objective is transformed into a linear combination of the new variables, thus a linear programming approach can be used.

The next step of transforming this control problem into a standard linear programming problem is to deal with the constraints. In a standard linear programming problem,

equality constraint is assumed. We can change the inequality constraints in Equation 3 and 4 into equality constraints by introducing new variables.

First let us use a change of variable to make the constraint on control input $\mathbf{u}$ in Equation 3 into single-side inequality. It is convenient in the LP formulation to express Equation 3 as

$$\mathbf{0} \leq \tilde{\mathbf{u}}(k) \leq \beta - \alpha \tag{9}$$

where

$$\tilde{\mathbf{u}}(k) = \mathbf{u}(k) - \alpha \tag{10}$$

Then we introduce slack and surplus variables to convert the inequality constraints into equality constraints. Since $\tilde{\mathbf{u}}(k)$ has single side inequality constraints, and $\mathbf{x}(k)$ has the constraints on two sides, thus the one slack variable vector and one surplus variable vector are introduced for $\mathbf{x}(k)$, and only one slack variable is introduced for $\tilde{\mathbf{u}}(k)$. By putting all the variables together, in the transformed standard linear programming problem the unknown variable is

$$\mathbf{z} = \begin{bmatrix} \mathbf{x}^+ & \mathbf{x}^- & \tilde{\mathbf{u}} & \mathbf{x}_\mathbf{s}^+ & \mathbf{x}_\mathbf{s}^- & \mathbf{u}_\mathbf{s}^+ \end{bmatrix}^T \tag{11}$$

where the subscript s denotes either slack or surplus variable. Based on this unknown variable defination, the performance index can be written as:

$$\mathbf{J} = \begin{bmatrix} \mathbf{w} & \mathbf{w} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{x}^+ \\ \mathbf{x}^- \\ \tilde{\mathbf{u}} \\ \mathbf{x}_\mathbf{s}^+ \\ \mathbf{x}_\mathbf{s}^- \\ \mathbf{u}_\mathbf{s}^+ \end{bmatrix} \tag{12}$$

and the linear equality constraint in the standard linear programming problem is

$$\begin{bmatrix} \mathbf{I} & -\mathbf{I} & -\mathbf{\Delta} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{I} & -\mathbf{I} & \mathbf{0} & \mathbf{0} & -\mathbf{I} & \mathbf{0} \\ \mathbf{I} & -\mathbf{I} & \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{x}^+ \\ \mathbf{x}^- \\ \tilde{\mathbf{u}} \\ \mathbf{x}_\mathbf{s}^+ \\ \mathbf{x}_\mathbf{s}^- \\ \mathbf{u}_\mathbf{s}^+ \end{bmatrix} = \begin{bmatrix} \mathbf{\Phi}\mathbf{x}(k) - \mathbf{x}_\mathbf{d}(k) + \mathbf{\Delta}\alpha \\ \theta - \mathbf{x}_\mathbf{d}(k) \\ \lambda - \mathbf{x}_\mathbf{d}(k) \\ \beta - \alpha \end{bmatrix} \tag{13}$$

We can write the problem in a more compact form as

$$min\ \mathbf{J} = \mathbf{c}^T\mathbf{z} \quad s.t. \quad \mathbf{Az} = \mathbf{b} \tag{14}$$

Thus the control problem is successfully changed into a linear programming problem, and we can see that the size of the unknown in the linear programming problem is $N = 4n + 2r$, and the number of constraints is $M = 3n + r$. We can also see that the matrix $\mathbf{A}$ is full rank, that is, $rank(\mathbf{A}) = M$. In the next section some implementation details of the linear programming solver is demonstrated.

4 Linear programming

In this project, the simplex method is used to find the optimal solution for the standard linear programming problem we formulated in the previous section. The selection of simplex method over the interior method have two reasons: first, the simplex search is fast, especially when the problem size is small, and this is a major consideration when this control algorithm is used for a real-time control when the computation must finish within one sample time step. Second, the interior point method gives an approximation of the optimal solution, which may cannot make the best use of the control with satisfying the constraint on it.

The idea of the simplex method is very simple: since the feasible set of the problem is a simplex, then simply check the corners can find the optimal point. It moves from corner to corner, decreasing the value of the objective function. As pointed out in the textbook, since the number of corners can grow exponentially with the problem size, the worst case of simplex method could be slow. Luckily in practice the best corner is found quickly.

Let us write the linear programming problem that we are discussing again as

$$min\ \mathbf{J} = \mathbf{c}^T\mathbf{z} \quad s.t. \quad \mathbf{Az} = \mathbf{b} \tag{15}$$

Keeping consistent with the specific problem in this project, let us assume the size of $\mathbf{A}$ is $M \times N$. Since $M < N$, this is an underdetermined linear system. If the matrix $\mathbf{A}$ is full rank, then M vector in the column space of $\mathbf{A}$ can be a set of basis of $\mathbf{A}$. Let us call the matrix formed by the basis vectors $\mathbf{B}$, and the complimentary of B to be $\mathbf{N}$. A feasible point can be found by solving the linear problem $\mathbf{Bz} = \mathbf{b}$, and by changing the vectors in the basis matrix $\mathbf{B}$ we can move to another corner point.

The basic procedures of the simplex method can be summarized as follows: at each iteration, test the optimality condition and stops if the current solution is optimal. If the current solution is not optimal, the algorithm:

1. Chooses one variable, called the entering variable, from the nonbasic variables and adds the corresponding column of the non-basis to the basis.

2. Chooses a variable, called the leaving variable, from the basic variables and removes the corresponding column from the basis.

3. Updates the current solution and the current objective value.

The selection of the entering and the leaving variables is done by solving two linear systems while maintaining the feasibility of the solution. The simplex method code in this project followed the examples given in the [4] and [5].

5 Simulation result

In this section, the LP based control is tested with several different system, they are: (a) a double integrator plant; (b) a N-th order integrator plant, and (c) a two-link robot arm's trajectory control. The first two systems are linear system and are used to test the scalability of the linear programming based control method. The third system is a nonlinear plant, and when using this proposed control method, a linearization is performed at every time step based on the present configuration of the robot arm. This test is to show that this on-line optimization control method is also applicable to nonlinear system (require the system's nonlinear dynamics is "smooth" enough so that it can be linearized at a certain time instant).

5.1 Double integrator plant control

The first system that being used for this simulation to evaluate the LP based control scheme. This is the simplest test case for the control strategy. The dynamic equation of the system is:

$$\frac{d}{dt}\begin{bmatrix} x \\ \dot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} x \\ \dot{x} \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u \tag{16}$$

A physical interpretation of this system might be: controlling the position of a concentrated mass by controlling the force acting on it, that is, $F = u$, and set the mass of the block to be unity. Figure 1 shows a diagram of the system.

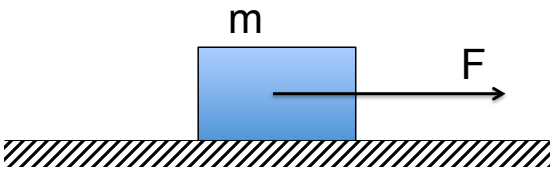


Figure 1: Physical interpretation of double integrator plant: control input: F, inertia of the block: m. No viscous friction is considered in this system.

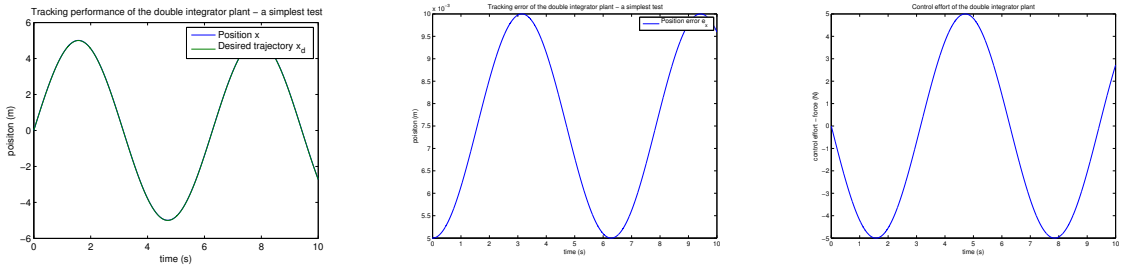
We discretized this system with a sampling time of 1ms. In this problem we want the system to follow the given trajectory of

$$\begin{bmatrix} xd \\ \dot{xd} \end{bmatrix} = \begin{bmatrix} 5 \times sin(t) \\ 5 \times cos(t) \end{bmatrix} \tag{17}$$

The constraints of the system are

$$-6 \leq x_1 \leq 6 \tag{18a}$$
$$-50 \leq x_2 \leq 50 \tag{18b}$$
$$-u_{max} \leq u \leq u_{max} \tag{18c}$$
$$\tag{18d}$$

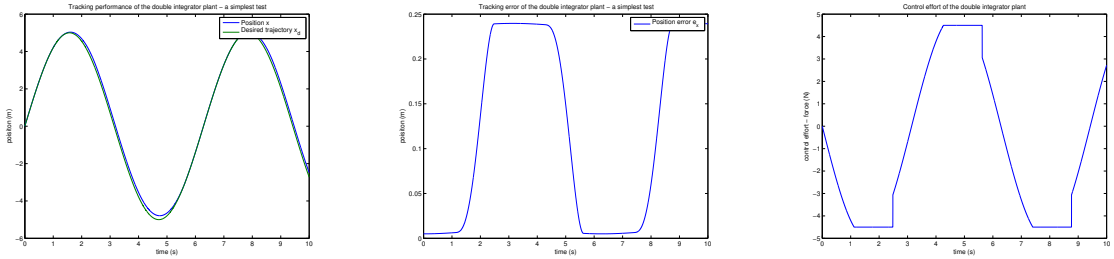
Here we set the limit on the control input to be a variable u_{max}. A simulation of this plant using the proposed linear programming is performed. First we set the limit on control effort $u_{max} = 10$. Since in steady state the needed u is smaller than this, this constraint basically means there is no constraint on the input of the system. Figure 2 shows the tracking performance and corresponding control input.



(a)Tracking performance. (b) Tracking error. (c) Control input u.

Figure 2: Tracking performance of the double integrator plant with $u_{max} = 10$.

To add some limit to the control input, we then set the limit on control input as $u_{max} = 4.5$. Figure 3 shows a tracking performance of the system and the corresponding control input signal u.

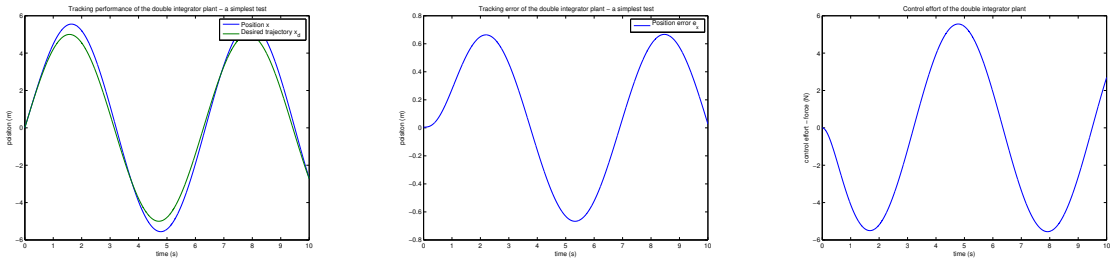


(a)Tracking performance. (b) Tracking error. (c) Control input u.

Figure 3: Tracking performance of the double integrator plant with $u_{max} = 4.5$.

We can see that in Figure **??** the control input signal is no longer linear, and the upper and lower limit of the control input signal shows that the linear programming based method can effectively limit the control effort within the limit.

A simulation of this system is also performed using a very commonly used optimal feedback controller based on a quadratic performance index, often known as linear quadratic regulator (LQR) control [6]. Figure 4 shows a tracking performance and the control input signal u. The optimal controller is designed with penalty weight on the two state variables are 10 and 1 respectively, and the penalty coefficient on the control input is 0.1. The calculated feedback gain by means of LQR is $K = [105.5]$, and the corresponding control input is $u(k) = -K(\mathbf{x}(k) - \mathbf{x_d}(k))$. The simulation result shows that significant larger control effort is used while the tracking error is worth than the on-line optimization method. Even when a hard constraint is given to the control output, like the simulation with $u_{max} = 4.5$, the tracking performance of the LP control is still better then the feedback control method. However, it need to point out that this feedback control algorithm takes significantly less real-time computation then the LP method.



(a)Tracking performance. (b) Tracking error (c) control input u.

Figure 4: Tracking performance of the double integrator plant with LQR control.

5.2 N-th order integrator plant control

In order to study the performance of the scalability of this control strategy, a n-th order integrator system is used to test this control algorithm. The plant's dynamics equation is

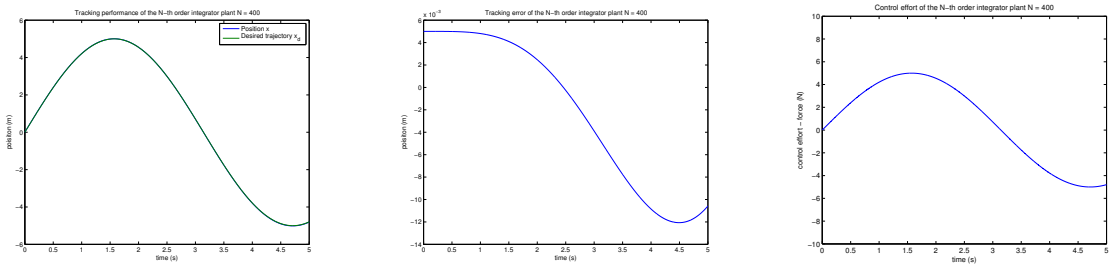
$$x^{(n)} = u \tag{19}$$

Transform this system into a state space form, the the system's dynamic equation is

$$\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} u \tag{20}$$

Again, we discretize the system with a sampling time of 1 ms, and the control goal of the system is also to make the variable of interest x_1 to track the desired trajectory $x_d = 5 \times sin(t)$.

First we show the tracking performance of this n-th order system. In this test case the dimension of the linear system is set to be $n = 400$, thus the dimension of the standard linear programming problem has a size of $N = 4n + 2r = 1602$. Setting the constraint of the state variable to be very loose, and define the limitation on the control input similar as that of the double integrator system. In the first test case we put the limitation as $u_max = 10$, which means there is no limitation when the system is in steady state. The simulation result is presented in Figure 5.

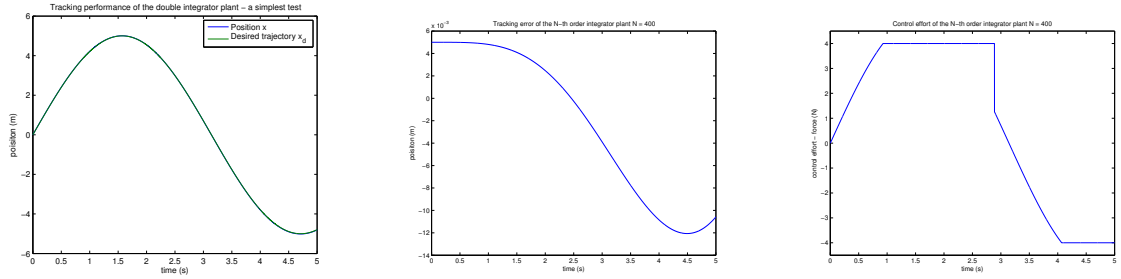


(a)Tracking performance (b) Tracking error (c) Control input

Figure 5: Tracking performance of the 400-th order integrator plant with $u_{max} = 10$.

In this simulation, we can see that the maximum tracking error is 12×10^3, which is good comparing with a control design with a reasonable feedback gain.

Similar to the double integrator system, in order to add some constraint to the system, we set the limit on control input as $u_{max} = 4$. The order of the linear dynamic system is kept as $n = 400$. Figure 6 shows a tracking performance of the system, where



(a)Tracking performance (b) Tracking error (c) Control input

Figure 6: Tracking performance of the 400-th order integrator plant with $u_{max} = 4$.

The signal in Figure 6 (c) shows that the LP based control successfully limited the control variable. The tracking error hasn't changed much. This experiment proofs the effectiveness of the LP based control strategy for high-dimension system.

Another test based on this system is the run time of the computation at a single sample instant. We measured the runtime of the simulation of 10 sample instants and then take the average to get the runtime of one linear programming problem, and used the size of the linear dynamic system as n = 50, 100, 150, ..., 1000. The corresponding linear programming problem will have a size of $N = 4n + 2r$, and for this test system $r = 1$. Figure 7 plots the runtime of one sample instant over the size of the linear programming problem N, where the blue line shows the simulation time using the Matlab internal function 'linprog', and the green line shows the simulation time using a self-implemented simplex function.

The simulation shows that the average runtime of the linear programming is approximately linear with the problem size. However, when this linear programming based control strategy is really used with a actual system, the worst case must be considered because the computation has to be done with in one time step. Thus the sample rate of the system might be limited by the computation time. This is a major challenge to this proposed control strategy, which may make it not suitable for problems with a large size.

5.3 Two-link robot arm system control

The last plant that is used to test this control method is more close to a real dynamic system. We consider the position control of a two-link robot arm in the horizontal plane (then the gravity of the linkages can be ignored). Figure 8 shows a diagram of the system being controlled.

Neglecting the viscous friction at the two joints, the dynamic equation of the system can be written as

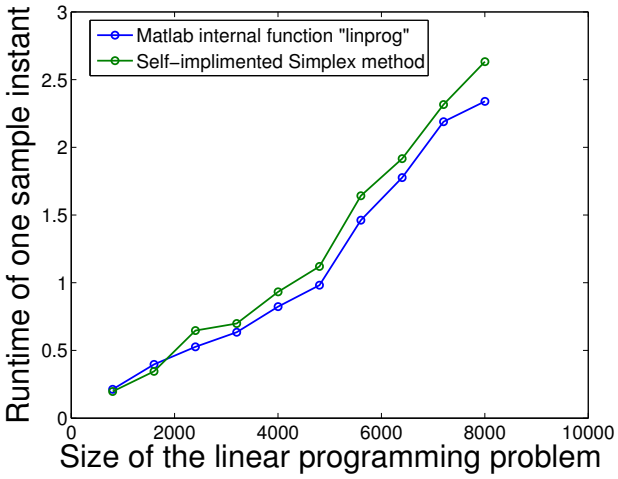


Figure 7: Runtime experiment for the linear programming based control problem.

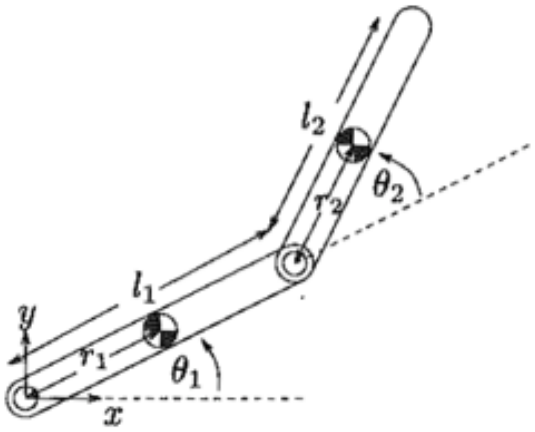


Figure 8: Diagram of the two-link planar robot arm system. θ_1 and θ_2 are the angles of the two joints. l_1 and l_2 are the length of the two linkages. r_1 and r_2 denote the distance between the center of mass of the linkage to the corresponding joint. The control input of the system are torques acting at the two joints.

$$\begin{bmatrix} I_1 + I_2 m_2 l_1 r_2 cos\theta_2 & I_2 + m_2 l_1 r_2 cos\theta_2 \\ I_2 + m_2 l_1 r_2 cos\theta_2 & I_2 \end{bmatrix} \begin{bmatrix} \ddot{\theta_1} \\ \ddot{\theta_2} \end{bmatrix} + \begin{bmatrix} -2m_2 l_1 r_2 \dot{\theta_2} & -m_2 l_1 r_1 sin\theta_2 \dot{\theta_2} \\ -m_2 l_1 r_1 sin\theta_2 \dot{\theta_1} & 0 \end{bmatrix} \begin{bmatrix} \dot{\theta_1} \\ \dot{\theta_2} \end{bmatrix} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{21}$$

By define a vector $\mathbf{q} = [\theta_1\ \theta_2]^T$, and $\mathbf{u} = [u_1\ u_2]^T$, the system can be written in a more compact form as

$$\mathbf{H}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} = \mathbf{u} \tag{22}$$

The detailed derivation of this dynamic equation is shown in [7]. From Equation 21 we can see that this is a nonlinear dynamic system, since the matrices $\mathbf{H}$ and $\mathbf{C}$ have dependency on the state variables. This nonlinearity will usually add great difficulty to the commonly used linear feedback control, since the feedback gains are fixed during all the exicution process. Adaptive control algorithm try to update the feedback gains in real time depend on the robot configuration, and win great success in the field of robotics. Being a control strategy that is doing on-line optimization, the LP based control also has the potential to deal with this nonlinearity. In this section a simulation is performed to verify the effectiveness of this control strategy on nonlinear problem.

Define the state variable as $\mathbf{x} = [\theta_1\ \theta_2\ \dot{\theta_1}\ \dot{\theta_2}]^T$, and put the system into a nonlinear 'state-space' form, the equation can be write as:

$$\frac{d}{dt} \begin{bmatrix} \mathbf{q} \\ \dot{\mathbf{q}} \end{bmatrix} = \begin{bmatrix} \mathbf{0}_{2\times 2} & \mathbf{I}_{2\times 2} \\ \mathbf{0}_{2\times 2} & \mathbf{H^{-1}C} \end{bmatrix} \begin{bmatrix} \mathbf{q} \\ \dot{\mathbf{q}} \end{bmatrix} + \begin{bmatrix} \mathbf{0}_{2\times 2} \\ \mathbf{H^{-1}} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{23}$$

If we linearize this equation by means of Taylor expansion, then the centrifugal and corriolis term (matrix $\mathbf{C}$) will disappear. This is the major drawback of the linear feedback control. However in the LP based control, we take the measurement of the state variable, and update the model based on our measurement every time step. That is, make the linearization to this nonlinear system based on the state at the sample instant.

Our control goal for this system is to make the end-effector of this robot arm to track a given trajectory: a circle in the task domain. Figure 9 shows a diagram of the system. In the figure, the black thick lines denote the robot arm, and the blue circle is the desired trajectory. We want the end-effector of the arm to track this circle with a constant speed in the task space, which is the Cartesian coordinate in this figure.

The control of this robot arm basically takes the following steps at one sample instant:

1. Take the measurement of the full state variable;
2. Determine the x-y space desired end-effector position (x_d, y_d) and the corresponding desired velocity;
3. Use inverse kinematics transformation of the robot model to calculate the desired trajectory in the joint space $[\mathbf{q_d}\ \dot{\mathbf{q}}_\mathbf{d}]^T$;

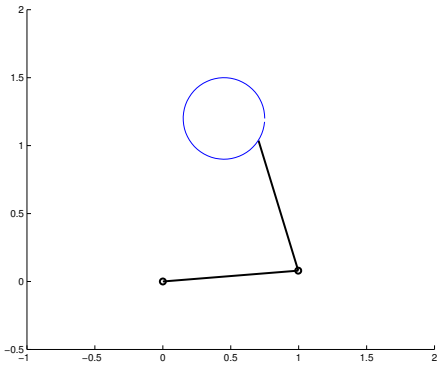
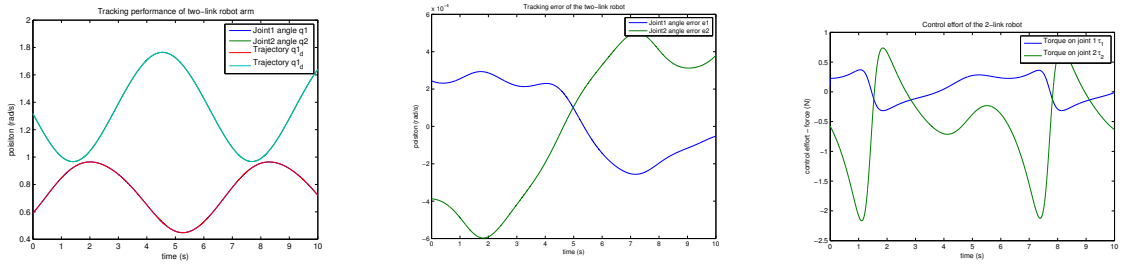


Figure 9: Trajectory tracking simulation of the two-link robot arm.

4. Formulate the linear system dynamics based on the current measurements;
5. Change the system into a standard linear programming problem by means of the procedure presented in the previous section;
6. Compute the control input u by means of linear programming.

Simulation of this two link robot is performed with a weighting factor to the state tracking errors to be $w = [10\ 10\ 1\ 1]$. Figure 10 presents the simulation result of the tracking performance, tracking error and the control input signals.



(a)Tracking performance. (b) Tracking error. (c) Control input u.

Figure 10: Tracking performance of two-link robot system.

The maximum tracking error during the cycle is 6×10^{-4}. This simulation validated the effectiveness of the LP based control for smooth nonlinear dynamic systems.

6 Conclusion and future work

In this project, an application of the linear programming optimization is studied in depth. A linear programming based control strategy is being implemented for problems

that have hard inequality constraints on state variables and control input variables. The control problem is successfully changed into a standard linear programming problem, and the simplex method is used to find the proper control signal. Several dynamic systems, either linear or nonlinear, are simulated to test the performance of the LP based controller. Simulation results demonstrate that when the computation time is not limiting the sampling rate, the LP based control have the advantages of: (a) better tracking accuracy, (b) can deal with smooth nonlinear system, (c) can strictly limit the variables within the constraints.

Further exploration in this topic can be: (a) Sensitivity study of the control performance by sweeping the parameters, such as the constraint limits and weighting coefficients; (b) study the effect of noise and disturbance signals to the system; (c) real-time implementation of this control strategy on an embedded computer and test the control on some hardware.

In the process of working on this project, the most challenging part should be changing the control problem into a linear programming problem, while the most time-consuming part is the implementation of the simplex method and the three systems for simulation. I am lucky to have worked on a project based on optimization, thus I can get a chance to explore more in depth in this material although there is no homework for this chapter. Significant thanks for this course! All the neither easy nor straightforward problem sets have gave me great chance for practicing Matlab coding, so that I can work this project out. Thank you 18.086!

References

[1] L. Zadeh and B. Whalen, "On optimal control and linear programming," *Automatic Control, IRE Transactions on*, vol. 7, no. 4, pp. 45–46, 1962.

[2] M. A. Dahleh and I. J. Diaz-Bobillo, *Control of uncertain systems: a linear programming approach*. Prentice-Hall, Inc., 1994.

[3] R. E. Kalman and R. S. Bucy, "New results in linear filtering and prediction theory," *Journal of basic engineering*, vol. 83, no. 1, pp. 95–108, 1961.

[4] G. Strang, *Computational science and engineering*. Wellesley-Cambridge Press Wellesley, 2007.

[5] G. Hadley, "Linear programming.." 1962.

[6] A. E. Bryson, *Applied optimal control: optimization, estimation and control*. CRC Press, 1975.

[7] M. I. of Technology. Project MAC. Engineering Robotics Group and M. Dertouzos, *Control robotics: The procedural control of physical processes*, 1973.