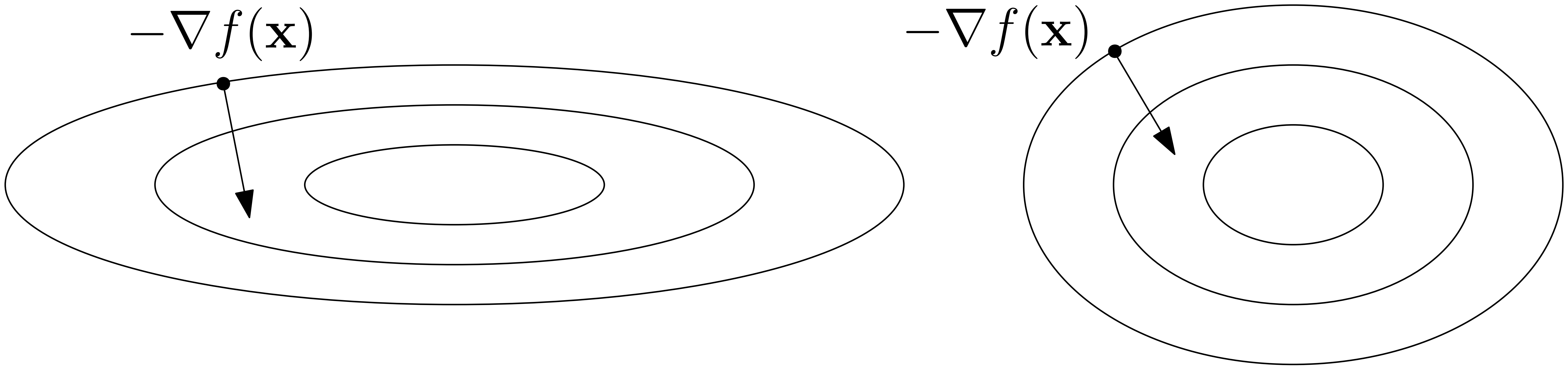
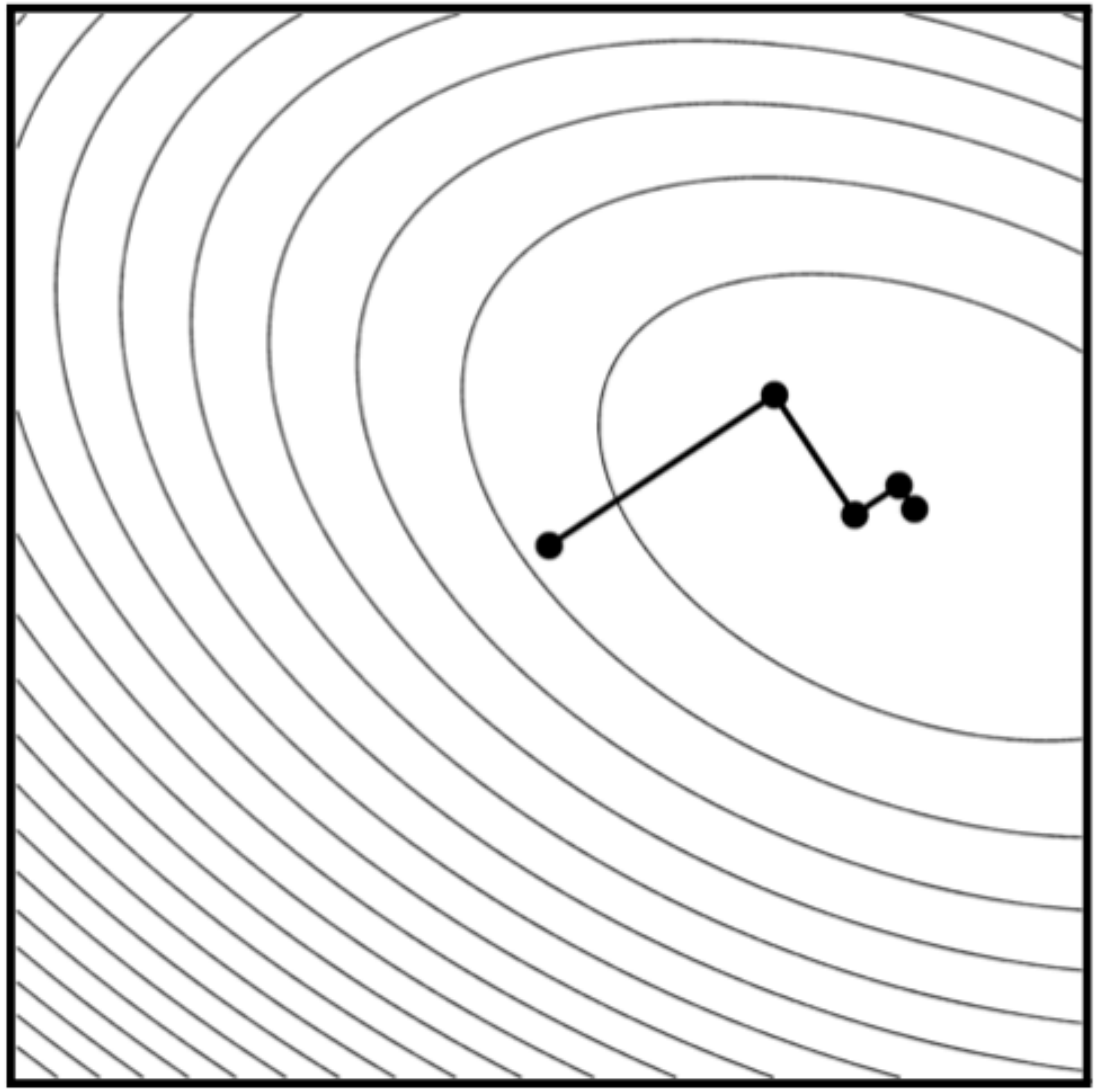
SD for SPD Systems – Behaviour

Steepest descent can suffer from slow convergence for poorly conditioned $\mathbf{A}$

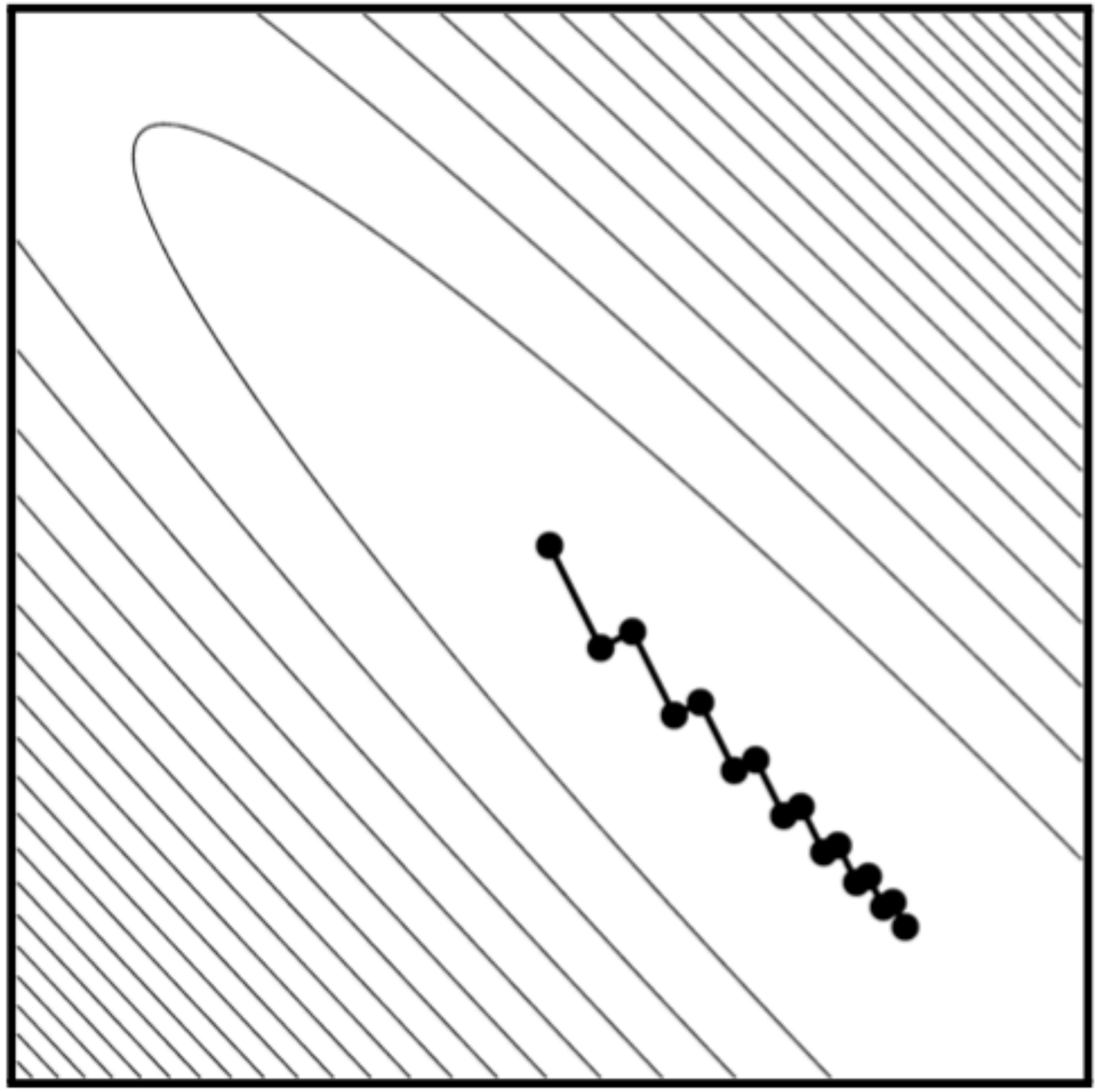
- as $\mathrm{cond}(\mathbf{A}) = \sigma_{\max}/\sigma_{\min}$ becomes large, $\mathbf{d}_k = -\nabla f(\mathbf{x}_k)$ may not point in the direction of a (global) minimum of $f(\mathbf{x})$



SD for SPD Systems – Behaviour



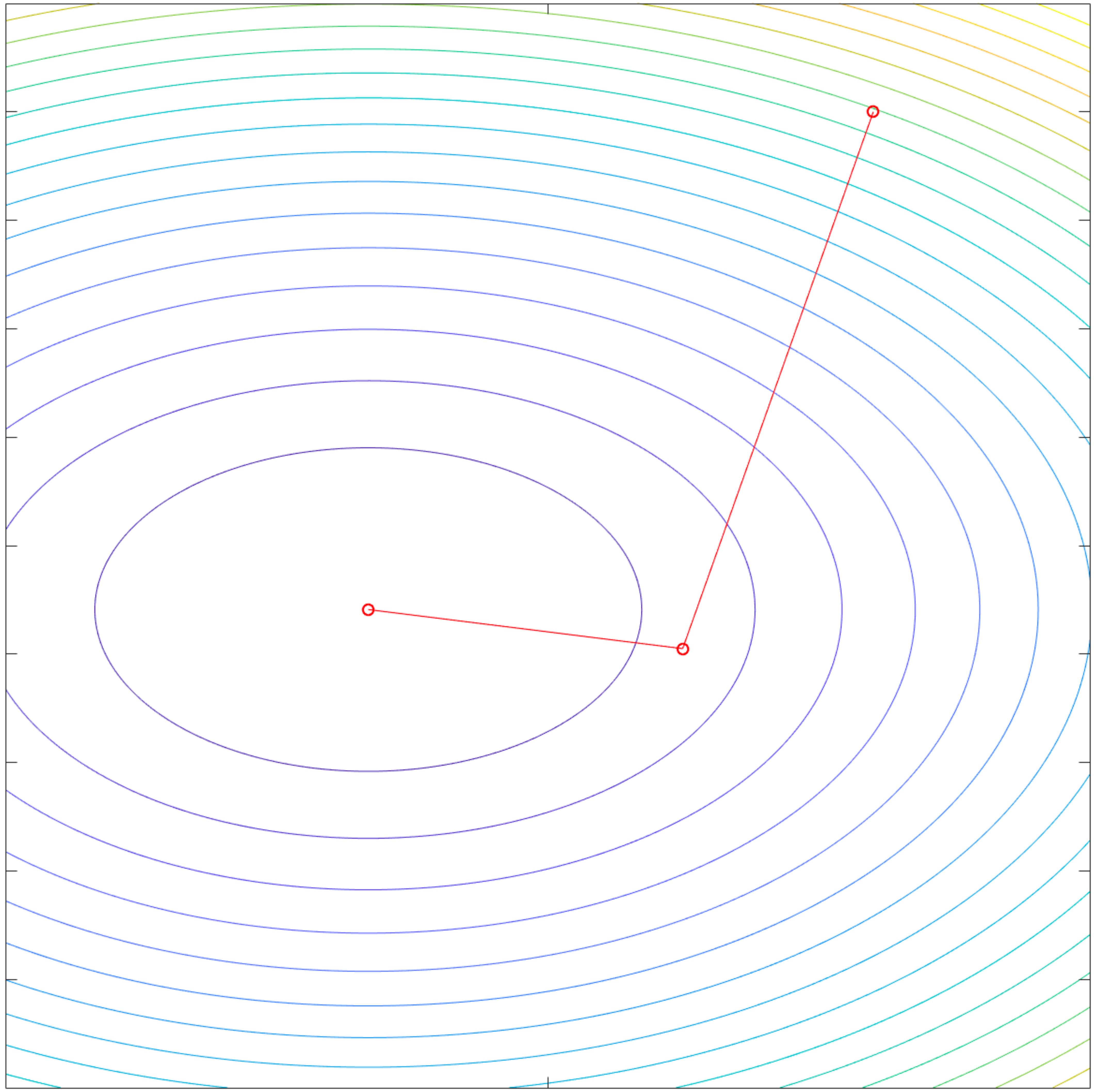
Well conditioned A



Poorly conditioned A

Motivating Conjugate Gradients

Instead of descending down the gradient direction, the **conjugate gradient** scheme takes directions designed to avoid zig-zagging



Conjugate Gradient – Better Directions

We arrived at SD from GD by choosing the optimal step size and *leaving the direction as the gradient*

- we observed that each pair of iterative descent directions are *perpendicular to each other*
  - zig zag
- CG seeks to find **descent directions** that avoid zig-zagging

Starting from the general iterative update rule, we now leave **both** the descent direction **and** step size variable:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \, \mathbf{d}_k$$

CG – General Step Size Optimum

$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k\, \mathbf{d}_k \longrightarrow f(\mathbf{x}_{k+1}) \longrightarrow$ and solve for α_k in $df(\mathbf{x}_{k+1})\,/\,d\alpha_k = 0$

$$
\begin{aligned}
f(\mathbf{x}_{k+1}) &= \tfrac{1}{2}\,(\mathbf{x}_k + \alpha_k\, \mathbf{d}_k)^{\mathrm{T}}\mathbf{A}(\mathbf{x}_k + \alpha_k\, \mathbf{d}_k) - \mathbf{b}^{\mathrm{T}}(\mathbf{x}_k + \alpha_k\, \mathbf{d}_k) + c \\
&= \tfrac{1}{2}\, \mathbf{x}_k^{\mathrm{T}}\mathbf{A}\mathbf{x}_k + \alpha_k\, \mathbf{d}_k^{\mathrm{T}}\mathbf{A}\mathbf{x}_k + \tfrac{1}{2}\,\alpha_k^2\, \mathbf{d}_k^{\mathrm{T}}\mathbf{A}\mathbf{d}_k - \mathbf{b}^{\mathrm{T}}\mathbf{x}_k - \alpha_k\, \mathbf{b}^{\mathrm{T}}\mathbf{d}_k + c \\
&= [\tfrac{1}{2}\, \mathbf{x}_k^{\mathrm{T}}\mathbf{A}\mathbf{x}_k - \mathbf{b}^{\mathrm{T}}\mathbf{x}_k + c] + [\alpha_k\, \mathbf{d}_k^{\mathrm{T}}(\mathbf{A}\mathbf{x}_k - \mathbf{b})] + \tfrac{1}{2}\,\alpha_k^2\, \mathbf{d}_k^{\mathrm{T}}\mathbf{A}\mathbf{d}_k \\
&= \qquad f(\mathbf{x}_k) \qquad + \alpha_k\, \mathbf{d}_k^{\mathrm{T}}\, \nabla f(\mathbf{x}_k) \qquad + \tfrac{1}{2}\,\alpha_k^2\, \mathbf{d}_k^{\mathrm{T}}\mathbf{A}\mathbf{d}_k
\end{aligned}
$$

$$
\frac{df(\mathbf{x}_{k+1})}{d\alpha_k} = 0 = \mathbf{d}_k^{\mathrm{T}}\, \nabla f(\mathbf{x}_k) + \alpha_k \mathbf{d}_k^{\mathrm{T}}\mathbf{A}\mathbf{d}_k \longrightarrow \boxed{\alpha_k = -\frac{\mathbf{d}_k^T \nabla f(\mathbf{x}_k)}{\mathbf{d}_k^T \mathbf{A}\mathbf{d}_k}}
$$

CG – Building Conjugate Directions

To find better search directions $\mathbf{d}_k$ we need a few definitions:

Two vectors $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^n$ are $\mathbf{A}$-conjugate (or $\mathbf{A}$-orthogonal) if

$$\mathbf{x}_1^T \mathbf{A} \mathbf{x}_2 = 0$$

- geometric interpretation?

A set of vectors $\mathcal{S} = \{\mathbf{x}_1, \ldots, \mathbf{x}_k\} \in \mathbb{R}^n$ are an $\mathbf{A}$-conjugate set if

$$\mathbf{x}_i^T \mathbf{A} \mathbf{x}_j = 0 \,, \forall i \neq j$$

CG – Building Conjugate Directions

<u>Note</u>: if $\mathbf{A} > 0$ and $\mathbf{A} = \mathbf{A}^{\mathrm{T}}$ then $\mathcal{S}$ are also linearly independent

- recall that linear independence means that

$$\alpha_1 \mathbf{x}_1 + \alpha_2 \mathbf{x}_2 + \cdots + \alpha_{k-1} \mathbf{x}_{k-1} + \alpha_k \mathbf{x}_k = \mathbf{0}$$

  can only hold if $\alpha_i = 0$, $i = 1, \ldots, k$.

- pre-multiplying by $\mathbf{A}$ and then by $\mathbf{x}_i^{\mathsf{T}}$ we arrive at

$$\begin{aligned}
\alpha_1 \mathbf{A}\mathbf{x}_1 + \alpha_2 \mathbf{A}\mathbf{x}_2 + \cdots + \alpha_{k-1} \mathbf{A}\mathbf{x}_{k-1} + \alpha_k \mathbf{A}\mathbf{x}_k &= \mathbf{0}, \\
\alpha_1 \mathbf{x}_i^{\mathsf{T}} \mathbf{A}\mathbf{x}_1 + \alpha_2 \mathbf{x}_i^{\mathsf{T}} \mathbf{A}\mathbf{x}_2 + \cdots + \alpha_{k-1} \mathbf{x}_i^{\mathsf{T}} \mathbf{A}\mathbf{x}_{k-1} + \alpha_k \mathbf{x}_i^{\mathsf{T}} \mathbf{A}\mathbf{x}_k &= \mathbf{0}, \\
\alpha_i \mathbf{x}_i^{\mathsf{T}} \mathbf{A}\mathbf{x}_i &= 0.
\end{aligned}$$

- since $\alpha_i \mathbf{x}_i^{\mathsf{T}} \mathbf{A}\mathbf{x}_i > 0, \forall i$ it stands that $\alpha_i = 0, \forall i$.

CG – Building Conjugate Directions

Given these definitions and relationships, we can now outline our strategy for choosing descent directions $\mathbf{d}_k$

- the descent directions $\mathbf{d}_k \in \mathbb{R}^n$ will form a finite $\mathbf{A}$-conjugate set
  - as such, they will also form a basis for $\mathbb{R}^n$
- we can thus express solutions $\mathbf{x}$ of $\mathbf{A}\mathbf{x} = \mathbf{b}$ as $\mathbf{x} = \sum_{i=1}^{n} \alpha_i \mathbf{d}_i$

We will iteratively build the $\mathbf{A}$-conjugate set (and, so too, the basis) such that at each step j we choose a new descent direction $\mathbf{d}_j$ to be $\mathbf{A}$-conjugate to all preceding descent directions, i.e., $\mathbf{d}_j^\top \mathbf{A} \mathbf{d}_i = 0\,, \forall i < j$

CG – Building Conjugate Directions

Drawing from steepest descent, we will seek $\mathbf{d}_k$s of the form

$$\mathbf{d}_{k+1} = -\nabla f(\mathbf{x}_{k+1}) + \beta_k \mathbf{d}_k$$

where $\mathbf{d}_k$ is $\mathbf{A}$-conjugate to the previous directions $\mathbf{d}_i, \forall i < k$

- this reduces the problem to finding the appropriate β_ks
  - note that with $\beta_k = 0, \forall k$ we recover the steepest descent directions

Pre-multiplying the equation for $\mathbf{d}_{k+1}$ by $\mathbf{A}$ and then $\mathbf{d}_k^\mathsf{T}$ gives:

$$\mathbf{A}\mathbf{d}_{k+1} = -\mathbf{A}\nabla f(\mathbf{x}_{k+1}) + \beta_k \mathbf{A}\mathbf{d}_k$$
$$\mathbf{d}_k^\mathsf{T}\mathbf{A}\mathbf{d}_{k+1} = -\mathbf{d}_k^\mathsf{T}\mathbf{A}\nabla f(\mathbf{x}_{k+1}) + \beta_k \mathbf{d}_k^\mathsf{T}\mathbf{A}\mathbf{d}_k$$

- this reduces the problem to finding the appropriate β_ks
  - note that with $\beta_k = 0, \forall k$ we recover the steepest descent directions

Pre-multiplying the equation for $\mathbf{d}_{k+1}$ by $\mathbf{A}$ and then $\mathbf{d}_k^\top$ gives:

$$\mathbf{A}\mathbf{d}_{k+1} = -\mathbf{A}\nabla f(\mathbf{x}_{k+1}) + \beta_k \mathbf{A}\mathbf{d}_k$$
$$\mathbf{d}_k^\top \mathbf{A}\mathbf{d}_{k+1} = -\mathbf{d}_k^\top \mathbf{A}\nabla f(\mathbf{x}_{k+1}) + \beta_k \mathbf{d}_k^\top \mathbf{A}\mathbf{d}_k$$

We need to force the LHS to be 0 to maintain $\mathbf{A}$-conjugacy, which yields

$$\boxed{\beta_k = \frac{\mathbf{d}_k^\top \mathbf{A}\nabla f(\mathbf{x}_{k+1})}{\mathbf{d}_k^\top \mathbf{A}\mathbf{d}_k}}$$

Conjugate Gradients – Algorithm

The **conjugate gradient algorithm** is a modification to SD as:

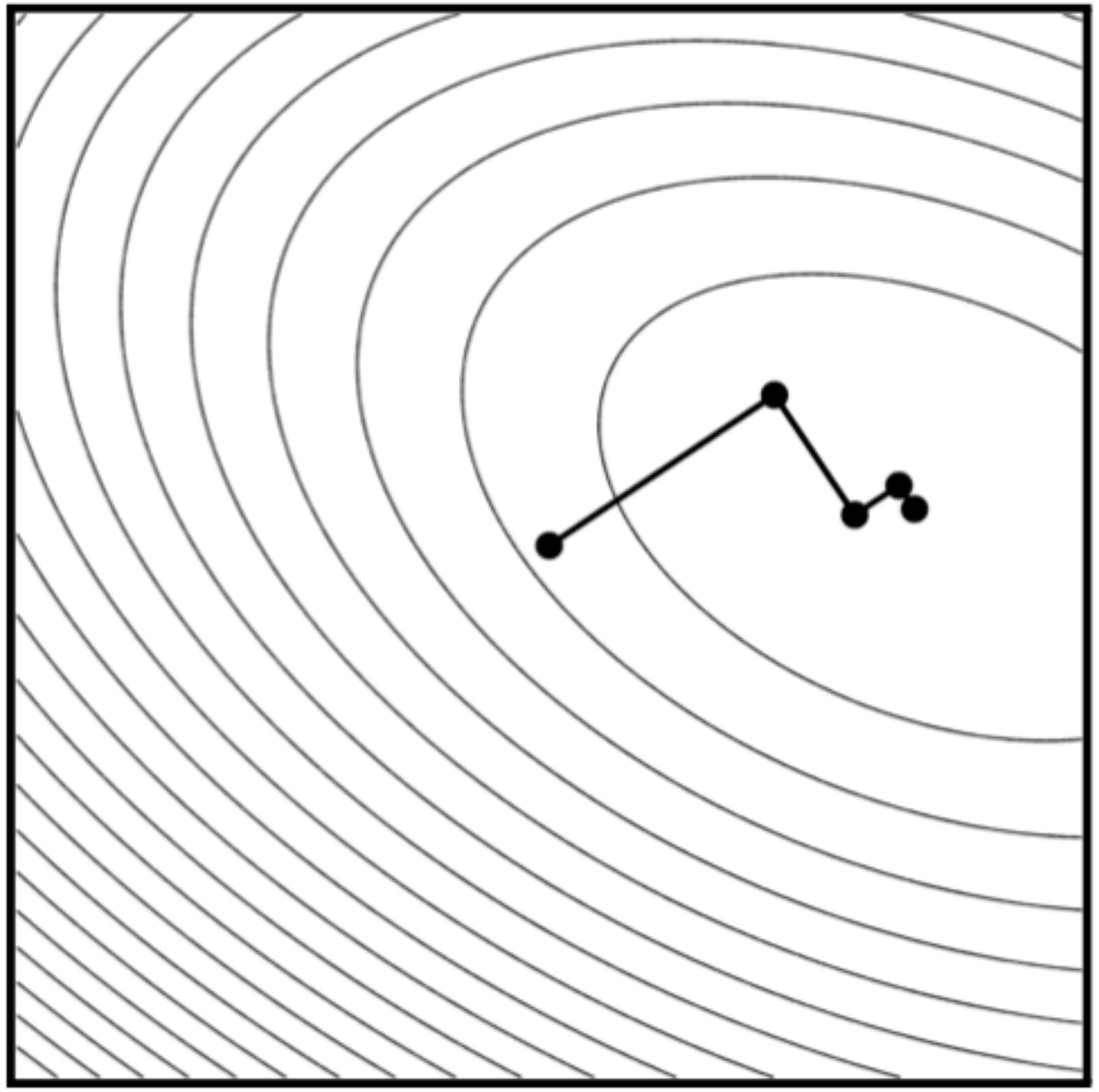
1. begin with any vector $\mathbf{x}_1 \in \mathbb{R}^n$ and set $\mathbf{d}_1 = -\nabla f(\mathbf{x}_1)$, then iteratively solve for
2. the minimizing distance $\alpha_k = \dfrac{-\mathbf{d}_k^\top \nabla f(\mathbf{x}_k)}{\mathbf{d}_k^\top \mathbf{A} \mathbf{d}_k}$ and next CG iterate $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k$, before forming the next CG descent direction as
3. $\mathbf{d}_{k+1} = -\nabla f(\mathbf{x}_{k+1}) + \beta_k \mathbf{d}_k$ with offset distance $\beta_k = \dfrac{\mathbf{d}_k^\top \mathbf{A} \nabla f(\mathbf{x}_{k+1})}{\mathbf{d}_k^\top \mathbf{A} \mathbf{d}_k}$

Conjugate Gradients for Linear Systems

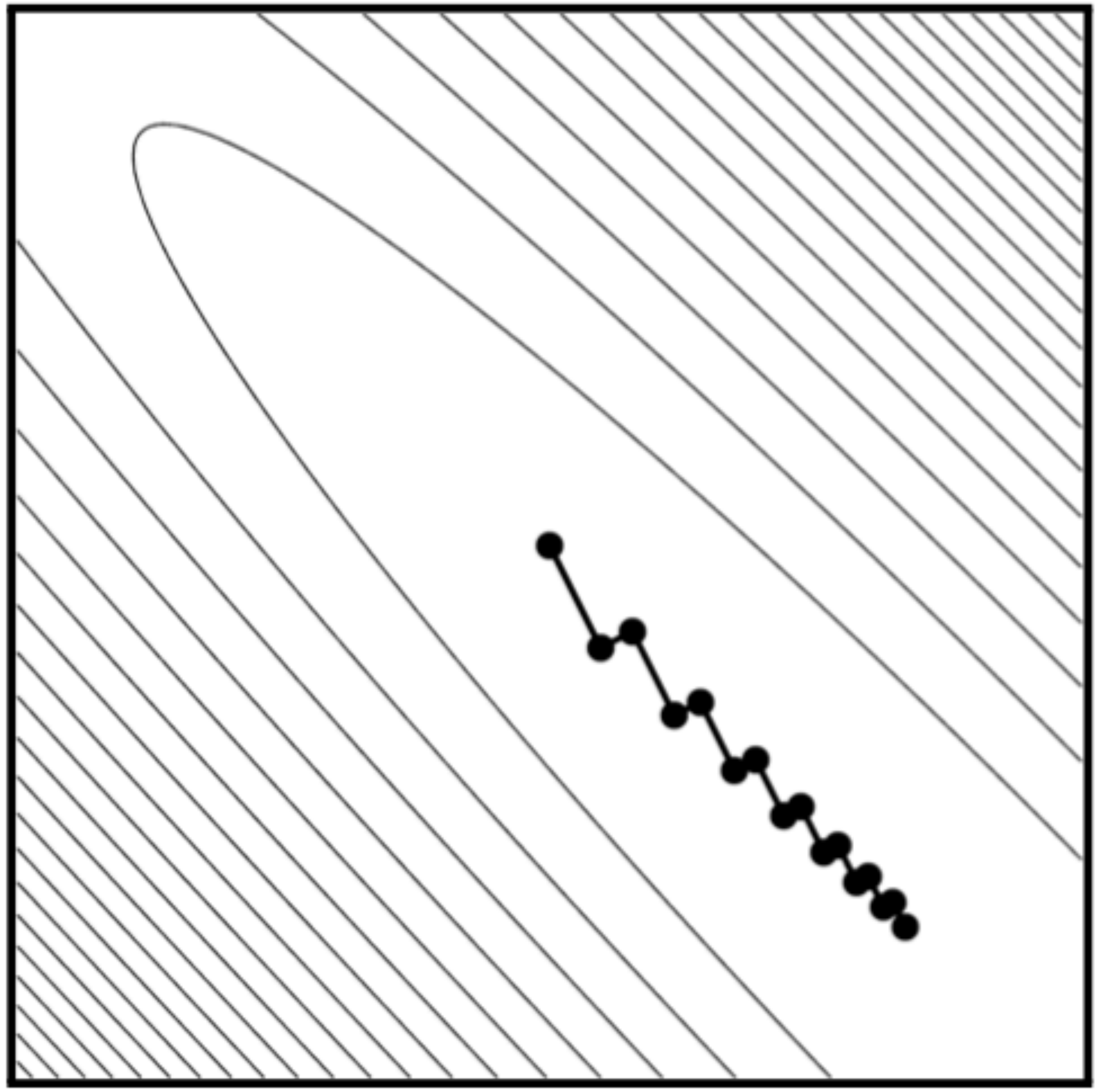
Conjugate gradient has much nicer convergence properties:

- it is **guaranteed*** to converge to the solution in **at most** n steps
    - this means $O(n^2)$ overall cost for sparse systems, and at worse the same $O(n^3)$ cost as direct methods for dense systems – in practice, often $n' \ll n$ iterations are needed
- the conjugate gradient algorithm follows the same structure as gradient descent, but requires some additional linear algebraic manipulation when constructing the conjugate descent directions

Gradient Descent vs. Conjugate Gradient

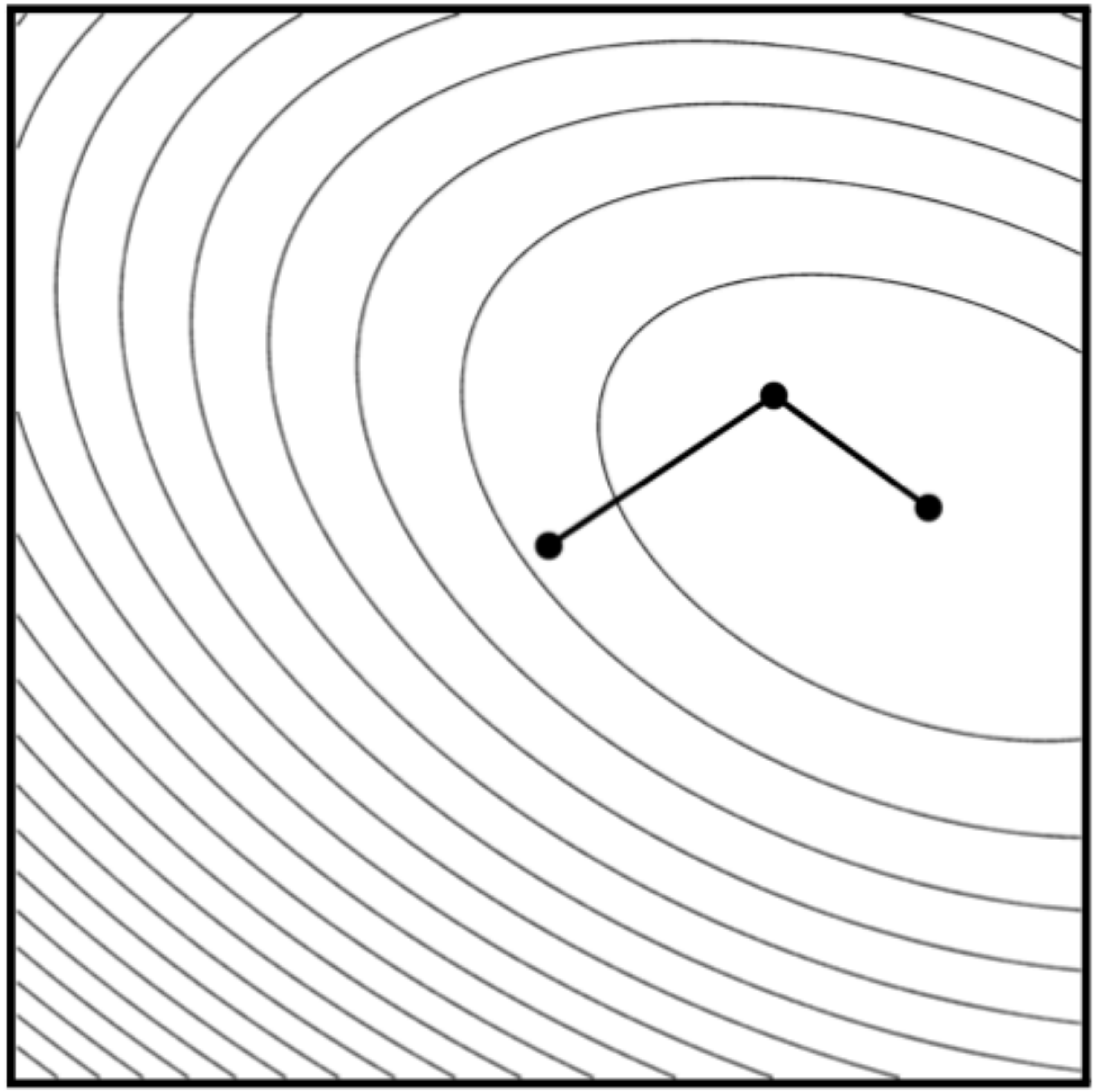


Well conditioned A

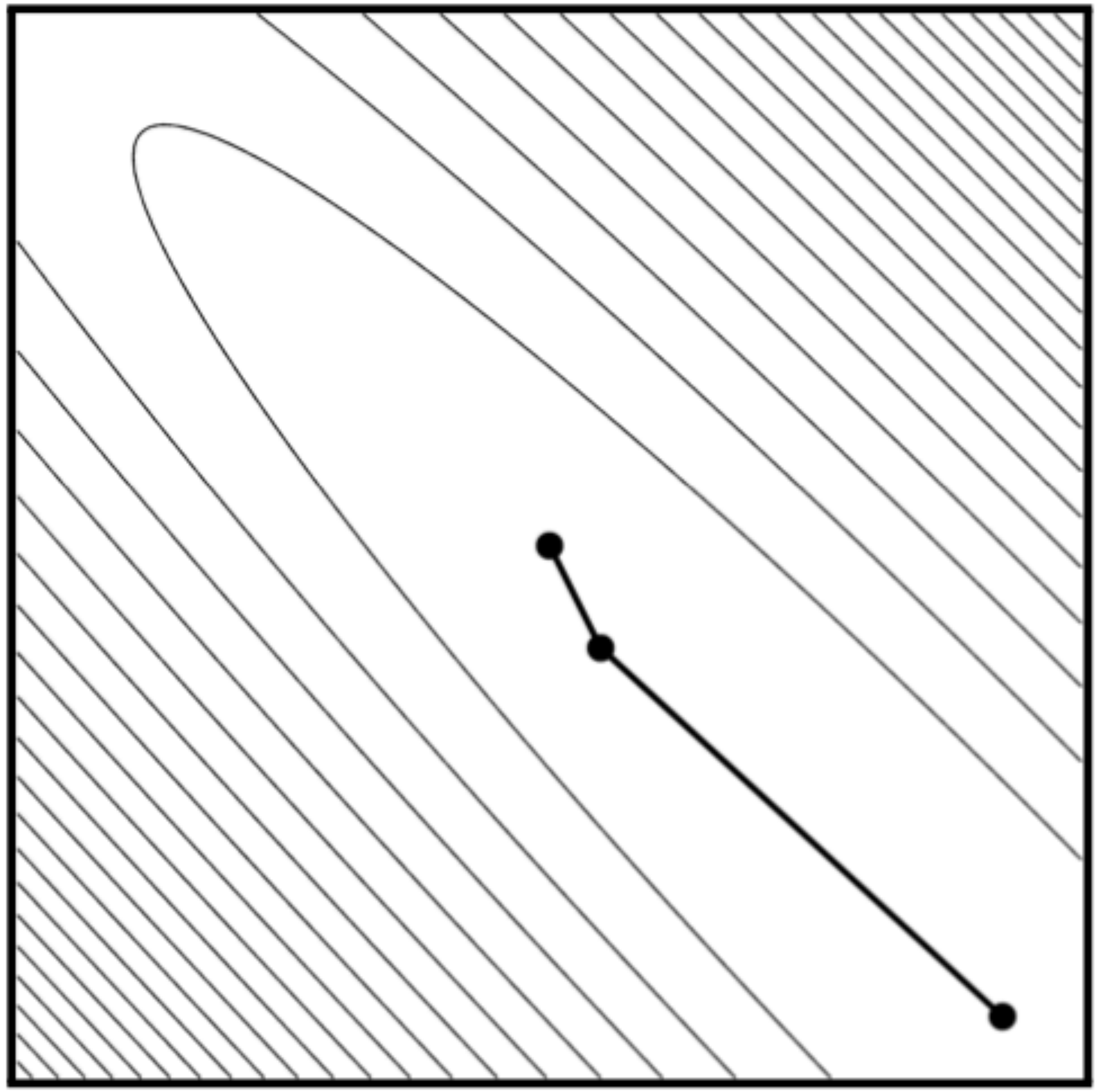


Poorly conditioned A

Gradient Descent vs. Conjugate Gradient



Well conditioned A



Poorly conditioned A

Gradient Descent – Summary

Gradient descent is a simple and powerful technique

- many extensions and deeply studied area

In its simplest form, it requires "only" the ability to evaluate the gradient of the function we wish to minimize

- can set step size manually, or take several steps with, e.g., adaptive sizes
- *line search* can be worth the additional costs; requires function evaluations for the 1D optimization

Gradient descent can be specialized to a linear solver with an optimal step size (that doesn't require a line search)

- conjugate gradient has better numerics and stronger guarantees