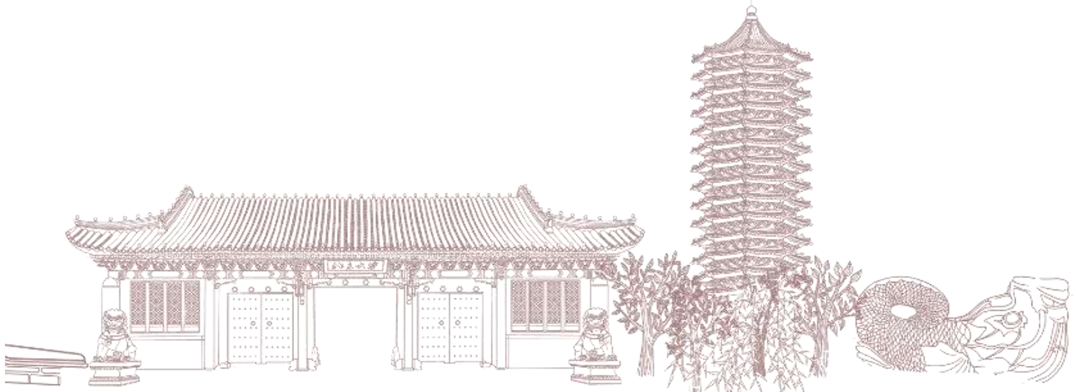
北京大學 PEKING UNIVERSITY

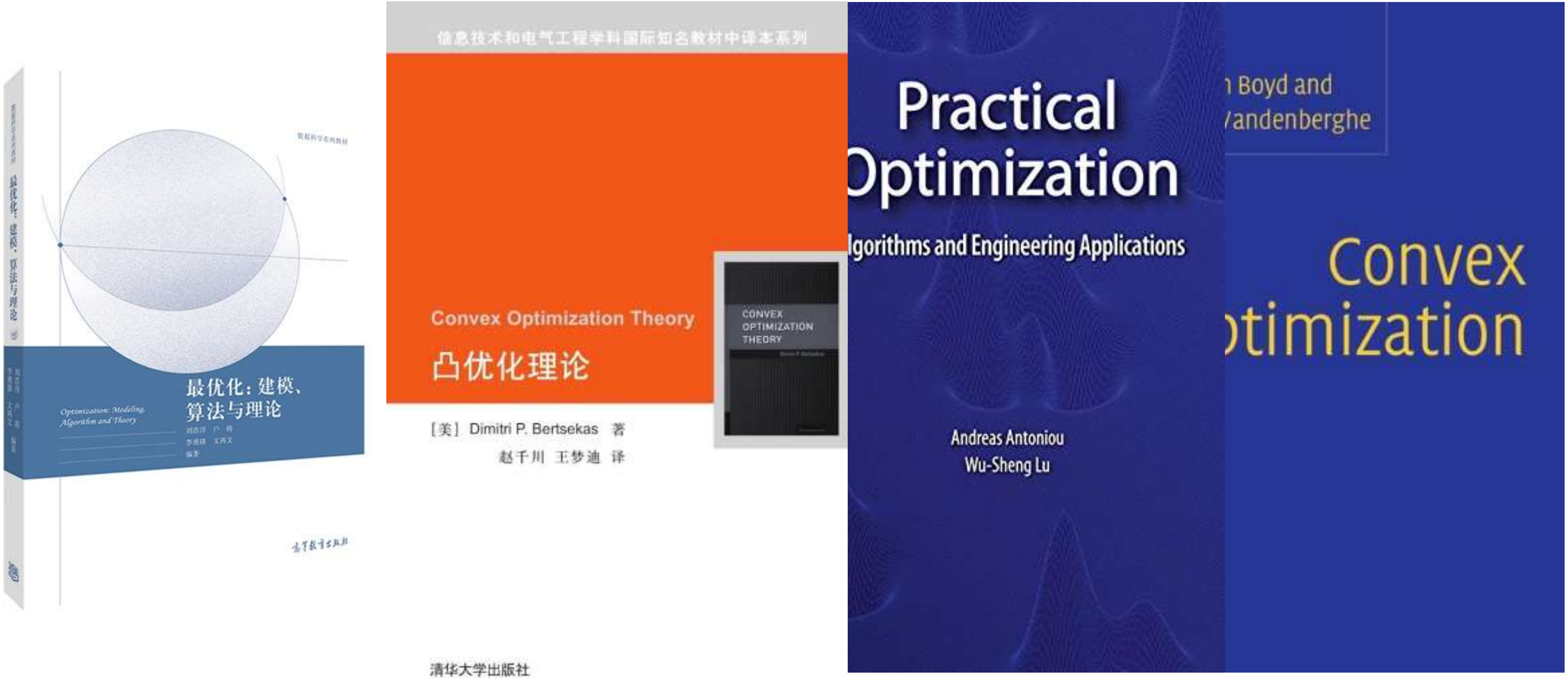
# 优化基础
# Optimization Basis

阮良旺

2024.6.24



# 优化是一个宏大的课题


最优化：建模、算法与理论
Optimization: Modeling, Algorithm and Theory
信息技术和电气工程学科国际知名教材中译本系列
Convex Optimization Theory
凸优化理论
[美] Dimitri P. Bertsekas 著
赵千川 王梦迪 译
清华大学出版社
CONVEX OPTIMIZATION THEORY
Practical Optimization
lgorithms and Engineering Applications
Andreas Antoniou
Wu-Sheng Lu
Boyd and
andenberghe
Convex
otimization

# 优化问题

$$\min f(x)$$
$$x \in C$$

- $x$是待优化的变量，可以是离散的（整数、选择、排序…），也可以是连续的（标量、矢量、张量…）
- $f(x)$是优化目标，可以是离散的、连续的、可导的…
- $C$是约束，规定了$x$可取的范围，可以是一个集合、一个子空间、取值范围…

# 无约束优化
# Unconstrained Optimization

# 最简单的情况：无约束优化

$$\min f(x)$$
$$x \in \mathbb{R}^n$$

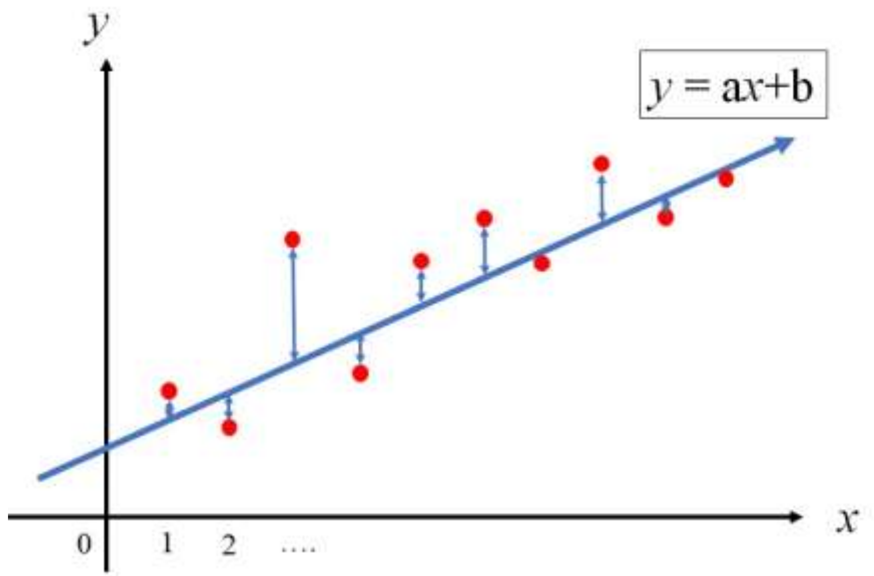
- $x \in \mathbb{R}^n$是连续的向量
- $C = \mathbb{R}^n$是全空间
- $f(x)$连续可微

# 示例：最小二乘法 (Least Squares)

$$y = \begin{pmatrix} g_0(x) & \cdots & g_{k-1}(x) \end{pmatrix} \begin{pmatrix} a_0 \\ \vdots \\ a_{k-1} \end{pmatrix} = \boldsymbol{g}^{\mathrm{T}}(x)\boldsymbol{a}$$

给定数据集 $(x_0, y_0), \cdots, (x_{n-1}, y_{n-1})$, 最小化

$$\sum_i \left(y_i - \boldsymbol{g}^{\mathrm{T}}(x_i)\boldsymbol{a}\right)^2$$



# 示例：最小二乘法 (Least Squares)

$$\mathbf{G} = \begin{pmatrix} g_0(x_0) & \cdots & g_{k-1}(x_0) \\ g_0(x_1) & \cdots & g_{k-1}(x_1) \\ & \vdots & \\ g_0(x_{n-1}) & \cdots & g_{k-1}(x_{n-1}) \end{pmatrix}, \boldsymbol{y} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{n-1} \end{pmatrix}$$

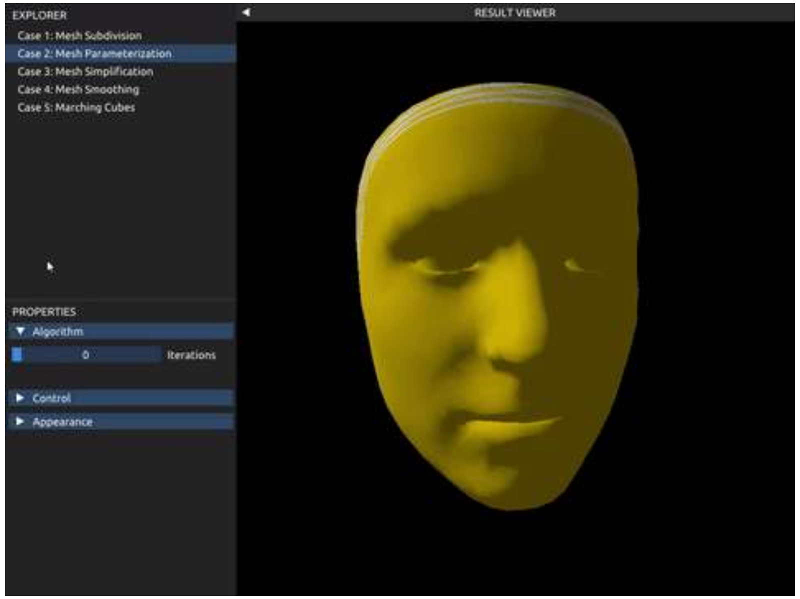
$$\sum_i \left(y_i - \boldsymbol{g}^{\mathrm{T}}(x_i)\boldsymbol{a}\right)^2 = \|\boldsymbol{y} - \mathbf{G}\boldsymbol{a}\|^2$$

$$\min_{\boldsymbol{a}} \|\boldsymbol{y} - \mathbf{G}\boldsymbol{a}\|^2 = \boldsymbol{a}^{\mathrm{T}}\mathbf{G}^{\mathrm{T}}\mathbf{G}\boldsymbol{a} - 2y^{\mathrm{T}}\mathbf{G}\boldsymbol{a} + \boldsymbol{y}^2$$

# 示例：曲面参数化

- 我们要为曲面上的每一个顶点定义一个二维纹理坐标$(u,v)$用于贴图，并最小化贴图的拉伸变形，尽可能让纹理坐标均匀分布在曲面上
- 一个简单的想法是给定曲面边界上的纹理坐标，然后最小化曲面内部每条边上纹理坐标的差值

$$\min \sum_{ij \in E} (u_i - u_j)^2 + (v_i - v_j)^2$$

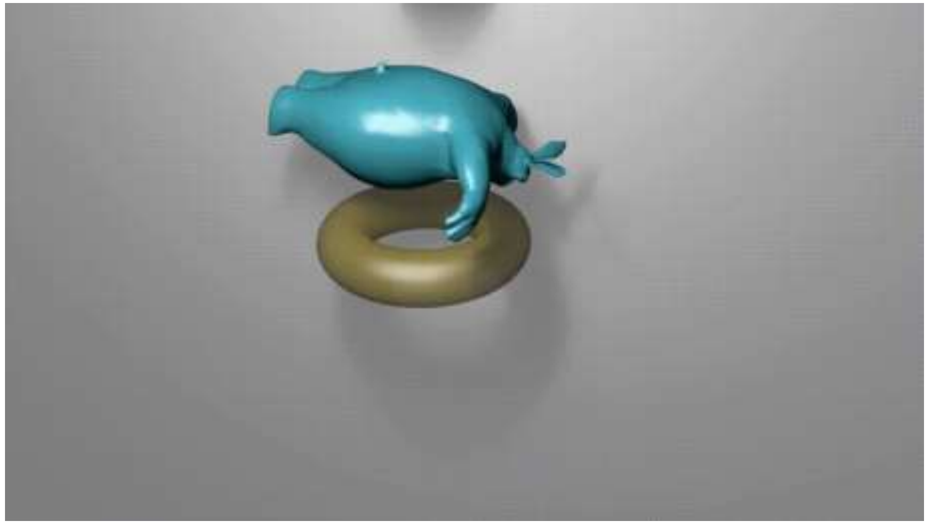


# 示例：弹性体模拟

在对所有顶点的位置$x$优化下面的增使用隐式欧拉法进行时间积分时，在每个时间步内，弹性体模拟等价于优化增量能量 (Incremental Potential)：

$$\min \frac{1}{2h^2}\|x - x^*\|_{\mathrm{M}}^2 + \Psi(x)$$

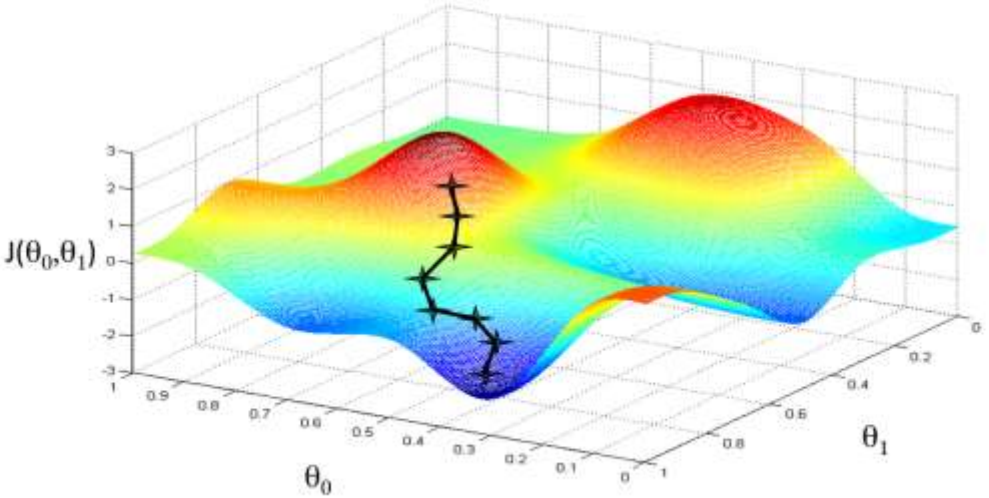
其中$h$是时间步长，$x^* = x^n + v^n h$，$\Psi(x)$是总弹性势能



# 线搜索 (Line Search)

$$x_{k+1} = x_k + \alpha_k p_k$$

每步迭代需要确定两个量：下降方向$p_k$和步长$\alpha_k$



# 梯度下降 (Gradient Descent)

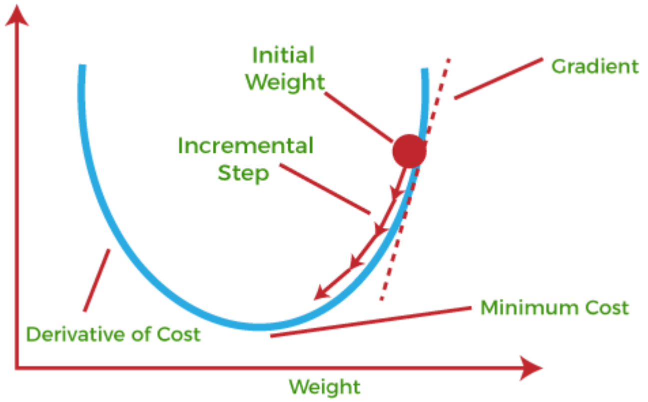
- 选择$p_k = -\nabla f(x_k)$，也就是下降最快的方向
- 选择$\alpha_k$使得

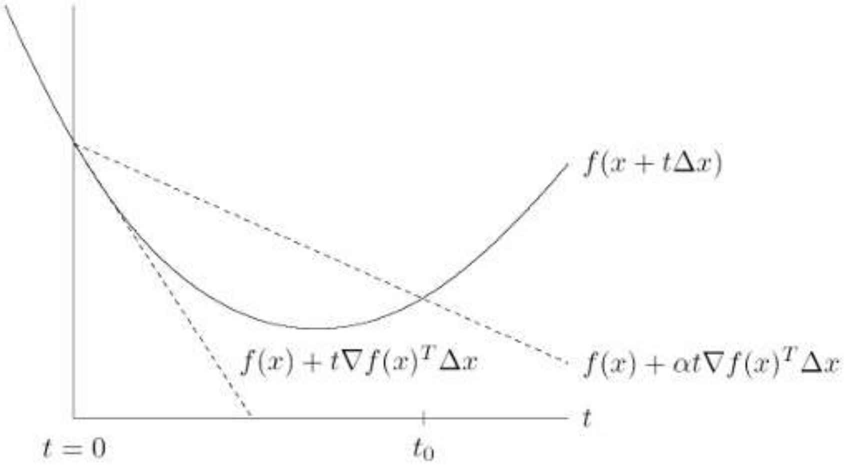
$$\min_{\alpha_k} f(x_k + \alpha_k p_k)$$

- 直接求解这个一维优化问题不一定容易
- 我们可以用近似的Armijo条件代替:

$$f(x_k + \alpha_k p_k) \le f(x_k) + c_1 \alpha_k p_k^T \nabla f(x_k)$$

- $c_1$是给定常数，一般比较小（1e-4）

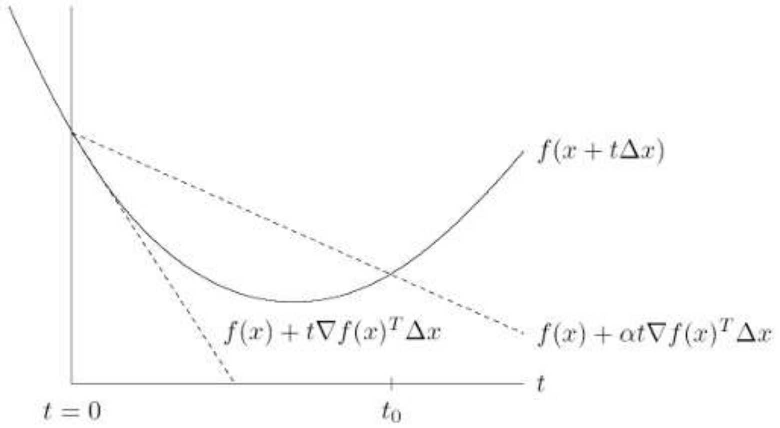


# Armijo条件

$$f(x_k+\alpha_k p_k) \le f(x_k) + c_1\alpha_k p_k^T \nabla f(x_k)$$

- 将$f(x)$在$x_k$局部做一阶展开，对应切线
  $$f(x_k+\Delta x) \approx f(x_k) + \nabla f(x_k)^T \Delta x$$
- 将切线斜率乘以$c_1 < 1$，要求我们最少应该按照$c_1\nabla f(x_k)$的斜率下降
- 为了避免$\alpha_k$接近0来满足条件，我们应该从一个大的初始值开始（比如1），然后逐步减小$\alpha_k$，直到满足条件



# 梯度下降算法

**Algorithm 3.2.1** Gradient method with Armijo rule

**(S0)** Choose $x^0 \in \mathbb{R}^n, \sigma, \beta \in (0,1), \varepsilon \geq 0$ and put $k := 0$.
**(S1)** If $\|\nabla f(x^k)\| \leq \varepsilon$: STOP.
**(S2)** Put $d^k := -\nabla f(x^k)$.
**(S3)** Determine $t_k > 0$ by

$$t_k := \max_{l \in \mathbb{N}_0} \beta^l \quad \text{s.t.} \quad f(x^k + \beta^l d^k) \leq f(x^k) + \beta^l \sigma \nabla f(x^k)^T d^k$$

**(S4)** Put $x^{k+1} := x^k + t_k d^k, k \leftarrow k + 1$and go to **(S1)**.

# 牛顿迭代 (Newton's Iteration)

- 在选择下降方向$p_k$时，梯度下降法只考虑了目标的一阶（梯度）信息，牛顿迭代考虑了目标的二阶（曲率）信息
- 将目标函数在$x_k$附近做二阶展开

$$f(x_k + p) \approx f(x_k) + p^T \nabla f(x_k) + \frac{1}{2} p^T \nabla^2 f(x_k) p \coloneqq m_k(p)$$

- 牛顿法希望下降方向$p_k$能够最小化$m_k$

$$p_k = \arg\min m_k(p)$$

- 令$\nabla m_k(p) = 0$，可以得到

$$p_k = -\left(\nabla^2 f(x_k)\right)^{-1} \nabla f(x_k)$$

# 牛顿迭代 (Newton's Iteration)

$$p_k = -\left(\nabla^2 f(x_k)\right)^{-1} \nabla f(x_k)$$

- $\nabla^2 f(x_k)$是一个矩阵，称为海森矩阵（Hessian matrix），因此在牛顿迭代法中我们需要解矩阵方程才能得到下降方向
- 如果$\nabla^2 f(x_k)$不正定，那么

$$\nabla f(x_k)^T p_k = -\nabla f(x_k)^T \left(\nabla^2 f(x_k)\right)^{-1} \nabla f(x_k)$$

有可能大于0，意味着不能保证目标函数一定下降

- 因此当$\nabla^2 f(x_k)$不正定时，一般需要额外处理$\nabla^2 f(x_k)$将其投影为正定矩阵，称为投影牛顿法（Projected Newton），或者退化回梯度下降

# 牛顿迭代算法

**Algorithm 3.3.3** Globalized Newton's method (optimization)

**(S0)** Choose $x^0 \in \mathbb{R}^n, \rho > 0, p > 2, \beta \in (0,1), \sigma \in (0,1/2), \varepsilon \geq 0$ and put $k := 0$.

**(S1)** If $\|\nabla f(x^k)\| \leq \varepsilon$: STOP.

**(S2)** Find a solution $d^k$ of the Newton equation

$$\nabla^2 f(x^k) d = -\nabla f(x^k). \tag{3.12}$$

If no solution can be found or if

$$\nabla f(x^k)^T d^k \leq -\rho \|d^k\|^p \tag{3.13}$$

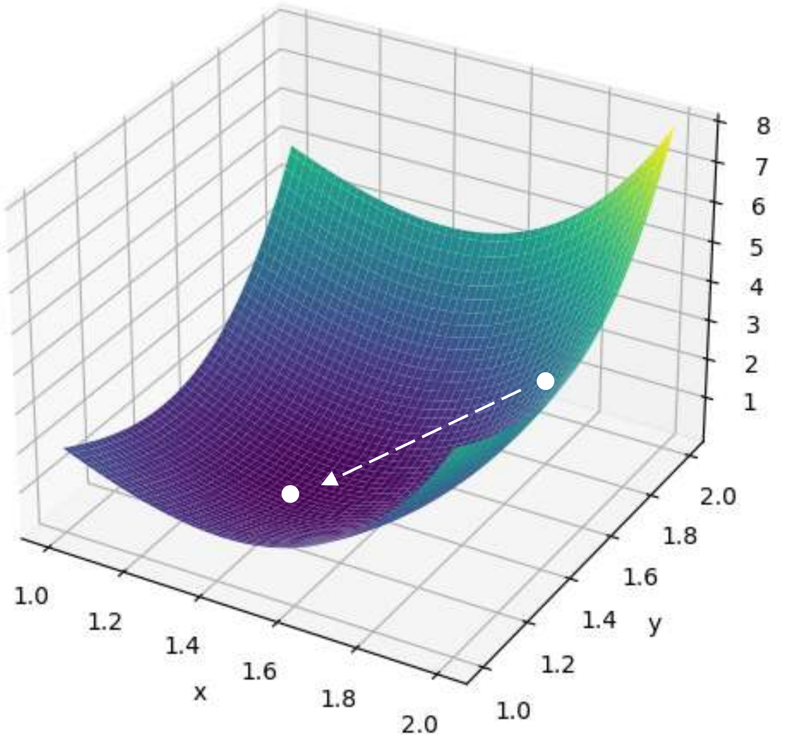
is violated put $d^k := -\nabla f(x^k)$.

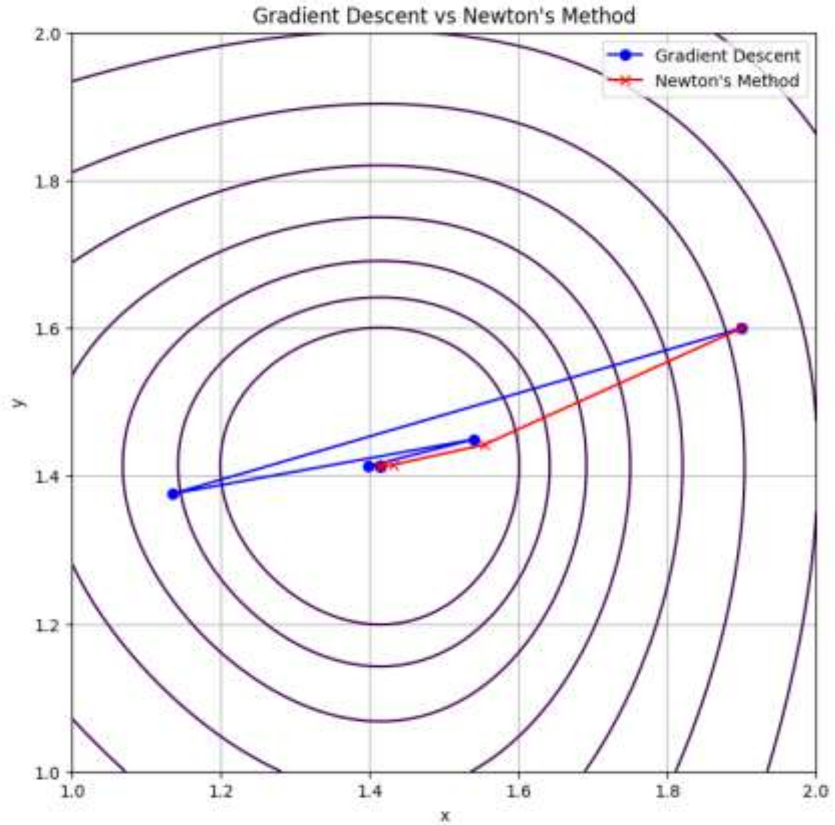
**(S3)** Determine $t_k$ by

$$t_k := \max_{l \in \mathbb{N}_0} \left\{ \beta^l \;\middle|\; f(x^k + \beta^l d^k) \leq f(x^k) + \beta^l \sigma \nabla f(x^k)^T d^k \right\}. \tag{3.14}$$

**(S4)** Put $x^{k+1} := x^k + t_k d^k, k \leftarrow k+1$ and go to **(S1)**.

# 比较




一般来说，牛顿下降有着更快的收敛速度，并且更不容易出现振荡（overshoot）的情况

# 拟牛顿法 (Quasi-Newton Method)

- 牛顿法虽然收敛快，但是需要求解海森矩阵的$\nabla^2 f(x_k)$方程，在$x$的自由度比较大的时候比较耗时
- 如果我们能找到一个更简单的矩阵$\mathrm{B}_k$近似$\nabla^2 f(x_k)$，就能大大提高收敛速度
- 使用这种想法的一类方法都称为拟牛顿法
- 我们可以将梯度下降视为使用单位矩阵I近似$\nabla^2 f(x_k)$的拟牛顿法，那么一个最直接的改进就是使用$\nabla^2 f(x_k)$的对角部分作为$\mathrm{B}_k$
- 常见的拟牛顿法还包括SR1，BFGS，L-BFGS等

# BFGS

From left to right: Broyden, Fletcher, Goldfarb, and Shanno

# BFGS

- 在每次迭代中，BFGS算法还会额外更新一个近似海森矩阵$B_k$
- 在迭代最开始时，我们初始化矩阵$B_0$，比如使用单位矩阵I
- 在每步迭代中我们更新$B_k$:

$$B_{k+1} = B_k + \Delta B_k$$

- 构造$\Delta B_k$的要求是$B_{k+1}$为$\nabla^2 f(x_{k+1})$的近似，表达为

$$\nabla f(x_{k+1}) - \nabla f(x_k) = B_{k+1}(x_{k+1} - x_k)$$

简记为

$$y_k = B_{k+1} s_k$$

# BFGS

$$B_{k+1} = B_k + \Delta B_k, \qquad y_k = B_{k+1} s_k$$

- BFGS假设$\Delta B_k$是一个秩为2的更新，可以写为

$$\Delta B_k = auu^T + bvv^T$$

其中$u, v$是两个向量，$a, b$是两个常数

- 代入近似海森矩阵的要求，得到

$$(B_k + auu^T + bvv^T)s_k = y_k$$
$$au^T s_k u + bv^T s_k v = y_k - B_k s_k$$

- 最直接的取法可以令

$$au^T s_k = 1, u = y_k, bv^T s_k = -1, v = B_k s_k$$

# BFGS

- 将$a, u, b, v$代入$\Delta \mathrm{B}_k$中，得到

$$\mathrm{B}_{k+1} = \mathrm{B}_k + \frac{y_k y_k^T}{y_k^T s_k} - \frac{\mathrm{B}_k s_k s_k^T \mathrm{B}_k}{s_k^T \mathrm{B}_k s_k}$$

- 这个形式的$B_{k+1}$可以使用 Shermann-Morrison-Woodbury公式写出显式的逆矩阵：

$$\mathrm{B}_{k+1}^{-1} = \mathrm{B}_k^{-1} + \left(1 + \frac{y_k^T \mathrm{B}_k^{-1} y_k}{s_k^T y_k}\right) \frac{s_k s_k^T}{s_k^T y_k} - \frac{s_k y_k^T \mathrm{B}_k^{-1} + \mathrm{B}_k^{-1} y_k s_k^T}{s_k^T y_k}$$

*Shermann-Morrison-Woodbury* *Let* $A$ *is* $n \times n$ *invertible matrix,* $u, v \in \mathbb{R}^n$. *Then* $A + uv^T$ *is invertible iff* $1 + v^T A^{-1} u \neq 0$, *and*

$$(A + uv^T)^{-1} = A^{-1} - \frac{A^{-1} u v^T A^{-1}}{1 + v^T A^{-1} u}$$

# BFGS

- $B_{k+1}^{-1}$可以等价地写为

$$B_{k+1}^{-1}=\left(I-\frac{y_k s_k^T}{s_k^T y_k}\right)^T B_k^{-1}\left(I-\frac{y_k s_k^T}{s_k^T y_k}\right)+\frac{s_k s_k^T}{s_k^T y_k}$$

- 可以发现如果$B_k^{-1}$是正定的，那么$B_{k+1}^{-1}$的前一部分一定正定，再假设$s_k^T y_k > 0$，那么后半部分也是正定的
- 因此，只要问题的形式保证在迭代的每一步中， $s_k^T y_k > 0$都成立，那么BFGS的近似海森矩阵就是正定的，保证迭代朝着减小目标的方向进行

# BFGS算法

**Algorithm 3.3.4** Globalized BFGS method

**(S0)** Choose $x^0 \in \mathbb{R}^n, H_0 \in \mathbb{R}^{n\times n}$ symmetric positive definite, $\sigma \in (0,1/2), \rho \in (\sigma,1), \varepsilon \geq 0$ and put $k := 0$.

**(S1)** If $\|\nabla f(x^k)\| \leq \varepsilon$: STOP.

**(S2)** Determine $d^k$ as a solution of $H_k d = -\nabla f(x^k)$.

**(S3)** Determine $t_k > 0$ such that

$$f(x^k + t_k d^k) \leq f(x^k) + \sigma t_k \nabla f(x^k)^T d^k \quad \text{and} \quad \nabla f(x^k + t_k d^k)^T d^k \geq \rho \nabla f(x^k)^T d^k.$$

**(S4)** Put

$$\begin{aligned} x^{k+1} &:= x^k + t_k d^k, \\ s^k &:= x^{k+1} - x^k, \\ y^k &:= \nabla f(x^{k+1}) - \nabla f(x^k), \\ H_{k+1} &:= H_k + \frac{y^k (y^k)^T}{(y^k)^T s^k} - \frac{H_k s^k (s^k)^T H_k}{(s^k)^T H_k s^k}. \end{aligned}$$

**(S5)** Put $k \leftarrow k+1$ and go to **(S1)**.

# L-BFGS

- BFGS虽然能近似海森矩阵，但我们需要$n \times n$的空间来储存稠密的$\mathrm{B}_k^{-1}$，在问题规模比较大时这是难以接受的
- L-BFGS是BFGS算法的省内存版本，只保留最近$m$次迭代的更新$\Delta\mathrm{B}_k$，而不是存储整个稠密矩阵，相当于使用秩为$2m$的低秩矩阵近似$\mathrm{B}_k$，与SVD压缩矩阵的思想相同
- 因此，L-BFGS将每步迭代储存和计算的开销降到了$O(n)$，代价是海森矩阵的近似更不准确了
- 由于其计算和储存的优势，L-BFGS是很多数值、工业软件的默认优化算法

# 回想：不动点迭代也是一种拟牛顿法

- 如果我们将求解$\mathrm{A}x = b$等价为下面的优化问题（假设A正定）：

$$\min f(x) = \frac{1}{2} x^T \mathrm{A} x - b^T x$$

- 那么有

$$\nabla f(x) = \mathrm{A}x - b, \nabla^2 f = \mathrm{A}$$

- 联系Jacobi/GS迭代的形式

$$x_{k+1} = x_k + \mathrm{M}^{-1}(b - \mathrm{A}x_k) = x_k - \mathrm{M}^{-1} \nabla f(x_k)$$

- 可以发现不动点迭代是使用M近似海森矩阵的拟牛顿法

# 带约束优化
# Constrained Optimization

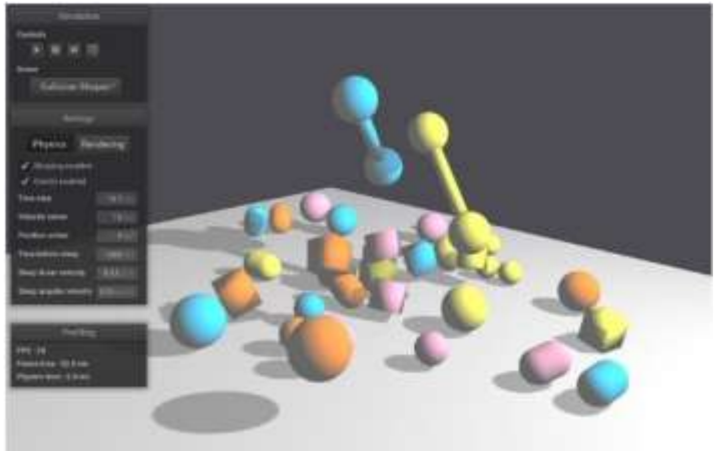
# 带约束优化

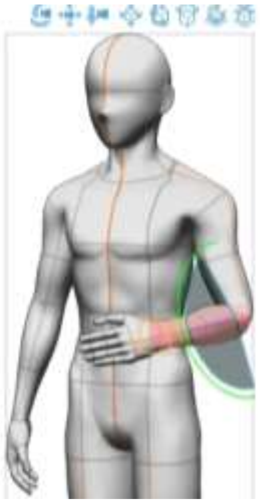
$$\min f(x)$$
$$x \in C$$

- $x \in \mathbb{R}^n$是连续的向量
- $C$是约束，分为等式约束和不等式约束
$$h_i(x) = 0, g_i(x) \leq 0$$
- $f(x)$连续可微

# 常见约束

- 碰撞 $d(A,B) \geq 0$
- 关节约束 $x_A = x_B$
- 关节角度限制 $\theta_{min} \leq \theta \leq \theta_{max}$
- 形变无反转 $\det \frac{\partial f}{\partial x} \geq 0$
- 归一化 $\sum_i p_i = 1$
- …





# 等式约束的最优条件

- 为了简单我们首先考虑只有等式约束的情况

$$\min f(x)$$
$$s.t.\ g_i(x) = 0, i = 1, \cdots, n$$

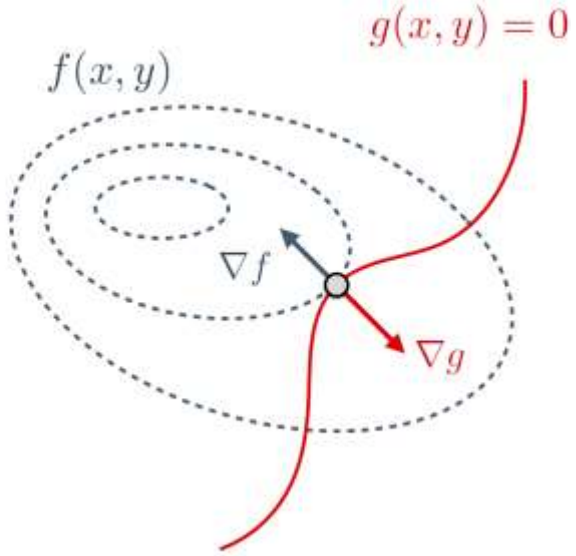
- 达到最优时应该满足什么条件?

# 等式约束的最优条件

- 继续简化考虑，假设在二维只有一个约束 $g(x,y)=0$
- 可行解对应的是$g(x,y)$的0等值线
- 当$f(x,y)$取到最小时，应该$f(x,y)$的梯度$\nabla f$垂直于等值线，否则我们沿着等值线方向走有可能让$f$进一步下降
- 换句话说，$\nabla f$应该与$\nabla g$平行，也即是说存在$\lambda \in \mathbb{R}$满足

$$\nabla f = \lambda \nabla g$$

- 当等式约束不只一个时，可行解空间就变成高维流形，在最优点依然要满足上面梯度的垂直关系



# 等式约束的最优条件

- 一般来说，对于带等式约束的优化问题

$$\min f(x)$$
$$s.t.\ g_i(x) = 0, i = 1, \cdots, n$$

- 在最优点$x^*$应该满足

$$g_i(x^*) = 0$$
$$\exists \lambda_i, s.t. \nabla f(x^*) = \sum_i \lambda_i \nabla g_i(x^*)$$

# 拉格朗日乘子 (Lagrange Multiplier)

$$\min f(x)$$
$$s.t.\ g_i(x) = 0, i = 1, \cdots, n$$

- 对于这样的带约束优化问题，我们定义其对应的拉格朗日函数为

$$L(x, \lambda_1, \cdots, \lambda_n) = f(x) - \sum_i \lambda_i g_i(x)$$

- 那么刚才我们推出的最优解应该满足的条件可以被简写为

$$\nabla L = 0$$

- 为什么?

# 拉格朗日乘子 (Lagrange Multiplier)

$$L(x, \lambda_1, \cdots, \lambda_n) = f(x) - \sum_i \lambda_i g_i(x)$$

- $\nabla_{\lambda_i} L = \frac{\partial L}{\partial \lambda_i} = -g_i(x) = 0$
- $\nabla_x L = \nabla_x L - \sum_i \lambda_i \nabla_x g_i(x) = 0$
- 这两个条件正好对应最优点的要求，因此$\nabla L = 0$就是最优条件
- 那是否意味着我们可以直接最小化$L$就能得到原问题的解?
- 并不是！如果最小化$L$，只要$g_i(x)$存在不为0的值，我们就能取$\lambda_i$为负无穷或正无穷使得$L \to -\infty$，最小化$L$没有意义

# Min-Max问题

$$L(x,\lambda_1,\cdots,\lambda_n)=f(x)-\sum_i \lambda_i g_i(x)$$

- 与带约束的原问题等价的拉格朗日优化问题是一个min-max问题：
$$\min_x \max_{\lambda_i} L(x,\lambda_1,\cdots,\lambda_n)$$
- 将$x$和$\lambda_i$想象成两组玩家在博弈，$x$的目标是最小化$L$，$\lambda_i$的目标是最大化$L$
- 如果$x$的选择让$g_i(x)\neq 0$，那么$\lambda_i$总可以让$L$趋向负无穷，$x$就输了，因此$x$只能选择保证$g_i(x)=0$，同时最小化$f(x)$，这也就是原本的带约束的优化问题
- 在二者博弈达到平衡点时，自然满足$\nabla L=0$

# 求解拉格朗日函数

$$\min_{x} \max_{\lambda_i} L(x, \lambda_1, \cdots, \lambda_n)$$

- 由于min-max的存在，我们无法直接用前面介绍的无约束优化方法优化拉格朗日函数
- 因此，我们只能尝试迭代求解平衡条件：

$$g_i(x^*) = 0$$

$$\nabla f(x^*) = \sum_i \lambda_i \nabla g_i(x^*)$$

- 比如使用牛顿法

# 牛顿法解方程

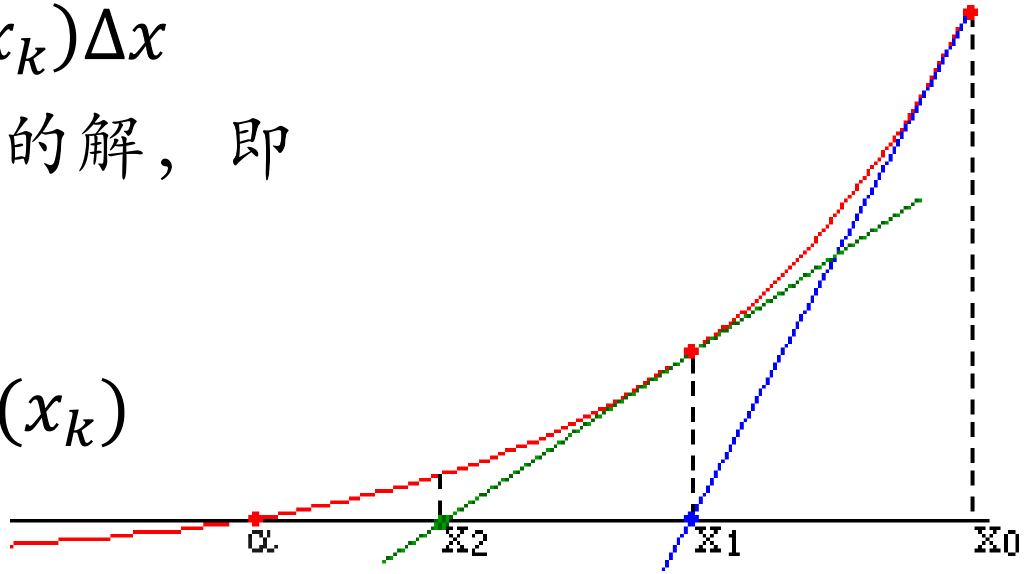
- 我们前面介绍了优化目标函数$f(x)$的牛顿法，换个角度看，我们其实是在求解$p(x)=\nabla f(x)=0$的方程
- 如果直接考虑$p(x)=0$的方程，在迭代到$x_k$时，我们在局部对$p(x)$做一阶展开（对应$f(x)$的二阶展开）：

$$p(x_k+\Delta x)=p(x_k)+\nabla p(x_k)\Delta x$$

- 牛顿法要求每次迭代找到一阶近似下的解，即$p(x_{k+1})=p(x_k+\Delta x)=0$，得到

$$\nabla p(x_k)\Delta x=-p(x_k)$$

$$x_{k+1}=x_k-\left(\nabla p(x_k)\right)^{-1}p(x_k)$$



# 使用牛顿法求解拉格朗日函数

- 将$\lambda_i$写为一个矢量$\lambda \in \mathbb{R}^n$，对应拉格朗日函数为$L = f - \lambda^T g$，最优条件为

$$g = 0, \qquad \nabla f - \lambda^T \nabla g = 0$$

- 在迭代的第$k$步，$\Delta x_k, \Delta\lambda_k$应该满足

$$(\nabla^2 f_k - \lambda^T \nabla^2 g_k)\Delta x_k - \nabla g_k^T \Delta\lambda_k = -(\nabla f_k - \lambda_k^T \nabla g_k)$$
$$\nabla g_k \Delta x_k = -g_k$$

- 写为矩阵方程的形式

$$\begin{pmatrix} \nabla^2 f_k - \lambda^T \nabla^2 g_k & -\nabla g_k^T \\ -\nabla g_k & 0 \end{pmatrix} \begin{pmatrix} \Delta x_k \\ \Delta\lambda_k \end{pmatrix} = \begin{pmatrix} -(\nabla f_k - \lambda_k^T \nabla g_k) \\ g_k \end{pmatrix}$$

- 这是一个对称不定的矩阵方程，可以使用LU，MINRES等求解

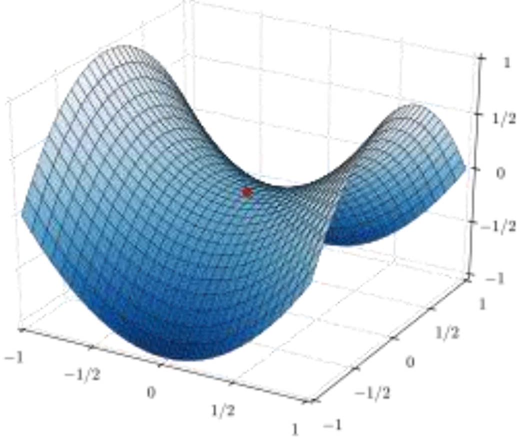
# 对偶问题 (Dual Problem)

- 直接求解拉格朗日函数需要同时求解$x$和$\lambda$，自由度更多了，如果使用牛顿法最终的矩阵也不是正定的，限制我们使用更快的矩阵求解器
- 事实上，我们可以将原问题化为对偶问题，只优化$\lambda$，就能得到跟原问题一样的最优解，而这只需要我们交换min-max的顺序

$$\min_x \max_\lambda L \to \max_\lambda \min_x L$$

# 对偶 (Duality)

- 对于一般的函数$L(x,\lambda)$，我们总是有下面的弱对偶关系成立
$$\min_x \max_\lambda L \geq \max_\lambda \min_x L$$
- 如果目标函数$f(x)$和约束$g(x)$是凸的，那么强对偶条件成立
$$\min_x \max_\lambda L = \max_\lambda \min_x L$$
- 关于凸集和凸函数可以延伸出优化中的重要概念凸优化(Convex Optimization)，但是这里我们不展开，不妨先抛弃理论上的严谨性假设强对偶一定成立



# 对偶问题 (Dual Problem)

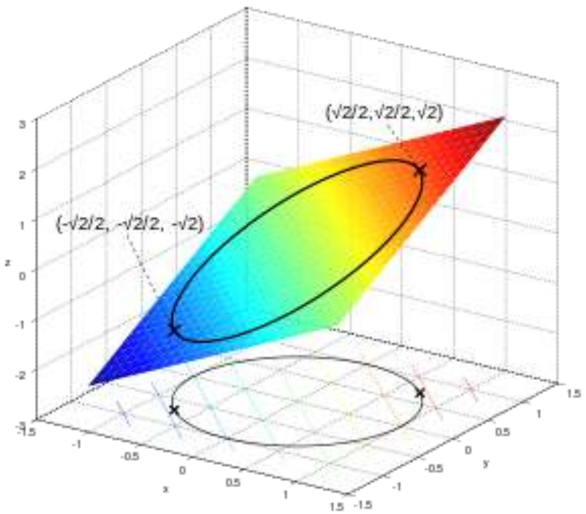
$$\min_x \max_\lambda L = \max_\lambda \min_x L$$

- 定义一个新的函数$G(\lambda) = \min_x L = \min_x f - \lambda^T g$
- 如果我们能够给出$G(\lambda)$的形式，那么原来带约束的优化问题就可以转化为下面的无约束优化问题

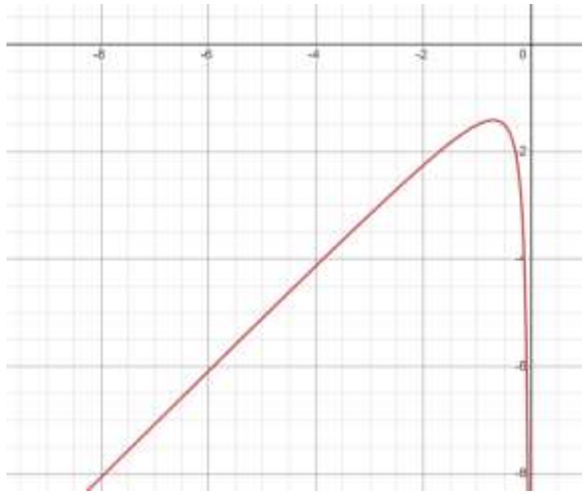
$$\max_\lambda G$$

- $G(\lambda)$称为对偶函数，对应的优化问题称为对偶问题，显然想比于原函数更容易求解

# 一个解析的对偶问题例子




- 假设我们要在约束$g(x,y)=x^2+y^2-1$上最小化$f(x,y)=x+y$，对应拉格朗日函数为
$$L=x+y-\lambda(x^2+y^2-1)$$
- 下面求对偶函数$G(\lambda)=\min_x L$
- 当$\lambda\geq 0$时，$L$可以取到$-\infty$，因此$G(\lambda)=-\infty$
- 当$\lambda<0$时，$L$为开口向上的二次函数，$x=y=\frac{1}{2\lambda}$时取到最小，此时$G(\lambda)=\lambda+\frac{1}{2\lambda}$
- $\lambda=-\frac{1}{\sqrt{2}}$时$G$可以取到最大值，此时$x=y=-\frac{\sqrt{2}}{2}$，可以从几何验证此时确实取到了$f$的最小值



# 对偶上升法 (Dual Ascent)

$$\max_{\lambda} G(\lambda) = \max_{\lambda} \min_{x} L(x, \lambda)$$

- 对偶上升法利用了对偶问题的形式，交替更新$x_k$和$\lambda_k$
- 在算法的每一步，我们首先固定$\lambda_k$，单独求解$x$的优化问题，相当于是计算$G(\lambda_k)$

$$x_{k+1} = \arg\min_{x} L(x, \lambda_k)$$

- 然后我们再用梯度上升更新$\lambda_k$

$$\lambda_{k+1} = \lambda_k + \alpha_k \nabla_{\lambda} G$$

其中$\alpha_k$是步长

# 对偶上升法 (Dual Ascent)

- 注意$G(\lambda) = L(x^*, \lambda)$其中$x^*$是极值点满足$\frac{\partial L}{\partial x^*} = 0$
- 因此$\nabla_\lambda G = \frac{\partial L(x^*,\lambda)}{\partial \lambda} = \frac{\partial L}{\partial x^*}\frac{\partial x^*}{\partial \lambda} + \frac{\partial L}{\partial \lambda} = \frac{\partial L}{\partial \lambda} = -g$
- 于是$\lambda_k$的更新表达式为

$$\lambda_{k+1} = \lambda_k - \alpha_k g(x_{k+1})$$

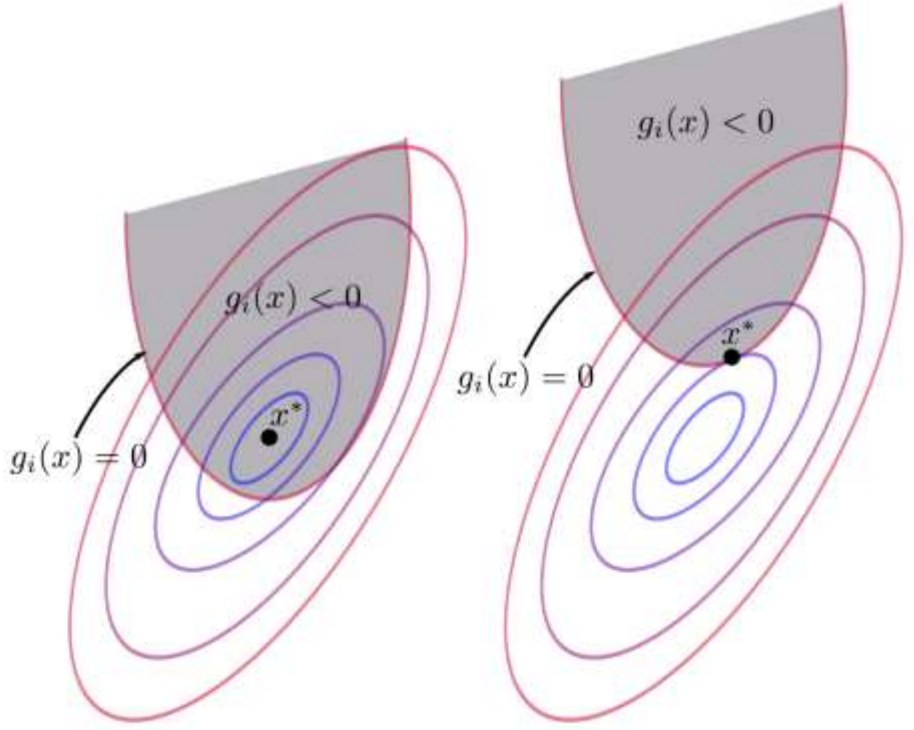
# 罚函数法 (Penalty Method)

- 相比于拉格朗日函数形式的优化方法，罚函数法更简单粗暴，将约束直接加入到优化目标中:

$$\min f(x) + \frac{\mu}{2} g(x)^T g(x)$$

- 其中$\mu$是给定的约束强度，越大表达约束越强
- 罚函数法无法保证约束一定满足，当$\mu$比较小的时候$g(x)$可能离0偏差较远，而$\mu$太大又会导致优化困难，因此需要调节参数达到最佳平衡
- 除了平方形式的罚函数，我们还可以选择对数函数等形式满足不同的需求

# 不等式约束




不等式约束相比等式约束处理更为复杂

关键在于不等式约束只有在变成等式约束时（约束边界）才会影响最优解

# 不等式约束

- 对于一般形式的带约束优化问题

$$\min f(x), s.t. h(x) = 0, g(x) \leq 0$$

- 我们依然可以定义对应的拉格朗日函数

$$L(x, \lambda, \mu) = f(x) + \lambda^T h(x) + \mu^T g(x)$$

- 只不过此时的min-max问题也要变成带约束的min-max问题

$$\min_x \max_\lambda \max_{\mu \geq 0} L(x, \lambda, \mu)$$

- 相对应的我们可以导出最优条件、对偶问题和对偶优化算法，这里限于篇幅就不展开了

# 总结

- 无约束的优化问题
  - 梯度下降
  - 牛顿法
  - 拟牛顿法
- 等式约束的优化问题
  - 拉格朗日函数与对偶
  - 牛顿法
  - 对偶上升法

# 参考材料

- An Introduction to Optimization Techniques in Computer Graphics
- Introduction to Mathematical Optimization with Python
- https://cmazzaanthony.github.io/coptim/index.html
- BFGS in a Nutshell: An Introduction to Quasi-Newton Methods
- https://esslab.jp/~ess/en/teaching/2022/cgo/
- https://github.com/Juyong/GeometricOptimization

# 谢谢