Iterative methods

We've covered direct methods (LU for $Ax = b$, QR for $\min \|Ax - b\|_2$, QRalg for $Ax = \lambda x$). These are

- Incredibly reliable, backward stable
- Works like magic if $n \lesssim 10000$
- But not if n larger!

A 'big' matrix problem is one for which direct methods aren't feasible. Historically,

- 1950: $n \geq 20$
- 1965: $n \geq 200$
- 1980: $n \geq 2000$
- 1995: $n \geq 20000$
- 2010: $n \geq 100000$
- 2020: $n \geq 1000000$ ($n \geq 50000$ on a standard desktop)

was considered 'very large'. For such problems, we need to turn to alternative algorithms: we'll cover **iterative** and **randomised** methods.

Direct vs. iterative methods

Idea of iterative methods:

- gradually refine solution iteratively
- each iteration should be (a lot) cheaper than direct methods, usually $O(n^2)$ or less
- can be (but not always) much faster than direct methods
- tends to be (slightly) less robust, nontrivial/problem-dependent analysis
- often, after $O(n^3)$ work it still gets the exact solution (ignoring roundoff errors)

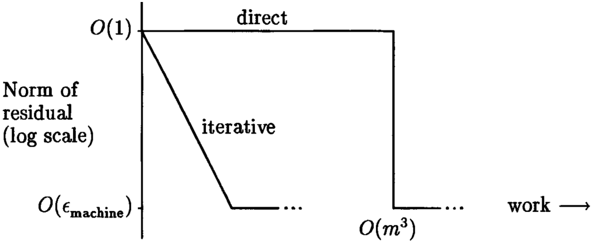


image from [Trefethen-Bau]

We'll focus on **Krylov subspace methods**.

Basic idea of Krylov: polynomial approximation

In Krylov subspace methods, we look for an (approximate) solution $\hat{x}$ (for $Ax = b$ or $Ax = \lambda x$) of the form (after kth iteration)

$$\boxed{\hat{x} = p_{k-1}(A)v},$$

where p_{k-1} is a polynomial of degree $k-1$, and $v \in \mathbb{R}^n$ arbitrary (usually $v = b$ for linsys, for eigenproblems v usually random)

Natural questions:

- Why would this be a good idea?
  - Clearly, 'easy' to compute
  - One example: recall power method $\hat{x} = A^{k-1}v = p_{k-1}(A)v$
    Krylov finds a "better/optimal" polynomial $p_{k-1}(A)$
  - We'll see more cases where Krylov is powerful
- How to turn into an algorithm?
  - Arnoldi (next), Lanczos

Orthonormal basis for $\mathcal{K}_k(A, b)$

Find approximate solution $\hat{x} = p_{k-1}(A)b$, i.e. in Krylov subspace

$$\mathcal{K}_k(A, b) := \text{span}([b, Ab, A^2b, \dots, A^{k-1}b])$$

First step: form an orthonormal basis Q, s.t. solution can be written as $x = Qy$

- Naive idea: Form matrix $[b, Ab, A^2b, \dots, A^{k-1}b]$, then QR
  - $[b, Ab, A^2b, \dots, A^{k-1}b]$ is usually terribly conditioned! Dominated by leading eigvec
  - Q is therefore extremely ill-conditioned, inaccurately computed

Orthonormal basis for $\mathcal{K}_k(A, b)$

Find approximate solution $\hat{x} = p_{k-1}(A)b$, i.e. in Krylov subspace

$$\mathcal{K}_k(A, b) := \text{span}([b, Ab, A^2b, \ldots, A^{k-1}b])$$

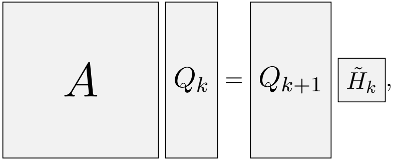
First step: form an orthonormal basis Q, s.t. solution can be written as $x = Qy$

- Naive idea: Form matrix $[b, Ab, A^2b, \ldots, A^{k-1}b]$, then QR
  - $[b, Ab, A^2b, \ldots, A^{k-1}b]$ is usually terribly conditioned! Dominated by leading eigvec
  - Q is therefore extremely ill-conditioned, inaccurately computed
- Much better solution: Arnoldi process
  - Multiply A once at a time to the latest orthonormal vector q_i
  - Then orthogonalise Aq_i against previous q_j's $(j = 1, \ldots, i-1)$ (as in Gram-Schmidt)

Arnoldi iteration

```
Set $q_1 = b/\|b\|_2$
For $k = 1, 2, \ldots,$
  set $v = Aq_k$
    for $j = 1, 2, \ldots, k$
      $h_{jk} = q_j^T v$, $v = v - h_{jk}q_j$ % orthogonalise against $q_j$ via modified G-S
    end for
      $h_{k+1,k} = \|v\|_2$, $q_{k+1} = v/h_{k+1,k}$
End for
```

- After k steps, $AQ_k = Q_{k+1}\tilde{H}_k = Q_k H_k + q_{k+1}[0, \ldots, 0, h_{k+1,k}]$, with $Q_k = [q_1, q_2, \ldots, q_k], Q_{k+1} = [Q_k, q_{k+1}]$, span$(Q_k)$ = span$([b, Ab, \ldots, A^{k-1}b])$



$$\tilde{H}_k = \underbrace{\begin{bmatrix} h_{1,1} & h_{1,2} & \ldots & h_{1,k} \\ h_{2,1} & h_{2,2} & \ldots & h_{2,k} \\ & \ddots & & \vdots \\ & & h_{k,k-1} & h_{k,k} \\ & & & h_{k+1,k} \end{bmatrix}}_{\mathbb{R}^{(k+1)\times k}\text{ upper Hessenberg}}, \quad Q_{k+1}^T Q_{k+1} = I_{k+1}$$

- Cost k A-multiplications$+O(k^2)$ inner products $(O(nk^2))$

Lanczos iteration

When A symmetric, Arnoldi simplifies to

$$AQ_k = Q_k T_k + q_{k+1}[0, \dots, 0, h_{k+1,k}],$$

where T_k is symmetric tridiagonal (proof: just note $H_k = Q_k^T A Q_k$ in Arnoldi)

$$\boxed{A}\,\boxed{Q_k} = \boxed{Q_{k+1}}\,\boxed{\tilde{T}_k}, \quad \tilde{T}_k = \underbrace{\begin{bmatrix} t_{1,1} & t_{1,2} & & \\ t_{2,1} & t_{2,2} & \ddots & \\ & \ddots & & t_{k-1,k} \\ & & t_{k,k-1} & t_{k,k} \\ & & & t_{k+1,k} \end{bmatrix}}_{\mathbb{R}^{(k+1)\times k} \text{ symmetric tridiagonal}}, \quad Q_{k+1}^T Q_{k+1} = I_{k+1}$$

- 3-term recurrence, orthogonalisation necessary only against last two vecs q_k, q_{k-1}
- Significant speedup over Arnoldi; cost k A-mult.$+O(k)$ inner products $(O(nk))$
- In floating-point arithmetic, sometimes computed Q_k lose orthogonality and reorthogonalisation necessary (nonexaminable)

The Lanczos algorithm for symmetric eigenproblem

Rayleigh-Ritz: given symmetric A and orthonormal Q, find approximate eigenpairs

1. Compute $Q^T AQ$
2. Eigenvalue decomposition $Q^T AQ = V\hat{\Lambda}V^T$
3. Approximate eigenvalues diag$(\hat{\Lambda})$ (Ritz values) and eigenvectors QV (Ritz vectors)

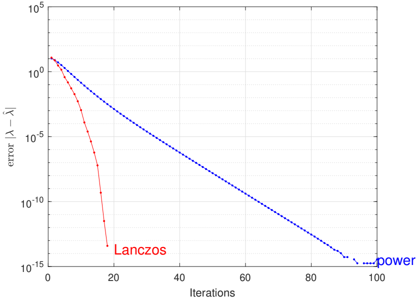
This is a **projection** method (similar alg. available for SVD)

Lanczos algorithm=Lanczos iteration+Rayleigh-Ritz

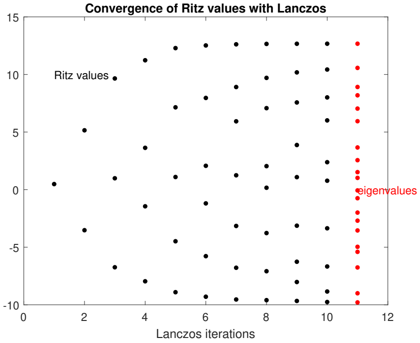
- In this case $Q = Q_k$, so simply $Q_k^T AQ_k = T_k$ (tridiagonal eigenproblem)
- Very good convergence to extremal eigenpairs
  - Recall from Courant-Fisher $\lambda_{\max}(A) = \max_x \frac{x^T Ax}{x^T x}$
  - Hence $\lambda_{\max}(A) \geq \underbrace{\max_{x \in \mathcal{K}_k(A,b)} \frac{x^T Ax}{x^T x}}_{\text{Lanczos output}} \geq \underbrace{\frac{v^T Av}{v^T v}, \quad v = A^{k-1}b}_{\text{power method}}$
  - Same for $\lambda_{\min}$, similar for e.g. λ_2

Experiments with Lanczos

Symmetric $A \in \mathbb{R}^{n\times n}, n = 100$, Lanczos/power method with random initial vector b



Convergence to dominant eigenvalue



Convergence of dominant eigenvalue