Quadrature-Based Arnoldi Restarts for Matrix Functions with Randomized Sketching

Jingyu Liu

February 19, 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

$$\boldsymbol{A}\boldsymbol{V}_m = \boldsymbol{V}_{m+1}\underline{\boldsymbol{H}}_m = \boldsymbol{V}_m\boldsymbol{H}_m + \boldsymbol{v}_{m+1}h_{m+1,m}\boldsymbol{e}_m^\top \tag{2.1}$$

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 Arnoldi approximation of $f(\boldsymbol{A})\boldsymbol{b}$ is given by

$$\boldsymbol{f}_m = \boldsymbol{V}_m f(\boldsymbol{H}_m)\boldsymbol{V}_m^*\boldsymbol{b} = \|\boldsymbol{b}\|\boldsymbol{V}_m f(\boldsymbol{H}_m)\boldsymbol{e}_1. \tag{2.2}$$

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

$$f(z) = \int_\Gamma \frac{g(t)}{t-z}\mathrm{d}z, \quad z \in \Omega \tag{2.3}$$

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* (2.3)*. Denote $\boldsymbol{f}_m$ the Arnoldi approximation* (2.2) *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 $\boldsymbol{f}_m$ can be expressed as*

$$f(\boldsymbol{A})\boldsymbol{b} - \boldsymbol{f}_m = \gamma_m\|\boldsymbol{b}\|\int_\Gamma \frac{g(t)}{\phi_m(t)}(t\boldsymbol{I}-\boldsymbol{A})^{-1}\mathrm{d}t\boldsymbol{v}_{m+1} =: \operatorname{err}_m(\boldsymbol{A})\boldsymbol{v}_{m+1}, \tag{2.4}$$

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

$$\operatorname{err}_m(z) = \gamma_m\|\boldsymbol{b}\|\int_\Gamma \frac{g(t)}{\phi_m(t)}(t-z)^{-1}\mathrm{d}t. \tag{2.5}$$

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

Lemma 2.2. *Let $\boldsymbol{f}_m$ be the Arnoldi approximation (2.2) 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 $\boldsymbol{H}_m$.*

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

$$p_{m-1}(z) = \int_\Gamma \left(1 - \frac{\phi_m(z)}{\phi_m(t)}\right) \frac{g(t)}{t-z} \mathrm{d}t, \tag{2.6}$$

where $\phi_m(z) = (z-\theta_1)\cdots(z-\theta_m)$, provided the integral in (2.6) 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 Arnoldi approximantion 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 Arnoldi 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

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

and

$$\begin{aligned}
\boldsymbol{f}_m &= \frac{1}{2\pi\mathrm{i}}\int_C f(t)\boldsymbol{x}_m(t)\mathrm{d}t\\
&= \frac{1}{2\pi\mathrm{i}}\int_C f(t)\boldsymbol{B}_m\left[(t\boldsymbol{I}-\boldsymbol{B}_m^\dagger\boldsymbol{A}\boldsymbol{B}_m)\right]^{-1}\boldsymbol{B}_m^\dagger\boldsymbol{b}\mathrm{d}t\\
&= \boldsymbol{B}_m f(\boldsymbol{B}_m^\dagger\boldsymbol{A}\boldsymbol{B}_m)\boldsymbol{B}_m^\dagger\boldsymbol{b}
\end{aligned} \tag{3.1}$$

This is called the FOM approximation. When $\boldsymbol{B}_m=\boldsymbol{V}_m$ is orthonormal, it can be directly found that the FOM approximation becomes the Arnoldi approximantion

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 $\mathcal{K}_{m+1}(\boldsymbol{A};\boldsymbol{b})$,

$$(1-\varepsilon)\|\boldsymbol{v}\|^2 \le \|\boldsymbol{S}\boldsymbol{v}\|^2 \le (1+\varepsilon)\|\boldsymbol{v}\|^2. \tag{3.2}$$

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

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

and

$$\hat{\boldsymbol{f}}_m = \frac{1}{2\pi\mathrm{i}}\int_C \hat{\boldsymbol{x}}_m(t)\mathrm{d}t = \boldsymbol{B}_m\frac{1}{2\pi\mathrm{i}}\int_C f(t)\left[(\boldsymbol{S}\boldsymbol{B}_m)^*(t\boldsymbol{S}\boldsymbol{B}_m - \boldsymbol{S}\boldsymbol{A}\boldsymbol{B}_m)\right]^{-1}(\boldsymbol{S}\boldsymbol{B}_m)^*(\boldsymbol{S}\boldsymbol{b})\mathrm{d}t. \tag{3.3}$$

This is called the sketched FOM approximantion of $f(\boldsymbol{A})\boldsymbol{b}$.

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

$$\begin{aligned}\left[(\boldsymbol{SB}_m)^*(t\boldsymbol{SB}_m - \boldsymbol{SAB}_m)\right]^{-1} &= \left[t(\boldsymbol{SB}_m)^*(\boldsymbol{SB}_m) - (\boldsymbol{SB}_m)^*(\boldsymbol{SAB}_m)\right]^{-1} \\ &= \left\{t\boldsymbol{I} - \left[(\boldsymbol{SB}_m)^*(\boldsymbol{SB}_m)\right]^{-1}(\boldsymbol{SB}_m)^*(\boldsymbol{SAB}_m)\right\}^{-1}\left[(\boldsymbol{SB}_m)^*(\boldsymbol{SB}_m)\right]^{-1} \\ &= \left\{t\boldsymbol{I} - (\boldsymbol{SB}_m)^\dagger(\boldsymbol{SAB}_m)\right\}^{-1}\left[(\boldsymbol{SB}_m)^*(\boldsymbol{SB}_m)\right]^{-1}\end{aligned}$$

Now,

$$\begin{aligned}\hat{\boldsymbol{f}}_m = \boldsymbol{B}_m \frac{1}{2\pi \mathrm{i}} \int_C f(t)\left\{t\boldsymbol{I} - (\boldsymbol{SB}_m)^\dagger(\boldsymbol{SAB}_m)\right\}^{-1} \mathrm{d}t \\ \left[(\boldsymbol{SB}_m)^*(\boldsymbol{SB}_m)\right]^{-1}(\boldsymbol{SB}_m)^*(\boldsymbol{Sb}) \\ = \boldsymbol{B}_m f\Big((\boldsymbol{SB}_m)^\dagger(\boldsymbol{SAB}_m)\Big)(\boldsymbol{SB}_m)^\dagger(\boldsymbol{Sb})\end{aligned} \tag{3.4}$$

Moreover, if $\boldsymbol{SB}_m$ is orthonormal, then (3.4) reduces to

$$\hat{\boldsymbol{f}}_m = \boldsymbol{B}_m f\Big((\boldsymbol{SB}_m)^*(\boldsymbol{SAB}_m)\Big)(\boldsymbol{SB}_m)^*(\boldsymbol{Sb}). \tag{3.5}$$

When $\boldsymbol{SB}_m$ is not orthonormal, we can use the thin QR decomposition $\boldsymbol{SB}_m = \boldsymbol{Q}_m\boldsymbol{R}_m$ where $\boldsymbol{Q}_m \in \mathbb{C}^{s\times m}$ is orthonormal and $\boldsymbol{R}_m \in \mathbb{C}^{m\times m}$ is upper-triangular. Substituting this into (3.4), we have

$$\hat{\boldsymbol{f}}_m = \boldsymbol{B}_m f\Big(\boldsymbol{R}_m^{-1}\boldsymbol{Q}_m^*(\boldsymbol{SAB}_m)\Big)\boldsymbol{R}_m^{-1}\boldsymbol{Q}_m^*(\boldsymbol{Sb}). \tag{3.6}$$

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 $\boldsymbol{SB}_m$ and $\boldsymbol{SAB}_m$.
3: **if** $\boldsymbol{SB}_m$ is orthonormal **then**
4: Compute the sketched FOM approximation $\hat{\boldsymbol{f}}_m$ using (3.5).
5: **else**
6: Compute the thin QR decomposition $\boldsymbol{SB}_m = \boldsymbol{Q}_m\boldsymbol{R}_m$.
7: Compute the sketched FOM approximation $\hat{\boldsymbol{f}}_m$ using (3.6).
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 $\boldsymbol{SB}_m$ and $\boldsymbol{SAB}_m$ during the process. Interestingly, in these two algorithms, we also have an Arnoldi-like decomposition, which will be discussed in the next section.

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
  $\boldsymbol{b}_{j+1} = \boldsymbol{w}_j / h_{j+1,j}$.
end for
$\boldsymbol{B}_m = [\boldsymbol{b}_1, \ldots, \boldsymbol{b}_m]$.

Algorithm 5 Sketched Arnoldi 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
$\boldsymbol{B}_m = [\boldsymbol{b}_1, \ldots, \boldsymbol{b}_m]$, $\boldsymbol{SB}_m = [\boldsymbol{q}_1, \ldots, \boldsymbol{q}_m]$, $\boldsymbol{SAB}_m = [\boldsymbol{p}_1, \ldots, \boldsymbol{p}_m]$.

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* (3.2) *with some $\varepsilon \in [0,1)$ and that $\boldsymbol{S}\boldsymbol{B}_m$ is orthonormal. Let $\boldsymbol{f}_m$ and $\hat{\boldsymbol{f}}_m$ denote the FOM and sketched FOM approximants for $f(\boldsymbol{A})\boldsymbol{b}$, respectively, as defined in* (3.1) *and* (3.5)*. 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 basis of the Krylov subspace $\mathcal{K}_m(\boldsymbol{A};\boldsymbol{b})$ and $\boldsymbol{H}_m$ is upper-Hessenberg. Then, we have the following lemma, which is an analogue of Lemma 2.4

Lemma 4.1. *Suppose an Arnoldi-like decomposition* (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 $\gamma_m = \prod_{j=1}^m h_{j+1,j}$.

Recall that the FOM approximantion (3.1) is $\boldsymbol{f}_m = \boldsymbol{B}_m f(\boldsymbol{G}_m)\boldsymbol{B}_m^\dagger\boldsymbol{b}$ where $\boldsymbol{G}_m = \boldsymbol{B}_m^\dagger\boldsymbol{A}\boldsymbol{B}_m$. It is remarkable that if we require $\boldsymbol{b}_{m+1}$ to be orthogonal to $\boldsymbol{B}_m$, then $\boldsymbol{G}_m = \boldsymbol{H}_m$. That is, the matrix $\boldsymbol{H}_m$ in the Arnoldi-like decomposition is exactly the projection of $\boldsymbol{A}$ onto the Krylov subspace $\mathcal{K}_m(\boldsymbol{A};\boldsymbol{b})$. In this case, the FOM approximant can be expressed as

$$\boldsymbol{f}_m = \boldsymbol{B}_m f(\boldsymbol{H}_m)\boldsymbol{B}_m^\dagger\boldsymbol{b}. \tag{4.2}$$

Suppose p_{m-1} is the unique polynomial of degree at most $m-1$ that interpolates f at the eigenvalues of $\boldsymbol{G}_m$, then $f(\boldsymbol{G}_m) = p_{m-1}(\boldsymbol{G}_m)$. Assume that $\boldsymbol{b} = \boldsymbol{b}_1\beta$, then by Lemma 4.1, we have

$$\begin{aligned}\boldsymbol{f}_m &= \boldsymbol{B}_m f(\boldsymbol{G}_m)\boldsymbol{B}_m^\dagger\boldsymbol{b} = \boldsymbol{B}_m f(\boldsymbol{G}_m)\boldsymbol{e}_1\beta = \boldsymbol{B}_m p_{m-1}(\boldsymbol{G}_m)\boldsymbol{e}_1\beta\\ &= \boldsymbol{B}_m p_{m-1}(\boldsymbol{H}_m)\boldsymbol{e}_1\beta = p_{m-1}(\boldsymbol{A})\boldsymbol{b}_1\beta = p_{m-1}(\boldsymbol{A})\boldsymbol{b}.\end{aligned}$$

Using this formula, we have an analogue result of Theorem 2.1.

Theorem 4.2 (Error of the FOM when $\boldsymbol{b}_{m+1}$ is orthogonal to $\boldsymbol{B}_m$)**.** *Let* (4.1) *be a given Arnoldi-like decomposition where $\boldsymbol{B}_m$ is a basis for the Krylov subspace $\mathcal{K}_m(\boldsymbol{A};\boldsymbol{b})$ and $\boldsymbol{b} = \boldsymbol{b}_1\beta$. Suppose f has an integral representation as in* (2.3)*. Denote $\boldsymbol{f}_m = \boldsymbol{B}_m f(\boldsymbol{G}_m)\boldsymbol{B}_m^\dagger\boldsymbol{b}$ the FOM approximation to $f(\boldsymbol{A})\boldsymbol{b}$ where $\boldsymbol{G}_m = \boldsymbol{B}_m^\dagger\boldsymbol{A}\boldsymbol{B}_m$. Let $\operatorname{spec}(\boldsymbol{G}_m) = \{\theta_1,\dots,\theta_m\} \subset \Omega$, $\phi_m(t) = (t-\theta_1)\cdots(t-\theta_m)$ and $\gamma_m = \prod_{j=1}^m h_{j+1,j}$. If $\boldsymbol{b}_{m+1} \perp \boldsymbol{B}_m$, then the error of FOM approximation can be expressed as*

$$f(\boldsymbol{A})\boldsymbol{b} - \boldsymbol{f}_m = \gamma_m\beta\int_\Gamma \frac{g(t)}{\phi_m(t)}(t\boldsymbol{I}-\boldsymbol{A})^{-1}\mathrm{d}t\boldsymbol{b}_{m+1} =: \mathrm{err}_m(\boldsymbol{A})\boldsymbol{b}_{m+1}. \tag{4.3}$$

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

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

Proof. By 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

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

Again by Lemma 4.1, we have $\phi_m(\boldsymbol{A})\boldsymbol{b} = \phi_m(\boldsymbol{A})\boldsymbol{b}_1\beta = \boldsymbol{B}_m\phi_m(\boldsymbol{H}_m)\boldsymbol{e}_1\beta + \boldsymbol{b}_{m+1}\gamma_m\beta$. But since $\boldsymbol{b}_{m+1} \perp \boldsymbol{B}_m$, we have $\boldsymbol{H}_m = \boldsymbol{G}_m$ and thus $\phi_m(\boldsymbol{H}_m) = \boldsymbol{0}$. 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 $0 \leq j \leq k$.

The corresponding functions about the above discussions are:

- "funm_quad_fom_last_orth_tarnoldi"
- "funm_quad_fom_last_orth_sarnoldi"

That is, after generating a basis $\boldsymbol{B}$ using the truncated Arnoldi process (Algorithm 4) or sketched Arnoldi process (Algorithm 5), we can perform an additional step to orthogonalize $\boldsymbol{b}_{m+1}$ against $\boldsymbol{B}_m$ and then compute the FOM approximant using (4.2).

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

Suppose we have a Arnoldi-like decomposition as in (4.1) where $\boldsymbol{B}_m$ is a basis for the Krylov subspace $\mathcal{K}_m(\boldsymbol{A}; \boldsymbol{b})$ and $\boldsymbol{b} = \boldsymbol{b}_1\beta$. Instead of requiring $\boldsymbol{b}_{m+1}$ to be orthogonal to $\boldsymbol{B}_m$, we require the sketched vector $\boldsymbol{S}\boldsymbol{b}_{m+1}$ to be orthogonal to the sketched Krylov subspace $\boldsymbol{S}\mathcal{K}_m(\boldsymbol{A}; \boldsymbol{b})$, i.e., $(\boldsymbol{S}\boldsymbol{B}_m)^*\boldsymbol{S}\boldsymbol{b}_{m+1} = \boldsymbol{0}$.

Consider now the sFOM approximant defined in (3.4). Denote $\boldsymbol{G}_m = (\boldsymbol{SB}_m)^\dagger(\boldsymbol{SAB}_m)$. Using

$$\boldsymbol{SAB}_m = \boldsymbol{SB}_m\boldsymbol{H}_m + \boldsymbol{Sb}_{m+1}h_{m+1,m}\boldsymbol{e}_m^\top$$

and $\boldsymbol{Sb}_{m+1} \perp \boldsymbol{SB}_m$, we have $\boldsymbol{G}_m = \boldsymbol{H}_m$. Therefore, the sFOM approximant reduces to

$$\hat{\boldsymbol{f}}_m = \boldsymbol{B}_m f(\boldsymbol{H}_m)(\boldsymbol{SB}_m)^\dagger(\boldsymbol{Sb}) = \boldsymbol{B}_m f(\boldsymbol{H}_m)\boldsymbol{e}_1\beta. \tag{5.1}$$

Suppose p_{m-1} is the unique polynomial of degree at most $m-1$ that interpolates f at the eigenvalues of $\boldsymbol{G}_m$, then $f(\boldsymbol{G}_m) = p_{m-1}(\boldsymbol{G}_m)$. Then using Lemma 4.1 again, we have

$$\hat{\boldsymbol{f}}_m = \boldsymbol{B}_m f(\boldsymbol{G}_m)\boldsymbol{e}_1\beta = \boldsymbol{B}_m p_{m-1}(\boldsymbol{G}_m)\boldsymbol{e}_1\beta = \boldsymbol{B}_m p_{m-1}(\boldsymbol{H}_m)\boldsymbol{e}_1\beta = p_{m-1}(\boldsymbol{A})\boldsymbol{b}_1\beta = p_{m-1}(\boldsymbol{A})\boldsymbol{b}.$$

Again, we can give an analogue result of Theorem 4.2 for the sFOM approximant.

Theorem 5.1 (Error of the sFOM approximation when $\boldsymbol{Sb}_{m+1}$ is orthogonal to $\boldsymbol{SB}_m$)**.** *Let* (4.1) *be a given Arnoldi-like decomposition where $\boldsymbol{B}_m$ is a basis for the Krylov subspace $\mathcal{K}_m(\boldsymbol{A};\boldsymbol{b})$ and $\boldsymbol{b} = \beta\boldsymbol{b}_1$. Suppose f has an integral representation as in* (2.3)*. Denote $\hat{\boldsymbol{f}}_m = \boldsymbol{B}_m f(\boldsymbol{G}_m)\boldsymbol{e}_1\beta$ the sFOM approximation to $f(\boldsymbol{A})\boldsymbol{b}$ where $\boldsymbol{G}_m = (\boldsymbol{SB}_m)^\dagger(\boldsymbol{SAB}_m)$. Let* $\operatorname{spec}(\boldsymbol{G}_m) = \{\theta_1,\dots,\theta_m\} \subset \Omega$*, $\phi_m(t) = (t-\theta_1)\cdots(t-\theta_m)$ and $\gamma_m = \prod_{j=1}^m h_{j+1,j}$. If $\boldsymbol{Sb}_{m+1} \perp \boldsymbol{SB}_m$, then the error of Arnoldi approximation can be expressed as*

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

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

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

Proof. 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

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

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

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

This completes the proof. □

5.1 Truncated Arnoldi Case

If $\boldsymbol{B}_m$ is generated by a truncated-Arnoldi process, we do not necessarily have $\boldsymbol{S}\boldsymbol{b}_{m+1} \perp \boldsymbol{S}\boldsymbol{B}_m$. Therefore, we conduct an additional orthogonalization step. Suppose $\boldsymbol{c} \in \mathbb{C}^m$ such that $\boldsymbol{b}_{m+1} = \hat{\boldsymbol{b}}_{m+1}\alpha + \boldsymbol{B}_m\boldsymbol{c}$, such that $\hat{\boldsymbol{b}}_{m+1} \perp \boldsymbol{S}\boldsymbol{B}_m$. Such $\boldsymbol{c}$ can be obtained by $\boldsymbol{c} = (\boldsymbol{S}\boldsymbol{B}_m)^\dagger(\boldsymbol{S}\boldsymbol{b}_{m+1})$. Then the Arnoldi-like decomposition (4.1) can be modifed as

$$\begin{aligned}\boldsymbol{A}\boldsymbol{B}_m &= \boldsymbol{B}_m\boldsymbol{H}_m + \boldsymbol{b}_{m+1}h_{m+1,m}\boldsymbol{e}_m^\top \\ &= \boldsymbol{B}_m\boldsymbol{H}_m + (\hat{\boldsymbol{b}}_{m+1} + \boldsymbol{B}_m\boldsymbol{c})h_{m+1,m}\boldsymbol{e}_m^\top \\ &= \boldsymbol{B}_m(\boldsymbol{H}_m + \boldsymbol{c}h_{m+1,m}\boldsymbol{e}_m^\top) + \hat{\boldsymbol{b}}_{m+1}h_{m+1,m}\boldsymbol{e}_m^\top \\ &= \boldsymbol{B}_m\hat{\boldsymbol{H}}_m + \hat{\boldsymbol{b}}_{m+1}h_{m+1,m}\boldsymbol{e}_m^\top.\end{aligned} \tag{5.2}$$

Then, the sFOM approximation is

$$\hat{\boldsymbol{f}}_m = \boldsymbol{B}_m f(\hat{\boldsymbol{H}}_m)(\boldsymbol{S}\boldsymbol{B}_m)^\dagger(\boldsymbol{S}\boldsymbol{b}) = \boldsymbol{B}_m f(\hat{\boldsymbol{H}}_m)\boldsymbol{e}_1\beta. \tag{5.3}$$

5.2 Sketched Arnoldi Case

If $\boldsymbol{B}_m$ is generated by a truncated-Arnoldi process, we already have $\boldsymbol{S}\boldsymbol{b}_{m+1} \perp \boldsymbol{S}\boldsymbol{B}_m$. Thus we can use the matrix $\boldsymbol{H}_m$ in the Arnoldi-like decomposition 4.1 directly and

$$\hat{\boldsymbol{f}}_m = \boldsymbol{B}_m f(\boldsymbol{H}_m)\boldsymbol{e}_1\beta. \tag{5.4}$$

The corresponding functions about the above discussions are:

- "funm_quad_sfom_last_orth_tarnoldi"
- "funm_quad_sfom_last_orth_sarnoldi"

6 Monitoring Basis Using Random Sketching

The idea of this method is, instead of fixing the dimension m of the Krylov space, we determine m adaptively. More specifically, we generate a basis $\boldsymbol{B}_m$ adaptively by the truncated Arnoldi process. An sketching matrix $\boldsymbol{S}$ is chosen to monitor the condition number of the skecthed basis $\boldsymbol{S}\boldsymbol{B}_m$ during the process, Once the ill-conditionness is detected, the Arnoldi process is halted and we do an update of the last vector $\boldsymbol{b}_{m+1}$ to ensure it is orthogonal to $\boldsymbol{B}_m$, as stated in Section 4. Then, we use the FOM approximation (4.2) or the sFom approximation (5.1) and restart under the guidence of Theorem 4.2. See Algorithm 6 for the Arnoldi process.

The corresponding functions about the above discussions are:

- "tarnoldi_last_orth_adaptive"
- "tarnoldi_last_sorth_adaptive"
- "funm_quad_ada_fom_last_orth_tarnoldi"
- "funm_quad_ada_sfom_last_sorth_tarnoldi"

Algorithm 6 Adaptive truncated Arnoldi process using a random sketching monitor.

Input: Matrix $\boldsymbol{A}$, vector $\boldsymbol{b}$, truancation parameter t, tolerance of condition number τ, max dimenstion of the Krylov subspace $m_{\max}$.

Output: Subspace dimension m, basis $\boldsymbol{B}_m \in \mathbb{C}^{N\times m}$ for the Krylov subspace $\mathcal{K}_m(\boldsymbol{A};\boldsymbol{b})$.

Set $s = s_0 = 30$.
Draw a sketching matrix $\boldsymbol{S} \in \mathbb{C}^{s\times N}$.
$\boldsymbol{b}_1 = \boldsymbol{b}/\|\boldsymbol{b}\|$.
$\boldsymbol{B}_1 = [\boldsymbol{b}_1]$, $\boldsymbol{P}_1 = [\boldsymbol{S}\boldsymbol{b}_1]$.
for $j = 1, \ldots, m_{\max}$ **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
  $\boldsymbol{b}_{j+1} = \boldsymbol{w}_j / h_{j+1,j}$.
  $\boldsymbol{B}_{j+1} = [\boldsymbol{B}_j, \boldsymbol{b}_{j+1}]$, $\boldsymbol{P}_{j+1} = [\boldsymbol{P}_j, \boldsymbol{S}\boldsymbol{b}_{j+1}]$.
  if $\mathrm{cond}(\boldsymbol{P}_{j+1}) > \tau$ **then**
    Break
  end if
  if $s < 2j$ **then**
    Draw a sketching matrix $\boldsymbol{S}_{\mathrm{incr}} \in \mathbb{C}^{s_0\times N}$.
    Update $\boldsymbol{S} \leftarrow \begin{bmatrix} \boldsymbol{S} \\ \boldsymbol{S}_{\mathrm{incr}} \end{bmatrix}$ and $\boldsymbol{P}_{j+1} \leftarrow \begin{bmatrix} \boldsymbol{P}_{j+1} \\ \boldsymbol{S}_{\mathrm{incr}}\boldsymbol{B}_{j+1} \end{bmatrix}$
    $s \leftarrow s + s_0$.
  end if
end for
$m = j$.
Update b_{m+1} to be orthogonal to $\boldsymbol{B}_m$.
$\boldsymbol{B}_m = [\boldsymbol{b}_1, \ldots, \boldsymbol{v}_m]$.

7 Numerical Experiments

Through all experiments, we set the sketching dimension to be roughly $2m$ where m is the dimension of the Krylov subspace. For the truncated Arnoldi process, we set the truncation parameter t to be 0, 1 and 2. We use the sparse sign 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 for fixed-m methods and 200 for adaptive methods and the maximum number of quadrature point updates is set to be approximately $8192\sqrt{2}$. 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 Arnoldi 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 Arnoldi process to generate the basis $\boldsymbol{B}_m$ and compute the sketched FOM approximant.
- aFOM-t: using the adaptive truncated Arnoldi process to generate the basis $\boldsymbol{B}_m$ and compute the FOM approximant.

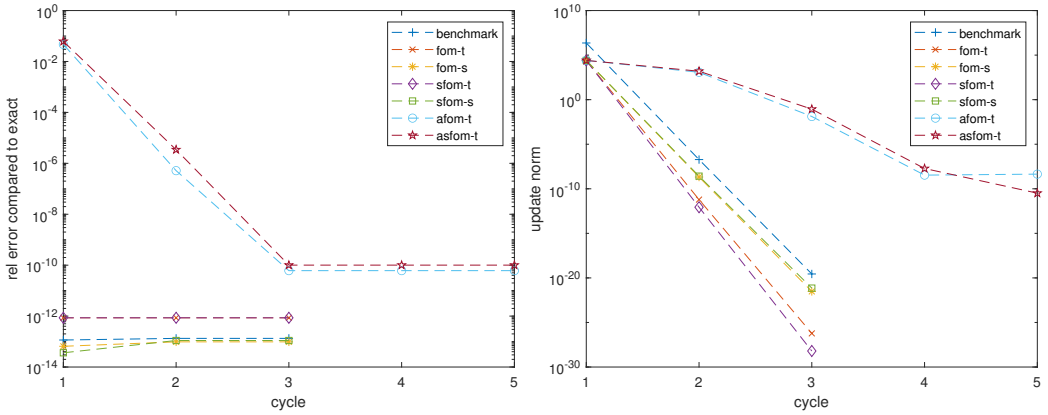
7.1 Network

We consider the computation of $f(\boldsymbol{A})\boldsymbol{b} = \mathrm{e}^{-\boldsymbol{A}}\boldsymbol{b}$, where $\boldsymbol{A}$ is given by the nonsymmetric binary adjacency matrix "wiki-Vote" of size $N = 8297$. We set $m = 100$.

Table 7.1 and Figure 7.1 show the results for $t = 2$.

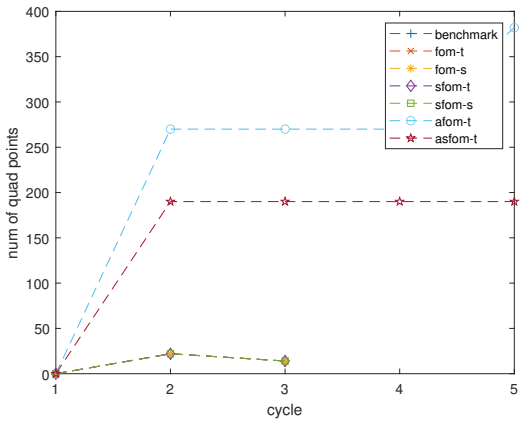
Method	total iter	rel error	time
benchmark	3	1.3342e-13	2.4085e-01
FOM-t	3	8.5748e-13	1.4808e-01
sFOM-t	3	8.5745e-13	1.3997e-01
FOM-s	3	9.7423e-14	2.5317e-01
sFOM-s	3	1.0998e-13	1.9325e-01
aFOM-t	5	6.1037e-11	1.2394e-01
asFOM-t	5	1.0050e-10	1.0430e-01

Table 7.1: Results for network, $t = 2$, $m = 100$

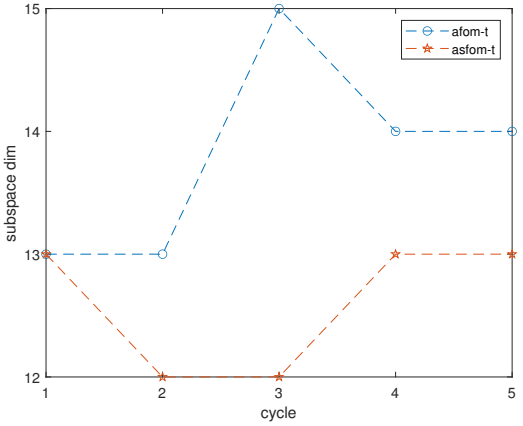


(a) Relative error.

(b) Norm of update.



(c) Number of quadrature points.



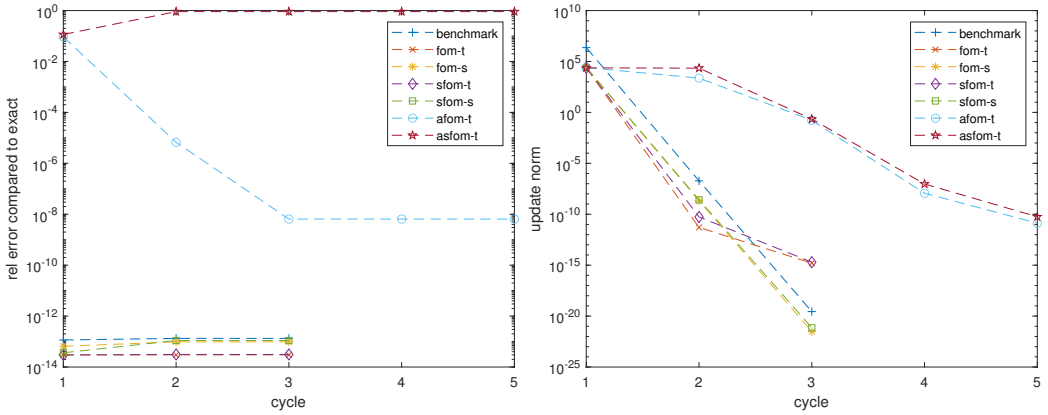
(d) Subspace dimensions.

Figure 7.1: Results for network, $t = 2$, $m = 100$.

Table 7.2 and Figure 7.2 show the results for $t = 1$.

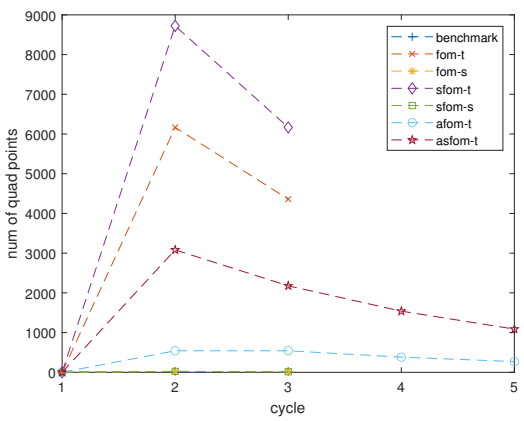
Method	total iter	rel error	time
benchmark	3	1.3342e-13	2.3348e-01
FOM-t	3	2.9985e-14	8.4925e-01
sFOM-t	3	3.1170e-14	1.1094e+00
FOM-s	3	9.7423e-14	2.4156e-01
sFOM-s	3	1.0998e-13	1.7611e-01
aFOM-t	5	6.4712e-09	1.1014e-01
asFOM-t	5	9.0960e-01	1.4135e-01

Table 7.2: Results for network, $t = 1$, $m = 100$.

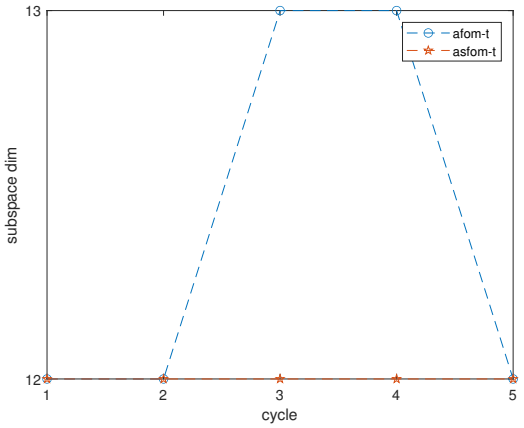


(a) Relative error.

(b) Norm of update.



(c) Number of quadrature points.



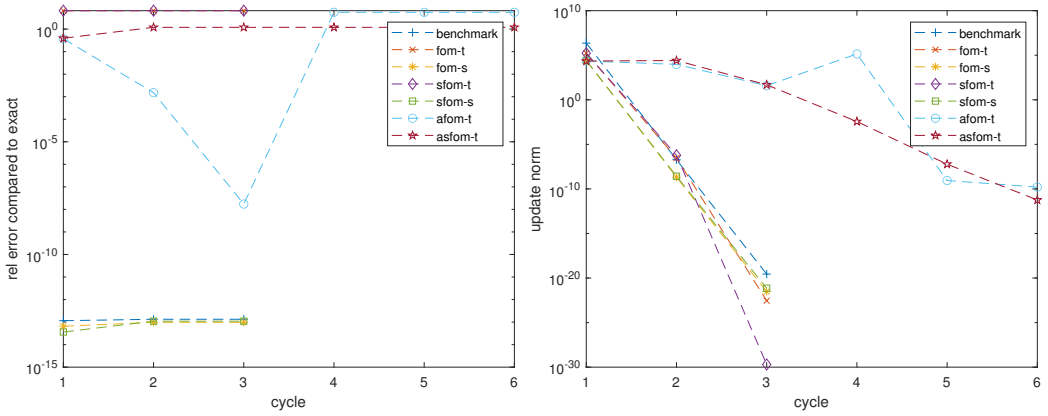
(d) Subspace dimensions.

Figure 7.2: Results for network, $t = 1$, $m = 100$,

Table 7.3 and Figure 7.3 show the results for $t = 0$.

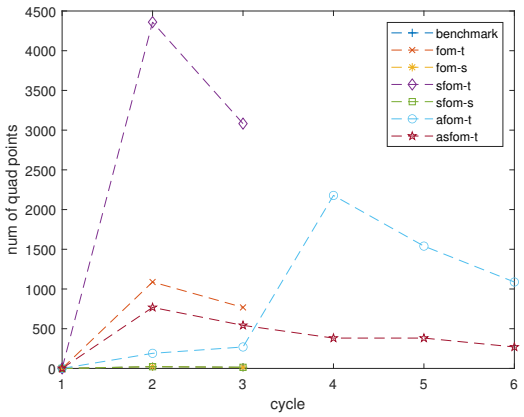
Method	total iter	rel error	time
benchmark	3	1.3342e-13	2.3338e-01
FOM-t	3	6.5645e+00	2.9884e-01
sFOM-t	3	6.5595e+00	6.4345e-01
FOM-s	3	9.7423e-14	2.3382e-01
sFOM-s	3	1.0998e-13	1.7276e-01
aFOM-t	6	5.5609e+00	1.3946e-01
asFOM-t	6	1.1979e+00	1.2174e-01

Table 7.3: Results for network, $t = 0$, $m = 100$,

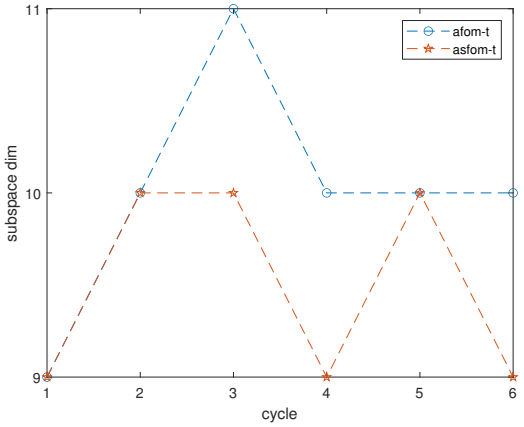


(a) Relative error.

(b) Norm of update.



(c) Number of quadrature points.



(d) Subspace dimensions.

Figure 7.3: Results for network, $t = 0$, $m = 100$,

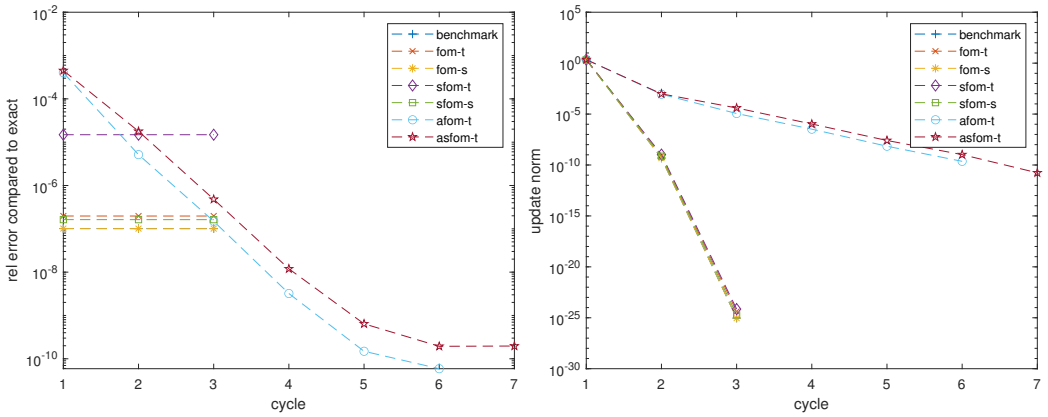
7.2 Lattice QCD

In this example, $m = 300$.

Table 7.4 and Figure 7.4 show the results for $t = 2$.

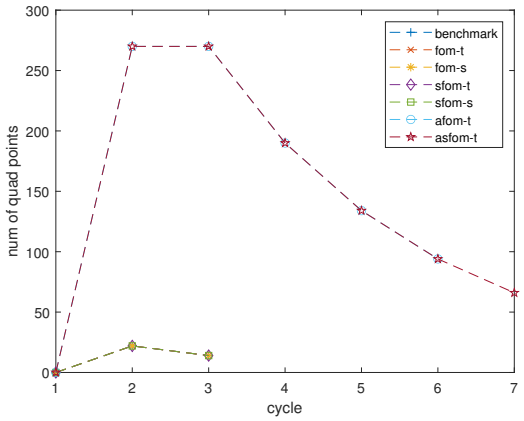
Method	total iter	rel error	time
benchmark	3	1.0114e-07	4.5064e+01
FOM-t	3	1.9739e-07	1.6007e+01
sFOM-t	3	1.4867e-05	7.9457e+00
FOM-s	3	1.0114e-07	4.7935e+01
sFOM-s	3	1.6288e-07	3.9101e+01
aFOM-t	6	5.9042e-11	1.4811e+01
asFOM-t	7	1.9699e-10	1.2892e+01

Table 7.4: Results for QCD, $t = 2$, $m = 300$,

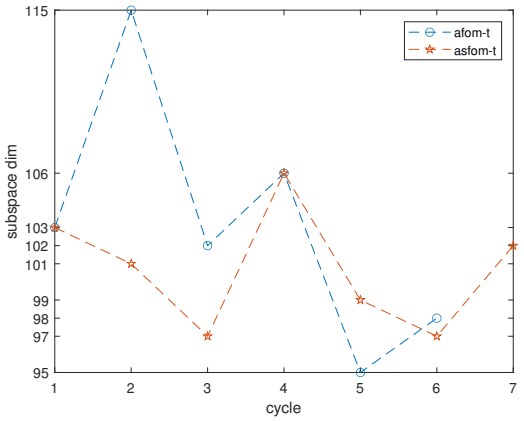


(a) Relative error.

(b) Norm of update.



(c) Number of quadrature points.



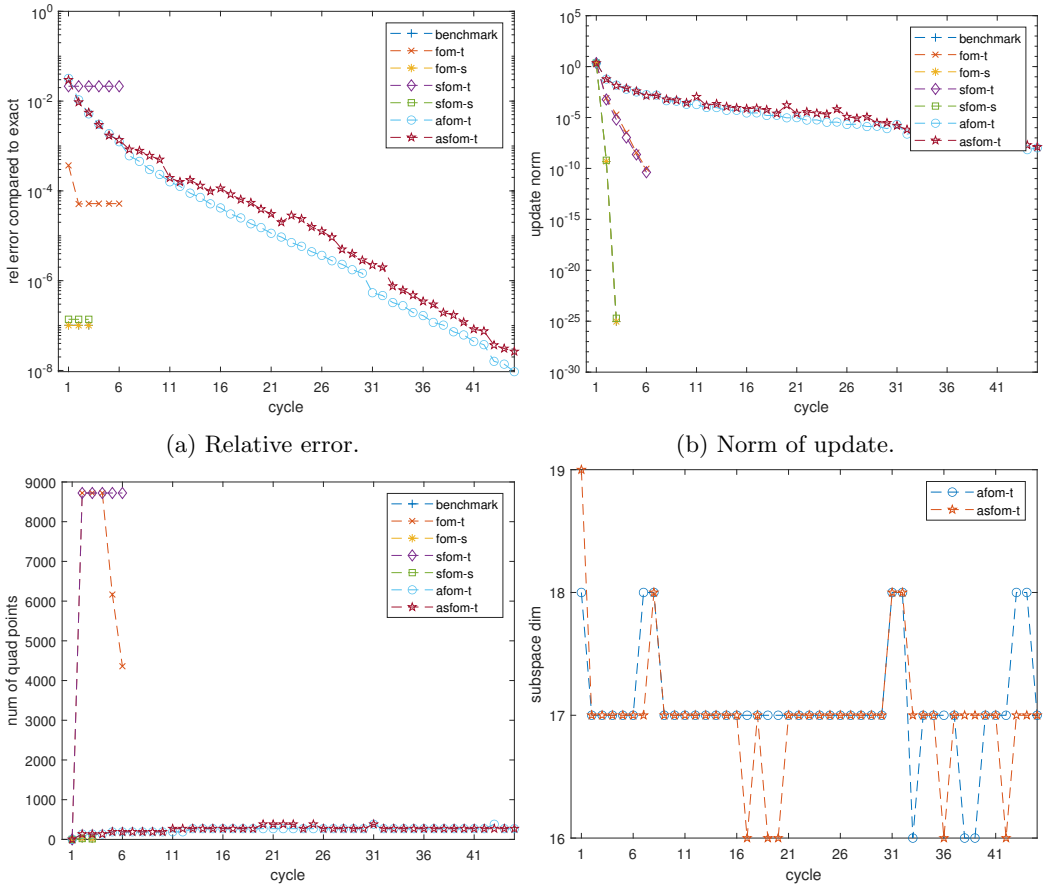
(d) Subspace dimensions.

Figure 7.4: Results for QCD, $t = 2$, $m = 300$,

Table 7.5 and Figure 7.5 show the results for $t = 1$.

Method	total iter	rel error	time
benchmark	3	1.0114e-07	4.5011e+01
FOM-t	6	5.2081e-05	1.1404e+02
sFOM-t	6	2.1433e-02	1.1898e+02
FOM-s	3	1.0114e-07	5.0366e+01
sFOM-s	3	1.3782e-07	3.9815e+01
aFOM-t	45	9.4104e-09	1.7115e+01
asFOM-t	45	2.6401e-08	1.5344e+01

Table 7.5: Results for QCD, $t = 1$, $m = 300$,



(a) Relative error.

(b) Norm of update.

(c) Number of quadrature points.

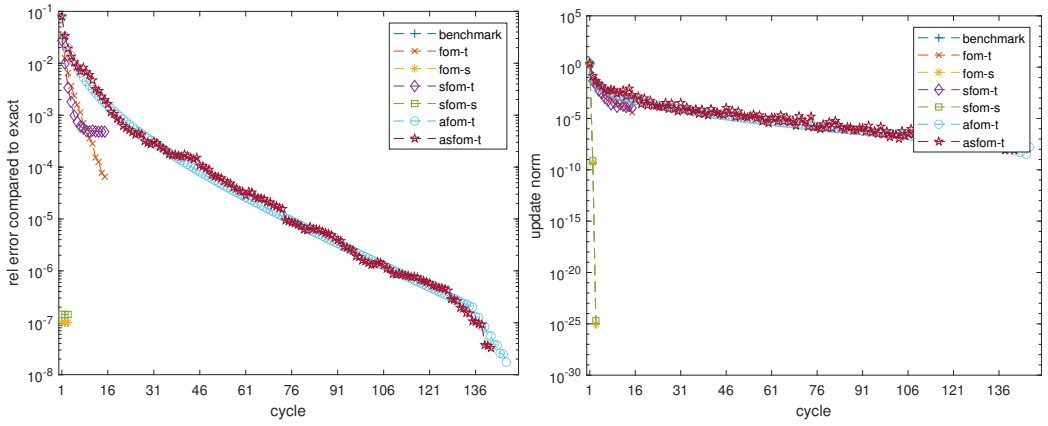
(d) Subspace dimensions.

Figure 7.5: Results for QCD, $t = 1$, $m = 300$,

Table 7.6 and Figure 7.6 show the results for $t = 0$.

Method	total iter	rel error	time
benchmark	3	1.0114e-07	4.5867e+01
FOM-t	15	6.5083e-05	1.6547e+02
sFOM-t	15	4.7910e-04	2.9379e+02
FOM-s	3	1.0114e-07	4.8148e+01
sFOM-s	3	1.4424e-07	4.1109e+01
aFOM-t	146	1.7489e-08	2.7303e+01
asFOM-t	141	3.2790e-08	2.4499e+01

Table 7.6: Results for QCD, $t = 0$, $m = 300$,



(a) Relative error.

(b) Norm of update.

(c) Number of quadrature points.

(d) Subspace dimensions.

Figure 7.6: Results for QCD, $t = 0$, $m = 300$,

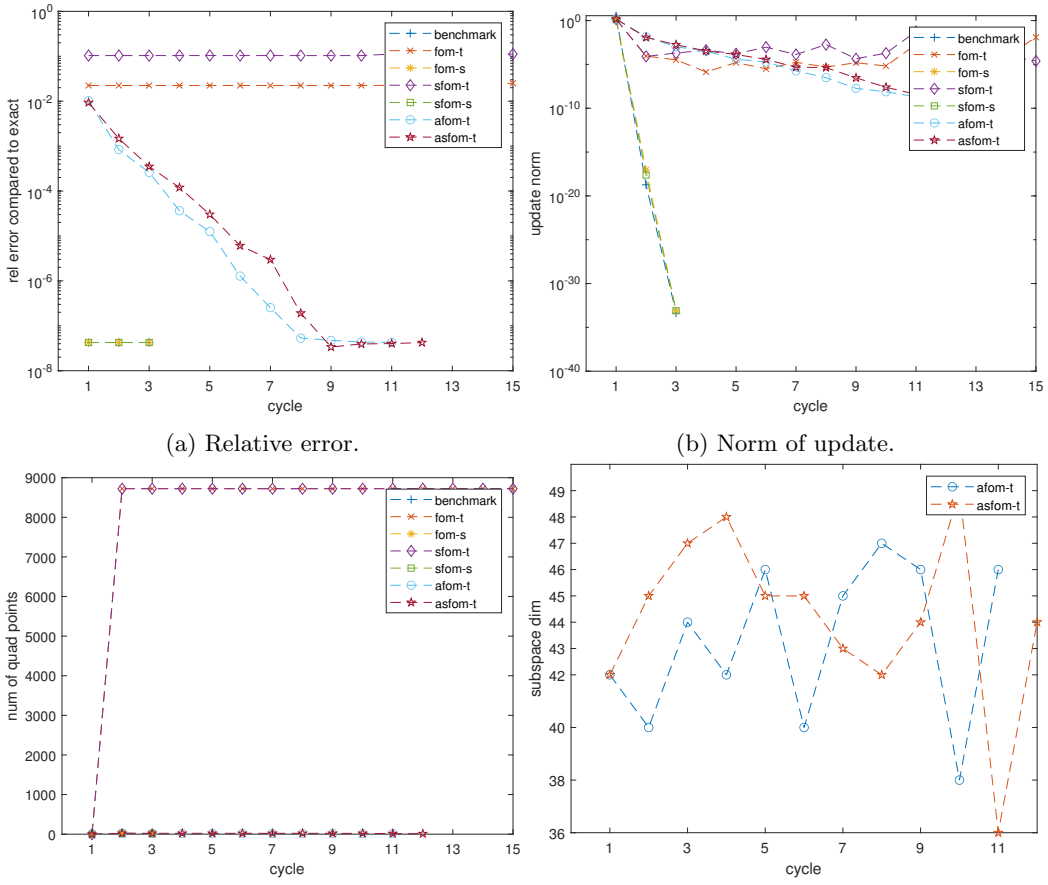
7.3 Fractional Laplacian

In this example, $m = 300$.

Table 7.7 and Figure 7.7 show the results for $t = 2$.

Method	total iter	rel error	time
benchmark	3	4.2498e-08	1.6861e+00
FOM-t	15	2.5144e-02	6.4962e+01
sFOM-t	15	1.1191e-01	6.4820e+01
FOM-s	3	4.2502e-08	1.1515e+00
sFOM-s	3	4.2503e-08	9.6743e-01
aFOM-t	11	4.2729e-08	6.4791e-01
asFOM-t	12	4.2283e-08	6.6324e-01

Table 7.7: Results for fractional Laplacian, $t = 2$, $m = 300$,



(a) Relative error.

(b) Norm of update.

(c) Number of quadrature points.

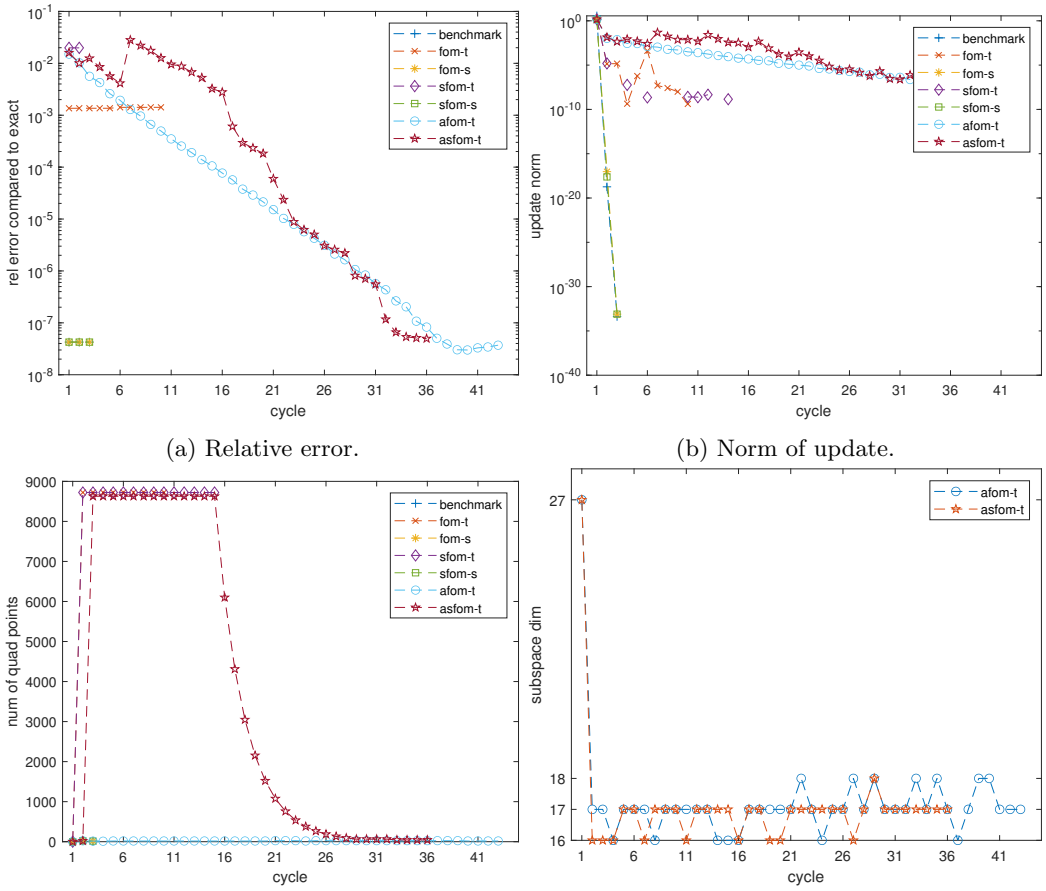
(d) Subspace dimensions.

Figure 7.7: Results for fractional Laplacian, $t = 2$, $m = 300$,

Table 7.8 and Figure 7.8 show the results for $t = 1$.

Method	total iter	rel error	time
benchmark	3	4.2498e-08	1.6136e+00
FOM-t	10	1.4190e-03	4.3678e+01
sFOM-t	15	NaN	6.8601e+01
FOM-s	3	4.2502e-08	1.1498e+00
sFOM-s	3	4.2503e-08	9.6242e-01
aFOM-t	43	3.7081e-08	1.2422e+00
asFOM-t	36	4.9721e-08	2.4454e+00

Table 7.8: Results for fractional Laplacian, $t = 1$, $m = 300$,



(a) Relative error.

(b) Norm of update.

(c) Number of quadrature points.

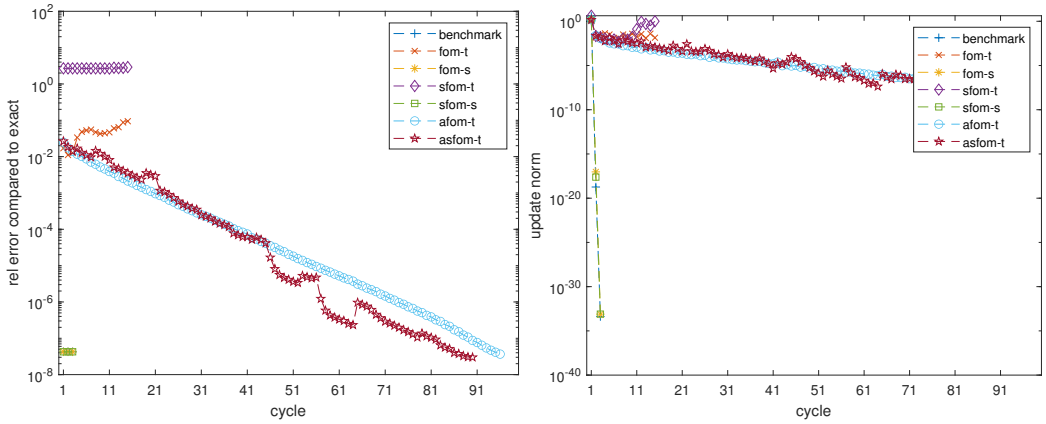
(d) Subspace dimensions.

Figure 7.8: Results for fractional Laplacian, $t = 1$, $m = 300$,

Table 7.9 and Figure 7.9 show the results for $t = 0$.

Method	total iter	rel error	time
benchmark	3	4.2498e-08	1.6243e+00
FOM-t	15	9.4321e-02	7.1189e+01
sFOM-t	15	2.9110e+00	7.1780e+01
FOM-s	3	4.2502e-08	1.1622e+00
sFOM-s	3	4.2503e-08	9.9637e-01
aFOM-t	96	3.6685e-08	1.4183e+00
asFOM-t	90	2.9992e-08	1.4526e+01

Table 7.9: Results for fractional Laplacian, $t = 0$, $m = 300$,



(a) Relative error.

(b) Norm of update.

(c) Number of quadrature points.

(d) Subspace dimensions.

Figure 7.9: Results for fractional Laplacian, $t = 0$, $m = 300$,

7.4 Implicit Deflation

7.4.1 Demo

We also trid implicit deflation to the adaptive methods, using the same exmaple in "demo_log.m". It is remarkable that although the original matrix $\boldsymbol{A}$ is symmetric, we do not exploit the symmetry in all methods. That is, the corresponding methods parameter ".hermitian" is always set to be 0.

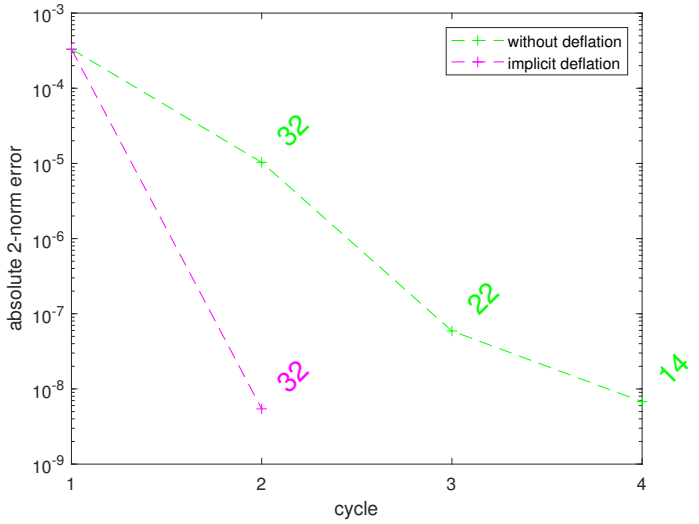
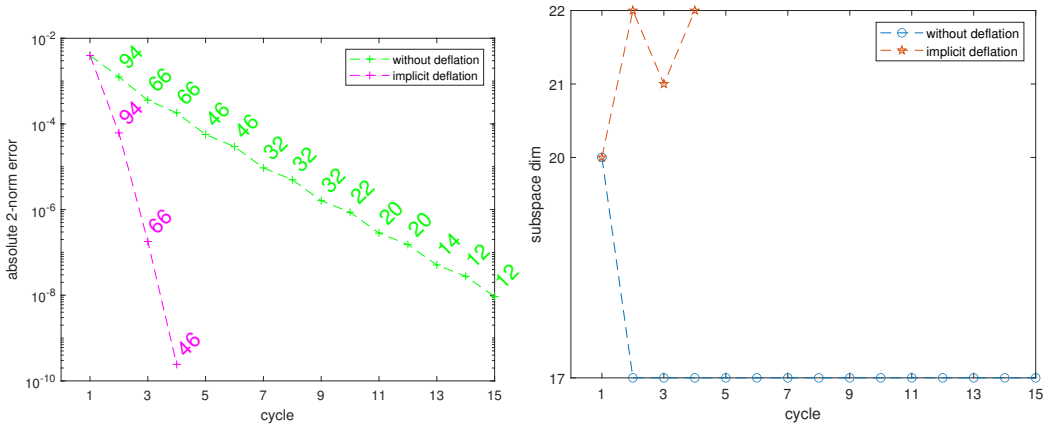
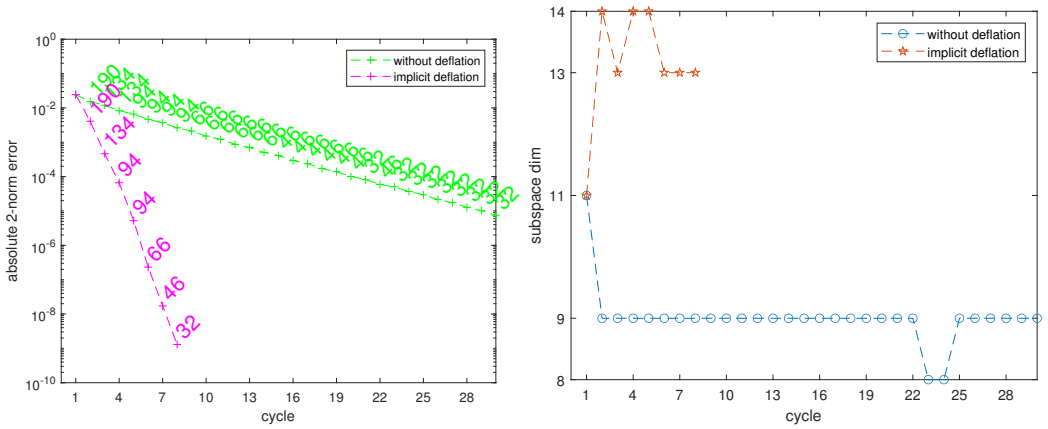


Figure 7.10: Implicit deflation for benchmark, $m = 30$.



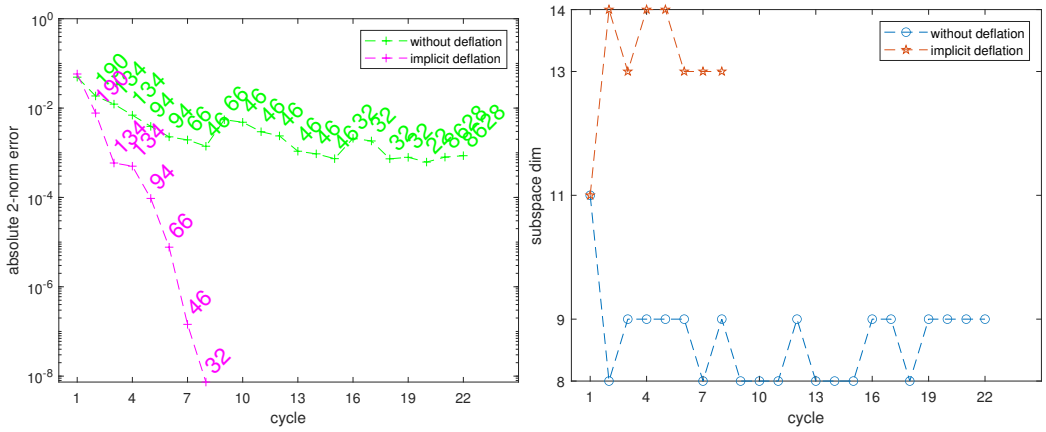
(a) Relative error. (b) Subspace dimensions.

Figure 7.11: Results for aFOM-t, $t = 1$, $m = 30$,



(a) Relative error. (b) Subspace dimensions.

Figure 7.12: Results for aFOM-t, $t = 0$, $m = 30$,



(a) Relative error. (b) Subspace dimensions.

Figure 7.13: Results for asFOM-t, $t = 0$, $m = 30$,

The behavior of asFOM-t with $t = 1$ without implicit deflation is strange. It keeps print the message that, for example, "6166 quadrature points were not enough. Trying 8722. Norm: 10626.1202", meaning that the quadrature approximation is very bad.

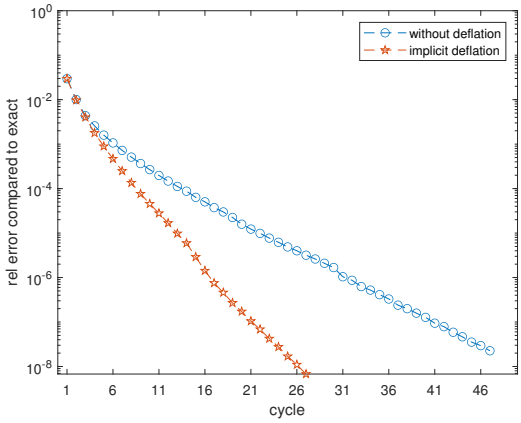
7.4.2 QCD

We also apply implicit deflation to the QCD example in Subsection 7.2. We set $m = 30$.

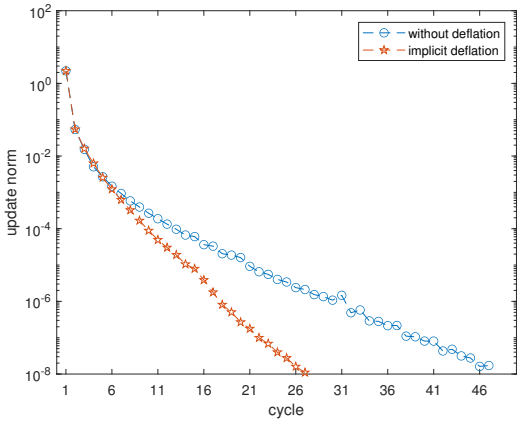
Table 7.10, Figure 7.14 and Figure 7.15 show the results for $t = 1$.

Method	total iter	rel error	time
aFOM-t (without deflation)	47	2.2548e-08	1.6948e+01
aFOM-t (implicit deflation)	27	6.7614e-09	1.6027e+01
asFOM-t (without deflation)	51	4.6105e-09	1.6493e+01
asFOM-t (implicit deflation)	32	6.2396e-09	1.6856e+01

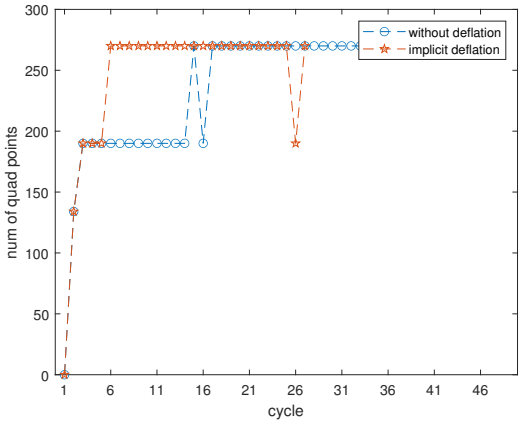
Table 7.10: Results for QCD, $t = 1$, $m = 30$,



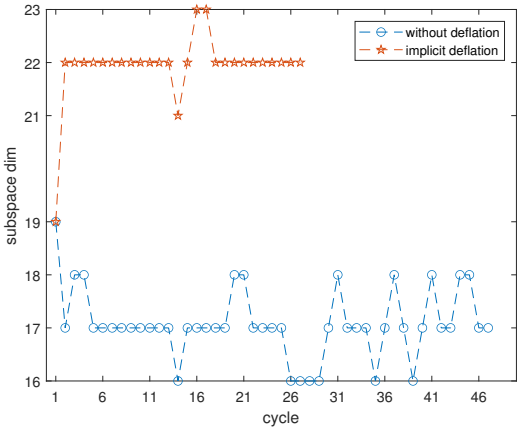
(a) Relative error.



(b) Norm of update.

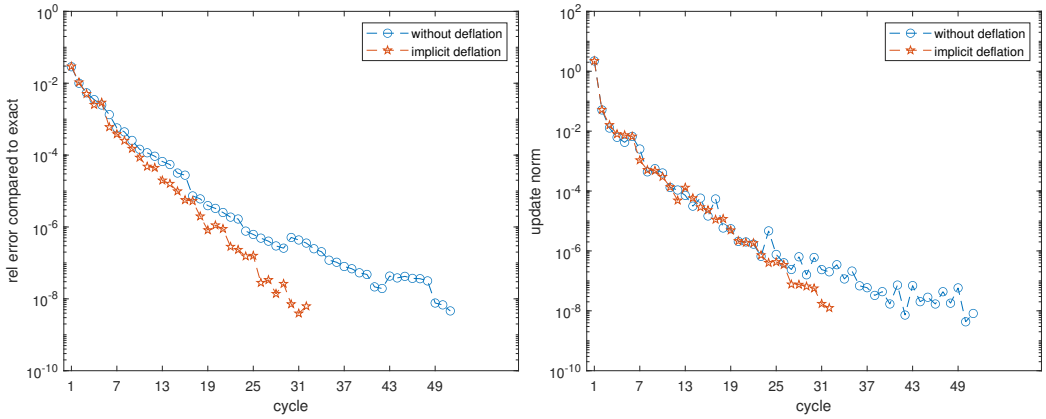


(c) Number of quadrature points.



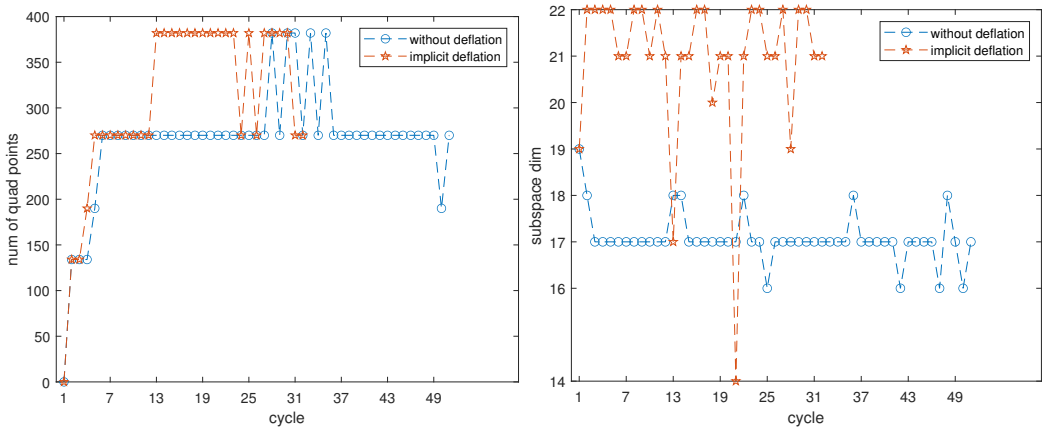
(d) Subspace dimensions.

Figure 7.14: Results for QCD using aFOM-t, $t = 1$, $m = 30$,



(a) Relative error.

(b) Norm of update.



(c) Number of quadrature points.

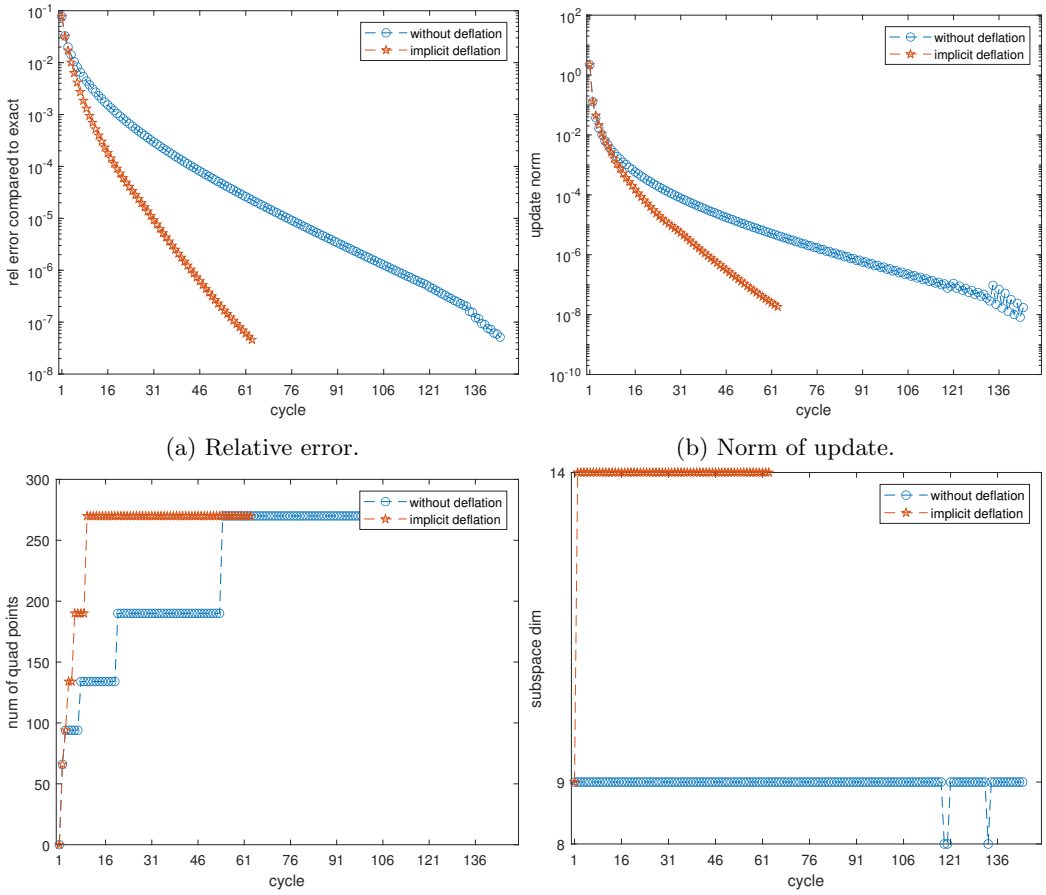
(d) Subspace dimensions.

Figure 7.15: Results for QCD using asFOM-t, $t = 1$, $m = 30$,

Table 7.10, Figure 7.14 and Figure 7.15 show the results for $t = 0$.

Method	total iter	rel error	time
aFOM-t (without deflation)	144	5.1368e-08	2.4616e+01
aFOM-t (implicit deflation)	63	4.5733e-08	2.3612e+01
asFOM-t (without deflation)	138	1.0481e-07	2.2175e+01
asFOM-t (implicit deflation)	69	7.5143e-09	2.4520e+01

Table 7.11: Results for QCD, $t = 0$, $m = 30$,



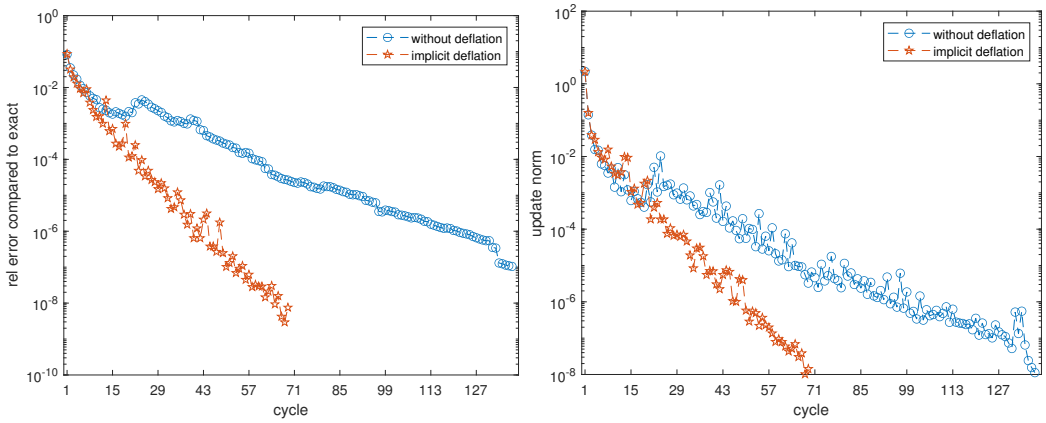
(a) Relative error.

(b) Norm of update.

(c) Number of quadrature points.

(d) Subspace dimensions.

Figure 7.16: Results for QCD using aFOM-t, $t = 0$, $m = 30$,



(a) Relative error.

(b) Norm of update.

(c) Number of quadrature points.

(d) Subspace dimensions.

Figure 7.17: Results for QCD using asFOM-t, $t = 0$, $m = 30$,

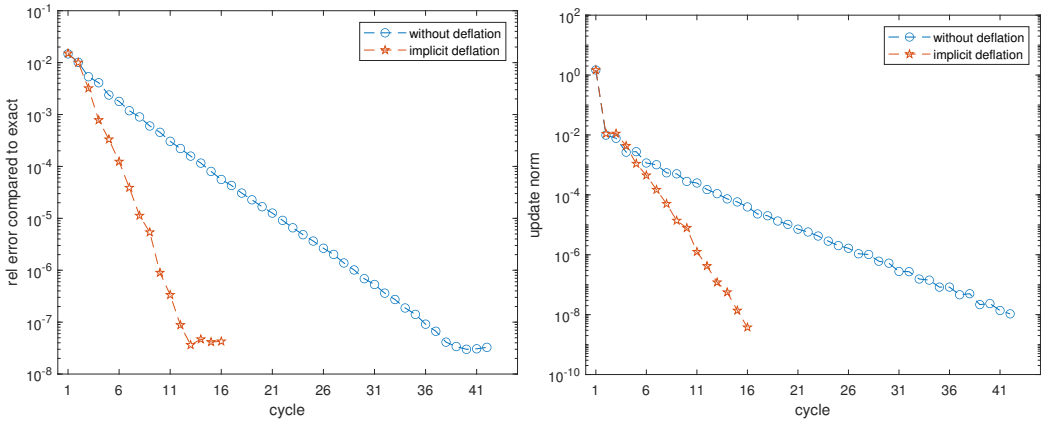
7.4.3 Fractional Laplacian

We also apply implicit deflation to the fractional Laplacian example in Subsection 7.3. We set $m = 100$.

Table 7.12, Figure 7.18 and Figure 7.19 show the results for $t = 1$.

Method	total iter	rel error	time
aFOM-t (without deflation)	42	3.2672e-08	1.1830e+00
aFOM-t (implicit deflation)	16	4.2570e-08	8.2713e-01
asFOM-t (without deflation)	37	5.1620e-08	1.0821e+00
asFOM-t (implicit deflation)	22	1.2325e-07	5.7848e+00

Table 7.12: Results for fractional Laplacian, $t = 1$, $m = 100$,



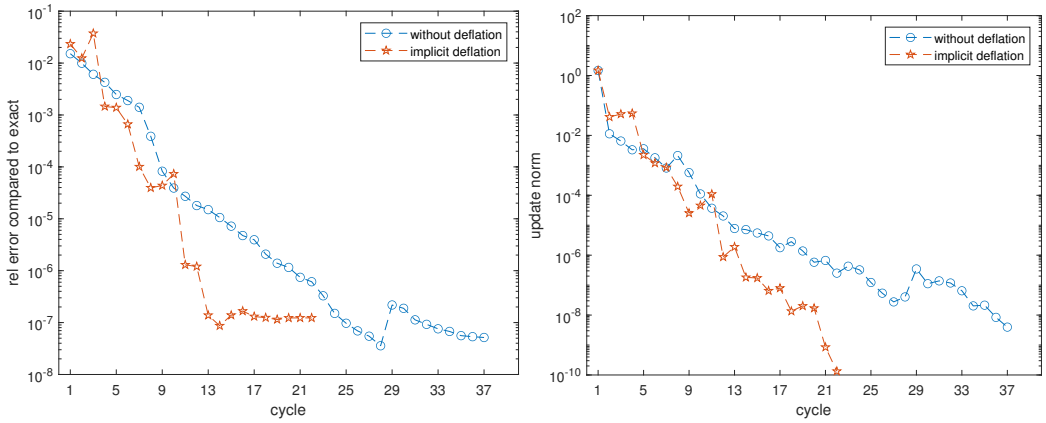
(a) Relative error.

(b) Norm of update.

(c) Number of quadrature points.

(d) Subspace dimensions.

Figure 7.18: Results for fractional Laplacian using aFOM-t, $t = 1$, $m = 100$,



(a) Relative error.

(b) Norm of update.

(c) Number of quadrature points.

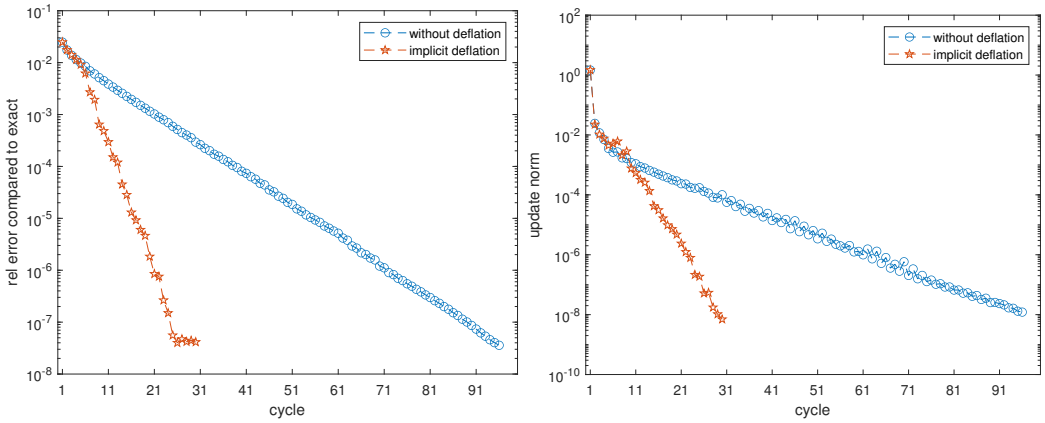
(d) Subspace dimensions.

Figure 7.19: Results for fractional Laplacian using asFOM-t, $t = 1$, $m = 100$,

Table 7.13, Figure 7.20 and Figure 7.21 show the results for $t = 0$.

Method	total iter	rel error	time
aFOM-t (without deflation)	96	3.5813e-08	1.3987e+00
aFOM-t (implicit deflation)	30	4.1388e-08	1.0602e+00
asFOM-t (without deflation)	95	3.2391e-08	1.3605e+01
asFOM-t (implicit deflation)	37	4.3496e-08	4.8878e+00

Table 7.13: Results for fractional Laplacian, $t = 0$, $m = 100$,



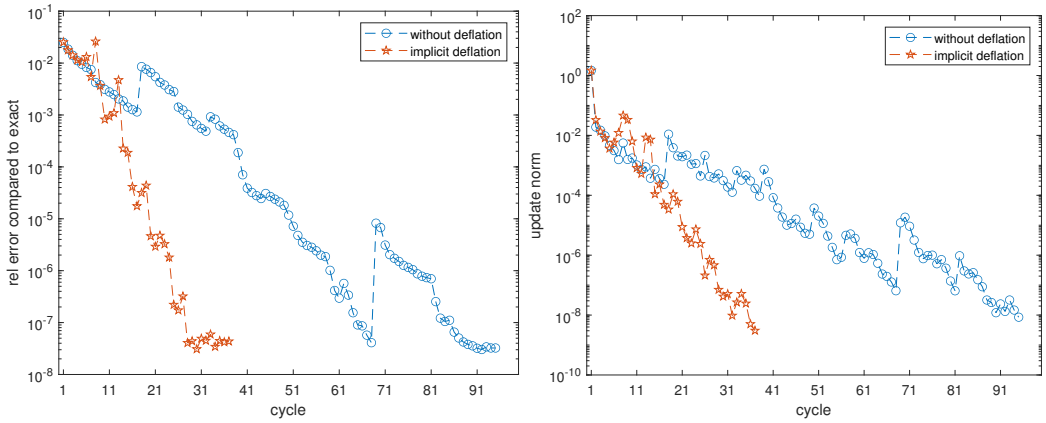
(a) Relative error.

(b) Norm of update.

(c) Number of quadrature points.

(d) Subspace dimensions.

Figure 7.20: Results for fractional Laplacian using aFOM-t, $t = 0$, $m = 100$,



(a) Relative error.

(b) Norm of update.

(c) Number of quadrature points.

(d) Subspace dimensions.

Figure 7.21: Results for fractional Laplacian using asFOM-t, $t = 0$, $m = 100$,

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.