Optimization Theory Notes

Descent!

Author: occupymars

Date: June. 23, 2025

Version: 0.1

Contents

Chapter 1 CMU Optimal Control

1.1 Lecture 1 - 3

Introduction

- ❏ *Stability*
- ❏ *Discretation*
- ❏ *Root Finding*

A stable point $x_{balance}$ is a point follows:

$$Re[eig(\frac{\partial f}{\partial x})|_{x=x_{balance}}] < 0$$

For example, pendulum system:

$$f(x) = \begin{bmatrix} \dot{\theta} \\ -\frac{g}{l}\sin(\theta) \end{bmatrix} \rightarrow \frac{\partial f}{\partial x} = \begin{bmatrix} 0 & 1 \\ -\frac{g}{l}\cos(\theta) & 0 \end{bmatrix}$$

at its balance point $\theta = \pi$, eigen value is $\pm\sqrt{(\frac{g}{l})}$, has a negative one, so this point is not stable. And at $\theta = 0$, eigen value is $0 \pm i\sqrt{(\frac{g}{l})}$, given a push, it will oscillate, so this point is a semi-stable one.

Now we talk about discretization methods. The most command and easy one is forward euler:

$$x_{k+1} = x_k + hf_c(x_k, u_k)$$

where f_c is system dynamic in continuous form, h is the integral step.

$$\frac{\partial x_N}{\partial x_0} = (\frac{\partial f_d}{\partial x})|_{x_0}^N = A_d^N$$

where f_d is system dynamic in discrete form, in order to stablize the system, determinant of eigen value of A_d should less then 1.

$$A_d = \frac{\partial f_d}{\partial x_k} = I + h\frac{\partial f_c}{\partial x_k}I + h\begin{bmatrix} 0 & 1 \\ -\frac{g}{l}\cos(\theta) & 0 \end{bmatrix}$$

now for $\theta = 0$, the eigen value is $1 \pm 0.313i$, is outside the unit circle, which means unstable. This discretization make the original semi-stable system becomes unstable! So be careful when deal with a system without damping.

Next way is called ***Runge-Kutta*** method:

$$\begin{aligned} K_1 &= f(x_k, u) \\ K_2 &= f(x_k + \frac{1}{2}hK_1, u) \\ K_3 &= f(x_k + \frac{1}{2}hK_2, u) \\ K_4 &= f(x_k + hK_3, u) \\ x_{k+1} &= x_k + \frac{h}{6}(K_1 + 2K_2 + 2K_3 + K_4) \end{aligned}$$

Third, we introduce backward euler method, $f_d(x_{k+1}, x_k, u_k) = 0$, express x_{k+1} using x_{k+1} and x_k, the most simple form is (if we assume) $x_{k+1} = x_k + hf(x_{k+1})$, start from $x_{k+1} = x_k$, iteratively (for example using newton's method) compute until we get $x_{k+1} = x_k + hf(x_{k+1})$. In contract to forward one, this will underestimate the acceleration, add a artifical damp to the system.

For the discretization of control, we have zero-order and first-order hold.

Fixed Point Iteration is a method to find root for a function. One of the most famous one is ***Newton's Method***,

$$f(x + \Delta x) \approx f(x) = \frac{\partial f}{\partial x}\Delta x = 0 \rightarrow \Delta x = -(\frac{\partial f}{\partial x})^{-1}f(x)$$

and use it iteratively.

Minimization Problem, use newton's method but on derivative of f, so we find a minimum (or a saddle point). In order to make sure newton's method will move in a descent way, we need to make sure the hessian $H = \nabla^2 f > 0$. When

the hessian is less than 0, we can introduce ***Damped Newton's Method***, in a while loop, $H = H + \beta I$ until hessian is greater than 0. If β is too large, then the step will be small, but in general, this method is good enough. **Tips**: do not use eigen value to check the hessian, use Cholesky decompesation, if it fails it means it is not positive definate.

In general, newton's method will return a large step, we can use regularization above to decrease it, but it is still not enough. We need to introduce ***Line Search***, one of the simplest method is ***Armijo rule***:

$$\textbf{while } f(x+\alpha\Delta x) > f(x) + b\alpha\nabla f(x)^\top \Delta x, \textbf{ do } \alpha = c\alpha$$

we start from $\alpha = 1$.

1.2 Lecture 4 - 5

Introduction

- ❏ *Optimization with Equality Constraint*
- ❏ *Optimization with Inequality Constraint*
- ❏ *Line Search*
- ❏ *Regularization*

Given problem:

$$\min_x f(x);\ f(x) : \mathbb{R}^n \to \mathbb{R}$$
$$s.t.\ c(x) = 0;\ c(x) : \mathbb{R}^n \to \mathbb{R}^m$$

The direction of derivative of c is the non-free direction of x, in order to follow the constraints, the derivative of f must be this non-free one, and should not have a free component. It is same as ∇f direction is perpendicular to constraint manifold: $\nabla f + \lambda \nabla c = 0$, from this we can define a lagrange function $L(x,\lambda) = f(x) + \lambda^T c(x)$, where its two derivative is exactly above two conditions:

$$\begin{aligned} \nabla_x L(x,\lambda) &= \nabla f(x) + \nabla c(x)^T \lambda = 0 \\ \nabla_\lambda L(x,\lambda) &= c(x) \end{aligned} \tag{1.1}$$

this is the KKT condition for equality constraint. But directly solve it may not possible for nonlinear system, so we can use the newton's method (use taylor expansion in first order):

$$\begin{bmatrix} \frac{\partial^2 L}{\partial^2 x} & (\frac{\partial c}{\partial x})^T \\ \frac{\partial c}{\partial x} & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta \lambda \end{bmatrix} = \begin{bmatrix} -\nabla_x L(x,\lambda) \\ -c(x) \end{bmatrix} \tag{1.2}$$

This is called **KKT system**, left matrix is the lagrange fucntion's hessian. The above equation has second order derivative of constraints, it maybe complicated, so when we do the iteration of the first term, we can use ***Gaussian Newtom Method***, which will ignore second term in:

$$\frac{\partial^2 L}{\partial^2 x} = -\nabla^2 f + \frac{\partial}{\partial x}[(\frac{\partial c}{\partial x})^T \lambda]$$

it is same as first linearlize then find the minimum, and it's fast.

Inequality Constraint: $\min_x f(x)$ under $\ni c(x) \geq 0$. The KKT conditions are:

- Stationarity, $\nabla f - (\frac{\partial c}{\partial x})^T \lambda = 0$
- Primal Feasibility, $c(x) \geq 0$
- Dual Feasibility, $\lambda \geq 0$
- Complementarity, $\lambda^T c(x) = 0$

If $c(x) = 0$, then $\lambda \geq 0$, the difference with equality one is from λ, is can not be less than 0, so it must in the same direction of c derivative (can not be negative direction).

Methods to solve it:

- ***Active Set***, use guess heuristic function to guess which inequality constraint is activated, after find, the constraint that not activative will be ingored, iterate with the equality constraint problem method, then continue to guess. This method is fast, and good when heuristic is good.
- ***Barrier Function***, turn constraint to cost, make the cost huge when the constraint are disobeyed. It is one of the ***Interior-point Method*** methods.

- Make constraint as penalty. It is not good as it will make the hessian illed, but we can first use this method (or the barrier method, as a polish) then use active-set method.
- ***Augmented Lagrangian***, inner layer optimize x, outer layer update lagrangian multiplier. The function is $\min_x f(x) - \tilde{\lambda}^T c(x) + \rho/2[\min(0, c(x))]^2$, get derivative $\frac{\partial f}{\partial x} - [\tilde{\lambda} - \rho c(x)]^T \frac{\partial c}{\partial x}$, then update with $\tilde{\lambda} = \tilde{\lambda} - \rho c(x)$. We start with $\tilde{\lambda} = 0$, meaning all constraints are not active, then optimize without the constraint, see which constraints are disobeyed, if it is disobeyed then update the lagrangian multiplier, and at next iteration it will have the tend to follow the constraint.

Regularization and line search of inequality constraint problem, we can write the original function as $\min_x \max_{\lambda \geq 0} f(x) - \lambda c(x)$ (if we change the min and max without any loss, then this is ***Strong Duality***, most of the convex problem are). Whenever $c(x) \leq 0$, inner loop blows up. In order to go to minimum rather than saddle point, we need to make sure the hessian has $dim(x)$ positive eigen values and $dim(\lambda)$ negative eigen values, make the KKT system Quasi-definite. The system with regularization is:

$$\begin{bmatrix} \frac{\partial^2 L}{\partial^2 x} + \alpha I & (\frac{\partial c}{\partial x})^T \\ \frac{\partial c}{\partial x} & -\alpha I \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta \lambda \end{bmatrix} = \begin{bmatrix} -\nabla_x L(x, \lambda) \\ -c(x) \end{bmatrix}, \alpha > 0 \tag{1.3}$$

we add negative eigen value to the λ term.

However, this may still have overshooting problem, we need line search with constraint. Consider a general optimization problem $\min_x f(x)$ under $\ni c(x) \geq 0$ and $\ni d(x) = 0$, the lagrangian is:

$$L(x, \lambda, \mu) = f(x) - \lambda^T c(x) + \mu^T d(x)$$

One of the possible KKT residual is $P(x, \lambda, \mu) = \frac{1}{2} \|\nabla L(x, \lambda, \mu)\|_2^2$, where:

$$\nabla L(x, \lambda, \mu) = \begin{bmatrix} \nabla_x L(x, \lambda, \mu) \\ \min(0, c(x)) \\ d(x) \end{bmatrix}$$

but this need too much computation, so other possible choise is:

$$P(x, \lambda, \mu) = f(x) + \rho \| \begin{bmatrix} \min(0, c(x)) \\ d(x) \end{bmatrix} \|_1$$

considers both function and violation of constraints.

And last possible choise is directly use augmented lagrangian:

$$P(x, \lambda, \mu) = f(x) - \tilde{\lambda}^T c(x) + \tilde{\mu}^T d(x) + \frac{\rho}{2} \| \min(0, c(x)) \|_2^2 + \frac{\rho}{2} \|d(x)\|_2^2$$

1.3 Lecture 6 -

Introduction

❏ *Deterministic Optimal Control* ❏ *LQR*

If we substitute x with u and f, then this is called ***Shooting Method*** or ***Indirect Method***, this has some problem, one is computing A^N will return illed problem, make the gradient blows up or disappear, second is we may not find a good initial u for the system, if we use x and u both, then we can use x as a guess, and when optimizing we will not at a werid position (for example a humanoid).

Chapter 2 Some Control Techniques

2.1 Model Predictive Control

2.1.1 Linearlization of Nonlinear Model

Given model continuous dynamic $\dot{x} = F(s, v)$, jacobian as $F_s(x, u), F_v(x, u)$, we can discretize it with multiple methods.

First we can use explicit Euler: start with $x^+ = x_k, A_k = I, B_k = 0$:

$$\begin{aligned} x^+ &\leftarrow x^+ + F(x^+, u_k)T_i \\ [A_k \quad B_k] &\leftarrow (I + F_s(x^+, u_k)T_i)\,[A_k \quad B_k] + [0 \quad F_v(x^+, u_k)T_i] \end{aligned} \tag{2.1}$$

this two update should be update in parallel, not sequntial. Repeat this $N_s = T_s/T_i$ times to get x^+, where T_s is the interval time step between two control points, this results an error of $\mathcal{O}(T_i)$.

Then if we use Runge-Kutta 4, we get:

Algorithm 1: N_S steps Runge-Kutta 4 with forward AD

Input: x_k, u_k
Output: x^+, A_k, B_k

$x^+ = x_k, \quad A_k = I, \quad B_k = 0;$
for $n = 1 : N_S$ **do**
  $k_1 \leftarrow F(x^+, u_k);$
  $\left[\frac{dk_1}{dx_k}, \frac{dk_1}{du_k}\right] \leftarrow F_s(x^+, u_k)[A_k\ B_k] + [0\ F_v(x^+, u_k)];$
  $k_2 \leftarrow F(x^+ + \frac{T_i}{2}k_1, u_k);$
  $\left[\frac{dk_2}{dx_k}, \frac{dk_2}{du_k}\right] \leftarrow F_s(x^+ + \frac{T_i}{2}k_1, u_k)[(A_k + \frac{T_i}{2}\frac{dk_1}{dx_k})\ (B_k + \frac{T_i}{2}\frac{dk_1}{du_k})] + [0\ F_v(x^+ + \frac{T_i}{2}k_1, u_k)];$
  $k_3 \leftarrow F(x^+ + \frac{T_i}{2}k_2, u_k);$
  $\left[\frac{dk_3}{dx_k}, \frac{dk_3}{du_k}\right] \leftarrow F_s(x^+ + \frac{T_i}{2}k_2, u_k)[(A_k + \frac{T_i}{2}\frac{dk_2}{dx_k})\ (B_k + \frac{T_i}{2}\frac{dk_2}{du_k})] + [0\ F_v(x^+ + \frac{T_i}{2}k_2, u_k)];$
  $k_4 \leftarrow F(x^+ + T_i k_3, u_k);$
  $\left[\frac{dk_4}{dx_k}, \frac{dk_4}{du_k}\right] \leftarrow F_s(x^+ + T_i k_3, u_k)[(A_k + T_i\frac{dk_3}{dx_k})\ (B_k + T_i\frac{dk_3}{du_k})] + [0\ F_v(x^+ + T_i k_3, u_k)];$
  $x^+ \leftarrow x^+ + \frac{T_i}{6}(k_1 + 2k_2 + 2k_3 + k_4);$
  $[A_k\ B_k] \leftarrow [A_k\ B_k] + \frac{T_i}{6}\left[\frac{dk_2}{dx_k}\ \frac{dk_2}{du_k}\right] + 2\left[\frac{dk_3}{dx_k}\ \frac{dk_3}{du_k}\right] + \left[\frac{dk_4}{dx_k}\ \frac{dk_4}{du_k}\right];$
end
return x^+, A_k, B_k

2.1.2 Real Time Iteration