机器人中的数值优化

1. preliminaries

1. Introduction

optimization problem

$$
\begin{array}{cc}
\min & f(x) \\
\text{s.t.} & g(x) \leq 0 \\
 & h(x) = 0
\end{array}
$$

default assumptions:

- the objective function is lower bounded
    - lower bounded : 目标函数有下界

      $$f(x) \geq L$$

    - 保证 物理限制和实际约束、 优化问题的稳定性
- the objective function has bounded sub-level sets
    - bounded sub-leveled set: 有界次水平集
    - $$S_\alpha = \{u \mid J(u) \leq \alpha\}$$
    - 可行解的存在性和可行域的有限性：有界次水平集保证了可行解集不会无限扩展。也就是说，在寻找最优解时，算法不会无穷无尽地搜索整个控制策略空间。
    - 算法的收敛性：优化算法在有界次水平集上工作时，能够确保算法逐步逼近最优解，不会陷入无穷大范围的搜索。这对保证算法的收敛性和计算效率至关重要。
    -

矩阵的条件数

如果矩阵 A 或常数项 b 的微小变化, 引起线性方程组 Ax = b 解的巨大变化，称此线性方程组为“病态”方程组, 矩阵 A 称为“病态”矩阵, 否则称方程组为“良态”方程组, A 称为“良态”矩阵。

矩阵的条件数（condition number）是一个用于度量矩阵在数值计算中敏感性的重要指标。它反映了线性系统解的变化相对于输入变化的敏感度。如果一个矩阵的条件数非常大，则意味着这个矩阵的逆矩阵

对输入的微小变化非常敏感，计算可能会不稳定或不准确。相反，如果条件数较小，则计算相对稳定和可靠。

具体来说，对于一个矩阵 A，它的条件数通常记作 $\kappa(A)$，可以用以下几种方式定义：

1. **基于范数的条件数**：
   条件数可以定义为矩阵的范数和其逆矩阵的范数的乘积。对于某种范数 $\|\cdot\|$（如 1-范数、2-范数或无穷范数），条件数定义为：
   $$\kappa(A) = \|A\| \cdot \|A^{-1}\|$$
2. **基于特征值的条件数**（针对 2-范数）：
   条件数也可以通过矩阵的奇异值来定义，特别是对于 2-范数（欧几里得范数），条件数是最大奇异值与最小奇异值的比值：
   $$\kappa_2(A) = \frac{\sigma_{\max}(A)}{\sigma_{\min}(A)}$$
   其中，$\sigma_{\max}(A)$ 和 $\sigma_{\min}(A)$ 分别是矩阵 A 的最大和最小奇异值。
3. **基于条件数的解释**：
   条件数 $\kappa(A)$ 提供了以下解释：
   - $\kappa(A) = 1$：矩阵 A 是最佳条件的，即最稳定的。
   - $\kappa(A)$ 较大：矩阵 A 是病态的（ill-conditioned），即对输入误差非常敏感，数值计算不稳定。
   - $\kappa(A) \to \infty$：矩阵 A 是奇异的或接近奇异，无法进行稳定的数值计算。

在实际应用中，通常使用2-范数来计算条件数，因为它具有明确的几何意义和易于解释的特性。然而，根据具体应用和需求，也可以选择其他合适的范数来计算条件数。

2.Convexity

why convexity?

- 凸函数在凸集合上的优化已经被充分研究
- 优化算法利用凸函数/集合的性质来分析收敛性
- 一些重要的问题具有凸的公式化/松弛:很多实际问题可以被转化为凸优化问题，或通过松弛技术近似为凸优化问题。这使得我们可以利用凸优化的方法来解决原本非凸的问题，从而简化求解过程
- 许多非凸函数在感兴趣的局部极小值附近是局部凸的
- 在实际中，非凸函数的局部极小值可能已经足够好

3.Convex sets

definition

A set is convex if all **convex combinations** of its points are also in the set.

convex cobinations:

$$\sum \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3$$

$$\sum \theta_i = 1$$

examples

超平面、半空间、球、多项式

Cone 锥不一定是凸的

Cone：$x \in C \Rightarrow ax \in C, \forall a \geq 0$

Second-order cone SOC: $C_2 = \{(x,t) \mid \|x\| \leq t\} \in \mathbb{R}^{n+1}$ 是凸的

Semi-definite cone: $\mathcal{S}_+^n = \left\{A \in \mathbb{R}^{n \times n} \mid A = A^T, A \succeq 0\right\}$ 是凸的

properties of convex sets

保凸性：

- The intersection of convex sets is convex：多面体
- set sum $A + B = \{x + y \mid x \in A, y \in B\}$
- set product $A \times B = \{(x,y) \mid x \in A, y \in B\}$

4. high order info of functions

Function

$$f(x) = f(x_1, x_2, x_3)$$

Gradient

$$\nabla f(x) = \begin{pmatrix} \partial_1 f(x) \\ \partial_2 f(x) \\ \partial_3 f(x) \end{pmatrix}$$

Hessian: Symmetric for smooth function 对于光滑函数是对称的

$$\nabla^2 f(x) = \begin{pmatrix} \partial_1^2 f(x) & \partial_1\partial_2 f(x) & \partial_1\partial_3 f(x) \\ \partial_2\partial_1 f(x) & \partial_2^2 f(x) & \partial_2\partial_3 f(x) \\ \partial_3\partial_1 f(x) & \partial_3\partial_2 f(x) & \partial_3^2 f(x) \end{pmatrix}$$

Approximation

$$f(x) = f(x - x_0) + x^T \nabla f(x - x_0) + \frac{1}{2} x^T \nabla^2 f(x - x_0) x + O\left(\|x - x_0\|^3\right)$$

Jacobian

The extension of the gradient to higher order.

$$f(x) : \mathbb{R}^n \to \mathbb{R}^m$$

$$f'(x) = \frac{df}{dx^T} = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & \cdots & \frac{\partial f_m}{\partial x_n} \end{pmatrix}$$

useful notation of differential

https://en.wikipedia.org/wiki/Matrix_calculus

$$\begin{aligned} & dA = 0 \\ & d(\alpha X) = \alpha(dX) \\ & d(AXB) = A(dX)B \\ & d(X + Y) = dX + dY \\ & d(X^\top) = (dX)^\top \\ & d(XY) = (dX)Y + X(dY) \\ & d\langle X, Y \rangle = \langle dX, Y \rangle + \langle X, dY \rangle \\ & d\left(\frac{X}{\phi}\right) = \frac{\phi dX - (d\phi)X}{\phi^2} \\ & d(tr(X)) = I \\ & df(g(x)) = \frac{df}{dg} \cdot dg(x) \end{aligned}$$

5. Convex Functions

Jensen's inequality

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta)f(y)$$

Epigraph：上方图

$$\mathrm{epi}(f) = \{(x, y) \mid f(x) \leq y\}$$

convex function == convex epigraph

why convex functions?

- 凸函数有凸的次水平集
- 凸函数的性质相对容易保持
  - 拟凸函数的和不一定是凸的
- 凸函数任何局部最优解就是全局最优解
- 凸函数在局部极小值附近是局部凸的
- 凸函数的许多运算是保凸的：
  - 非负加权和
  - 仿射变换
  - 绝对值
  - 范数
  - 最大特征值
  - trace
  - 线性运算

convex functions property

- 凸函数在线性近似的上方
- 一阶导数为0 就是最优解（只对凸函数成立）
- 如果光滑函数对于任何X的hessian是半正定的，那么它是凸的
- 对于非凸函数，极小值点的hessian是半正定的
- The Hessian is a good local model of a smooth function

6. Unconstrained Optimization for Nonconvex Functions

Line search steepest gradient descent

$$x^{k+1} = x^k + \tau d,$$
$$d = -\nabla f(x^k)$$

- constant step size $\tau = constant$
- Diminishing the step size $\tau = \frac{1}{k}$
- Exact line search $\tau = \arg\min_\alpha f(x^k + \alpha d)$
- Inexact line search $\tau \in \left\{\alpha \mid f(x^k) - f(x^k + \alpha d) \geq -c \cdot \alpha d^{\mathrm{T}} \nabla f(x^k)\right\}$
- constant step size:
  - too large: may oscillate and diverge
  - too small: may take too long to converge
- Diminishing the step size:
  - converge but expensive
- Excact line search:
  - generally nontrivial
- Inexact line search:
  - easy to satisfy

Inexcact line search (Armijo condition)

1. compute search gradient $d = -\nabla f(x^k)$
2. while $f(x^k + \tau d) \geq f(x^k) + c\tau d^T \nabla f(x^k)$
   i. $\tau = \tau/2$
3. Update iterate $x^{k+1} = x^k + \tau d$

7. Modified Damped Newton's Method

Newtons's method

root finding problem

1st order Taylor expansion

$$f(\boldsymbol{x}) \approx \hat{f}(\boldsymbol{x}) \triangleq f(\boldsymbol{x}_k) + \nabla f(\boldsymbol{x}_k)^T (\boldsymbol{x} - \boldsymbol{x}_k) = 0$$

$$f(x) = 0 \rightarrow x^{k+1} = x^k - \frac{f(x^k)}{f'(x^k)}$$

minimization problem

2nd order Taylor expansion

$$f(\boldsymbol{x}) \approx \hat{f}(\boldsymbol{x}) \triangleq f(\boldsymbol{x}_k) + \nabla f(\boldsymbol{x}_k)^T(\boldsymbol{x}-\boldsymbol{x}_k) + \frac{1}{2}(\boldsymbol{x}-\boldsymbol{x}_k)^T\nabla^2 f(\boldsymbol{x}_k)(\boldsymbol{x}-\boldsymbol{x}_k)$$

$$\nabla \hat{f}(\boldsymbol{x}) = \nabla f(\boldsymbol{x}_k)^T + \nabla^2 f(\boldsymbol{x}_k)(\boldsymbol{x}-\boldsymbol{x}_k) = 0$$

$$x^{k+1} = x^k - \nabla^2 f(\boldsymbol{x}_k)^{-1}\nabla f(\boldsymbol{x}_k)$$

8. Penalty Methods

$$P_E(x,\sigma) = f(x) + \frac{1}{2}\sigma\sum_{i\in\mathcal{E}} c_i^2(x)$$

由于这种罚函数对不满足约束的点进行惩罚，在迭代过程中点列一般处于可行域之外，因此它也被称为外点罚函数．二次罚函数的特点如下：对于非可行点而言，当σ变大时，惩罚项在罚函数中的权重加大，对罚函数求极小，相当于迫使其极小点向可行域靠近；在可行域中，PE(x,σ)的全局极小点与约束最优化问题的最优解相同。

- 参数选择：在算法6.1中，参数 σ 的选择非常关键。如果σ 增长太快，子问题将变得难以求解；反之，如果增长太慢，算法需要的迭代次数会增加（例如，while循环次数增加）。合理的做法是根据当前罚函数的求解难度来确定σ 的增幅。如果当前子问题收敛迅速，可以在下一步选择较大的σ ；否则，不宜过多增加 σ
- 终止条件：在前面的例子中提到 罚函数 在 σ 较小时可能无界，此时迭代就会发散。当求解子问题时，一旦检测到迭代发散，应立即终止迭代并增加罚因子。
- 精度要求：为了确保收敛，子问题的求解精度必须足够高，残差需要趋近于零。

9. ALM