Functions of large-scale matrices

How do we compute $f(A)$ if A is large and sparse? Huge recent research topic.

Most of the time, one wants $f(A)b$ rather than $f(A)$, because $f(A)$ is full (unless f and A are special, e.g., square of a banded matrix). Some of the main techniques:

1. Replace f with an approximating polynomial/rational function on a region U that includes the spectrum of A (how?).
2. Contour integration:

$$\frac{1}{2\pi i}\int_\Gamma f(z)(zI-A)^{-1}\mathrm{d}z \approx \sum_{k=1}^{n} w_k f(x_k)(x_k I - A)^{-1}.$$

3. Ad-hoc methods, involving e.g. discretization of differential equations: for instance, $\exp(A)b = v(1)$ where $\dot{v}(t) = Av(t)$, $v(0) = b$.

(And actually 2. and 3. are special cases of 1.)

Arnoldi for matrix functions

Another possibility with the "Swiss-army knife" algorithm for large matrices: Arnoldi.

Let us recap Arnoldi (with matrix functions in mind).

Krylov subspace

$$\begin{aligned} K_n(A, b) &= \operatorname{span}(b, Ab, A^2b, \ldots, A^{n-1}b) \\ &= \{p(A)b \colon p \text{ polynomial of degree } < n\}. \end{aligned}$$

Suppose we have computed the vectors $b, Ab, A^2b, \ldots, A^{n-1}b, A^nb$ explicitly.
This gives us a "recipe" to evaluate Av for any $v \in K_n(A, b)$:

$$\begin{aligned} Av &= A(\alpha_0 b + \alpha_1 Ab + \cdots + \alpha_{n-1}A^{n-1}b) = p(A)b \\ &= (\alpha_0 Ab + \alpha_1 A^2b + \cdots + \alpha_{n-1}A^nb) = Ap(A)b. \end{aligned}$$

Towards Arnoldi

This recipe often is not satisfying: $b, Ab, A^2b, \ldots A^{n-1}b$ converge to the leading eigenvector of A (power method), so when n gets large these vectors tend to be parallel (and often also huge/small).

Some tasks, e.g. determining $\alpha_0, \ldots, \alpha_{n-1}$ given v, are hopelessly ill-conditioned:

$$V_n = \begin{bmatrix} b & Ab & \ldots & A^{n-1}b \end{bmatrix}, \quad a = \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_{n-1} \end{bmatrix} = V_n^+ v.$$

(Moore–Penrose pseudoinverse).

It would be much better to work with an orthogonal basis of $\operatorname{Im} V_n = K_n(A, b)$, e.g., the Q factor of $V = QR$: then the coordinates are simply $a = V_n^* v$.

Arnoldi as "recipe"

The "recipe" for Av comes from the columns of

$$A \begin{bmatrix} b & Ab & \dots & A^{n-1}b \end{bmatrix} = \begin{bmatrix} b & Ab & \dots A^{n-1}b & A^n b \end{bmatrix} \begin{bmatrix} 0 & & & \\ 1 & \ddots & & \\ & \ddots & 0 & \\ & & 1 & 0 \\ & & & 1 \end{bmatrix}.$$

With (nested) orthonormal bases of V_n and V_{n+1}, it becomes

$$A \underbrace{\begin{bmatrix} v_1 & v_2 & \dots v_n \end{bmatrix}}_{V_n} = \underbrace{\begin{bmatrix} v_1 & v_2 & \dots & v_n & v_{n+1} \end{bmatrix}}_{V_{n+1}} \underline{H}_n$$

for a certain $\underline{H}_n \in \mathbb{C}^{(n+1)\times n}$ (to be determined).

Arnoldi iteration

We can compute the v's iteratively, one after the other: suppose we already have $v_1, \ldots, v_j$. Then,

$$Av_j = v_1\alpha_{1,j} + v_2\alpha_{2,j} + \cdots + v_j\alpha_{j-1,j} + v_{j+1}\alpha_{j+1,j}$$

Thanks to orthogonality, we can compute and subtract $v_i^* A v_j = \alpha_{i,j}$ for $i = 1, \ldots, j$ (Gram–Schmidt process), and we are left with $v_{j+1}\alpha_{j+1,j}$:

```
w = A*V(:,j);
for i = 1:j
        alpha(i,j) = V(:,i)' * w;
        w = w - V(:,i) * alpha(i,j);
end
alpha(j+1,j) = norm(w);
V(:,j+1) = w / alpha(j+1,j);
```

Starting point: $v_1 = \frac{b}{\|b\|}$.

Remarks on Arnoldi

This algorithm computes nested bases for the Krylov subspaces:

$$K_j(A, b) = \mathrm{Im}\begin{bmatrix} v_1 & v_2 & \dots & v_j \end{bmatrix}, \quad j = 1, 2, \dots$$

Why did we start with $w = Av_j$ and not another vector? Because this way we can prove that $\alpha_{j+1,j} \neq 0$.

Lemma

Suppose V_j has full column rank (so that the α's are uniquely determined). Then, $v_j \in K_j(A, b) \setminus K_{j-1}(A, b)$.

Proof Induction: assume this holds up to step j. Then, $v_j = p(A)b$ with p of degree exactly $j - 1$. Hence, $w = Av_j = q(A)b$ with $q(A) = Ap(A)$ of degree exactly j, and the same holds for $v_{j+1} = w - \sum_{i=1}^{j} \alpha_{ij} v_j$ (since we are subtracting polynomials of degree $< j$).

Arnoldi: the 'recipe'

Gathering all the relations involving the Av_j in a matrix, we get

$$A \underbrace{\begin{bmatrix} v_1 & \dots & v_n \end{bmatrix}}_{V_n} = \underbrace{\begin{bmatrix} v_1 & \dots & v_{n+1} \end{bmatrix}}_{V_{n+1}} \underbrace{\begin{bmatrix} \alpha_{1,1} & \alpha_{1,2} & \alpha_{1,3} & \dots & \alpha_{1,n} \\ \alpha_{1,2} & \alpha_{2,2} & \alpha_{2,3} & \dots & \alpha_{2,n} \\ 0 & \alpha_{3,2} & \alpha_{3,3} & \dots & \alpha_{3,n} \\ 0 & 0 & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \alpha_{n,n} \\ 0 & \dots & \dots & 0 & \alpha_{n+1,n} \end{bmatrix}}_{\underline{H}_n}.$$

When $\alpha_{n+1,n} = 0$ (breakdown), $AV_n = V_n H_n$, where H_n is $\underline{H}_n$ without the last row. This is an invariant subspace relation.

Note that $H_n = V_n^* A V_n$.

Formula for $p(A)b$

Lemma

For all polynomials with $\deg p < n$,

$$p(A)b = V_n p(H_n) V_n^* b = V_n p(H_n) e_1 \|b\|.$$

Proof: it is sufficient to show that $A^j b = V_n H_n^j V_n^* b$ for $j < n$.

$$V_n H_n^j V_n^* = V_n V_n^* A V_n V_n^* A \cdots V_n V_n^* A V_n V_n^* A V_n V_n^* b$$

Let us start from the right. $V_n V_n^*$ is the orthogonal projection matrix onto the Krylov space. Since $b \in K_n(A, b)$, $V_n V_n^* b = b$. Now the rightmost part reads $V_n V_n^* A b$; but this equals Ab because $Ab \in K_n(A, b)$, and so on.

Arnoldi, matrix functions, and polynomial approximations

Idea: let us take $c = V_n f(H_n) e_1 \|b\|$ as an approximation of $f(A)$. This approximation is exact when f is a polynomial of degree $< n$.

Moreover,

$$c = V_n f(H_n) e_1 \|b\| = V_n \tilde{p}(H_n) e_1 \|b\| = \tilde{p}(A) b,$$

where $\tilde{p}$ is the interpolating polynomial to f on the spectrum of H_n (not that of A!)

Known behaviour from Arnoldi theory: for many matrices, the eigenvalues of H_n (Ritz values) approximate the outermost eigenvalues of A.

What is going on: for a diagonalizable $A = W \Lambda W^{-1}$, we are computing $c = W \tilde{p}(\lambda) W^{-1} b$ instead of $f(A)b = W f(\lambda) W^{-1} b$;

- for eigenvalues "on the outside", $f(\lambda) \approx \tilde{p}(\lambda)$ because $f(\mu) = \tilde{p}(\mu)$ for a nearby Ritz value $\mu \in \Lambda(H)$.
- for eigenvalues "on the inside", they may be different, but hopefully $|f(\lambda)|$ is smaller and does not contribute too much.

A more precise error bound

Theorem

Let A be normal, and $\mathbb{W}(A) \subseteq \mathbb{C}$ be the convex hull of $\Lambda(A)$. Let $p(x)$ be the best-approximation polynomial to f in $\mathbb{W}(A)$, i.e., the one that minimizes $\delta = \max_{x \in \mathbb{W}(A)} |f(x) - p(x)|$. Then,

$$\|f(A)b - c\| \leq 2\delta \|b\|.$$

(And, magically, Arnoldi does all this without knowing p!)

Proof Note that the eigenvalues of H_n are in $\mathbb{W}(A)$: indeed, $Hx = \lambda x \implies \frac{x^* H x}{x^* x} = \frac{x^* V^* A V x}{x^* x}$ is a Rayleigh quotient for A. Since the Arnoldi approximation is exact on p,

$$\begin{aligned}
\|f(A)b - c\| &= \|f(A)b - V_n f(H_n) V_n^* b\| \\
&= \|(f-p)(A)b - V_n (f-p)(H_n) V_n^* b\| \\
&\leq \|(f-p)(A)b\| + \|V_n (f-p)(H_n) V_n^* b\| \\
&\leq \delta \|b\| + \delta \|b\|.
\end{aligned}$$

Note

A similar bound can be obtained also for non-normal A, by defining the field of values or numerical range

$$\mathbb{W}(A) = \left\{ \frac{x^* A x}{x^* x} \right\} = \{\text{set of Rayleigh quotients of } A\},$$

(which in general is larger than hull($\Lambda(A)$)), and using

Crouzeix-Palencia theorem

There is a universal constant $2 \leq C \leq 2\sqrt{2}$ (its optimal value is still an open problem) such that for any matrix A and function f

$$\|f(A)\| \leq C \max_{x \in \mathbb{W}(A)} |f(x)|.$$

Arnoldi variants [Güttel '13]

What if f takes its larger values at some internal point of the spectrum of A, e.g., $f(x) = \frac{1}{x}$ and A has both positive and negative eigenvalues (or complex values not all in the same half-plane)?

Idea: change the Arnoldi iteration!

- Extended Arnoldi: constructs an orthonormal basis for

  $$\{p(A)b\colon p = \alpha_{-n_1}x^{-n_1} + \alpha_{-n_1+1}x^{-n_1+1} + \cdots + \alpha_{n_2-1}x^{n_2-1}\}$$
  $$= A^{-n_1}K_{n_1+n_2}(A, b) = K_{n_1+n_2}(A, A^{-n_1}b);$$

  (Laurent polynomials)
- Rational Arnoldi: given $q_{n-1}(z)$ of degree $n-1$, o.n. basis for

  $$\{r(A)b\colon r(z) = p(z)/q_{n-1}(z),\ p \text{ any polynomial of degree } < n\}$$
  $$= q_{n-1}(A)^{-1}K_n(A, b) = K_n(A, q_{n-1}(A)^{-1}b);$$

  (rational functions with fixed denominator $q_{n-1}(z)$)

can be constructed, e.g., extended Krylov, i.e., Krylov on A and A^{-1} 'at the same time', or rational Krylov, which takes poles

Extended Arnoldi

Idea: at each step, start with suitable continuation vector $v \in \operatorname{Im} V_j$. Compute

- $w = A^{-1}v$ if you want to add a negative power of z to your space $\ell(A)b$
- $w = Av$ if you want to add a positive power of z to your space of Laurent polynomials of the form $\ell(A)b$,

and then orthogonalize it w.r.t. $v_1, \ldots, v_j$.
Only detail: the continuation vector must ensure that $\alpha_{j+1,j} \neq 0$.

A working choice: take the v_k from the last iteration k with a power of the same kind (positive or negative).

Putting together all orthogonalization relations yields a relation of the form

$$AV_{n+1}\underline{K}_n = V_{n+1}\underline{H}_n$$

This provides a 'recipe' to compute products Ax for every $x \in \operatorname{Im} V_n$.

Rational Arnoldi

Similar idea. At each step j, we start from a pole polynomial $q_{j-1}(z) = (z - \xi_1)(z - \xi_2)\dots(z - \xi_{j-1})$ of degree $j - 1$, and an orthonormal basis for

$$q_{j-1}(A)^{-1} K_j(A, b) = \left\{ r(A)b \colon r(z) = \frac{p(z)}{q_{j-1}(z)},\ \deg(p) < j \right\}.$$

We extend the space by adding a new pole ξ_j (possibly repeated).

We start from a suitable continuation vector v (typically one of the v_i), compute $w = (A - \xi_j I)^{-1} v$, and orthogonalize it against all previous basis vectors v_i.

With some minor changes, one can allow for poles ξ_j equal to ∞ (i.e., "traditional" Arnoldi).

(Usually one takes the last pole ξ_n to be ∞ (a traditional Arnoldi step), so the last row of $\underline{K}_n$ is 0 and $A_n = H_n K_n^{-1}$.)

Arnoldi approximation

Once one has computed a suitable approximation space V_n,

$$f(A)b \approx V_n f(A_n) V_n^* b, \quad A_n = V_n^* A V_n,$$

and the approximation is exact on Laurent polynomials (resp. rational functions with fixed denominators).

Analogous versions of the previous approximation results hold for extended/rational Arnoldi hold, replacing polynomials with Laurent polynomials or rational functions with fixed denominators. (Indeed, in the proofs we basically used only the property $A_n = V_n^* A V_n$ and the exactness property.)

Costs and benefits

Computational cost:

- Extended Arnoldi: typically computed with a (sparse) LU factorization of A, once, so that we can reuse it for each product with A^{-1}.
- Rational Arnoldi: typically computed with one sparse direct solve at each step. More degrees of freedom due to choice of poles. (Adaptively? From interpolation theory?)

Both are more expensive than Arnoldi.

Key issues: how much better is rational interpolation (for your f and A) than polynomial interpolation, so that the trade-off is convenient? How to choose good poles ξ_j?

Lots of current research on it. No details here. (I am not an expert myself!)
More detail in the review paper [Güttel '13].

Matlab examples

Using Rktoolbox by S. Güttel `http://guettel.com/rktoolbox/`.

```
>> rng(0); A = randn(100) + 10*eye(100);
>> v = eig(A); plot(real(v), imag(v), 'x');
>> b = randn(size(A,1), 1);
>> poles = -20:-1;
>> [V, K, H] = rat_krylov(A, b, poles);
>> A0 = K(1:end-1,:) \ H(1:end-1,:);
>> v = eig(A); w = eig(A0);
>> plot(real(v), imag(v), 'x', real(w), imag(w), 'o');
>> F = V(:, 1:end-1)*expm(A0) * V(:, 1:end-1)';
>> norm(expm(A) - F) / norm(expm(A))
```

Try again with other choice of poles, a symmetric matrix, regular Arnoldi (poles$=\infty$), extended Arnoldi (poles$=\infty$ and 0), ...