深蓝学院
shenlanxueyuan.com

Numerical Optimization in Robotics

Lecture 2: Unconstrained Optimization

无约束优化 Hession free方法

主讲人 Zhepei Wang

Ph.D. in Robotics
Zhejiang University

Recommended Reading List

1. Liu DC, Nocedal J. On the limited memory BFGS method for large scale optimization. Mathematical Programming. 1989 Aug;45(1):503-28.
2. Li DH, Fukushima M. On the global convergence of the BFGS method for nonconvex unconstrained optimization problems. SIAM Journal on Optimization. 2001;11(4):1054-64.
3. Lewis AS, Overton ML. Nonsmooth optimization via quasi-Newton methods. Mathematical Programming. 2013 Oct;141(1):135-63.
4. Nash SG, Nocedal J. A numerical study of the limited memory BFGS method and the truncated-Newton method for large scale optimization. SIAM Journal on Optimization. 1991 Aug;1(3):358-72.
5. Shewchuk JR. An introduction to the conjugate gradient method without the agonizing pain.
6. Nash SG. Preconditioning of truncated-Newton methods. SIAM Journal on Scientific and Statistical Computing. 1985 Jul;6(3):599-616.

Recommended Reading List

Hessian-Free Optimization

BFGS-Type Quasi-Newton Methods

1. Liu DC, Nocedal J. On the limited memory BFGS method for large scale optimization. Mathematical Programming. 1989 Aug;45(1):503-28.
2. Li DH, Fukushima M. On the global convergence of the BFGS method for nonconvex unconstrained optimization problems. SIAM Journal on Optimization. 2001;11(4):1054-64.
3. Lewis AS, Overton ML. Nonsmooth optimization via quasi-Newton methods. Mathematical Programming. 2013 Oct;141(1):135-63.
4. Nash SG, Nocedal J. A numerical study of the limited memory BFGS method and the truncated-Newton method for large scale optimization. SIAM Journal on Optimization. 1991 Aug;1(3):358-72.

Newton-CG Method

用于求解线性方程

5. Shewchuk JR. An introduction to the conjugate gradient method without the agonizing pain.
6. Nash SG. Preconditioning of truncated-Newton methods. SIAM Journal on Scientific and Statistical Computing. 1985 Jul;6(3):599-616.

牛顿法的特点是：收敛速度快，迭代次数少，但是当Hession矩阵很稠密时， 每次迭代的计算量很大。随着数据规模的增大，那么hession矩阵会越来越大，需要存储的空间会增多，计算量也会增大，有时候大到不可计算，所以随着海量数据，牛顿法不再使用

拟牛顿法是构造近似矩阵，并用它取代牛顿法中Hession矩阵的逆

Quasi-Newton Methods

Recall Newton's Method

回顾牛顿方法

By second-order Taylor expansion,

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

Minimizing quadratic approximation

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

$$\Longrightarrow \boldsymbol{x} = \boldsymbol{x}_k - \left[\nabla^2 f(\boldsymbol{x}_k)\right]^{-1}\nabla f(\boldsymbol{x}_k)$$

hession矩阵的逆

provided $\nabla^2 f(\boldsymbol{x}_k) \succ \boldsymbol{O}$ Positive Definite (PD)

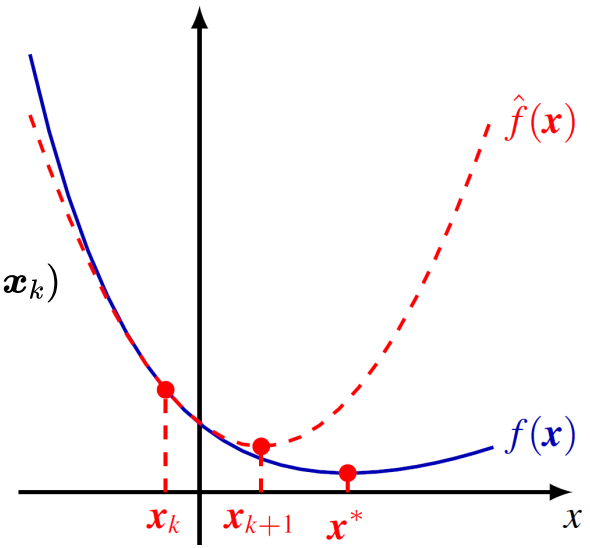
需要保证hession矩阵严格正定

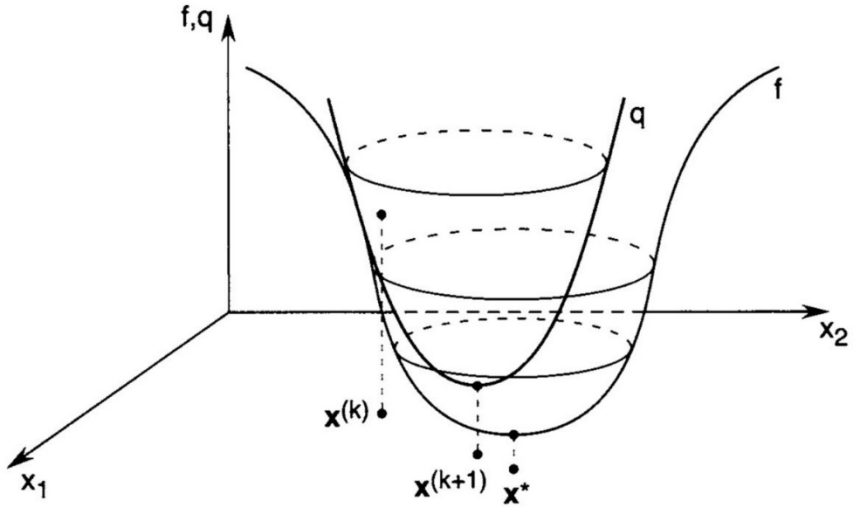
Newton step

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \left[\nabla^2 f(\boldsymbol{x}_k)\right]^{-1}\nabla f(\boldsymbol{x}_k)$$

Note. If the function is quadratic, then Newton's method gets to the optimum in a single step.

如果函数本身是一个二次型，得出的曲率就是自身，可以一步迭代

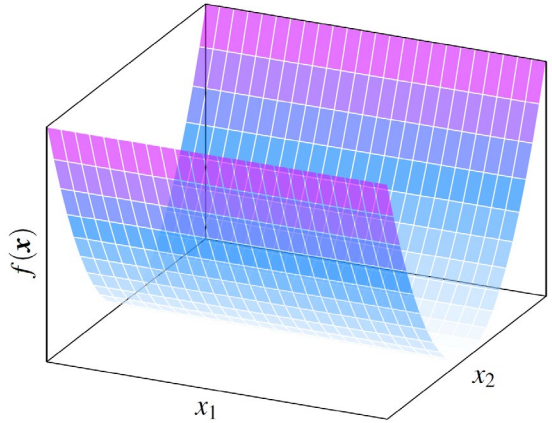




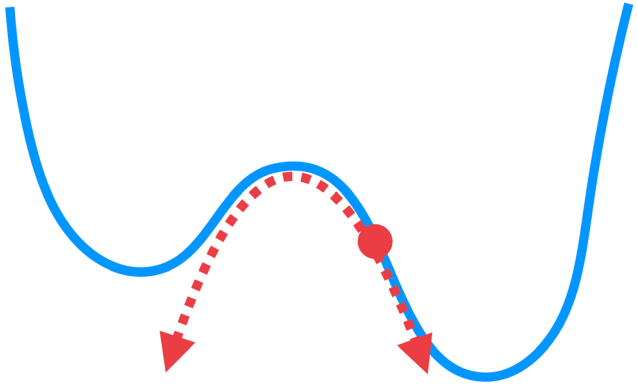
Recall Newton's Method

Drawbacks: In practice Hessian can be singular and indefinite

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \left[\nabla^2 f(\boldsymbol{x}_k)\right]^{-1} \nabla f(\boldsymbol{x}_k)$$



Singular Hessian!



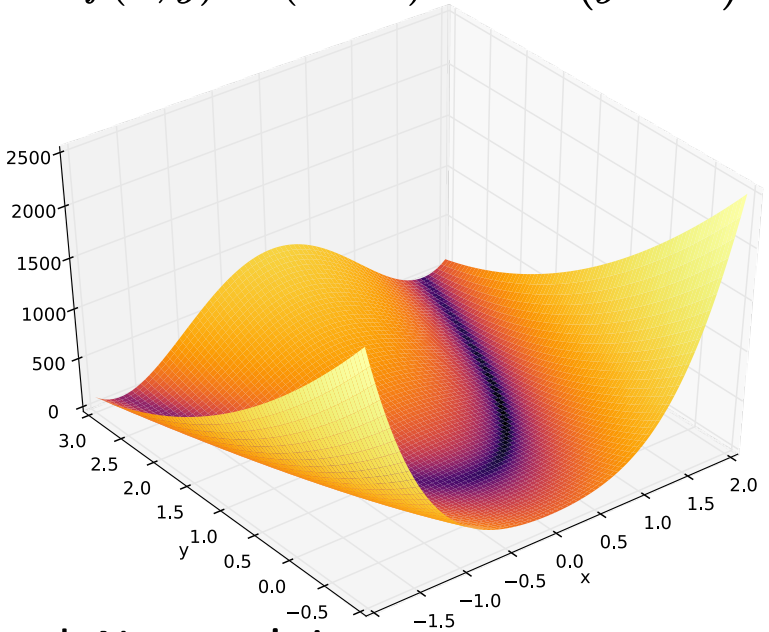
If start at a point of negative curvature, the Newton goes up!

(negative) search direction: must form an acute angle with the gradient

Why Quasi-Newton Methods

Rosenbrock banana function

$$f(x, y) = (1 - x)^2 + 100\left(y - x^2\right)^2$$

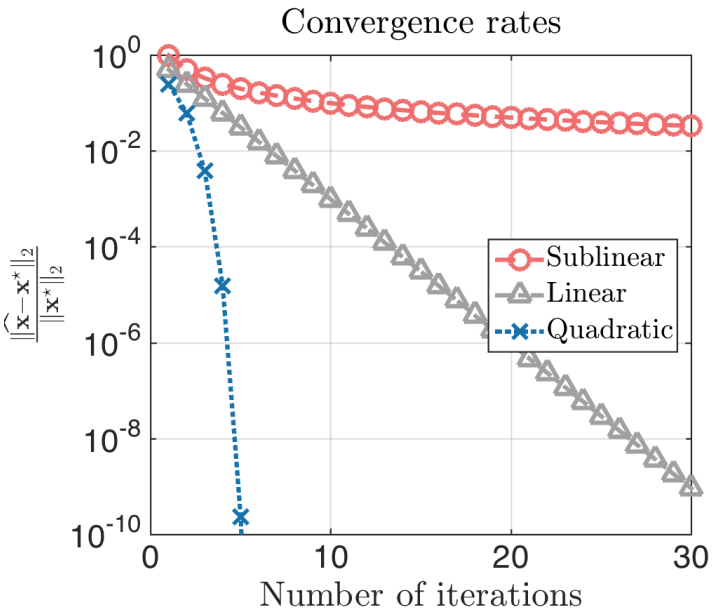


- Gradient descent is cheap
  - Compute partials: $O(N)$
- Newton's method is expensive
  - Compute mixed partials: $O(N^2)$
  - Linear solver: $O(N^3)$
- Other problems
  - When quadratic approximation is bad, Newton's is a waste
  - Hessian becomes poorly conditioned
  - Nonconvex problems: indefinite Hessian = ascent step

Why Quasi-Newton Methods

Normally, for convex smooth functions, we have

Newton's Method Gradient Methods



$$e^k = x^k - x^*$$

Linear $\quad \left\|e^{k+1}\right\| = C\left\|e^k\right\|$

Quadratic $\quad \left\|e^{k+1}\right\| = C\left\|e^k\right\|^2$

Superlinear $\quad \left\|e^{k+1}\right\| = C\left\|e^k\right\|^p, p > 1$

As good as quadratic (why?)

Attain super-linear performance without all the fuss

Quasi-Newton Methods

Newton approximation (PD Hessian *H*)

$$f(x) - f(x^k) \approx (x - x^k)^T g + \frac{1}{2}(x - x^k)^T H (x - x^k)$$

Solve $H^k d^k = -g^k$

Quasi-Newton approximation

$$f(x) - f(x^k) \approx (x - x^k)^T g^k + \frac{1}{2}(x - x^k)^T M^k (x - x^k)$$

Solve $M^k d^k = -g^k$ 自己设置一个M矩阵

What conditions should *M* satisfy?

- It should not need full 2nd-order derivatives
- The linear equations have closed-form solutions
- It should be lightweight and compact to store
- It must preserves descent directions
- It should contain curvature information (local quadratic apporx.)

Quasi-Newton Methods

When is descent direction preserved?

$$M^k d^k = -g^k$$

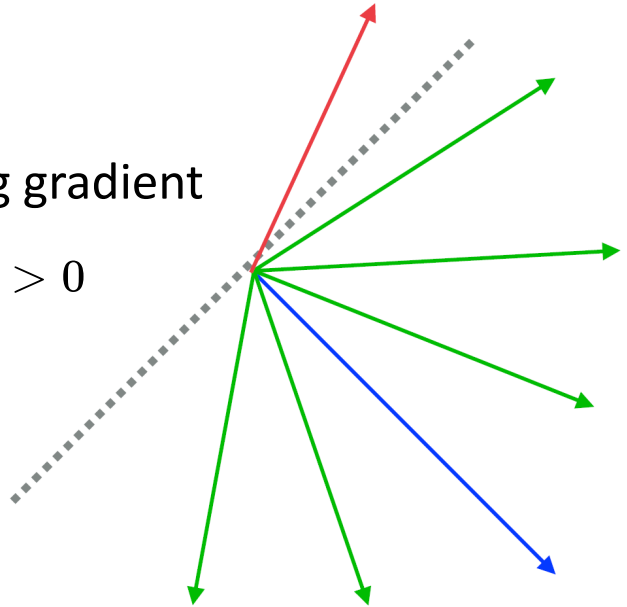
Negative gradient step

Search direction

Search direction makes acute angle with neg gradient

$$\langle -g, -M^{-1}g \rangle = \langle g, M^{-1}g \rangle = g^T M^{-1} g > 0$$

Must be PD (Positive Definite)



Quasi-Newton Methods

Which kind of curvature info is needed?

$$\nabla f(x) - \nabla f(y) \approx H(x - y)$$

梯度泰勒展开

$$\Delta x = x^{k+1} - x^k$$
$$\Delta g = \nabla f\left(x^{k+1}\right) - \nabla f\left(x^k\right)$$

Secant condition

$$\Delta g \approx M^{k+1} \Delta x$$

or

$$\Delta x \approx B^{k+1} \Delta g, \; M^{k+1} B^{k+1} = I$$

Construct *B* directly from gradient/variable diff

Quasi-Newton Methods

Infinitely many *B* satisfy the secant condition?

$$\Delta x = B^{k+1}\Delta g$$

How to choose the best one?

- Assume the function is convex
- Assume an initial guess is given

$$\min_{B} \quad \|B - B^k\|^2$$
$$\text{s.t.} \quad B = B^{\mathrm{T}}$$
$$\Delta x = B\Delta g$$

scale invariant

$$\min_{B} \quad \|H^{\frac{1}{2}}(B - B^k)H^{\frac{1}{2}}\|^2$$
$$\text{s.t.} \quad B = B^{\mathrm{T}}$$
$$\Delta x = B\Delta g$$

$$H = \int_0^1 \nabla^2 f\left[(1-\tau)x^k + \tau x^{k+1}\right] d\tau$$

Quasi-Newton Methods

Recover the curvature (approx. inv. Hessian) from gradients

Broyden-Fletcher-Goldfarb-Shanno (BFGS) update

$$B^{k+1} = \left(I - \frac{\Delta x \Delta g^T}{\Delta g^T \Delta x}\right) B^k \left(I - \frac{\Delta g \Delta x^T}{\Delta g^T \Delta x}\right) + \frac{\Delta x \Delta x^T}{\Delta g^T \Delta x}$$

$$B^0 = I,\ \Delta x = x^{k+1} - x^k,\ \Delta g = \nabla f\left(x^{k+1}\right) - \nabla f\left(x^k\right)$$

What conditions should *M = 1/B* satisfy?

- It should not need full 2nd-order derivatives ✓
- The linear equations have closed-form solutions ✓
- It should be lightweight and compact to store ✓
- It must preserves descent directions ? strict convexity (necessary?)
- It should contain curvature information ✓

BFGS update preserves PD if $\Delta g^T \Delta x > 0$ 向量内积>0

Quasi-Newton Methods

BFGS method for strictly convex functions

initialize $x^0,\ g^0 \leftarrow \nabla f(x^0),\ B^0 \leftarrow I,\ k \leftarrow 0$

while $\|g^k\| > \delta$ do

$d \leftarrow -B^k g^k$

$t \leftarrow$ backtracking line search (Armijo)

$x^{k+1} \leftarrow x^k + td$

$g^{k+1} \leftarrow \nabla f(x^{k+1})$

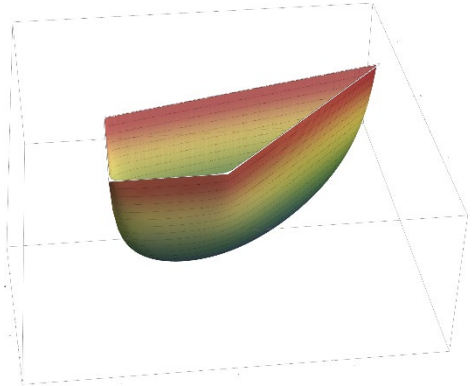
$B^{k+1} \leftarrow \mathrm{BFGS}(B^k, g^{k+1} - g^k, x^{k+1} - x^k)$

$k \leftarrow k + 1$

end while

return

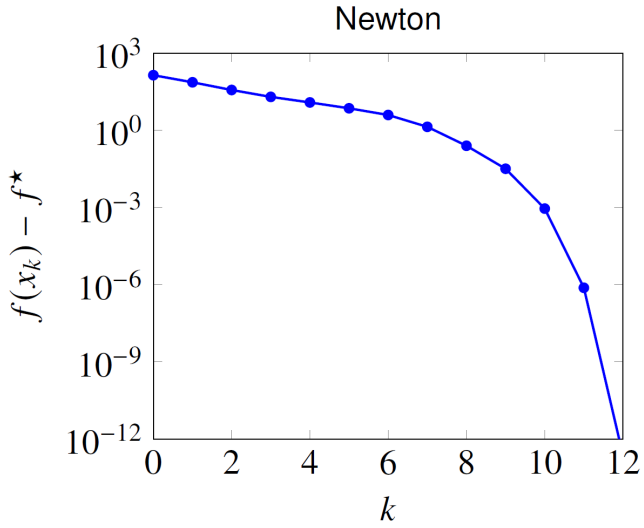
Quasi-Newton Methods

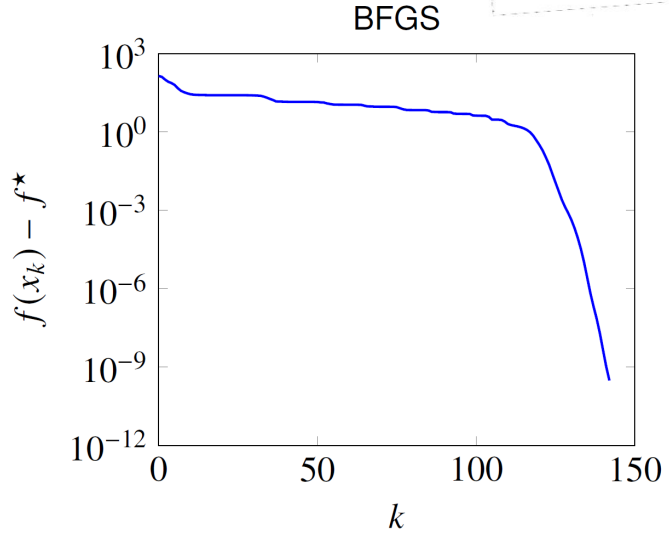


Example

$$f(x) = c^T x - \sum_{i=1}^{m} \log(b_i - a_i^T x), \; x \in \mathbb{R}^n$$

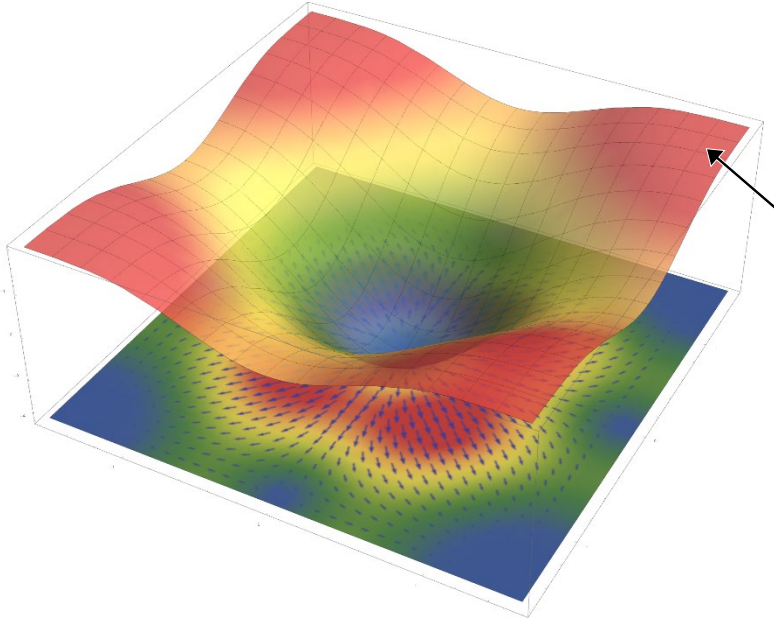
$$n = 100, \; m = 500$$





- cost per Newton iteration: $O(n^3)$
- cost per BFGS iteration: $O(n^2)$

Quasi-Newton Methods



- negative curvature
- nonconvex
- non-monotonic grad

Restrictions:

- Strict gradient monotonicity does not hold in general
- Curvature info far from optimum is less important
- Iteration cost is quadratic in dim
- Applicability to nonconvex function remains to be verified
- Applicability to nonsmooth function remains to be verified

How to apply to more nonconvex/nonsmooth functions?

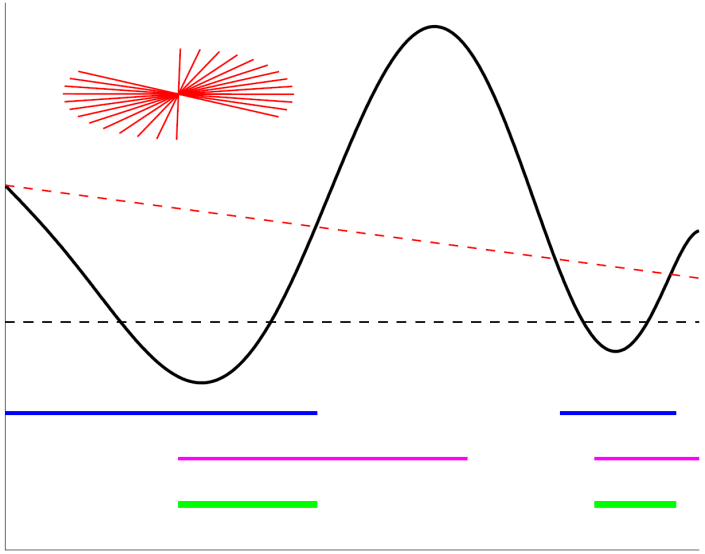
Quasi-Newton Methods

$\Delta g^T \Delta x > 0$? Ensure PD of approximation: Wolfe conditions!

1. weak Wolfe conditions $0 < c_1 < c_2 < 1 \quad \text{typically } c_1 = 10^{-4},\ c_2 = 0.9$

$$f(x^k) - f(x^k + \alpha d) \geq -c_1 \cdot \alpha d^{\mathrm{T}} \nabla f(x^k)$$ sufficient decrease condition

$$d^{\mathrm{T}} \nabla f(x^k + \alpha d) \geq c_2 \cdot d^{\mathrm{T}} \nabla f(x^k)$$ curvature condition



can prevent slow progress

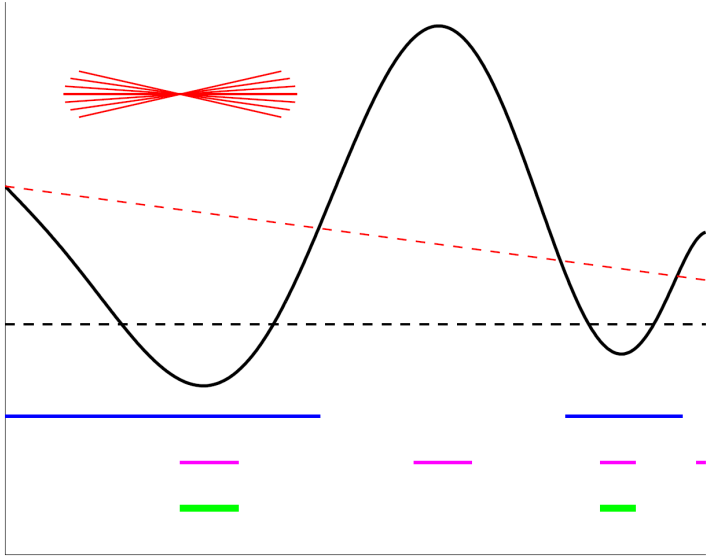
Quasi-Newton Methods

$\Delta g^T \Delta x > 0$? Ensure PD of approximation: Wolfe conditions!

2. strong Wolfe conditions $0 < c_1 < c_2 < 1 \quad \text{typically } c_1 = 10^{-4},\ c_2 = 0.9$

$$f(x^k) - f(x^k + \alpha d) \geq -c_1 \cdot \alpha d^{\mathrm{T}} \nabla f(x^k)$$ sufficient decrease condition

$$\left| d^{\mathrm{T}} \nabla f(x^k + \alpha d) \right| \leq \left| c_2 \cdot d^{\mathrm{T}} \nabla f(x^k) \right|$$ strong curvature condition



suppress oscillation

Quasi-Newton Methods

strong Wolfe conditions

⇩

weak Wolfe conditions

⇩

$\Delta g^T \Delta x > 0$

⇩

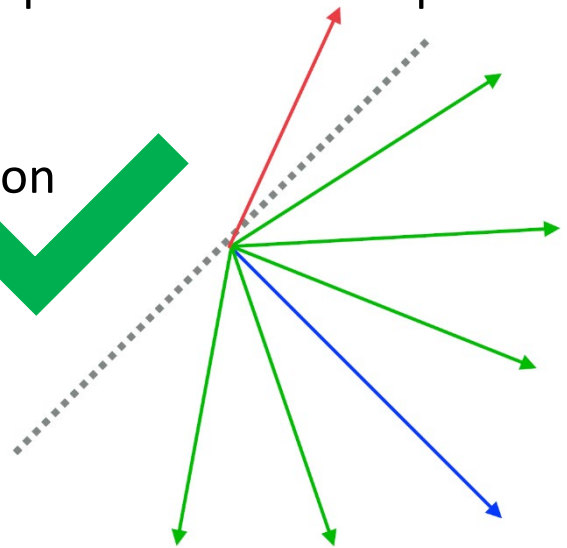
BFGS update is PD

⇗

descent direction

Inexact line search suffices (detailed later)

Various interpolation schemes presented



Quasi-Newton Methods

Wolfe conditions cannot guarantee the convergence of BFGS

The cautious update (Li and Fukushima 2001) with mild conditions is needed.

$$B^{k+1} = \begin{cases} \left(I - \dfrac{\Delta x \Delta g^T}{\Delta g^T \Delta x}\right) B^k \left(I - \dfrac{\Delta g \Delta x^T}{\Delta g^T \Delta x}\right) + \dfrac{\Delta x \Delta x^T}{\Delta g^T \Delta x} & \text{if } \Delta g^T \Delta x > \epsilon ||g_k|| \Delta x^T \Delta x, \ \ \epsilon = 10^{-6} \\ B^k & \text{otherwise} \end{cases}$$

The cautious update guarantees zero grad to be an accumulation point if

- The func has bounded sub-level sets
- The func has Lipschitz continuous grad

Quasi-Newton Methods

BFGS method for possibly nonconvex functions

initialize $x^0,\ g^0 \leftarrow \nabla f(x^0),\ B^0 \leftarrow I,\ k \leftarrow 0$

while $\|g^k\| > \delta$ do

$d \leftarrow -B^k g^k$

$t \leftarrow$ inexact line search (Wolfe)

$x^{k+1} \leftarrow x^k + td$

$g^{k+1} \leftarrow \nabla f(x^{k+1})$

$B^{k+1} \leftarrow \text{Cautious-BFGS}(B^k, g^{k+1} - g^k, x^{k+1} - x^k)$

$k \leftarrow k + 1$

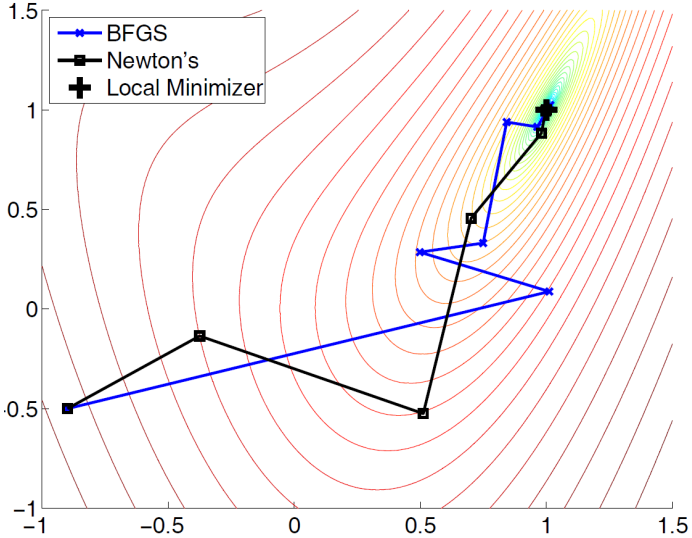
end while

return

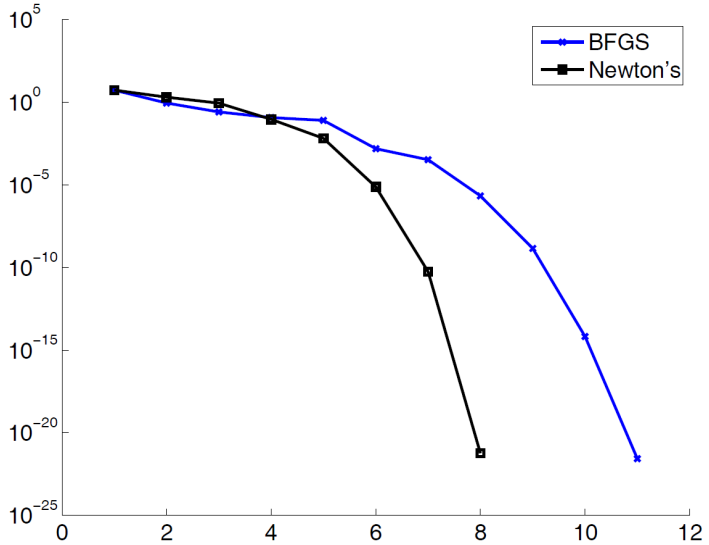
Quasi-Newton Methods

Example

$$f(x) = (1 - x_1)^2 + \left(x_2 - x_1^2\right)^2$$



(a) *Paths followed.*

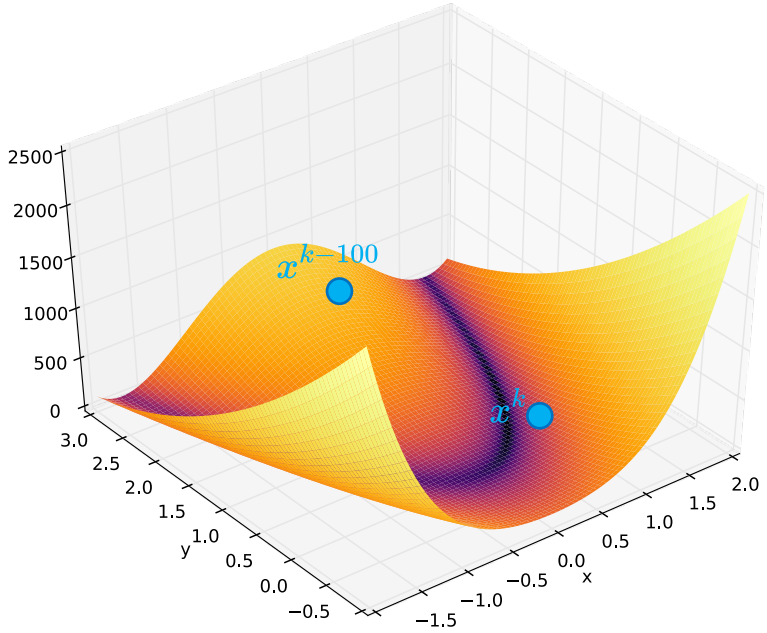


(b) *Function value at each iterate.*

Satisfactory convergence rate

Quasi-Newton Methods

保留历史的delta g 与 delta x



- Is the curvature info at x^{k-100} useful to iteration at x^k ?
- Is it possible to further reduce the $O(n^2)$ cost per iteration?

Only exploit last m+1 pairs of $\{x^k, g^k\}$!

Quasi-Newton Methods

Limited-memory BFGS (L-BFGS): do not store B^k explicitly

$$s^k = x^{k+1} - x^k, \quad y^k = g^{k+1} - g^k, \quad \rho^k = 1/\langle s^k, y^k \rangle \text{ where } \langle a, b \rangle := a^T b$$

- Instead we store up to m (e.g., m = 30) values of $s^k,\ y^k,\ \rho^k$

$B^{k+1} \leftarrow \text{BFGS}(B^k, g^{k+1} - g^k, x^{k+1} - x^k)$

$k \leftarrow k + 1$

repeat for m times $O(mn^2)$

$-B^k g^k = d$

$d \leftarrow g^k$ $O(mn)$

for $i = k-1, k-2, \ldots, k-m$

$\alpha^i \leftarrow \rho^i \langle s^i, d \rangle$

$d \leftarrow d - \alpha^i y^i$

end (for)

$\gamma \leftarrow \rho^{k-1} \langle y^{k-1}, y^{k-1} \rangle$

$d \leftarrow d \,/\, \gamma$

for $i = k-m, k-m+1, \ldots, k-1$

$\beta \leftarrow \rho^i \langle y^i, d \rangle$

$d \leftarrow d + s^i (\alpha^i - \beta)$

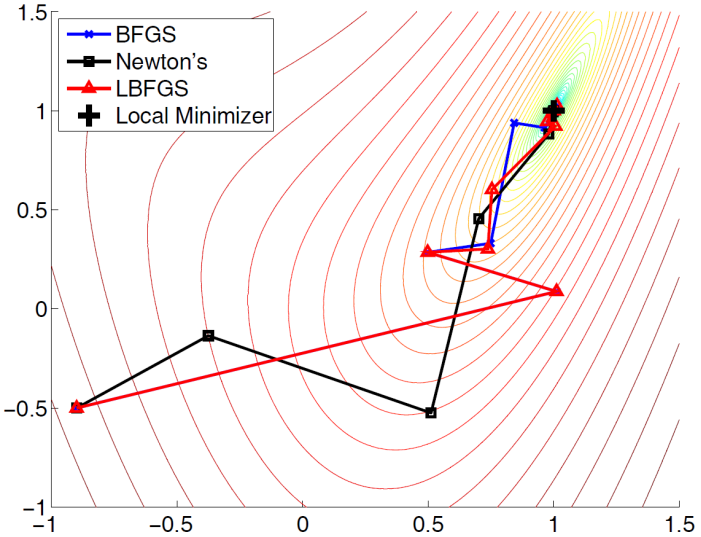
end (for)

return search direction d

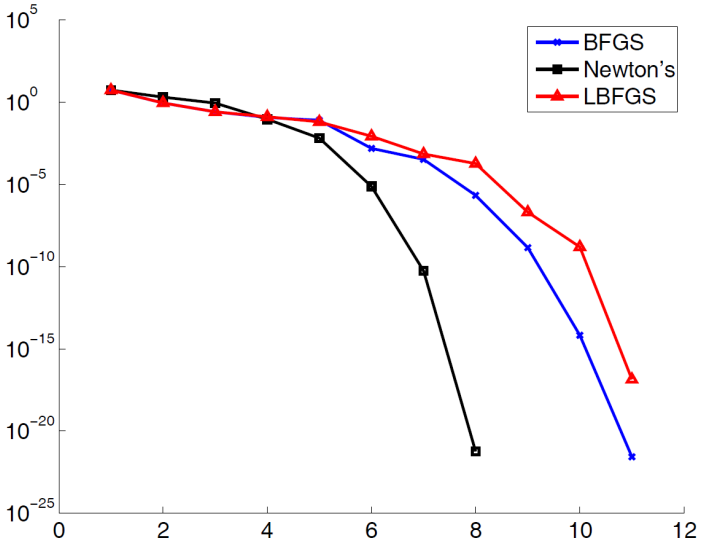
Quasi-Newton Methods

Example

$$f(x) = (1 - x_1)^2 + \left(x_2 - x_1^2\right)^2$$



(a) *Paths followed.*



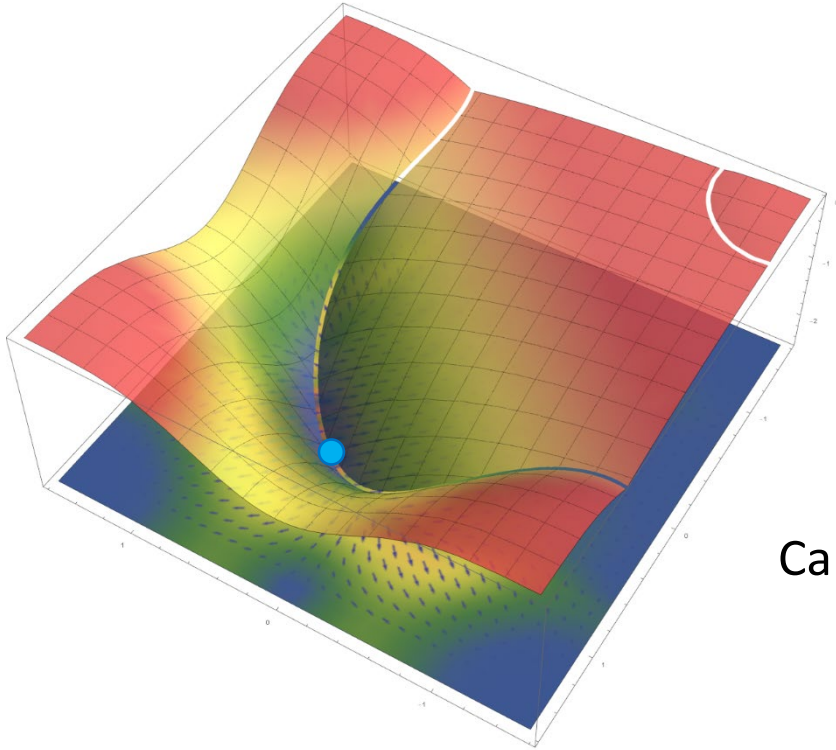
(b) *Function value at each iterate.*

	Newton's	BFGS	LBFGS
Work per iteration	$\mathcal{O}(n^3)$	$\mathcal{O}(n^2)$	$\mathcal{O}(mn)$

L-BFGS is almost the 1st choice for efficient smooth nonconvex opt

Quasi-Newton Methods

处理非光滑的场景



What if functions are nonsmooth?

Troubles with nonsmoothness :

- Gradient may not not exist 梯度不存在
- Negative sub-grad does not descent 方向不对
- Curvature can be very large

Can we apply L-BFGS to nonsmooth function?

Practically yes! (Lewis and Overton 2013)

收敛速率会降低

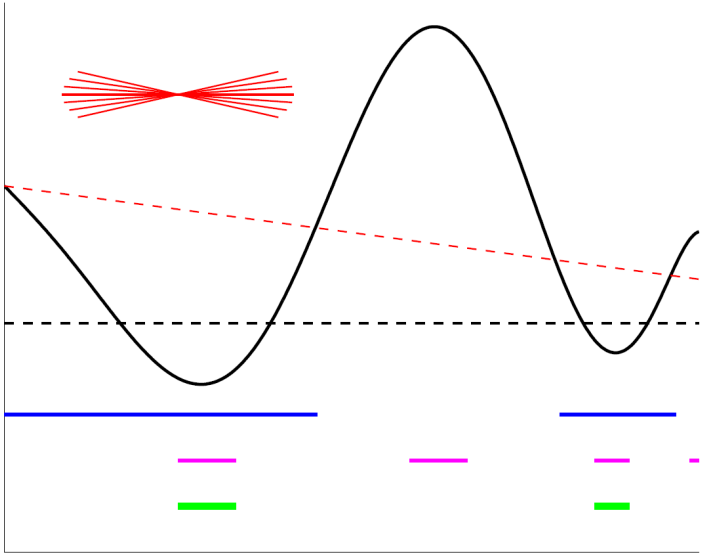
Quasi-Newton Methods

Key issues when applying L-BFGS to nonsmooth functions

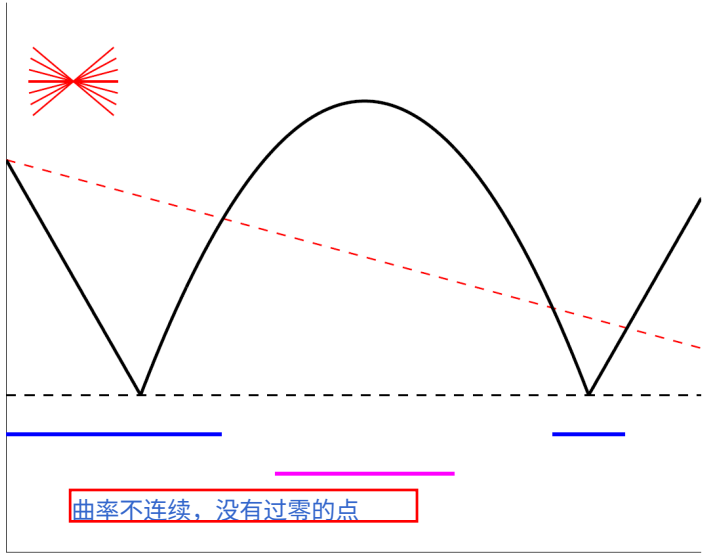
Recall: strong Wolfe conditions

$$f(x^k) - f(x^k + \alpha d) \geq -c_1 \cdot \alpha d^{\mathrm{T}} \nabla f(x^k)$$ sufficient decrease condition

$$\left| d^{\mathrm{T}} \nabla f(x^k + \alpha d) \right| \leq \left| c_2 \cdot d^{\mathrm{T}} \nabla f(x^k) \right|$$ strong curvature condition



(a) *Smooth* $\phi(\alpha)$.



(b) *nonsmooth* $\phi(\alpha)$. conditions fail!

Quasi-Newton Methods

Key issues when applying L-BFGS to nonsmooth functions

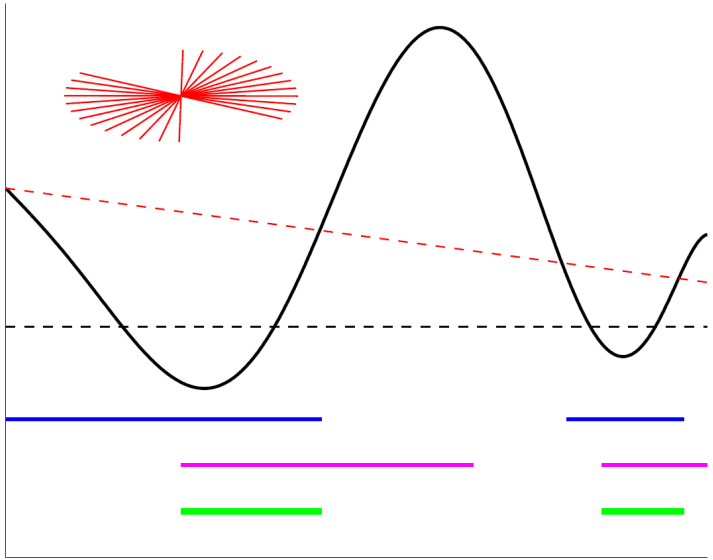
Recall: weak Wolfe conditions

$$f(x^k) - f(x^k + \alpha d) \geq -c_1 \cdot \alpha d^{\mathrm{T}} \nabla f(x^k)$$

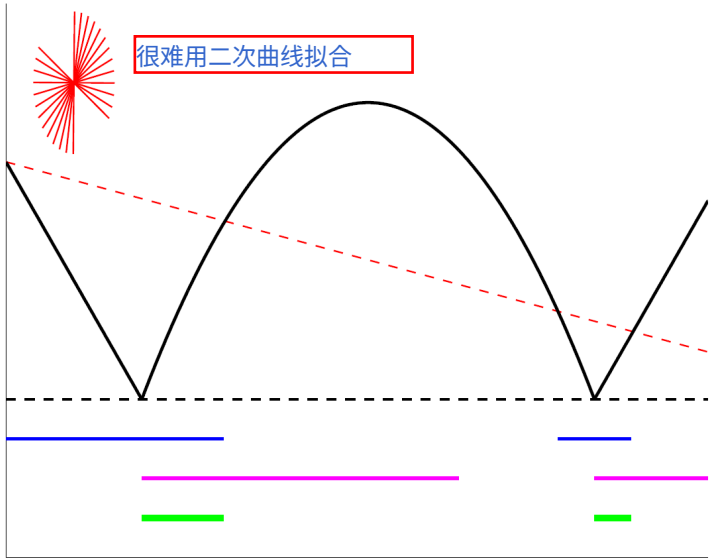
sufficient decrease condition

$$d^{\mathrm{T}} \nabla f(x^k + \alpha d) \geq c_2 \cdot d^{\mathrm{T}} \nabla f(x^k)$$

curvature condition



(a) *Smooth* $\phi(\alpha)$.



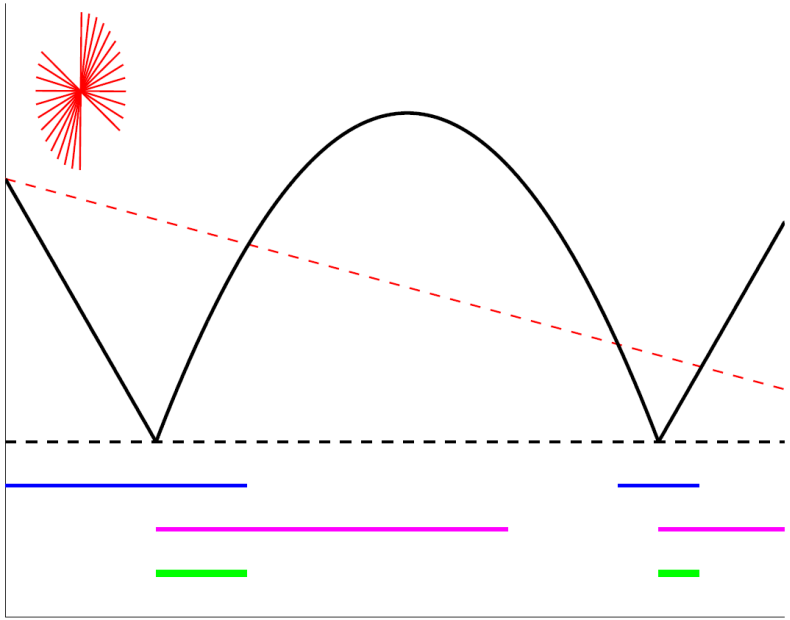
(b) *nonsmooth* $\phi(\alpha)$. conditions work!

Quasi-Newton Methods

weak Wolfe conditions should be used for nonsmooth functions

$$S(\alpha): f(x^k) - f(x^k + \alpha d) \geq -c_1 \cdot \alpha d^{\mathrm{T}} \nabla f(x^k)$$

$$C(\alpha): d^{\mathrm{T}} \nabla f(x^k + \alpha d) \geq c_2 \cdot d^{\mathrm{T}} \nabla f(x^k)$$



Lewis & Overton line search:

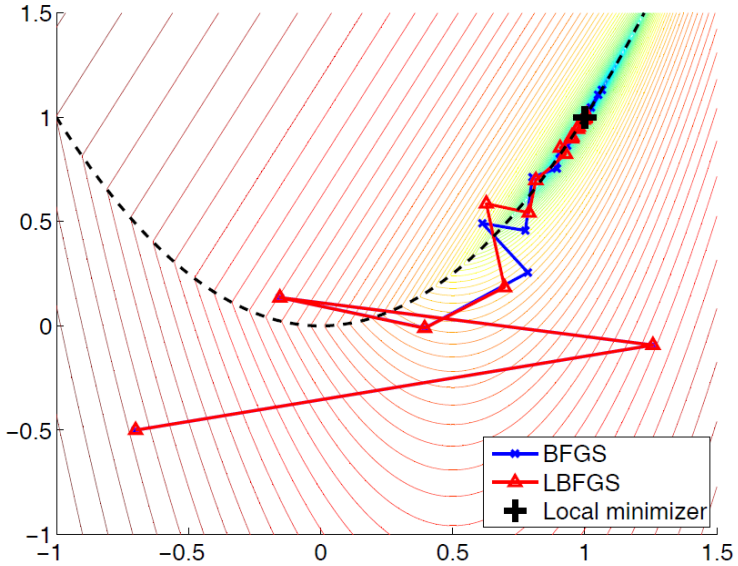
- weak Wolfe conditions
- no interpolation used

$l \leftarrow 0$
$u \leftarrow +\infty$
$\alpha \leftarrow 1$
repeat
  if $S(\alpha)$ fails
    $u \leftarrow \alpha$
  else if $C(\alpha)$ fails
    $l \leftarrow \alpha$
  else
    return α
  if $u < +\infty$
    $\alpha \leftarrow (l + u)/2$
  else
    $\alpha \leftarrow 2l$
end (repeat)

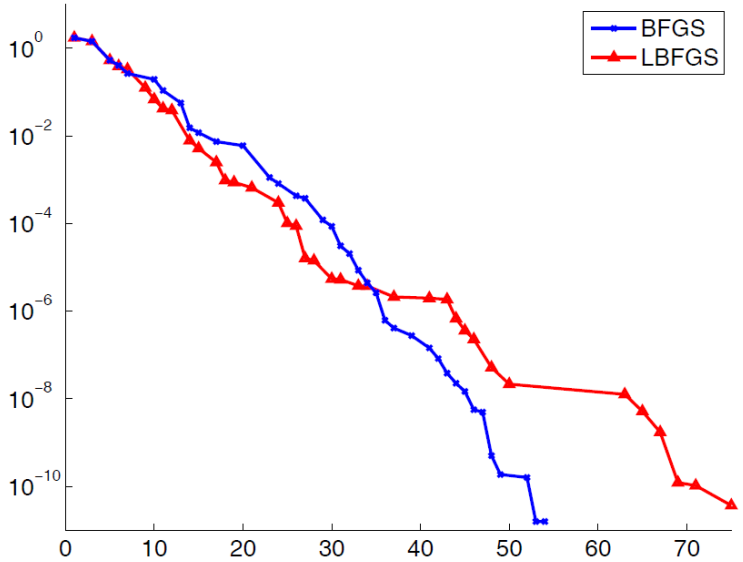
Quasi-Newton Methods

Example

$$f(x) = (1 - x_1)^2 + \left|x_2 - x_1^2\right|$$



(a) *Paths followed. The black dashed line indicates the curve* $x_2 = x_1^2$ *across which* f *is nonsmooth.*

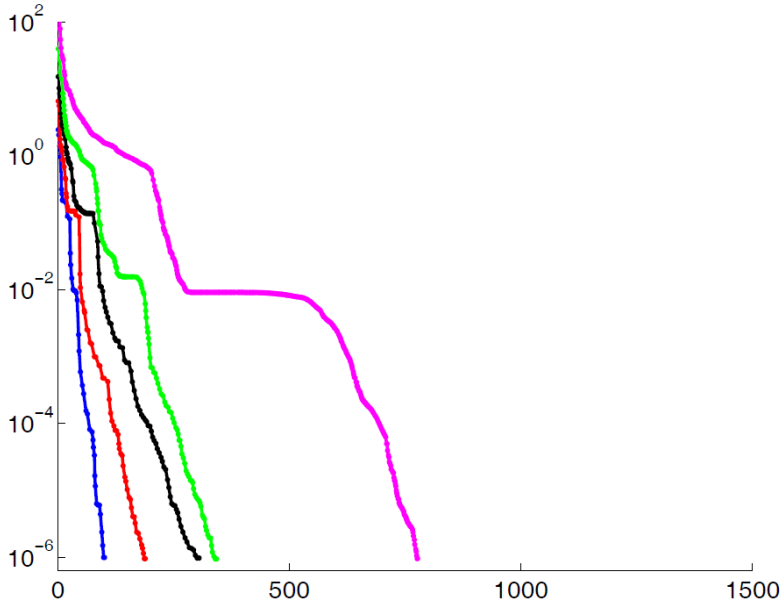


(b) *Function value at each iterate after the line search.*

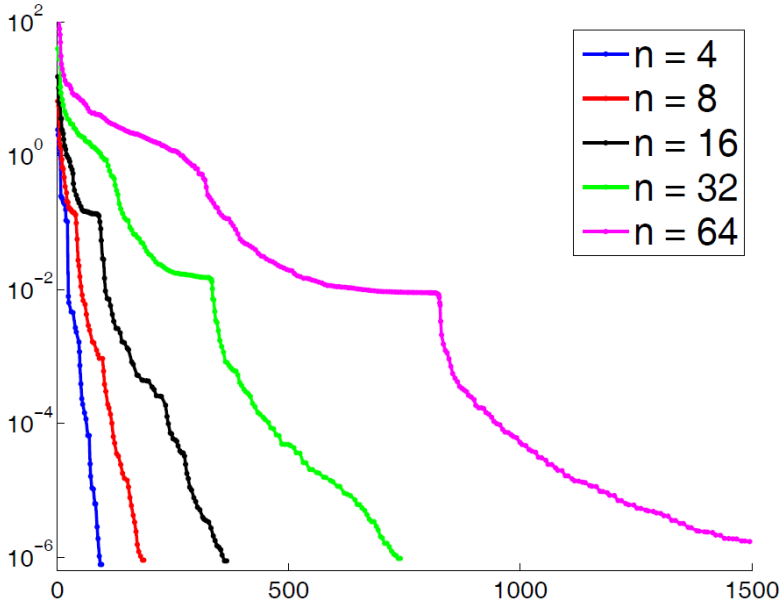
Quasi-Newton Methods

Example $f(x) = 4\|Ax\| + 3e_1^T Ax, \kappa(A) = n^2, \ f^* \equiv 0$

Ill-conditioned nonsmooth func whose *A* is PD and randomly generated



LBFGS with $m = 25$



BFGS

Quasi-Newton Methods

L-BFGS method for possibly nonconvex nonsmooth functions

initialize $x^0,\ g^0 \leftarrow \nabla f(x^0),\ B^0 \leftarrow I,\ k \leftarrow 0$

while $\|g^k\| > \delta$ do

$d \leftarrow -B^k g^k$

$t \leftarrow$ Lewis Overton line search

$x^{k+1} \leftarrow x^k + td$

$g^{k+1} \leftarrow \nabla f\left(x^{k+1}\right)$

$B^{k+1} \leftarrow$ Cautious-Limited-Memory-BFGS$\left(g^{k+1} - g^k, x^{k+1} - x^k\right)$

$k \leftarrow k + 1$

end while

return

Quasi-Newton Methods

Now we have detailed a practically powerful quasi-newton method:

BFGS Update

+

Cautious Update

+

Limited Memory Two-Loop Recursion

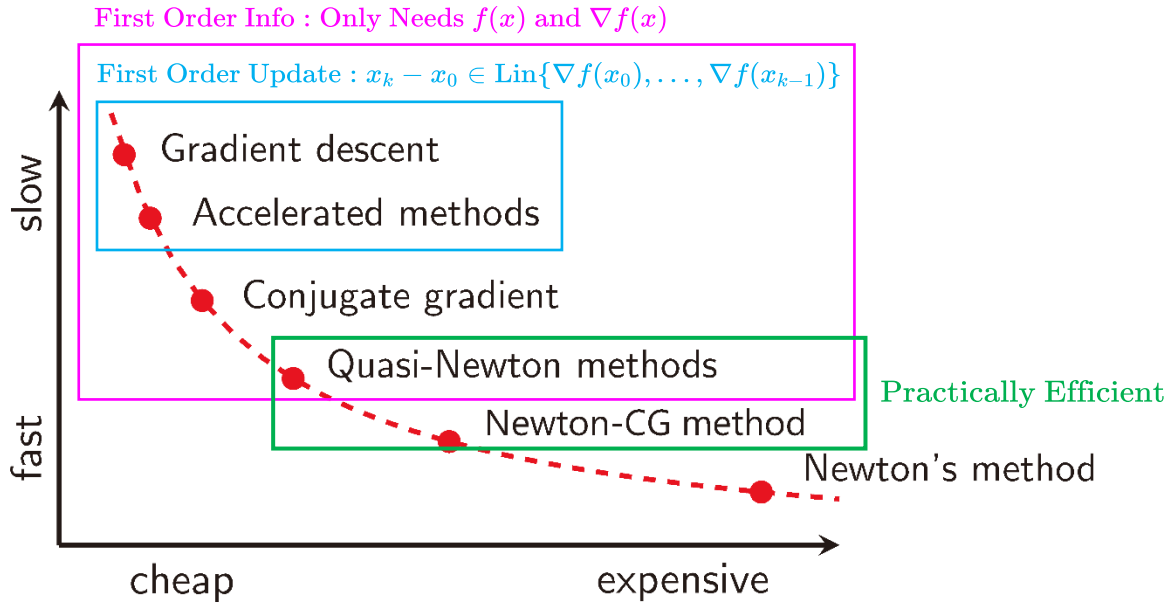
+

Lewis-Overton Line Search

Quasi-Newton Methods

Is quasi-Newton class the only way to avoid Hessian?

$$\underset{x}{\text{minimize}} \quad f(x)$$



Newton-CG Method

Newton-CG Method

Complexity of function/gradient/Hessian

$$f:\mathbb{R}^n \to \mathbb{R} \quad = \quad \nabla f = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{pmatrix} \quad < \quad \nabla^2 f = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{pmatrix}$$

Function Gradient Hessian

About computational complexity :

- Computing gradient is at least $O(n)$
- Computing gradient is as easy as computing the function
- Computing Hessian is at least $O(n^2)$ whose inversion is about $O(n^3)$

Newton-CG Method

Fortunately

Newton-CG Method

Complexity of function/gradient/Hessian-vec product

$$f : \mathbb{R}^n \to \mathbb{R} \quad = \quad \nabla f = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{pmatrix} \quad = \quad (\nabla^2 f)\xi = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{pmatrix} \begin{pmatrix} \xi_1 \\ \xi_2 \\ \vdots \\ \xi_n \end{pmatrix}$$

Function Gradient Hessian-vec product

Even if exact value is not available, just compute $(\nabla^2 f)\xi \approx \frac{\nabla f(x + \delta\xi) - \nabla f(x)}{\delta}$

However, we need inverse-Hessian-vec product.
What can be exploited via Hessian-vec product?

Newton-CG Method

The most prominent large-scale linear solver:
(Linear) Conjugate Gradient Method

$$Ax = b$$

$A \in \mathbb{R}^{n \times n}$:	a known PD mat
$x \in \mathbb{R}^n$:	an unknown vec
$b \in \mathbb{R}^n$:	a known vec

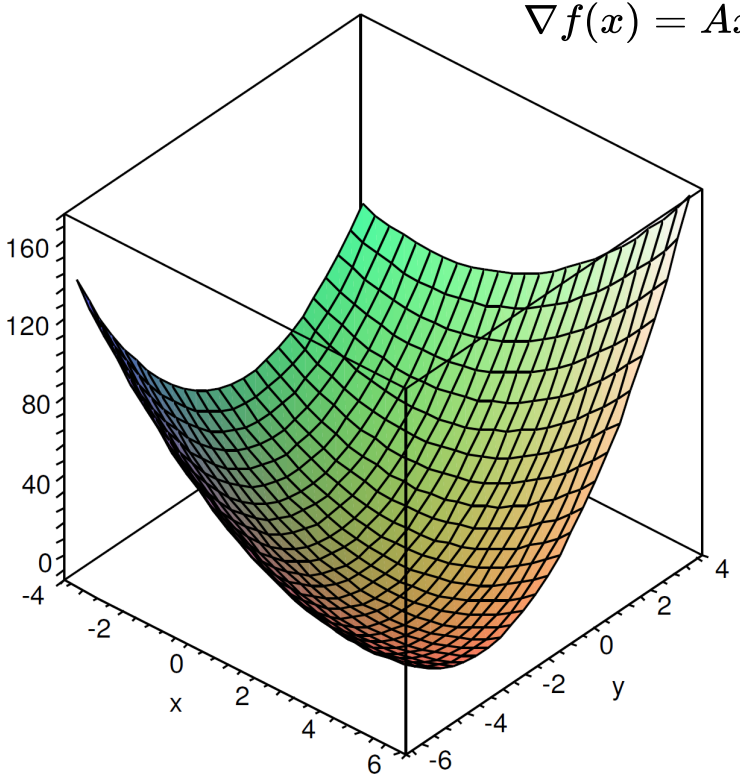
Solve $Ax = b$ via the interface $\gamma(x) := Ax$

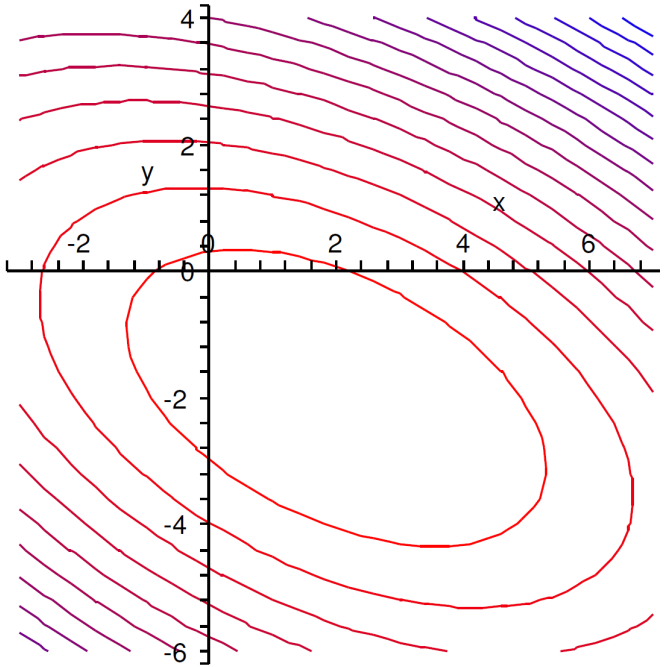
Newton-CG Method

(Linear) Conjugate Gradient Method

$$Ax = b \quad \Longleftrightarrow \quad \nabla f(x) = Ax - b = 0$$

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

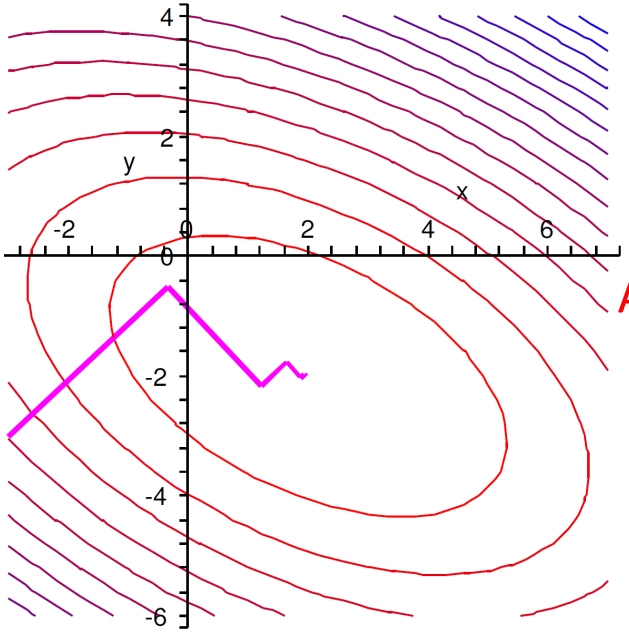




Newton-CG Method

(Linear) Conjugate Gradient Method

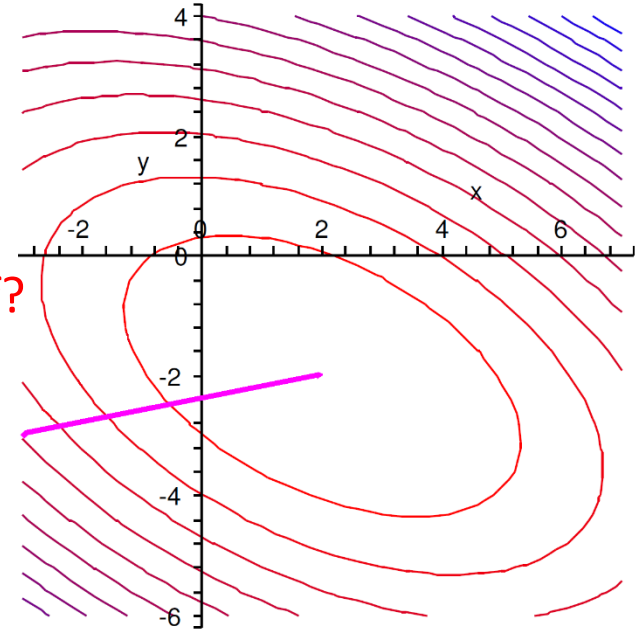
Recall steepest gradient descent



Infinite steps for an exact solution

Any trade-off?

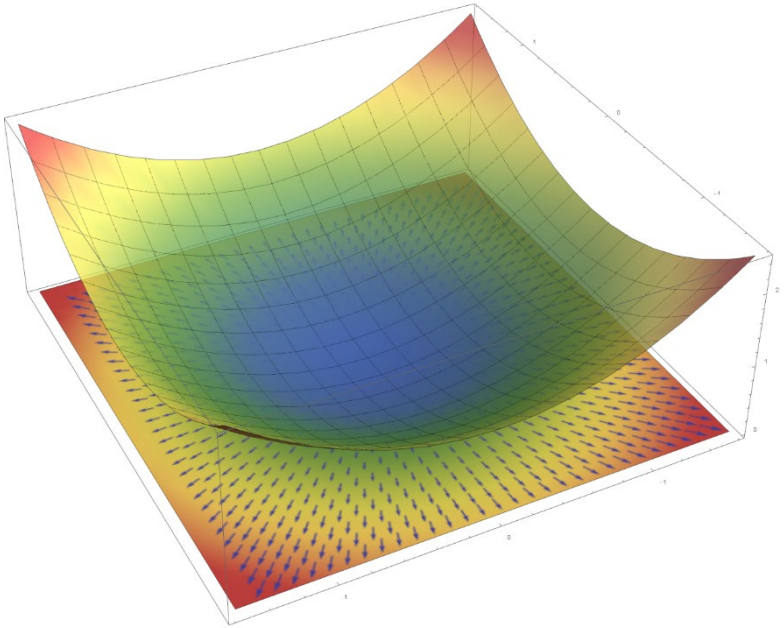
Recall Newton's method

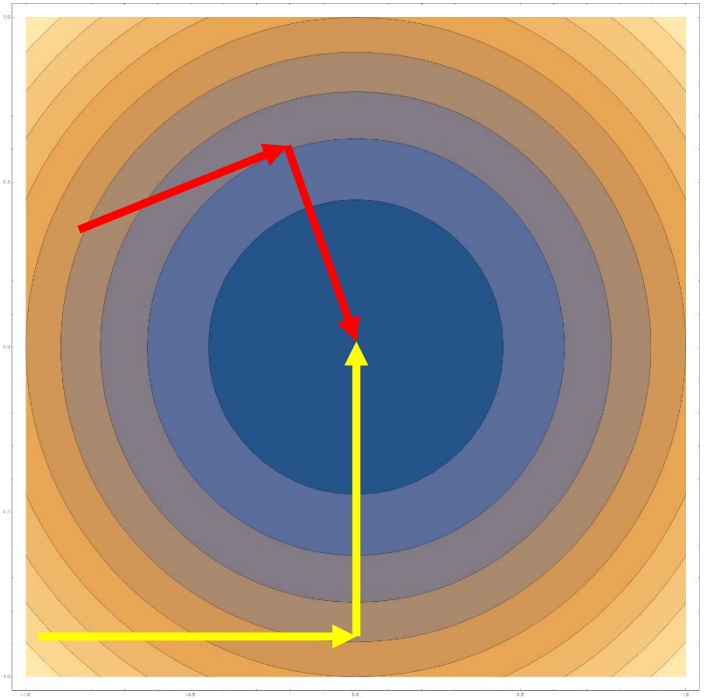


One step for an exact solution but with Hessian-inv

Newton-CG Method

(Linear) Conjugate Gradient Method



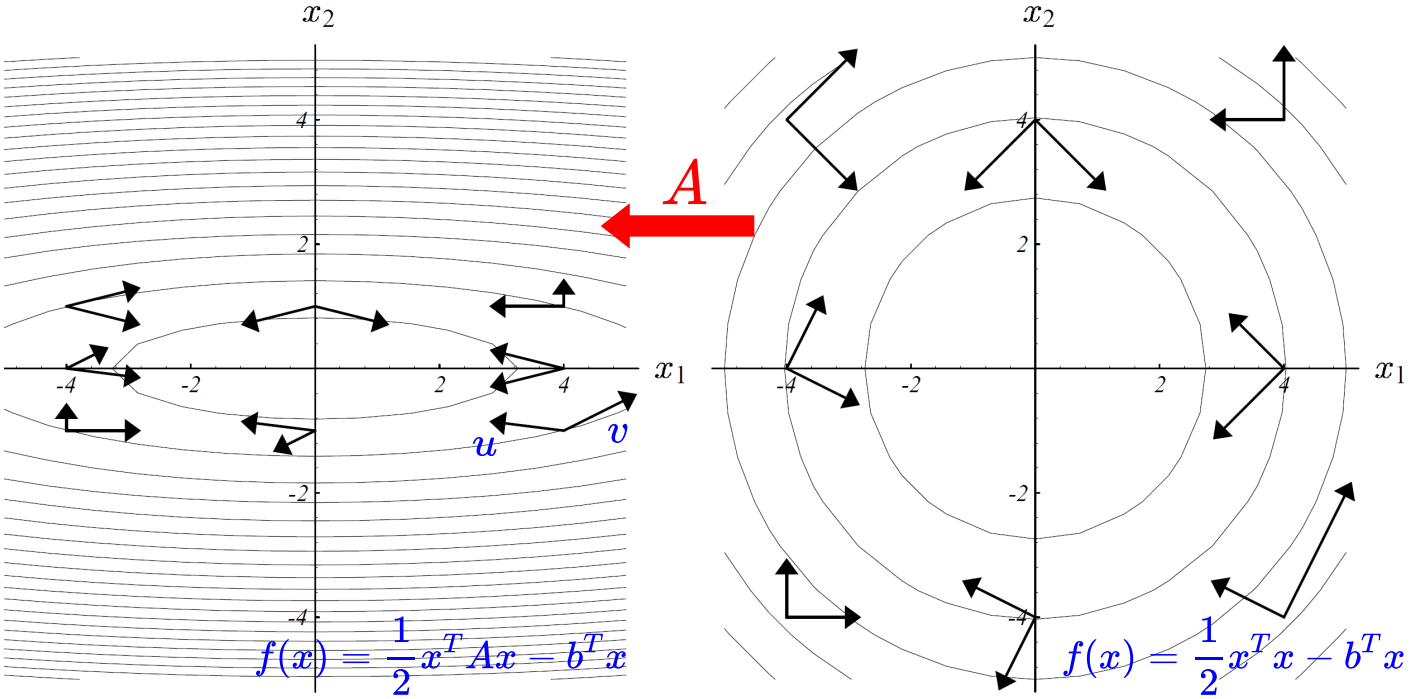


A simple case: $f(x) = \frac{1}{2}x^T x - b^T x$

At most n orthogonal steps with exact line search

Newton-CG Method

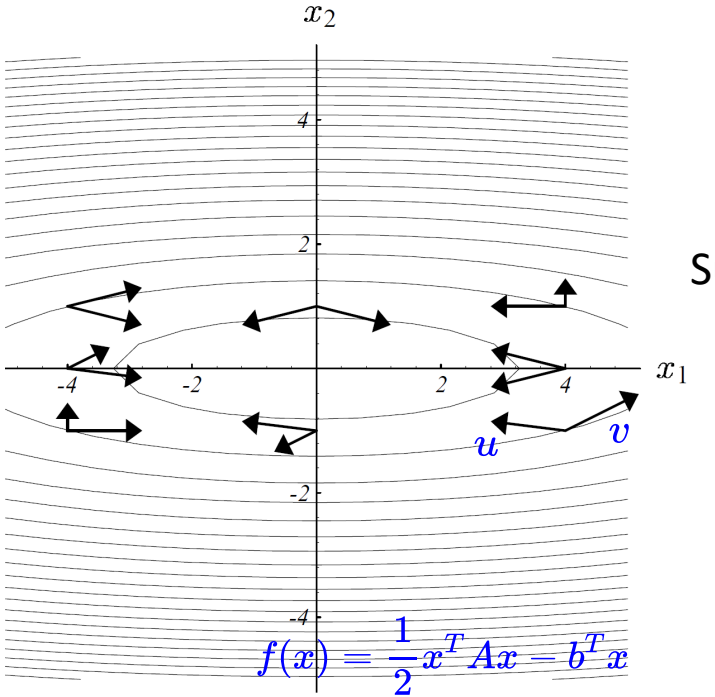
(Linear) Conjugate Gradient Method



u and v is conjugate $\Leftrightarrow$ $u^{\mathrm{T}} Av = 0$ $\Leftrightarrow$ u and v is A-orthogonal

Newton-CG Method

(Linear) Conjugate Gradient Method



Given n mutually conjugate directions

$$u^1, u^2, \ldots, u^n$$

Successively conduct closed-form exact line search

$$\alpha \leftarrow \frac{b^T u^k - (x^k)^T Au^k}{(u^k)^T Au^k}$$

interface $\gamma(u^k)$

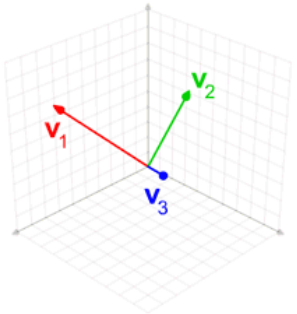
$$x^{k+1} \leftarrow x^k + \alpha u^k$$

Obtain an exact solution in n steps

How to find conjugate directions?

Newton-CG Method

(Linear) Conjugate Gradient Method



$$
\begin{aligned}
\mathbf{u}^1 &= \mathbf{v}^1, & \mathbf{e}^1 &= \frac{\mathbf{u}^1}{\left\|\mathbf{u}^1\right\|} \\
\mathbf{u}^2 &= \mathbf{v}^2 - \operatorname{proj}_{\mathbf{u}^1}\left(\mathbf{v}^2\right), & \mathbf{e}^2 &= \frac{\mathbf{u}^2}{\left\|\mathbf{u}^2\right\|} \\
\mathbf{u}^3 &= \mathbf{v}^3 - \operatorname{proj}_{\mathbf{u}^1}\left(\mathbf{v}^3\right) - \operatorname{proj}_{\mathbf{u}^2}\left(\mathbf{v}^3\right), & \mathbf{e}^3 &= \frac{\mathbf{u}^3}{\left\|\mathbf{u}^3\right\|} \\
\mathbf{u}^4 &= \mathbf{v}^4 - \operatorname{proj}_{\mathbf{u}^1}\left(\mathbf{v}^4\right) - \operatorname{proj}_{\mathbf{u}^2}\left(\mathbf{v}^4\right) - \operatorname{proj}_{\mathbf{u}^3}\left(\mathbf{v}^4\right), & \mathbf{e}^4 &= \frac{\mathbf{u}^4}{\left\|\mathbf{u}^4\right\|} \\
&\vdots & &\vdots \\
\mathbf{u}^k &= \mathbf{v}^k - \sum_{j=1}^{k-1} \operatorname{proj}_{\mathbf{u}^j}\left(\mathbf{v}^k\right), & \mathbf{e}^k &= \frac{\mathbf{u}^k}{\left\|\mathbf{u}^k\right\|}.
\end{aligned}
$$

conjugate Gram–Schmidt process

$$\operatorname{proj}_{\mathbf{u}}(\mathbf{v}) = \frac{\langle \mathbf{u}, \mathbf{v} \rangle_{\mathrm{A}}}{\langle \mathbf{u}, \mathbf{u} \rangle_{\mathrm{A}}} \mathbf{u}, \quad \langle \mathbf{u}, \mathbf{v} \rangle_{\mathrm{A}} = \mathbf{u}^{\mathrm{T}} \mathrm{A} \mathbf{v}$$

Gram–Schmidt process

$$\operatorname{proj}_{\mathbf{u}}(\mathbf{v}) = \frac{\langle \mathbf{u}, \mathbf{v} \rangle}{\langle \mathbf{u}, \mathbf{u} \rangle} \mathbf{u}$$

construct conjugate directions via (conjugate) Gram–Schmidt process

Newton-CG Method

(Linear) Conjugate Gradient Method

linearly independent $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k$

conjugate Gram–Schmidt process

mutually conjugate $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_k$

$$\mathbf{u}^k = \mathbf{v}^k - \sum_{j=1}^{k-1} \mathrm{proj}_{\mathbf{u}^j}(\mathbf{v}^k)$$

but the recurrence is too long!

Newton-CG Method

(Linear) Conjugate Gradient Method Chooses $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k$ Such That

- All linearly independent vectors are generated on the fly
- The conjugate Gram–Schmidt enjoys a short recurrence

Newton-CG Method

(Linear) Conjugate Gradient Method

If the linearly independent vector is chosen as

$$\mathrm{v}^k = b - Ax^k$$

The history-dependent coefficients in conjugate G-S process becomes

$$\mathrm{proj}_{\mathrm{u}^j}(\mathrm{v}^k) = 0,\ \forall j \leq k-2$$

Thus we get a short recurrence

$$\mathrm{u}^k = \mathrm{v}^k - \sum_{j=1}^{k-1} \mathrm{proj}_{\mathrm{u}^j}(\mathrm{v}^k) = \mathrm{v}^k + \frac{\|\mathrm{v}^k\|^2}{\|\mathrm{v}^{k-1}\|^2}\mathrm{u}^{k-1}$$

Newton-CG Method

(Linear) Conjugate Gradient Method

$x^1 \leftarrow x_{ini}$

$\mathrm{v}^1 \leftarrow b - Ax^1$

$\mathrm{u}^1 \leftarrow \mathrm{v}^1$

$k \leftarrow 1$

while $\mathrm{v}^k \neq 0$

$\alpha \leftarrow \|\mathrm{v}^k\|^2 / (\mathrm{u}^k)^{\mathrm{T}} A\mathrm{u}^k$

$x^{k+1} \leftarrow x^k + \alpha \mathrm{u}^k$

$\mathrm{v}^{k+1} \leftarrow \mathrm{v}^k - \alpha A\mathrm{u}^k$

$\beta \leftarrow \|\mathrm{v}^{k+1}\|^2 / \|\mathrm{v}^k\|^2$

$\mathrm{u}^{k+1} \leftarrow \mathrm{v}^{k+1} + \beta \mathrm{u}^k$

$k \leftarrow k + 1$

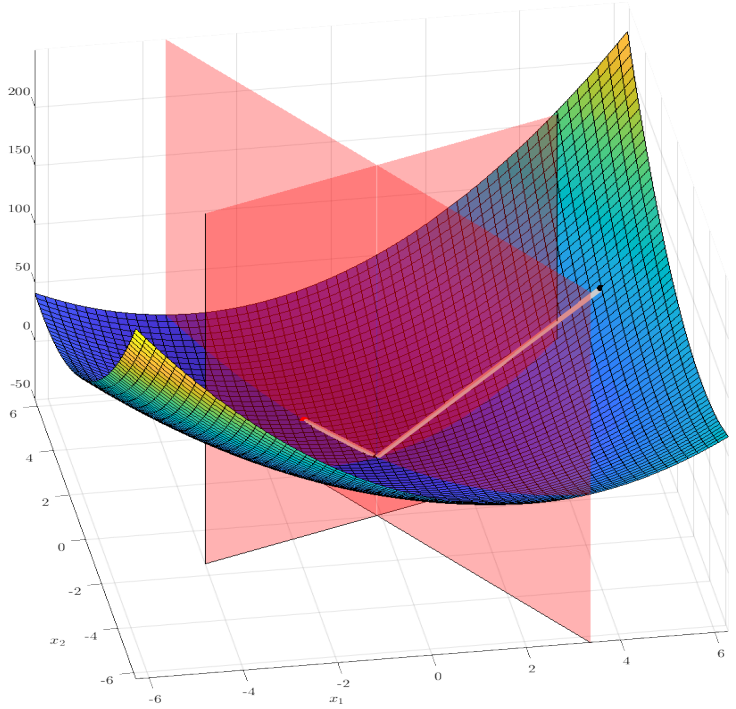
end (while)

return x^k

float-point precision:

$\|\mathrm{v}^k\| \geq \epsilon$

interface $\gamma(u^k)$



If exact arithmetic is used, CG always finds an exact solution in n steps

Newton-CG Method

(Linear) Conjugate Gradient Method

$x^1 \leftarrow x_{ini}$
$\mathrm{v}^1 \leftarrow b - Ax^1$
$\mathrm{u}^1 \leftarrow \mathrm{v}^1$
$k \leftarrow 1$
while $\mathrm{v}^k \neq 0$
  $\alpha \leftarrow \|\mathrm{v}^k\|^2 / (\mathrm{u}^k)^{\mathrm{T}} A\mathrm{u}^k$
  $x^{k+1} \leftarrow x^k + \alpha \mathrm{u}^k$
  $\mathrm{v}^{k+1} \leftarrow \mathrm{v}^k - \alpha A\mathrm{u}^k$
  $\beta \leftarrow \|\mathrm{v}^{k+1}\|^2 / \|\mathrm{v}^k\|^2$
  $\mathrm{u}^{k+1} \leftarrow \mathrm{v}^{k+1} + \beta \mathrm{u}^k$
  $k \leftarrow k + 1$
end (while)
return x^k

float-point precision:
$\|\mathrm{v}^k\| \geq \epsilon$

Features:

- CG can either be interpreted as a direct or iterative solver
- CG is often used with preconditioner
- CG only requires mat-vec product
- CG directly exploits mat sparsity without producing fill-in
- CG normally converges fast

$A \in \mathbb{R}^{555\times555}$, $\kappa(A) > 10^{10}$ **Method**	**Number of Iterations**
Gauss Seidel	208,000
Block SOR methods	765
Preconditioned **conjugate gradient**	25

Newton-CG Method

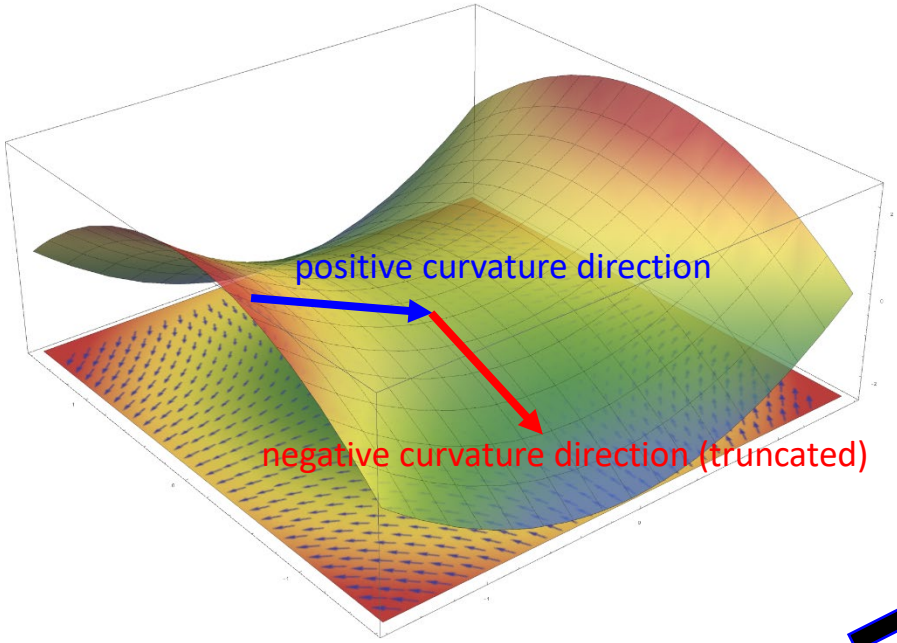
Newton-CG Method

$$\text{Apply CG to solve } (\nabla^2 f)d = -\nabla f \text{ via the interface } \gamma(\xi) := (\nabla^2 f)\xi$$

- How to handle indefinite Hessian?
- Should the linear system be solved exactly?

Newton-CG Method

Newton-CG Method



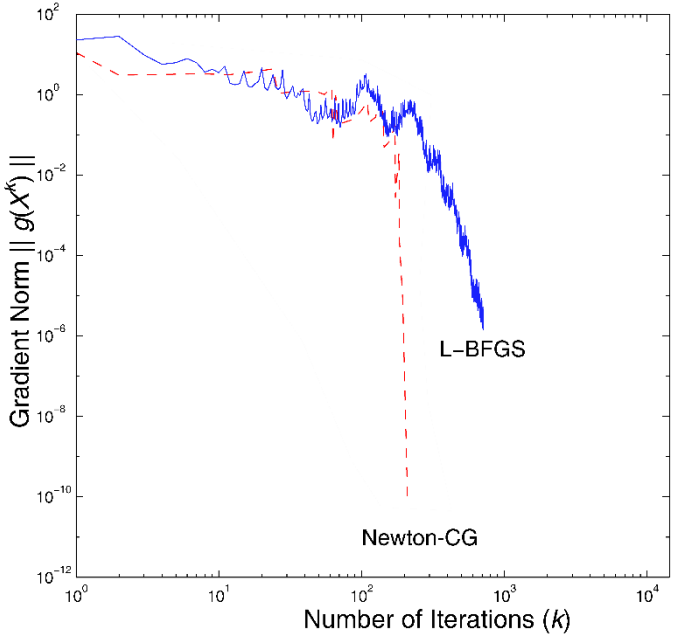
$x^1 \leftarrow x_{ini}, \quad k \leftarrow 1$
while $\|\nabla f(x^k)\| > \delta$
  $\epsilon^k \leftarrow \min(1, \|\nabla f(x^k)\|)/10$
  $d^1 \leftarrow 0, \quad \mathrm{v}^1 \leftarrow -\nabla f(x^k)$
  $\mathrm{u}^1 \leftarrow \mathrm{v}^1, \quad j \leftarrow 1$
  while $\|\mathrm{v}^j\| > \epsilon^k \|\nabla f(x^k)\|$
    if $(\mathrm{u}^j)^{\mathrm{T}} \nabla^2 f(x^k) \mathrm{u}^j \leq 0$
      if $j = 1$
        $d^j \leftarrow -\nabla f(x^k)$
      end (if)
    break
    end (if)
    $\alpha \leftarrow \|\mathrm{v}^j\|^2 / (\mathrm{u}^j)^{\mathrm{T}} \nabla^2 f(x^k) \mathrm{u}^j$
    $d^{j+1} \leftarrow d^j + \alpha \mathrm{u}^j$
    $\mathrm{v}^{j+1} \leftarrow \mathrm{v}^j - \alpha A \mathrm{u}^j$
    $\beta \leftarrow \|\mathrm{v}^{j+1}\|^2 / \|\mathrm{v}^j\|^2$
    $\mathrm{u}^{j+1} \leftarrow \mathrm{v}^{j+1} + \beta \mathrm{u}^j$
    $j \leftarrow j + 1$
  end (while)
  $\alpha \leftarrow$ backtracking line search (Armijo)
  $x^{k+1} \leftarrow x^k + \alpha d^j$
end (while)

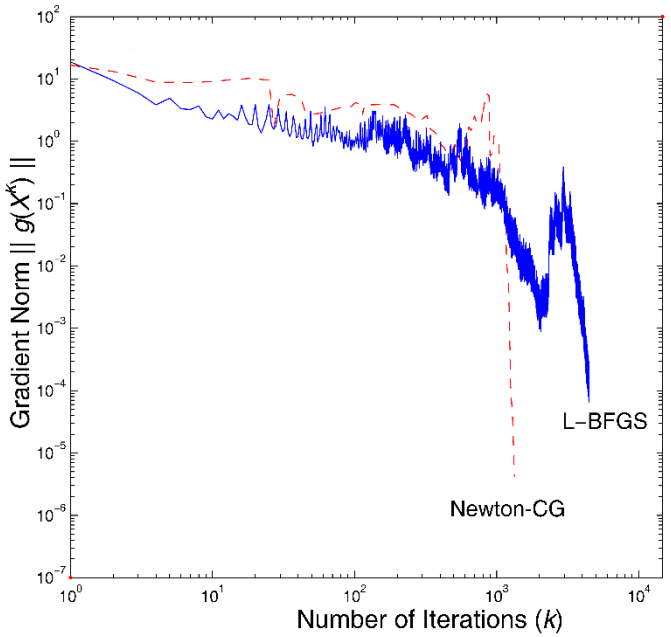
Truncated CG:

- Start with zero as the initial direction
- Truncating negative curvature direction

Newton-CG Method

Newton-CG Method





Molecular Dynamics Simulation (66 variables)					
Minimizer	Iters.(Hess.-vec calls.)	f	$\|\nabla f\|$	f&∇ evals.	CPU time
Newton-CG	29(210)	-15.25	7.67×10^{-11}	44	1.12 sec.
L-BFGS	711	-15.25	1.4×10^{-6}	740	1.39

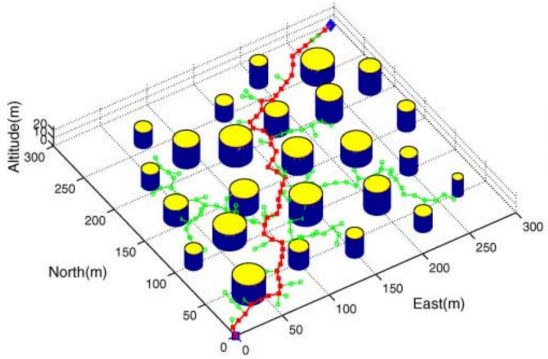
Molecular Dynamics Simulation (1704 variables)					
Minimizer	Iters.(Hess.-vec calls.)	f	$\|\nabla f\|$	f&∇ evals.	CPU time
Newton-CG	65(1335)	-2773.70	4.2×10^{-6}	240	5.21 min.
L-BFGS	4486	-2792.96	6.3×10^{-5}	4622	12.61

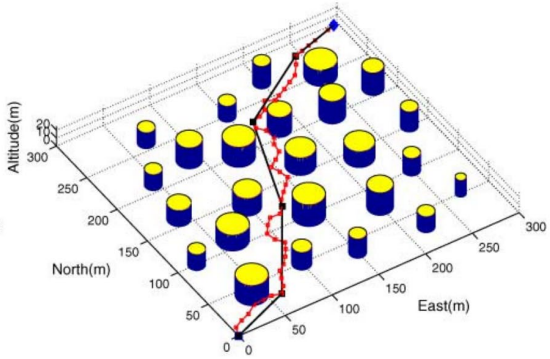
Newton-CG normally achieves lower gradient norm.

LBFS的应用

Application:
Smooth Navigation Path Generation

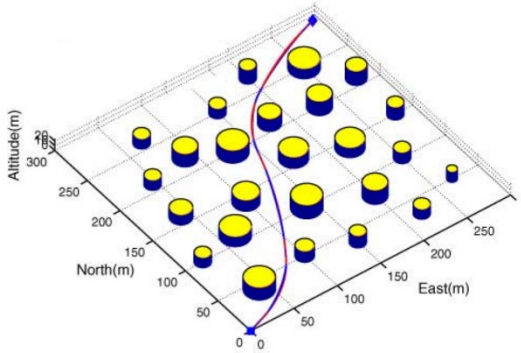
Smooth Navigation Path Generation





- RRT*-like planners generate low-dim paths efficiently.
- Nonsmooth paths are unsatisfactory for mobile robots.
- Unconstrained opt can efficiently smooth these paths.

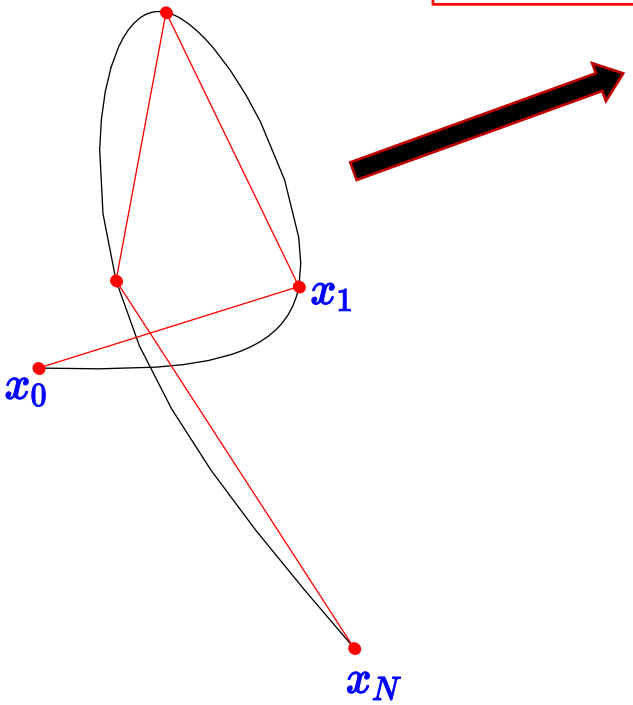
使用无约束优化进行处理



Smooth Navigation Path Generation

Cubic Spline (Minimum Stretch Energy) Path Parameterization

光滑路径参数化



The *i*-th segment $p_i(s) = a_i + b_i s + c_i s^2 + d_i s^3, s \in [0,1]$

Curve and its 1st/2nd order derivative are continuous.

多项式的零阶导数，一阶导数，二阶导数连续

$$C^2: \ p_{i-1}(1) = p_i(0), \ p_{i-1}^{(1)}(1) = p_i^{(1)}(0), \ p_{i-1}^{(2)}(1) = p_i^{(2)}(0),$$

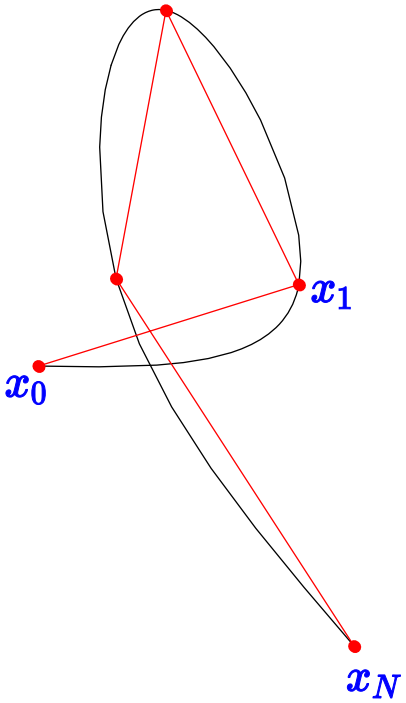
Assume stationary boundary conditions.

假设开始与结束的速度为0

$$p_1^{(1)}(0) = 0, \ p_N^{(1)}(1) = 0$$

Smooth Navigation Path Generation

Cubic Spline (Minimum Stretch Energy) Path Parameterization



A cubic curve sequentially crossing $x_0, x_1, \ldots, x_N$ is given by

$$
\begin{aligned}
a_i &= x_i \\
b_i &= D_i \\
c_i &= 3(x_{i+1} - x_i) - 2D_i - D_{i+1} \\
d_i &= 2(x_i - x_{i+1}) + D_i + D_{i+1}
\end{aligned}
$$

where

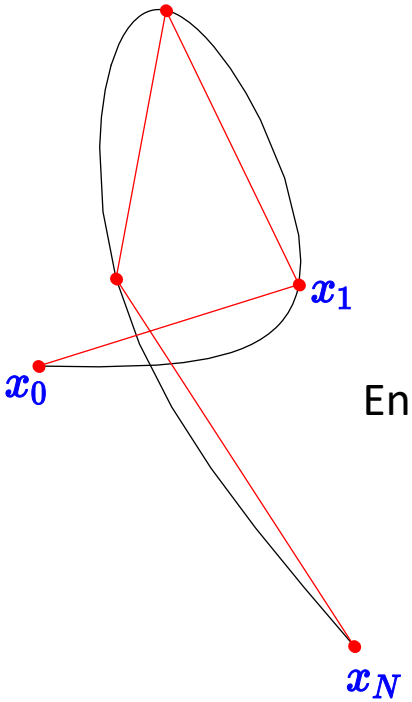
带状矩阵求解线性方程，Di表示的是速度

$$
\begin{bmatrix} D_1 \\ D_2 \\ D_3 \\ D_4 \\ \vdots \\ D_{n-2} \\ D_{n-1} \end{bmatrix} = \begin{bmatrix} 4 & 1 & & & & & \\ 1 & 4 & 1 & & & & \\ & 1 & 4 & 1 & & & \\ & & 1 & 4 & 1 & & \\ & & & \ddots & \ddots & \ddots & \\ & & & & 1 & 4 & 1 \\ & & & & & 1 & 4 \end{bmatrix}^{-1} \begin{bmatrix} 3(x_2 - x_0) \\ 3(x_3 - x_1) \\ 3(x_4 - x_2) \\ 3(x_5 - x_3) \\ \vdots \\ 3(x_{n-1} - x_{n-3}) \\ 3(x_n - x_{n-2}) \end{bmatrix}, \text{ and } D_0 = D_N = 0
$$

Thus the entire trajectory is determined by $x_0, x_1, \ldots, x_N$

Smooth Navigation Path Generation

Cubic Spline (Minimum Stretch Energy) Path Generation



Minimize the stretch energy to make it as smooth as possible

多项式系数，有解析解

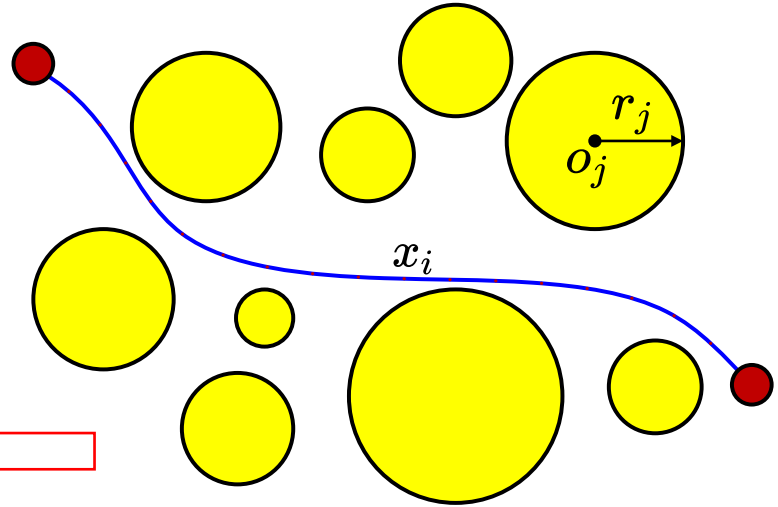
$$\text{Energy}(x_1, x_2, \ldots, x_{N-1}) = \sum_{i=0}^{N} \int_0^1 \left\| p_i^{(2)}(s) \right\|^2 \mathrm{d}s$$

Energy is a function of points since all coefficients are determined by points.

Smooth Navigation Path Generation

Cubic Spline (Minimum Stretch Energy) Path Generation

使用圆形障碍物，来进行碰撞



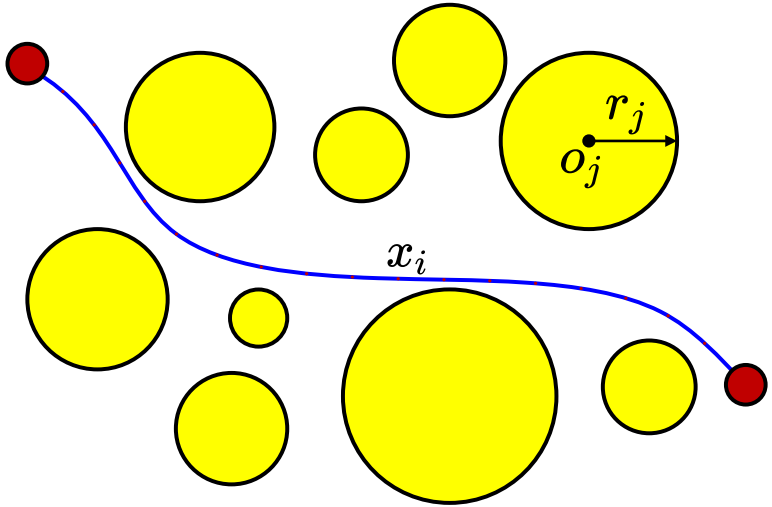
势场优化

$$\text{Potential}(x_1, x_2 \ldots, x_{N-1}) = 1000 \sum_{i=1}^{N-1} \sum_{j=1}^{M} \max(r_j - \|x_i - o_j\|,\ 0)$$

Ensure the curve is collision-free via potential functions!

Smooth Navigation Path Generation

Cubic Spline (Minimum Stretch Energy) Path Generation



Complete the smooth path generation by unconstrained minimization:

potential 非连续

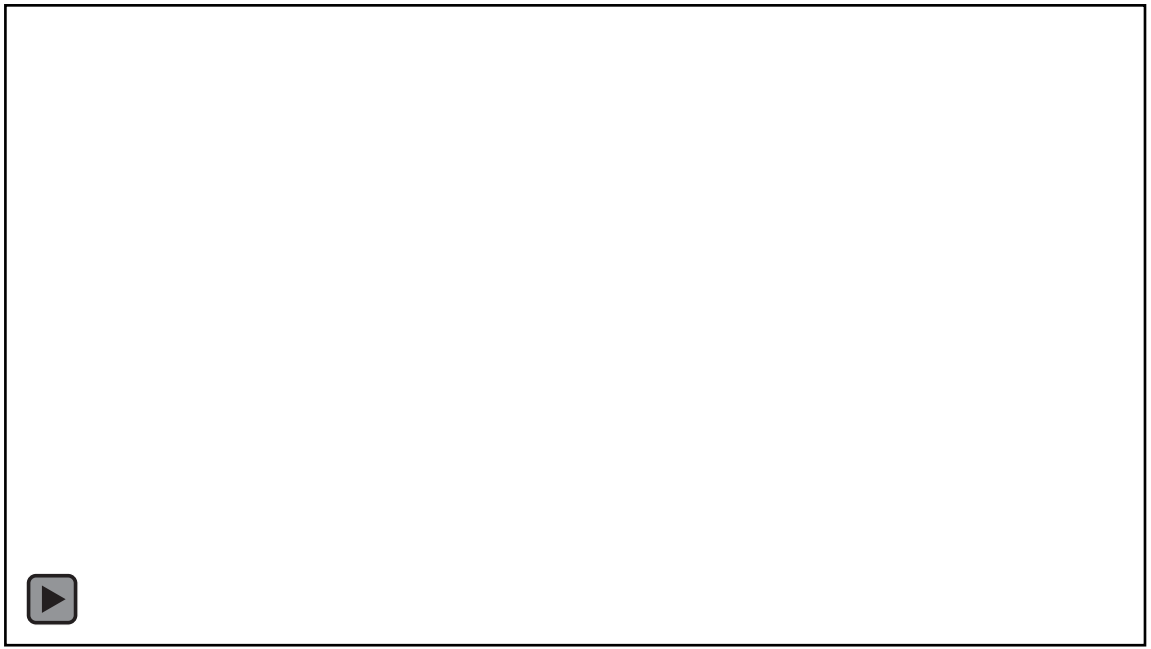
$$\min_{x_1, x_2 \ldots, x_{N-1}} \text{Energy}(x_1, x_2 \ldots, x_{N-1}) + \text{Potential}(x_1, x_2 \ldots, x_{N-1})$$

The non-smooth path can provide an initial guess for decision variables.

Smooth Navigation Path Generation

Cubic Spline (Minimum Stretch Energy) Path Generation

$$\min_{x_1, x_2 \ldots, x_{N-1}} \text{Energy}(x_1, x_2 \ldots, x_{N-1}) + \text{Potential}(x_1, x_2 \ldots, x_{N-1})$$



Thanks for Listening!

Ack.: The slide is prepared by Zhepei Wang and Fei Gao, improved by Lingfeng Wang, Song Zhao, and Shuo Yang.