Lecture X: Iterative methods for sparse systems

Singular-value decomposition vs. Spectral decomposition

What is it?

$A = U S V^\dagger$ (U, V unitary; S diagonal)

$= \sum_i \sigma_i \vec{u}_i \vec{v}_i^\dagger$ (σ_i: sing. values ≥ 0; $\vec{u}_i$: left singular vectors; $\vec{v}_i$: right singular vectors; $\{u_i\}, \{v_i\}$ orthonormal)

$A = T D T^{-1}$ (D diagonal)

if A is normal:

$A = T D T^\dagger$ (T: same unitary!)

$= \sum_i \lambda_i \vec{t}_i \vec{t}_i^\dagger$ (λ_i: e-values; $\vec{t}_i$: e-vectors (orthonormal))

When?

all matrices!

square, diagonalizable matrices

$\mathbb{R}$ or $\mathbb{C}$?

real if A is real

complex even if A is real

eg. $\begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \to e^{\pm i\theta}$

real if A is real & symmetric

Other properties

$\|A\| = \max$ sing. value(A)

$\|A\| \neq \max |\text{eigenvalue}|$ in general

eg.: $\begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$ has evalues $0, 0$

$A \to A + I$ does not shift sing. values in general, eg. $\begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \to \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$

svs $1, 1$ svs $2, 0$

evs $1, -1$ evs $2, 0$

$A \to A + \alpha I$ shifts e-values by α

Example:

$$\begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} = \underbrace{1}_{\sigma_1} \cdot \underbrace{\begin{pmatrix} c \\ s \end{pmatrix}}_{\vec{u}_1} \underbrace{(1 \;\; 0)}_{\vec{v}_2^\dagger} + \underbrace{1}_{\sigma_2} \cdot \underbrace{\begin{pmatrix} -s \\ c \end{pmatrix}}_{\vec{u}_2} \underbrace{(0 \;\; 1)}_{\vec{v}_2^\dagger}$$

$$= \underbrace{e^{-i\theta}}_{\lambda_1} \underbrace{\frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ i \end{pmatrix}}_{\vec{v}_1} \underbrace{\frac{(1 \;\; -i)}{\sqrt{2}}}_{\vec{v}_1^\dagger} + e^{i\theta} \underbrace{\frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -i \end{pmatrix}}_{\vec{v}_2} \underbrace{\frac{(1 \;\; i)}{\sqrt{2}}}_{\vec{v}_2^\dagger}$$

λ_2

They can't be entirely different. What's the connection?

Recall: For any $m \times n$ matrix A,

$A^\dagger A \geq 0$.

In particular, $A^\dagger A$ is Hermitian (and thus normal), and its eigenvalues are all real and ≥ 0.

Proposition: Let A be an $m \times n$ real or complex matrix.

Then,

- The nonzero singular values of A are the square-roots of the eigenvalues of A^TA (same as the square-roots of the e-values of AA^T).
- Eigenvectors of A^TA are right singular vectors of A.
- Eigenvectors of AA^T are left singular vectors of A.

Corollary: AA^T and A^TA have the same nonzero eigenvalues.

Corollary: How to compute the SVD of a matrix?
Answer: Prove this proposition!

Proof:
Let $A = \sum_i \sigma_i \vec{u}_i \vec{v}_i^T$ be the SVD of A.
(We don't know what it is, maybe, but it exists!)

$$\Rightarrow A^TA = \left(\sum_i \sigma_i \vec{v}_i \vec{u}_i^T\right)\left(\sum_j \sigma_j \vec{u}_j \vec{v}_j^T\right)$$

$$= \sum_i \sigma_i^2 \vec{v}_i \vec{v}_i^T \quad \text{since } \vec{u}_i^T\vec{u}_j = \vec{u}_i \cdot \vec{u}_j = \begin{cases} 1 & \text{if } i=j \\ 0 & \text{if } i \neq j \end{cases}$$

$$\Rightarrow A^TA\vec{v}_j = \sigma_j^2 \vec{v}_j$$

ie., the right singular vector $\vec{v}_j$ is an e-vector of A^TA, with e-value σ_j^2. ✓

$\Rightarrow$ By diagonalizing A^TA, we now know the singular values σ_i and right singular vectors $\vec{v}_i$.
To get $\vec{u}_i$, note $\vec{u}_i = \frac{1}{\sigma_i} A\vec{v}_i$. ✓ □

Corollary: For any A,

$$\|A\| = \sqrt{\text{max e-value of } A^TA}$$
$$= \sqrt{\text{max e-value of } AA^T}.$$

For normal matrices, the relationship between singular values and eigenvalues is even simpler.

A normal

$$\Rightarrow A = U D U^\dagger$$

(U: unitary; D: diag. w/ e-values)

$$= \sum_i \lambda_i \vec{u}_i \vec{u}_i^\dagger$$

($\vec{u}_i$: e-vectors (columns of U))

$$= \sum_i |\lambda_i| \cdot \left(\frac{\lambda_i}{|\lambda_i|}\vec{u}_i\right) \cdot \vec{u}_i^\dagger$$

$$= \sum_i |\lambda_i| \cdot \left(\frac{\lambda_i}{|\lambda_i|}\vec{u}_i\right) \cdot \vec{u}_i^\dagger$$

This is an SVD for A! (sing. values; left/right sing. vectors)

Claim: If A is normal, then its singular values = absolute values of eigenvalues.

Corollary: If A is normal, $\|A\| = \max|\text{eigenvalue of } A|$.

(Again, this is generally false for non-normal matrices.)

Example: If A is orthogonal or unitary, all its singular values are 1.

(Of course, we already knew this; an SVD for A is $A = \sum_i (Ae_i)e_i^T$, since the set $\{Ae_i\}$ is orthonormal.)

Exercise: Which of these matrices is p.s.d.?

$$A = \begin{pmatrix} 3 & 4 & 5 & 0 & 1 \\ 4 & 2 & 1 & 0 & 1 \\ 5 & 1 & 10 & 2 & 1 \\ 0 & 0 & 2 & 2 & 1 \\ 1 & 1 & 1 & 2 & 3 \end{pmatrix}$$ No! $A^T \neq A$

$$B = \begin{pmatrix} 3 & 1 & 1 & 0 & 1 \\ 1 & 2 & 1 & 0 & 1 \\ 1 & 1 & -1 & 2 & 1 \\ 0 & 0 & 2 & 2 & 1 \\ 1 & 1 & 1 & 1 & 3 \end{pmatrix}$$ No! $\vec{e}_3^T B \vec{e}_3 = -1 < 0$

$$C = \begin{pmatrix} 5 & 2 & 0 & 2 & 1 \\ 2 & 6 & 1 & 2 & 0 \\ 0 & 1 & 5 & 0 & 0 \\ 2 & 2 & 0 & 6 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$ No! Observe the submatrix $C' = \begin{pmatrix} 5 & 1 \\ 1 & 0 \end{pmatrix}$, $\det C' = -1 < 0$

or: $(-x \;\; 1) C' \begin{pmatrix} -x \\ 1 \end{pmatrix} = 5x^2 - 2x = x(5x-2) < 0$ for $0 < x < \frac{2}{5}$

$$D = \begin{pmatrix} 8 & -2 & 3 & \frac{1}{2} & 0 \\ -2 & 6 & 1 & 1 & -1 \\ 3 & 1 & 10 & 2 & 1 \\ \frac{1}{2} & 1 & 2 & 5 & 1 \\ 0 & -1 & 1 & 2 & 7 \end{pmatrix}$$ Yes! $D = D^T$ ✓

Also D is diagonally dominant: $\forall i,\ D_{ii} > \sum_{j \neq i} |D_{ij}|$.

Assume for contradiction $D\vec{v} = \lambda\vec{v}$ with $\lambda < 0$, $\vec{v} \neq \vec{0}$.
Let $i = \text{argmax}\,|v_j|$.
We may assume that $v_i > 0$. (If $v_i < 0$, multiply by -1)

$$(D\vec{v})_i = D_{ii}v_i + \sum_{j\neq i} D_{ij}v_j$$

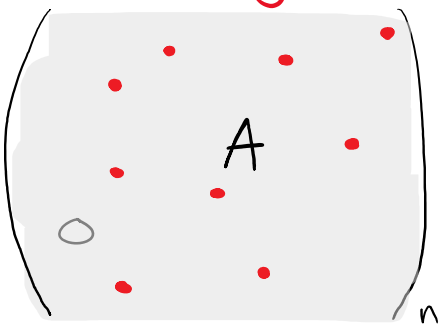
$$\lambda v_i < 0$$

$$\geq D_{ii}v_i - \sum_{j\neq i} |D_{ij}| \cdot |v_j| \quad (\leq v_i)$$

$$\geq v_i \left(D_{ii} - \sum_{j\neq i} |D_{ij}|\right)$$

$$> 0. \quad \text{X contradiction} \quad \square$$

Sparse linear systems



$$\vec{x} = \vec{b}$$

Think $n \lesssim 10^8$, # of $\bullet \approx c \cdot n$
$\Rightarrow$ few GB to store A
gigabytes

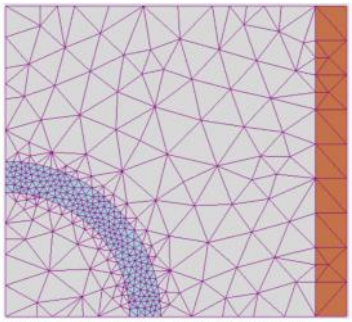
~~giga~~ ~~tera~~ petabytes to store A^{-1}
~years to compute A^{-1} or LU (Gaussian elim.)
$O(n^3)$ complexity

Today: Iterative methods for approximately solving sparse linear systems

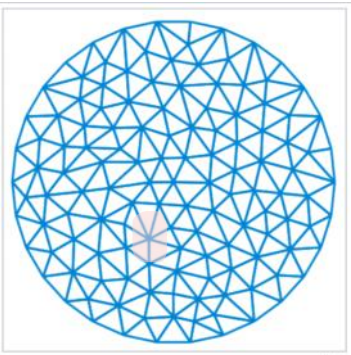
Examples:

Graph analysis
Facebook has ~10^9 users,
Google indexes ~10^{14} webpages
} with sparse connections

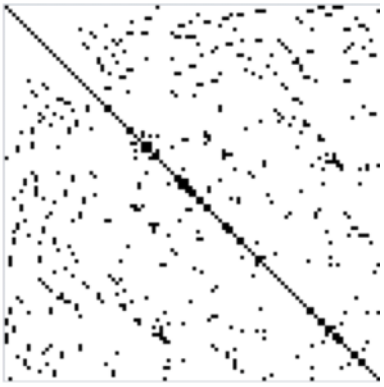
Differential equation discretization
finite difference or finite element methods
https://en.wikipedia.org/wiki/Finite_element_method



FEM mesh created by an analyst prior to finding a solution to a magnetic problem using FEM software. Colours indicate that the analyst has set material properties for each zone, in this case a conducting wire coil in orange; a ferromagnetic component (perhaps iron) in light blue; and air in grey. Although the geometry may seem simple, it would be very challenging to calculate the magnetic field for this setup without FEM software, using equations alone.



Solving the two-dimensional problem $u_{xx} + u_{yy} = -$ in the disk centered at the origin and radius 1, with zero boundary conditions.
(a) The triangulation.



(b) The sparse matrix L of the discretized linear system

$$a_{ij} = \begin{cases} \text{nonzero} & \text{if there is an edge } (i,j) \text{ (or } (i,i)) \\ 0 & \text{otherwise} \end{cases}$$

$\Rightarrow$ max # entries in a row = 1 + max. vertex degree

Example: Foster & Fedkiw '01 "Shrek"

Figure 7: *A fully articulated animated character interacts with viscous mud. The environment surrounding the character is 150x200x150 cells. That resolution is sufficient to accurately model the character filling his mouth with mud. A 3D control curve is used to eject (spit) the mouthful of mud later in the sequence. This example runs at three minutes per frame.*

$\approx 10^{6.7}$

Remark: $1000 \times 1000 \times 1000 = 10^9$ (3D grid)

Example: ECMWF 10-day weather forecasts

9km horizontal,
137 vertical levels
$\Downarrow$
10^9 grid points
(~100 vars at each point)

https://www.ecmwf.int/en/about/media-centre/news/2016/new-forecast-model-cycle-brings-highest-ever-resolution

GRADIENT DESCENT

Define a cost function

$f(\vec{x}) = \|A\vec{x} - \vec{b}\|^2$ or $f(\vec{x}) = (\vec{x}^*)^T A (\vec{x}^*)$

Define a cost function

$$C(\vec{x}) = \|A\vec{x} - \vec{b}\|^2$$

$$= \|A(x - x^*)\|^2$$

$$= (x - x^*)^T A^T A (x - x^*)$$

pos. semi-definite

or

$$C(\vec{x}) = (x - x^*)^T A (x - x^*)$$

if A is already pos. def.

Finding solution $\vec{x}^*$ $\Leftrightarrow$ Minimizing $C(\vec{x})$

Observe: C's level sets are ellipsoids. $\{\vec{x} \mid C(\vec{x}) = c\}$

If A has SVD

$$A = \sum_i \sigma_i \vec{u}_i \vec{v}_i^T$$

then $A^T A = \sum_i \sigma_i^2 \vec{v}_i \vec{v}_i^T$

$$\Rightarrow C(\vec{x}) = \sum_i \sigma_i^2 \left| v_i \cdot (x - x^*) \right|^2$$

Examples:

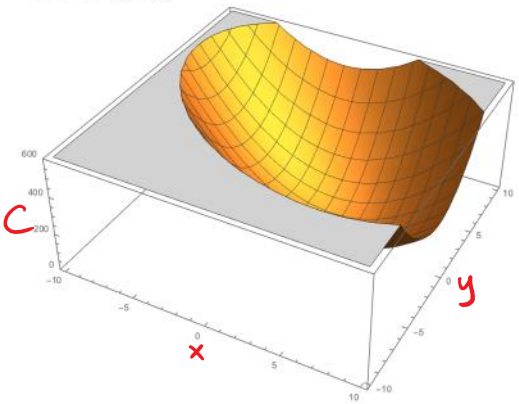
$A = \begin{pmatrix} 1 & -2 \\ 2 & 2 \end{pmatrix}$;

b = A.{2, 2}

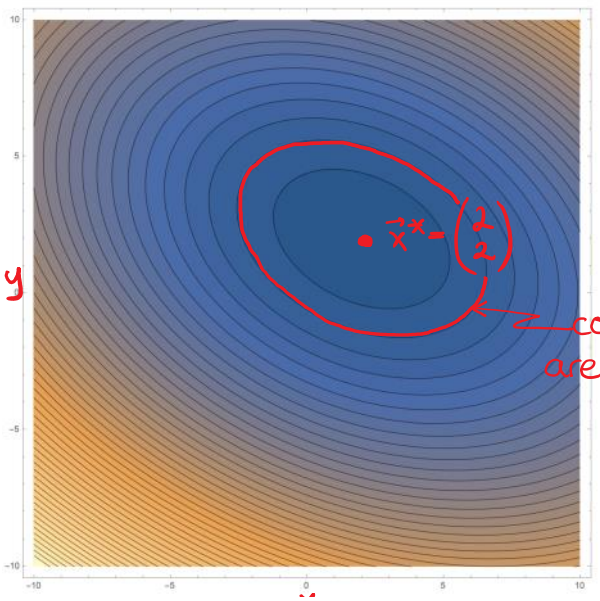
{-2, 8}

```
Plot3D[
 Norm[A.{x, y} - b]^2,
 {x, -10, 10}, {y, -10, 10},
 PlotRange → {0, 600}]
```

cost function C(x,y)




```
ContourPlot[
 Norm[A.{x, y} - b]^2,
 {x, -10, 10}, {y, -10, 10},
 Contours → 50]
```



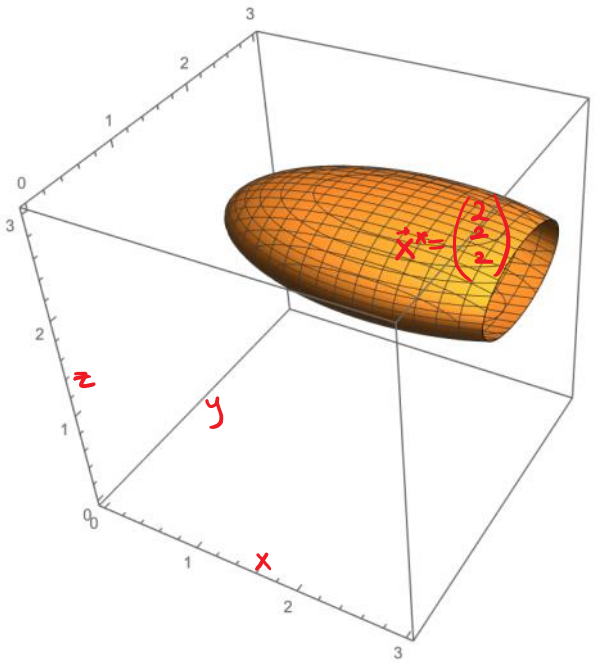
contours are ellipses (in 2D)

Ellipsoid in higher dimensions:

```
A = DiagonalMatrix[{1, 2, 4}];
b = A.{2, 2, 2};
ContourPlot3D[
 Norm[A.{x, y, z} - b]^2 == 3,
 {x, 0, 3}, {y, 0, 3}, {z, 0, 3}
]
```

Goal: Find center, the solution $\vec{x}^*$

$A\vec{x}^* = b$, $C(\vec{x}^*) = 0$.



"Gradient descent" = move downhill in the steepest direction, $-\nabla C$

$$x^{t+1} = x^t - \beta \nabla C(x^t)$$
$$= x^t - \beta \left(\frac{\partial C}{\partial x_1}, \dots, \frac{\partial C}{\partial x_n} \right)\Big|_{x^t}$$

For $C(x) = (x - x^*)^T A (x - x^*)$,

$$= \sum_{ij} a_{ij} (x_i - x_i^*)(x_j - x_j^*)$$

$$\frac{\partial C}{\partial x_i} = 2 \sum_j a_{ij} (x_j - x_j^*) = \left(2A(x - x^*) \right)_i$$

$$\Rightarrow \nabla C = \left(\frac{\partial C}{\partial x_1}, \dots, \frac{\partial C}{\partial x_n} \right)$$
$$= 2A(x - x^*)$$
$$= 2(Ax - b)$$

What is the minimum cost? Let $\vec{r}_{\text{residual}} = A\vec{x} - \vec{b}$

$$C\left(x - \tfrac{1}{2}\beta \nabla C\right) = (x - \beta r - x^*)^T A (x - \beta r - x^*)$$
$$= \beta^2 \cdot r^T A r - \beta \left(r^T A (x - x^*) + (x - x^*) A r \right) + \text{const. indep. of } \beta$$

$$= (r^T A r)\beta^2 - 2\|r\|^2 \cdot \beta + \text{const.} \quad (r = \vec{r})$$

Setting $\frac{d}{d\beta} = 0 \Rightarrow$ $\boxed{\beta = \frac{\|r\|^2}{\vec{r}^T A \vec{r}}}$

$$\Rightarrow \boxed{x^{t+1} = x^t - \frac{\|\vec{r}\|^2}{\vec{r}^T A \vec{r}} \cdot \vec{r}} \quad \text{where } \vec{r} = Ax^t - b$$

Gradient descent algorithm:

Input: A, b, x_0, t_{max}, tol

$\vec{r}_0 = A\vec{x}_0 - \vec{b}$

FOR $t = 0, 1, \ldots, t_{max}$:

IF $\|\vec{r}_t\| < (tol) \cdot \|b\|$, BREAK

$$\vec{z} = A\vec{r}_t \qquad \beta = \frac{\|\vec{r}_t\|^2}{\vec{r}_t \cdot \vec{z}}$$

$$\vec{x}_{t+1} = \vec{x}_t - \beta \vec{r}_t$$
$$\vec{r}_{t+1} = \vec{r}_t - \beta \vec{z}$$

RETURN $\vec{x}_t$

Mathematica:

```
GradientDescent[A_, b_, x0_, tmax_, tol_] := Module[{list, x, r, rnorm, bnorm, z, β, t},
 bnorm = Norm[b];
 x = x0;
 list = {x};
 r = A.x - b;
 For[t = 0, t < tmax, t++,
  rnorm = Norm[r];
  If[rnorm < tol bnorm, Break[];];
  z = A.r;
  β = rnorm^2/r.z;
  x = x - β r;
  r = r - β z;
  AppendTo[list, x];
 ];
 "This returns not just the (approximate) solution, but the whole list of values
   x0,...,xt tried along the way.";
 list
];
```

Example:

```
A = {{2, -1.5}, {-1.5, 4}};
"A is positive definite:"
Eigenvalues[A]

"The exact solution:"
xstar = {3, 3}

b = A.{3, 3};
```

A is positive definite:

{4.80278, 1.19722}

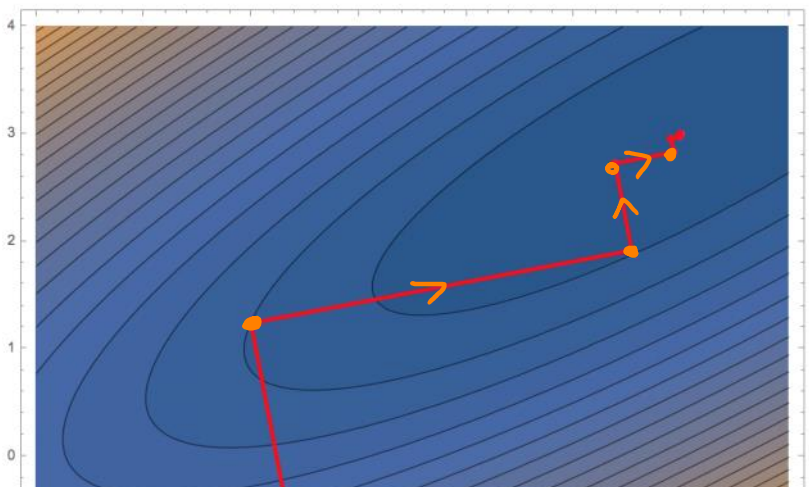
The exact solution:

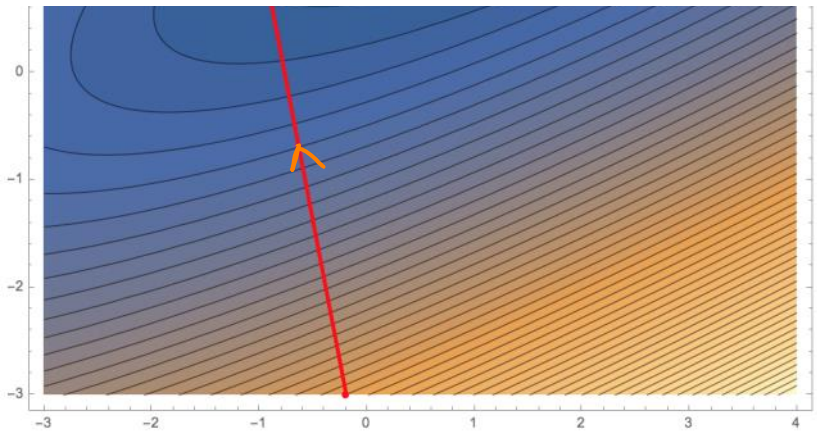
{3, 3}

```
x0 = {0, -4};
list = GradientDescent[A, b, x0, 1000, 10^-4]
```

{{0, -4}, {-1.0037, 1.24155}, {2.53695, 1.91955}, {2.38203, 2.72858}, {2.92853, 2.83323}, {2.90462, 2.95811}, {2.98897, 2.97426}, {2.98528, 2.99353}, {2.9983, 2.99603}, {2.99773, 2.999}, {2.99974, 2.99939}, {2.99965, 2.99985}}

```
"Display the results:";
iter = ListPlot[list, PlotRange → 3 {{-1, 1}, {-1, 1}}, Joined → True,
   PlotStyle → {Red, Thickness[.005]}, PlotMarkers → Automatic];
contour = ContourPlot[Norm[A.{x, y} - b]^2, {x, -3, 4}, {y, -3, 4}, Contours → 50];
Show[contour, iter]
```





Observe: Consecutive steps are orthogonal, $r_{t+1} \cdot r_t = 0$ (Exercise)

Observe: This finds an approximate solution.

Smaller condition number $\Rightarrow$ Rounder ellipse
$\Rightarrow$ Faster convergence

Remark: If A is not positive definite, the descent rule is
$\vec{x}_{t+1} = \vec{x}_t - \beta A^T \vec{r}_t$

Remark: Although not generally the fastest way of solving a set of linear equations, gradient descent is very robust, and **many** variants are used in **many** applications.

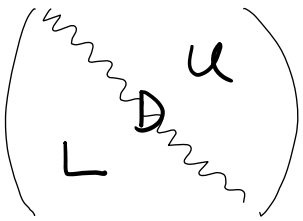
Convergence: $C_t \leq C_0 \cdot \left(1 - \frac{2}{K+1}\right)^{2t} \leq C_0 \cdot \exp\left(-\frac{4t}{K+1}\right)$

$\Rightarrow$ to get $\frac{C_t}{C_0} \leq 2^{-p}$, set $t = \frac{\log 2}{4}(K+1)p$ $O(K)$

Remark: A not pos. def. $\Rightarrow O(K^2)$ convergence

Conjugate gradient algorithm has $O(\sqrt{K})$ convergence

JACOBI and GAUSS-SEIDEL iterative methods




```
A = {{2, -1.5}, {-1.5, 4}};
L = LowerTriangularize[A, -1]
U = UpperTriangularize[A, 1]
Diag = DiagonalMatrix@Diagonal[A]
{{0., 0.}, {-1.5, 0.}}
{{0., -1.5}, {0., 0.}}
{{2, 0}, {0, 4}}
```

Jacobi iteration

$A\vec{x} = \vec{b}$

$\Rightarrow D\vec{x} = -(L+U)\vec{x} + \vec{b}$

Gauss-Seidel iteration

$\Rightarrow (D+L)\vec{x} = -U\vec{x} + \vec{b}$

$A\vec{x} = \vec{b}$

$\Rightarrow D\vec{x} = -(L+U)\vec{x} + \vec{b}$

$\vec{x}_{t+1} = -D^{-1}(L+U)\vec{x}_t + D^{-1}\vec{b}$

$\Rightarrow (D+L)\vec{x} = -U\vec{x} + \vec{b}$

$\vec{x}_{t+1} = -(D+L)^{-1}U\vec{x}_t + (D+L)^{-1}\vec{b}$

compute using back-subst. (don't compute $(D+L)^{-1}$)

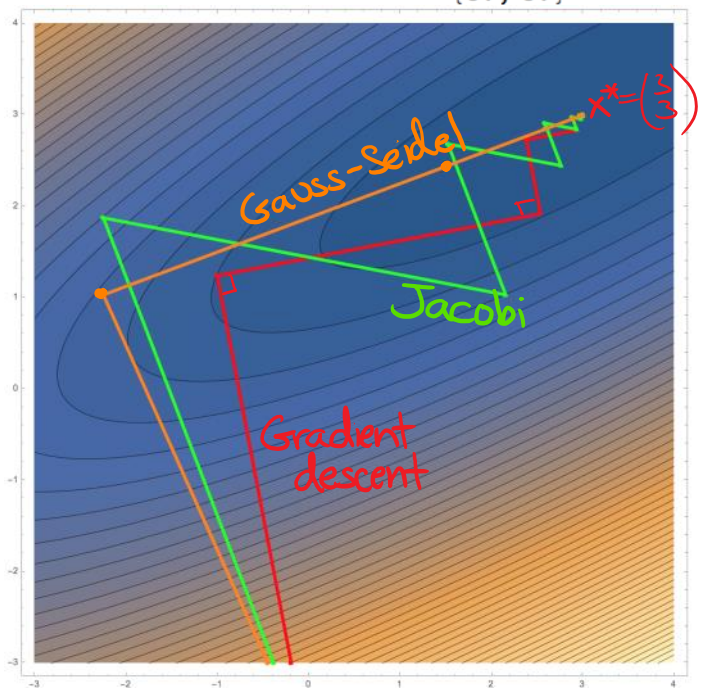
$\vec{x}_{t+1} = E\vec{x}_t + \vec{q}$

```
"Jacobi iteration";
e = -Inverse[Diag].(L + U);
q = Inverse[Diag].b;

x = {0, -4};
listJacobi = {x};
For[k = 1, k ≤ 1000, k++,
  x = e.x + q;
  AppendTo[listJacobi, x];
 ];
x
```

```
{3., 3.}
```

```
"Gauss-Seidel iteration";
e = -Inverse[Diag].(L + U);
q = LinearSolve[Diag + L, b];

x = {0, -4};
listGS = {x};
For[k = 1, k ≤ 1000, k++,
  x = -LinearSolve[Diag + L, U.x] + q;
  AppendTo[listGS, x];
 ];
x
```

```
{3., 3.}
```



PRECONDITIONING

Intuition: Iterative methods are slow when κ the condition # is large, ie., the ellipse is long and thin.

Let's round the ellipsoid.

$A\vec{x} = \vec{b} \iff M^{-1}A\vec{x} = M^{-1}\vec{b}$

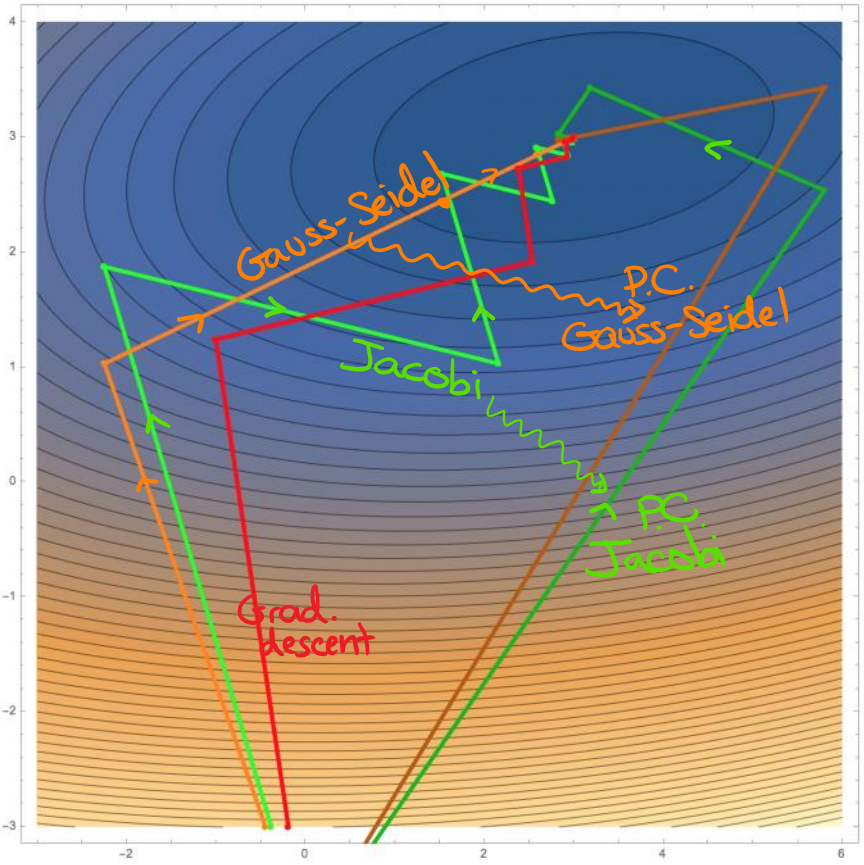
$M^{-1}A$ hopefully closer to I (rounder)

M should be
- "close to" A
- easy to solve $M\vec{x} = \vec{b}$

Example: Jacobi preconditioner

$M = D$ diagonal part of A

Example: $A = \begin{pmatrix} 2 & -1.5 \\ -1.5 & 4 \end{pmatrix}$ $M = \begin{pmatrix} 3 & -1.5 \\ -1.5 & 3 \end{pmatrix}$



Stopping criteria

When should you stop?

$$x - x^* = A^{-1}(Ax - b)$$
$$\Rightarrow \|x - x^*\| \le \|A^{-1}\| \cdot \|Ax - b\|$$

$$b = A \cdot A^{-1} b$$
$$\Rightarrow \|b\| \le \|A\| \cdot \|x^*\|$$

$$\Rightarrow \frac{\|x - x^*\|}{\|x^*\|} \le \underbrace{\|A^{-1}\| \cdot \|A\|}_{\kappa \text{ condition \#}} \cdot \frac{\|Ax - b\|}{\|b\|}$$

$\Rightarrow$ You are close to $\vec{x}^*$ if $\kappa \frac{\|Ax - b\|}{\|b\|}$ is small.

If you don't know κ, hopefully you can guess an upper bound.

Convergence analysis: for $A > 0$

$$\vec{x}_t = \vec{x}_{t-1} - \beta(A\vec{x}_{t-1} - \vec{b})$$

$$\Rightarrow \quad x_t - x^* = (x_{t-1} - x^*) - \beta A(x_{t-1} - x^*) \qquad \text{since } Ax^* = b$$

$$= (I - \beta A)(x_{t-1} - x^*)$$

$$\Rightarrow \|x_t - x^*\| \le \|I - \beta A\| \cdot \|x_{t-1} - x^*\|$$

$$\le \cdots$$

$$\le \|I - \beta A\|^t \cdot \|x_0 - x^*\|$$

initial error

If A's e-values are $\lambda_1 \ge \lambda_2 \ge \cdots \ge \lambda_n \ge 0$, then $I - \beta A$ has e-values

$$1 - \beta\lambda_1 \le 1 - \beta\lambda_2 \le \cdots \le 1 - \beta\lambda_n$$

$$\Rightarrow \|I - \beta A\| = \max\{|1 - \beta\lambda_1|, |1 - \beta\lambda_n|\}.$$

The best choice of β will make these equal in magnitude, centered on 0:

$$\Rightarrow -(1 - \beta\lambda_1) = +(1 - \beta\lambda_n)$$

$$\Rightarrow \beta = \frac{2}{\lambda_1 + \lambda_n}$$

With this choice for β,

$$\|I - \beta A\| = 1 - \frac{2\lambda_n}{\lambda_1 + \lambda_n} \le 1 - \frac{\lambda_n}{\lambda_1}$$

$\le 2\lambda_1$

$$= 1 - \frac{1}{(\text{condition \# of } A)}$$

$$\Rightarrow \|x_t - x^*\| \le \left(1 - \frac{1}{\kappa}\right)^t \cdot \|x_0 - x^*\|$$

$$\Rightarrow t = \kappa \cdot \left(\log\frac{1}{\varepsilon} + \log\frac{1}{\|x_0 - x^*\|}\right) \text{ ensures } \|x_t - x^*\| \le \varepsilon$$

using $\left(1 - \frac{1}{\kappa}\right)^\kappa \le \frac{1}{e}$

Convergence analysis: general A

$$x_t = x_{t-1} - 2\alpha A^T(Ax_{t-1} - b)$$

$$\Rightarrow x_t - x^* = (x_{t-1} - x^*) - 2\alpha A^T A(x_{t-1} - x^*)$$

$$= (I - 2\alpha A^T A)(x_{t-1} - x^*)$$

$\Rightarrow$ error magnitude

$$\|\vec{x}_t - \vec{x}^*\| \le \|I - 2\alpha A^TA\| \cdot \|\vec{x}_{t-1} - \vec{x}^*\|$$

If the e-values of A^TA are

$$\lambda_1 \ge \lambda_2 \ge \cdots \ge \lambda_n \ge 0$$

↑ these are (singular values of A)^2

$I - 2\alpha A^TA$ has e-values

$$1 - 2\alpha\lambda_1 \le \cdots \le 1 - 2\alpha\lambda_n$$

$$\Rightarrow \|I - 2\alpha A^TA\| = \max_i |1 - 2\alpha\lambda_i|$$

$$\max\{|1 - 2\alpha\lambda_1|, |1 - 2\alpha\lambda_n|\}$$

The best choice of α will make these equal in magnitude, centered on 0:

$$\Rightarrow -(1 - 2\alpha\lambda_1) = +(1 - 2\alpha\lambda_n)$$

$$\Rightarrow \alpha = \frac{1}{\lambda_1 + \lambda_n}$$

With this choice for α,

$$\|I - 2\alpha A^TA\| = 1 - \underbrace{\frac{2\lambda_n}{\lambda_1 + \lambda_n}}_{\le 2\lambda_1} \le 1 - \frac{\lambda_n}{\lambda_1}$$

$$= 1 - \frac{1}{\text{condition\# of } A^TA}$$

$$= 1 - \frac{1}{(\text{condition\# of } A)^2}$$

$$\Rightarrow \|x^t - x^*\| \le \left(1 - \frac{1}{\kappa^2}\right)^t \cdot \|x^0 - x^*\|$$

$$\Rightarrow t = \kappa^2\left(\log\frac{1}{\varepsilon} + \log\frac{1}{\|x^0 - x^*\|}\right) \quad \text{since } \left(1 - \frac{1}{\kappa^2}\right)^{\kappa^2} \le e^{-1}$$

ensures $\|x^t - x^*\| \le \varepsilon$.