Quadrature-Based Arnoldi Restarts for Matrix Functions with Randomized Sketching

Jingyu Liu

February 6, 2026

1 Introduction

This note considers the computation of $f(\boldsymbol{A})\boldsymbol{b}$ where $\boldsymbol{A} \in \mathbb{C}^{N\times N}$ is a matrix, $\boldsymbol{b}$ is a vector in $\mathbb{C}^N$ and f is a suitable matrix function. Typically, $\boldsymbol{A}$ is non-Hermitian, large but sparse.

2 Quadrature-Based Arnoldi Restarts

A commonly used method is the Krylov subspace method. Suppose we have the Arnoldi decomposition $\boldsymbol{A}\boldsymbol{V}_m = \boldsymbol{V}_{m+1}\underline{\boldsymbol{H}}_m$ for the Krylov subspace $\mathcal{K}_m(\boldsymbol{A};\boldsymbol{b})$, where $\boldsymbol{V}_m \in \mathbb{C}^{N\times m}$ is orthonormal and $\underline{\boldsymbol{H}}_m \in \mathbb{C}^{(m+1)\times m}$ is upper-Hessenberg. The starting vector $\boldsymbol{v}_1$ is chosen to be $\boldsymbol{b}/\|\boldsymbol{b}\|$. If we define $\boldsymbol{H}_m = \boldsymbol{V}_m^*\boldsymbol{A}\boldsymbol{V}_m$, then the FOM approximation of $f(\boldsymbol{A})\boldsymbol{b}$ is given by

$$\boldsymbol{f}_m = \boldsymbol{V}_m f(\boldsymbol{H}_m)\boldsymbol{V}_m^*\boldsymbol{b} = \|\boldsymbol{b}\|\boldsymbol{V}_m f(\boldsymbol{H}_m)\boldsymbol{e}_1. \tag{2.1}$$

Our goal is to find a way to restart for the Krylov method. We first give a formula for the approximation error. Suppose $\Omega \subset \mathbb{C}$ is a region and $f\colon \Omega \to \mathbb{C}$ is analytic with the integral representation

$$f(z) = \int_\Gamma \frac{g(t)}{t-z}\mathrm{d}z, \quad z \in \Omega \tag{2.2}$$

where $\Gamma \subset \mathbb{C}\backslash\Omega$ is a closed contour and $g\colon \Gamma \to \mathbb{C}$ is a known suitable function. We have the following theorem.

Theorem 2.1 (Error of the Arnoldi approximation, [2])**.** *Suppose f has an integral representation as in* (2.2)*. Denote $\boldsymbol{f}_m$ the FOM approximation* (2.1) *to $f(\boldsymbol{A})\boldsymbol{b}$. Let* $\operatorname{spec}(\boldsymbol{H}_m) = \{\theta_1,\dots,\theta_m\} \subset \Omega$*, $\phi_m(t) = (t-\theta_1)\cdots(t-\theta_m)$ and $\gamma_m = \prod_{j=1}^m h_{j+1,j}$. Then, the error of the FOM approximation can be expressed as*

$$f(\boldsymbol{A})\boldsymbol{b} - \boldsymbol{f}_m = \gamma_m\|\boldsymbol{b}\|\int_\Gamma \frac{g(t)}{\phi_m(t)}(t\boldsymbol{I}-\boldsymbol{A})^{-1}\mathrm{d}t\boldsymbol{v}_{m+1} =: \mathrm{err}_m(\boldsymbol{A})\boldsymbol{v}_{m+1}, \tag{2.3}$$

provided the integral exists. Therefore, the error function is given by

$$\mathrm{err}_m(z) = \gamma_m\|\boldsymbol{b}\|\int_\Gamma \frac{g(t)}{\phi_m(t)}(t-z)^{-1}\mathrm{d}t. \tag{2.4}$$

The proof of this theorem is based on the following three results.

Lemma 2.2. *Let $\boldsymbol{f}_m$ be the FOM approximation (2.1) to $f(\boldsymbol{A})\boldsymbol{b}$. Then, $\boldsymbol{f}_m = p_{m-1}(\boldsymbol{A})\boldsymbol{b}$ where p_{m-1} is the unique polynomial of degree at most $m-1$ that interpolates f at the eigenvalues of $\boldsymbol{H}_m = \boldsymbol{V}^\dagger \boldsymbol{A}\boldsymbol{V}_m$.*

Lemma 2.3. *Suppose f has an integral representation as in (2.2). The interpolating polynomial p_{m-1} with interpolation points $\theta_1, \ldots, \theta_m \subset \Omega$ is given as*

$$p_{m-1}(z) = \int_\Gamma \left(1 - \frac{\phi_m(z)}{\phi_m(t)}\right) \frac{g(t)}{t-z} \mathrm{d}t, \tag{2.5}$$

where $\phi_m(z) = (z-\theta_1)\cdots(z-\theta_m)$, provided the integral in (2.5) exists.

Lemma 2.4. *Let $p(z) = c_0 + c_1 z + \cdots + c_m z^m$ be the unique polynomial of degree exact m. Then,*

$$p(\boldsymbol{A})\boldsymbol{b} = \|\boldsymbol{b}\|(\boldsymbol{V}_m p(\boldsymbol{H}_m)\boldsymbol{e}_1 + \boldsymbol{v}_{m+1}\gamma_m c_m).$$

Using $\boldsymbol{v}_{m+1}$ as the new starting vector, suppose we have a new Arnoldi decomposition

$$\boldsymbol{A}\boldsymbol{V}_m^+ = \boldsymbol{V}_{m+1}^+ \underline{\boldsymbol{H}}_m^+$$

with $\boldsymbol{v}_1^+ = \boldsymbol{v}_{m+1}$. The correction term can be computed through

$$\mathrm{err}_m(\boldsymbol{A})\boldsymbol{v}_{m+1} = \boldsymbol{V}_m^+ \mathrm{err}_m(\boldsymbol{H}_m^+)\boldsymbol{V}_m^{+,*}\boldsymbol{v}_{m+1} = \boldsymbol{V}_m^+ \mathrm{err}_m(\boldsymbol{H}_m^+)\boldsymbol{e}_1.$$

This gives a general framework for restarting the Arnoldi method for matrix functions, as stated in algorithm 1.

Algorithm 1 Restarted FOM approximation for $f(\boldsymbol{A})\boldsymbol{b}$

Input: Matrix $\boldsymbol{A}$, vector $\boldsymbol{b}$, function f, subspace dimension m.
Output: $\boldsymbol{f}_m^{[k]} \approx f(\boldsymbol{A})\boldsymbol{b}$.

1: Compute the Arnoldi decomposition $\boldsymbol{A}\boldsymbol{V}_m^{[0]} = \boldsymbol{V}_{m+1}^{[0]} \underline{\boldsymbol{H}}_m^{[0]}$ where $\boldsymbol{v}_1^{[0]} = \boldsymbol{b}/\|\boldsymbol{b}\|$.
2: $\boldsymbol{f}_m^{[0]} = \|\boldsymbol{b}\| \boldsymbol{V}_m^{[0]} f(\boldsymbol{H}_m^{[0]})\boldsymbol{e}_1$.
3: **for** $k = 1, 2, \ldots$ **do**
4: Determine the error function $\mathrm{err}_m^{[k-1]}(z)$.
5: Compute the Arnoldi decomposition $\boldsymbol{A}\boldsymbol{V}_m^{[k]} = \boldsymbol{V}_{m+1}^{[k]} \underline{\boldsymbol{H}}_m^{[k]}$ where $\boldsymbol{v}_1^{[k]} = \boldsymbol{v}_{m+1}^{[k-1]}$.
6: Update the approximation $\boldsymbol{f}_m^{[k]} = \boldsymbol{f}_m^{[k-1]} + \boldsymbol{V}_m^{[k]} \mathrm{err}_m^{[k-1]}(\boldsymbol{H}_m^{[k]})\boldsymbol{e}_1$.
7: **end for**

We have the following easy-to-verify corollary.

Corollary 2.5 ([2])**.** *Under the same assumptions as Theorem 2.1 after k ($k \geq 0$) restarts can be expressed as*

$$f(\boldsymbol{A})\boldsymbol{b} - \boldsymbol{f}_m^{[k]} = \gamma_m^{[0]} \cdots \gamma_m^{[k]} \|\boldsymbol{b}\| \int_\Gamma \frac{g(t)}{\phi_m^{[0]}(t)\cdots\phi_m^{[k]}(t)} (t\boldsymbol{I} - \boldsymbol{A})^{-1} \boldsymbol{v}_{m+1}^{[k]} \mathrm{d}t =: \mathrm{err}_m^{[k]}(\boldsymbol{A})\boldsymbol{v}_{m+1}^{[k]}.$$

Numerically, the computation in terms of err_m refers to a quadrature formula

$$\widehat{\mathrm{err}}_m(z) = \gamma_m \|\boldsymbol{b}\| \sum_{i=1}^{\ell} w_i \frac{g(t_i)}{\phi_m(t_i)} \frac{1}{t_i - z}.$$

Combing all these together, we have the quadrature-based restart Arnoldi approximation as in algorithm 2. One important feature of this algorithm is that the number of quadrature points ℓ can be adapted during the iterations to achieve a desired accuracy.

Algorithm 2 Quadrature-based restart Arnoldi approximation for $f(\boldsymbol{A})\boldsymbol{b}$

Input: Matrix $\boldsymbol{A}$, vector $\boldsymbol{b}$, function f, subspace dimension m, initial number of quadrature points ℓ, quadrature tolerance τ.
Output: $\boldsymbol{f}_m^{[k]} \approx f(\boldsymbol{A})\boldsymbol{b}$.

Compute the Arnoldi decomposition $\boldsymbol{A}\boldsymbol{V}_m^{[0]} = \boldsymbol{V}_{m+1}^{[0]} \underline{\boldsymbol{H}}_m^{[0]}$ where $\boldsymbol{v}_1^{[0]} = \boldsymbol{b}/\|\boldsymbol{b}\|$.
$\boldsymbol{f}_m^{[0]} = \|\boldsymbol{b}\| \boldsymbol{V}_m^{[0]} f(\boldsymbol{H}_m^{[0]}) \boldsymbol{e}_1$.
$\tilde{\ell} = \lceil \ell/\sqrt{2} \rceil$.
for $k = 1, 2, \ldots$ **do**
  Determine the error function $\mathrm{err}_m^{[k-1]}(z)$ according to Corollary 2.5.
  Compute the Arnoldi decomposition $\boldsymbol{A}\boldsymbol{V}_m^{[k]} = \boldsymbol{V}_{m+1}^{[k]} \underline{\boldsymbol{H}}_m^{[k]}$ where $\boldsymbol{v}_1^{[k]} = \boldsymbol{v}_{m+1}^{[k-1]}$.
  Choose quadrature points and weights $\{(t_i, w_i)\}_{i=1}^{\ell}$ and $\{(\tilde{t}_i, \tilde{w}_i)\}_{i=1}^{\tilde{\ell}}$.
  Set accurate = false and refined = false.
  while accurate = false **do**
    Compute $\boldsymbol{h}_m^{[k]} = \widehat{\mathrm{err}}_m^{[k-1]}(\boldsymbol{H}_m^{[k]}) \boldsymbol{e}_1$ using $\{(t_i, w_i)\}_{i=1}^{\ell}$.
    Compute $\tilde{\boldsymbol{h}}_m^{[k]} = \widehat{\mathrm{err}}_m^{[k-1]}(\boldsymbol{H}_m^{[k]}) \boldsymbol{e}_1$ using $\{(\tilde{t}_i, \tilde{w}_i)\}_{i=1}^{\tilde{\ell}}$.
    if $\|\boldsymbol{h}_m^{[k]} - \tilde{\boldsymbol{h}}_m^{[k]}\| \le \tau \|\tilde{\boldsymbol{h}}_m^{[k]}\|$ **then**
      accurate = true.
    else
      refined = true.
      Update $\tilde{\ell} = \ell$, $\ell = \lceil \sqrt{2}\tilde{\ell} \rceil$.
    end if
  end while
  Update the approximation $\boldsymbol{f}_m^{[k]} = \boldsymbol{f}_m^{[k-1]} + \boldsymbol{V}_m^{[k]} \boldsymbol{h}_m^{[k]}$.
  if refined = false **then**
    Update $\ell = \tilde{\ell}$, $\tilde{\ell} = \lceil \ell/\sqrt{2} \rceil$.
  end if
end for

3 Randomized Sketching for Computing Matrix Functions

Consider the computation of $f(\boldsymbol{A})\boldsymbol{b}$. Using the integral definition of matrix functions, we have

$$f(\boldsymbol{A})\boldsymbol{b} = \frac{1}{2\pi \mathrm{i}} \int_C f(t)(t\boldsymbol{I} - \boldsymbol{A})^{-1} \boldsymbol{b} \mathrm{d}t = \frac{1}{2\pi \mathrm{i}} \int_C f(t) \boldsymbol{x}(t) \mathrm{d}t,$$

where C is a closed contour enclosing $\mathrm{spec}(\boldsymbol{A})$ and $\boldsymbol{x}(t)$ is the solution to the shifted linear system $(t\boldsymbol{I}-\boldsymbol{A})\boldsymbol{x}(t)=\boldsymbol{b}$.

Using this integral representation, the FOM can be expressed as

$$\boldsymbol{f}_m = \|\boldsymbol{b}\|\boldsymbol{V}_m f(\boldsymbol{H}_m)\boldsymbol{e}_1 = \|\boldsymbol{b}\|\boldsymbol{V}_m \frac{1}{2\pi\mathrm{i}}\int_C f(t)(t\boldsymbol{I}-\boldsymbol{H}_m)^{-1}\boldsymbol{e}_1\mathrm{d}t = \frac{1}{2\pi\mathrm{i}}\int_C f(t)\boldsymbol{x}_m(t)\mathrm{d}t,$$

where $\boldsymbol{x}_m(t)=\|\boldsymbol{b}\|\boldsymbol{V}_m(t\boldsymbol{I}-\boldsymbol{H}_m)^{-1}\boldsymbol{e}_1$. In fact, $\boldsymbol{x}_m(t)$ is the FOM approximation to the shifted linear system $(t\boldsymbol{I}-\boldsymbol{A})\boldsymbol{x}(t)=\boldsymbol{b}$. To see this, suppose $\boldsymbol{x}(t)$ is given by $\boldsymbol{x}_m(t)=\boldsymbol{V}_m\boldsymbol{y}_m(t)$ where $\boldsymbol{y}_m(t)$ is to be determined. Define the residual $\boldsymbol{r}_m(t)=\boldsymbol{b}-(t\boldsymbol{I}-\boldsymbol{A})\boldsymbol{x}_m(t)$. The Galerkin condition requires that the residual is orthogonal to $\mathcal{K}_m(\boldsymbol{A};\boldsymbol{b})$, i.e., $\boldsymbol{V}_m^*\boldsymbol{r}_m(t)=0$. This leads to $\boldsymbol{y}_m(t)=\|\boldsymbol{b}\|(t\boldsymbol{I}-\boldsymbol{H}_m)^{-1}\boldsymbol{e}_1$.

Suppose now we have a basis (not necessarily orthonormal) $\boldsymbol{B}_m\in\mathbb{C}^{N\times m}$ for the Krylov subspace $\mathcal{K}_m(\boldsymbol{A};\boldsymbol{b})$. We want to approximate $\boldsymbol{x}(t)$ by an vector $\boldsymbol{x}_m(t)=\boldsymbol{B}_m\boldsymbol{y}_m(t)$. The Galerkin condition gives

$$\begin{aligned}
\boldsymbol{y}_m(t) &= \big[(t\boldsymbol{B}_m^*\boldsymbol{B}_m-\boldsymbol{B}_m^*\boldsymbol{A}\boldsymbol{B}_m)\big]^{-1}\boldsymbol{B}_m^*\boldsymbol{b}\\
&= \big[(t\boldsymbol{I}-(\boldsymbol{B}_m^*\boldsymbol{B}_m)^{-1}\boldsymbol{B}_m^*\boldsymbol{A}\boldsymbol{B}_m)\big]^{-1}(\boldsymbol{B}_m^*\boldsymbol{B}_m)^{-1}\boldsymbol{B}_m^*\boldsymbol{b}\\
&= \big[(t\boldsymbol{I}-\boldsymbol{B}_m^\dagger\boldsymbol{A}\boldsymbol{B}_m)\big]^{-1}\boldsymbol{B}_m^\dagger\boldsymbol{b}
\end{aligned}$$

and

$$\begin{aligned}
\boldsymbol{f}_m &= \frac{1}{2\pi\mathrm{i}}\int_C f(t)\boldsymbol{x}_m(t)\mathrm{d}t\\
&= \frac{1}{2\pi\mathrm{i}}\int_C f(t)\boldsymbol{B}_m\big[(t\boldsymbol{I}-\boldsymbol{B}_m^\dagger\boldsymbol{A}\boldsymbol{B}_m)\big]^{-1}\boldsymbol{B}_m^\dagger\boldsymbol{b}\mathrm{d}t\\
&= \boldsymbol{B}_m f(\boldsymbol{B}_m^\dagger\boldsymbol{A}\boldsymbol{B}_m)\boldsymbol{B}_m^\dagger\boldsymbol{b}
\end{aligned}\tag{FOM}$$

We use a randomized sketching technique to give another approximation for $\boldsymbol{x}(t)$. Suppose it is given by $\hat{\boldsymbol{x}}_m(t)=\boldsymbol{B}_m\hat{\boldsymbol{y}}_m(t)$ where $\hat{\boldsymbol{y}}_m(t)$ is to be determined. Let $\boldsymbol{S}\in\mathbb{C}^{s\times N}$ $(m\le s\ll N)$ be a sketching matrix such that for some $\varepsilon\in[0,1)$ for all vectors $\boldsymbol{v}$ in $\mathcal{K}_{m+1}(\boldsymbol{A};\boldsymbol{b})$,

$$(1-\varepsilon)\|\boldsymbol{v}\|^2\le\|\boldsymbol{S}\boldsymbol{v}\|^2\le(1+\varepsilon)\|\boldsymbol{v}\|^2.\tag{3.1}$$

Unlike the Arnoldi approximation, instead of requiring the residual $\hat{\boldsymbol{r}}_m(t)=\boldsymbol{b}-(t\boldsymbol{I}-\boldsymbol{A})\hat{\boldsymbol{x}}_m(t)$ to be orthogonal to the Krylov subspace $\mathcal{K}_m(\boldsymbol{A};\boldsymbol{b})$, we require the sketched residual $\boldsymbol{S}\hat{\boldsymbol{r}}_m(t)$ to be orthogonal to the sketched Krylov subspace $\boldsymbol{S}\mathcal{K}_m(\boldsymbol{A};\boldsymbol{b})$, i.e., $(\boldsymbol{S}\boldsymbol{B}_m)^*\boldsymbol{S}\hat{\boldsymbol{r}}_m(t)=0$. This gives

$$\hat{\boldsymbol{y}}_m(t)=\big[(\boldsymbol{S}\boldsymbol{B}_m)^*(t\boldsymbol{S}\boldsymbol{B}_m-\boldsymbol{S}\boldsymbol{A}\boldsymbol{B}_m)\big]^{-1}(\boldsymbol{S}\boldsymbol{B}_m)^*(\boldsymbol{S}\boldsymbol{b})$$

and

$$\hat{\boldsymbol{f}}_m=\frac{1}{2\pi\mathrm{i}}\int_C\hat{\boldsymbol{x}}_m(t)\mathrm{d}t=\boldsymbol{B}_m\frac{1}{2\pi\mathrm{i}}\int_C f(t)\big[(\boldsymbol{S}\boldsymbol{B}_m)^*(t\boldsymbol{S}\boldsymbol{B}_m-\boldsymbol{S}\boldsymbol{A}\boldsymbol{B}_m)\big]^{-1}(\boldsymbol{S}\boldsymbol{B}_m)^*(\boldsymbol{S}\boldsymbol{b})\mathrm{d}t.\tag{sFOM}$$

Suppose $\boldsymbol{S}\boldsymbol{B}_m$ has full column rank (and this is always true), then

$$\begin{aligned}
\big[(\boldsymbol{S}\boldsymbol{B}_m)^*(t\boldsymbol{S}\boldsymbol{B}_m-\boldsymbol{S}\boldsymbol{A}\boldsymbol{B}_m)\big]^{-1} &= \big[t(\boldsymbol{S}\boldsymbol{B}_m)^*(\boldsymbol{S}\boldsymbol{B}_m)-(\boldsymbol{S}\boldsymbol{B}_m)^*(\boldsymbol{S}\boldsymbol{A}\boldsymbol{B}_m)\big]^{-1}\\
&= \Big\{t\boldsymbol{I}-\big[(\boldsymbol{S}\boldsymbol{B}_m)^*(\boldsymbol{S}\boldsymbol{B}_m)\big]^{-1}(\boldsymbol{S}\boldsymbol{B}_m)^*(\boldsymbol{S}\boldsymbol{A}\boldsymbol{B}_m)\Big\}^{-1}\big[(\boldsymbol{S}\boldsymbol{B}_m)^*(\boldsymbol{S}\boldsymbol{B}_m)\big]^{-1}\\
&= \Big\{t\boldsymbol{I}-(\boldsymbol{S}\boldsymbol{B}_m)^\dagger(\boldsymbol{S}\boldsymbol{A}\boldsymbol{B}_m)\Big\}^{-1}\big[(\boldsymbol{S}\boldsymbol{B}_m)^*(\boldsymbol{S}\boldsymbol{B}_m)\big]^{-1}
\end{aligned}$$

Now,

$$\begin{aligned}\hat{\boldsymbol{f}}_m &= \boldsymbol{B}_m \frac{1}{2\pi \mathrm{i}} \int_C f(t) \Big\{ t\boldsymbol{I} - (\boldsymbol{S}\boldsymbol{B}_m)^\dagger (\boldsymbol{S}\boldsymbol{A}\boldsymbol{B}_m) \Big\}^{-1} \big[(\boldsymbol{S}\boldsymbol{B}_m)^* (\boldsymbol{S}\boldsymbol{B}_m) \big]^{-1} \mathrm{d}t \\ &\quad (\boldsymbol{S}\boldsymbol{B}_m)^* (\boldsymbol{S}\boldsymbol{b}) \qquad \text{(sFOM}')\\ &= \boldsymbol{B}_m f\Big((\boldsymbol{S}\boldsymbol{B}_m)^\dagger (\boldsymbol{S}\boldsymbol{A}\boldsymbol{B}_m) \Big) (\boldsymbol{S}\boldsymbol{B}_m)^\dagger (\boldsymbol{S}\boldsymbol{b})\end{aligned}$$

Moreover, if $\boldsymbol{S}\boldsymbol{B}_m$ is orthonormal, that is, $(\boldsymbol{S}\boldsymbol{B}_m)^*(\boldsymbol{S}\boldsymbol{B}_m) = \boldsymbol{I}$, then (sFOM$'$) reduces to

$$\hat{\boldsymbol{f}}_m = \boldsymbol{B}_m f\Big((\boldsymbol{S}\boldsymbol{B}_m)^* (\boldsymbol{S}\boldsymbol{A}\boldsymbol{B}_m) \Big) (\boldsymbol{S}\boldsymbol{B}_m)^* (\boldsymbol{S}\boldsymbol{b}). \qquad \text{(sFOM}'')$$

When $\boldsymbol{S}\boldsymbol{B}_m$ is not orthonormal, we can use the thin QR decomposition $\boldsymbol{S}\boldsymbol{B}_m = \boldsymbol{Q}_m \boldsymbol{R}_m$ where $\boldsymbol{Q}_m \in \mathbb{C}^{s\times m}$ has orthonormal columns and $\boldsymbol{R}_m \in \mathbb{C}^{m\times m}$ is upper-triangular. Substituting this into (sFOM$'$), we have

$$\hat{\boldsymbol{f}}_m = \boldsymbol{B}_m f\Big(\boldsymbol{R}_m^{-1} \boldsymbol{Q}_m^* (\boldsymbol{S}\boldsymbol{A}\boldsymbol{B}_m) \Big) \boldsymbol{R}_m^{-1} \boldsymbol{Q}_m^* (\boldsymbol{S}\boldsymbol{b}). \qquad \text{(sFOM}''')$$

It is worth noting that **JINGYU: this expression is a bit different from [3]. This is because I adjust the position of the inverse matrix and the current version is more easy to understand and analyze.**

The above discussions are summarized in algorithm 3.

Algorithm 3 Sketched FOM approximation for $f(\boldsymbol{A})\boldsymbol{b}$

Input: Matrix $\boldsymbol{A}$, vector $\boldsymbol{b}$, function f, subspace dimension m.
Output: $\boldsymbol{f}_m^{[k]} \approx f(\boldsymbol{A})\boldsymbol{b}$.

1: Draw a sketching matrix $\boldsymbol{S} \in \mathbb{C}^{s\times N}$.
2: Generate a basis $\boldsymbol{B}_m \in \mathbb{C}^{N\times m}$ for the Krylov subspace $\mathcal{K}_m(\boldsymbol{A}; \boldsymbol{b})$ as well as $\boldsymbol{S}\boldsymbol{B}_m$ and $\boldsymbol{S}\boldsymbol{A}\boldsymbol{B}_m$.
3: **if** $\boldsymbol{S}\boldsymbol{B}_m$ is orthonormal **then**
4: Compute the sketched FOM approximation $\hat{\boldsymbol{f}}_m$ using (sFOM$''$).
5: **else**
6: Compute the thin QR decomposition $\boldsymbol{S}\boldsymbol{B}_m = \boldsymbol{Q}_m \boldsymbol{R}_m$.
7: Compute the sketched FOM approximation $\hat{\boldsymbol{f}}_m$ using (sFOM$'''$).
8: **end if**

We give two strategies to generate the basis $\boldsymbol{B}_m$. The first one is to use the truncated Arnoldi process, as illustrated in Algorithm 4. The second one is based on the Gram–Schmidt process on the sketched vectors. This is illustrated in Algorithm 5. Note that in Algorithm 5, we also compute $\boldsymbol{S}\boldsymbol{B}_m$ and $\boldsymbol{S}\boldsymbol{A}\boldsymbol{B}_m$ during the process. Interestingly, in these two algorithms, we also have an Arnoldi-like decomposition, which will be discussed in the next section.

As for the error analysis of the sketched FOM approximant, we have the following result:

Theorem 3.1 (Error of the sketched FOM approximant)**.** *Suppose $\boldsymbol{S}$ is a sketching matrix of $\mathcal{K}_{m+1}(\boldsymbol{A}; \boldsymbol{b})$ satisfying* (3.1) *with some $\varepsilon \in [0, 1)$ and that $\boldsymbol{S}\boldsymbol{B}_m$ is orthonormal. Let $\boldsymbol{f}_m$ and $\hat{\boldsymbol{f}}_m$*

Algorithm 4 Truncated Arnoldi process

Input: Matrix $\boldsymbol{A}$, vector $\boldsymbol{b}$, subspace dimension m, truancation parameter t.
Output: Basis $\boldsymbol{B}_m \in \mathbb{C}^{N\times m}$ for the Krylov subspace $\mathcal{K}_m(\boldsymbol{A};\boldsymbol{b})$.

$\boldsymbol{b}_1 = \boldsymbol{b}/\|\boldsymbol{b}\|$.
for $j = 1, 2, \ldots, m$ **do**
  $\boldsymbol{w}_j = \boldsymbol{A}\boldsymbol{b}_j$.
  for $i = \max(1, j-t+1), \ldots, j$ **do**
    $h_{i,j} = \boldsymbol{b}_i^* \boldsymbol{w}_j$.
    $\boldsymbol{w}_j = \boldsymbol{w}_j - \boldsymbol{b}_i h_{i,j}$.
  end for
  $h_{j+1,j} = \|\boldsymbol{w}_j\|$.
  if $h_{j+1,j} = 0$ **then**
    Stop.
  end if
  $\boldsymbol{v}_{j+1} = \boldsymbol{w}_j / h_{j+1,j}$.
end for
$\boldsymbol{V}_m = [\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_m]$.

Algorithm 5 Sketched Gram–Schmidt process

Input: Matrix $\boldsymbol{A}$, vector $\boldsymbol{b}$, subspace dimension m, sketching dim s.
Output: Basis $\boldsymbol{B}_m \in \mathbb{C}^{N\times m}$ for the Krylov subspace $\mathcal{K}_m(\boldsymbol{A};\boldsymbol{b})$.

Draw a sketching matrix $\boldsymbol{S} \in \mathbb{C}^{s\times N}$.
$\boldsymbol{w}_0 = \boldsymbol{b}$, $\boldsymbol{p}_0 = \boldsymbol{S}\boldsymbol{w}_0$
$\beta = \|\boldsymbol{p}_0\|$.
$\boldsymbol{q}_1 = \boldsymbol{p}_0/\beta$, $\boldsymbol{b}_1 = \boldsymbol{w}_0/\beta$
for $j = 1, 2, \ldots, m$ **do**
  $\boldsymbol{w}_j = \boldsymbol{A}\boldsymbol{b}_j$, $\boldsymbol{p}_j = \boldsymbol{S}\boldsymbol{w}_j$.
  for $i = 1, \ldots, j$ **do**
    $h_{i,j} = \boldsymbol{q}_i^* \boldsymbol{p}_j$.
    $\boldsymbol{p}_j = \boldsymbol{p}_j - \boldsymbol{q}_i h_{i,j}$, $\boldsymbol{w}_j = \boldsymbol{w}_j - \boldsymbol{b}_i h_{i,j}$.
  end for
  $h_{j+1,j} = \|\boldsymbol{p}_j\|$.
  if $h_{j+1,j} = 0$ **then**
    Stop.
  end if
  $\boldsymbol{q}_{j+1} = \boldsymbol{p}_j / h_{j+1,j}$, $\boldsymbol{b}_{j+1} = \boldsymbol{w}_j / h_{j+1,j}$.
end for
$\boldsymbol{B}_m = [\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_m]$, $\boldsymbol{S}\boldsymbol{B}_m = [\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_m]$, $\boldsymbol{S}\boldsymbol{A}\boldsymbol{B}_m = [\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_m]$.

denote the FOM and sketched FOM approximants for $f(\boldsymbol{A})\boldsymbol{b}$, respectively, as defined in (FOM) *and* (sFOM$''$)*. Then,*

$$\|\boldsymbol{f}_m - \hat{\boldsymbol{f}}_m\| \le \sqrt{\frac{1+\varepsilon}{1-\varepsilon}}\|\boldsymbol{b}\|\big\|f(\boldsymbol{B}_m^\dagger \boldsymbol{A}\boldsymbol{B}_m) - f\big((\boldsymbol{S}\boldsymbol{B}_m)^*\boldsymbol{S}\boldsymbol{A}\boldsymbol{B}_m\big)\big\|.$$

4 The Arnoldi-like decomposition

In this section, we give some discussions on the Arnoldi-like decomposition. Suppose we have the following relationship:

$$\boldsymbol{A}\boldsymbol{B}_m = \boldsymbol{B}_m\boldsymbol{H}_m + \boldsymbol{b}_{m+1}h_{m+1,m}\boldsymbol{e}_m^\top = \boldsymbol{B}_{m+1}\underline{\boldsymbol{H}}_m, \tag{4.1}$$

where $\boldsymbol{B}_m \in \mathbb{C}^{N\times m}$ is a basisfor the Krylov subspace $\mathcal{K}_m(\boldsymbol{A};\boldsymbol{b})$. Then, we have the following lemma, which is an analogue of Lemma 2.4

Lemma 4.1. *Suppose* (4.1) *holds. Let $p(z) = c_0 + c_1 z + \cdots + c_m z^m$ be the unique polynomial of degree exact m. Then,*

$$p(\boldsymbol{A})\boldsymbol{b}_1 = \boldsymbol{B}_m p(\boldsymbol{H}_m)\boldsymbol{e}_1 + \boldsymbol{b}_{m+1}\gamma_m c_m,$$

where $\gamma_m = \prod_{j=1}^m h_{j+1,j}$.

It is remarkable that if we require $\boldsymbol{b}_{m+1}$ to be orthogonal to $\boldsymbol{B}_m$, then $\boldsymbol{B}_m^\dagger\boldsymbol{A}\boldsymbol{B}_m = \boldsymbol{H}_m$. That is, the matrix $\boldsymbol{H}_m$ in the Arnoldi-like decomposition is exactly the projection of $\boldsymbol{A}$ onto the Krylov subspace $\mathcal{K}_m(\boldsymbol{A};\boldsymbol{b})$. Note that in this case, the FOM approximant $\boldsymbol{f}_m$ defined in (FOM) can be expressed as $\boldsymbol{f}_m = \boldsymbol{B}_m f(\boldsymbol{H}_m)\boldsymbol{B}_m^\dagger\boldsymbol{b}$.

Under this condition, we can give an analogue result of Theorem 2.1.

Theorem 4.2 (Error of the Arnoldi approximation)**.** *Let* (4.1) *be a given Arnoldi-like decomposition where $\boldsymbol{B}_m$ is a basis for the Krylov subspace $\mathcal{K}_m(\boldsymbol{A};\boldsymbol{b})$. Assume $\boldsymbol{b} = \beta\boldsymbol{b}_1$ and $\boldsymbol{b}_{m+1}$ is orthogonal to $\boldsymbol{B}_m$. Suppose f has an integral representation as in* (2.2)*. Denote $\boldsymbol{f}_m = \boldsymbol{B}_m f(\boldsymbol{H}_m)\boldsymbol{B}_m^\dagger\boldsymbol{b}$ the FOM approximation to $f(\boldsymbol{A})\boldsymbol{b}$. Let $\operatorname{spec}(\boldsymbol{H}_m) = \{\theta_1,\dots,\theta_m\} \subset \Omega$, $\phi_m(t) = (t-\theta_1)\cdots(t-\theta_m)$ and $\gamma_m = \prod_{j=1}^m h_{j+1,j}$. Then, the error of Arnoldi approximation can be expressed as*

$$f(\boldsymbol{A})\boldsymbol{b} - \boldsymbol{f}_m = \gamma_m\beta\int_\Gamma \frac{g(t)}{\phi_m(t)}(t\boldsymbol{I}-\boldsymbol{A})^{-1}\mathrm{d}t\boldsymbol{b}_{m+1} =: \mathrm{err}_m(\boldsymbol{A})\boldsymbol{b}_{m+1}. \tag{4.2}$$

provided the integral exists. Therefore, the error function is given by

$$\mathrm{err}_m(z) = \gamma_m\|\boldsymbol{b}\|\int_\Gamma \frac{g(t)}{\phi_m(t)}(t-z)^{-1}\mathrm{d}t.$$

Proof. Using Lemma 2.2, suppose p_{m-1} is the unique polynomial of degree at most $m-1$ that interpolates f at the eigenvalues of $\boldsymbol{H}_m$, then $\boldsymbol{f}_m = p_{m-1}(\boldsymbol{A})\boldsymbol{b}$. Using Lemma 2.3, we have

$$p_{m-1}(z) = \int_\Gamma\left(1 - \frac{\phi_m(z)}{\phi_m(t)}\right)\frac{g(t)}{t-z}\mathrm{d}t,$$

where $\phi_m(z) = (z-\theta_1)\cdots(z-\theta_m)$. Therefore,

$$f(z) - p_{m-1}(z) = \int_\Gamma \frac{\phi_m(z)}{\phi_m(t)}\frac{g(t)}{t-z}\mathrm{d}t$$

and

$$f(\boldsymbol{A})\boldsymbol{b} - \boldsymbol{f}_m = \int_\Gamma \frac{\phi_m(\boldsymbol{A})}{\phi_m(t)}(t\boldsymbol{I} - \boldsymbol{A})^{-1} g(t)\mathrm{d}t\boldsymbol{b}.$$

Note that by Lemma 4.1, we have $\phi_m(\boldsymbol{A})\boldsymbol{b} = \beta\phi_m(\boldsymbol{A})\boldsymbol{b}_1 = \boldsymbol{b}_{m+1}\gamma_m\beta$. Then,

$$f(\boldsymbol{A})\boldsymbol{b} - \boldsymbol{f}_m = \gamma_m\beta \int_\Gamma \frac{g(t)}{\phi_m(t)}(t\boldsymbol{I} - \boldsymbol{A})^{-1}\mathrm{d}t\boldsymbol{b}_{m+1}.$$

This completes the proof. □

Corollary 4.3. *Under the same assumptions as Theorem 4.2 after k ($k \geq 0$) restarts can be expressed as*

$$f(\boldsymbol{A})\boldsymbol{b} - \boldsymbol{f}_m^{[k]} = (\gamma_m^{[0]} \cdots \gamma_m^{[k]})(\beta^{[0]} \cdots \beta^{[k]}) \int_\Gamma \frac{g(t)}{\phi_m^{[0]}(t) \cdots \phi_m^{[k]}(t)}(t\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{b}_{m+1}^{[k]}\mathrm{d}t =: \mathrm{err}_m^{[k]}(\boldsymbol{A})\boldsymbol{b}_{m+1}^{[k]},$$

where we assume $\boldsymbol{b} = \beta^{[0]}\boldsymbol{b}_1^{[0]}$, $\boldsymbol{b}_{m+1}^{[j-1]} = \beta^{[j]}\boldsymbol{b}_1^{[j]}$ for $1 \leq j \leq k$, and that $\boldsymbol{b}_{m+1}^{[j]}$ is orthogonal to $\boldsymbol{B}_m^{[j]}$ for $0 \leq j \leq k$.

5 Quadrature-Based Arnoldi Restarts for Matrix Functions with Randomized Sketching

In Algorithm 2, using Algorithm 3 to compute the Arnoldi approximant in each cycle, we can incorporate randomized sketching into the quadrature-based Arnoldi restart framework. That is, we use randomized sketching to compute the FOM approximant $\boldsymbol{h}_m^{[k]}$ for each matrix function in Algorithm 2.

6 Numerical Experiments

Through all experiments, we set the sketching dimension to be roughly $1.2m$ where $m = 30$ is the dimension of the Krylov subspace. For the truncated Arnoldi process, we set the truncation parameter to be 5. We use the Gaussian random matrix as the sketching matrix $\boldsymbol{S}$. The tolerance for quadrature rule is set to be 10^{-7} and the stopping accuracy for the restart loop is set to be 10^{-8}. The maximum number of restart is set to be 15 and the maximum number of quadrature point updates is set to be 1024. We use Algorithm 2 as a benchmark, comparing both runtime and accuracy.

Here are some notations of the methods:

- FOM-t: using the truncated Arnoldi process to generate the basis $\boldsymbol{B}_m$ and compute the FOM approximant.
- FOM-s: using the sketched Gram–Schmidt process to generate the basis $\boldsymbol{B}_m$ and compute the FOM approximant.
- sFOM-t: using the truncated Arnoldi process to generate the basis $\boldsymbol{B}_m$ and compute the sketched FOM approximant.
- sFOM-s: using the sketched Gram–Schmidt process to generate the basis $\boldsymbol{B}_m$ and compute the sketched FOM approximant.

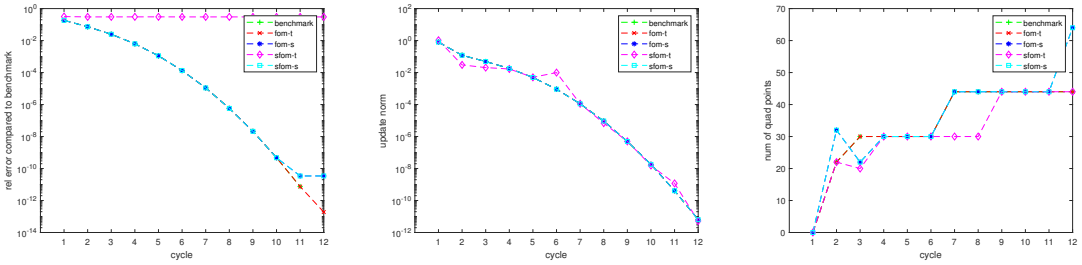
6.1 Matrix Exponential

The second example is from Section 5.3 of [2], where the matrix function is given by $f(\boldsymbol{A}) = \exp(s\boldsymbol{A})$ and the matrices $\boldsymbol{A} \in \mathbb{R}^{N\times N}$ correspond to the standard finite difference discretization of a two-dimensional convection-diffusion operator on the unit square with constant convection field and different convection parameters ν. The case $\nu = 0$ corresponds to a symmetric problem and for increasing ν the non-normal matrix $\boldsymbol{A}$ has eigenvalues with large imaginary parts. We choose $s = 2 \cdot 10^{-3}$ and use 500 discretization points in both spatial directions, resulting in a matrix of size $N = 500^2$.

The relavent results are shown in Table 6.1 and Figure 6.1.

Table 6.1: Results for $\nu = 1$.

Method	total iter	rel error (to benchmark)	time
benchmark	12	0.0000e+00	7.5840e+00
FOM-t	12	1.8963e-13	3.7249e+00
sFOM-t	12	3.0167e-01	4.2954e+00
FOM-s	12	3.4316e-11	5.1359e+00
sFOM-s	12	3.4323e-11	5.1527e+00



(a) Relative error to benchmark. (b) Norm of update. (c) Number of quadrature points.

Figure 6.1: Results for matrix exponential.

6.2 Matrix Inverse Square Root

The matrix function is given by $f(\boldsymbol{A}) = \boldsymbol{A}^{-1/2}$ and the matrices $\boldsymbol{A} \in \mathbb{R}^{N\times N}$ correspond to the 2d Laplacian plus some non-Hermitian matrix. We use 500 discretization points in both spatial directions, resulting in a matrix of size $N = 500^2$.

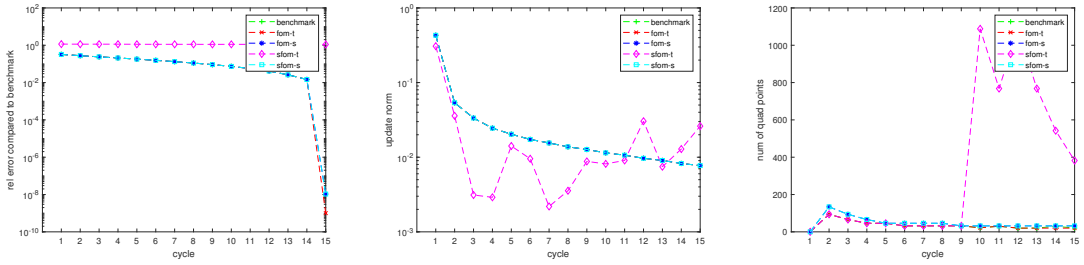
The relavent results are shown in Table 6.2 and Figure 6.2.

6.3 Matrix Log

The matrix function is given by $f(\boldsymbol{A}) = \boldsymbol{A}^{-1/2}$ and the matrices $\boldsymbol{A} \in \mathbb{R}^{N\times N}$ correspond to the 2d Laplacian plus some non-Hermitian matrix. We use 500 discretization points in both spatial directions, resulting in a matrix of size $N = 500^2$.

Table 6.2: Results for $\nu = 1$.

Method	total iter	rel error (to benchmark)	time
benchmark	15	0.0000e+00	9.1982e+00
FOM-t	15	1.0072e-09	4.6174e+00
sFOM-t	15	1.1073e+00	5.5324e+00
FOM-s	15	1.0606e-08	6.3656e+00
sFOM-s	15	1.0608e-08	6.4689e+00



(a) Relative error to benchmark. (b) Norm of update. (c) Number of quadrature points.

Figure 6.2: Results for matrix inverse square root.

The relavent results are shown in Table 6.3 and Figure 6.3.

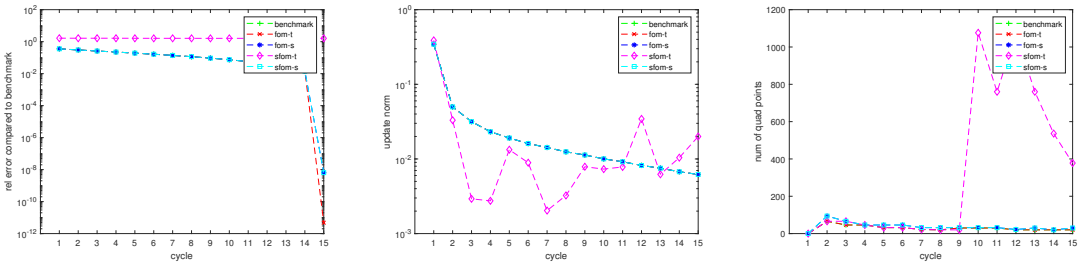
Table 6.3: Results for $\nu = 1$.

Method	total iter	rel error (to benchmark)	time
benchmark	15	0.0000e+00	9.2247e+00
FOM-t	15	4.7997e-12	4.5907e+00
sFOM-t	15	1.6143e+00	6.2550e+00
FOM-s	15	6.4539e-09	6.3414e+00
sFOM-s	15	6.4540e-09	6.3217e+00

7 What is next?

The current code is quite naive and need to be further improved. Some issues that need to be addressed:

- In the current implementation, the restarting vector for the k-th restart is still $\boldsymbol{v}_{m+1}^{[k-1]}$.
- In the current implementation, we still use the returned $\boldsymbol{H}_m$ by Algorithm 4 or 5 to compute the error function.



(a) Relative error to benchmark. (b) Norm of update. (c) Number of quadrature points.

Figure 6.3: Results for matrix logarithm.

- In the current implementation, we use the same sketching matrix for all restarts. It may be better to use different sketching matrices for different restarts.

The first two issues may bring problems since Theorem 2.1 may not hold. I think we need to develop a corresponding version for nonorthonormal basis.

References

[1] A. Cortinovis, D. Kressner, and Y. Nakatsukasa, *Speeding up krylov subspace methods for computing $f(A)b$ via randomization*, SIAM J. Matrix Anal. Appl., 45 (2024), pp. 619–633.

[2] A. Frommer, S. Güttel, and M. Schweitzer, *Efficient and stable Arnoldi restarts for matrix functions based on quadrature*, SIAM J. Matrix Anal. Appl., 35 (2014), pp. 661–683.

[3] S. Güttel and M. Schweitzer, *Randomized sketching for Krylov approximations of large-scale matrix functions*, SIAM J. Matrix Anal. Appl., 44 (2023), pp. 1073–1095.