MA3227 Numerical Analysis II

Lecture 5: Krylov Subspace Methods

Simon Etter

National University of Singapore

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.

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 $O(\mathrm{nnz}(A))$ 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 $(A^k\,b)_{k=1}^k$ 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 $O(\mathrm{nnz}(A))$ 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:     $w[i] = w[i] + A[i, j]\, v[j]$
4: end for
```

- Scalar-vector products and vector sums can be evaluated in $O\big(\mathrm{length}(b)\big)$ operations, and we must have $\mathrm{length}(b) \leq \mathrm{nnz}(A)$ 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

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

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*.

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

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

given a matrix $A \in \mathbb{R}^{m \times m}$, a vector $b \in \mathbb{R}^m$ 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 $n = 0, 1, 2, \dots, n_{\max}$ 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 $\|b - Ax_n\|_2 \le \tau$ 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

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

can be computed by assembling

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

and setting

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

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

$\mathcal{K}_n(A, b) = \operatorname{span}\{b, Ab, \dots, A^{n-1}b\}$ 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

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

Krylov Subspace Methods

Now that we have settled the algorithmic details, let us move on to studying the runtime, memory and convergence of the GMRES algorithm.

Thm: Runtime of GMRES

The first n iterations of the GMRES algorithm from slide 11 require $O(n \,\mathrm{nnz}(A) + n^2 \,\mathrm{length}(b))$ runtime and $O(n \,\mathrm{length}(b))$ memory.

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

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

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 be continued]