AARHUS UNIVERSITY

Data Analytics and Machine Learning

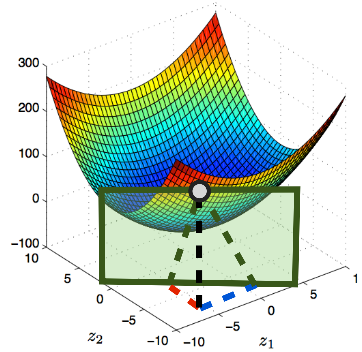
Conjugate Gradient Methods

Henrik Karstoft
Carl Schultz
Alexandros Iosifidis

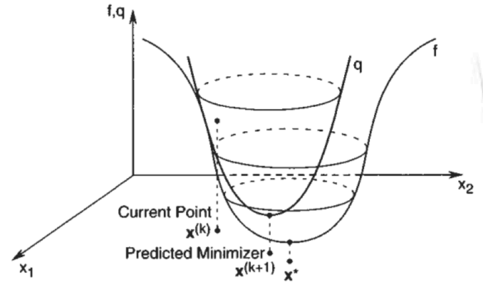
TODAY'S OUTLINE

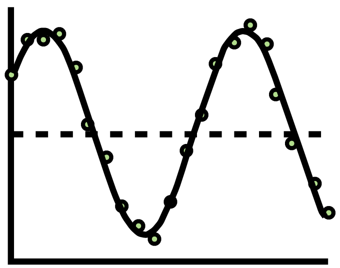
- quick recap
- Part 1. Quasi-Newton
- Part 2. Conjugate methods
- Part 3. Steepest Descent
- Part 4. some concepts
- Part 5. Conjugate Gradient

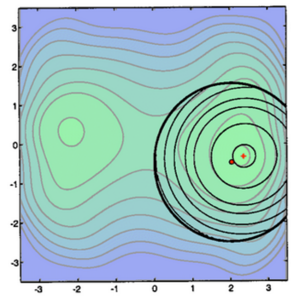
quick recap



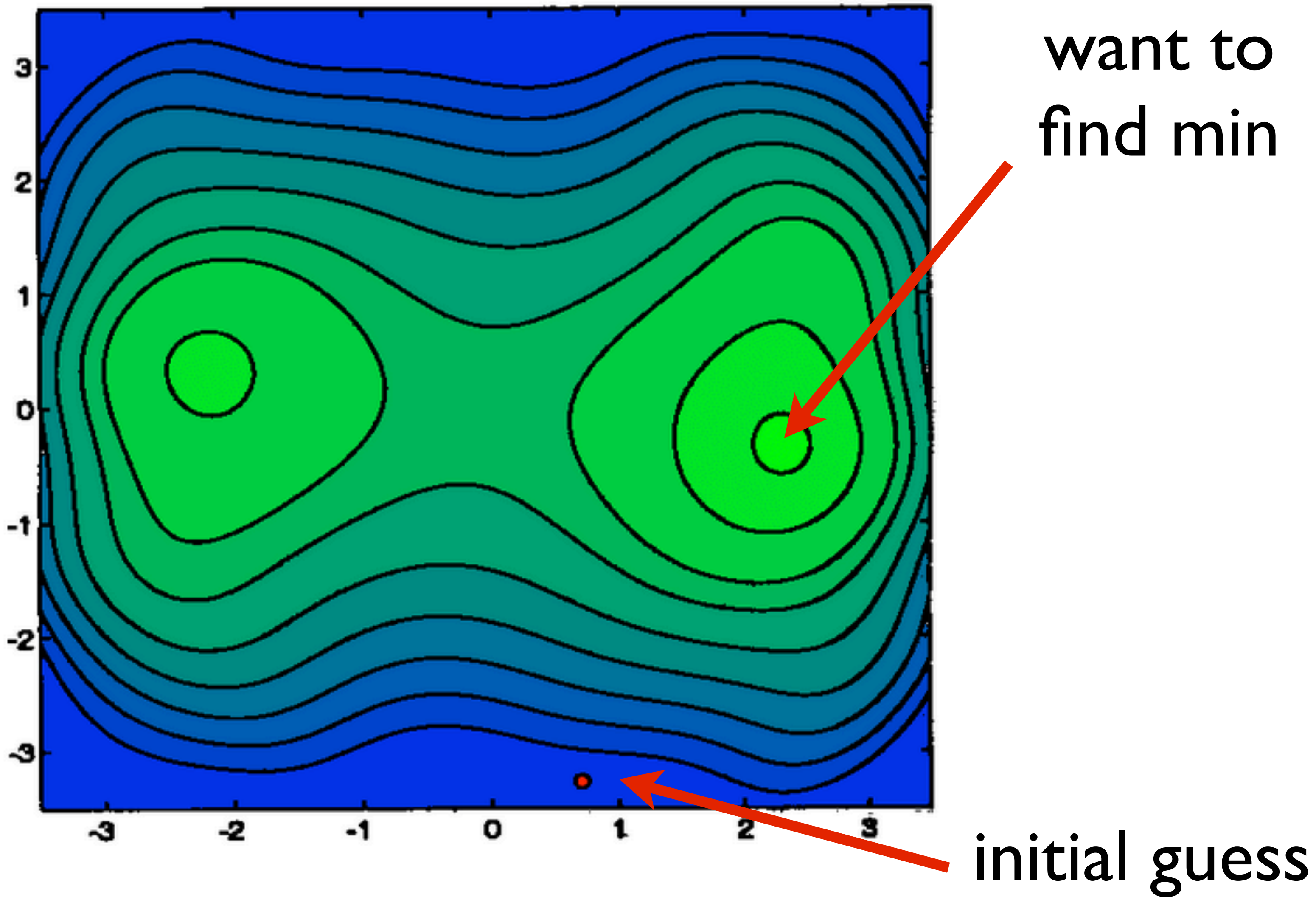
- preliminaries
- Newton method nD
- Newton's method for optimisation
- curve fitting







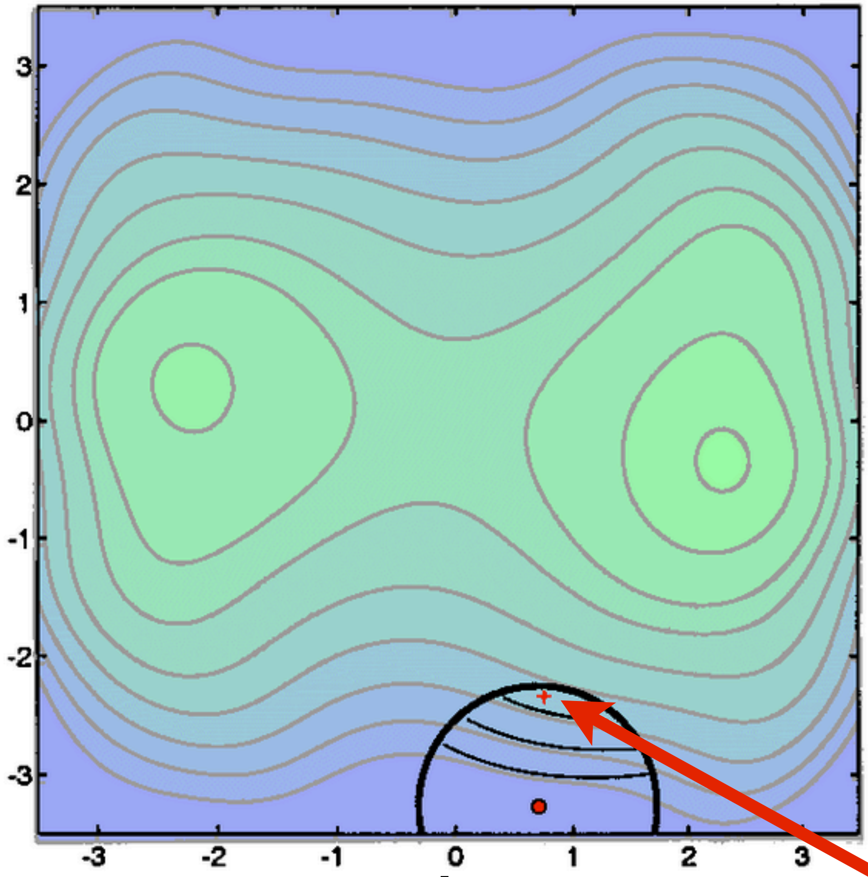
trust regions



http://www.applied-mathematics.net/optimization/optimizationIntro.html

trust regions

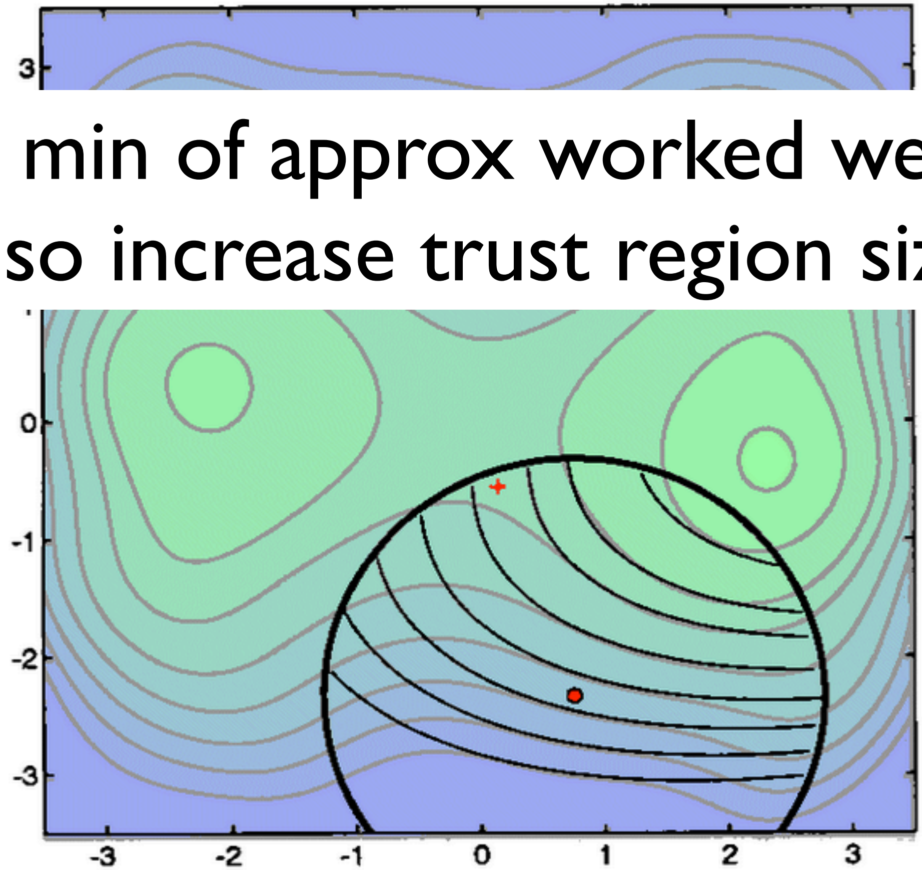
iteration 1



quad approx

min of approximation

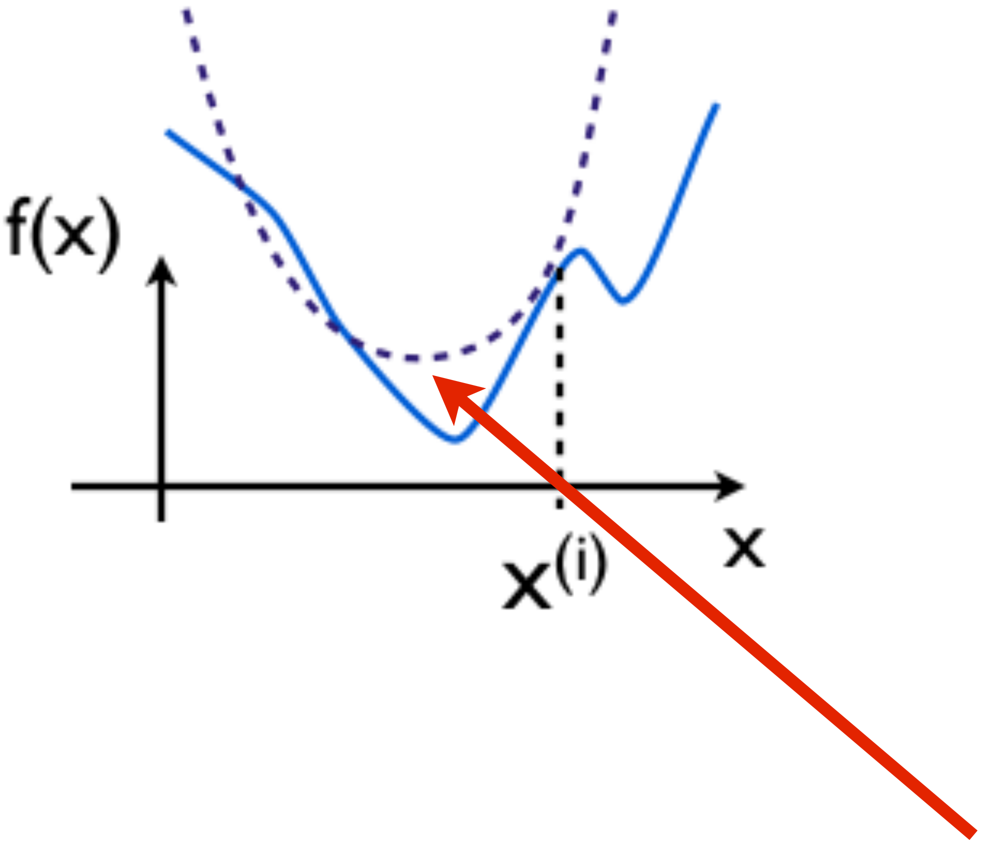
iteration 2



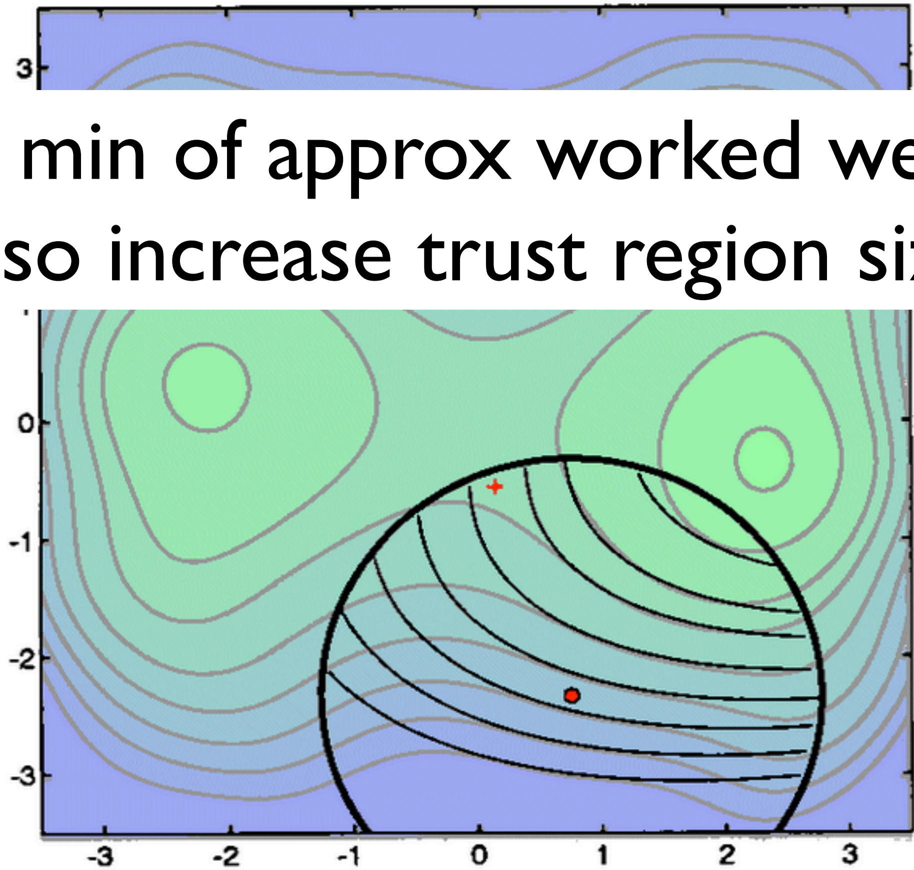
min of approx worked well
so increase trust region size

http://www.applied-mathematics.net/optimization/optimizationIntro.html

trust regions



iteration 2

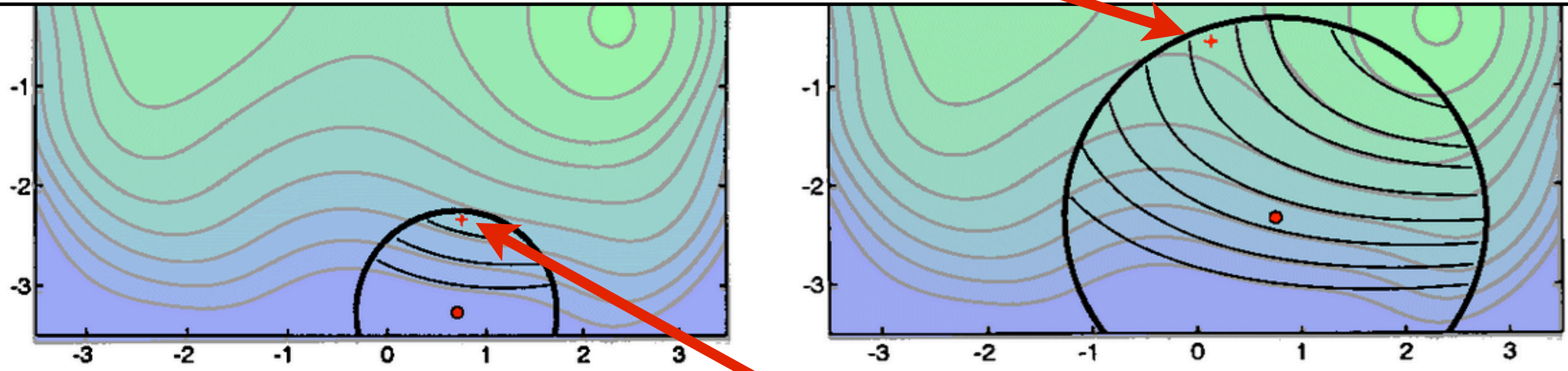


min of approx worked well
so increase trust region size

min of approximation

http://www.applied-mathematics.net/optimization/optimizationIntro.html

note. at each step we are not finding the *actual* min of the underlying function inside the circle (quadratic) in the images below - we are (1) **approximating** the underlying function with a quadratic, (2) finding the one single point that is the min of our approximation $x^{(i+1)}$, and only then (3) seeing what the actual underlying function value is at that point $x^{(i+1)}$. Think about the 1D case, at each step we are only looking at the min of our quadratic approximation, same idea in this example.



quad approx

min of approximation

http://www.applied-mathematics.net/optimization/optimizationIntro.html

Part 1

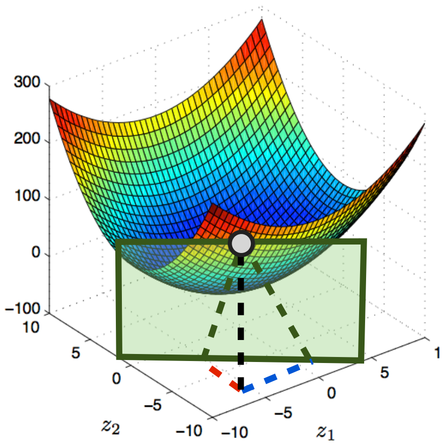
Quasi-Newton

one issue with Newton methods ...

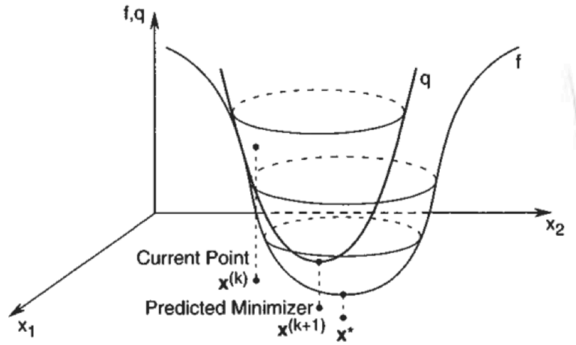
J, H might not be available, or too expensive...

$$X^{(i+1)} = X^{(i)} - J^{(i)-1} f(X^{(i)})$$

$$X^{(i+1)} = X^{(i)} - H^{(i)-1} \nabla f(X^{(i)})$$



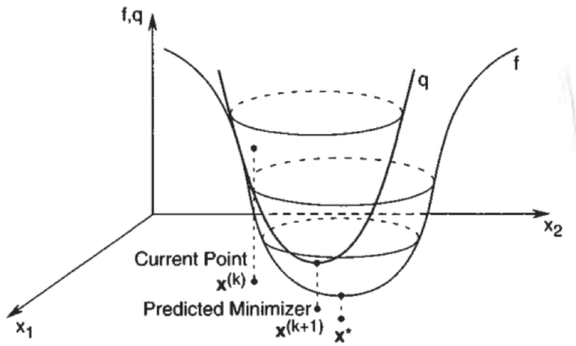
nD root finding



nD optimisation

replace Hessian with approximation B

$$X^{(i+1)} = X^{(i)} - B^{(i)-1} \nabla f(X^{(i)})$$



nD optimisation

replace Hessian with approximation B

$$X^{(i+1)}=X^{(i)} - B^{(i)-1} \nabla f(X^{(i)})$$

solve linearised system with approximate Hessian, B:

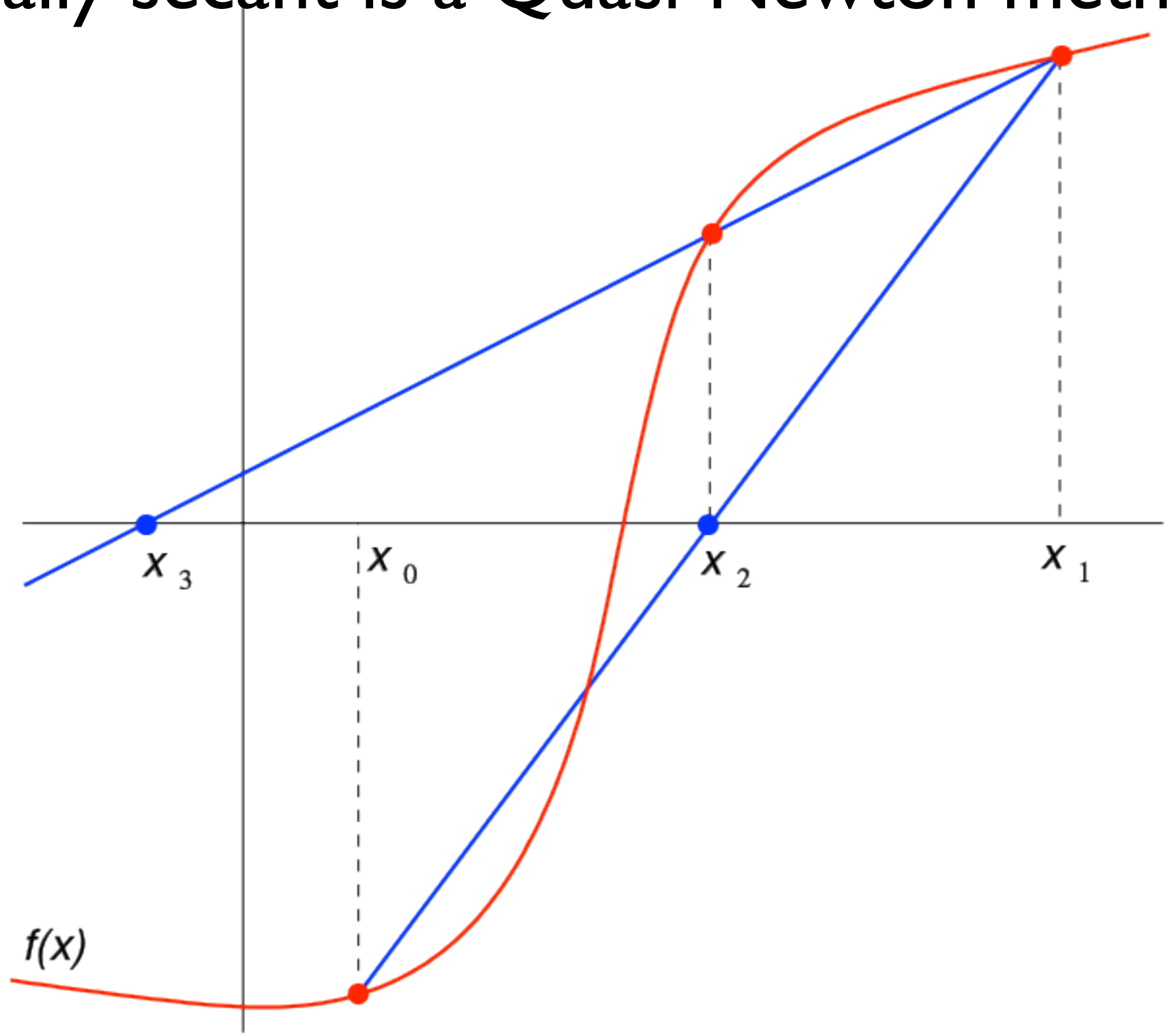
$$B^{(i)} \Delta X^{(i)} = \nabla f(X^{(i)})$$

make sure B has key properties (symmetry, positive-definiteness, etc.) to make solving fast, and approximations good

update B at each iteration

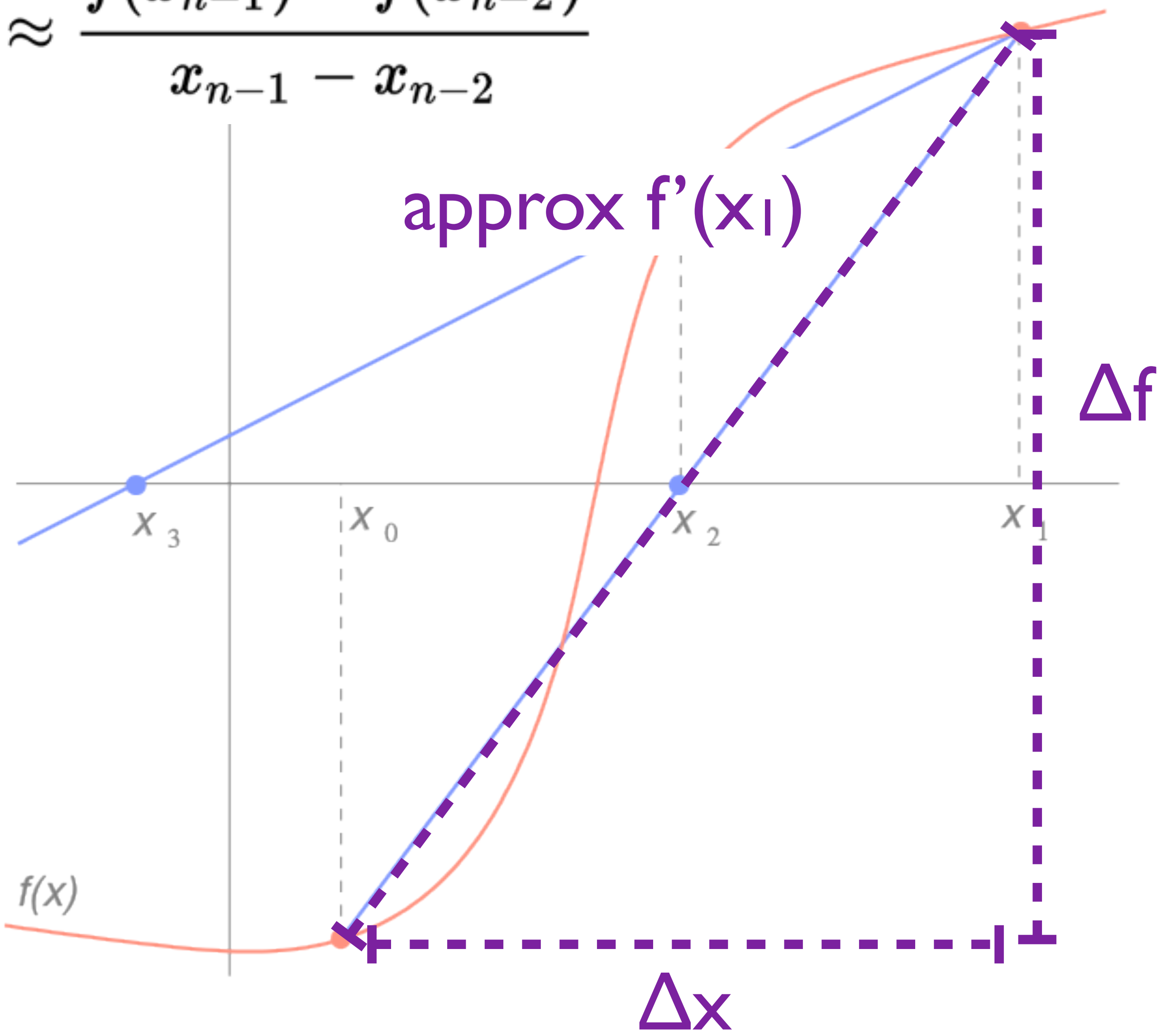
related to secant method

(actually secant is a Quasi-Newton method)



https://en.wikipedia.org/wiki/Secant_method secant method approximates derivative:

$$f'(x_{n-1}) \approx \frac{f(x_{n-1}) - f(x_{n-2})}{x_{n-1} - x_{n-2}}$$



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

Second-order Taylor expansion around iterate x_k:

$$f(x_k + \Delta x) \approx \underline{f(x_k)} + \underline{\nabla f(x_k)^T \Delta x} + \frac{1}{2}\underline{\Delta x^T B \Delta x}$$

gradient of this approximation:

$$\nabla f(x_k + \Delta x) \approx \nabla f(x_k) + B \Delta x$$

...setting gradient to zero gives us familiar Newton step

Quasi-Newton condition (secant equation), B must satisfy:

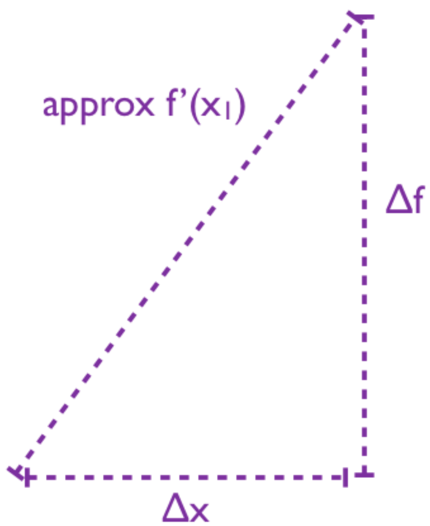
$$\nabla f(x_k + \Delta x) = \nabla f(x_k) + B \Delta x$$

secant method

$$f'(x_{n-1}) \approx \frac{f(x_{n-1}) - f(x_{n-2})}{x_{n-1} - x_{n-2}}$$

Quasi-Newton condition

$$B = \frac{\nabla f(x_k + \Delta x) - \nabla f(x_k)}{\Delta x}$$

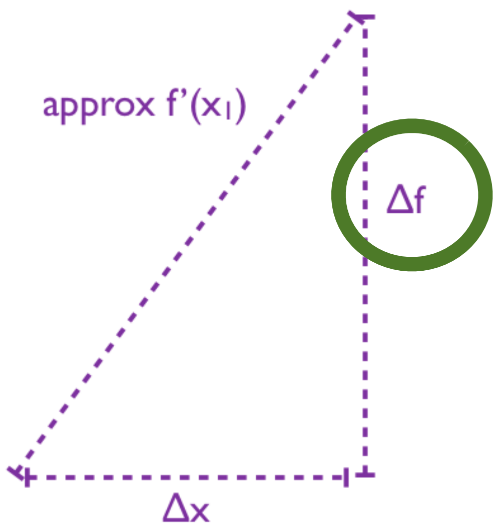


secant method

$$f'(x_{n-1}) \approx \frac{f(x_{n-1}) - f(x_{n-2})}{x_{n-1} - x_{n-2}}$$

Quasi-Newton condition

$$B = \frac{\nabla f(x_k + \Delta x) - \nabla f(x_k)}{\Delta x}$$

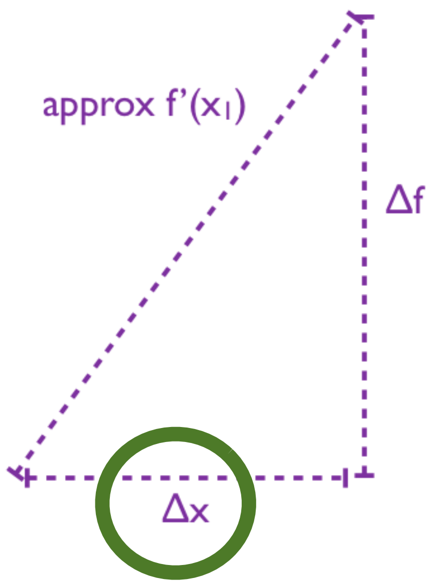


secant method

$$f'(x_{n-1}) \approx \frac{f(x_{n-1}) - f(x_{n-2})}{x_{n-1} - x_{n-2}}$$

Quasi-Newton condition

$$B = \frac{\nabla f(x_k + \Delta x) - \nabla f(x_k)}{\Delta x}$$

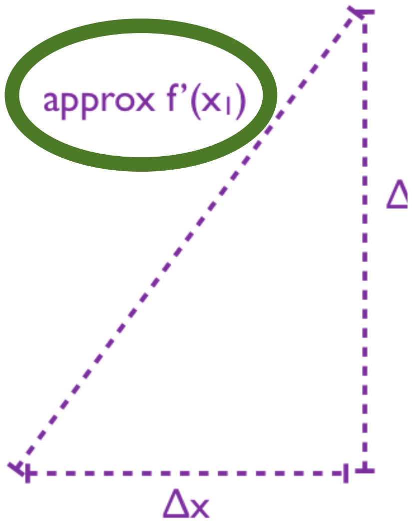


secant method

$$f'(x_{n-1}) \approx \frac{f(x_{n-1}) - f(x_{n-2})}{x_{n-1} - x_{n-2}}$$

Quasi-Newton condition

$$B = \frac{\nabla f(x_k + \Delta x) - \nabla f(x_k)}{\Delta x}$$



...as iterations increase, B converges to true Hessian

just a few examples

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

	B update
BFGS	$B_k + \frac{y_k y_k^T}{y_k^T \Delta x_k} - \frac{B_k \Delta x_k (B_k \Delta x_k)^T}{\Delta x_k^T B_k \, \Delta x_k}$
SR1	$B_k + \frac{(y_k - B_k \, \Delta x_k)(y_k - B_k \, \Delta x_k)^T}{(y_k - B_k \, \Delta x_k)^T \, \Delta x_k}$
Broyden	$B_k + \frac{y_k - B_k \Delta x_k}{\Delta x_k^T \, \Delta x_k} \, \Delta x_k^T$

SUMMARY Part 1. Quasi-Newton

- replace Jakobian/Hessian with approximation
- they generalise secant method
- fast and more robust than Newton

Part 2

Conjugate Methods

slides based on

An Introduction to
the Conjugate Gradient Method
Without the Agonizing Pain
Edition $1\frac{1}{4}$

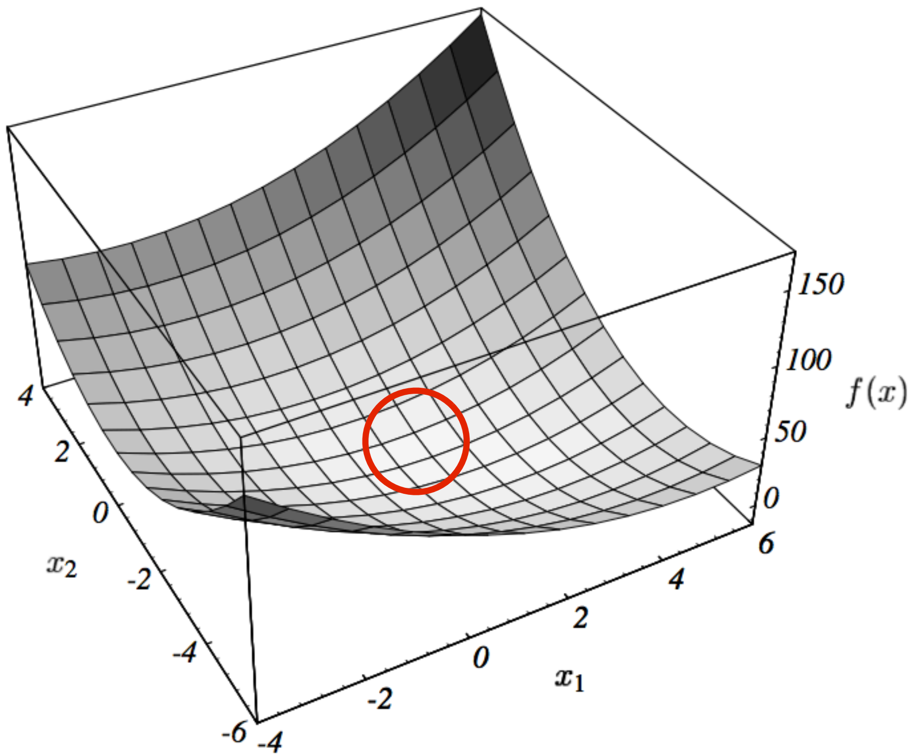
Jonathan Richard Shewchuk
August 4, 1994

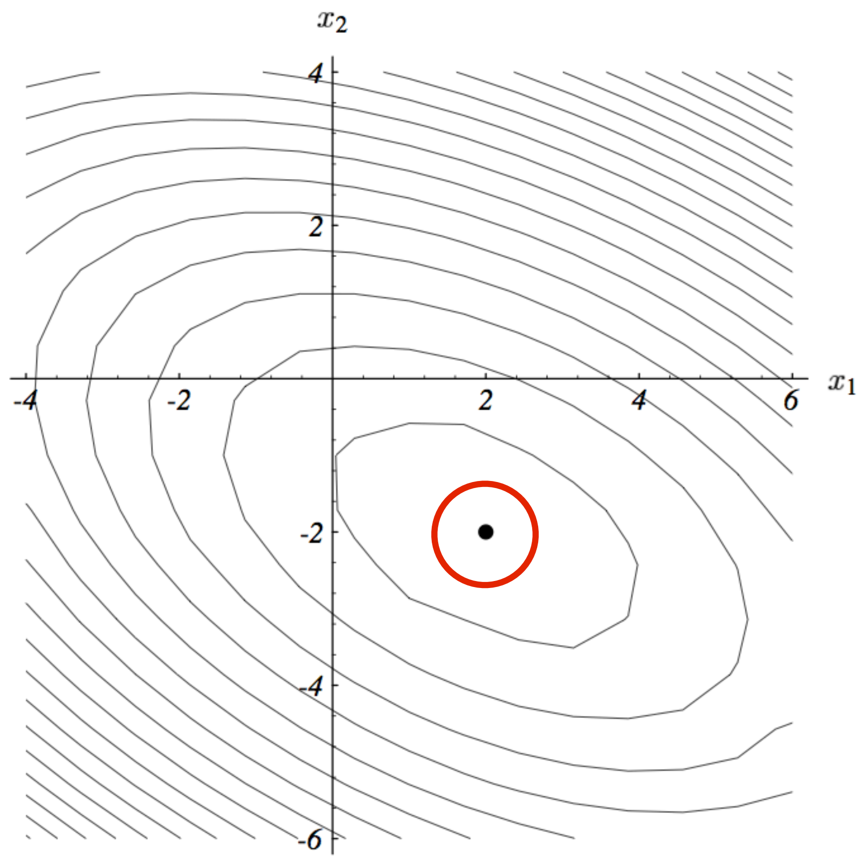
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

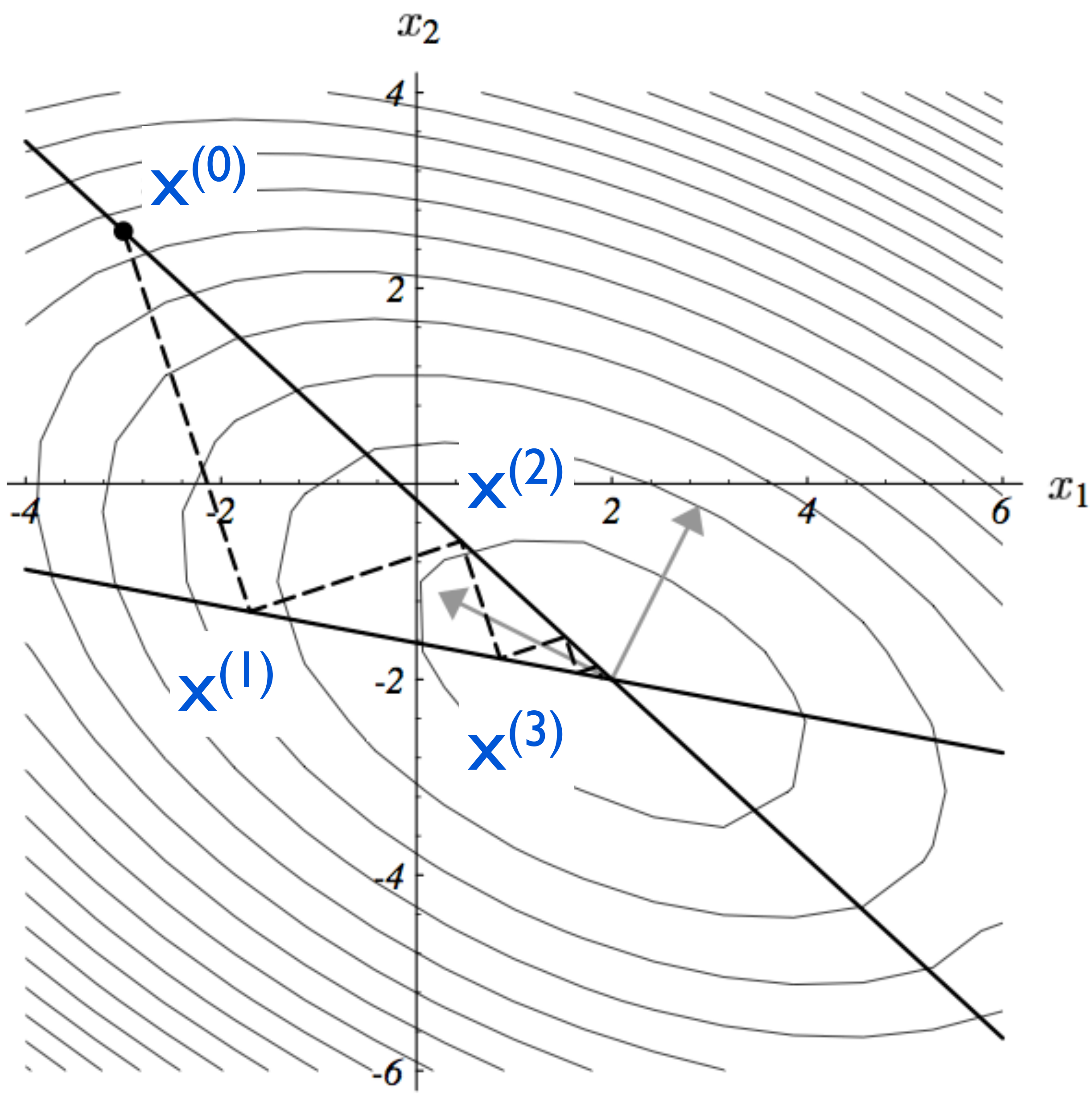
The Conjugate Gradient Method is the most prominent iterative method for solving sparse systems of linear equations. Unfortunately, many textbook treatments of the topic are written with neither illustrations nor intuition, and their victims can be found to this day babbling senselessly in the corners of dusty libraries. For this reason, a deep, geometric understanding of the method has been reserved for the elite brilliant few who have painstakingly decoded

we want to find the min



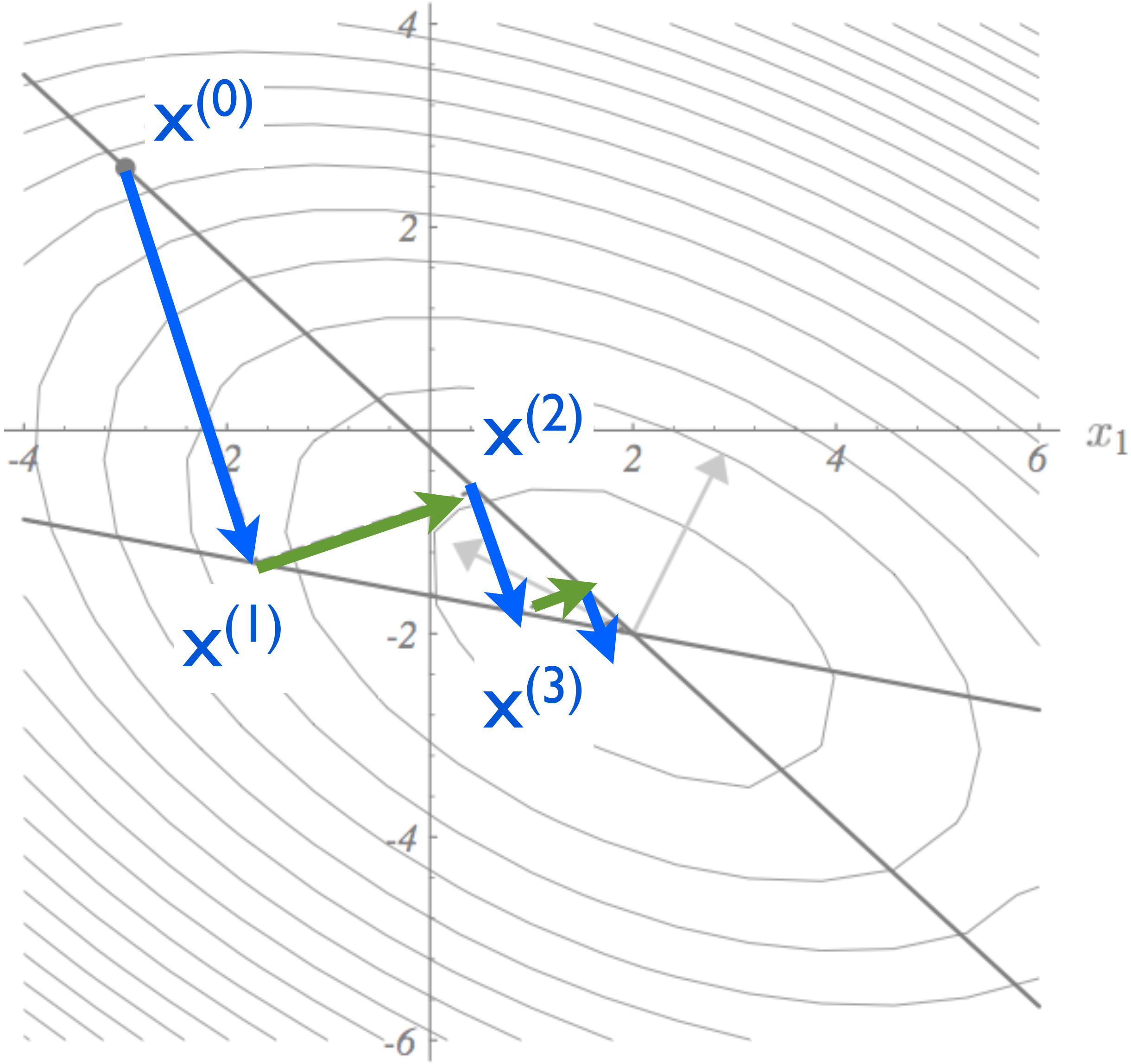


classic zigzag from steepest descent

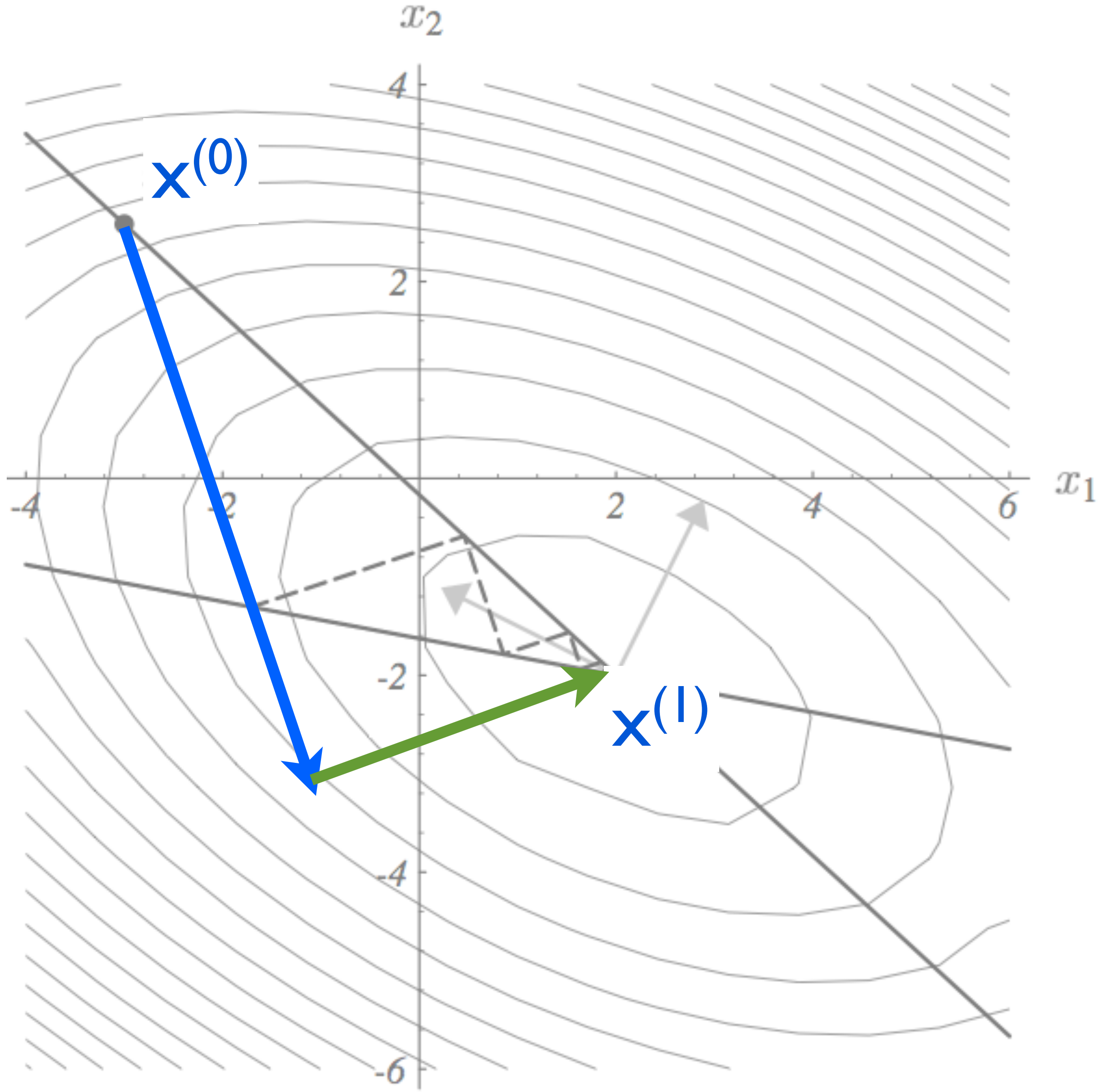


each “zig” and “zag” requires an iteration observation: moving in the **same direction** many times

(in example: just two directions)

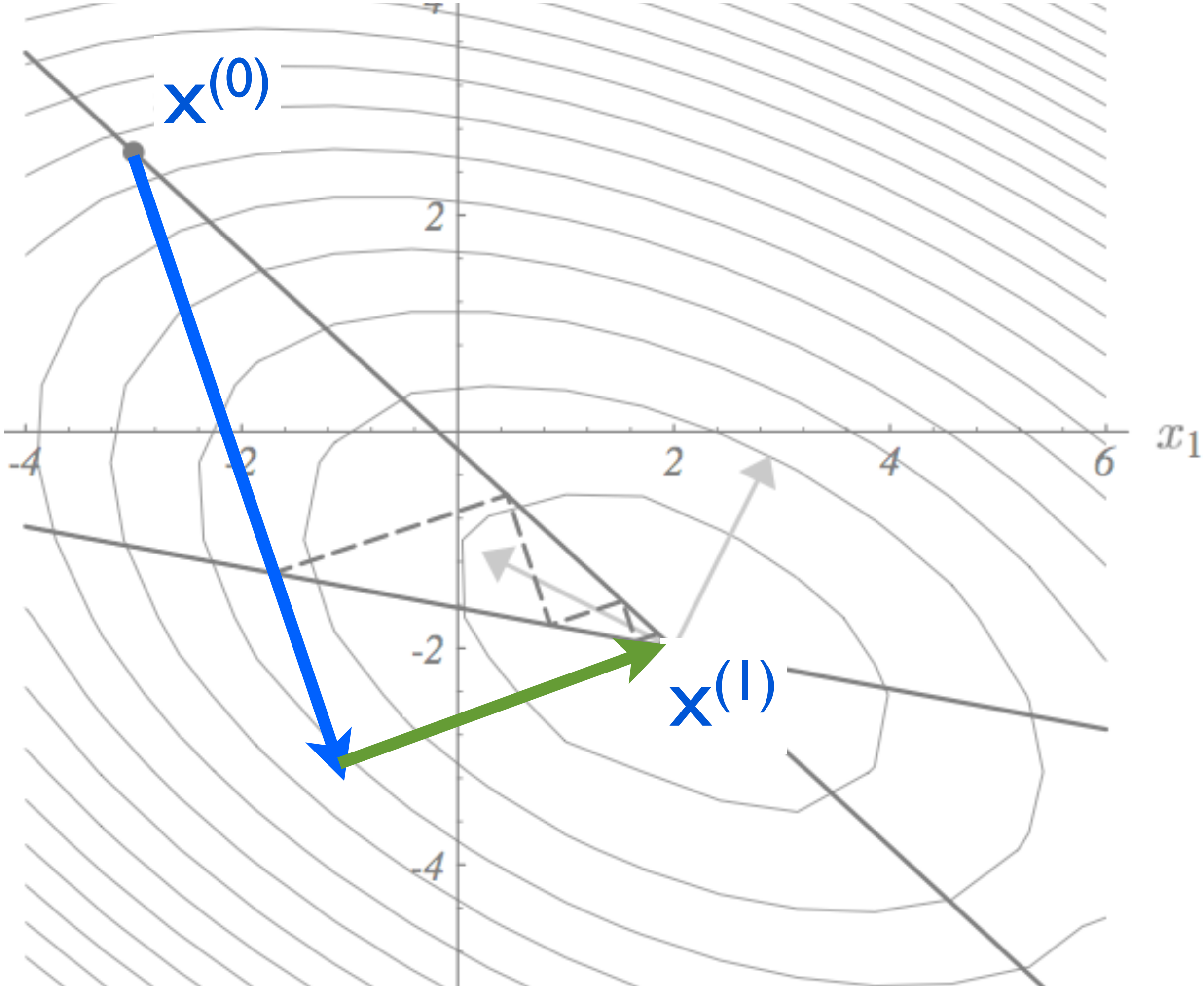


conjugate methods: combine all these zigzags into one step per "direction"

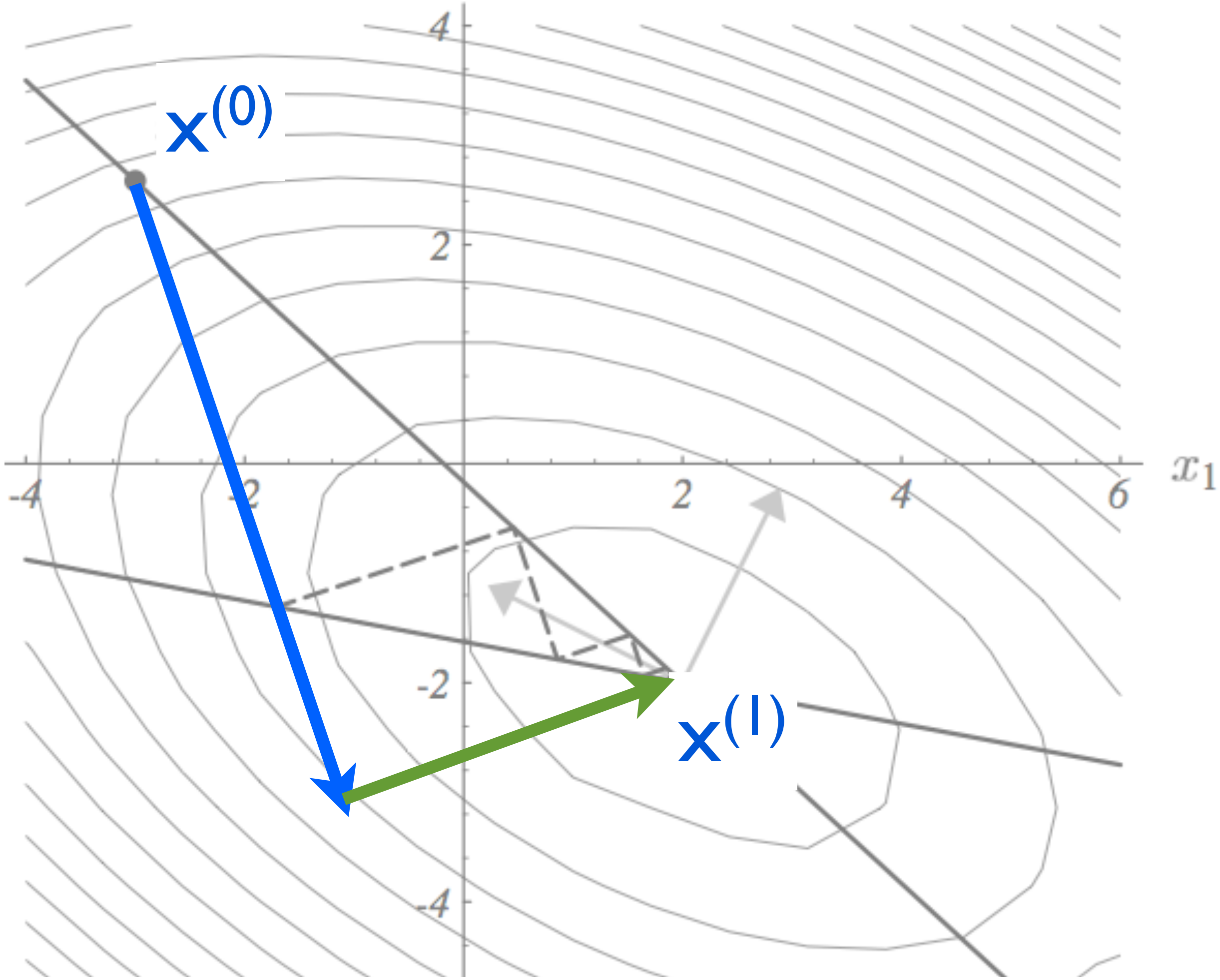


we don’t want any redundant steps, i.e. each direction should be “linearly independent”

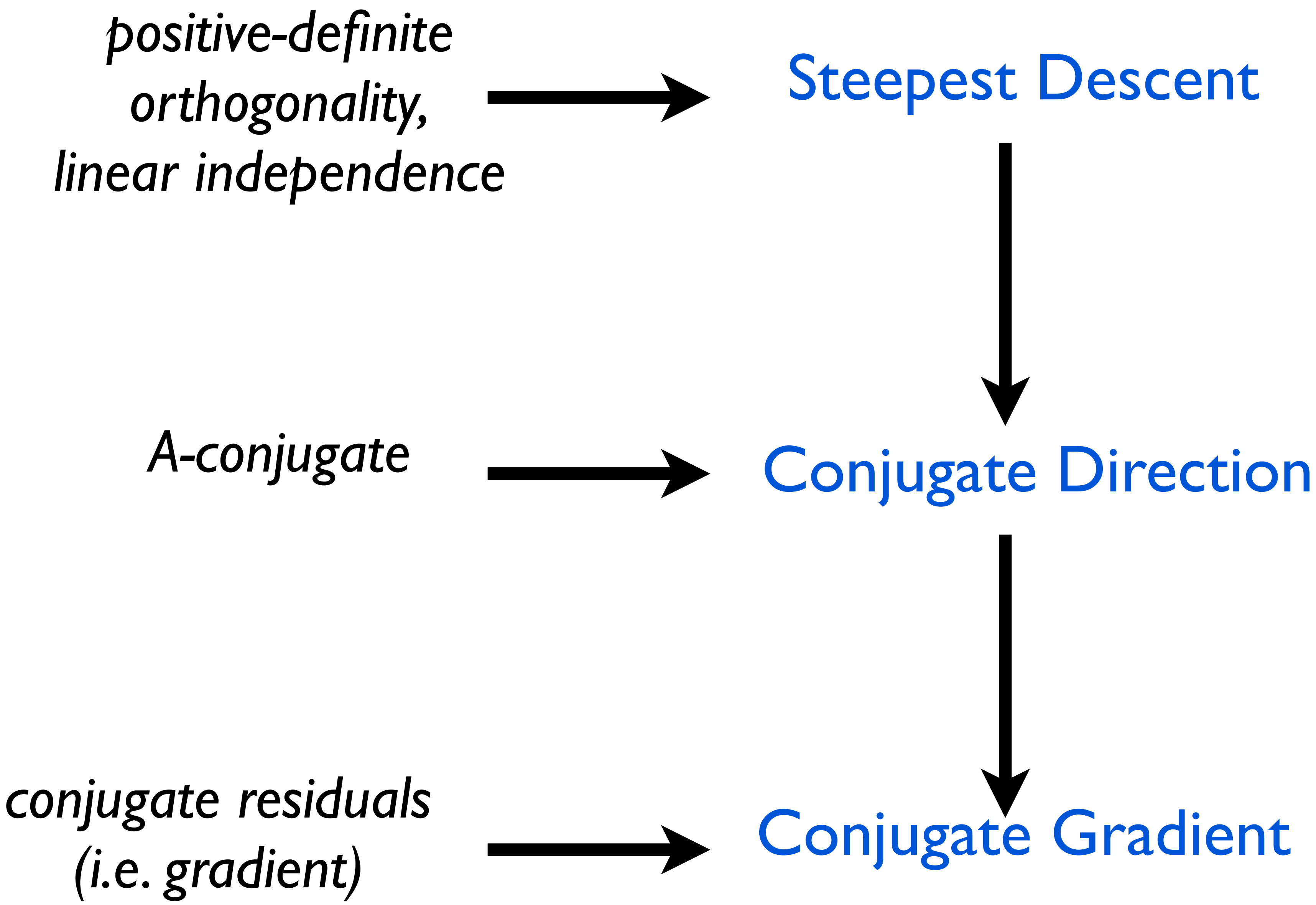
(that ‘s where the term “conjugate” comes in)



each direction gives us new “information” about the minimum that no other direction gives us conjugate **direction** method: find some good directions that are "conjugate" (can be computationally expensive)



conjugate **gradient** method: use the gradient to find these "conjugate" directions really efficiently (can even do this incrementally as we go)

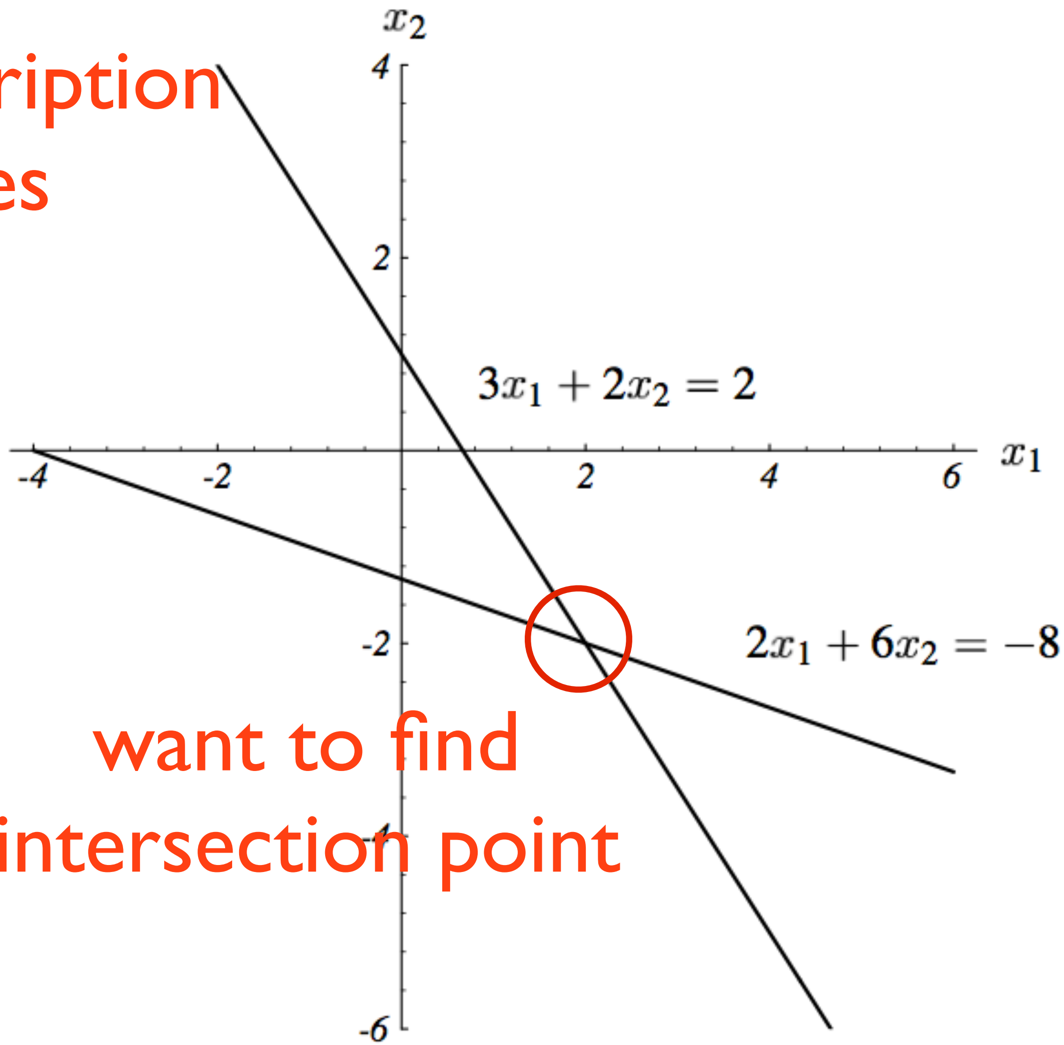

positive-definite orthogonality, linear independence
Steepest Descent
A-conjugate
Conjugate Direction
conjugate residuals (i.e. gradient)
Conjugate Gradient

Part 3

Steepest Descent

solving systems of linear equations

given description of lines



want to find intersection point

solving systems of linear equations

$$Ax = b$$

$$\begin{bmatrix} A_{11} & A_{12} & \dots & A_{1n} \\ A_{21} & A_{22} & & A_{2n} \\ \vdots & & \ddots & \vdots \\ A_{n1} & A_{n2} & \dots & A_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

given want this given

$$Ax - b = 0$$

this seems to be a **root-finding** task

Ax - b = 0

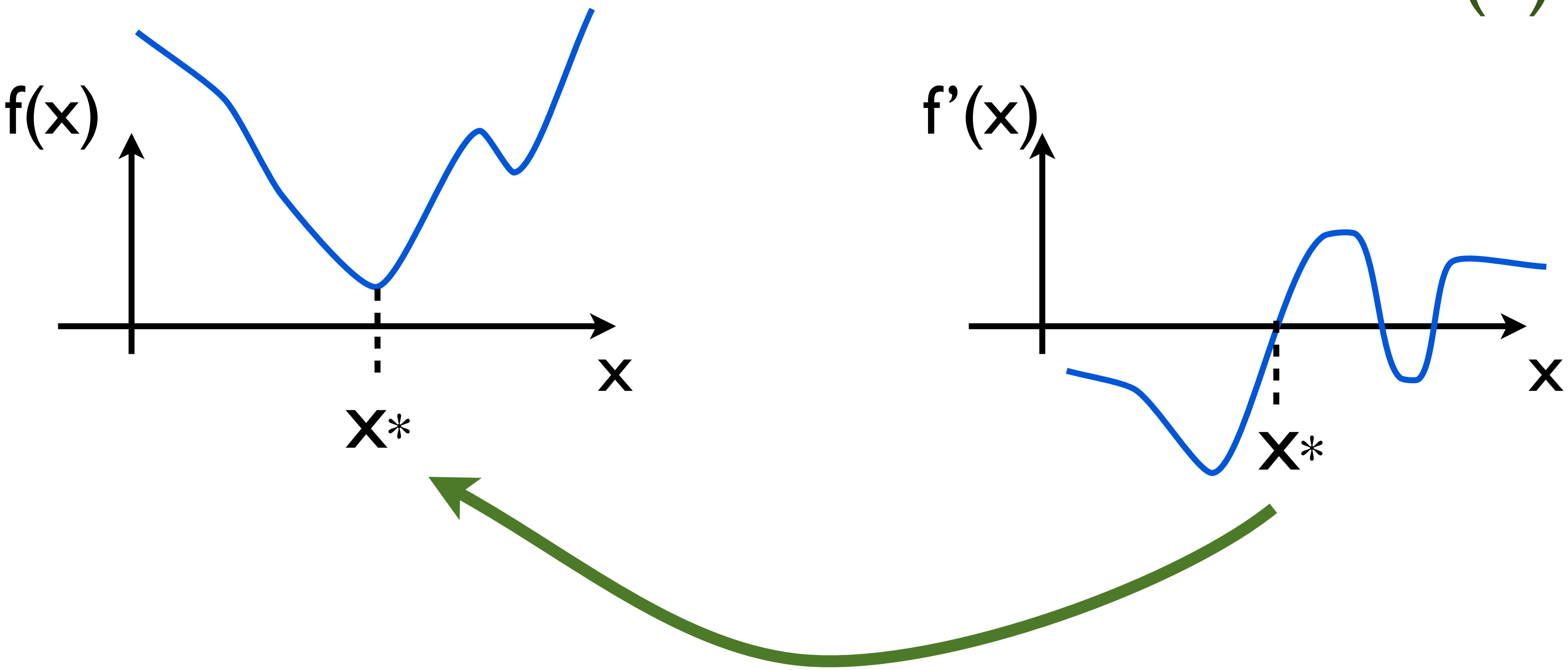
let's try to solve as optimisation task

from last lecture, what is relationship between root-finding and optimisation tasks?

? root of linear equations

root for linear equations is
minimum for quadratic equations

find root of f'(x)

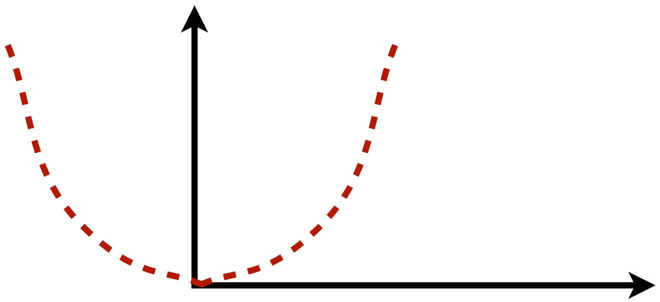


that'll give us the minimum of f(x)

1D case

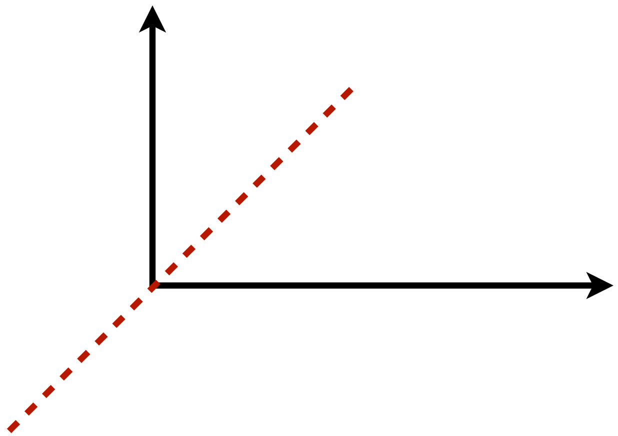
quadratic

?



line

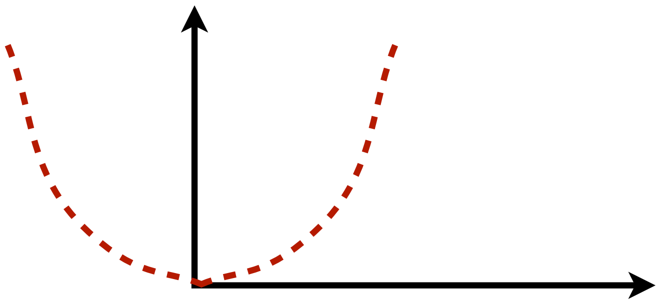
$f'(x)=mx^1 + c$



1D case

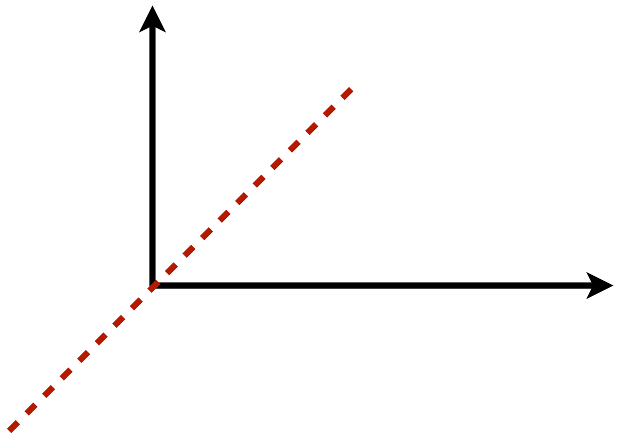
quadratic

$f(x)=(\frac{m}{2})x^2 + cx + k$

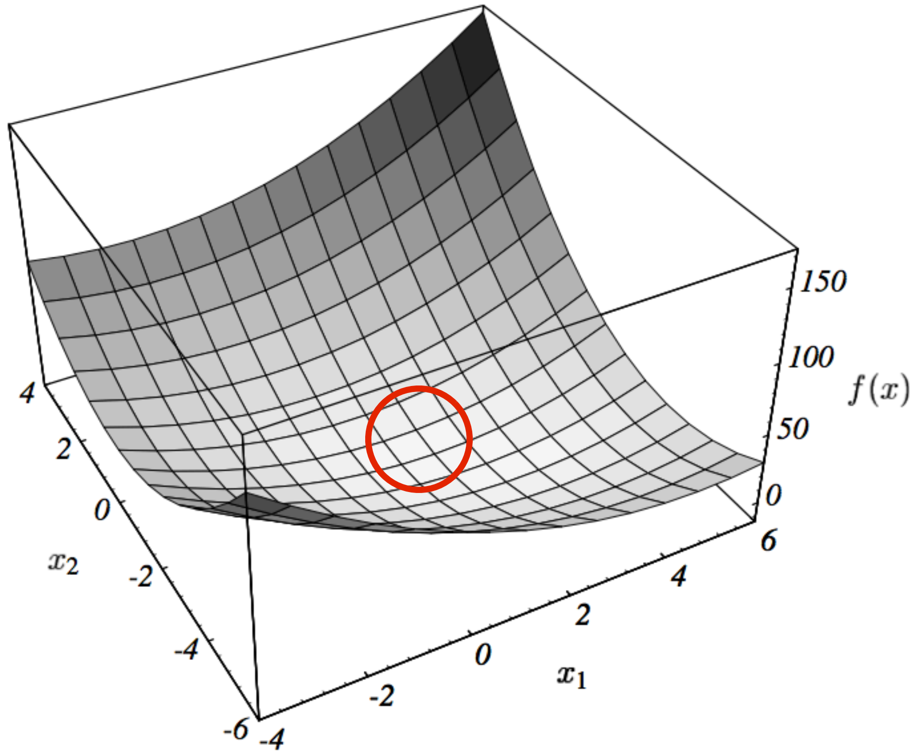


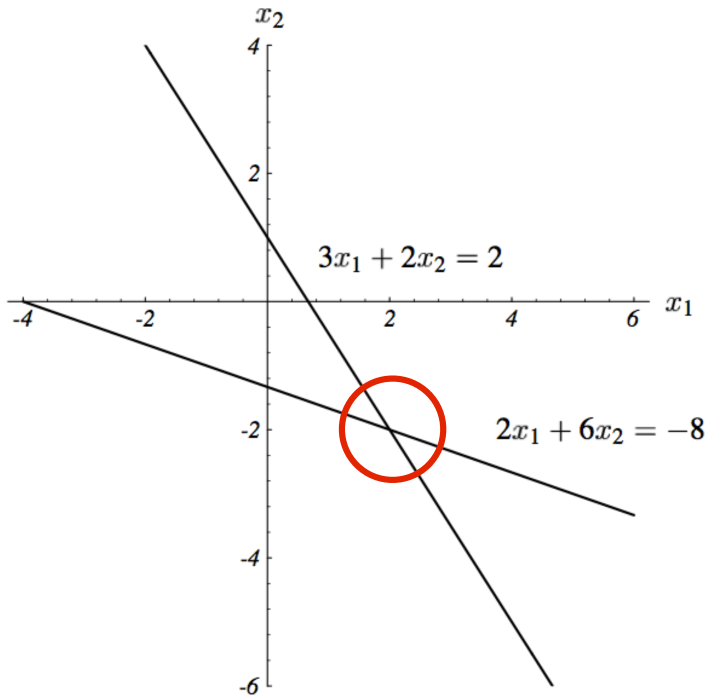
line

$f'(x)=mx^1 + c$



root for linear equations is
minimum for quadratic equations

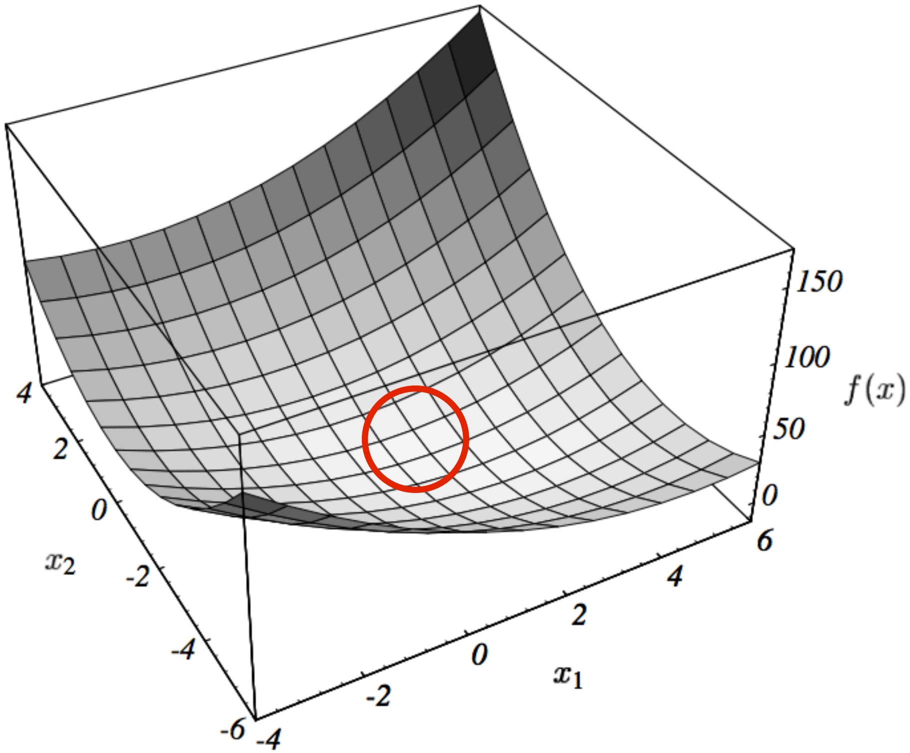


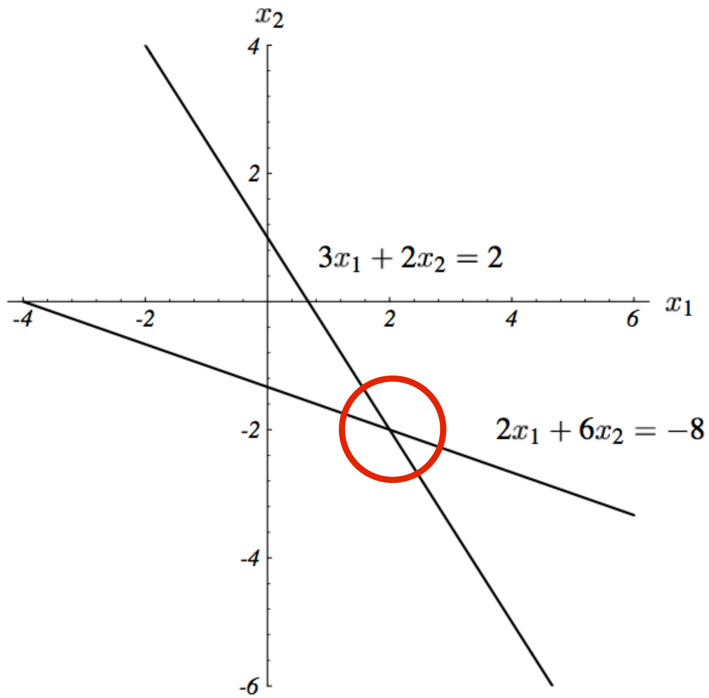


?

$f(X) = Ax - b$

root for linear equations is
minimum for quadratic equations

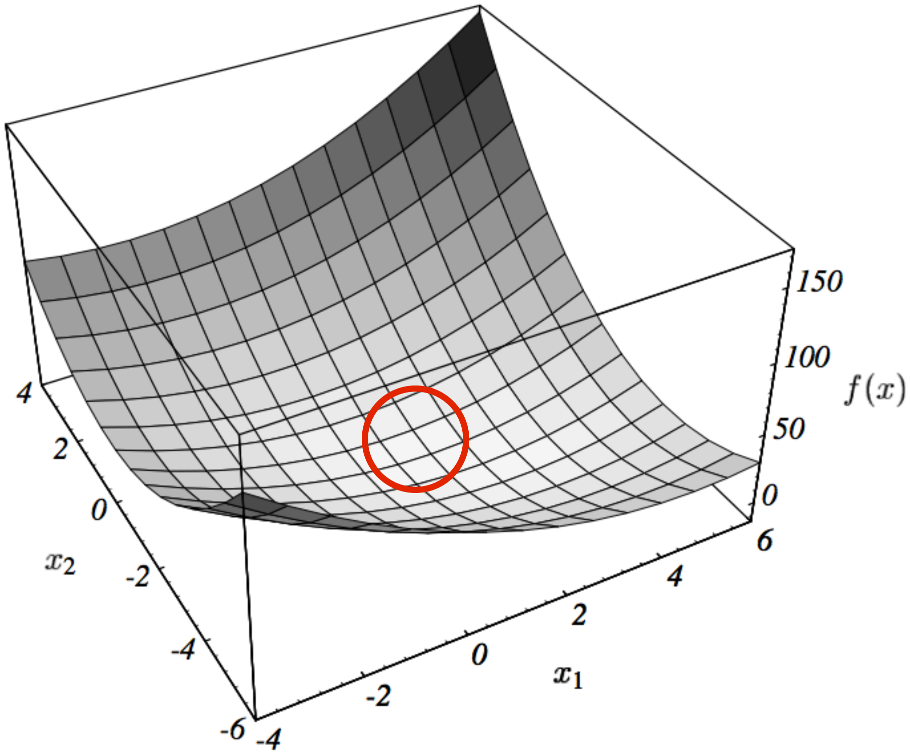


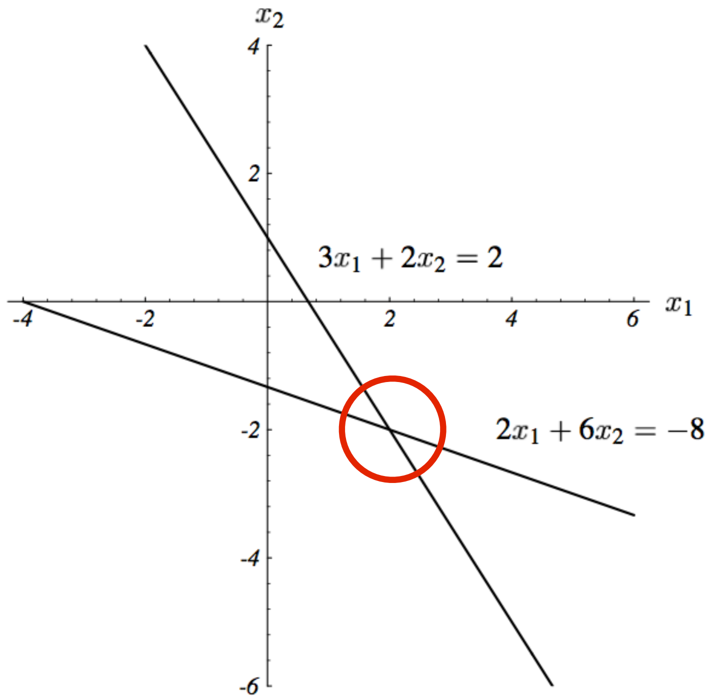


$$f(X) = \frac{1}{2} x^T A x - b^T x + c$$

$$f(X) = Ax - b$$

this particular relationship only holds if **A** is:
symmetric and **positive-definite**





$$f(X) = \frac{1}{2} x^T A x - b^T x + c$$

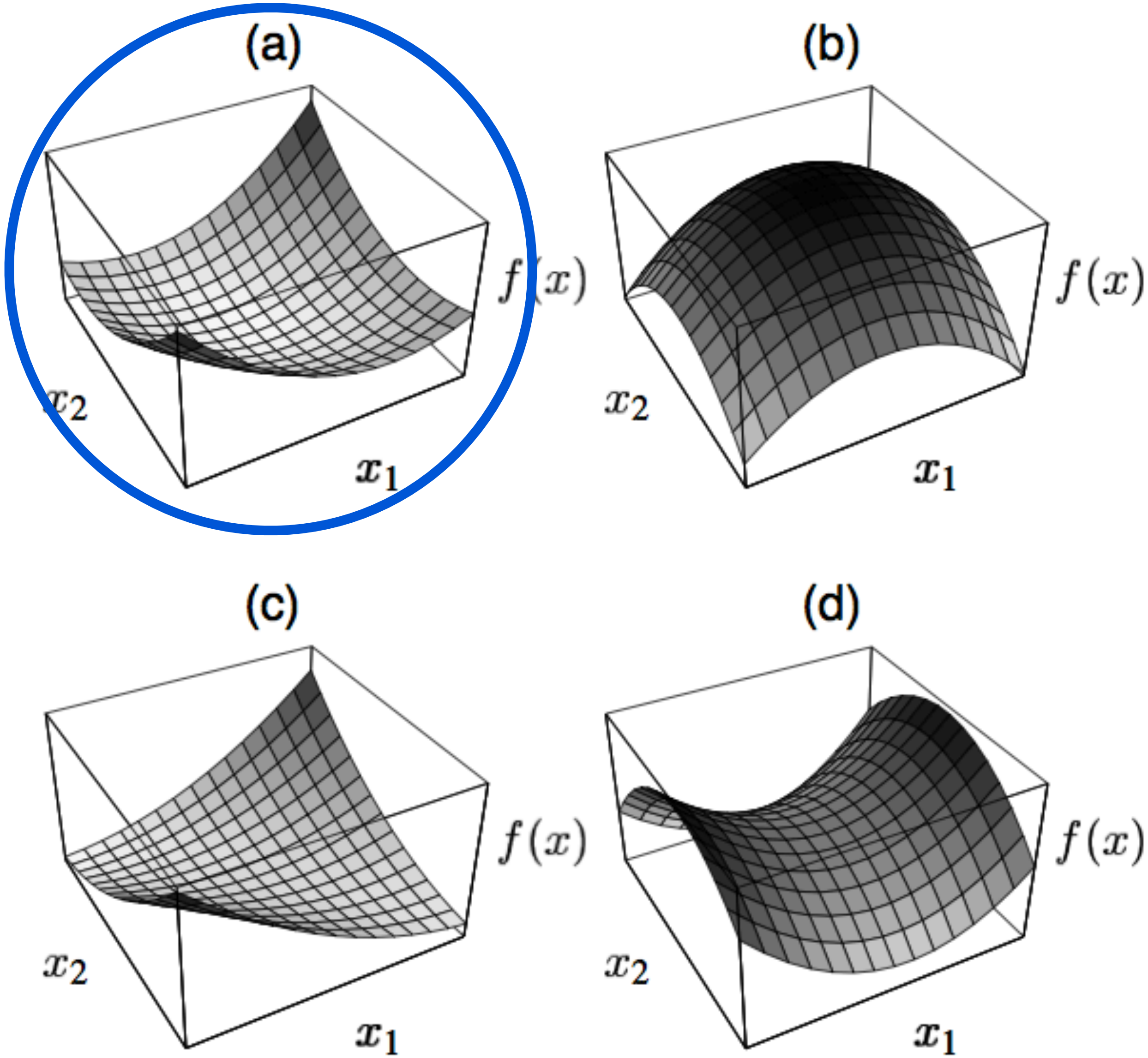
$$f(X) = Ax - b$$

Part 4

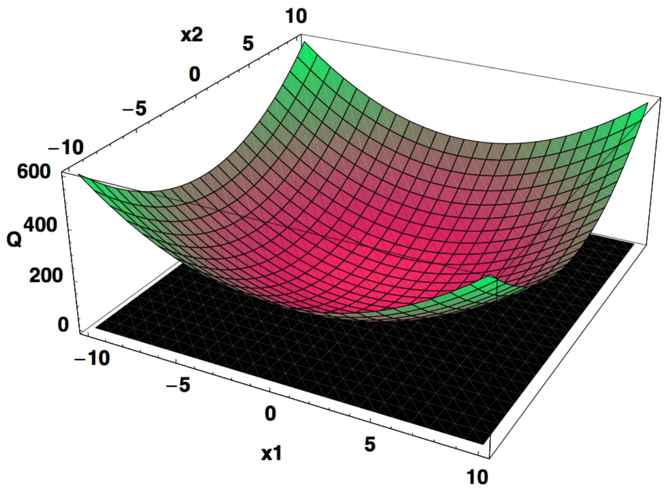
some concepts ...

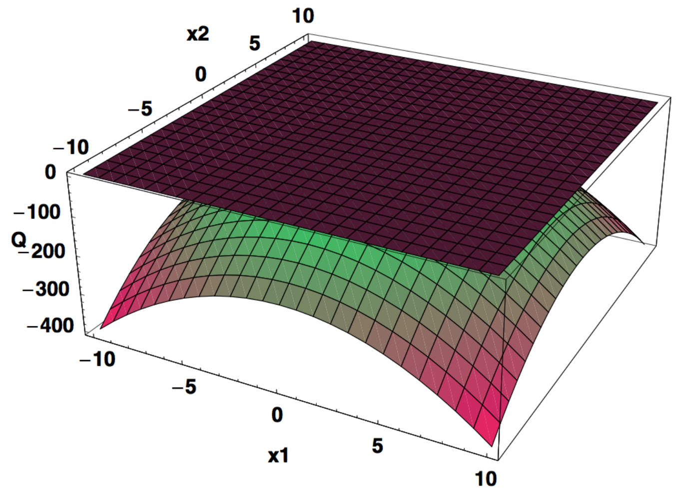
- positive definiteness
- orthogonal, linear independence
- error, residual

positive-definiteness describes bowl **shape** of paraboloid



for **non-zero** x, matrix A is....





positive definite

$x^T A x > 0$

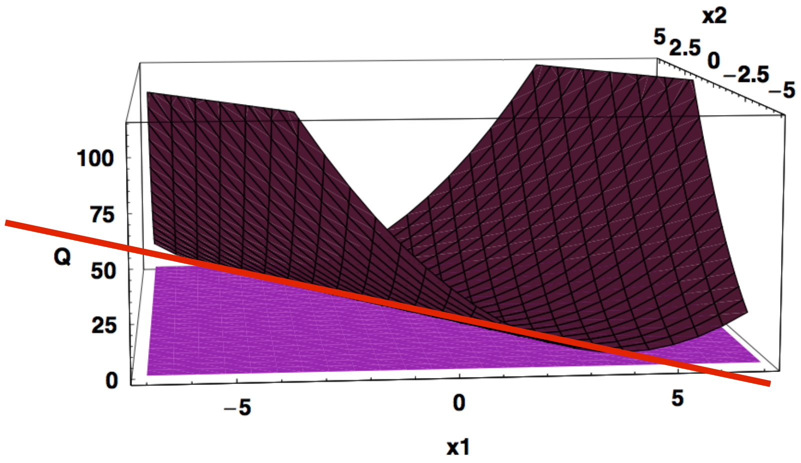
negative definite

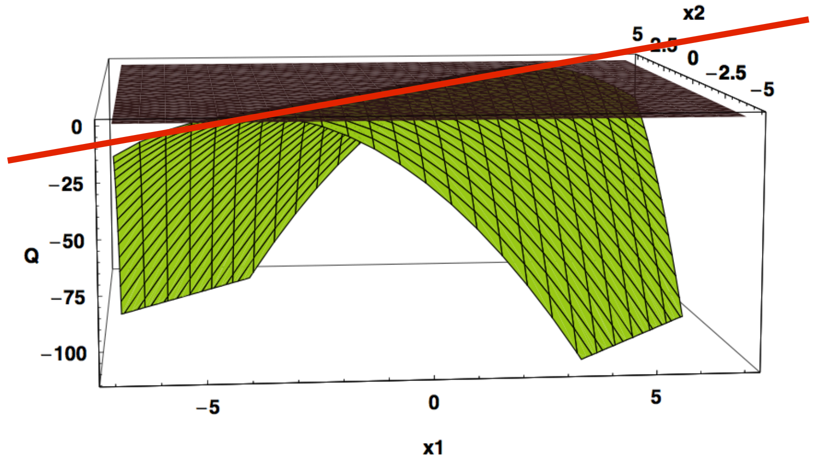
$x^T A x < 0$

Arne Hallam

www2.econ.iastate.edu/classes/econ501/Hallam/documents/Quad_Forms_000.pdf

for **non-zero** x, matrix A is....





positive semi-definite

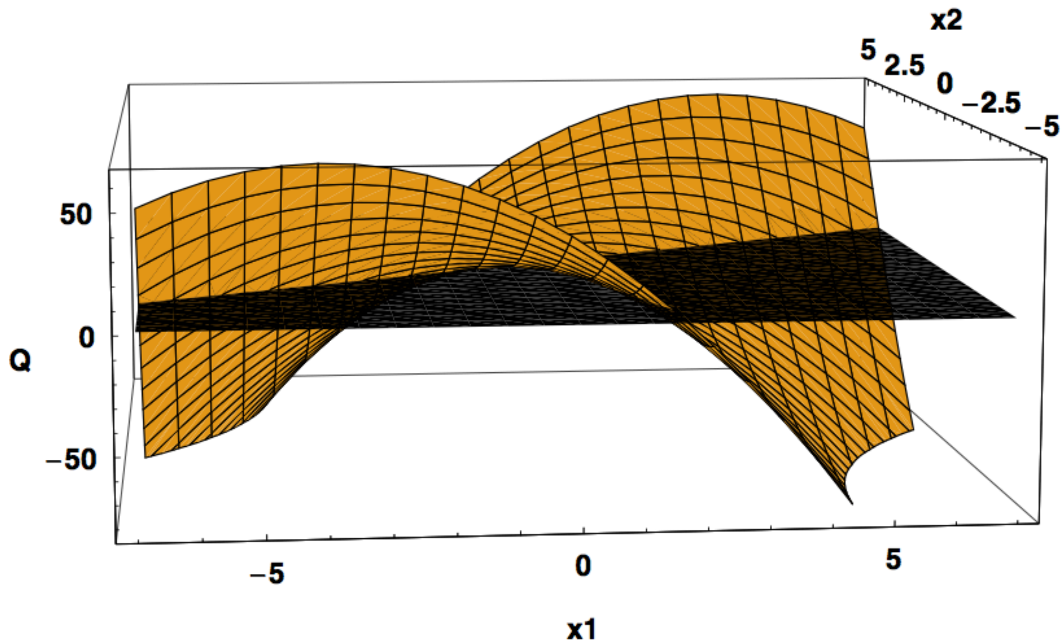
negative semi-definite

$x^T A x \geq 0$

$x^T A x \leq 0$

Arne Hallam

www2.econ.iastate.edu/classes/econ501/Hallam/documents/Quad_Forms_000.pdf for **non-zero** x, matrix A is....



indefinite

$x^T Ax$ sometimes ≥ 0, sometimes ≤ 0

Arne Hallam

www2.econ.iastate.edu/classes/econ501/Hallam/documents/Quad_Forms_000.pdf some intuition about this inequality:

$$x^T A x > 0$$

let's consider 1D case.

real value *a* is positive-definite if:

$$a x^2 > 0$$

...for any non-zero x

examples of *a*?

some intuition about this inequality:

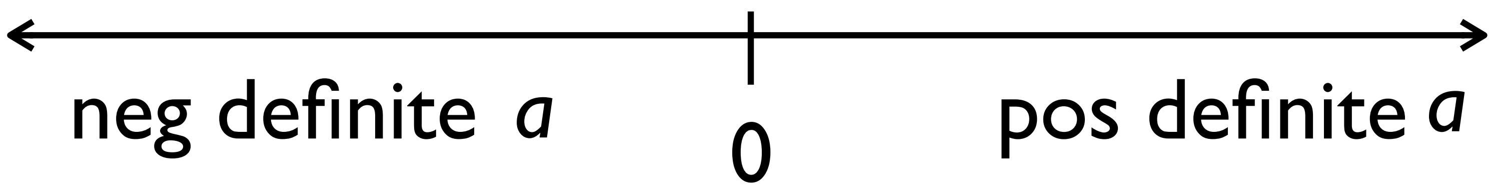
$$x^T A x > 0$$

let's consider 1D case.

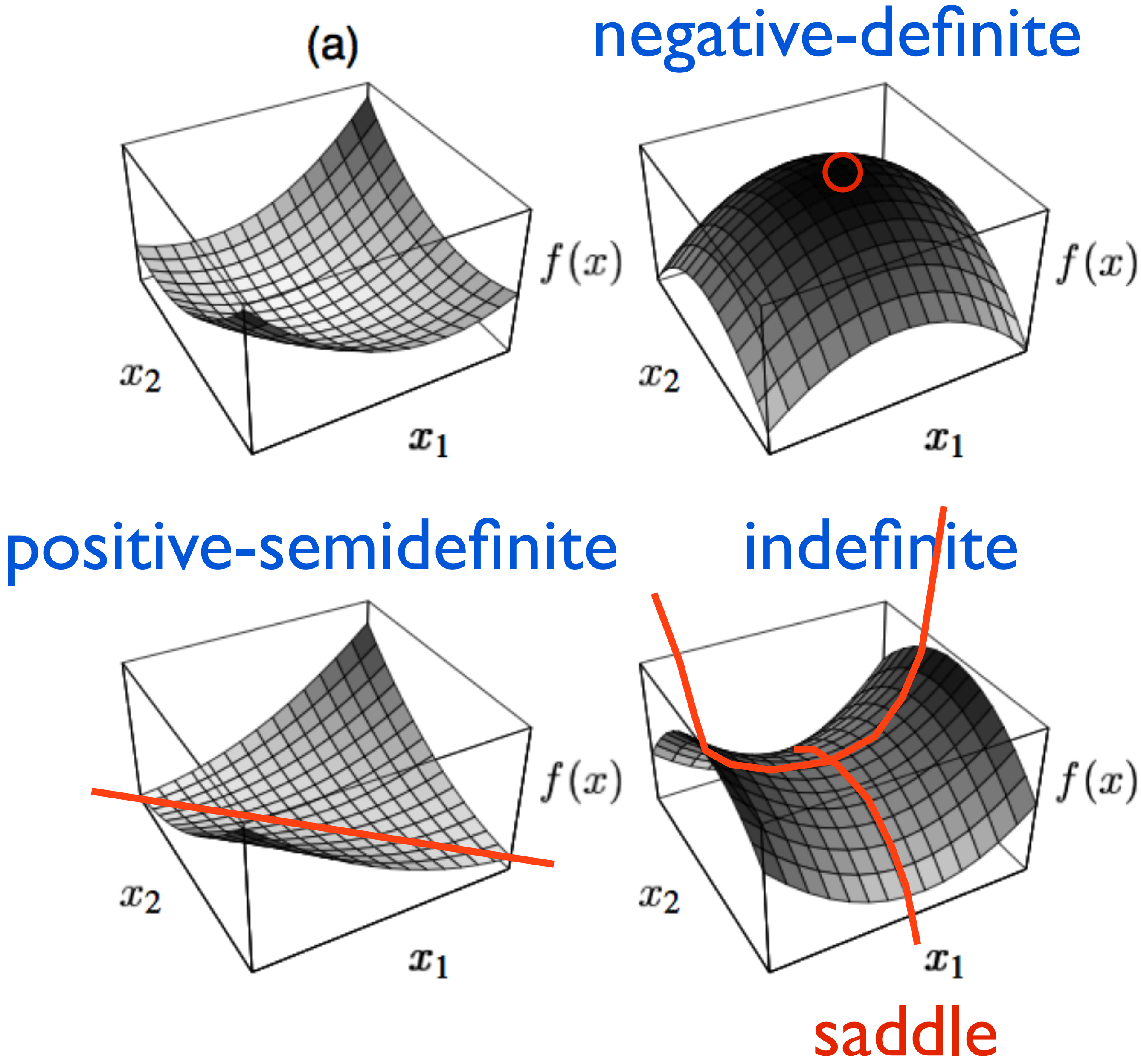
real value *a* is positive-definite if:

$$a x^2 > 0$$

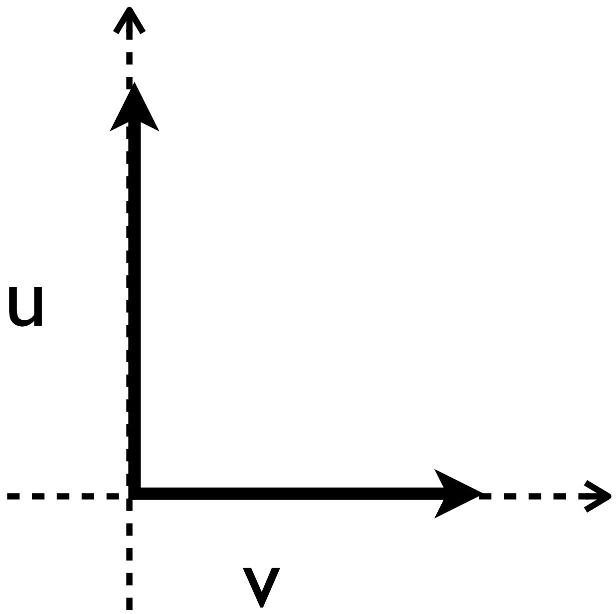
...for any non-zero x

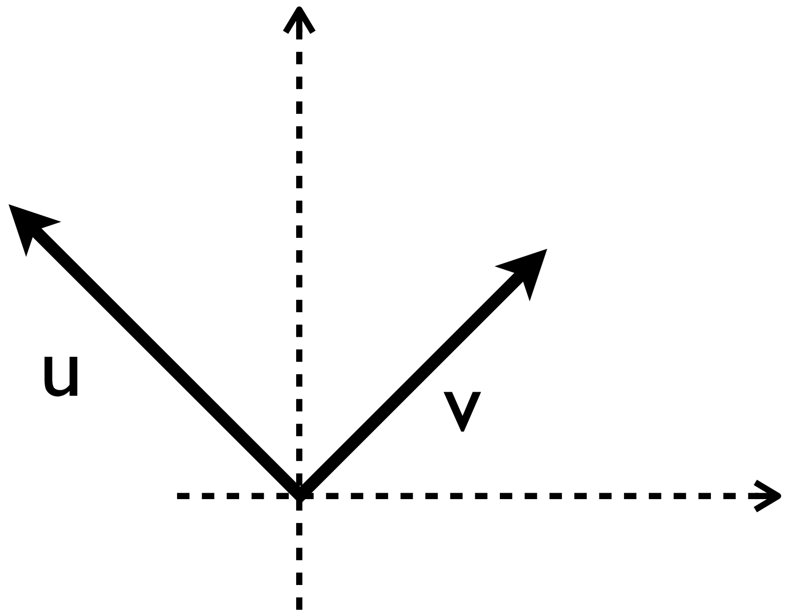


positive-definiteness describes bowl shape of paraboloid



orthogonal

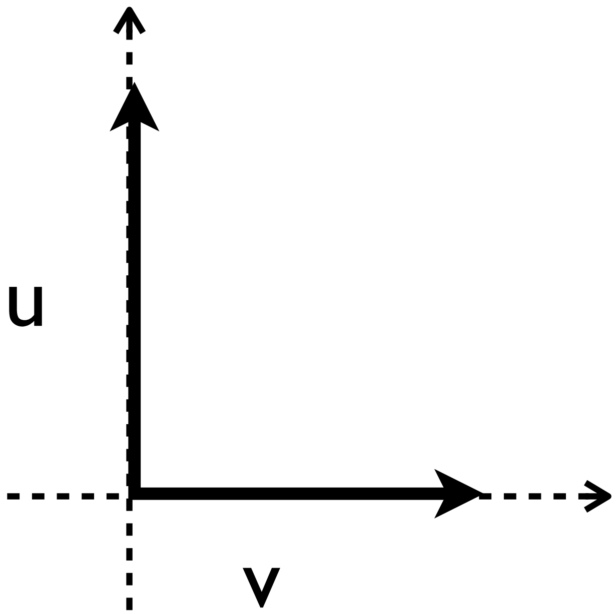


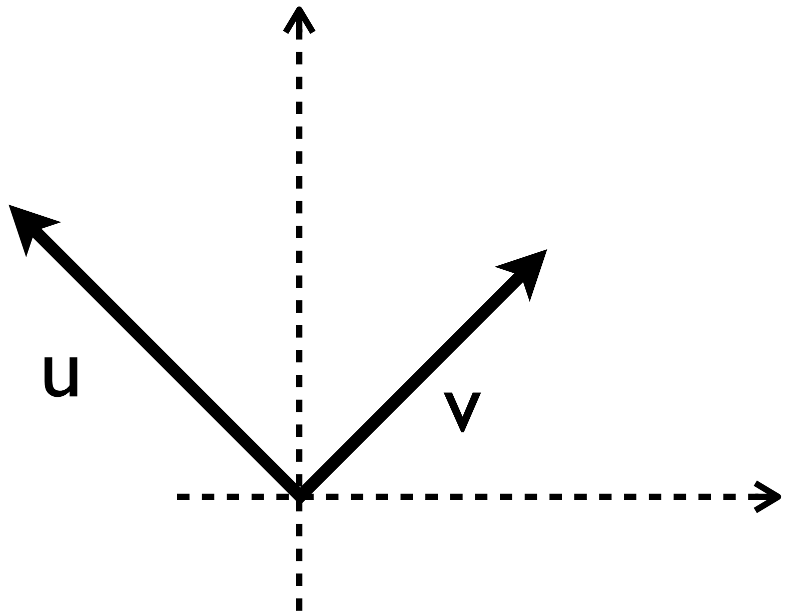


$$u \cdot v = 0$$

(u,v are perpendicular)

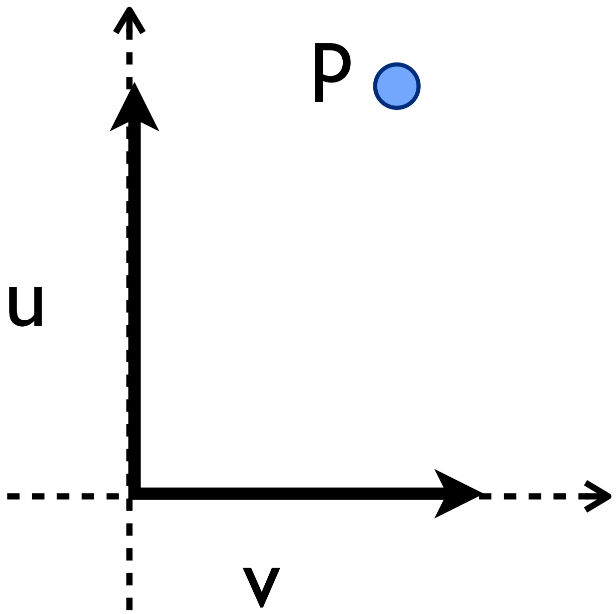
orthogonal

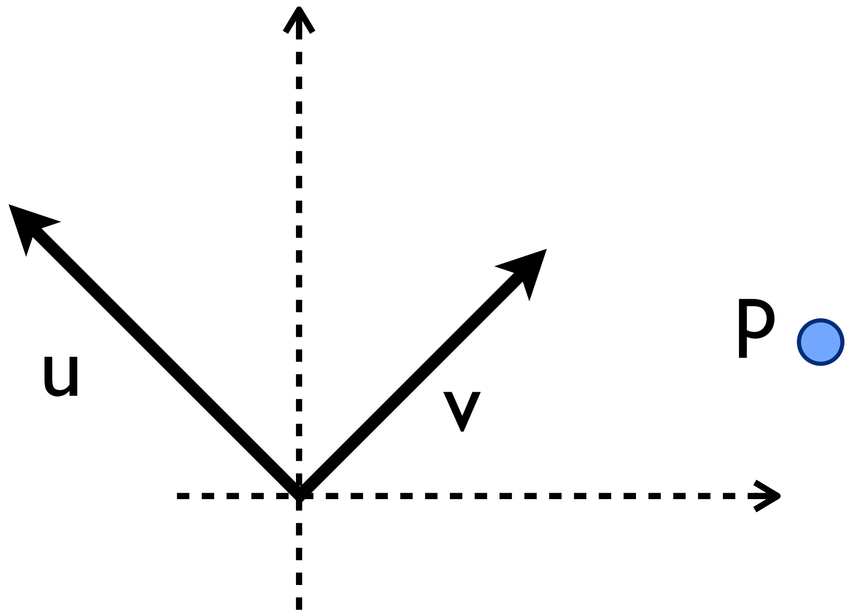




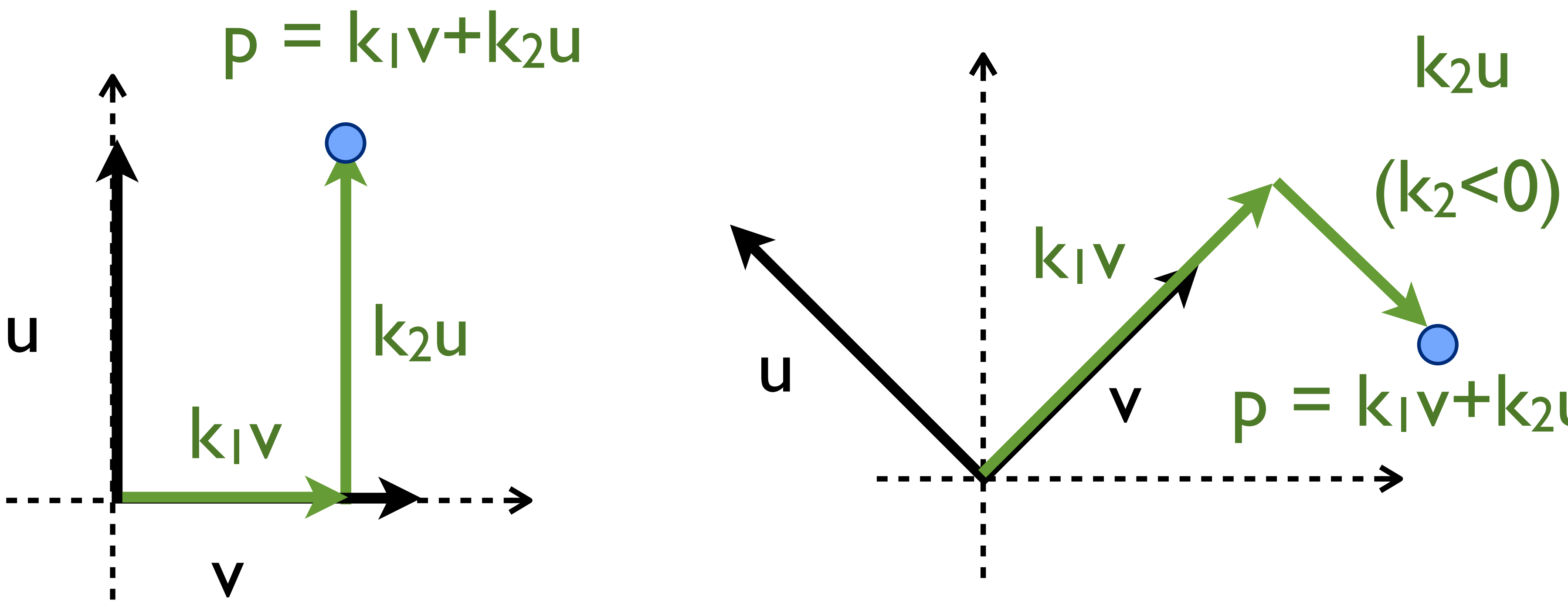
e.g. we need both these vectors to describe all other 2D points - both u,v contribute some useful, unique "information", no redundancy

orthogonal





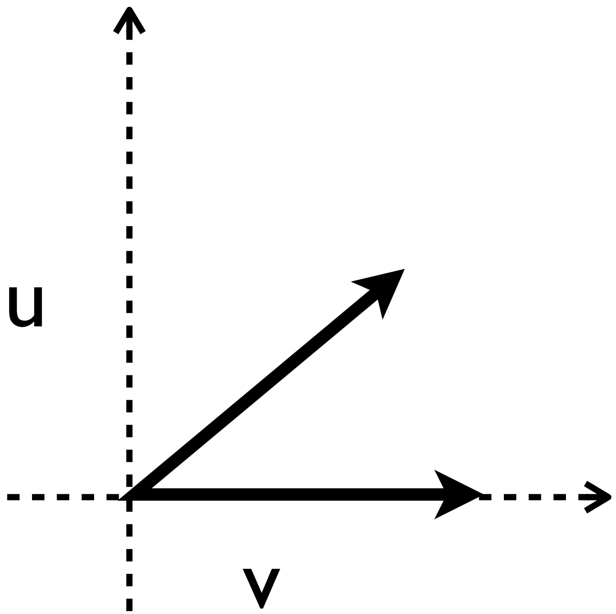
orthogonal

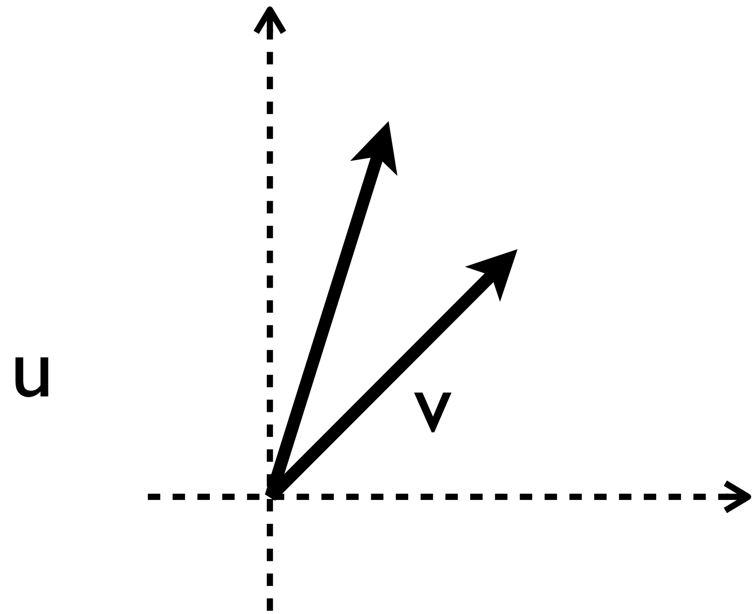


linear combinations of u,v

what about these vectors - definitely not orthogonal!

why not orthogonal?

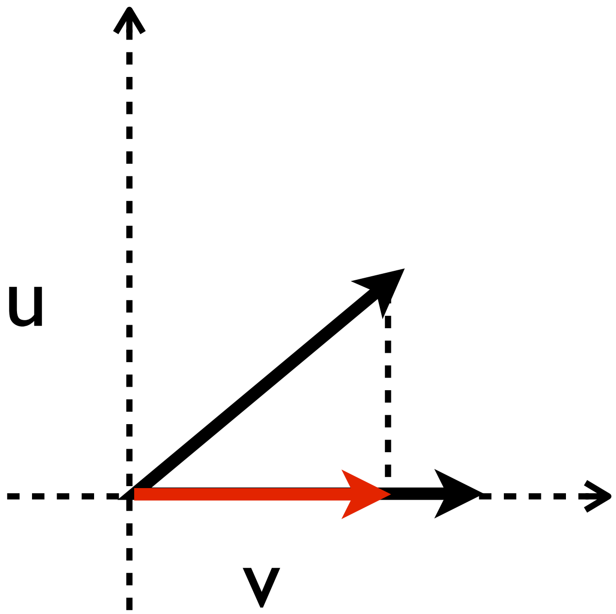


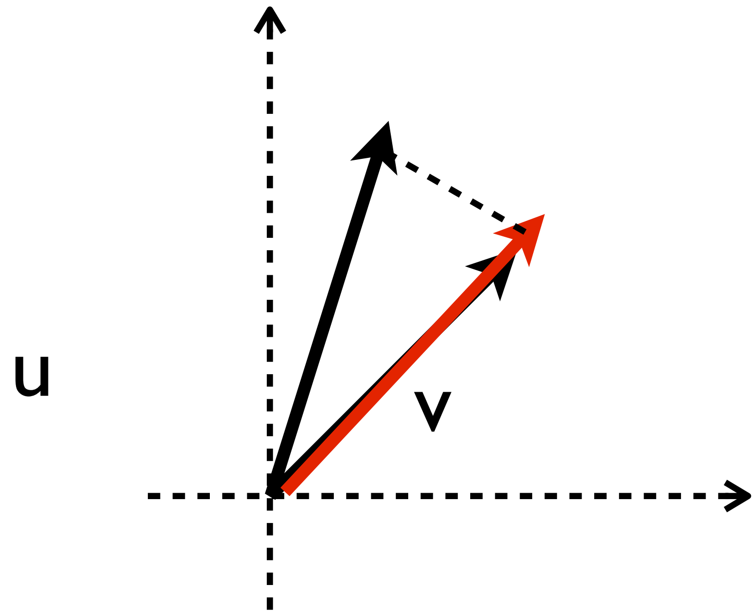


what about these vectors - definitely not orthogonal!

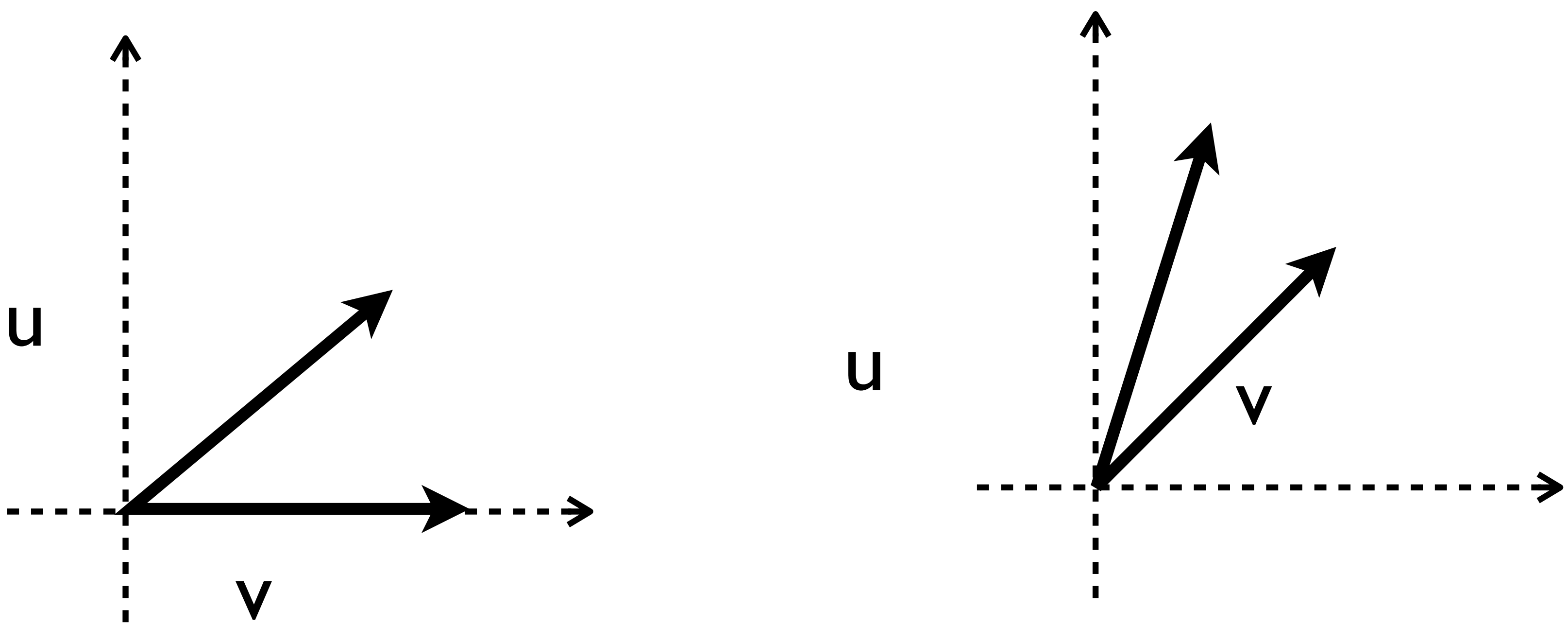
dot product not zero:

$u \cdot v \neq 0$

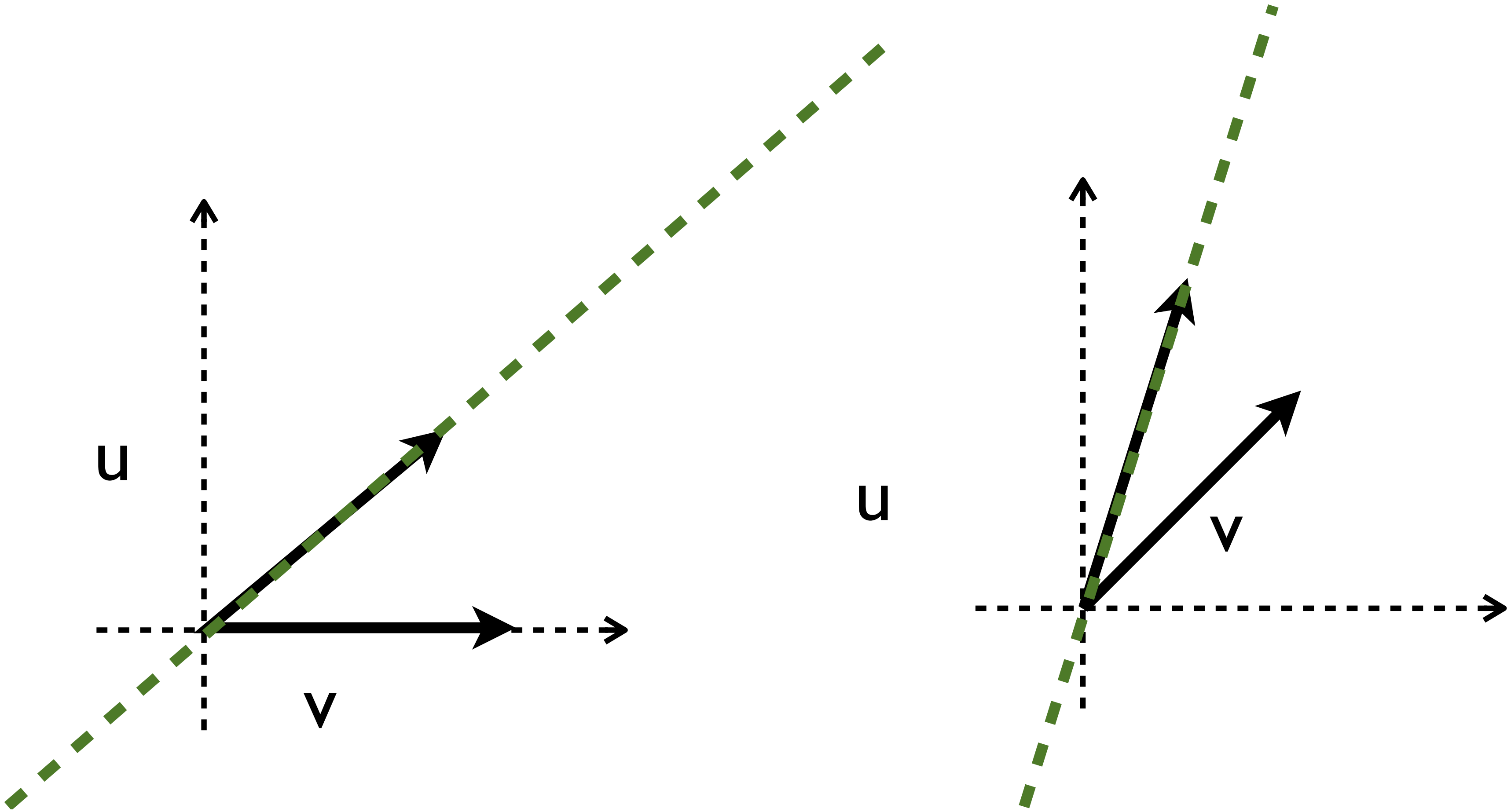




intuition: vector projection (casting a "shadow" onto another vector) related to dot product is there any redundancy? i.e. can vector *u* be used to describe *v* (or vice versa)?

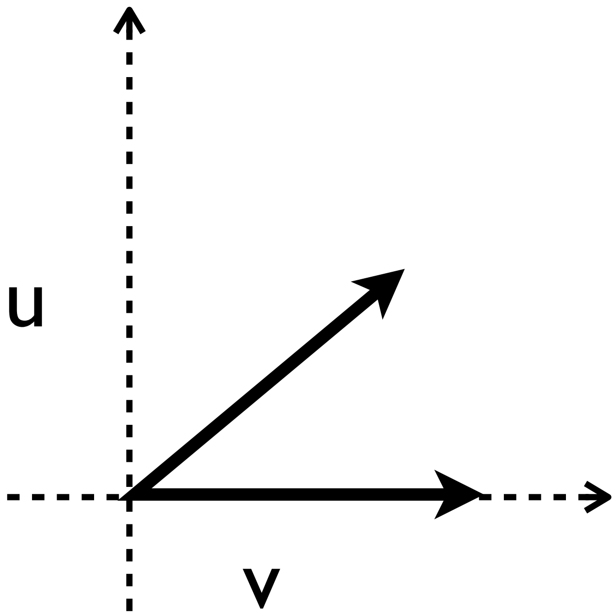


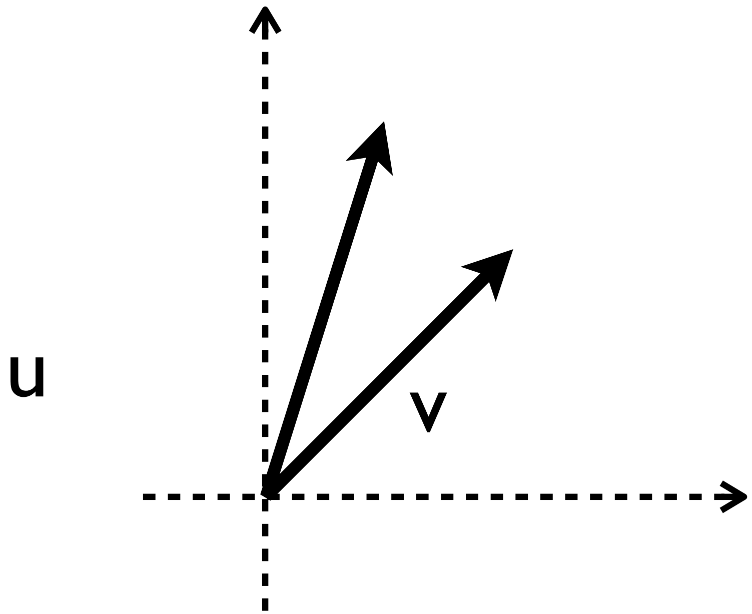
“describe” means: define *v* as linear combination of *u* linear combination: can (1) scale vectors, and (2) add them together



so, not orthogonal, but each still contributing unique "information" - **linearly independent**

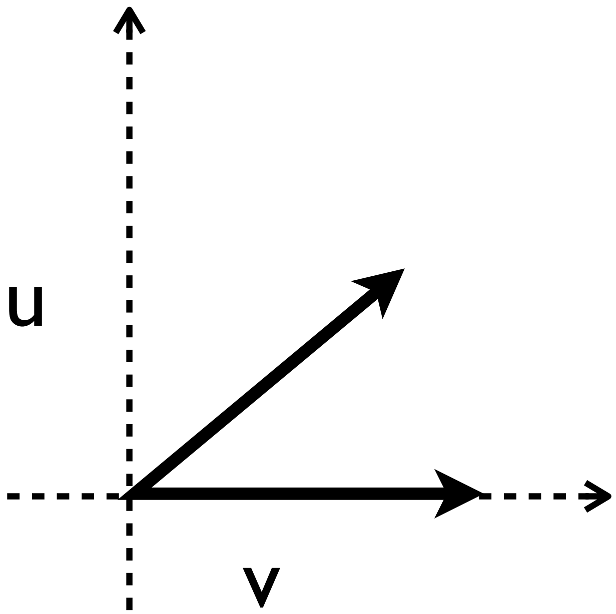
can I use them to describe **any** 2D point?

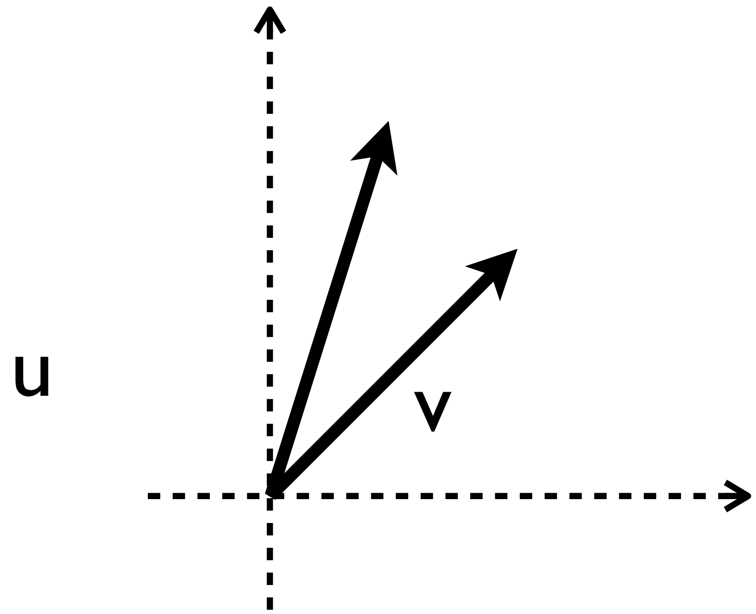




can I use them to describe **any** 2D point?

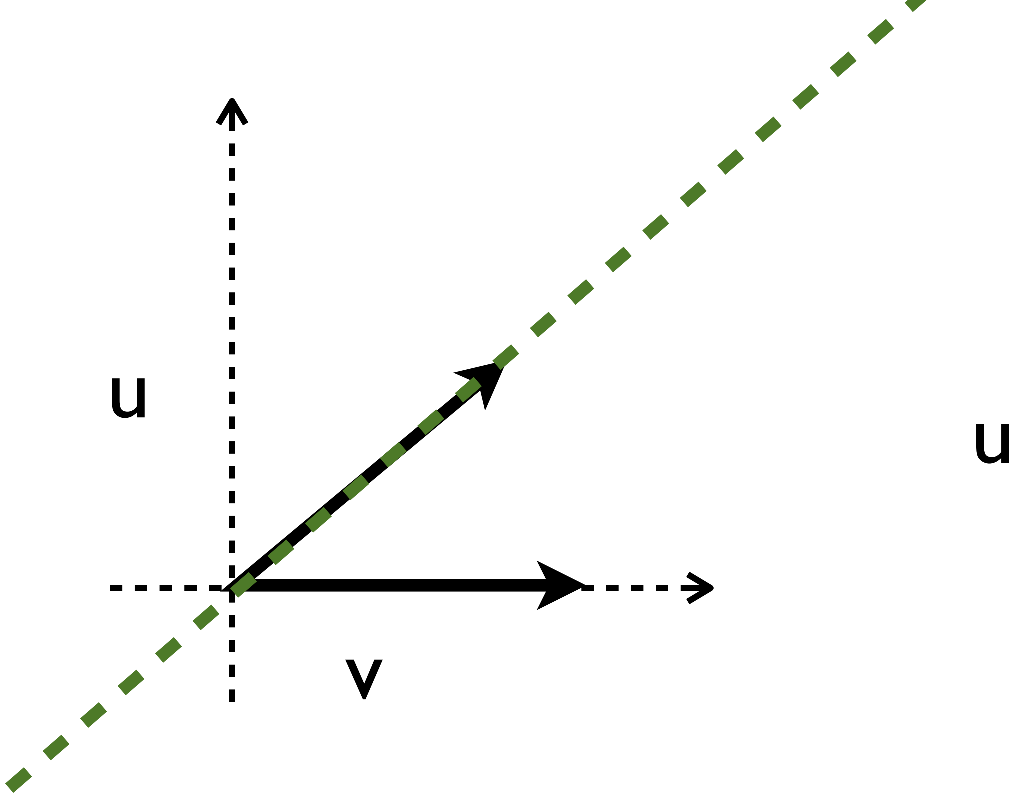
yes! notion of **span** - span of u,v is set of all linear combinations of u,v

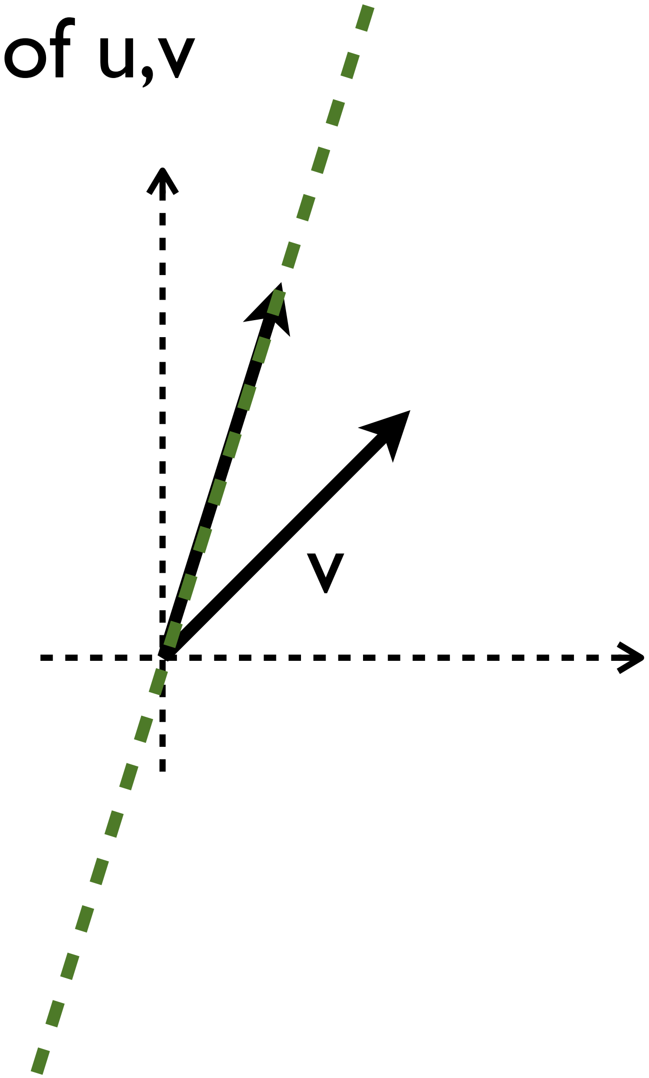




can I use them to describe **any** 2D point?

yes! notion of **span** - span of u,v is set of all linear combinations of u,v



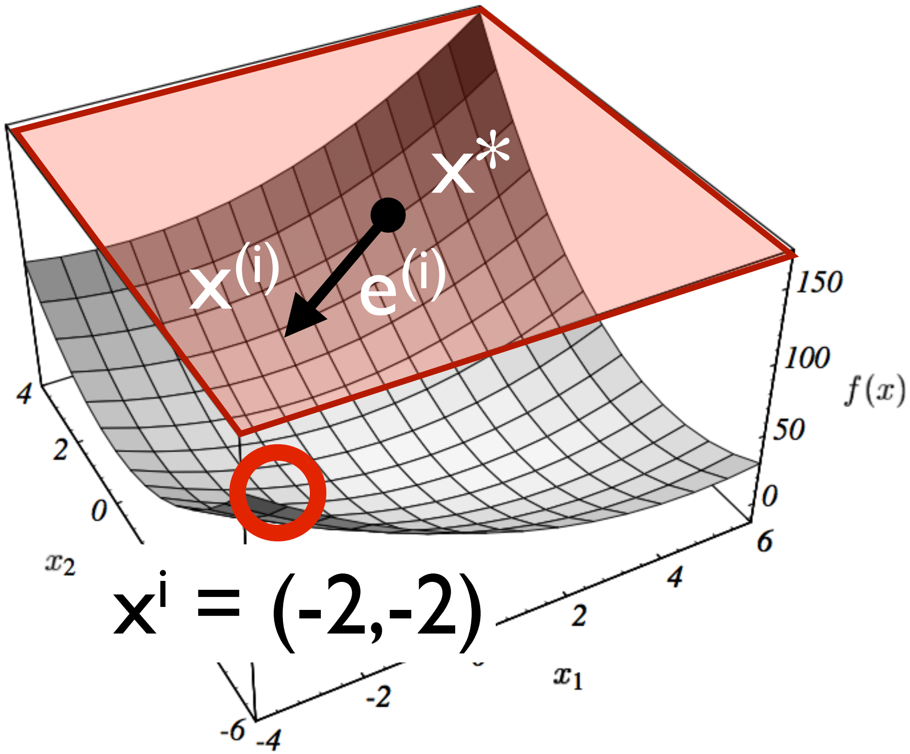


we can make u as long as we like, then add v (which we can also make as long as we like)

steepest descent "slides" down bowl

error: distance from current $x^{(i)}$ to solution x^*

$$e^{(i)} = x^{(i)} - x^*$$

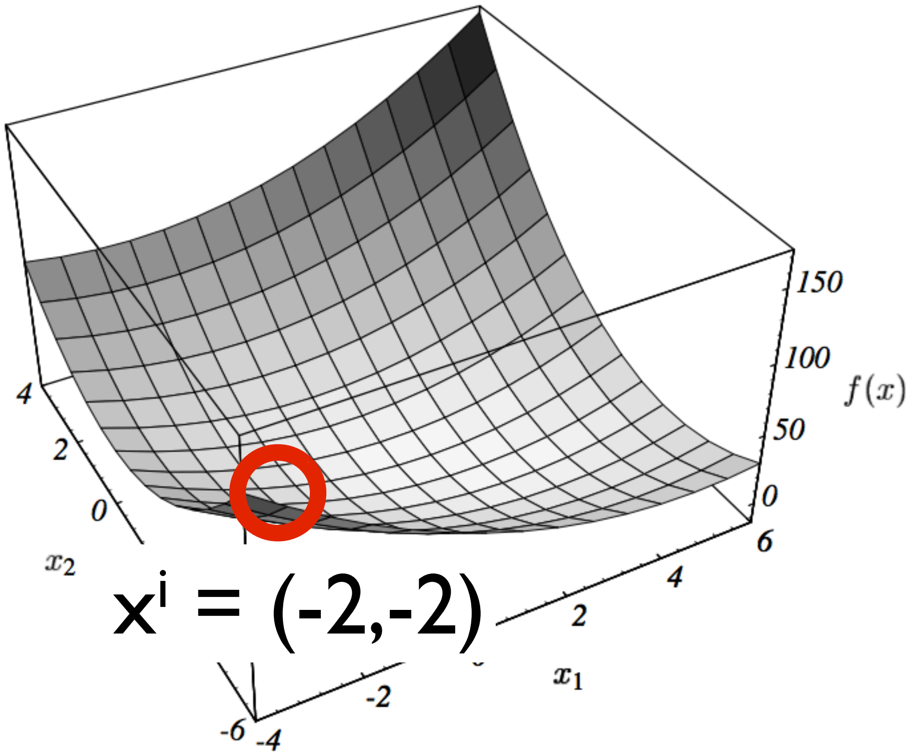


$x^i = (-2,-2)$

steepest descent "slides" down bowl

residual: how far we are from making "Ax" equal "b"

$$r^{(i)} = b - Ax$$



$x^i = (-2,-2)$

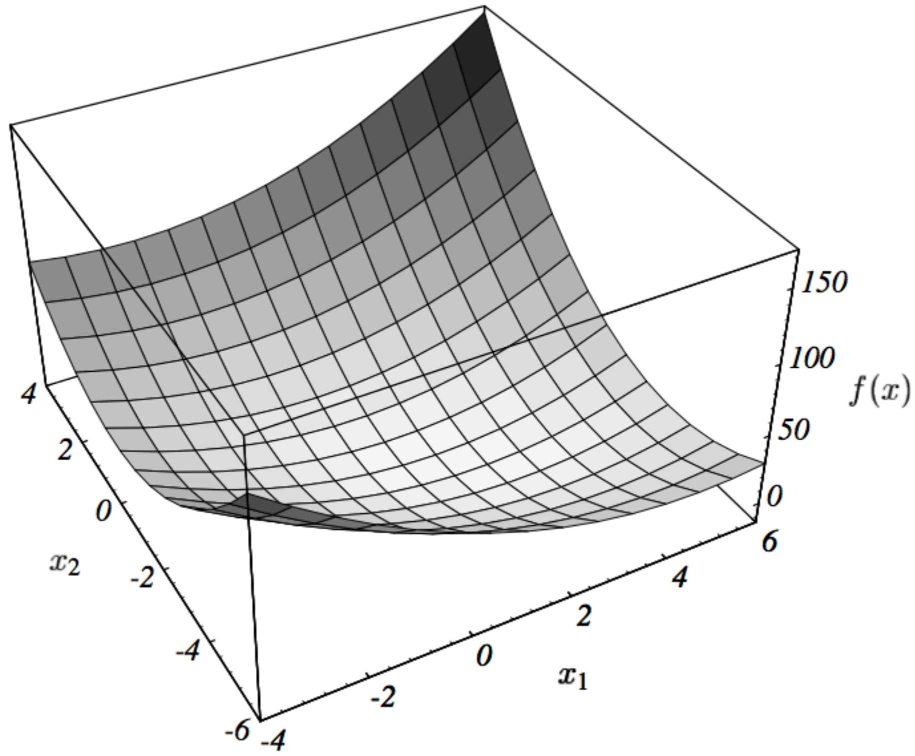
SUMMARY Part 4. concepts

- positive definiteness
- orthogonal, linear independence
- error, residual

back to steepest descent

steepest descent "slides" down bowl

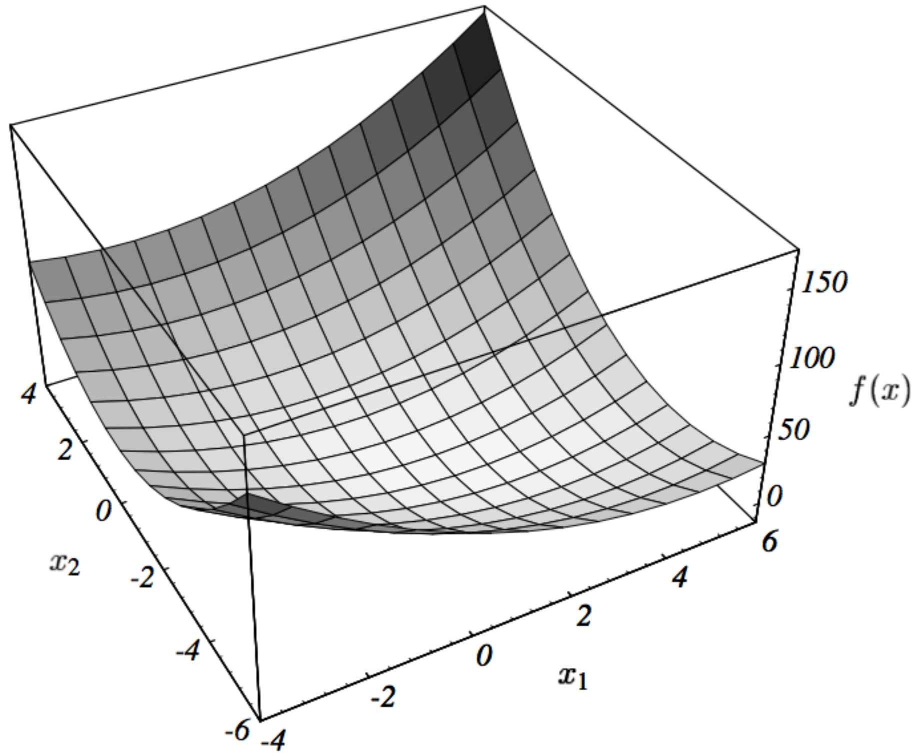
1. pick point $x^{(0)}$ e.g. $x^{(0)}=(-2,-2)$
2. pick "steepest" direction to move $r^{(0)}$
3. pick step size α



$$x^{(1)} = x^{(0)} + \alpha\, r^{(0)}$$

steepest descent "slides" down bowl

2. pick "steepest" direction to move:
   opposite of gradient at $x^{(i)}$



steepest descent “slides” down bowl

2. pick “steepest” direction to move:
   opposite of gradient at $x^{(i)}$

$$\nabla f(X) = ?$$

steepest descent "slides" down bowl

2. pick "steepest" direction to move: opposite of gradient at $x^{(i)}$

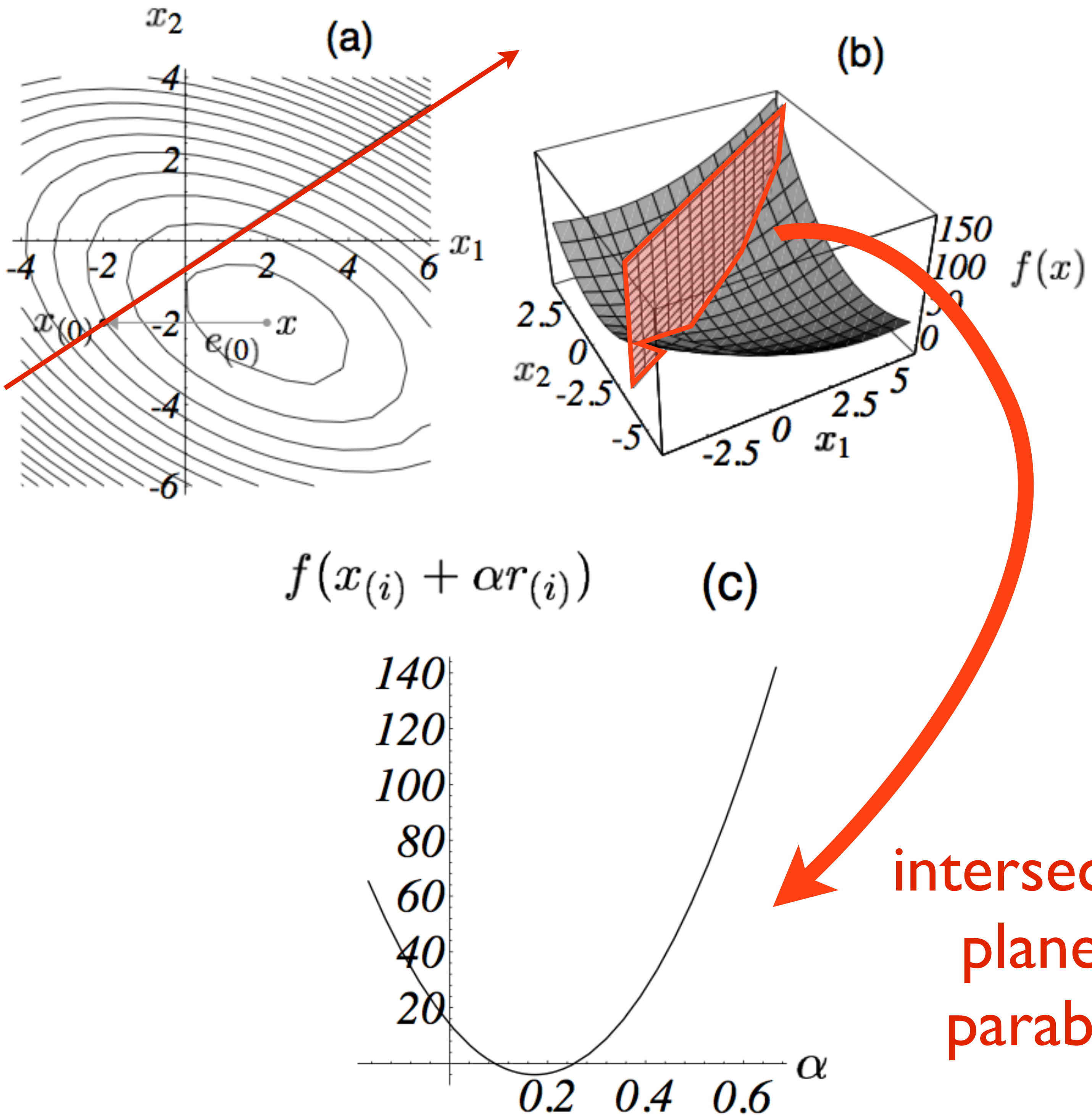
$$\nabla f(X) = Ax - b$$

(it's the original equation...)

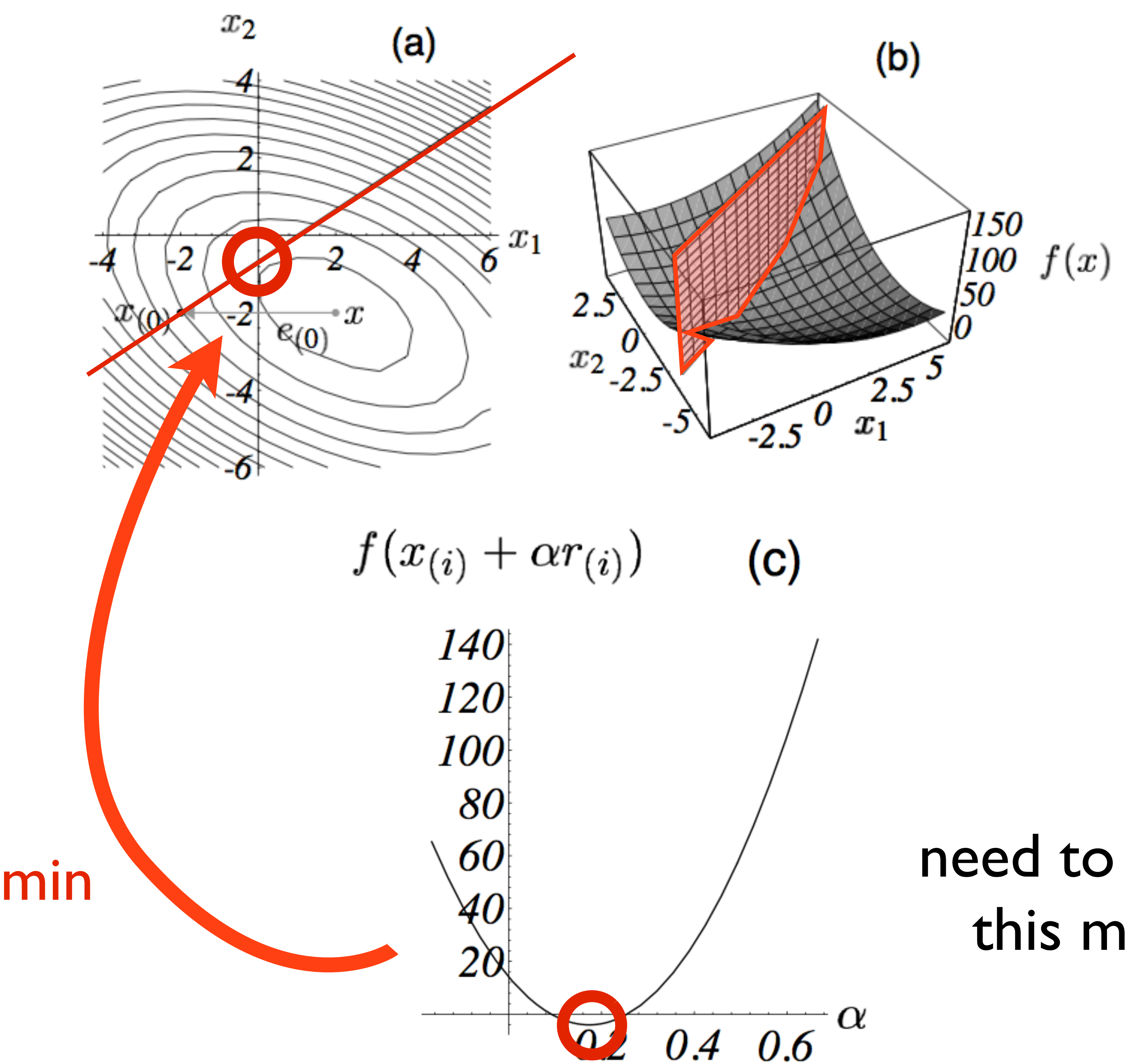
$$-\nabla f(X) = -Ax + b$$

(opposite direction of gradient)

$$-\ \nabla f(X) = -Ax + b$$

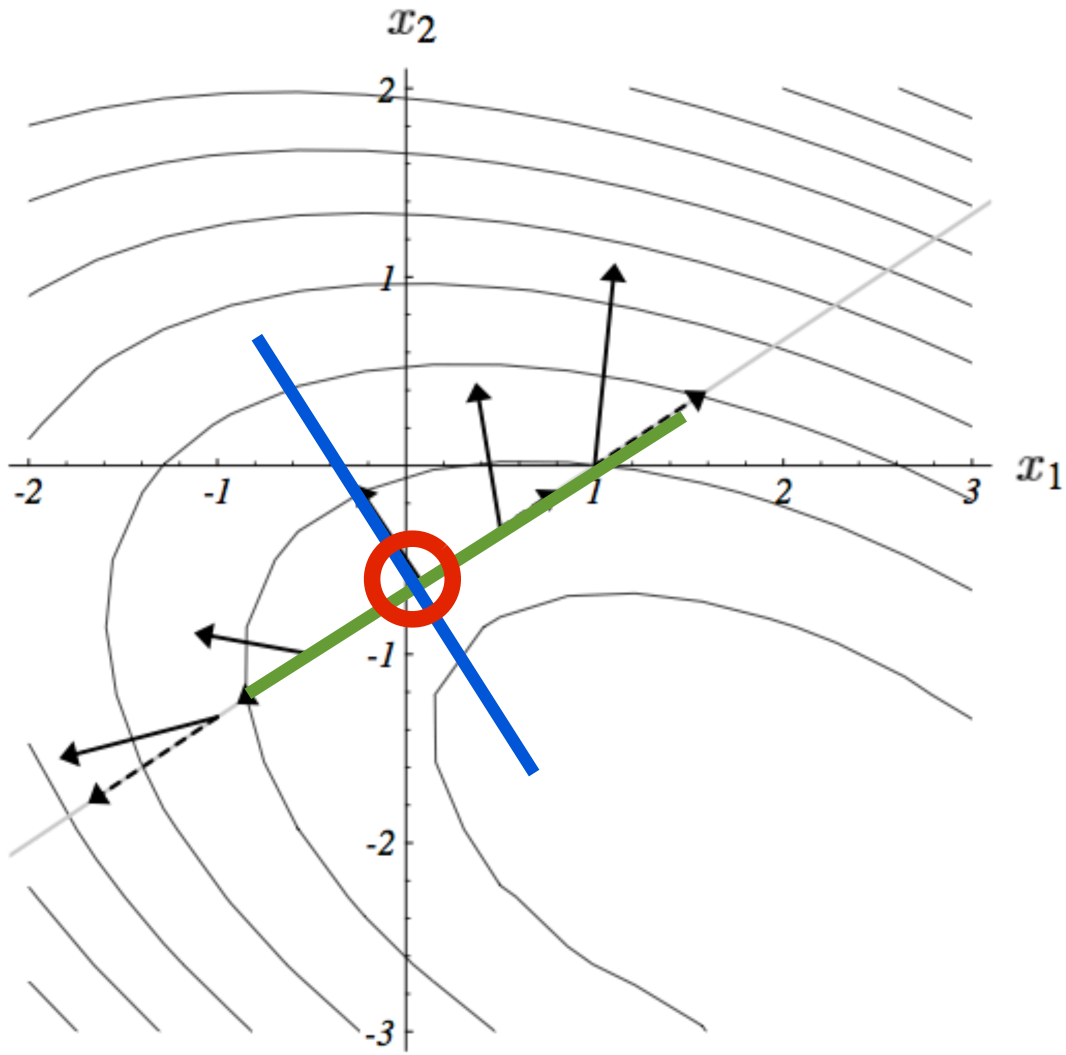


intersection of plane and paraboloid

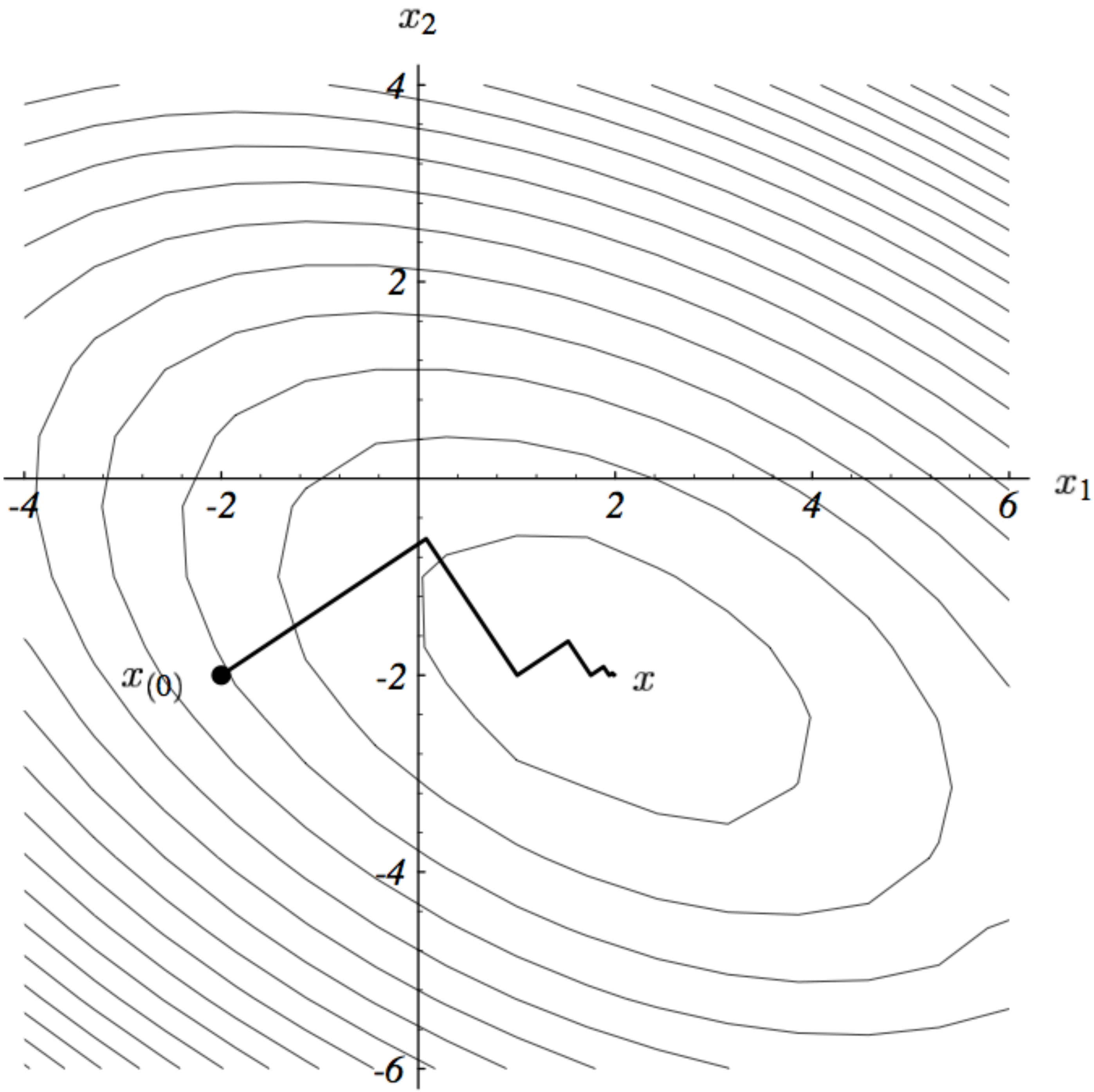


need to find
this min

our min point is also exactly where search line is orthogonal to gradient



repeat, we see classic **zigzag** behaviour

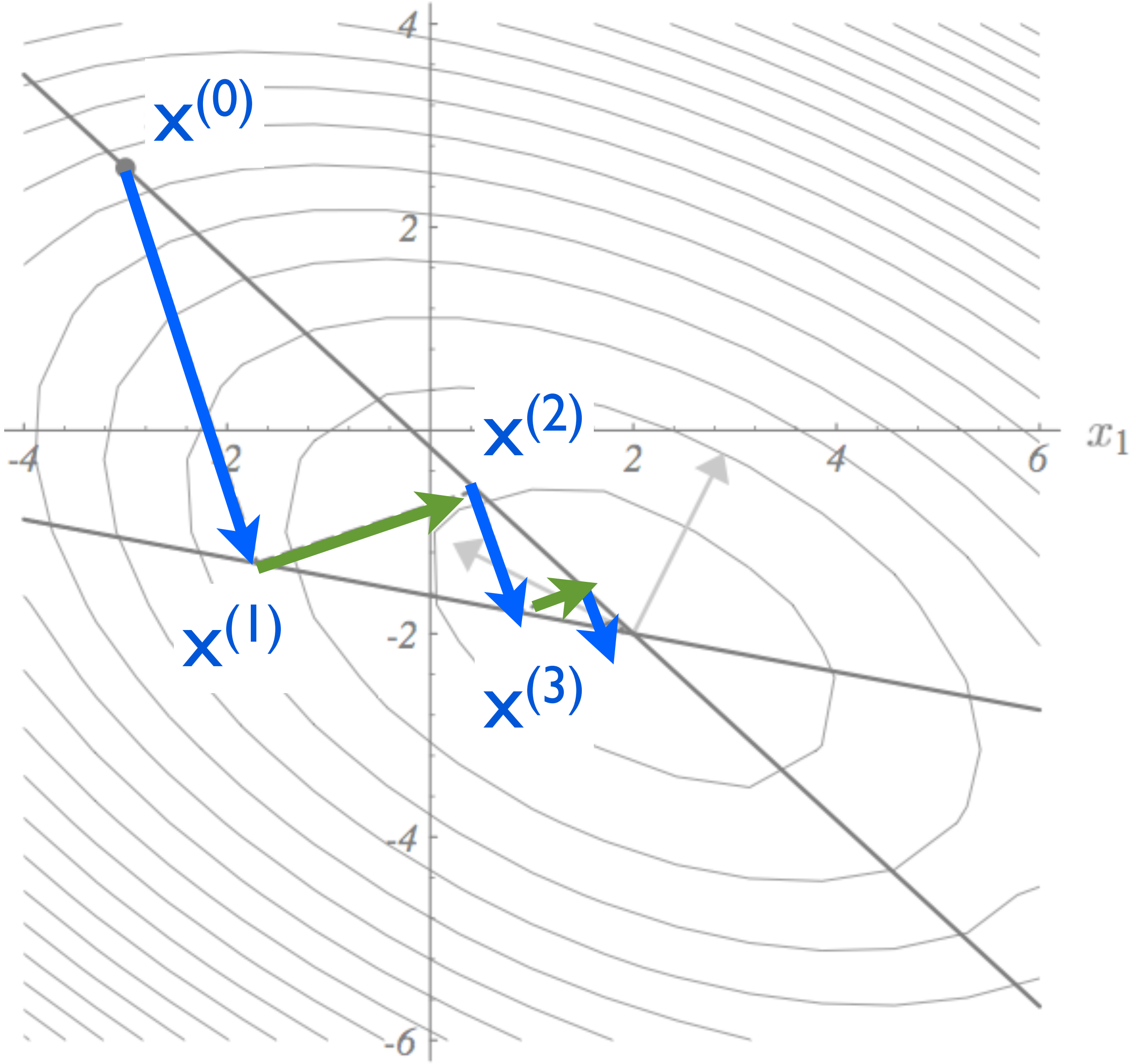


Part 5

conjugate direction and gradient methods

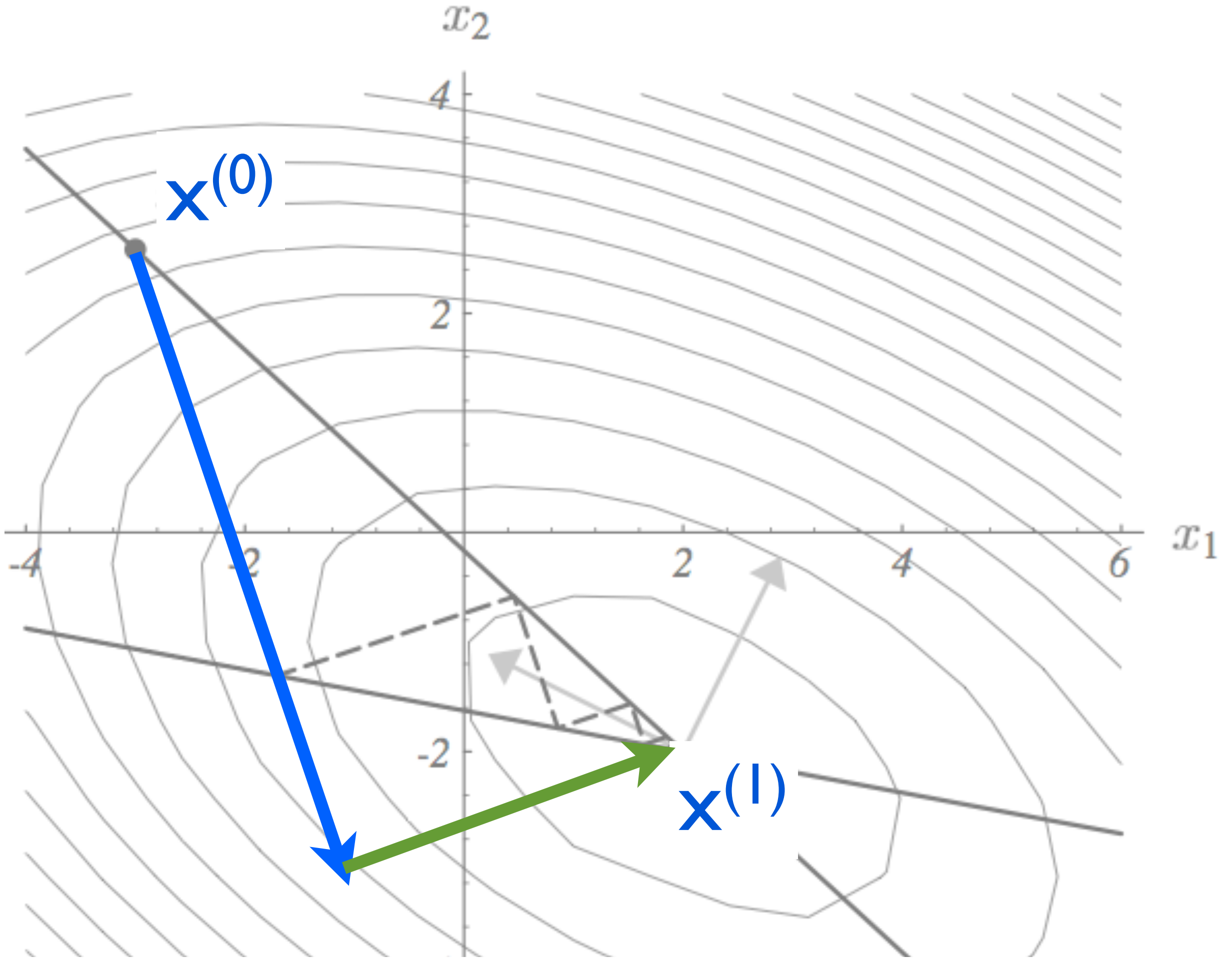
observation: moving in the **same direction** many times

(in example: just two directions)



note - zigzag directions are **orthogonal**

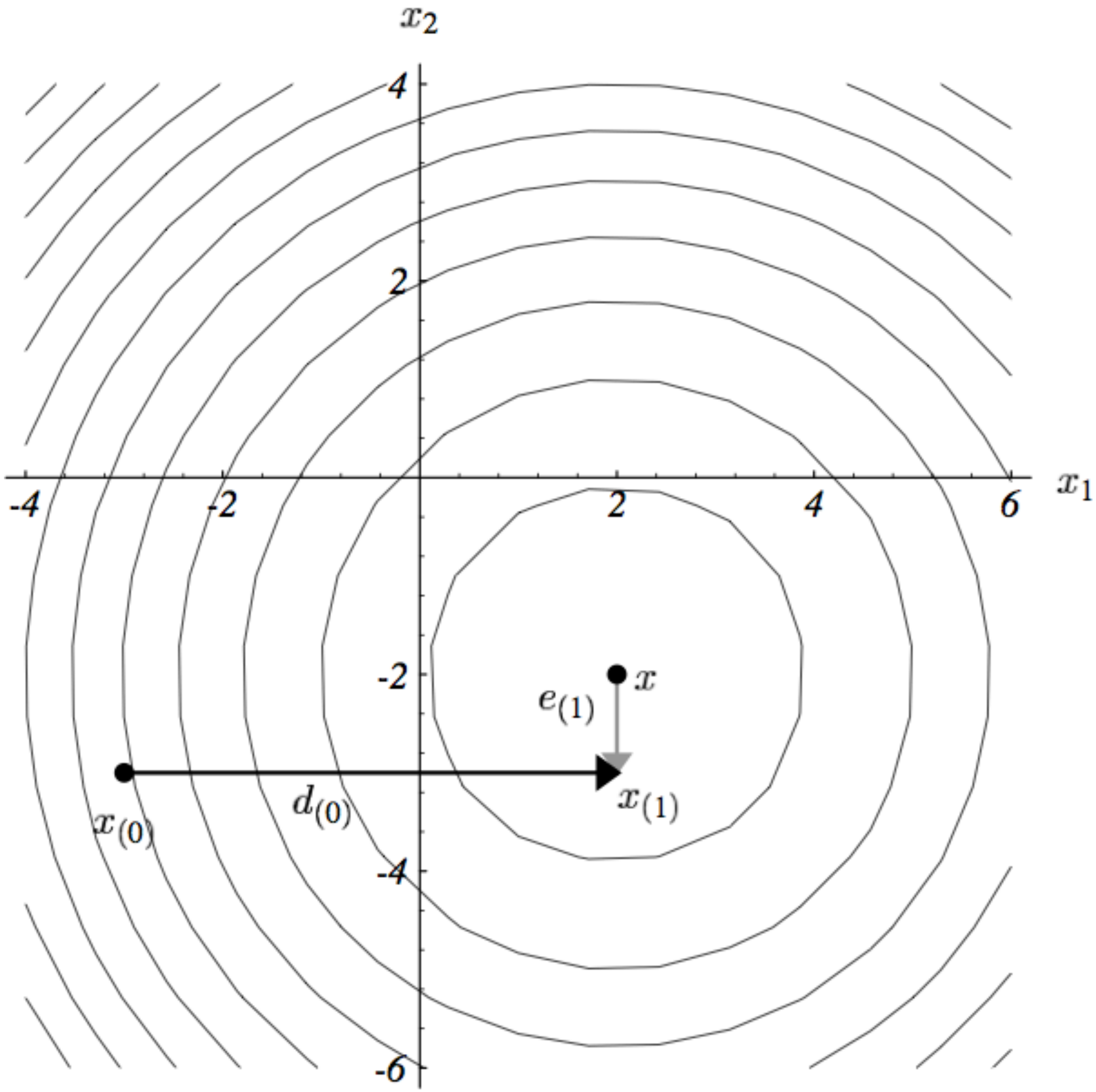
conjugate methods: combine all these zigzags into one step per "direction"



idea: pick orthogonal search directions $d^{(0)},\ldots,d^{(n-1)}$, for each $d^{(i)}$ choose step size $\alpha^{(i)}$ then we'll find x^*

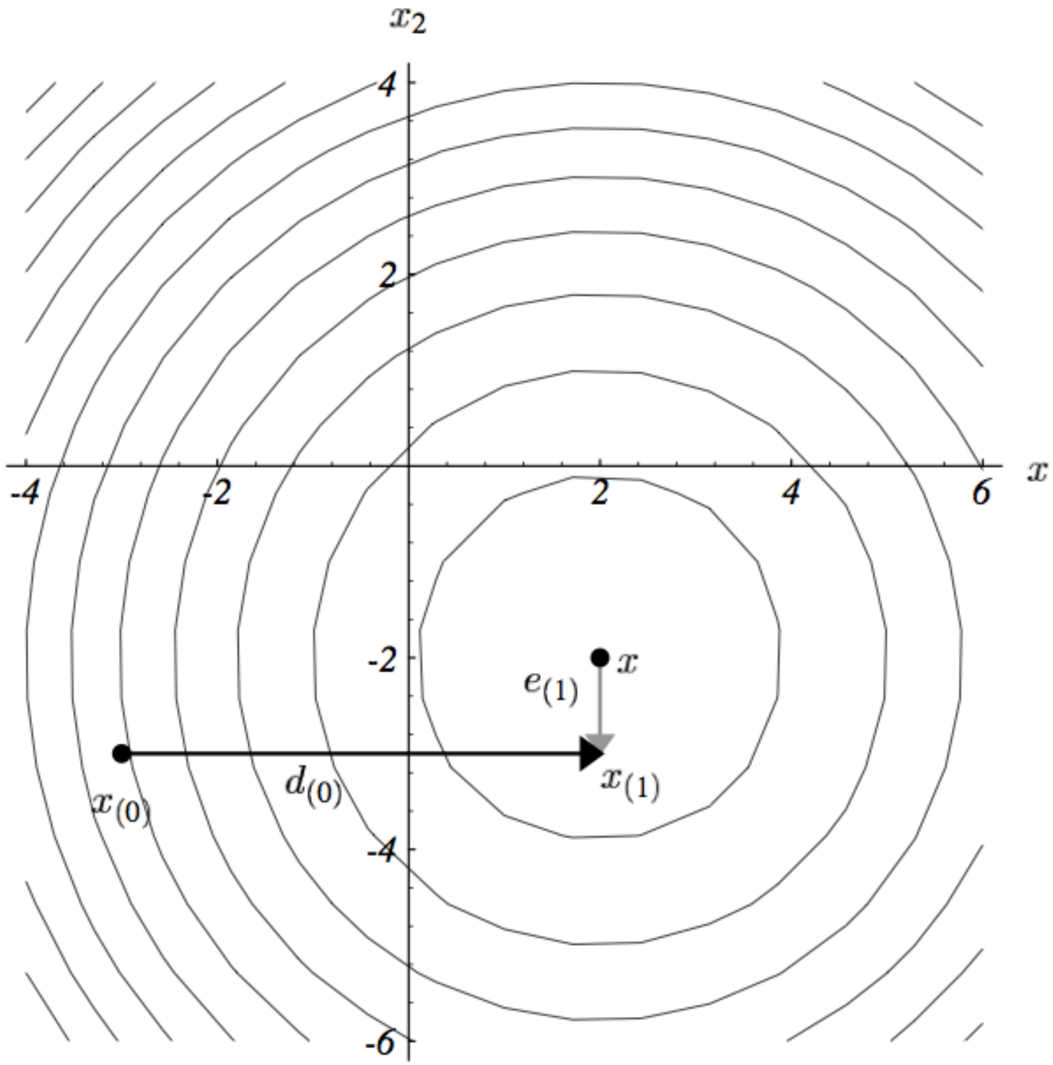
$$x^{(i+1)} = x^{(i)} + \alpha^{(i)}\, d^{(i)}$$

idea: try coordinate axes as search direction



$$d_{(i)}^T e_{(i+1)} \quad = \quad ?$$

(vectors are ...)

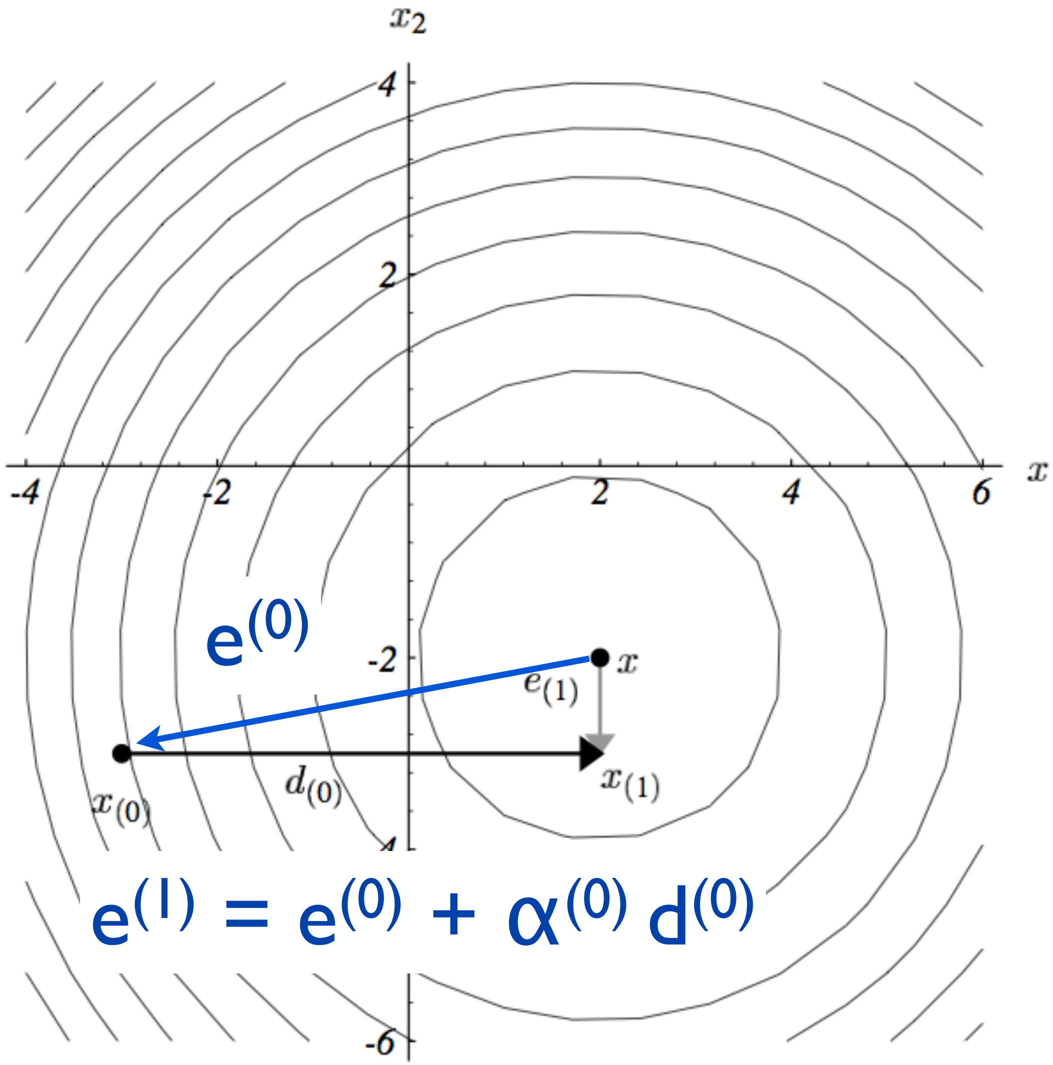


$$d_{(i)}^T e_{(i+1)} \quad = \quad 0$$

(vectors are orthogonal)

$$e^{(1)} = x^{(1)} - x^*$$

$$x^{(i+1)} = x^{(i)} + \alpha^{(i)} d^{(i)}$$

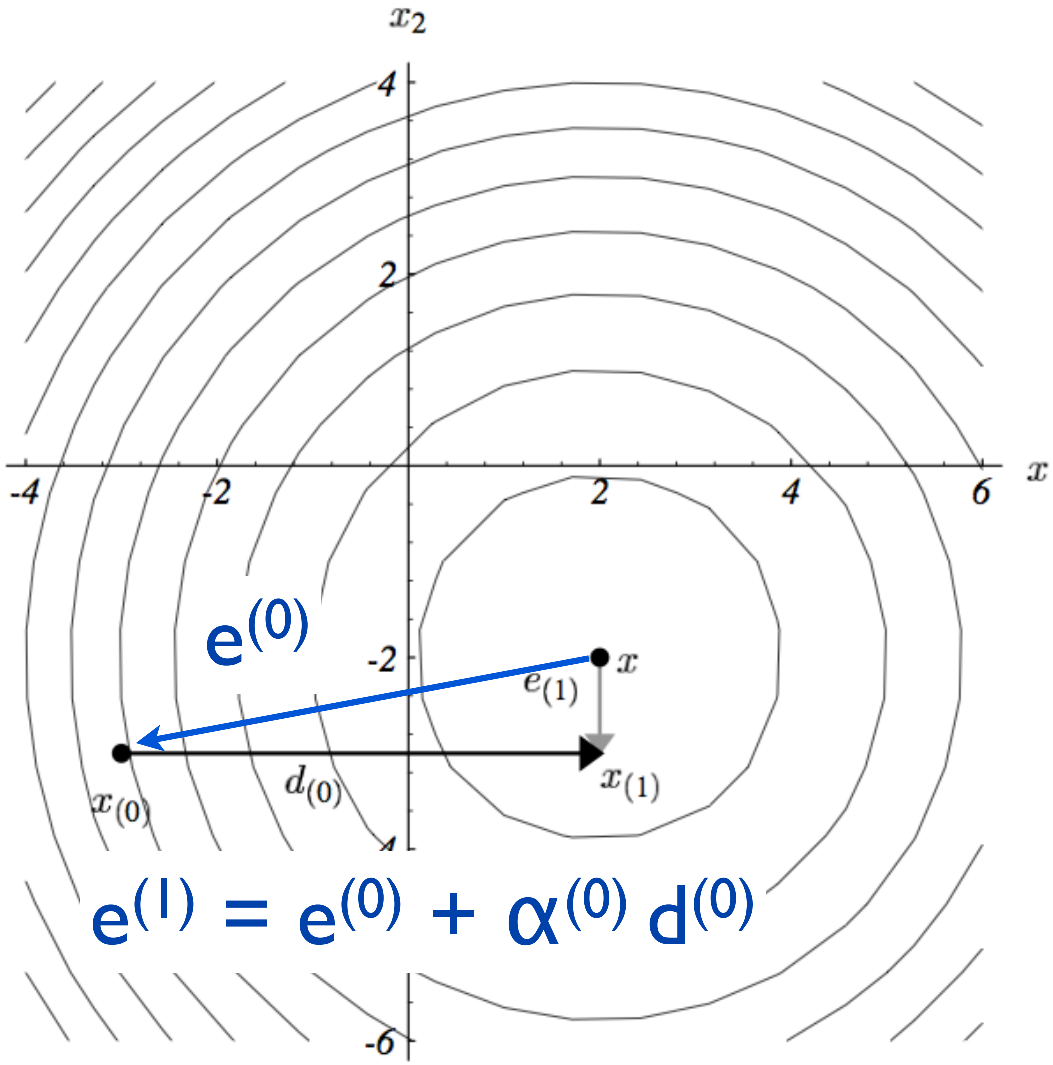


$$e^{(1)} = e^{(0)} + \alpha^{(0)} d^{(0)}$$

$$
\begin{aligned}
d_{(i)}^T e_{(i+1)} &= 0 \\
d_{(i)}^T \left(e_{(i)} + \alpha_{(i)} d_{(i)}\right) &= 0 \qquad \text{(by Equation 29)}
\end{aligned}
$$

$$e^{(1)} = x^{(1)} - x^*$$

$$x^{(i+1)} = x^{(i)} + \alpha^{(i)} d^{(i)}$$



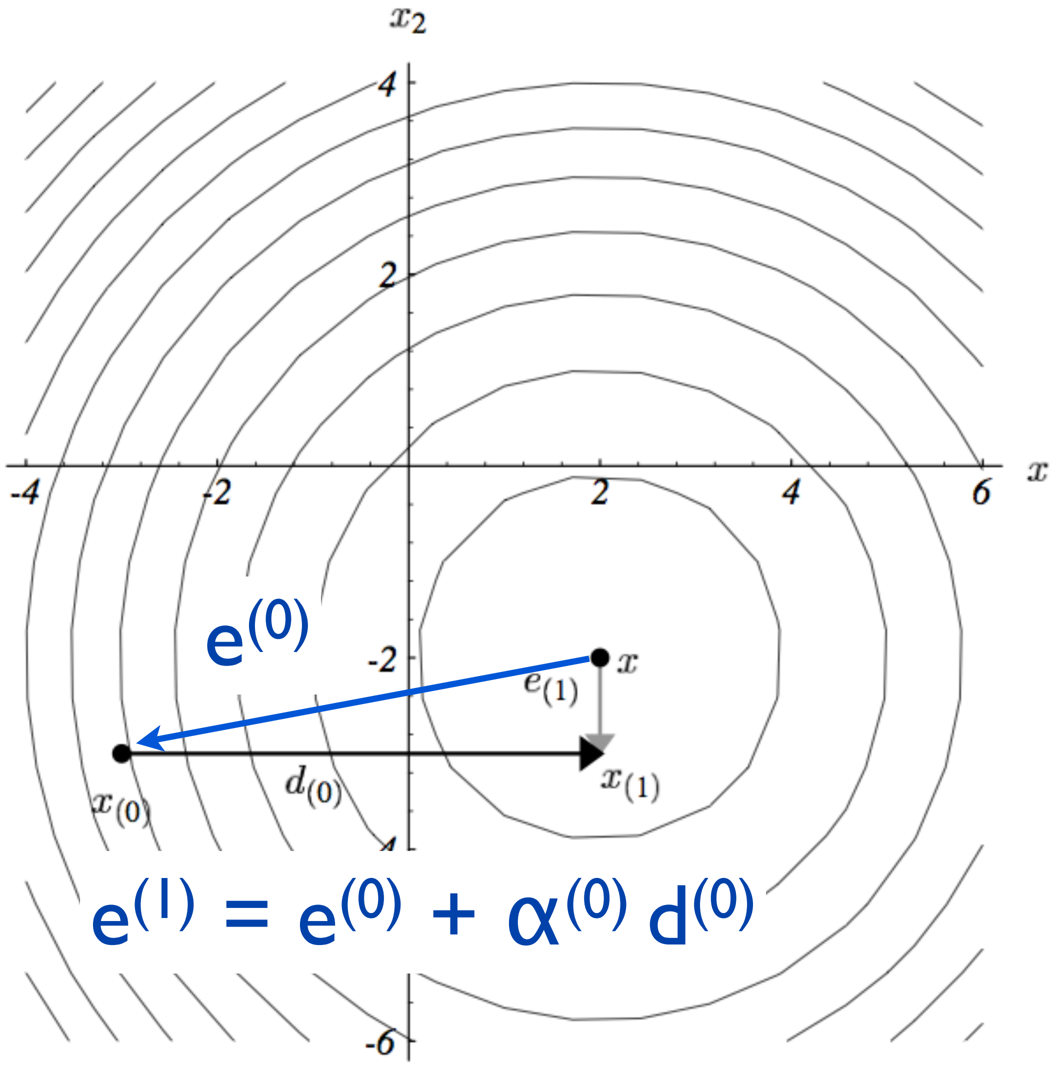
$$e^{(1)} = e^{(0)} + \alpha^{(0)} d^{(0)}$$

$$
\begin{aligned}
d_{(i)}^T e_{(i+1)} &= 0 \\
d_{(i)}^T (e_{(i)} + \alpha_{(i)} d_{(i)}) &= 0 \qquad \text{(by Equation 29)} \\
\alpha_{(i)} &= -\frac{d_{(i)}^T e_{(i)}}{d_{(i)}^T d_{(i)}}.
\end{aligned}
$$

$e^{(1)} = x^{(1)} - x^*$

$x^{(i+1)} = x^{(i)} + \alpha^{(i)}\, d^{(i)}$

can we calculate $\alpha_{(i)}$?



$e^{(1)} = e^{(0)} + \alpha^{(0)}\, d^{(0)}$

$$
\begin{aligned}
d_{(i)}^T e_{(i+1)} &= 0 \\
d_{(i)}^T \left(e_{(i)} + \alpha_{(i)} d_{(i)}\right) &= 0 \qquad \text{(by Equation 29)} \\
\alpha_{(i)} &= -\frac{d_{(i)}^T e_{(i)}}{d_{(i)}^T d_{(i)}}.
\end{aligned}
$$

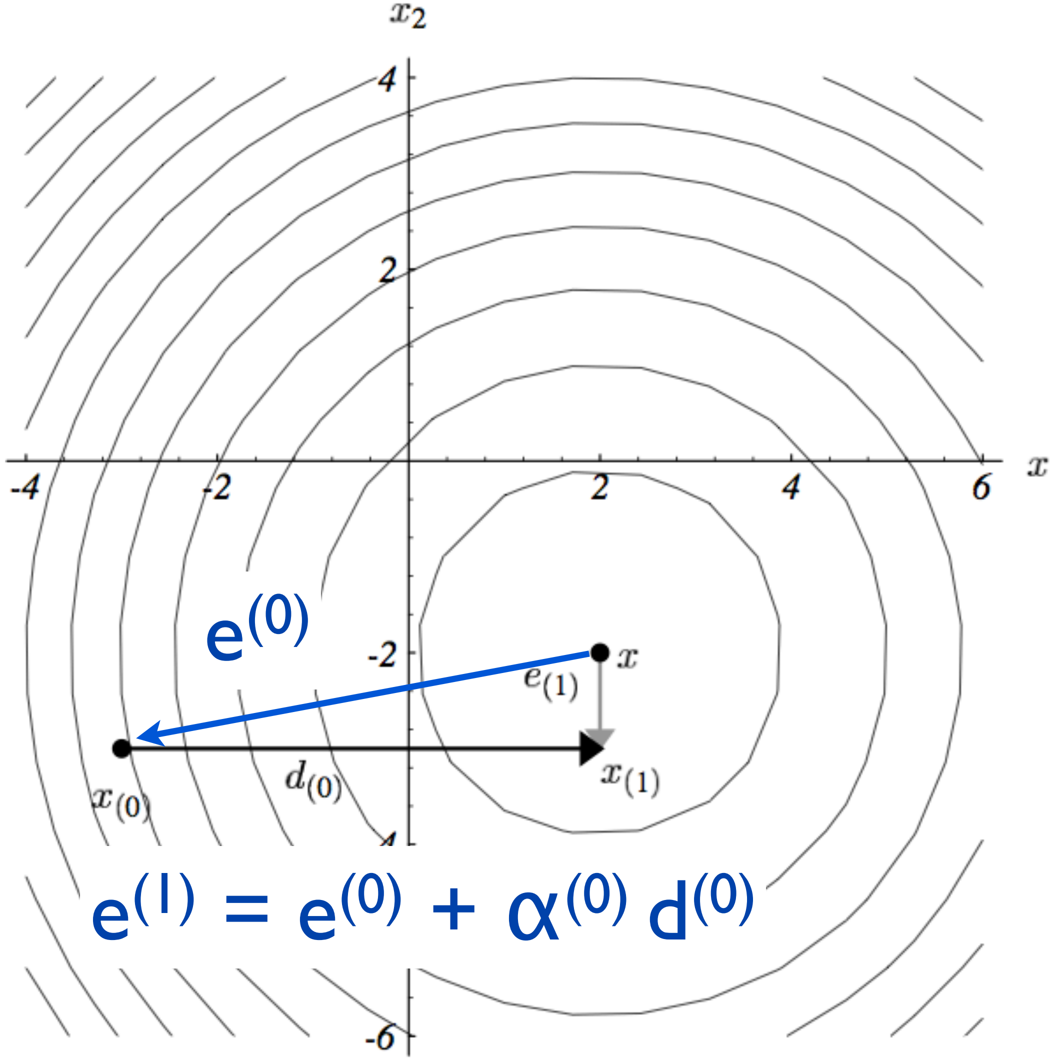
$$e^{(1)} = x^{(1)} - x^*$$

$$x^{(i+1)} = x^{(i)} + \alpha^{(i)} d^{(i)}$$

...to find $\alpha_{(i)}$ we need $e_{(i)}$

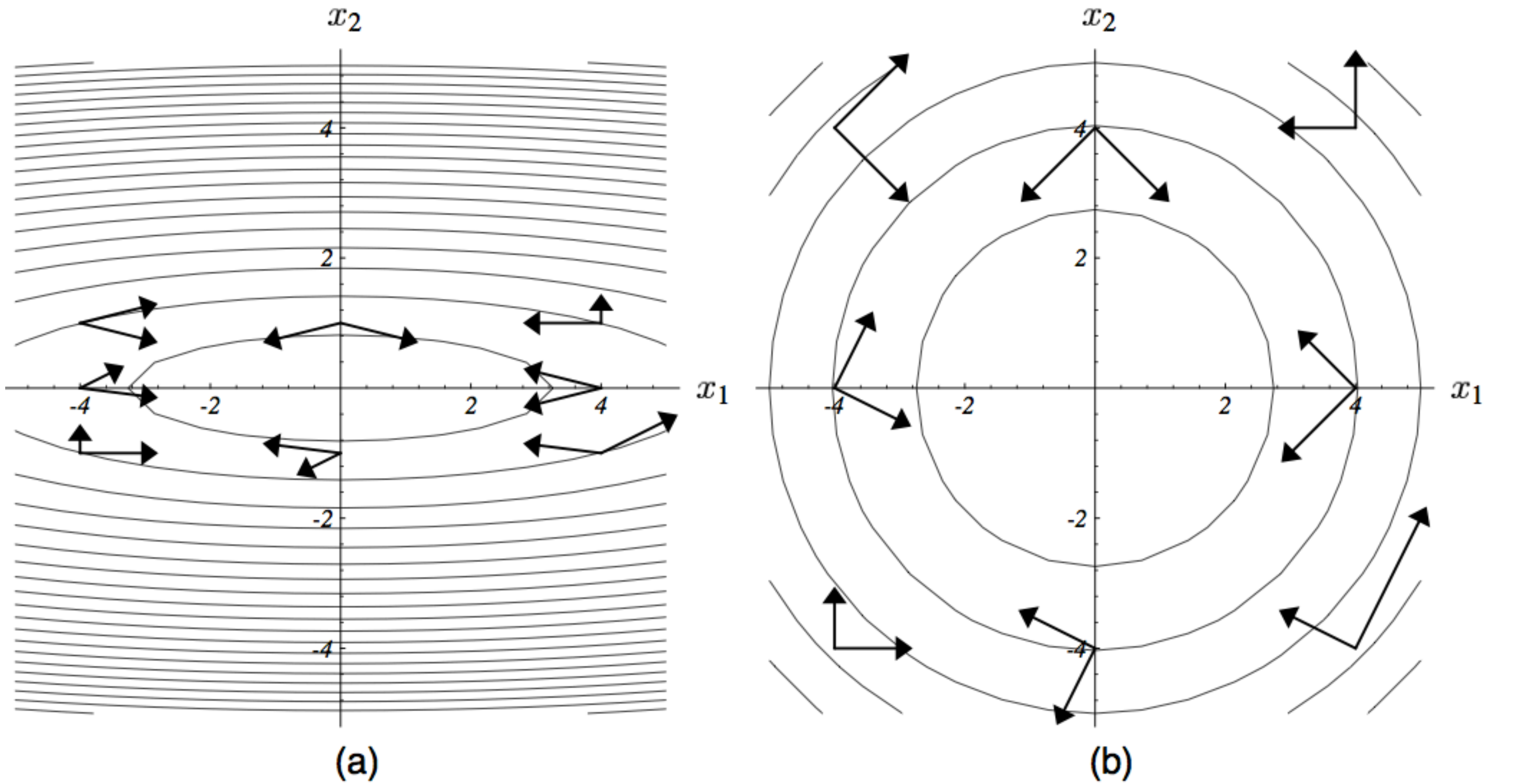
...to find $e_{(i)}$ we need x^*

...but x^* is the solution!



$$e^{(1)} = e^{(0)} + \alpha^{(0)} d^{(0)}$$

instead, make directions **A-orthogonal** ("conjugate")

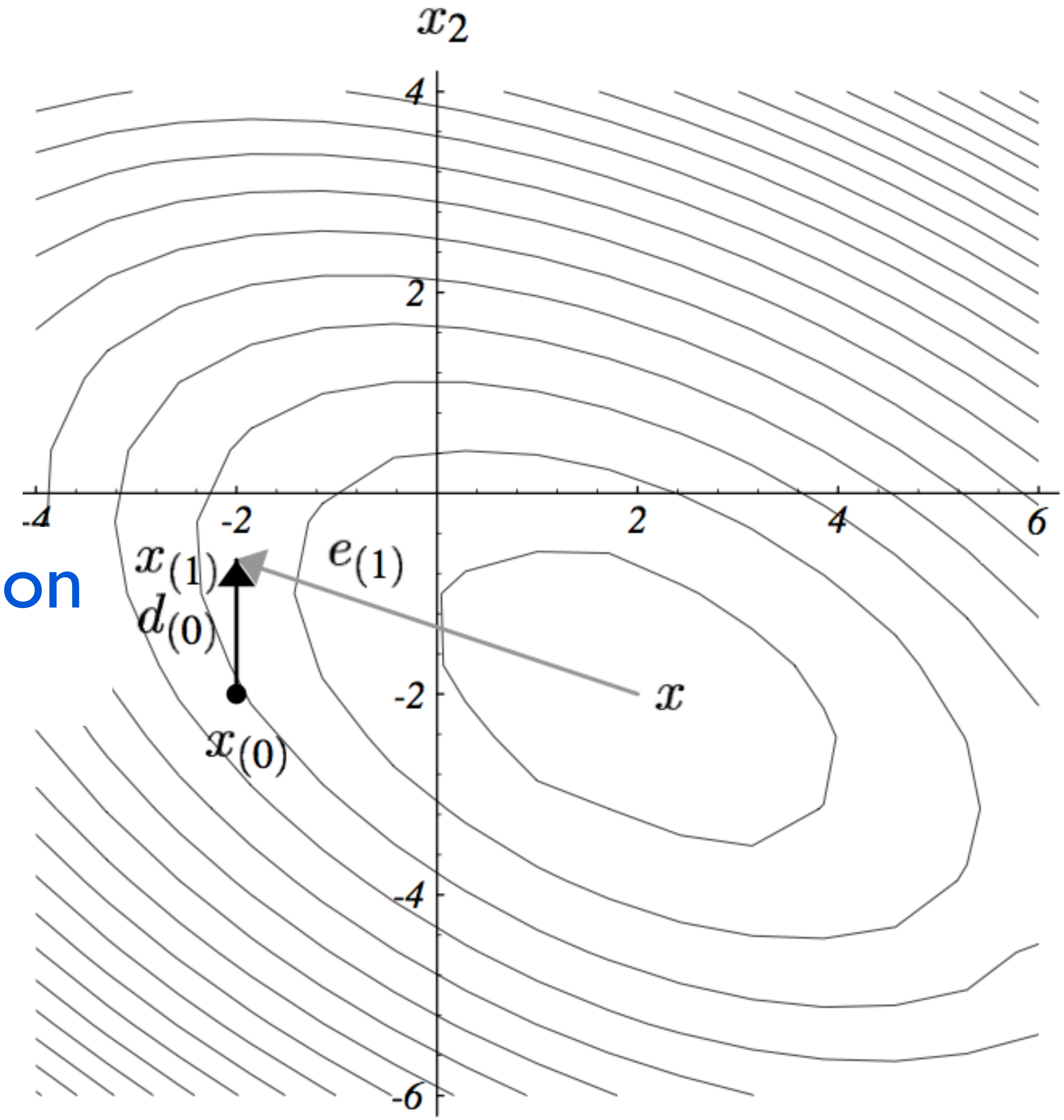


pairs of vectors are A-orthogonal.... because these are orthogonal

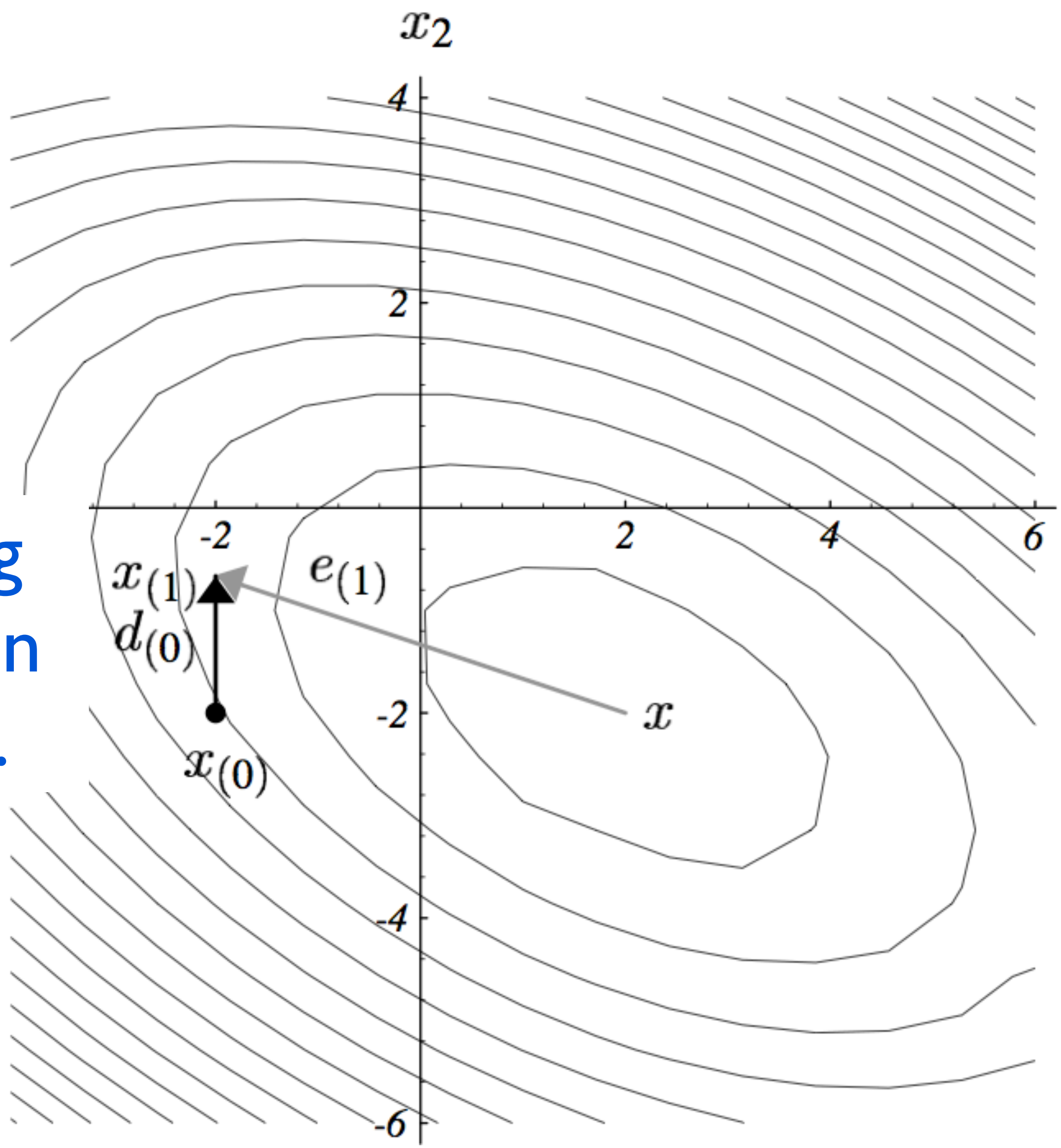
$$u^T A v = 0$$

instead, make directions **A-orthogonal** ("conjugate")

pick direction $d_{(0)}$

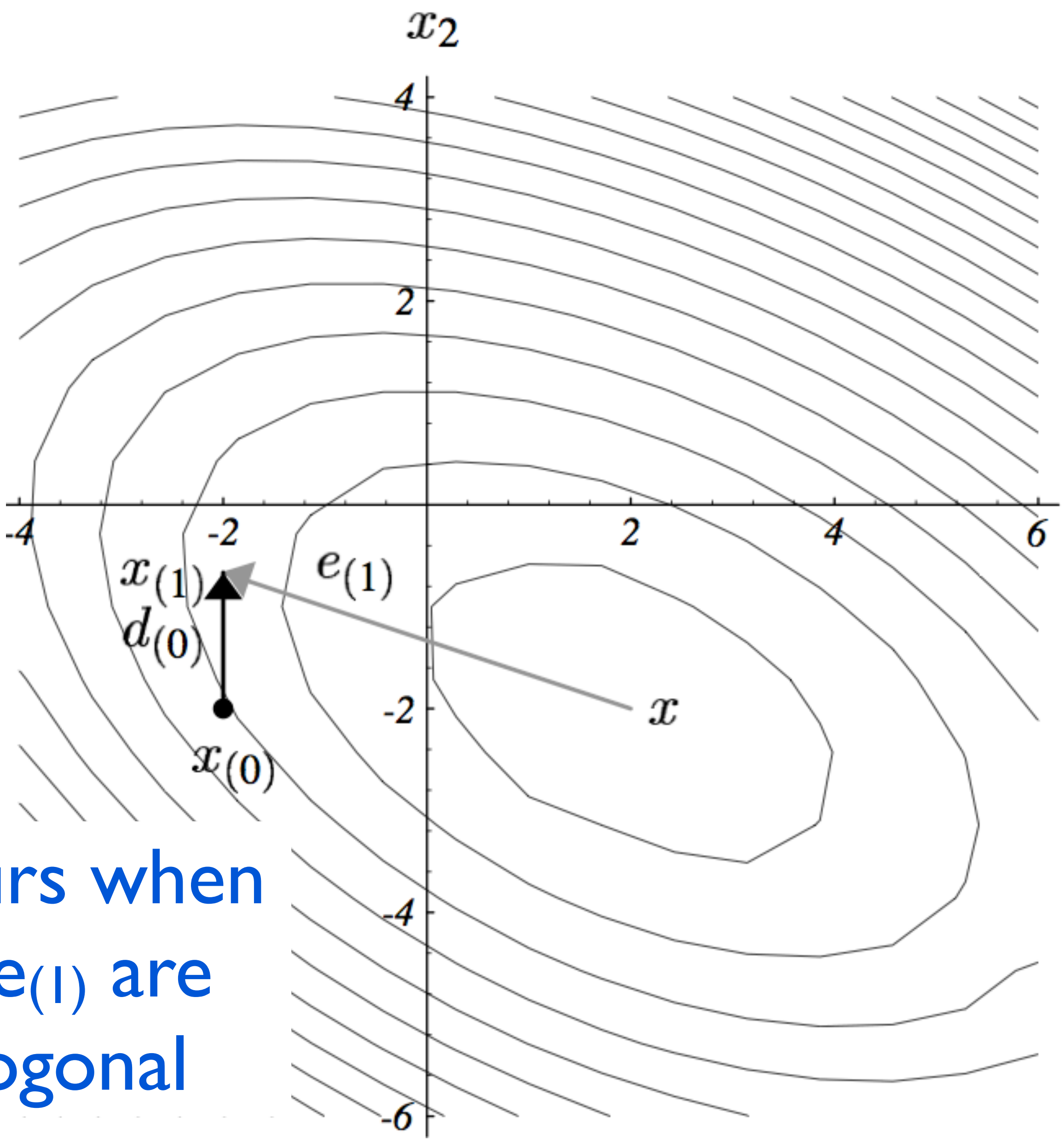


instead, make directions **A-orthogonal** ("conjugate")



move along $d_{(0)}$ until min point $x_{(1)}$...

instead, make directions **A-orthogonal** ("conjugate")



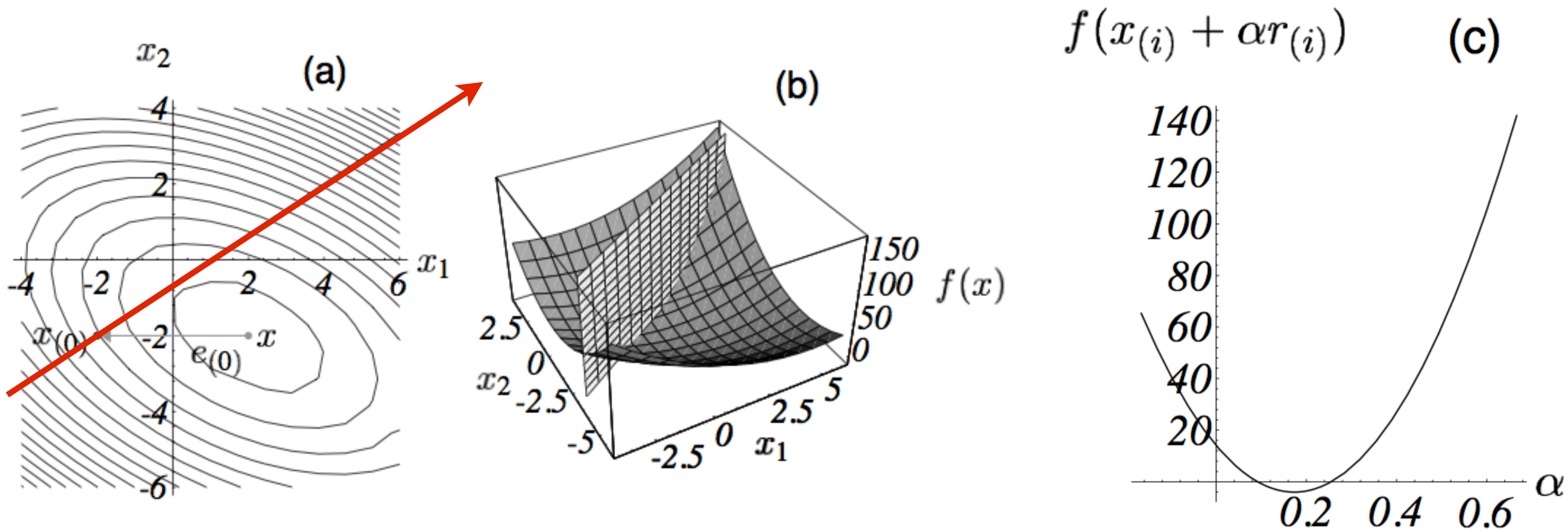
..min occurs when $d_{(0)}$ and $e_{(1)}$ are A-orthogonal

instead, make directions **A-orthogonal** ("conjugate")

$$d_{(i)}^T A d_{(j)} = 0$$

new requirement: $e_{(i+1)}$ is A-orthogonal to $d_{(i)}$

same as finding min point along search direction, like in steepest descent



before we had this:

$$\alpha_{(i)} = -\frac{d_{(i)}^T e_{(i)}}{d_{(i)}^T d_{(i)}}$$

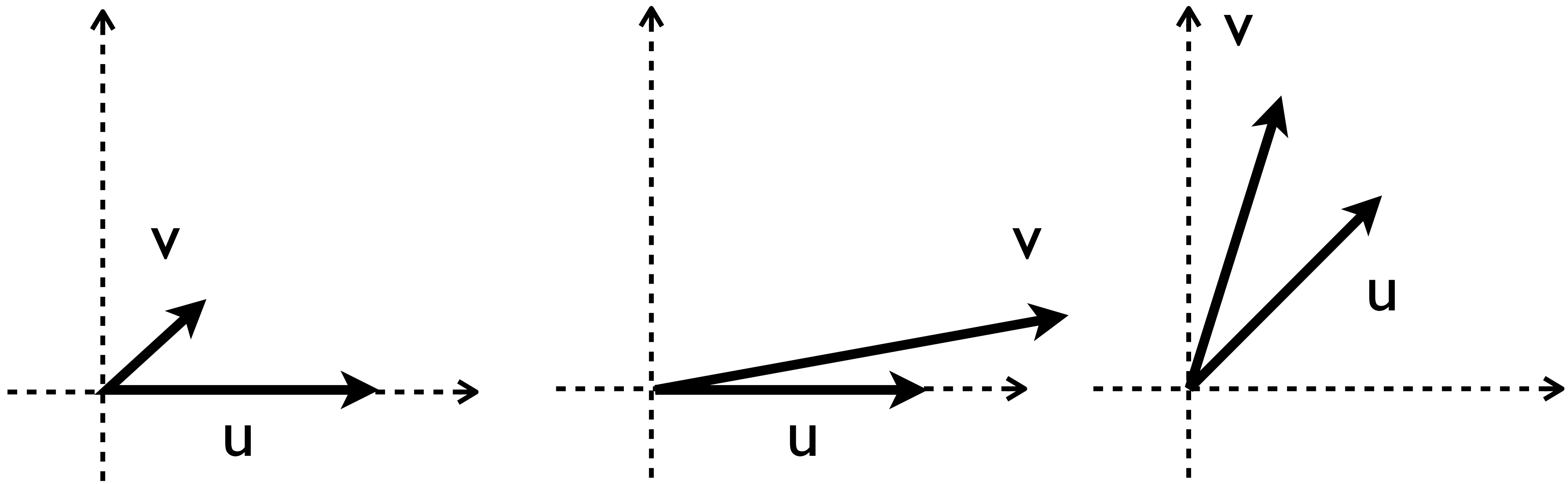
now we have this:

$$\begin{aligned}\alpha_{(i)} &= -\frac{d_{(i)}^T A e_{(i)}}{d_{(i)}^T A d_{(i)}} \\ &= \frac{d_{(i)}^T r_{(i)}}{d_{(i)}^T A d_{(i)}}.\end{aligned}$$

can we calculate $\alpha_{(i)}$?

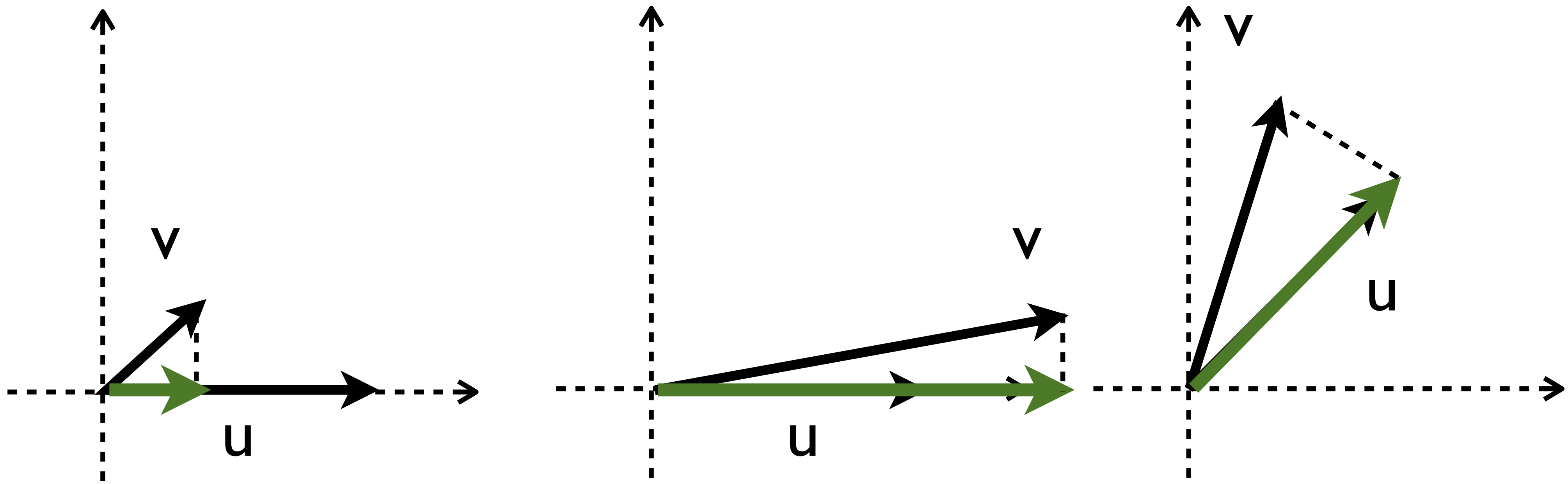
Gram-Schmidt

make linearly independent vectors orthogonal



Gram-Schmidt

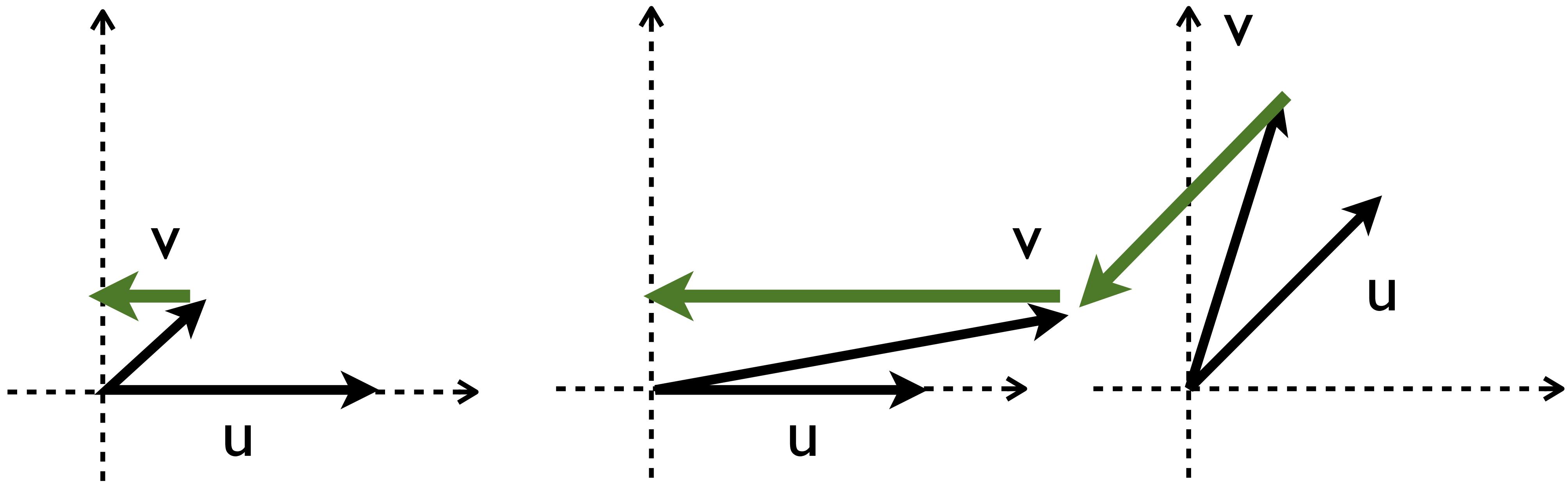
make linearly independent vectors orthogonal



step 1. project

Gram-Schmidt

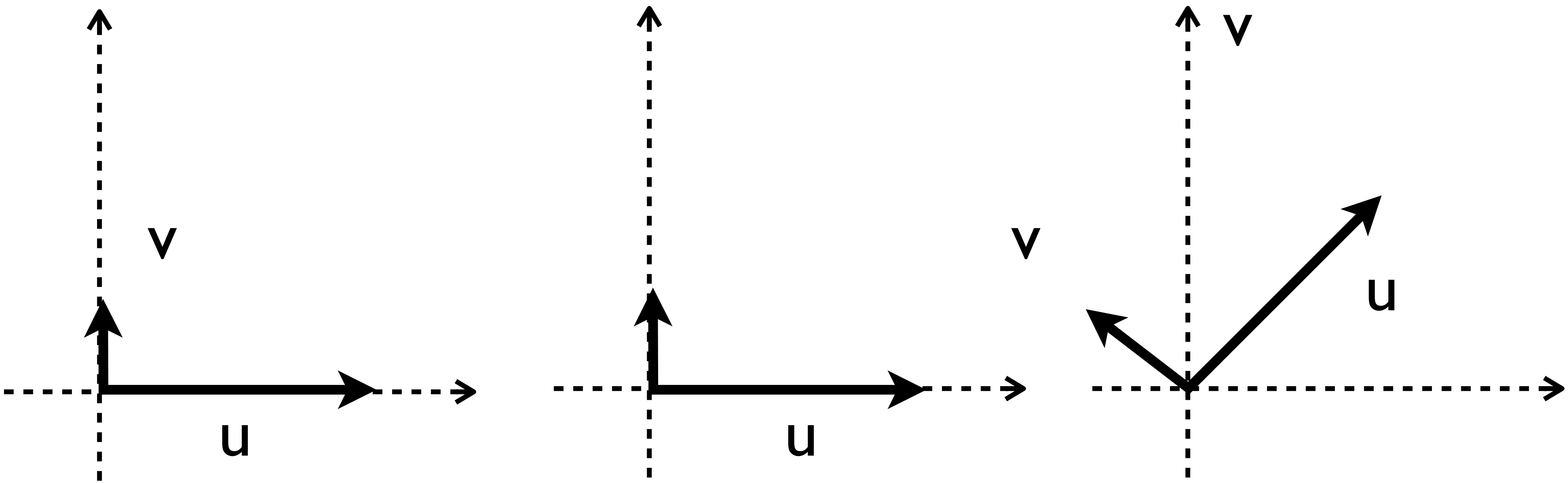
make linearly independent vectors orthogonal



step 2. subtract from v

Gram-Schmidt

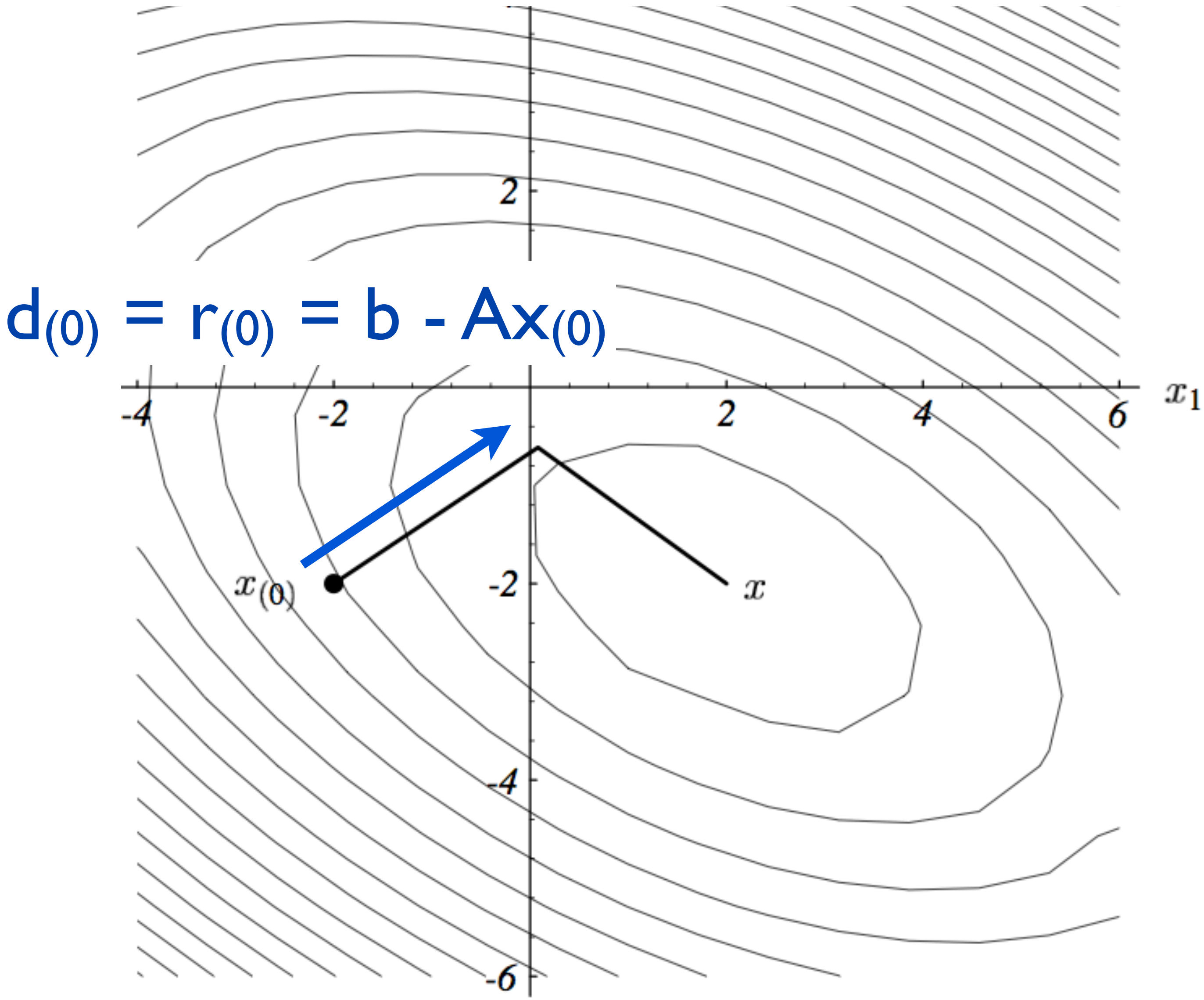
make linearly independent vectors orthogonal



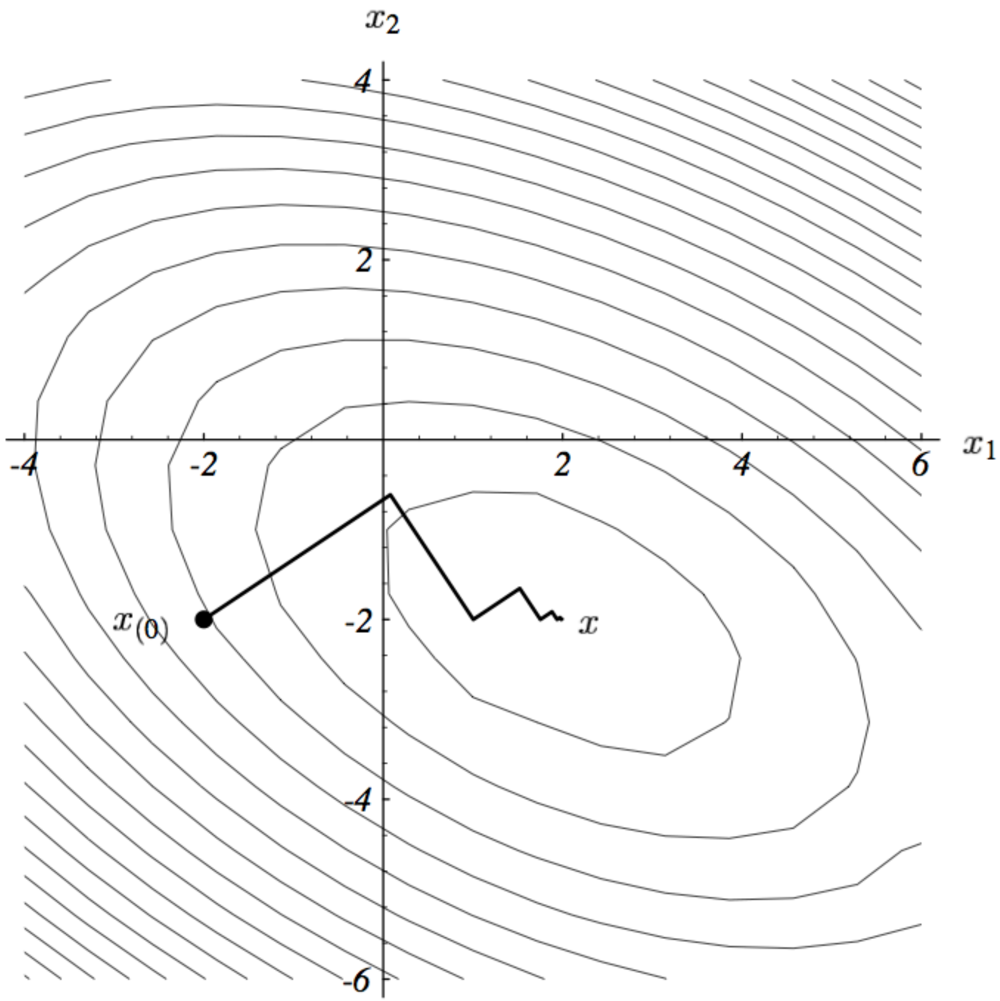
step 3. done!

conjugate gradients method

pick directions by residuals (just like steepest descent), but then conjugate!

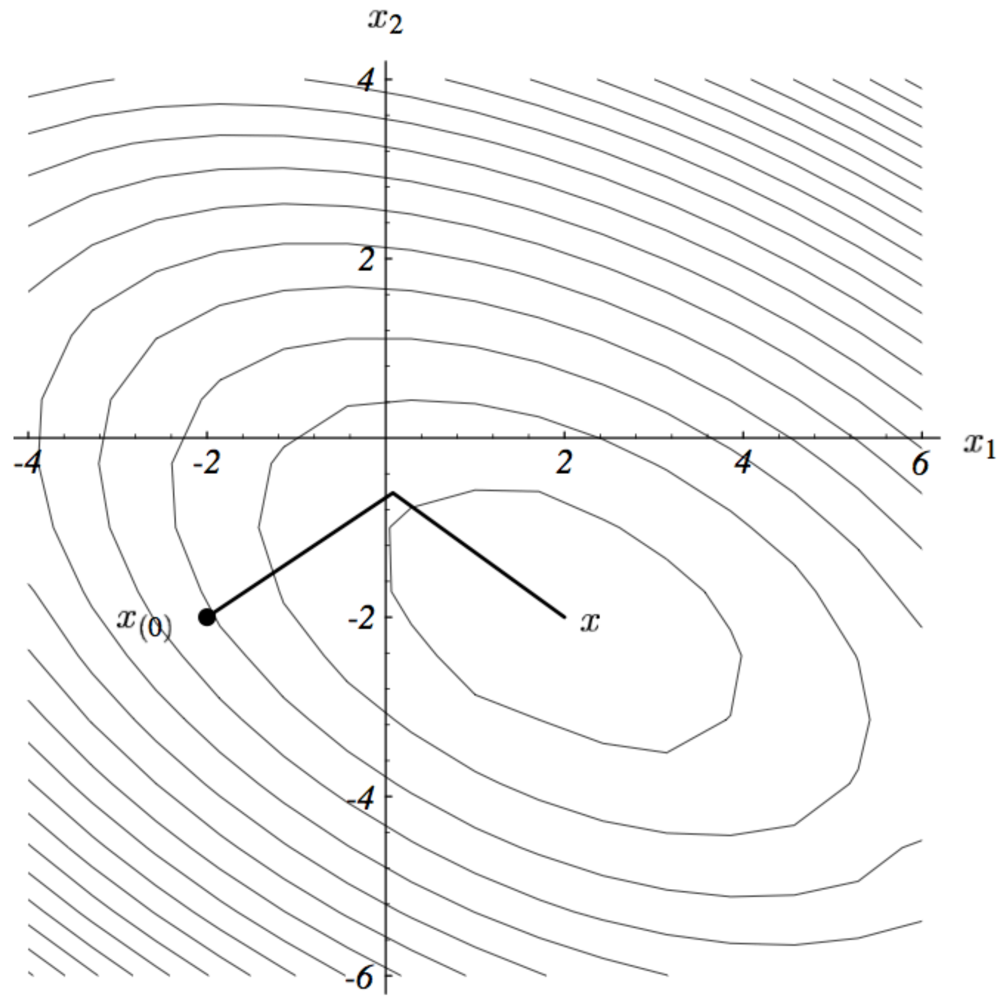


steepest descent



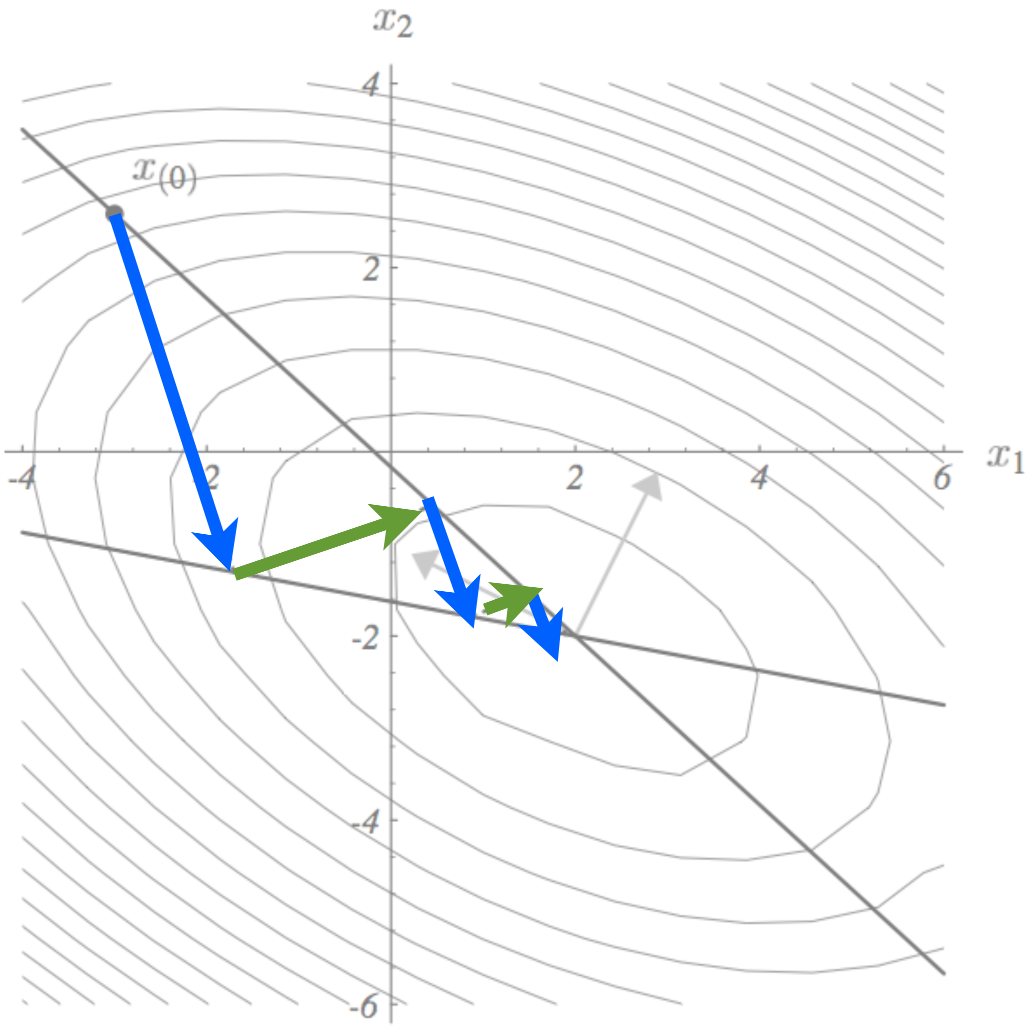
follow residuals
(zigzag)

conjugate gradients



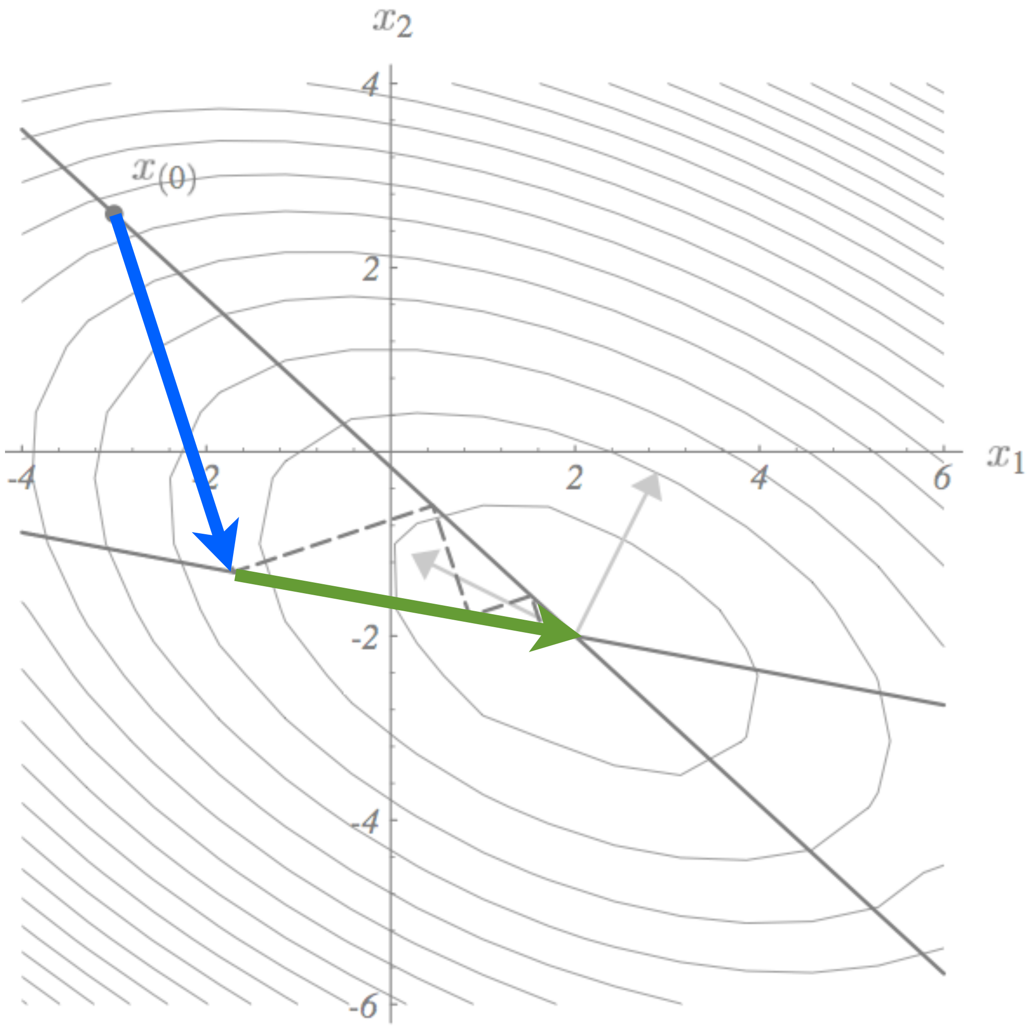
follow conjugate directions

steepest descent



always turns orthogonal

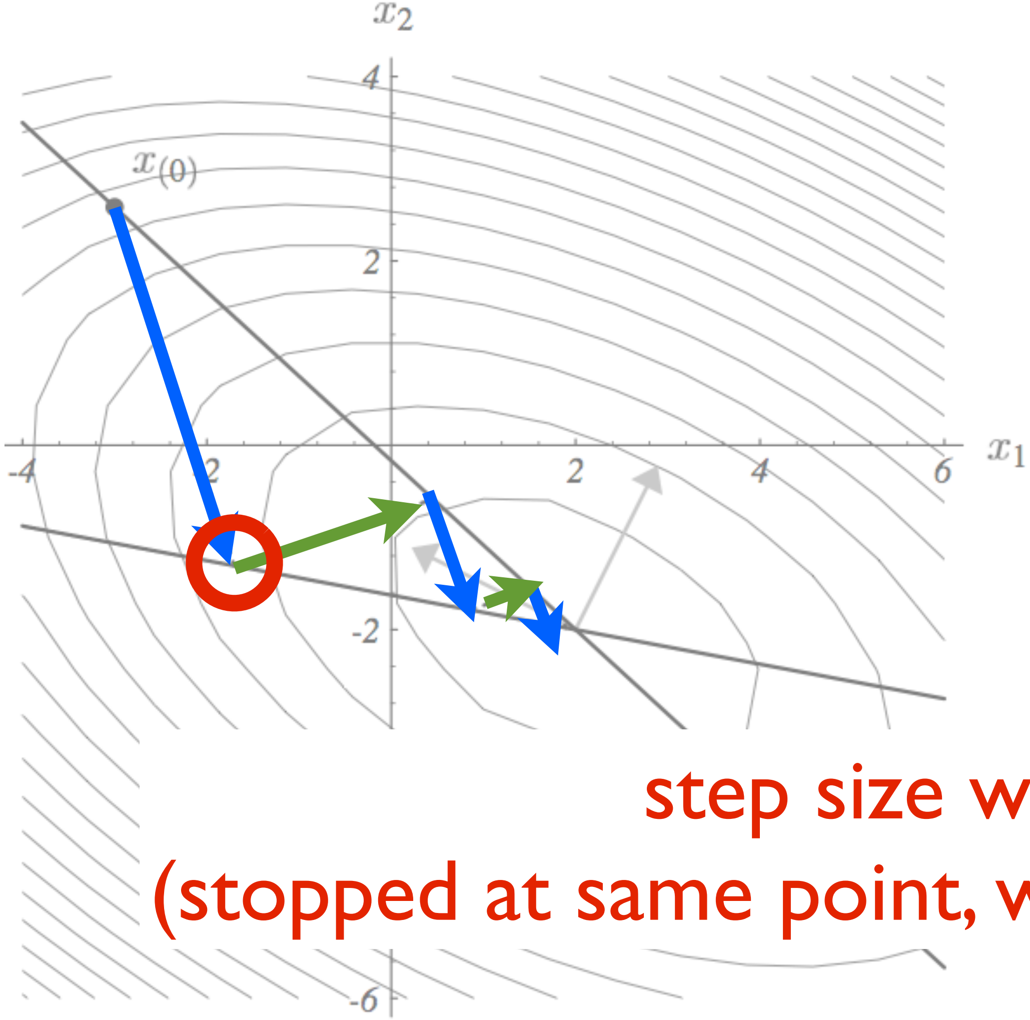
conjugate gradients

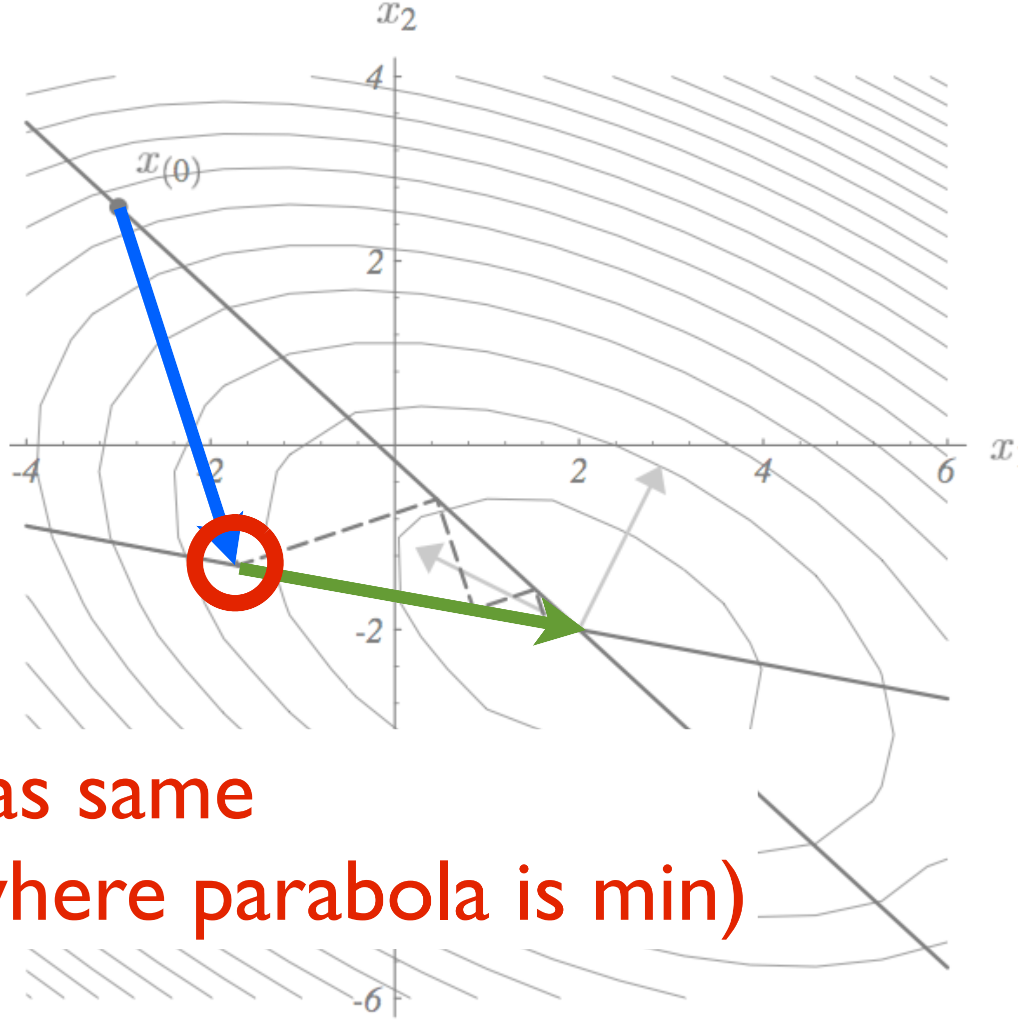


always turns A-orthogonal

steepest descent

conjugate gradients





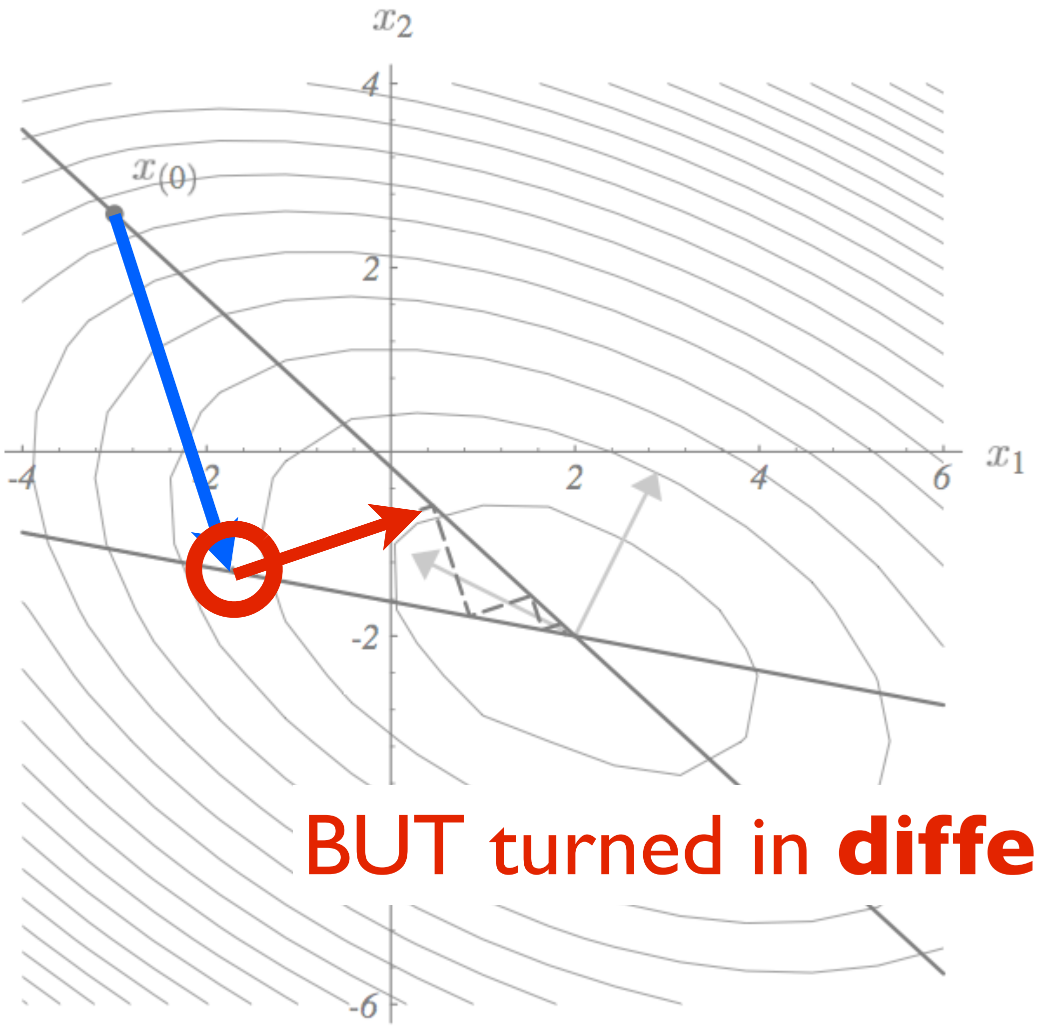
step size was same

(stopped at same point, where parabola is min)

always turns orthogonal

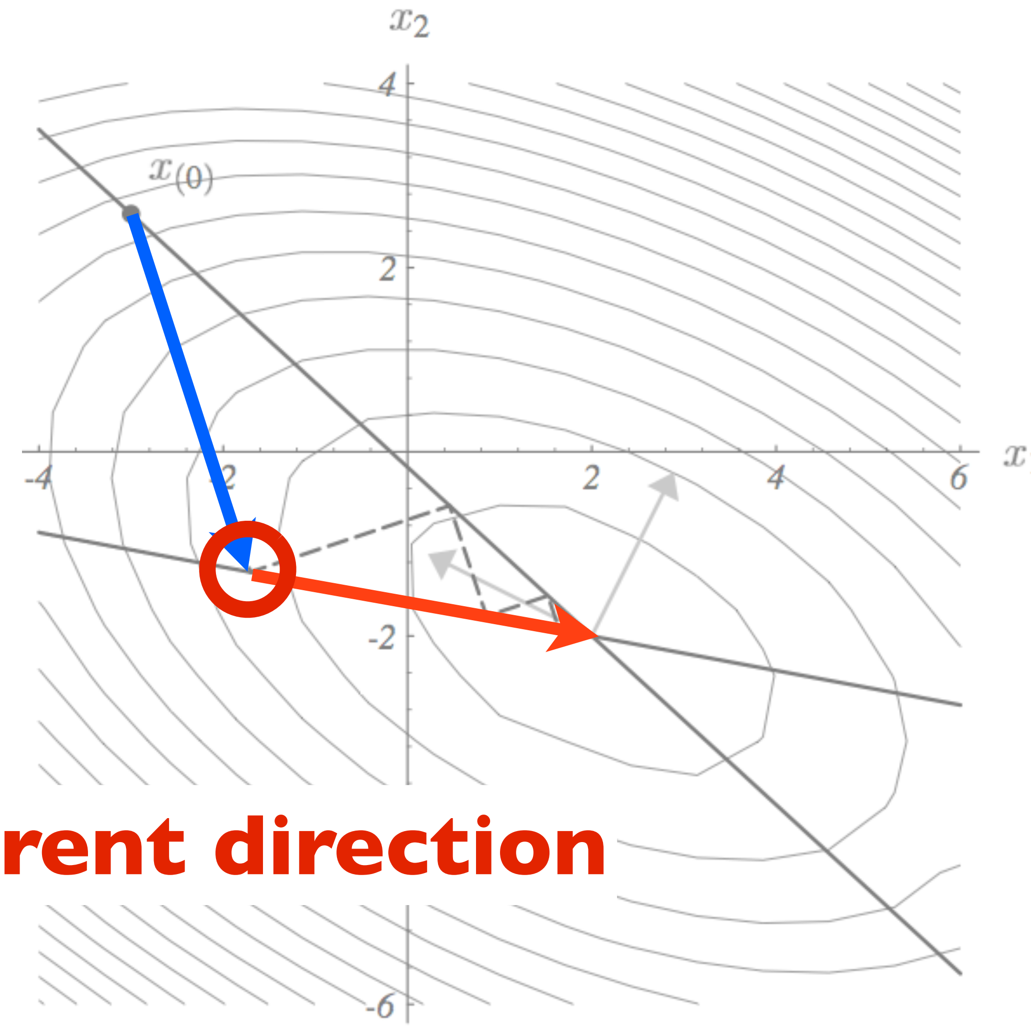
always turns A-orthogonal

steepest descent



always turns orthogonal

conjugate gradients



always turns A-orthogonal

BUT turned in **different direction**

initial direction from residual (steepest descent) $d_{(0)} = r_{(0)} = b - Ax_{(0)},$

line search $\alpha_{(i)} = \frac{r_{(i)}^T r_{(i)}}{d_{(i)}^T A d_{(i)}}$ (by Equations 32 and 42),

next x $x_{(i+1)} = x_{(i)} + \alpha_{(i)} d_{(i)},$

initial direction from residual (steepest descent) $d_{(0)} = r_{(0)} = b - Ax_{(0)}$,

line search $\alpha_{(i)} = \frac{r_{(i)}^T r_{(i)}}{d_{(i)}^T A d_{(i)}}$ (by Equations 32 and 42),

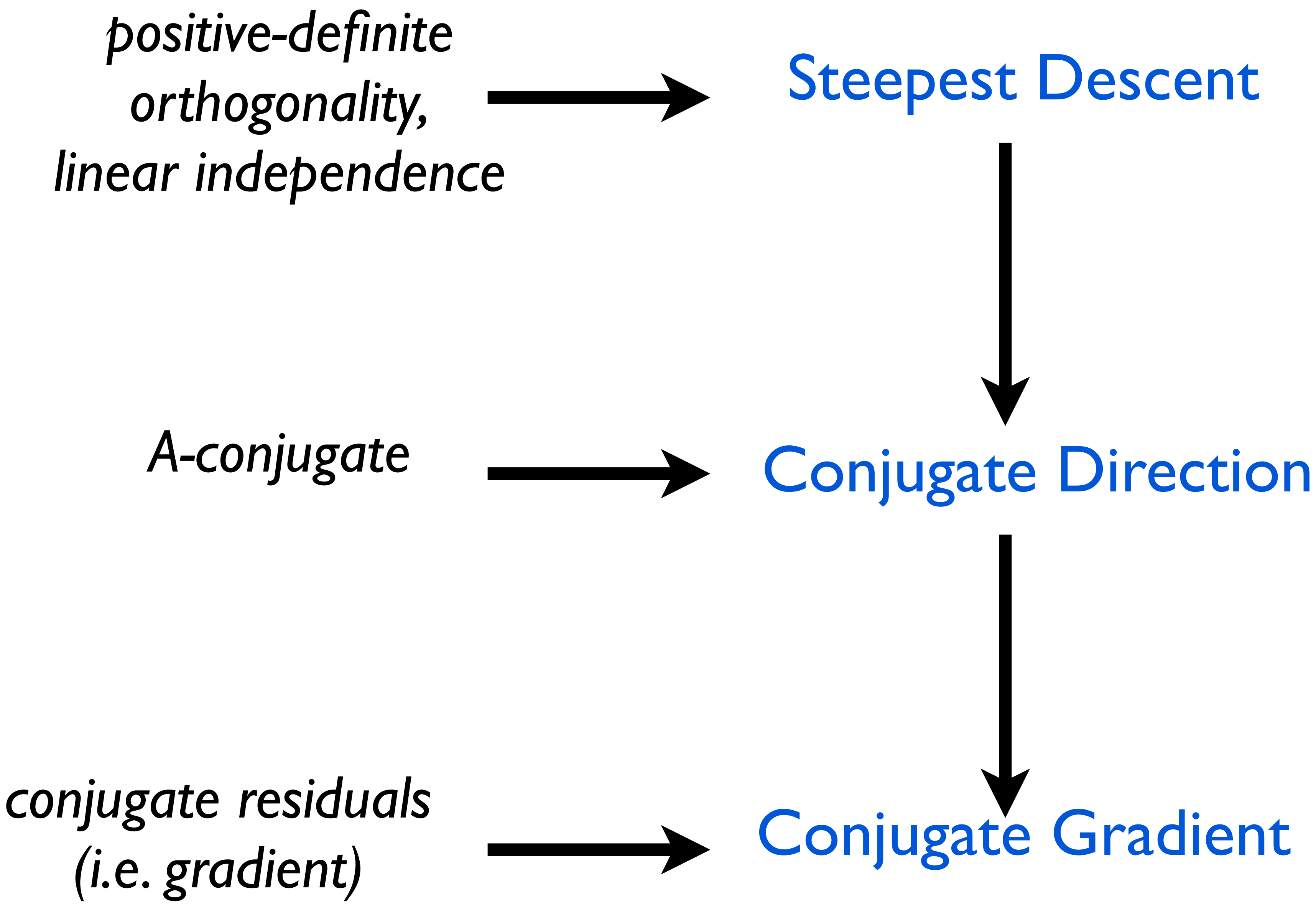
next x $x_{(i+1)} = x_{(i)} + \alpha_{(i)} d_{(i)}$,

residual update $r_{(i+1)} = r_{(i)} - \alpha_{(i)} A d_{(i)}$,

Gram-Schmidt constants (for conjugation) $\beta_{(i+1)} = \frac{r_{(i+1)}^T r_{(i+1)}}{r_{(i)}^T r_{(i)}}$,

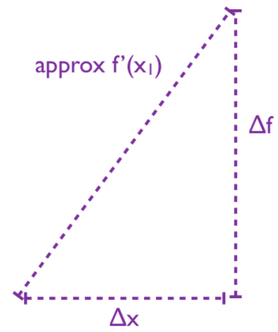
next direction (conjugation) $d_{(i+1)} = r_{(i+1)} + \beta_{(i+1)} d_{(i)}$.

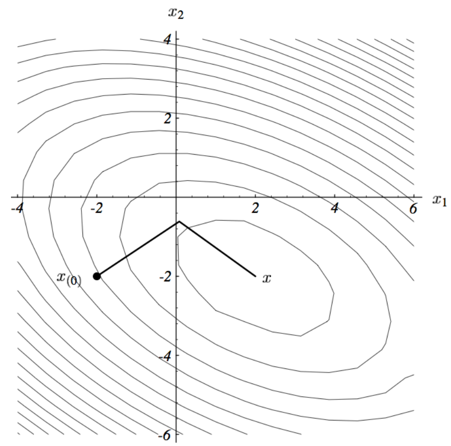
next direction built from residual and all previous directions ("conjugate" to all previous directions)

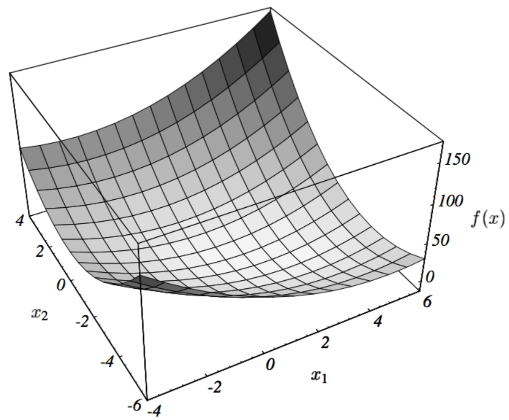

positive-definite orthogonality, linear independence
Steepest Descent
A-conjugate
Conjugate Direction
conjugate residuals (i.e. gradient)
Conjugate Gradient

TODAY'S SUMMARY

- Part 1. Quasi-Newton
- Part 2. Conjugate methods
- Part 3. Steepest Descent
- Part 4. some concepts
- Part 5. Conjugate Gradient



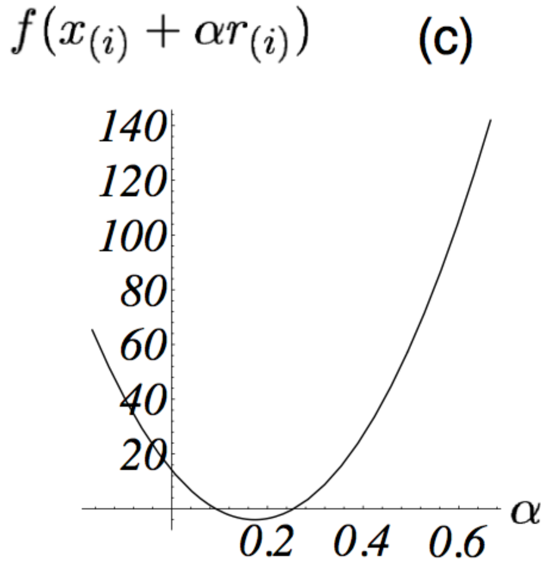




some proofs

show that A-orthogonality occurs at min point along search direction

min point occurs when ...



show that A-orthogonality occurs at min point along search direction

$$\frac{d}{d\alpha} f(x_{(i+1)}) = 0$$

show that A-orthogonality occurs at min point along search direction

$$\frac{d}{d\alpha} f(x_{(i+1)}) = 0$$

chain rule:

$$\frac{d}{d\alpha} f(x_{(i)}) = f'(x_{(i)})^T \frac{d}{d\alpha} x_{(i)} = f'(x_{(i)})^T d_{(i-1)}$$

show that A-orthogonality occurs at min point along search direction

$$\begin{aligned}\frac{d}{d\alpha} f(x_{(i+1)}) &= 0 \\ f'(x_{(i+1)})^T \frac{d}{d\alpha} x_{(i+1)} &= 0\end{aligned}$$

steepest descent...
(residual direction away from gradient)

show that A-orthogonality occurs at min point along search direction

$$
\begin{aligned}
\frac{d}{d\alpha} f(x_{(i+1)}) &= 0 \\
f'(x_{(i+1)})^T \frac{d}{d\alpha} x_{(i+1)} &= 0 \\
-r_{(i+1)}^T d_{(i)} &= 0
\end{aligned}
$$

$r_{(i)} = -Ae_{(i)}$

(we derived this before)

show that A-orthogonality occurs at min point along search direction

$$
\begin{aligned}
\frac{d}{d\alpha} f(x_{(i+1)}) &= 0 \\
f'(x_{(i+1)})^T \frac{d}{d\alpha} x_{(i+1)} &= 0 \\
-r_{(i+1)}^T d_{(i)} &= 0 \\
d_{(i)}^T A e_{(i+1)} &= 0.
\end{aligned}
$$

...so direction $d_{(i)}$ is A-orthogonal with $e_{(i+1)}$ when we find min point along search direction

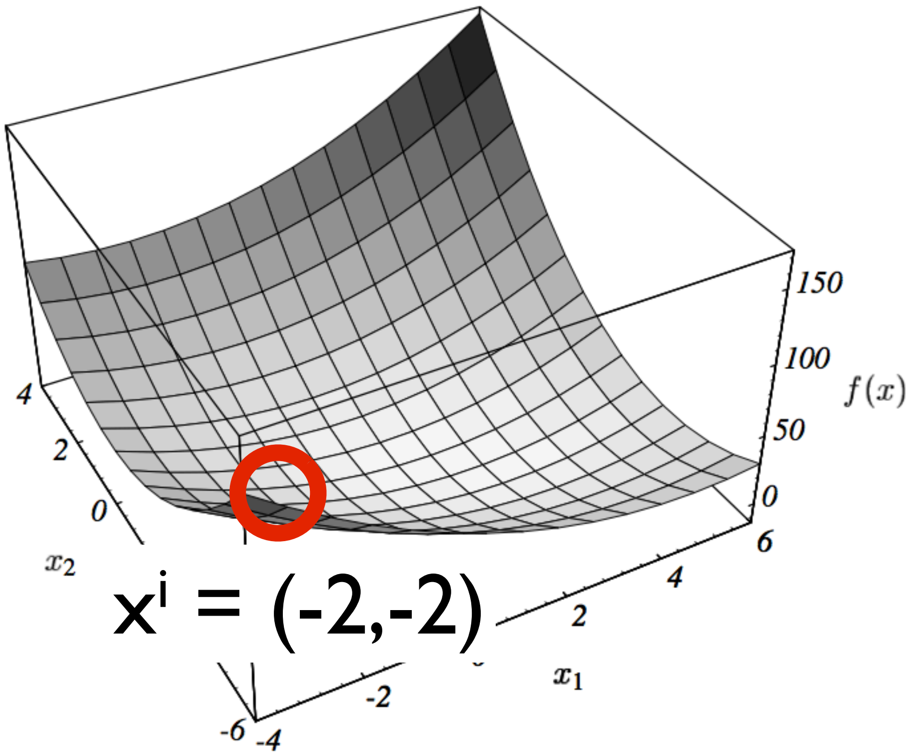
steepest descent "slides" down bowl

error $e^{(i)} = x^{(i)} - x^*$

residual $r^{(i)} = b - Ax^{(i)}$

$r^{(i)} = ?$

$r^{(i)} = -A\, e^{(i)}$ ← try to derive this



$x^i = (-2,-2)$

steepest descent "slides" down bowl

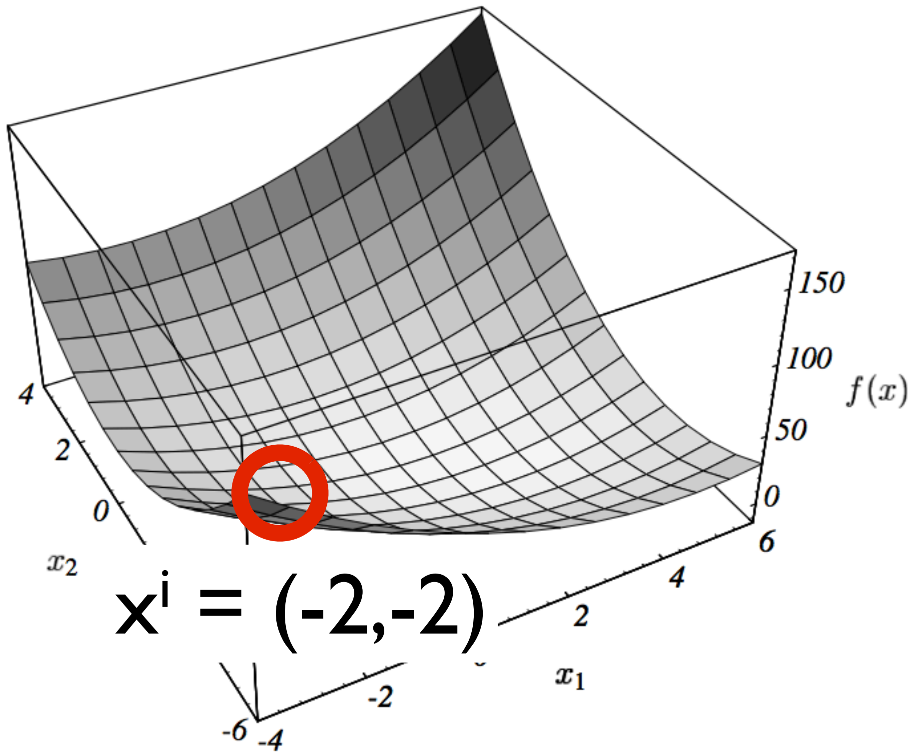
$$e^{(i)} = x^{(i)} - x^*$$

$$r^{(i)} = b - Ax^{(i)}$$

$$r^{(i)} = -A\,e^{(i)}$$

$$x^{(i)} = e^{(i)} + x^*$$

$$r^{(i)} = b - A(e^{(i)} + x^*)$$



$x^i = (-2,-2)$

steepest descent "slides" down bowl

$$e^{(i)} = x^{(i)} - x^*$$

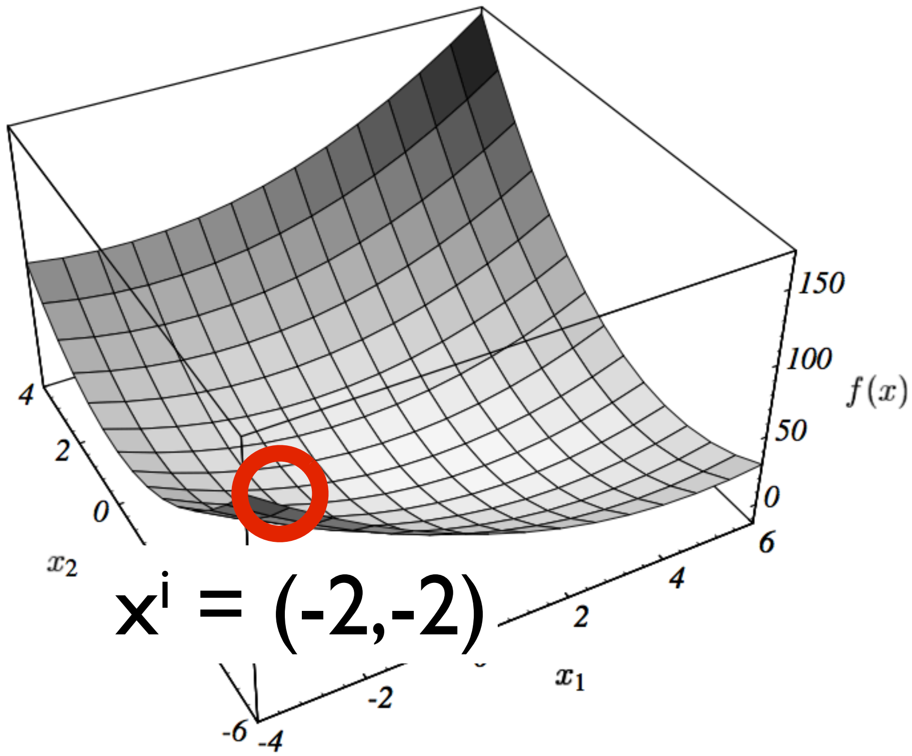
$$r^{(i)} = b - Ax^{(i)}$$

$$x^{(i)} = e^{(i)} + x^*$$

$$r^{(i)} = b - A(e^{(i)} + x^*)$$

$$r^{(i)} = b - Ae^{(i)} - Ax^*$$

$$r^{(i)} = -A\, e^{(i)}$$



$x^i = (-2,-2)$

steepest descent "slides" down bowl

$$e^{(i)} = x^{(i)} - x^*$$

$$r^{(i)} = b - Ax^{(i)}$$

$$r^{(i)} = -A\, e^{(i)}$$

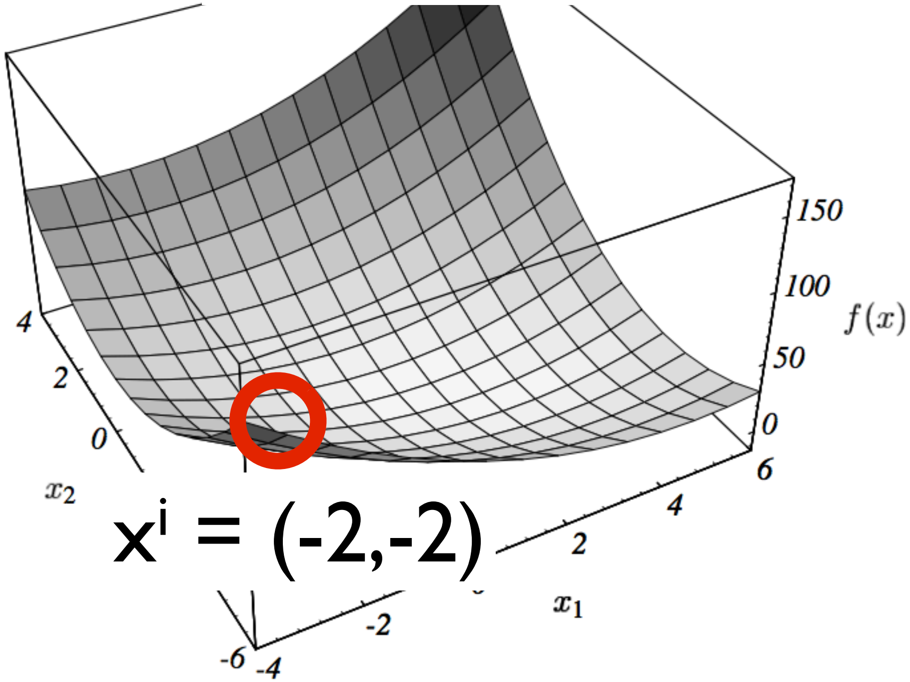
$$x^{(i)} = e^{(i)} + x^*$$

$$r^{(i)} = b - A(e^{(i)} + x^*)$$

$$r^{(i)} = \underline{b} - Ae^{(i)} - \underline{Ax^*}$$

$$r^{(i)} = -Ae^{(i)}$$

$b - Ax^* = 0$ (root)



$x^i = (-2,-2)$