Solving systems of equations

Storing an $m \times m$ matrix with $m = 100\,000$ requires $\approx 80GB$. And even if we managed to do it, applying an algorithm like Gaussian elimination, with complexity $\mathcal{O}(m^3)$, is prohibitive.

Luckily, many real-world matrices are sparse: for instance 3, 10 nonzeros per row.

This includes matrices from graph/networks, KKT systems, discretization of differential equations...
Some examples (from the Suitesparse Matrix Collection) in the next slide.

Some real-world matrices

- Adjacency matrices from networks and graphs.

```
% ''Friendship matrix'' on a group of 34 people
M = load('karate.mat').Problem.A;
% Road network of Luxembourg
M = load('luxembourg_osm.mat').Problem.A;
```

  For instance, in some applications centrality indices are computed by solving $(I - \alpha A)\mathbf{x} = \texttt{ones(n,1)}$.

- In both engineering and video game programming, one often models complex objects as "networks of points joined by forces", and then solves problems on them.

```
% From a structural stability problem (Boeing)
M = load('msc00726.mat').Problem.A;
```

- KKT matrices in optimization, $\begin{bmatrix} D_1 & A \\ A^T & 0 \end{bmatrix}$, often with D_1 diagonal and tall-thin A (possibly already sparse).

Storing and using a sparse matrix

Basic format to store sparse matrices: as list of non-zero (i, j, A_{ij}).

```
>> sprandn(10,10,0.3)
```

(Detail: if the indices j are listed increasingly, they can be compressed further. The most well-known format is known as CSC/CSR — compressed sparse column/row.)

We can operate on them directly in this format, e.g., in Python pseudocode:

```
"Compute the product w = A*v"
def compute_product(A, v):
        w = zeros(size(A, 1))
        for (i, j, Aij) in A:
                w[i] += Aij * v[j]
        return w
```

Optimization to solve linear systems

Given a $n \times n$ matrix $Q \succ 0$ and a vector $\mathbf{v} = -\mathbf{q} \in \mathbb{R}^n$, we wish to minimize

$$\min f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T Q\mathbf{x} - \mathbf{v}^T\mathbf{x} + const.$$

This is equivalent to solving $\mathbf{g} = Q\mathbf{x} - \mathbf{v} = 0$, i.e., the linear system $Q\mathbf{x} = \mathbf{v}$.

We shall see a particularly efficient algorithm that uses concepts from both linear algebra and optimization. It computes at each step the best (in a certain sense) possible approximation $\mathbf{x}_k$ to the solution $\mathbf{x}_*$.

It is particularly suited to large problem with sparse matrices.

Intro to conjugate gradient

Let us start from a simple quadratic problem with $Q = I$:

$$\min_{\mathbf{y}\in\mathbb{R}^m} \frac{1}{2}\|\mathbf{y}-\mathbf{w}\|^2 + const = \min \frac{1}{2}\mathbf{y}^T\mathbf{y} - \mathbf{w}^\top\mathbf{y} + const$$

$$= \min \frac{1}{2}(y_1^2 + y_2^2 + \cdots + y_m^2)$$

$$- (w_1y_1 + w_2y_2 + \cdots + w_my_m) + const$$

This problem is separable: starting from $\mathbf{y}_0 = \mathbf{0}$, we optimize on each coordinate separately and generate the sequence of vectors

$$\mathbf{y}_1 = \begin{bmatrix} w_1 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad \mathbf{y}_2 = \begin{bmatrix} w_1 \\ w_2 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad \mathbf{y}_3 = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad \cdots$$

At each step, we add a multiple of a new search direction $\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3, \ldots$. They are all orthogonal to each other.

Convergence guaranteed after m iterations.

Subspace optimality

At each step, we solve a 1D problem and choose $\mathbf{y}_k$ to solve

$$\mathbf{y}_k = \arg\min f(\mathbf{y}) \text{ over } \begin{bmatrix} w_1 \\ \vdots \\ w_{k-1} \\ * \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \{\mathbf{y}_{k-1} + \alpha \mathbf{e}_k : \alpha \in \mathbb{R}\},$$

(line search), but we also get for free a stronger property:

$$\mathbf{y}_k = \arg\min f(\mathbf{y}) \text{ over } \begin{bmatrix} * \\ \vdots \\ * \\ * \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \text{span}(\mathbf{e}_1, \ldots, \mathbf{e}_k).$$

Orthogonal directions

We can proceed similarly with any set of orthogonal search directions $U = [\mathbf{u}_1, \mathbf{u_2}, \ldots, \mathbf{u}_m]$ instead of the canonical basis $\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_m$. Write

$$\mathbf{w} = U \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_m \end{bmatrix}, \quad \|\mathbf{w}\| = \|\mathbf{c}\|$$

and find

$$\mathbf{y}_k = \min f(\mathbf{y}) \text{ over } U \begin{bmatrix} c_1 \\ \vdots \\ c_{k-1} \\ * \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \{\mathbf{y}_{k-1} + \alpha \mathbf{u}_k : \alpha \in \mathbb{R}\},$$

$$= \min f(\mathbf{y}) \text{ over } U \begin{bmatrix} * \\ \vdots \\ * \\ * \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \text{span}(\mathbf{u}_1, \ldots, \mathbf{u}_m).$$

The algorithm

Given orthogonal search dirs $\mathbf{u}_1, \ldots, \mathbf{u}_m$ (i.e., $\mathbf{u}_i^T \mathbf{u}_j = 0$ when $i \neq j$)

$\mathbf{y}_0 \leftarrow \mathbf{0}$;
for $k = 1, 2, 3, \ldots, m$ **do**
  $\mathbf{y}_k \leftarrow \arg\min \|\mathbf{y} - \mathbf{w}\|^2 + \mathit{const}$ over $\{\mathbf{y}_{k-1} + \alpha \mathbf{u}_k\}$;
  `// univariate quadratic problem in` α
end

Change of variable

This simple problem is actually equivalent to any quadratic problem via a change of basis: given $R \in \mathbb{R}^{m \times m}$ invertible, $\mathbf{y} = R\mathbf{x}$,

$$\min \frac{1}{2}\mathbf{y}^T\mathbf{y} - \mathbf{w}^T\mathbf{y} + const = \min \frac{1}{2}\mathbf{x}^T \underbrace{R^T R}_{=Q}\mathbf{x} - \underbrace{\mathbf{w}^T R}_{=\mathbf{v}^T}\mathbf{x} + const.$$

We can solve the (difficult) problem on the **x**-space by looking at the (easier) one on the **y**-space, with coordinate descent.

Important detail: in the old problem, $\mathbf{w}$ is both the linear term appearing in the objective function and the solution $\mathbf{y}_* = \mathbf{w}$; in the new problem, $\mathbf{v} = R^T\mathbf{w}$, but $\mathbf{x}_* = Q^{-1}\mathbf{v} = R^{-1}\mathbf{w}$: indeed we can rewrite the objective function as

$$\min_{\mathbf{y}} \frac{1}{2}\|\mathbf{y}-\mathbf{w}\|^2 + C = \min_{\mathbf{x}} \frac{1}{2}\|R(\mathbf{x}-\mathbf{x}_*)\|^2 + C = \min_{\mathbf{x}} \frac{1}{2}(\mathbf{x}-\mathbf{x}_*)^T R^T R(\mathbf{x}-\mathbf{x}_*)$$

Definition: the Q-norm of a vector $\mathbf{z}$ is $\|\mathbf{z}\|_Q = (\mathbf{z}^T Q\mathbf{z})^{1/2}$.

Since $Q \succ 0$, it is still true that $\|\mathbf{z}\|_Q \geq 0$, with equality iff $\mathbf{z} = \mathbf{0}$.

Q-orthogonality

Search directions: $R\mathbf{d}_k = \mathbf{u}_k$. These are orthogonal in the $\mathbf{y}$-space ($\mathbf{u}_i^T \mathbf{u}_j = 0$ when $i \neq j$), but in the $\mathbf{y}$-space the relation becomes

$$\mathbf{d}_j^T \underbrace{R^T R}_{=Q} \mathbf{d}_i = 0$$

Definition: vectors $\mathbf{d}_i, \mathbf{d}_j$ are called Q-orthogonal if $\mathbf{d}_j^T Q \mathbf{d}_j = 0$.

The algorithms

In the y space:

Given orthogonal search dirs $\mathbf{u}_1, \ldots, \mathbf{u}_m$ (i.e., $\mathbf{u}_i^T \mathbf{u}_j = 0$ when $i \neq j$)

$\mathbf{y}_0 \leftarrow \mathbf{0}$;
for $k = 1, 2, 3, \ldots, m$ **do**
  $\mathbf{y}_k \leftarrow \arg\min \|\mathbf{y} - \mathbf{w}\|^2 + const$ over $\{\mathbf{y}_{k-1} + \alpha \mathbf{u}_k\}$;
  `// univariate quadratic problem in` α
end

In the x space:

Given Q-orthogonal search dirs $\mathbf{d}_1, \ldots, \mathbf{d}_m$ (i.e., $\mathbf{d}_i^T Q \mathbf{d}_j = 0$ when $i \neq j$)

$\mathbf{x}_0 \leftarrow \mathbf{0}$;
for $k = 1, 2, 3, \ldots, m$ **do**
  $\mathbf{x}_k \leftarrow \arg\min \mathbf{x}^T Q \mathbf{x} + \mathbf{v}^T \mathbf{x} + const$ over $\{\mathbf{x}_{k-1} + \alpha \mathbf{d}_k\}$;
  `// univariate quadratic problem in` α
end

Details

- We do not need to know R, nor $\mathbf{x}_*$, nor *const*: it is enough to have Q and $\mathbf{v}$!
- Subspace optimality: $\mathbf{x}_k = \min f(\mathbf{x})$ for $\mathbf{x} \in \text{span}(\mathbf{d}_1, \ldots, \mathbf{d}_k)$.
- Convergence guaranteed in m steps — but we hope to do better!
- Important missing part: how to choose the $\mathbf{d}_i$'s?
  Optimization suggests: it should be loosely in the direction of the residual $\mathbf{r}_j = -\mathbf{g}_j = \mathbf{v} - Q\mathbf{x}_j$. But residuals are noth Q-orthogonal.
  We shall see that a special property holds: if we set $\mathbf{d}_j = \mathbf{r}_j + \beta_j \mathbf{d}_{j-1}$, it is sufficient to choose β_j to impose $\mathbf{d}_{j-1}^T Q \mathbf{d}_j = 0$; Q-orthogonality with all previous search directions holds automatically.

Conjugate gradient — implementation

Three ingredients: current iterate $\mathbf{x}_j$, residual $\mathbf{r}_j = \mathbf{v} - Q\mathbf{x}_j = -\mathbf{g}_j$, and search direction $\mathbf{d}_j$.

CG iteration

$\mathbf{x}_0 = \mathbf{0}, \mathbf{r}_0 = \mathbf{d}_0 = \mathbf{v}$;
for $j = 1{:}n$ **do**
  $\alpha_j = (\mathbf{r}_{j-1}^T \mathbf{r}_{j-1})/(\mathbf{d}_{j-1}^T Q \mathbf{d}_{j-1})$; `// exact line search`
  $\mathbf{x}_j = \mathbf{x}_{j-1} + \alpha_j \mathbf{d}_{j-1}$;
  $\mathbf{r}_j = \mathbf{r}_{j-1} - \alpha_j Q \mathbf{d}_{j-1}$; `// residual update (check!)`
  $\beta_j = (\mathbf{r}_j^T \mathbf{r}_j)/(\mathbf{r}_{j-1}^T \mathbf{r}_{j-1})$;
  $\mathbf{d}_j = \mathbf{r}_j + \beta_j \mathbf{d}_{j-1}$; `// Q-orthogonal (we'll see why)`
end

The formula for the exact line search α_j is not obvious, but we will have the tools to prove it later.

Storage: 3 vectors: $\mathbf{x}_j, \mathbf{r}_j, \mathbf{d}_j$. No need to keep previous iterates.

Black-box algorithms

Cost: $n \times$ (1 mat-vec product for $Q\mathbf{d}_{j-1} + \mathcal{O}(m)$).

Dominant part: computing n products $\mathbf{d}_j \mapsto Q\mathbf{d}_j$.

Note that we only need a function ("oracle" in CS terms) `compute_product(d) = Q*d`: this is a so-called black box algorithm.

CG is fast whenever `compute_product` is fast: a sparse Q yields $\mathcal{O}(\text{nnz}(Q))$, but not only.

Krylov spaces

A new linear algebra concept that will help us analyze CG.

Definition

Given $Q \in \mathbb{R}^{m \times m}$ (not nec. symmetric), $\mathbf{v} \in \mathbb{R}^m$, and $n \leq m$, the Krylov space $K_n(Q, \mathbf{v})$ is the linear subspace

$$K_n(Q, \mathbf{v}) = \text{span}(\mathbf{v}, Q\mathbf{v}, Q^2\mathbf{v}, \ldots, Q^{n-1}\mathbf{v}),$$

That is, the set of vectors that we can write as

$$\mathbf{w} = \underbrace{(c_0 I + c_1 Q + c_2 Q^2 + \cdots + c_{n-1} Q^{n-1})}_{:= p(Q)} \mathbf{v};$$

any polynomial of degree $d < n$ in Q, multiplied by $\mathbf{v}$.

Krylov spaces, polynomials and degrees

Assume $\mathbf{v}, Q\mathbf{v}, Q^2\mathbf{v}, \ldots, Q^{n-1}\mathbf{v}$ are linearly independent; then the coordinates

$$\mathbf{w} = \mathbf{v}c_0 + Q\mathbf{v}c_1 + \ldots Q^2\mathbf{v}c_2 + \cdots + Q^{n-1}\mathbf{v}c_{n-1}$$

of any vector $\mathbf{w} \in K_n(Q, \mathbf{v})$ are unique. For each $\mathbf{w}$ the degree d of the polynomial such that $\mathbf{w} = p(Q)\mathbf{v}$ is well-defined.

Trivial but useful facts

- If $\mathbf{w}$ has degree d, then $\mathbf{w} \in K_{d+1}(Q, \mathbf{v}) \setminus K_d(Q, \mathbf{v})$.
- If $\mathbf{w}$ has degree d, then $Q\mathbf{w}$ has degree $d + 1$.

Krylov spaces — characterization

Observation $K_n(Q, \mathbf{v})$ is the set of vectors that I can obtain, starting from $S = \{\mathbf{v}\}$, with these operations:

- Multiply by Q: add to the set $Q\mathbf{w}$, where $\mathbf{w}$ is any element of S;
- Linear combinations: add to the set $\mathbf{w}_1\alpha_1 + \cdots + \mathbf{w}_k\alpha_k$, where the $\mathbf{w}_i$ belong to S;

and the first operation is performed fewer than n times.

This matches well our "oracle" idea: the allowed operations are linear combinations and invoking the oracle; $K_n(Q, \mathbf{v})$ is the set of vectors that I can obtain by calling the oracle fewer than n times.

Krylov spaces and optimization

Observation The iterates of gradient descent lie in Krylov spaces.

Suppose we are looking for

$$\min f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T Q\mathbf{x} - \mathbf{v}^T x + const, \quad \mathbf{x}_0 = \mathbf{0}.$$

At each step we take a gradient $\mathbf{g}_k := Q\mathbf{x}_k - \mathbf{v}$ and use it to compute $\mathbf{x}_{k+1}$.

$\mathbf{x}_0 = \mathbf{0}$

$\mathbf{x}_1 = \mathbf{x}_0 - (Q\mathbf{x}_0 - \mathbf{v})\alpha_1 = \mathbf{v}\alpha_1,$

$\mathbf{x}_2 = \mathbf{x}_1 - (Q\mathbf{x}_1 - \mathbf{v})\alpha_2 = \mathbf{v}\alpha_1 - (Q\mathbf{v}\alpha_1 - \mathbf{v})\alpha_2 \in \mathrm{span}(\mathbf{v}, Q\mathbf{v})$

$\mathbf{x}_3 = \mathbf{x}_2 - (Q\mathbf{x}_2 - \mathbf{v})\alpha_3 \in \mathrm{span}(\mathbf{v}, Q\mathbf{v}, Q^2\mathbf{v})$

We have

$\mathbf{g}_0, \mathbf{x}_1 \in K_1(Q, \mathbf{v})$, 0 products with Q required

$\mathbf{g}_1, \mathbf{x}_2 \in K_2(Q, \mathbf{v}) \setminus K_1(Q, \mathbf{v})$, 1 product with Q required

$\mathbf{g}_2, \mathbf{x}_3 \in K_3(Q, \mathbf{v}) \setminus K_2(Q, \mathbf{v})$, 2 products with Q required

.

Search space = Krylov space

Theorem

Assume that $\mathbf{v}, Q\mathbf{v}, \ldots, Q^{n-1}\mathbf{v}$ are linearly independent. Then, after each step n of CG (starting from $\mathbf{x}_0 = 0$),

- $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$
- $\mathbf{r}_0, \mathbf{r}_1, \ldots, \mathbf{r}_{n-1}$
- $\mathbf{d}_0, \mathbf{d}_1, \ldots, \mathbf{d}_{n-1}$.

are bases of $K_n(Q, \mathbf{v})$.

Proof

1. Using the formulas that define the method, show inductively that $\mathbf{x}_j, \mathbf{r}_{j-1}, \mathbf{d}_{j-1}$ have degree $j - 1$.
2. Observe that if we have a polynomial $p_0(t)$ of degree 0, one $p_1(t)$ of degree 1, ..., one $p_{n-1}(t)$ of degree $n - 1$, then we can write any polynomial of degree $\leq n - 1$ as a linear combination of them.

Orthogonality in CG

Theorem

At each step $\mathbf{r}_i^T \mathbf{r}_j = \mathbf{d}_i^T Q \mathbf{d}_j = 0$ for all $i < j$.

The $\mathbf{r}_i$ are orthogonal (not -normal), and the $\mathbf{d}_i$ are Q-orthogonal.

Proof (sketch) Assume it holds for $j-1$ (induction!). We show only that $\mathbf{r}_i^T \mathbf{r}_j = 0$ for all $i < j$; the other part is similar.

From $\mathbf{r}_j = \mathbf{r}_{j-1} - \alpha Q \mathbf{d}_{j-1}$ it follows that

$$\mathbf{r}_i^T \mathbf{r}_j = \mathbf{r}_i^T \mathbf{r}_{j-1} - \alpha_j \mathbf{r}_i^T Q \mathbf{d}_{j-1}.$$

- For $i < j-1$, $\mathbf{r}_i^T \mathbf{r}_{j-1}$ is zero by induction, since $\mathbf{r}_i \in \text{span}(\mathbf{d}_0, \mathbf{d}_1, \ldots, \mathbf{d}_i)$ is Q-orthogonal to $\mathbf{d}_{j-1}$.
- For $i = j-1$, the RHS is zero if we can prove that $\alpha_j = \frac{\mathbf{r}_{j-1}^T \mathbf{r}_{j-1}}{\mathbf{r}_{j-1}^T Q \mathbf{d}_{j-1}}$. This is *almost* the formula for α_j, but the denominator is not $\mathbf{d}_{j-1}^T Q \mathbf{d}_{j-1}$. However $\mathbf{d}_{j-1} = \mathbf{r}_{j-1} + \beta_{j-1} \mathbf{d}_{j-2}$, so the two denominators differ by $\beta_{j-1} \mathbf{d}_{j-2}^T Q \mathbf{d}_{j-1} = 0$ (by induction).

The other half of the proof

(Not shown, just here for completeness.)
It remains to prove the second half of the induction step, i.e.,

$$0 = \mathbf{d}_i^T Q \mathbf{d}_j = \mathbf{d}_i^T Q(\mathbf{r}_j + \beta_j \mathbf{d}_{j-1}).$$

For $i < j - 1$, this follows by induction and the fact that $Q\mathbf{d}_i \in K_{j-1}(Q, \mathbf{v})$ is orthogonal to $\mathbf{r}_j$ by the first half of the proof.
For $i = j - 1$, this holds if we can prove that

$$\beta_j = -\frac{\mathbf{d}_{j-1}^T Q \mathbf{r}_j}{\mathbf{d}_{j-1}^T Q \mathbf{d}_{j-1}} = \frac{\mathbf{r}_j^T(-\alpha Q \mathbf{d}_{j-1})}{\mathbf{d}_{j-1}^T(\alpha Q \mathbf{d}_{j-1})} = \frac{\mathbf{r}_j^T(\mathbf{r}_j - \mathbf{r}_{j-1})}{\mathbf{d}_{j-1}^T(\mathbf{r}_{j-1} - \mathbf{r}_j)}.$$

This quantity is equal to the formula for β_j because $\mathbf{d}_{j-1}, \mathbf{r}_{j-1} \in K_j(Q, \mathbf{v})$ are orthogonal to $\mathbf{r}_j$, and $\mathbf{d}_{j-1}^T \mathbf{r}_{j-1} = (\mathbf{r}_{j-1} + \beta_{j-1}\mathbf{d}_{j-2})^T \mathbf{r}_{j-1} = \mathbf{r}_{j-1}^T \mathbf{r}_{j-1}$.

Lucky breakdown

Breakdown $\Longrightarrow$ solution: Suppose that for a certain n the vectors $\mathbf{v}, Q\mathbf{v}, \ldots Q^n\mathbf{v}$ are linearly dependent, i.e., $Q^n\mathbf{v}$ can be written as a linear combination of the previous ones, i.e.,

$$K_n(Q, v) = K_{n-1}(Q, \mathbf{v}).$$

In particular, since $\mathbf{r}_n \in K_n(Q, v)$, we have

$$\mathbf{r}_n = c_0\mathbf{r}_0 + c_1\mathbf{r}_1 + \cdots + c_{n-1}\mathbf{r}_{n-1}.$$

But we can still use the steps of our proof to show that $\mathbf{r}_n^T\mathbf{r}_j = 0$ for $j < n$.
Then, we must have $\|\mathbf{r}_n\|^2 = \mathbf{r}_n^T\mathbf{r}_n = 0$, by orthogonality.

Convergence of CG

Geometric idea: the level curves are ellipsoids in the $\mathbf{x}$-space but circles in the $\mathbf{y}$-space.

Convergence is guaranteed in at most m iterations, $\mathbf{x}_m = \mathbf{x}_*$, but it can be much faster. For instance, when $Q = I$ we converge in 1 step.

Optimality $\implies$ $\|\mathbf{x}_k - \mathbf{x}_*\|_Q$ and $f(\mathbf{x}_k)$ decrease monotonically. However, $\|\mathbf{x}_k - \mathbf{x}_*\|$ or $\|\mathbf{r}_k\|$ do *not*, in general.

Optimality $\implies$ $\|\mathbf{x}_k - \mathbf{x}_*\|_Q$ and $f(\mathbf{x}_k)$ decrease faster than any other method that produces $\mathbf{x}_n \in K(Q, \mathbf{v})$. E.g., gradient method, heavy ball variants, ...

Convergence speed of CG

The convergence speed depends on the effectiveness of polynomial approximation of the eigenvalues of Q.

Theorem

$$\frac{\|\mathbf{x}_n - \mathbf{x}_*\|_Q}{\|\mathbf{x}_0 - \mathbf{x}_*\|_Q} \leq \min_{r(t)} \max_{i=1,2,\ldots,m} |r(\lambda_i)|,$$

where $\lambda_1, \ldots, \lambda_m$ are the eigenvalues of Q, and the minimum is over all polynomials r of degree $\leq n$, normalized such that $r(0) = 1$.

Proof

$\mathbf{x}_n \in K_n(Q, \mathbf{v}) \iff \mathbf{x}_n = p(Q)\mathbf{v}$ for a polynomial p of degree $< n$.

$$\begin{aligned}\|\mathbf{x}_n - \mathbf{x}_*\|_Q = \min_{\mathbf{x} \in K_n(Q,\mathbf{v})} \|\mathbf{x} - \mathbf{x}_*\|_Q &= \min_{p(t)} \|\mathbf{x}_* - p(Q)Q\mathbf{x}_*\|_Q \\ &= \min_{\substack{r(t)=1-tp(t) \\ \text{of degree} \leq n}} \|r(Q)\mathbf{x}_*\|_Q.\end{aligned}$$

Convergence speed of CG (cont.)

We can use the formulas from our slideset on orthogonality to give better expressions in terms of an eigenvalue decomposition $Q = UDU^T$, with U orthogonal and D diagonal containing the eigenvalues:

$$r(Q) = U \begin{bmatrix} r(\lambda_1) & & & \\ & r(\lambda_2) & & \\ & & \ddots & \\ & & & r(\lambda_m) \end{bmatrix} U^T.$$

Moreover, if $\mathbf{x}_* = U\mathbf{c}$, then $\|\mathbf{x}_*\|_Q^2 = \sum_i \lambda_i c_i^2$, and

$$\|r(Q)\mathbf{x}_*\|_Q^2 = \left\| U \begin{bmatrix} r(\lambda_1) & & & \\ & r(\lambda_2) & & \\ & & \ddots & \\ & & & r(\lambda_m) \end{bmatrix} \mathbf{c} \right\|_Q^2 = \sum_i \lambda_i r(\lambda_i)^2 c_i^2.$$

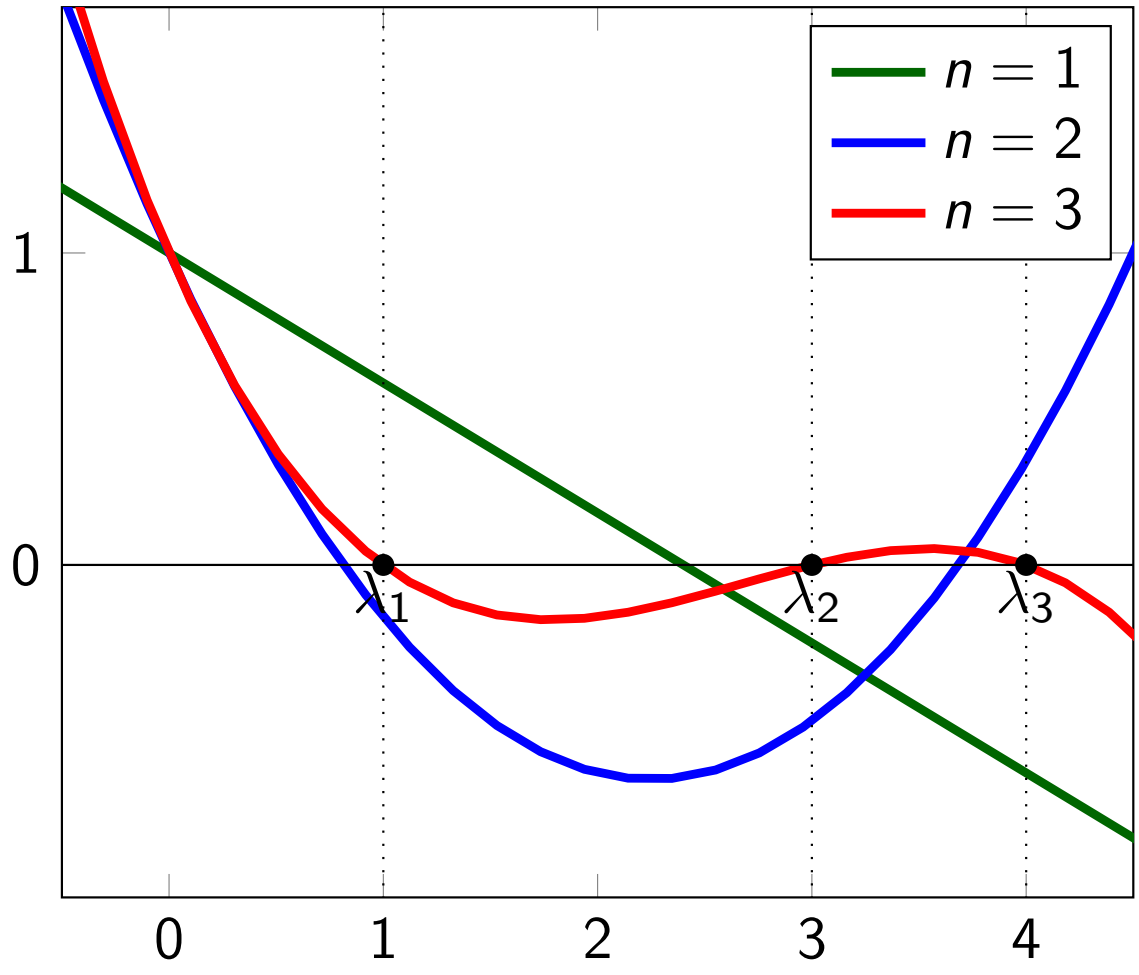
Convergence speed of CG (cont.)

From these two expressions it follows that

$$\begin{aligned}\frac{\|\mathbf{x}_n - \mathbf{x}_*\|_Q^2}{\|\mathbf{x}_0 - \mathbf{x}_*\|_Q^2} &= \frac{\|r(Q)\mathbf{x}_*\|_Q^2}{\|-\mathbf{x}_*\|_Q^2} \\ &= \frac{\lambda_1 r(\lambda_1)^2 c_1^2 + \cdots + \lambda_m r(\lambda_m)^2 c_m^2}{\lambda_1 c_1^2 + \cdots + \lambda_m c_m^2} \\ &\leq \max_{\lambda_1,\ldots,\lambda_m} r(\lambda_i)^2.\end{aligned}$$

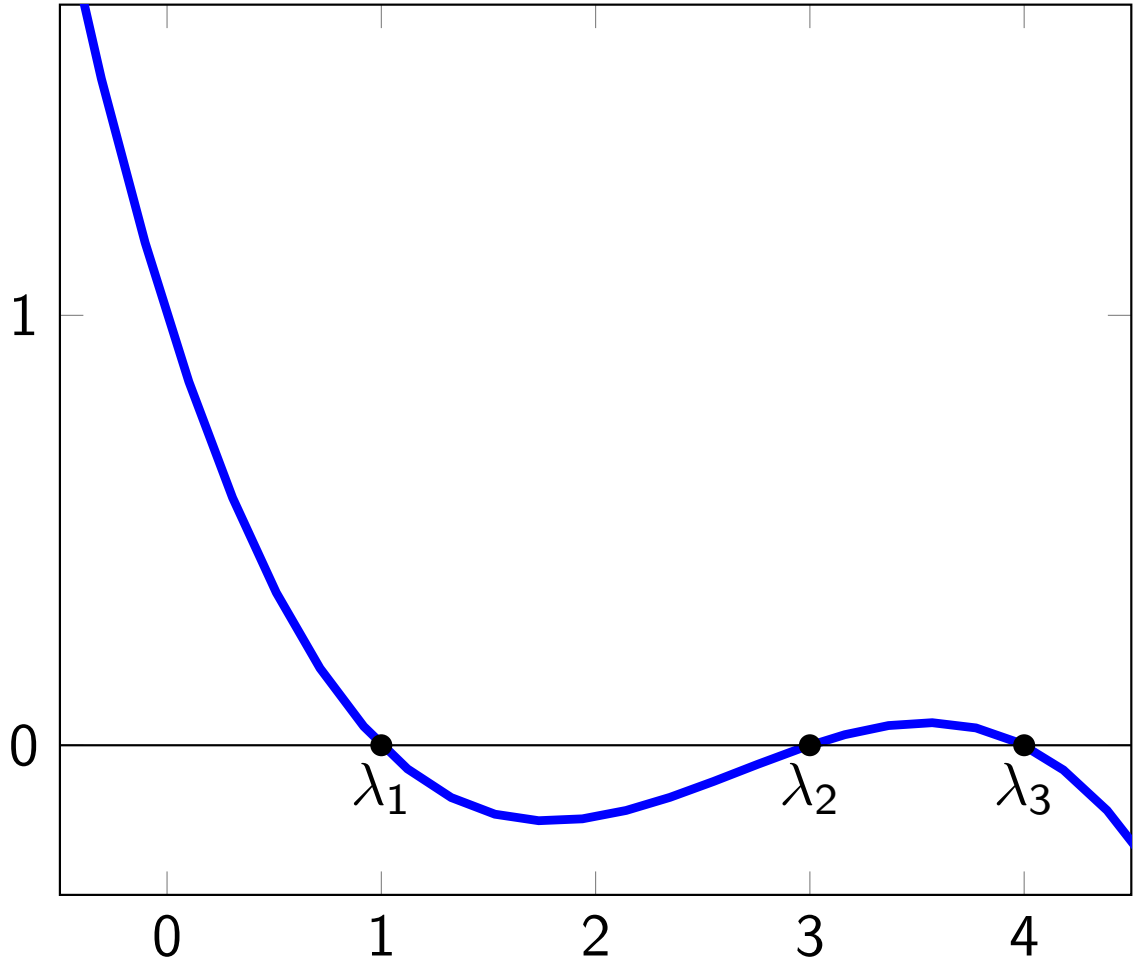
CG finds the best polynomial

CG converges as well as the best possible polynomial $r(t)$; and we don't even need to compute it explicitly!



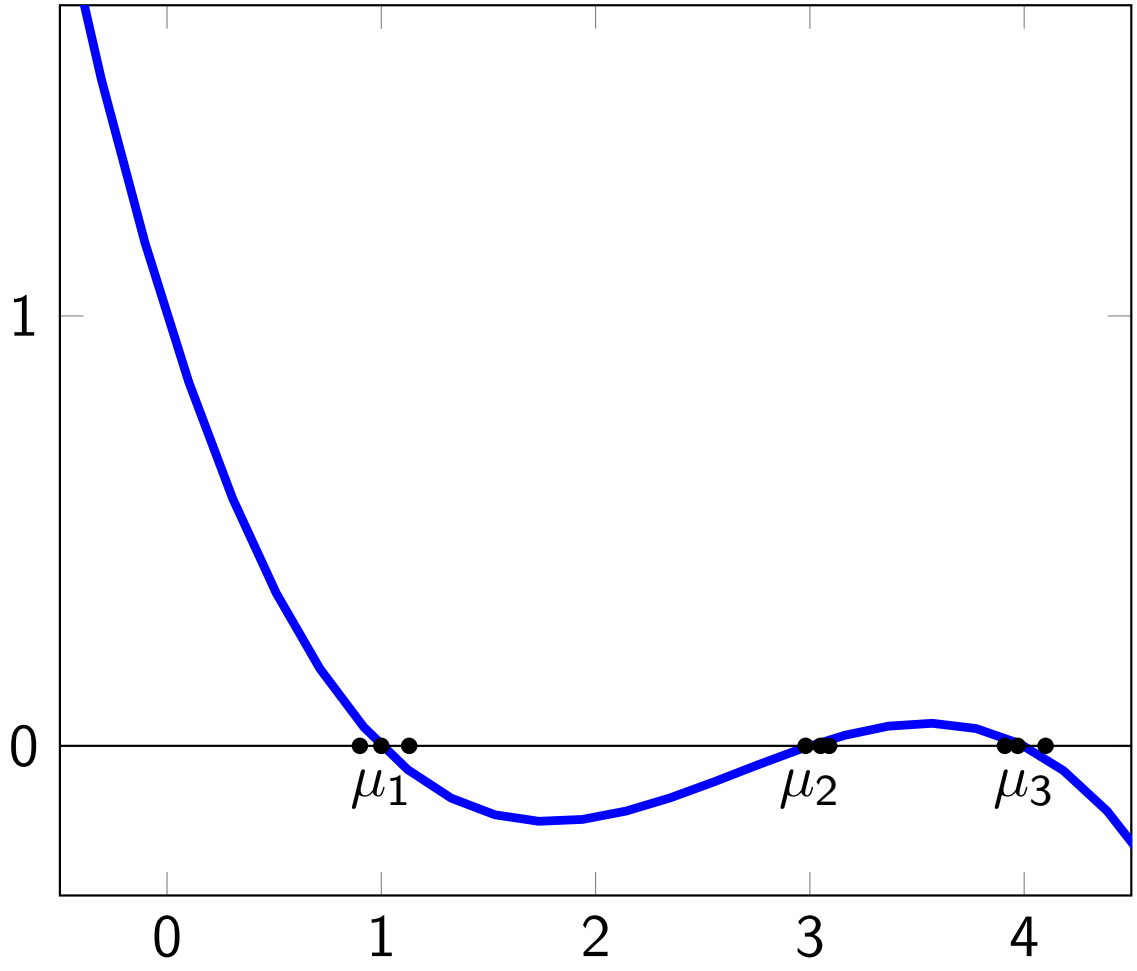
Repeated eigenvalues

If $Q \in \mathbb{R}^{m \times m}$ has only $n < m$ distinct eigenvalues, then we can find $r(t)$ such that $r(\lambda_i) = 0, r(0) = 1$, by interpolation $\implies$ $\|\mathbf{x}_n - \mathbf{x}_*\|_Q = 0$. CG finds the exact solution in n steps!



Clustered eigenvalues

Similar case: if the eigenvalues of Q are clustered around n values $\mu_1, \ldots, \mu_n$, then the interpolation polynomial r on the μ_i is likely to have small $|r(\lambda_i)|$ for all $i \implies$ small residual after n steps.



Linear convergence

Theorem (linear convergence)

Let $\lambda_{\max}$, $\lambda_{\min}$ be the maximum/minimum eigenvalue of Q; then, CG converges with rate

$$\frac{\|\mathbf{x}_n - \mathbf{x}_*\|_Q}{\|\mathbf{x}_0 - \mathbf{x}_*\|_Q} \leq 2\left(\frac{\sqrt{\lambda_{\max}} - \sqrt{\lambda_{\min}}}{\sqrt{\lambda_{\max}} + \sqrt{\lambda_{\min}}}\right)^n.$$

(Proof: find a polynomial such that $\max_{\lambda \in [\lambda_{\min}, \lambda_{\max}]} |r(\lambda)| = RHS$.)

We can rewrite that constant in terms of $\kappa(Q) = \frac{\lambda_{\max}}{\lambda_{\min}}$, the condition number of Q (this definition is valid only for $Q \succ 0$!). This quantity measures how "imbalanced" the eigenvalues of Q are.

$$\frac{\sqrt{\lambda_{\max}} - \sqrt{\lambda_{\min}}}{\sqrt{\lambda_{\max}} + \sqrt{\lambda_{\min}}} = \frac{\sqrt{\kappa(Q)} - 1}{\sqrt{\kappa(Q)} + 1}.$$

This is a faster rate than that of the gradient method, $\frac{\kappa - 1}{\kappa + 1}$.

Matlab examples

```
>> rng(0);
>> A = randn(5); Q = A'*A;
>> v = randn(5, 1);
>> x = pcg(Q, v);
pcg converged at iteration 5 to a solution
with relative residual 4e-13.
>> norm(Q*x-v) / norm(v)
ans =
   4.0223e-13
>> [x, ~, ~, ~, resvec] = pcg(Q, v); semilogy(resvec)
% sudden convergence at the last iteration
% this is normal for small-scale matrices
% CG really shines with large, sparse matrices
```

```
>> rng(0); n = 1000;
>> A = sprandsym(n, 5/n); % 5 nonzeros/row on avg
>> min(eig(A)) % A is symmetric but not posdef
>> Q = 10*speye(n) + A; % to get a posdef matrix
>> spy(Q)
>> min(eig(Q)), max(eig(Q))
ans =
    3.9943
ans =
   16.0227
>> v = randn(n, 1);
>> pcg(Q, v, 1e-18, 200);
Warning: Input tol may not be achievable by PCG
         Try to use a bigger tolerance
> In pcg (line 90)
pcg converged at iteration 33 to a solution
with relative residual 2e-16.
```

```
>> Q = 6.0058*speye(n) + A; % more ill-conditioned
>> min(eig(Q)), max(eig(Q))
ans =
   1.0563e-04
ans =
   12.0285
>> >> pcg(Q, v, 1e-18, 200); % slower convergence
Warning: Input tol may not be achievable by PCG
         Try to use a bigger tolerance
> In pcg (line 90)
pcg stopped at iteration 86 without converging to
the desired tolerance 2.2e-16
because the method stagnated.
The iterate returned (number 82) has relative
residual 7.6e-14.
```

```
>> A = bucky(); spy(A) % 60x60 test matrix, repeated eigs
>> Q = 2.6181 * speye(size(A)) + A;
>> v = randn(size(A,1), 1);
>> [x, ~, ~, ~, resvec] = pcg(Q, v); % exact convergence
>> semilogy(resvec);
```