MA3227 Numerical Analysis II

Lecture 5: Krylov Subspace Methods

Simon Etter

Semester II, AY 2020/2021

Krylov Subspace Methods

Problem statement

Approximately solve $Ax = b$ by setting
$x \approx p(A)\, b$ for some polynomial $p(x)$.

The matrix polynomial $p(A)$ in the above statement is defined as follows.

Def: Matrix polynomials

Given a matrix A and a polynomial $p(x) = \sum_{k=0}^{n} c_k\, x^k$, we define

$$p(A) = \sum_{k=0}^{n} c_k\, A^k$$

where A^k denotes the usual matrix power given by $A^0 = I$, $A^{k+1} = A\, A^k$.

Terminology: Krylov subspace methods

Linear system solvers of the above form are known as *Krylov subspace methods* for reasons which I shall explain later.

Krylov Subspace Methods

Why polynomial approximation?

The above problem statement may seem peculiar at first, but it is actually based on an idea which occurs throughout numerical analysis:

If you have a complicated function $f(x)$ which you do not know how to evaluate (efficiently), then try replacing $f(x)$ with a polynomial $p(x)$.

Polynomials can be evaluated using only addition and multiplication; hence computing $p(x)$ is often very simple and fast.

The "complicated function" that we aim to evaluate in this lecture is

$$(A, b) \mapsto A^{-1}b.$$

We have seen in Lecture 4 that we could evaluate this function using an LU factorisation, but the runtime of doing so is often fairly large, and in particular larger than $O(\text{nnz}(A))$ due to fill-in.

This problem disappears as promised once we replace A^{-1} with a polynomial approximation $p(A)$.

Krylov Subspace Methods

Thm: Runtime of $p(A)\,b$

$p(A)\,b$ can be evaluated in $O\big(\text{degree}(p)\,\text{nnz}(A)\big)$ operations.

Proof. Only three operations are required to evaluate $p(A)\,b$, namely

- the matrix-vector product $(A, v) \mapsto Av$,
- the scalar-vector product $(a, v) \mapsto av$, and
- the vector sum $(w, v) \mapsto w + v$.

Given these three operations, we can evaluate $(A^k\,b)_{k=0}^{n}$ through recursive application of the matrix-vector product,

$$A^0\,b = b, \qquad (A^{k+1}\,b) = A\,(A^k\,b),$$

and then we can use the scalar-vector product and vector sum to evaluate

$$p(A)\,b = \sum_{k=0}^{\deg(p)} c_k\,(A^k b).$$

These formulae require $O(\deg(p))$ executions of each of the three basic operations. It is therefore enough to show that each basic operation requires $O(\text{nnz}(A))$ runtime to verify the claim.

Krylov Subspace Methods

Proof (continued).

We therefore observe:

- A single matrix-vector product can be evaluated in $O(\text{nnz}(A))$ operations using the following algorithm.

Algorithm Sparse matrix-vector product $w = Av$

1: Initialise $w = 0$
2: **for** (i, j) such that $A[i, j] \neq 0$ **do**
3: $\quad w[i] = w[i] + A[i, j]\, v[j]$
4: **end for**

- Scalar-vector products and vector sums can be evaluated in $O\big(\text{length}(b)\big)$ operations, and we must have $\text{length}(b) \leq \text{nnz}(A)$ since otherwise A would not be invertible.

Krylov Subspace Methods

The above proof shows that evaluating $p(A)\,b$ is simple and fast once the polynomial $p(x)$ has been determined. It therefore remains to specify an equally fast rule for determining $p(x)$, which is the problem that I will tackle next.

Let us begin this endeavour by introducing a shorthand notation for the space in which we are looking for $p(x)$.

Notation: Set of polynomials $\mathcal{P}_n$

Throughout this lecture, I will write $\mathcal{P}_n$ to denote the set

$$\mathcal{P}_n = \big\{\text{polynomials } p(x) \text{ of degree}(p) \leq n\big\}.$$

Krylov Subspace Methods

Next, let us observe that in most applications, we are concerned about one of the following two notions of error.

Def: Error and residual

Let $\tilde{x}$ be an approximate solution to the linear system $Ax = b$.
We then call $A^{-1}\,b - \tilde{x}$ the *error* in $\tilde{x}$, and $b - A\tilde{x}$ the *residual* of $\tilde{x}$.

Which notion of error is more important depends on the details of the application. For example:

- We are primarily concerned about the *error* when solving the discrete Poisson equation, because the error tells us to what extent we can trust the approximate concentration $\tilde{u}_n$.
- We are primarily concerned about the *residual* if $Ax = b$ represents a statistical model which tries to explain the data b based on some parameters x, because the residual tells us how well the approximate parameters $\tilde{x}$ reproduce the experimental data.

Krylov Subspace Methods

Considering the above, we conclude that ideally we would be choosing $p \in \mathcal{P}_n$ such that it minimises *either* the error *or* the residual depending on which is the relevant notion of error in the given application.

Unfortunately, minimising the error $A^{-1}b - p(A)\,b$ is usually not possible because doing so would require that we already know the exact solution $A^{-1}b$, and if we knew the exact solution then we would not bother computing an approximation $p(A)\,b \approx A^{-1}b$.

We therefore conclude that minimising the residual $b - A\,p(A)\,b$ is usually the best we can do. Fortunately, minimising the residual turns out to be equivalent to minimising the error in some sense.

Lemma: Equivalence of error and residual

We have

$$\|A\|^{-1}\,\|b - A\tilde{x}\| \;\leq\; \|A^{-1}b - \tilde{x}\| \;\leq\; \|A^{-1}\|\,\|b - A\tilde{x}\|.$$

Proof. Immediate consequence of

$$b - A\tilde{x} = A\,(A^{-1}b - \tilde{x}) \qquad \text{and} \qquad A^{-1}b - \tilde{x} = A^{-1}\,(b - A\tilde{x}).$$

Krylov Subspace Methods

The above suggests that we determine $p \in \mathcal{P}_n$ by setting

$$p = \arg\min_{p \in \mathcal{P}_n} \|b - A\,p(A)\,b\|.$$

The norm $\|\cdot\|$ in this rule could in principle be chosen arbitrarily, but it turns out that reasonable algorithms and convergence theories can be developed only for a few special choices.

One such choice is the 2-norm, which leads us to the following rule.

Def: GMRES minimisation problem

The problem of determining

$$p = \arg\min_{p \in \mathcal{P}_n} \|b - A\,p(A)\,b\|_2$$

is known as the *GMRES minimisation problem*.

Etymology of GMRES

GMRES is an abbreviation for "Generalised Minimal RESidual".

The "minimal residual" part of this name is self-explanatory, and the reason for the "generalised" will become clear later.

Krylov Subspace Methods

Combining the Krylov subspace idea $p(A)\,b \approx A^{-1}b$ with the GMRES rule for choosing $p(x)$ leads us to our first concrete Krylov subspace method.

Def: GMRES iteration

The problem of computing

$$x_n = p(A)\,b \qquad \text{where} \qquad p = \underset{p \in \mathcal{P}_n}{\arg\min} \|b - A\,p(A)\,b\|_2$$

given a matrix A, a vector b and a polynomial degree n is known as a *GMRES iteration*.

Krylov Subspace Methods

The above GMRES iteration is usually run repeatedly for increasingly larger degrees n until we either reach a maximal degree $n_{\max} \in \mathbb{N}$ or meet a residual tolerance τ.

Algorithm GMRES

1: **for** $n = 0, 1, 2, \ldots, n_{\max}$ **do**
2: Compute $x_n = p(A)\, b$ where $p = \arg\min_{p \in \mathcal{P}_n} \|b - A\, p(A)\, b\|_2$.
3: **if** $\|b - Ax_n\|_2 \leq \tau$ **then** **return** x_n
4: **end for**
5: **return** $x_{n_{\max}}$.

This procedure is known as the *GMRES algorithm*.

Krylov Subspace Methods

GMRES as an iterative algorithm

The above terminology turns out to be somewhat confusing, so let me emphasise once more that *GMRES algorithm* refers to solving the GMRES minimisation problem for increasingly larger degrees n while each individual solve is called a *GMRES iteration*.

This terminology is confusing because it is inaccurate: "to iterate" usually means to generate a sequence $x_{n+1} = f(x_n)$ by recursively evaluating a single function $f(x)$, and this is not what GMRES does; GMRES generates the sequence $x_n = f(n)$ by evaluating a function which depends on only the iteration counter n but not on the previous state x_{n-1}.

A partial justification for this technically inaccurate terminology is that even though the nth GMRES iterate x_n is logically independent of x_{n-1}, it turns out that much of the internal state required for computing $(x_k)_{k=0}^{n-1}$ can be reused for computing x_n. GMRES is thus not an iteration from a mathematical point of view, but it starts to look a lot like an iteration once you consider implementation details.

Krylov Subspace Methods

The fact that the computations for $(x_k)_{k=0}^{n-1}$ can be reused when computing x_n is also noticeable in the following result.

Theorem

The runtime of executing the first n GMRES iteration is the same (in the big O sense) as that of executing only the nth GMRES iteration.

Proof. Omitted (see slide 14).

The above sets GMRES apart from e.g. the finite difference method, where having solved the discrete Poisson equation on a grid with $n-1$ points in each direction does not help you at all with solving the same equation on an n-point grid.

It also motivates why GMRES can afford to compute $x_n = p(A)\,b$ for increasingly larger degrees n until sufficient accuracy is achieved, while the same would not be practical for the finite difference method.

Krylov Subspace Methods

Algorithmic details

I deliberately kept the above description of the GMRES algorithm fairly abstract, and I did so because of two complementary reasons.

- Devising a fast and numerically robust implementation of GMRES requires taking into account a large number of technical details.
- You do not need to know these details to understand the GMRES algorithm.

Having said this, I should add that there is one more algorithmic detail that you do need to be aware of. See next slide.

Krylov Subspace Methods

Thm: GMRES as a least squares problem

The GMRES solution

$$x_n = p(A)\, b \qquad \text{where} \qquad p = \underset{p \in \mathcal{P}_n}{\arg\min} \, \| b - A\, p(A)\, b \|_2$$

can be computed by assembling

$$V_n = \begin{pmatrix} b & Ab & A^2 b & \dots & A^n b \end{pmatrix}$$

and setting

$$x_n = V_n\, c_n \qquad \text{where} \qquad c_n = \underset{c_n \in \mathbb{R}^{n+1}}{\arg\min} \, \| b - A\, V_n\, c_n \|_2 .$$

This is a standard least squares problem which can be solved using the QR factorisation.

Proof. The claim follows immediately from the fact that

$$p(A)\, b = \sum_{k=0}^{n} c_n[k]\, A^k\, b = V_n\, c_n .$$

Krylov Subspace Methods

The linear subspaces spanned by the columns of V_n have been given their own name.

Def: Krylov subspace

$\mathcal{K}_n(A, b) = \text{span}\{b, Ab, \ldots, A^{n-1}b\}$ is called the nth *Krylov subspace*.

This definition explains why the class of algorithms discussed in this lecture are called *Krylov subspace methods:* the approximate solutions computed by these algorithms are given by

$$x_n = p(A)\, b \quad \text{where} \quad p \in \mathcal{P}_n \qquad \Longleftrightarrow \qquad x_n \in \mathcal{K}_{n+1}(A, b).$$

Code example

See `gmres_example()` for a demonstration of how GMRES is used in practice.

Krylov Subspace Methods

The GMRES algorithm implicitly assumes that using larger degrees n will lead to longer runtimes but smaller errors. Let us next verify and quantify this assumption by studying the asymptotic behaviours of the runtime and error as functions of n.

Thm: Runtime and memory of GMRES

The first n iterations of the GMRES algorithm from slide 11 require

$$O(n\,\mathrm{nnz}(A) + n^2\,\mathrm{length}(b)) \text{ runtime} \quad \text{and} \quad O(n\,\mathrm{length}(b)) \text{ memory.}$$

Partial proof. $O(n\,\mathrm{nnz}(A))$ runtime is required for evaluating the n matrix-vector products in

$$V_n = \begin{pmatrix} b & Ab & A^2b & \dots & A^nb \end{pmatrix},$$

and $O(n\,\mathrm{length}(b))$ memory is required for storing V_n.

The remaining $O(n^2\,\mathrm{length}(b))$ runtime is required for solving the least squares problem $\arg\min_{x_n \in \mathcal{K}_{n+1}(A,b)} \|b - Ax_n\|$. We do not have the tools necessary to derive this part of the runtime estimate.

Krylov Subspace Methods

To discuss the convergence of GMRES, it will be useful to relate functions of matrices $f : \mathbb{R}^{n\times n} \to \mathbb{R}^{n\times n}$ with scalar functions $f : \mathbb{C} \to \mathbb{C}$. The key tool for doing so is the eigendecomposition defined as follows.

Def: Eigendecomposition

The eigendecomposition of a matrix $A \in \mathbb{R}^{n\times n}$ is a pair of matrices $V, \Lambda \in \mathbb{C}^{n\times n}$ such that $A = V\Lambda V^{-1}$ and Λ is diagonal.

Rewriting $A = V\Lambda V^{-1}$ in the form $AV = V\Lambda$ and considering the kth column in this identity, we obtain

$$A\, V[:, k] = V[:, k]\, \Lambda[k, k].$$

This shows that the columns of V are the eigenvectors and the diagonal entries of Λ are the eigenvalues of A.

Notation: I will use the abbreviation $\lambda_k = \Lambda[k, k]$ in the following.

Krylov Subspace Methods

Eigendecompositions will play an important role in the upcoming discussion, so let us refresh our memories on some key facts.

Lemma: Basic eigendecomposition facts

- There are matrices which do not have an eigendecomposition, but these matrices are sufficiently rare that we can ignore them for the purposes of this lecture.
- Eigendecompositions are unique except for the following operations:
  - We can arrange the eigenvalues and -vectors in any arbitrary order, i.e. $V\Lambda V^{-1} = VP\Lambda P^T V^T$ for any permutation matrix P.
  - If some eigenvalues are repeated, say $\lambda_1 = \ldots = \lambda_k$, then we can choose the corresponding eigenvectors $V[:, 1{:}k]$ to be any basis for $\operatorname{span} V[:, 1{:}k]$.
- If A is symmetric, then the eigenvalues and -vectors are real and the eigenvectors can be chosen to be orthogonal (i.e. $V^{-1} = V^T$).
- If A is not symmetric, then the eigenvalues and -vectors can be complex even if A is real.

Proof. See any linear algebra textbook.

Krylov Subspace Methods

We can now relate matrix and scalar functions as follows.

Def: Function of a matrix

Let $A = V\Lambda V^{-1}$ be the eigendecomposition of A, and let $f : \mathbb{C} \to \mathbb{C}$. The matrix function $f(A)$ is then given by

$$f(A) = V\, f(\Lambda)\, V^{-1}$$

where

$$f(\Lambda) \;=\; \mathrm{diag}\Big(f(\lambda_1), \ldots, f(\lambda_n)\Big) \;=\; \begin{pmatrix} f(\lambda_1) & & \\ & \ddots & \\ & & f(\lambda_n) \end{pmatrix}.$$

Krylov Subspace Methods

Let us next establish some basic facts.

Lemma

The above definition of matrix functions is consistent with the earlier definition of matrix polynomials, i.e. we have

$$p(x) = \sum_{k=0}^{n} c_k x^k \quad \Longleftrightarrow \quad p(A) = \sum_{k=0}^{n} c_k A^k.$$

Proof.

$$V\, p(\Lambda)\, V^{-1} = V \left(\sum_{k=0}^{n} c_k \Lambda^k \right) V^{-1} = \sum_{k=0}^{n} c_k V \Lambda^k V^{-1}$$
$$= \sum_{k=0}^{n} c_k \left(V \Lambda V^{-1} \right)^k = \sum_{k=0}^{n} c_k A^k.$$

Krylov Subspace Methods

Lemma: 2-norm of matrix functions

Let $A = V\Lambda V^{-1}$ be the eigendecomposition of A. Then,

$$\|f(A)\|_2 \quad \leq \quad \kappa_2(V) \quad \max_k |f(\lambda_k)|.$$

The $\kappa_2(V)$ in the above formula is refers to the following quantity.

Def: Condition number of a matrix

$\kappa_2(A) = \|A\|_2 \, \|A^{-1}\|_2$ is called the *2-norm condition number* of A.

Proof (not examinable). We have

$$\|f(A)\|_2 = \|V \, f(\Lambda) \, V^{-1}\|_2 \leq \|V\|_2 \, \|f(\Lambda)\|_2 \, \|V^{-1}\|_2 = \kappa_2(V) \, \|f(\Lambda)\|_2.$$

It thus remains to show that

$$\big\|\mathrm{diag}(d_1, \ldots, d_n)\big\|_2 = \max_k |d_k|,$$

which is a basic result from linear algebra.

Krylov Subspace Methods

We can now establish the following estimate for the GMRES residual.

Thm: GMRES residual estimate, version 1
Let $A = V\Lambda V^{-1}$ be the eigendecomposition of A. Then,

$$\min_{p\in\mathcal{P}_n} \|b - A\,p(A)\,b\|_2 \quad \leq \quad \kappa_2(V)\ \|b\|_2\ \min_{p\in\mathcal{P}_n} \max_k \left|1 - \lambda_k\,p(\lambda_k)\right|.$$

Proof.

$$\begin{aligned} \|b - A\,p(A)\,b\|_2 &\leq \|I - A\,p(A)\|_2\,\|b\|_2 \\ &\leq \kappa_2(V)\ \|b\|_2\ \max_k \left|1 - \lambda_k\,p(\lambda_k)\right|. \end{aligned}$$

Krylov Subspace Methods

This estimate is easier to interpret if we rewrite it as follows.

Thm: GMRES residual estimate, version 2
Let $A = V\Lambda V^{-1}$ be the eigendecomposition of A. Then

$$\min_{p\in\mathcal{P}_n} \|b - A\,p(A)\,b\|_2 \quad \leq \quad \kappa_2(V)\;\|b\|_2 \min_{q\in\mathcal{P}_{n+1}} \max_k \frac{|q(\lambda_k)|}{|q(0)|}.$$

Proof. Immediate consequence of the following lemma.

Lemma (not examinable)

$$\min_{p\in\mathcal{P}_n} \max_k \left|1 - \lambda_k\,p(\lambda_k)\right| \;=\; \min_{q\in\mathcal{P}_{n+1}} \max_k \frac{|q(\lambda_k)|}{|q(0)|}$$

Proof. See next slide.

Krylov Subspace Methods

Proof (not examinable).

Given $p \in \mathcal{P}_n$, I can construct a polynomial $q \in \mathcal{P}_{n+1}$ by setting

$$q(x) = 1 - x\,p(x).$$

This polynomial satisfies $q(0) = 1$; hence we conclude that

$$\min_{p\in\mathcal{P}_n} \max_k \left|1 - \lambda_k\, p(\lambda_k)\right| \geq \min_{q\in\mathcal{P}_{n+1}} \max_k \frac{|q(\lambda_k)|}{|q(0)|}.$$

Conversely, given $q \in \mathcal{P}_{n+1}$, I observe that

$$1 - q(x)/q(0) \;\in\; \mathcal{P}_{n+1} \qquad \text{and} \qquad \Big(1 - q(x)/q(0)\Big)_{x=0} = 0.$$

According to the fundamental theorem of algebra, there hence exists a polynomial $p \in \mathcal{P}_n$ such that

$$1 - q(x)/q(0) = x\,p(x) \qquad \Longleftrightarrow \qquad 1 - x\,p(x) = \frac{q(x)}{q(0)}.$$

This shows that

$$\min_{p\in\mathcal{P}_n} \max_k \left|1 - \lambda_k\, p(\lambda_k)\right| \leq \min_{q\in\mathcal{P}_{n+1}} \max_k \frac{|q(\lambda_k)|}{|q(0)|}.$$

Krylov Subspace Methods

From residual estimate to convergence estimate

Our next aim will be to translate the above GMRES residual estimate

$$\min_{p \in \mathcal{P}_n} \|b - A\,p(A)\,b\|_2 \quad \leq \quad \kappa_2(V)\ \|b\|_2 \min_{q \in \mathcal{P}_{n+1}} \max_k \frac{|q(\lambda_k)|}{|q(0)|}$$

into a convergence estimate of the form

$$\min_{p \in \mathcal{P}_n} \|b - A\,p(A)\,b\|_2 \quad \leq \quad O\big(f(n)\big).$$

To do so, I observe:

- The factors $\kappa_2(V)$ and $\|b\|_2$ in the residual estimate do not depend on the degree n. These factors hence influence only the prefactor hidden by the big O notation, but not the asymptotic behaviour.
- The only factor in the residual estimate which does depend on n is the one highlighted in green. In addition to n, this factor depends on only the eigenvalues λ_k but not on the eigenvectors V or the right-hand side b. The asymptotic convergence behaviour $f(n)$ of GMRES hence depends on only the eigenvalues λ_k.

Krylov Subspace Methods

From residual estimate to convergence estimate (continued)

It follows that we are looking for a convergence estimate of the form

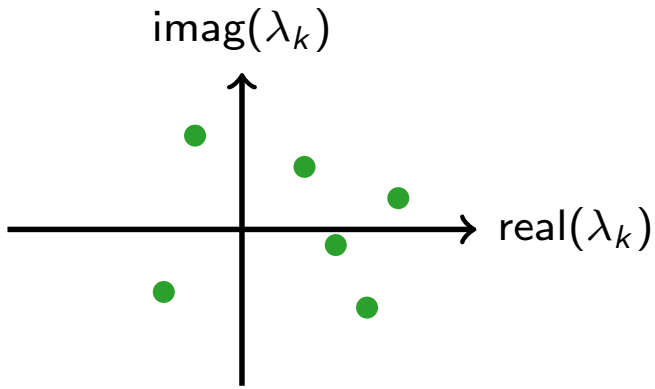
$$\min_{q \in \mathcal{P}_{n+1}} \max_k \frac{|q(\lambda_k)|}{|q(0)|} \quad = \quad O\big(f(n, \{\lambda_k\})\big).$$

Unfortunately, establishing such an estimate for a fully general set of eigenvalues $\{\lambda_k\}$ is not possible because such a set contains more information than what can reasonably be handled using pen and paper.

Example. How would you determine

$$q = \arg\min_{q \in \mathcal{P}_{n+1}} \max_k \frac{|q(\lambda_k)|}{|q(0)|}$$

given the eigenvalues



Krylov Subspace Methods

From residual estimate to convergence estimate (continued)

Instead of establishing a single convergence estimate which is both rigorous and general, I will therefore next describe a sequence of results which are either rigorous or general, but not both.

Krylov Subspace Methods

Rule-of-thumb GMRES convergence estimate

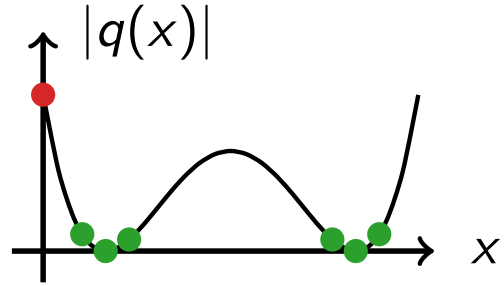
GMRES converges faster the more the eigenvalues cluster around a few points far away from 0.

"Proof." Let me denote the cluster points by $(x_k)_k$. We can then obtain a good upper bound on

$$\min_{q \in \mathcal{P}_{n+1}} \max_k \frac{|q(\lambda_k)|}{|q(0)|}$$

by considering the particular polynomial

$$q(x) = \prod_k (x - x_k)^{n_k} \in \mathcal{P}_{\sum_k n_k} \qquad \longleftrightarrow$$



This polynomial leads to smaller values of $|q(\lambda_k)|$ the closer the λ_k are to one of the cluster points, and it leads to a larger value of $|q(0)|$ the further these cluster points are from 0.

Krylov Subspace Methods

Thm: Finite termination of GMRES

Assume A has m distinct eigenvalues. Then,

$$\min_{p \in \mathcal{P}_n} \left\| b - Ap(A)\, b \right\|_2 = 0 \qquad \text{for all } n \geq m-1,$$

i.e. the GMRES solution $x_n = p(A)\, b$ is exact if degree$(p) \geq m-1$.

Proof. Let $\left(\lambda_k\right)_{k=1}^{m}$ be the distinct eigenvalues, and consider the polynomial

$$\hat{q}(x) = \prod_{k=1}^{m} \left(x - \lambda_k\right) \in \mathcal{P}_m.$$

According to the above GMRES residual estimate, we then have

$$\min_{p \in \mathcal{P}_{m-1}} \|b - A\, p(A)\, b\|_2 \quad \leq \quad \kappa_2(V)\, \|b\|_2 \, \max_k \, \frac{|\hat{q}(\lambda_k)|}{|\hat{q}(0)|} \quad = \quad 0.$$

Krylov Subspace Methods

Remark 1

An $m \times m$ matrix A has at most m distinct eigenvalues; hence the above result implies in particular that the GMRES solution

$$x_n = p(A)\, b \quad \text{is exact if} \quad \text{degree}(p) \geq \text{length}(b) - 1.$$

This insight is of no practical use, however, because the runtime of GMRES for $n = \text{length}(b) - 1$ is

$$O\big(n\, \text{nnz}(A) + n^2\, \text{length}(b)\big) = O\big(\text{length}(b)^3\big),$$

which is as expensive as a dense LU factorisation.

Remark 2

It is customary to call $\text{degree}(p) = 0$ the *first* iteration of GMRES, $\text{degree}(p) = 1$, the *second* iteration of GMRES, and so on.

Under this convention, the above observation can be summarised as saying that GMRES applied to a matrix A with m distinct eigenvalues converges in at most m iterations.

Krylov Subspace Methods

Thm: GMRES convergence for positive eigenvalues

Assume all eigenvalues of A are contained in an interval $[c, d] \subset (0, \infty)$, and set $\kappa = d/c$. Then,

$$\min_{p \in \mathcal{P}_n} \|b - A\,p(A)\,b\|_2 \;\leq\; 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^n,$$

i.e. GMRES converges exponentially with rate $\rho(\kappa) = \frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}$.

This rate increases monotonically from $\rho(1) = 0$ to $\rho(\infty) = 1$, where

- $\rho(1) = 0$ means that GMRES converges instantly if all eigenvalues fall on a single point $\lambda_k = c = d$ (cf. slide 30), and
- $\rho(\infty) = 1$ means that GMRES does not converge at all if either c is close to 0 or d is very large (cf. slide 29).

Krylov Subspace Methods

Proof sketch (not examinable).

The result can be shown by inserting the shifted Chebyshev polynomial

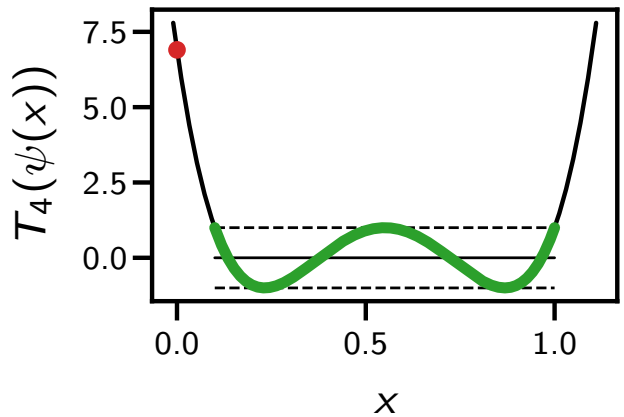
$$q(x) = T_{n+1}\big(\psi(x)\big) \qquad \text{with} \qquad \psi(x) = \tfrac{2}{d-c}\left(x - \tfrac{d+c}{2}\right)$$

into the residual estimate

$$\min_{p \in \mathcal{P}_n} \|b - A\,p(A)\,b\|_2 \quad \leq \quad \kappa_2(V)\ \|b\|_2\ \max_k\ \tfrac{|q(\lambda_k)|}{|q(0)|}.$$

$T_n(x)$ denotes the nth Chebyshev polynomial, and $\psi(x)$ is the unique linear polynomial $\psi \in \mathcal{P}_1$ such that $\psi([c,d]) = [-1,1]$.

The detailed computations are beyond the scope of this module, but let me point out that choosing $q(x)$ to be a Chebyshev polynomial is natural due to the equioscillation property of these polynomials.



Krylov Subspace Methods

Limitations of our convergence analysis

The above analysis is based on the assumption that the convergence of GMRES is determined solely by the $\min_q \max_q$ factor in the residual estimate

$$\min_{p\in\mathcal{P}_n} \|b - A\,p(A)\,b\|_2 \quad \leq \quad \kappa_2(V)\;\|b\|_2 \min_{q\in\mathcal{P}_{n+1}} \max_k \frac{|q(\lambda_k)|}{|q(0)|}.$$

This may fail to be true, however, because the above inequality provides only an upper bound; it is possible that the GMRES residual on the left behaves differently from the upper bound on the right if the inequality is loose enough.

I will next present two examples where the above convergence analysis indeed fails to to make a reasonably accurate prediction on the convergence of the GMRES algorithm.

Luckily, situations like these occur rarely in practice. My motivation for presenting them is hence only to point out that the convergence of GMRES is a fairly delicate and complicated issue.

Krylov Subspace Methods

Example: Limitations of our convergence analysis

`convergence_subtleties()` plots the GMRES residuals $\|b - Ax_n\|$ as a function of n for three different linear systems

$$A_1 x = b_1, \qquad A_1 x = b_2, \qquad A_2 x = b_1.$$

A_1 and A_2 have exactly the same eigenvalues $\lambda_k \in [0.1, 1]$. According to the theorem on slide 32, GMRES should hence converge exponentially with rate $\rho(10)$ for all three of the above systems, but this is not true:

- GMRES applied to $A_1 x = b_1$ converges as expected.
- GMRES applied to $A_1 x = b_2$ converges faster than expected. The reason for this is that b_2 is exactly 0 in the direction of half of the eigenvectors, and this renders the corresponding eigenvalues irrelevant for the convergence of GMRES.
- GMRES applied to $A_2 x = b_1$ converges slower than expected. The reason for this is the excessively large value of $\kappa_2(V_2)$ (see output).

While irrelevant to the main point of this example, I would also like to point out that the GMRES residual for $A_2 x = b_1$ drops to zero very sharply at $n = 50$. This is a manifestation of the finite termination property described on slide 30.

Krylov Subspace Methods

Outlook

We have now completed the discussion of the convergence of GMRES.

I will next present two tricks which can help improve the performance of GMRES in some cases.

Krylov Subspace Methods

Restarted GMRES

Recall from slide 17 that n GMRES iterations require

$$O(n\,\mathrm{nnz}(A) + n^2\,\mathrm{length}(b)) \text{ runtime} \quad \text{and} \quad O(n\,\mathrm{length}(b)) \text{ memory},$$

and that these terms represent

- the runtime of executing n matrix-vector products (matvecs),
- the runtime of solving the least squares problem (lsq), and
- the memory footprint of storing $V_n = \begin{pmatrix} b & \dots & A^n b \end{pmatrix}$, respectively.

We observe that the runtime scales quadratically in n and the memory scales linearly in n. This usually means that performing a large number of GMRES iterations is not feasible both for runtime and memory reasons.

Luckily, there is a simple trick which often allows us to run GMRES repeatedly for a small number of iterations rather than once for a large number of iterations. This trick is known as *restarted GMRES*.

Krylov Subspace Methods

Restarted GMRES (continued)

The idea behind restarted GMRES is to replace the standard, "one-shot" GMRES approximation

$$p(A)\, b \approx A^{-1}\, b, \tag{1}$$

with a sequence of approximations $(x_k)_{k=0}^{\infty}$ given by

$$x_0 = 0, \qquad x_{k+1} \;=\; x_k + p(A)\,\big(b - Ax_k\big). \tag{2}$$

The rationale for doing so is the following.

- If $p(A)$ were the exact inverse, then (2) would reduce to

$$x_{k+1} \;=\; x_k + A^{-1}\,(b - Ax_k) \;=\; x_k + A^{-1}b - x_k \;=\; A^{-1}\,b$$

  and thus (1) and (2) would be equivalent
- If $p(A)$ is only an approximate inverse, then (2) yields a sequence of approximations $(x_k)_k$ which may converge to $A^{-1}\,b$ even for fairly crude $p(A) \approx A^{-1}$, while (1) yields just a single approximation $p(A)\,b$ with no means for improvement other than $p(A)$.

Krylov Subspace Methods

The following definition formalises the idea on the previous slide.

Def: Restarted GMRES

The sequence $(x_k)_{k=0}^{\infty}$ given by

$$x_0 = 0, \qquad x_{k+1} \;=\; x_k + p_k(A)\left(b - Ax_k\right)$$

where

$$p_k = \underset{p_k \in \mathcal{P}_{n_{\text{inner}}}}{\arg\min} \left\| \left(I - A\,p(A)\right)\left(b - Ax_k\right) \right\|_2$$

is known as *restarted GMRES*.

The parameter $n_{\text{inner}} \in \mathbb{N}$ is called the number of *inner iterations*, while the step $x_k \to x_{k+1}$ is called an *outer iteration*.

Restarted GMRES in Julia

`IterativeSolvers.gmres` actually performs restarted GMRES.

The number of inner iterations can be specified using the `restart` keyword argument, and the `maxiter` keyword argument specifies an upper bound on the overall number of inner iterations.

Krylov Subspace Methods

The obvious advantage of restarted GMRES is that it reduces the runtime to linear in n_{outer} and the memory to independent of n_{outer}.

Thm: Runtime and memory of restarted GMRES

n_{outer} outer iterations of restarted GMRES require

$$O(n_{\text{inner}}^2 \, n_{\text{outer}}) \text{ runtime} \quad \text{and} \quad O(n_{\text{inner}}) \text{ memory.}$$

Proof. Immediate consequence of the GMRES requirements on slide 17.

Krylov Subspace Methods

However, this reduction in runtime comes at the price that the residual estimate must be modified as follows.

Thm: Residual estimate for restarted GMRES

Let $A = V\Lambda V^{-1}$ be the eigendecomposition of A, and denote by $(x_k)_k$ the sequence constructed by restarted GMRES with n_{inner} inner iterations. Then,

$$\|b - Ax_k\|_2 \quad \leq \quad \left(\kappa_2(V) \min_{q \in \mathcal{P}_{n_{\text{inner}}+1}} \max_{\ell} \frac{|q(\lambda_\ell)|}{|q(0)|} \right)^k \|b\|_2.$$

Proof. Following the same arguments as those leading to the GMRES residual estimate on slide 24, we obtain

$$\begin{aligned} \|b - Ax_{k+1}\|_2 &= \min_{p_k \in \mathcal{P}_{n_{\text{inner}}}} \left\| \left(I - A\,p(A)\right)(b - Ax_k) \right\|_2 \\ &\leq \kappa_2(V) \min_{q \in \mathcal{P}_{n_{\text{inner}}}} \max_{\ell} \frac{|q(\lambda_\ell)|}{|q(0)|} \|b - Ax_k\|_2. \end{aligned}$$

The claim then follows by recursion.

Krylov Subspace Methods

Restarted GMRES (conclusion)

The above estimate

$$\|b - Ax_k\|_2 \quad \leq \quad \left(\kappa_2(V) \min_{q \in \mathcal{P}_{n_{\text{inner}}+1}} \max_{\ell} \frac{|q(\lambda_\ell)|}{|q(0)|} \right)^k \|b\|_2.$$

indicates that the convergence of restarted GMRES can fall anywhere between the following two extreme cases.

- If [blue factor] $\approx \rho^{n_{\text{inner}}}$ where $\rho \in [0, 1]$ denotes the rate of convergence of standard GMRES, then restarted GMRES will converge roughly as fast as standard GMRES.
- If [blue factor] ≥ 1, then restarted GMRES may not converge at all.

Realistic examples close to each of these extreme cases are presented in `restarted_gmres_good()` and `restarted_gmres_bad()`.

The one-sentence summary of restarted GMRES is thus as follows:

Replacing standard with restarted GMRES reduces the runtime per iteration, and it may either have no significant impact on the speed of convergence or it may completely ruin the convergence of GMRES depending on the properties of A.

Krylov Subspace Methods

Preconditioning

It is sometimes possible to accelerate the convergence of GMRES by replacing the original linear system

$$Ax = b \qquad \text{with} \qquad (P_L^{-1} A P_R^{-1})(P_R x) = P_L^{-1} b$$

for some invertible matrices P_L and P_R. This technique is known as *preconditioning*, and the matrices P_L and P_R are called *left* and *right preconditioners*, respectively.

Krylov Subspace Methods

Example

If we chose $P_L = A$ and $P_R = I$, then we have

$$P_L^{-1} A = A^{-1} A = I.$$

The identity has only a single distinct eigenvalue, namely $\lambda_k = 1$. GMRES applied to

$$P_L^{-1} A x = P_L^{-1} b$$

hence converges in a single iteration.

Of course, choosing $P_L = A$ is not practical because doing so would require us to be able to evaluate $P_L^{-1} b = A^{-1} b$, and this is precisely the problem that we are trying to solve using GMRES.

Def: Ideal preconditioner

$P_L = A$ (or $P_R = A$) is known as the *ideal preconditioner* for obvious reasons.

Krylov Subspace Methods

Choice of preconditioner

The above example indicates that an effective preconditioner should balance the following two conflicting goals.

- We should have $P_L^{-1} A P_R^{-1} \approx I$ such that GMRES applied to

$$(P_L^{-1} A P_R^{-1})\,(P_R x) = P_L^{-1} b$$

  converges rapidly.

- It should be possible to compute $P_L^{-1} b$ and $P_R^{-1} b$ rapidly such that

$$P_R^{-1}\ p\big(P_L^{-1} A P_R^{-1}\big)\ (P_L^{-1} b)$$

  can be evaluated efficiently.

You will see an example of such a preconditioner in the midterm assignment.

Krylov Subspace Methods

We have now concluded the discussion of GMRES and move on to discussing other Krylov subspace methods.

The first such method is just a minor variation on GMRES.

Def: MinRes iteration and MinRes algorithm

If A is symmetric, then the GMRES problem of computing

$$p(A)\,b \approx A^{-1}b \qquad \text{where} \qquad p = \operatorname*{arg\,min}_{p \in \mathcal{P}_n} \|b - A\,p(A)\,b\|_2$$

is known as a *MinRes iteration*.

MinRes iterations give rise to a *MinRes algorithm* in exactly the same way as the GMRES iterations give rise to the GMRES algorithm.

Etymology of "MinRes" and "GMRES"

"MinRes" is an abbreviation for "minimal residual".

The existence of this method explains why there is a "G" in "GMRES": MinRes was developed before GMRES, and thus GMRES is the MinRes method "generalised" to arbitrary A rather than just symmetric A.

Krylov Subspace Methods

The reason why "symmetric GMRES" has been given its own name is that symmetry leads to a significant reduction in runtime and memory.

Thm: Runtime and memory of MinRes

The first n iterations of the MinRes algorithm require

$$O(n\,\mathrm{nnz}(A) + n\,\mathrm{length}(b)) \text{ runtime} \quad \text{and} \quad O(\mathrm{length}(b)) \text{ memory.}$$

These estimates involve one power of n less than the corresponding estimates for GMRES in the highlighted terms.

Proof. Omitted.

An important consequence of this estimate is that MinRes does not require restarts because the runtime is linear in n and the memory is $O(1)$ already for the standard algorithm.

Krylov Subspace Methods

Unlike the runtime, the convergence theory of MinRes is essentially the same as that of GMRES.

Thm: MinRes residual estimate

Let $A = V \Lambda V^T$ be the eigendecomposition of the symmetric matrix A. Then

$$\min_{p \in \mathcal{P}_n} \|b - A\,p(A)\,b\|_2 \quad \leq \quad \cancel{\kappa_2(V)}\ \|b\|_2 \min_{q \in \mathcal{P}_{n+1}} \max_k \frac{|q(\lambda_k)|}{|q(0)|}.$$

Proof. Immediate corollary of the estimate on slide 24.

The only differences between the above result and the earlier GMRES residual estimate are the following.

- We have $\kappa_2(V) = 1$ because the eigenvectors of a symmetric matrix are orthogonal.
- The polynomial minimisation problem is slightly easier because the eigenvalues of a real, symmetric matrix are guaranteed to be real.

Krylov Subspace Methods

MinRes convergence theory

We conclude from the above that the convergence of MinRes exhibits the same properties as that of GMRES.

- The convergence of MinRes is mainly determined by the eigenvalues of A but not the eigenvectors V and not the right-hand side b.

  In fact, the dependence on V is even less for MinRes than for GMRES since $\kappa_2(V) = 1$ rules out that the above estimate can become lose due to a very large $\kappa_2(V)$ (recall the "Limitations of our convergence analysis" example on slide 35).
- MinRes converges faster the more the eigenvalues cluster around a few points far away from 0.
- MinRes produces the exact solution after at most length(b) iterations.
- We have exponential convergence with a known rate $\rho(d/c)$ if all eigenvalues are contained in an interval $[c, d] \subset (0, \infty)$.

Krylov Subspace Methods

MinRes preconditioning

We have seen on slide 43 that the convergence of GMRES can sometimes be improved by replacing

$$Ax = b \qquad \text{with} \qquad (P_L^{-1} A P_R^{-1})\,(P_R x) = P_L^{-1} b.$$

The same trick also works for MinRes, but P_L and P_R must now satisfy certain conditions to preserve the symmetry of the linear system.

At first sight, it may seem that the necessary condition should be $P_R = P_L^T$ so that

$$P_L^{-1} A P_R^{-1} \;=\; P_L^{-1} A P_L^{-T}$$

is symmetric, but it turns out that considering such a *symmetric preconditioner* is more cumbersome than it should be.

Instead, the best way to precondition MinRes is to consider only left preconditioning

$$P_L^{-1} A x = P_L^{-1} b$$

but demand that P_L is *symmetric and positive definite* (see next slide).

Krylov Subspace Methods

Def: Symmetric, positive definite (spd) matrix

A symmetric matrix $A \in \mathbb{R}^{n \times n}$ is called *positive definite* if

$$v^T A v > 0 \qquad \text{for all } v \in \mathbb{R}^n \setminus \{0\}.$$

An important fact about spd matrices is the following.

Theorem: A matrix is spd if and only if all its eigenvalues are positive.

Proof. Omitted.

The following slides will explain in detail how we can use MinRes to solve the non-symmetric linear system $P^{-1}Ax = P^{-1}b$ and why doing so is better than symmetric preconditioning.

Unfortunately, this story is quite long and convoluted, and while understanding MinRes preconditioning is important, it is not important enough to justify forcing this whole story on you.

I will therefore skip this part in class and continue on slide 58 instead.

Krylov Subspace Methods

The following corollary will be important for MinRes preconditioning.

Corollary (not examinable)
If A is spd, then $A^{1/2}$ exists and we have $(A^{1/2})^2 = A$.

Proof. According to the matrix function definition on slide 20, we have

$$A^{1/2} = V\Lambda^{1/2}V^T.$$

This operation is well defined since A spd $\iff$ $\lambda_k > 0$, and we have

$$A^{1/2}\,A^{1/2} = V\Lambda^{1/2}V^T\,V\Lambda^{1/2}V^T = V\Lambda V^T = A.$$

Krylov Subspace Methods

MinRes preconditioning (continued, not examinable)

I claimed on slide 50 that the best way to do MinRes preconditioning is to choose an spd matrix P and consider

$$P^{-1}Ax = P^{-1}b.$$

It is now time that I admit that this claim was technically a lie; the proper way to do MinRes preconditioning is actually to choose an spd matrix P and then apply MinRes to

$$\left(P^{-1/2}AP^{-1/2}\right)\left(P^{-1/2}x\right) = P^{-1/2}b,$$

but it turns out that this results in an algorithm which looks in many ways like "MinRes applied to $P^{-1}Ax = P^{-1}b$".

The following slides will explain further.

Krylov Subspace Methods

Lemma (not examinable)

The matrix-vector products in MinRes applied to

$$\left(P^{-1/2} A P^{-1/2}\right)\left(P^{-1/2} x\right) = P^{-1/2} b$$

can be rearranged such that we only need to compute $v \mapsto P^{-1}\, v$ rather than $v \mapsto P^{-1/2}\, v$. In particular, we have

$$P^{-1/2}\, p\big(P^{-1/2} A P^{-1/2}\big)\, P^{-1/2} b \;=\; p(P^{-1} A)\, P^{-1} b. \tag{3}$$

Partial proof. Equation (3) follows from the fact that

$$P^{-1/2}\left(P^{-1/2} A P^{-1/2}\right)^k P^{-1/2} = \left(P^{-1} A\right)^k P^{-1}\, b,$$

which can be verified using induction over k.

Verifying the more general first part of the statement requires knowledge about details of the MinRes algorithm not covered in this module; hence I omit this part.

Krylov Subspace Methods

MinRes preconditioning (continued, not examinable)

The right-hand side of the above equation

$$P^{-1/2}\, p\big(P^{-1/2} A P^{-1/2}\big)\, P^{-1/2} b \;=\; p(P^{-1}A)\, P^{-1} b$$

is exactly the expression that would result if we applied GMRES to

$$P^{-1} A x = P^{-1} b.$$

We can thus interpret the above result as saying that MinRes applied to

$$\big(P^{-1/2} A P^{-1/2}\big)\,(P^{-1/2} x) = P^{-1/2} b$$

behaves in some sense like GMRES applied to $P^{-1}Ax = P^{-1}b$.

This analogy is further strengthened by the corollary on the next slide.

Krylov Subspace Methods

Corollary (not examinable)

MinRes applied to

$$\left(P^{-1/2}AP^{-1/2}\right)\left(P^{-1/2}x\right) = P^{-1/2}b$$

produces the same approximate solutions as GMRES applied to

$$P^{-1}Ax = P^{-1}b.$$

Proof. It follows from

$$P^{-1/2}\,p\left(P^{-1/2}AP^{-1/2}\right)P^{-1/2}b \;=\; p(P^{-1}A)\,P^{-1}b \qquad (3)$$

that the above MinRes and GMRES algorithms produce the same solutions if they pick the same polynomials $p(x)$, and this is indeed the case since (3) also implies that

$$\begin{aligned}\arg\min_{p\in\mathcal{P}_n}\left\|b - A\,P^{-1/2}\,p\left(P^{-1/2}AP^{-1/2}\right)P^{-1/2}b\right\|_2 \\ = \quad \arg\min_{p\in\mathcal{P}_n}\left\|b - A\,p\left(P^{-1}A\right)P^{-1}b\right\|_2.\end{aligned}$$

Krylov Subspace Methods

MinRes preconditioning (conclusion, not examinable)

We can summarise the above as saying that there exists an algorithm which runs at the speed of MinRes but which behaves like GMRES applied to $P^{-1}Ax = P^{-1}b$ in all other relevant aspects.

This is the precise meaning of the statement "MinRes preconditioning amounts to choosing an spd matrix P and solving $P^{-1}Ax = P^{-1}b$".

Now that we know how left-preconditioned MinRes works, let us finally address the question why left preconditioning is better than symmetric preconditioner. This question has two fairly simple answers:

- Finding P such that $P^{-1}A \approx I$ is usually easier than finding P such that $P^{-1}AP^{-T} \approx I$.
- Symmetric preconditioning requires multiplication with both P^{-1} and P^{-T} while left preconditioning requires multiplication with only P^{-1}. We may thus expect left preconditioning to be roughly a factor 2 faster than symmetric preconditioning.

Krylov Subspace Methods

The final Krylov subspace method considered in this lecture assumes that A itself is spd and optimises the residual in (almost) the following norm.

Def: Energy or A-norm

If A is spd, then $\|v\|_A = \sqrt{v^T A v}$ is called the *energy* or *A-norm*.

Thm: The energy norm is indeed a norm.

Proof. Omitted.

The precise formulation of our final Krylov method is as follows.

Def: Conjugate gradients (CG) iteration and algorithm

Let A be an spd matrix. The problem of computing

$$p(A)\, b \approx A^{-1} b \qquad \text{where} \qquad p = \underset{p \in \mathcal{P}_n}{\arg\min} \|b - A\, p(A)\, b\|_{A^{-1}}$$

is then known as a *conjugate gradients (CG) iteration*.

CG iterations give rise to a *CG algorithm* in exactly the same way as the GMRES iterations give rise to the GMRES algorithm.

Krylov Subspace Methods

Etymology of "conjugate gradients"

The name "conjugate gradients" is derived from the fact that CG can be interpreted as the gradient descent algorithm for nonlinear optimisation with a certain conjugation condition applied to the above minimisation problem.

Krylov Subspace Methods

Conjugate gradients is by far the most well-known Krylov subspace method. There are several reasons for this.

Conjugate gradients is simple.

Conjugate gradients can be translated into just 8 lines of basic linear algebra operations which are very easy to implement in any programming environment.

Algorithm Conjugate gradients

1: $x_0 = 0, r_0 = b, p_0 = r_0$
2: **for** $k = 1, \ldots, n_{\max}$ **do**
3: $\quad \alpha_k = (r_{k-1}^T r_{k-1})/(p_{k-1}^T A p_{k-1})$
4: $\quad x_k = x_{k-1} + \alpha_k \, p_{k-1}$
5: $\quad r_k = r_{k-1} - \alpha_k \, A p_{k-1}$
6: $\quad \beta_k = (r_k^T r_k)/(r_{k-1}^T r_{k-1})$
7: $\quad p_k = r_k + \beta_k \, p_{k-1}$
8: **end for**

Krylov Subspace Methods

Conjugate gradients is fast.

It follows from the algorithm on the previous slide that conjugate gradients can be evaluated using

$$O(n\,\mathrm{nnz}(A) + n\,\mathrm{length}(b)) \text{ runtime} \quad \text{and} \quad O(\mathrm{length}(b)) \text{ memory.}$$

These are the same requirements as MinRes, but the prefactors hidden by the big O notation are slightly better for conjugate gradients.

Symmetric and positive definite matrices are ubiquitous.

Out of the three Krylov subspace methods presented in this lecture, conjugate gradients imposes the most stringent assumptions on A, namely A must be spd. This may seem to indicate that conjugate gradients should be the most specialised of the three methods, but this is not true: it turns out that most linear systems arising in applied mathematics are spd; hence the apparent lack of generality of conjugate gradients on paper is usally not a problem in practice.

Krylov Subspace Methods

Conjugate gradients controls the error in the A-norm.

Recall from slide 58 that conjugate gradients solves

$$p = \arg\min_{p \in \mathcal{P}_n} \|b - A\,p(A)\,b\|_{A^{-1}}$$

It turns out that the A^{-1}-norm of the residual is in fact the same as the A-norm of the error,

$$\begin{aligned}
\|b - A\tilde{x}\|_{A^{-1}}^2 &= (b - A\tilde{x})^T A^{-1} (b - A\tilde{x}) \\
&= (A^{-1}b - \tilde{x})^T A^T A^{-1} A (A^{-1}b - \tilde{x}) \\
&= (A^{-1}b - \tilde{x})^T A (A^{-1}b - \tilde{x}) \\
&= \|A^{-1}b - \tilde{x}\|_A^2;
\end{aligned}$$

hence CG not only minimises the residual, but it also minimises the error in a norm which often carries physical meaning (it is called the "energy norm" for a reason).

Krylov Subspace Methods

Convergence theory for conjugate gradients

Conjugate gradients optimises a different target function compared to GMRES; hence the GMRES convergence theory does not immediately carry over to conjugate gradients.

Fortunately, it turns out that only minor adjustments are needed to remedy this circumstance. This slide and the following will provide the details.

Lemma

Assume A is spd and has eigenvalues $(\lambda_k)_k$. Then,

$$\|f(A)\|_A \quad \leq \quad \max_k |f(\lambda_k)|.$$

Proof (not examinable). See next slide.

Krylov Subspace Methods

Proof (not examinable).

Let $V\Lambda V^T = A$ be the eigendecomposition of A and consider an arbitrary vector b. We then have

$$\begin{aligned}
\|f(A)\,b\|_A^2 &= b^T f(A)^T\, A\, f(A)\, b \\
&= b^T\, V\, f(\Lambda)\, V^T\, V\Lambda V^T\, V\, f(\Lambda)\, V^T\, b \\
&= b^T\, V\, \Lambda\, f(\Lambda)^2\, V^T\, b \\
&= \sum_{k=1}^{n} \lambda_k\, f(\lambda_k)^2\, (V^T\, b)[k]^2 \\
&\leq \left(\max_k |f(\lambda_k)|^2\right) \sum_{k=1}^{n} \lambda_k\, (V^T\, b)[k]^2 \\
&= \left(\max_k |f(\lambda_k)|^2\right) b^T V\Lambda V^T b \\
&= \left(\max_k |f(\lambda_k)|^2\right) \|b\|_A^2
\end{aligned}$$

where on the fifth line I used that λ_k and $(V^T b)[k]^2$ are both positive.

Krylov Subspace Methods

Thm: Error estimate for conjugate gradients

Assume A is spd and its eigenvalues are given by $(\lambda_k)_k$. Then,

$$\min_{p \in \mathcal{P}_n} \|A^{-1}b - p(A)\,b\|_A \quad \leq \quad \|A^{-1}b\|_A \min_{q \in \mathcal{P}_{n+1}} \max_k \frac{|q(\lambda_k)|}{|q(0)|}.$$

Proof. We have

$$\begin{aligned} \|A^{-1}b - p(A)\,b\|_A &\leq \|I - p(A)\,A\|_A \, \|A^{-1}b\|_A \\ &\leq \|A^{-1}b\|_A \max_k \left|1 - p(\lambda_k)\,\lambda_k\right|. \end{aligned}$$

The minimisation problem in $p(x)$ can then be translated into a minimisation problem for $q(x)$ using the result from slide 24.

Krylov Subspace Methods

Convergence theory for conjugate gradients (conclusion)

The above result shows that the convergence of conjugate gradients is again determined by the factor

$$\min_{q \in \mathcal{P}_{n+1}} \max_k \frac{|q(\lambda_k)|}{|q(0)|}.$$

The convergence theory of conjugate gradients is hence the same as that of GMRES and MinRes.

Preconditioning for conjugate gradients

Just like for MinRes, preconditioning for conjugate gradients amounts to choosing an spd matrix P and solving

$$P^{-1}Ax = P^{-1}b \qquad \text{instead of} \qquad Ax = b.$$

Krylov Subspace Methods

Rounding errors in Krylov subspace methods (not examinable)

We are now nearing the end of our discussion of Krylov subspace methods applied to general linear systems $Ax = b$, but before we can move on there is one last point that I should mention:

> The rounding errors due to finite machine arithmetic may lead to noticeable deviations between theory and practice.

Examples of this phenomenon include:

- GMRES and MinRes applied to the same linear system $Ax = b$ may not produce the same sequence of approximate solutions, see `gmres_vs_minres()`.
- Krylov subspace methods applied to an $m \times m$ linear system $Ax = b$ may fail to converge to the exact solution after at most m iterations, see `finite_termination()`.

The practical consequences of rounding errors are as follows.

- Good: Rounding errors have (usually) no impact on the accuracy of Krylov subspace approximations upon convergence.
- Bad: Rounding errors can delay convergence, but usually only by a few iterations.

Krylov Subspace Methods

Krylov methods for solving $-\Delta_n^{(d)} u_n = f$

We have now completed the discussion of Krylov subspace methods for general linear systems $Ax = b$.

In the remainder of this lecture, I will discuss how Krylov subspace methods compare against LU factorisation when applied to the finite-difference-discretised Poisson equation $-\Delta_n^{(d)} u_n = f$.

To do so, I must execute four steps:

- Determine the eigenvalues of $-\Delta_n^{(d)}$.
- Translate these eigenvalues into a convergence rate $\rho(\{\lambda_k\})$.
- Translate the convergence rate into a number of iterations $n(\rho)$.
- Translate the number of iterations into a runtime estimate.

Krylov Subspace Methods

Krylov methods for solving $-\Delta_n^{(d)} u_n = f$ (continued)

We have seen in Lecture 3 that the eigenvalues of $-\Delta_n^{(d)}$ are given by

$$\lambda_{k_1,\ldots,k_d} = \lambda_{k_1} + \ldots + \lambda_{k_d}$$

where

$$\lambda_k \;=\; 2\,(n+1)^2 \left(\, 1 - \cos\left(\pi \tfrac{k}{n+1}\right) \right) \;\in\; \Big[\pi^2 + O(n^{-2}),\; 4\,(n+1)^2\Big].$$

The eigenvalues of $-\Delta_n^{(d)}$ are thus contained in the interval

$$\Big[\,\pi^2\, d + O(n^{-2}),\; 4d\,(n+1)^2\,\Big] \;\subset\; \big(0, \infty\big).$$

We can hence solve $-\Delta_n^{(d)} u_n = f$ using the conjugate gradients algorithm, and this method will converge exponentially with rate

$$\rho(\kappa) = \tfrac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1} \qquad \text{where} \qquad \kappa = \tfrac{4d\,(n+1)^2}{\pi^2\, d + O(n^{-2})} = O(n^2)$$

according to the result from slide 32.

Krylov Subspace Methods

Krylov methods for solving $-\Delta_n^{(d)} u_n = f$ (continued)

To achieve $\rho^k = \tau$, we must therefore choose

$$\begin{aligned} k &= \frac{\log(\tau)}{\log(r)} = O\big(\log(\tau)\,(1-r)^{-1}\big) = O\Big(\log(\tau)\,\big(1-\tfrac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\big)^{-1}\Big) \\ &= O\Big(\log(\tau)\,\big(\tfrac{\sqrt{\kappa}+1-\sqrt{\kappa}+1}{\sqrt{\kappa}+1}\big)^{-1}\Big) = O\Big(\log(\tau)\,\sqrt{\kappa}\Big) \\ &= O\big(\log(\tau)\,n\big), \end{aligned}$$

where in the second step I used that $\log(r)^{-1} = O\big((1-r)^{-1}\big)$ since

$$\lim_{r\to 1} \frac{\log(r)^{-1}}{(1-r)^{-1}} = \lim_{r\to 1} \frac{1-r}{\log(r)} = \lim_{r\to 1} \frac{-1}{1/r} = -1$$

according to L'Hôpital's rule.

Numerical demonstration

See `cg_poisson_*d()`.

Krylov Subspace Methods

Conjugate gradients for solving $-\Delta_n^{(d)} u_n = f$ (continued)

Combining the above number of iterations with the

$$O(k\,\mathrm{nnz}(A) + k\,\mathrm{length}(b)) \text{ runtime} \quad \text{and} \quad O(\mathrm{length}(b)) \text{ memory}$$

requirements for conjugate gradients, we obtain the following table.

	Runtime		**Memory**	
	LU	CG	LU	CG
$d = 1$	$O(N)$	$O(N^2)$	$O(N)$	$O(N)$
$d = 2$	$O(N^{3/2})$	$O(N^{3/2})$	$O(N \log(N))$	$O(N)$
$d = 3$	$O(N^2)$	$O(N^{4/3})$	$O(N^{4/3})$	$O(N)$

$N = n^d$ denotes the number of unknowns.

This shows that Krylov subspace methods are better than LU for solving the discrete Poisson equation in three dimensions.

Krylov Subspace Methods

Final words on Krylov subspace methods

The story of this lecture can be summarised as follows.

- Krylov subspace methods allow us to solve $Ax = b$ using only the matrix-vector product $v \mapsto Av$ and a few elementary vector operations.
- This circumstance allows us to solve sparse linear systems without fill-in and therefore often with a significantly smaller memory footprint compared to LU factorisation.
- Krylov subspace methods are not necessarily faster than LU factorisation. In particular, we have seen that conjugate gradients applied to $-\Delta_n^{(d)} u_n = f$ only beats LU factorisation for $d = 3$.

This indicates that choosing an algorithm for solving large, sparse linear systems requires a fairly complicated assessment of the properties of the linear system on the one hand and the available computational resources on the other.

Krylov Subspace Methods

Summary

Krylov subspace methods:

$$\text{Compute} \qquad x_n = p(A)\, b \qquad \text{where} \qquad p = \operatorname*{arg\,min}_{p \in \mathcal{P}_n} \|b - A\, p(A)\, b\|.$$

Concrete Krylov subspace methods:

	GMRES	**MinRes**	**Conjugate Gradients**
Residual norm	2-norm	2-norm	A^{-1} norm
Assumptions on A	none	symmetric	symmetric and positive definite
Runtime	$O(n\,\mathrm{nnz}(A) + n^2\,\mathrm{len}(b))$	$O(n\,\mathrm{nnz}(A) + n\,\mathrm{len}(b))$	$O(n\,\mathrm{nnz}(A) + n\,\mathrm{len}(b))$
Memory	$O(n\,\mathrm{len}(b))$	$O(\mathrm{len}(b))$	$O(\mathrm{len}(b))$

Krylov Subspace Methods

Summary (continued)

Convergence theory:

- Krylov subspace methods converge faster the more the eigenvalues cluster around a few points far away from 0.
- $p(A)\,b$ is exact if $\text{degree}(p) \geq [\#\text{ distinct eigenvalues}] - 1$.
- We can prove exponential convergence if the eigenvalues are contained in some interval $[c, d] \subset (0, \infty)$.

Performance tricks:

- Restarted GMRES for avoiding the quadratic runtime and linear memory of standard GMRES.
- Preconditioning for improving the speed of convergence.

Krylov Subspace Methods

Summary (continued)

Conjugate gradients vs LU factorisation for $-\Delta_n^{(d)} u_n = f$:

	Runtime		**Memory**	
	LU	CG	LU	CG
$d = 1$	$O(N)$	$O(N^2)$	$O(N)$	$O(N)$
$d = 2$	$O(N^{3/2})$	$O(N^{3/2})$	$O(N \log(N))$	$O(N)$
$d = 3$	$O(N^2)$	$O(N^{4/3})$	$O(N^{4/3})$	$O(N)$

$N = n^d$ denotes the number of unknowns.