An Augmented Lanczos Algorithm for the Efficient Computation of a Dot-Product of a Function of a Large Sparse Symmetric Matrix

Roger B. Sidje[1], Kevin Burrage[1], and B. Philippe[2]

[1] Department of Mathematics
University of Queensland
Brisbane QLD 4072, Australia
{rbs,kb}@maths.uq.edu.au
[2] INRIA/IRISA
Campus de Beaulieu
35042 Rennes Cedex, France
philippe@irisa.fr

Abstract. The Lanczos algorithm is appreciated in many situations due to its speed and economy of storage. However, the advantage that the Lanczos basis vectors need not be kept is lost when the algorithm is used to compute the action of a matrix function on a vector. Either the basis vectors need to be kept, or the Lanczos process needs to be applied twice. In this study we describe an augmented Lanczos algorithm to compute a dot product relative to a function of a large sparse symmetric matrix, without keeping the basis vectors.

1 Introduction

To compute the action of a matrix-function on a vector, $f(A)v$, where A is a large sparse symmetric matrix of order n, and f is a function for which A is admissible (i.e., $f(A)$ is defined), the general principle of the Lanczos method relies on projecting the operator A onto the Krylov subspace $\mathcal{K}_m(A,v) = \text{Span}\{v, Av, \ldots, A^{m-1}v\}$ and approximating the action of the function in the subspace. Here, m is the prescribed dimension of the Krylov subspace and in practice $m \ll n$. The classical Lanczos Algorithm 1 will start with the vector v and perform a three-term recurrence scheme to construct a symmetric tridiagonal matrix T_m and an orthonormal basis V_m of $\mathcal{K}_m(A,v)$:

$$T_m = \begin{bmatrix} \alpha_1 & \beta_2 & & \\ \beta_2 & \alpha_2 & \ddots & \\ & \ddots & \ddots & \beta_m \\ & & \beta_m & \alpha_m \end{bmatrix}, \qquad V_m = [v_1, v_2, \ldots, v_m]$$

and for any $j = 1, \ldots, m$ we have the fundamental relations

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2659, pp. 693–704, 2003.

$$\begin{aligned} V_j^T V_j &= I \\ AV_j &= V_j T_j + \beta_{j+1} v_{j+1} e_j^T \\ V_j^T A V_j &= T_j \end{aligned} \tag{1}$$

where $V_j = [v_1, \ldots, v_j]$, T_j is the jth leading submatrix of T_m, e_j is the jth column of the identity matrix of appropriate size. The pseudo-code for the Lanczos process is indicated below.

ALGORITHM 1: Lanczos(m, A, v)
$\beta = \|v\|_2 \,; \quad v_1 := v/\beta \,; \quad \beta_1 := 0 \,;$
for $j := 1 : m$ **do**
  $p := Av_j - \beta_j v_{j-1} \,;$
  $\alpha_j := v_j^T p \,; \quad p := p - \alpha_j v_j \,;$
  $\beta_{j+1} := \|p\|_2 \,; \quad v_{j+1} := p/\beta_{j+1} \,;$
endfor

From this, $v = \beta V_m e_1$ where $\beta = \|v\|_2$. To compute $w = f(A)v$, the underlying principle is to use the approximation $w \approx \widetilde{w}$ where

$$\widetilde{w} = \beta V_m f(T_m) e_1. \tag{2}$$

Thus, the distinctive feature is that the original large problem is converted to the smaller problem (2) which is more desirable. Moreover the special structure of T_m (symmetric tridiagonal) means that, when possible, the smaller matrix-function action $f(T_m)e_1$ can be handled in an efficient, tailor-made manner. Theory and practice over the years have confirmed that this simplified process is amazingly effective for a wide class of problems. In general however, the basis $V_m = [v_1, \ldots, v_m]$ has to be kept for the final evaluation of $\widetilde{w} = \beta V_m f(T_m) e_1$. The basis V_m is a dense matrix of size $n \times m$, and when n is very large, such a storage is seen as a drawback in certain configurations. In specific contexts such as linear systems (conjugate gradient-type methods) or eigenvalue problems (where eigenvectors are not needed), it is possible to avoid storing the basis vectors and this is considered as a significant strength of the Lanczos scheme. In principle however, it is also possible to avoid this storage by applying the Lanczos process twice. But the perception that Lanczos storage savings are lost when evaluating $f(A)v$ has prompted some authors to offer alternative methods outside the Krylov-based paradigm. For instance, Bergamaschi and Vianello [2] used Chebyshev series expansions to compute the action of the exponential operator in the case of large sparse symmetric matrices arising after space discretization of 2D and 3D parabolic PDEs by finite differences and finite elements. The configuration used for their experiments was a 600Mhz Alpha station with 128MB of RAM. Although the Chebyshev series method requires an *a priori* estimation of the leftmost and rightmost eigenvalues of A, and is known to converge slower than the Krylov approach, they showed that when memory is at premium, some advantages of the Krylov method may need to be traded.

In this study, we present an augmented Lanczos algorithm which alleviates the drawback, allowing the Krylov framework to be fully retained when the issue

at hand is really to compute $u^T f(A)v$ for given vectors u and v, i.e., when we are interested in a weighted sum of the action of matrix-function on a vector. As a particular example, taking $u = e_i$ and $v = e_j$ allows retrieving the ij-th entry of $f(A)$. Assuming the matrix is positive definite, this was used to bound entries of a matrix function [1,5,6]. In this study however, we do not assume that the matrix is positive definite. Another example worth mentioning is the evaluation of the norm $\|f(A)v\|_2$ which can be handled as a special case. Thus, compared to a method such as the Chebyshev method, the benefits are two-fold:

- Economy of storage: the Lanczos basis vectors need not be saved. This alleviates the primary drawback that motivated the Chebyshev method, and
- Potential re-usability: the computed data can be re-used to get $\{u_p^T f(A)v\}$ for a family of vectors $\{u_p\}$. Note that because the matrix is symmetric, computing $\{u_p^T f(A)v\}$ is algorithmically equivalent to computing $\{u^T f(A)v_q\}$ (but not equivalent to computing $\{u_p^T f(A)v_q\}$ which would need several sweeps). The computed data can also be re-used to get $u^T f_p(A)v$ for a family of functions $\{f_p\}$. This can be useful when the argument matrix depends on another variable, e.g., the matrix may belong to the parameterized family $A(t) = tA$ where t is the 'time' variable. As a concrete example, evaluating $\int_0^T u^T f(tA)v$ may involve discretizing the integration domain and computing the matrix function at several knots. Sometimes also, the matrix may vary in terms of rank-updates – such problems can be seen in Bernstein and Van Loan [3] though in the unsymmetric context. A careful implementation of our scheme can offer a suitable re-usable framework where appropriate.

2 The Augmented Lanczos Algorithm

The principle of augmenting a Krylov subspace is well-known in the literature and has been used in several other contexts. Differences between approaches usually lie in the strategy by which the augmentation is made. Probably, the most popular Krylov algorithm obtained via augmentation is the GMRES algorithm of Saad and Schultz [11] which resulted from taking full advantage of the information computed by the Arnoldi procedure (or FOM – Full Orthogonalization Method). Our proposed augmented scheme falls in a class of similar approaches. To get $u^T f(A)v$, the idea is to augment the Lanczos basis V_m as follows. Define the orthogonal projector onto V_m as $P_m = V_m V_m^T$, and let

$$\hat{V}_{m+1} = [V_m \;\; | \;\; \hat{v}_{m+1}]$$

where

$$\hat{v}_{m+1} = \frac{(I - P_m)u}{\|(I - P_m)u\|_2},$$

and take an approximation in this augmented subspace as $u^T w \approx u^T \hat{w}$ where

$$\hat{w} = \beta \hat{V}_{m+1} f(\hat{V}_{m+1}^T A \hat{V}_{m+1}) e_1.$$

Now, we have the symmetric matrix

$$\hat{V}_{m+1}^T A \hat{V}_{m+1} = \begin{bmatrix} V_m^T \\ \hat{v}_{m+1}^T \end{bmatrix} A [V_m \;\mid\; \hat{v}_{m+1}] = \begin{bmatrix} T_m & V_m^T A \hat{v}_{m+1} \\ \hat{v}_{m+1}^T A V_m & \hat{v}_{m+1}^T A \hat{v}_{m+1} \end{bmatrix}$$

and expanding further, we get

$$\hat{v}_{m+1}^T A V_m = \hat{v}_{m+1}^T (V_m T_m + \beta_{m+1} v_{m+1} e_m^T) = \beta_{m+1} \hat{v}_{m+1}^T v_{m+1} e_m^T$$

and letting

$$\hat{\beta}_{m+1} \equiv \beta_{m+1} \hat{v}_{m+1}^T v_{m+1} \tag{3}$$

and

$$\hat{\alpha}_{m+1} \equiv \hat{v}_{m+1}^T A \hat{v}_{m+1} \tag{4}$$

we see that

$$\hat{V}_{m+1}^T A \hat{V}_{m+1} \equiv \hat{T}_{m+1} = \left[\begin{array}{ccc|c} & & & 0 \\ & T_m & & \vdots \\ & & & 0 \\ & & & \hat{\beta}_{m+1} \\ \hline 0 & \cdots & 0 & \hat{\beta}_{m+1} \;\; \hat{\alpha}_{m+1} \end{array}\right].$$

Hence $\hat{T}_{m+1}$ remains *tridiagonal*. Furthermore, just as with the standard Lanczos process, the work at one step is a dovetail update from the work at the previous step, without the need to store the entire set of basis vectors. Indeed $V_m^T u = [v_1^T u, \ldots, v_m^T u]^T = [(V_{m-1}^T u)^T, v_m^T u]^T$ and $(I - P_m)u = (I - V_m V_m^T)u = (I - v_m v_m^T) \cdots (I - v_1 v_1^T)u$ can be computed incrementally as basis vectors become available. We now give additional details on how $\hat{\alpha}_{m+1}$ can be efficiently updated instead of using just (4). Direct calculation shows that

$$\begin{aligned} \hat{\alpha}_{m+1} \equiv \hat{v}_{m+1}^T A \hat{v}_{m+1} &= \frac{u^T (I - P_m) A (I - P_m) u}{\|(I - P_m)u\|_2^2} \\ &= \frac{u^T A u - u^T V_m T_m V_m^T u - 2\beta_{m+1} (v_{m+1}^T u)(v_m^T u)}{\|(I - P_m)u\|_2^2}. \end{aligned}$$

It is therefore clear that we also have

$$\hat{\alpha}_m = \frac{u^T A u - u^T V_{m-1} T_{m-1} V_{m-1}^T u - 2\beta_m (v_m^T u)(v_{m-1}^T u)}{\|(I - P_{m-1})u\|_2^2}.$$

But further calculation shows that

$$u^T V_m T_m V_m^T u = u^T V_{m-1} T_{m-1} V_{m-1}^T u + 2\beta_m (v_{m+1}^T u)(v_m^T u) + \alpha_m (v_m^T u)^2,$$

and hence we end up with the updating formula

$$\hat{\alpha}_{m+1} \|(I - P_m)u\|_2^2 = \hat{\alpha}_m \|(I - P_{m-1})u\|_2^2 - 2\beta_{m+1} (v_{m+1}^T u)(v_m^T u) - \alpha_m (v_m^T u)^2$$

in which we also have

$$\|(I - P_m)u\|_2^2 = \|(I - P_{m-1})u\|_2^2 + (v_m^T u)^2.$$

Therefore, the vector u interacts with A only through one extra matrix-vector product needed to form the initial $\hat{\alpha}_2$. And from there subsequent updates are made by scalar recurrences using quantities readily available.

Remarks

Rather than adding the augmented vector u at the end, we could have put it at the beginning. And then, we could either 1) orthogonalize u against each incoming vector as before, or 2) orthogonalize each incoming vector against u.

1. In the first case, we would get $\tilde{V}_{m+1} = [\hat{v}_{m+1} \mid V_m]$, with V_m *unchanged* and $\hat{v}_{m+1}$ resulting from orthogonalizing u against each incoming vector as before. This is therefore equivalent to a permutation $\tilde{V}_{m+1} = \hat{V}_{m+1}\widetilde{P}_{m+1}$ where $\widetilde{P}_{m+1} = [e_{m+1}, e_1, \ldots, e_m]$ is the $(m+1)$-by-$(m+1)$ permutation matrix that produces the effect. It follows that
$$\tilde{V}_{m+1}^T A \tilde{V}_{m+1} = \widetilde{P}_{m+1}^T \hat{V}_{m+1}^T A \hat{V}_{m+1} \widetilde{P}_{m+1} = \widetilde{P}_{m+1}^T \hat{T}_{m+1} \widetilde{P}_{m+1}$$
and this shows that the tridiagonal form is destroyed. If we leave out $\hat{v}_{m+1}$ we recover the classical Lanczos quantities, whereas our primary interest in this study is to explore an augmented approach and summarize our findings.
2. If u is added at the beginning and each incoming vector is orthogonalized against it, the resulting matrix from iterating on $\{u, v, Av, \ldots\}$ is not tridiagonal (although the restriction onwards from v is). Augmenting at the beginning is also what is often termed as *locking* or *deflation* when used as a means to accelerate the convergence of eigenvalue problems or linear systems.

It is worth noting that when dealing with the same augmented subspace $\mathcal{K}_m(A, v) + \{u\}$, there exists transformation matrices between the computed bases. Our proposed approximation scheme offers the advantage of retaining the standard Lanczos elements. Existing Lanczos implementations would therefore provide a good starting code base and as we shall see, the scheme is also well suited for establishing further results. It should also be clear such an augmented scheme can be applied to the Arnoldi procedure. But, by itself, the Arnoldi procedure already needs all basis vectors to perform the Gram-Schmidt sweeps.

3 Alternative Approaches

To compute $u^T f(A)v$, our motivation was to avoid keeping the entire set of Lanczos basis vectors. Nor did we want to apply the process twice and incur the associated cost. Aside from leaving out u and embedding $u^T V_m$ in the classical Lanczos scheme as noted earlier, these goals could be achieved by other approaches such as the block Lanczos algorithm and the bi-orthogonal Lanczos algorithm in the ways outlined below.

If started with $[v, u]$, the block Lanczos computes a block tridiagonal matrix and an orthonormal basis V_{2m} of $\mathcal{K}_m(A, v) + \mathcal{K}_m(A, u)$. Since u and v can be expressed in the computed basis as $u = V_{2m}V_{2m}^T u$ and $v = V_{2m}V_{2m}^T v$, an approximation to $u^T f(A)v$ could be retrieved without storing all of V_{2m}. However, doing a block Lanczos in this context is twice the cost of the augmented scheme, although the approach satisfies a particular theoretical characterization as we shall see below.

Another approach is the bi-orthogonal Lanczos algorithm which is usually meant for unsymmetric problems. But it is possible to deliberately apply it here to simultaneously compute a basis of $\mathcal{K}_m(A, v)$ and a basis of $\mathcal{K}_m(A, u)$ from where to approximate $u^T f(A)v$. Below we briefly summarize this well-known algorithm using its general, unsymmetric notation. We do not dwell here on the obvious notational simplifications that can be made since the matrix is symmetric.

ALGORITHM 2: Bi-orthogonal Lanczos

$\beta := \|v\|_2$; $\quad \alpha := (u^T v)/\beta$;
$v_1 := v/\beta$; $\quad u_1 := u/\alpha$;
for $j := 1 : m$ **do**
  $p := Av_j$; $\quad q := A^T u_j$;
  if $j > 1$ **then**
    $p := p - \gamma_j v_{j-1}$; $\quad q := q - \beta_j u_{j-1}$;
  endif
  $\alpha_j := u_j^T p$;
  $p := p - \alpha_j v_j$; $\quad q := q - \alpha_j u_j$; $\quad s := q^T p$;
  $\beta_{j+1} := \sqrt{|s|}$;
  $\gamma_{j+1} := \text{sign}(s)\, \beta_{j+1}$;
  $v_{j+1} := p/\beta_{j+1}$; $\quad u_{j+1} := q/\gamma_{j+1}$;
endfor

Upon completion of this algorithm, we get a tridiagonal matrix

$$T_m = \begin{bmatrix} \alpha_1 & \gamma_2 & & \\ \beta_2 & \alpha_2 & \ddots & \\ & \ddots & \ddots & \gamma_m \\ & & \beta_m & \alpha_m \end{bmatrix},$$

and $V_m \equiv [v_1, \ldots, v_m]$ is a basis of $\mathcal{K}_m(A, v)$, $U_m \equiv [u_1, \ldots, u_m]$ is a basis of $\mathcal{K}_m(A^T, u)$. The bases are bi-orthogonal, i.e.,

$$U_m^T V_m = V_m^T U_m = I \tag{5}$$

and furthermore we have the relations

$$AV_m = V_m T_m + \beta_{m+1} v_{m+1} e_m^T \tag{6}$$

$$U_m^T A V_m = T_m \tag{7}$$

$$A^T U_m = U_m T_m^T + \gamma_{m+1} u_{m+1} e_m^T. \tag{8}$$

From these, we see that $v = \beta V_m e_1$ and $u = \alpha U_m e_1$ and the approximation is computed as $u^T f(A)v = \alpha\beta e_1^T U_m^T f(A) V_m e_1 \approx \alpha\beta e_1^T f(U_m^T A V_m) e_1 = \alpha\beta e_1^T f(T_m) e_1$. And this relationship allows us to avoid storing the bases explicitly. In our symmetric context, we replace A^T with A. Each step requires two matrix-vector products, so it is twice as expensive as of the augmented scheme, but as we shall see below the bi-orthogonal Lanczos algorithm turns out to have certain theoretical characteristics similar to that of the block Lanczos.

4 Some Properties

The notation used here is that introduced earlier for each relevant method. We first start with the proposed augmented process. In analogy with (1) the augmented process satisfies the following relations

$$\begin{aligned}
\hat{V}_{m+1}^T \hat{V}_{m+1} &= I \\
\hat{V}_{m+1}^T A \hat{V}_{m+1} &= \hat{T}_{m+1} \\
A\hat{V}_{m+1} &= \hat{V}_{m+1}\hat{T}_{m+1} + (\beta_{m+1} v_{m+1} - \hat{\beta}_{m+1}\hat{v}_{m+1}) e_m^T - \hat{\beta}_{m+1} v_m e_{m+1}^T + \\
&\quad + (A - \hat{\alpha}_{m+1} I)\hat{v}_{m+1} e_{m+1}^T \\
&= \hat{V}_{m+1}\hat{T}_{m+1} + (I - \hat{v}_{m+1}\hat{v}_{m+1}^T)\Big(\beta_{m+1} v_{m+1} e_m^T + \\
&\quad + (I - v_m v_m^T) A \hat{v}_{m+1} e_{m+1}^T\Big)
\end{aligned}$$

where e_m and e_{m+1} are of length $m+1$. We have already seen that the first and second relations hold by construction. To get the third relation, we write

$$A\hat{V}_{m+1} = A[V_m \;\mid\; \hat{v}_{m+1}] = [V_m T_m + \beta_{m+1} v_{m+1} e_m^T \;\mid\; A\hat{v}_{m+1}]$$

and

$$\begin{aligned}
\hat{V}_{m+1}\hat{T}_{m+1} &= [V_m \;\mid\; \hat{v}_{m+1}] \begin{bmatrix} T_m & \hat{\beta}_{m+1} e_m \\ \hat{\beta}_{m+1} e_m^T & \hat{\alpha}_{m+1} \end{bmatrix} \\
&= [V_m T_m + \hat{\beta}_{m+1}\hat{v}_{m+1} e_m^T \;\mid\; \hat{\beta}_{m+1} v_m + \hat{\alpha}_{m+1}\hat{v}_{m+1}]
\end{aligned}$$

and the relation comes after some algebraic manipulation. The last follows from the fact that $\hat{\alpha}_{m+1} = \hat{v}_{m+1}^T A \hat{v}_{m+1}$ and $\beta_{m+1}\hat{v}_{m+1}^T v_{m+1} = \hat{\beta}_{m+1} = v_m^T A \hat{v}_{m+1}$.

The basis V_m and the tridiagonal matrix T_m computed by the original Lanczos process are known to satisfy $p(A)v_1 = V_m p(T_m)e_1$ for any polynomial p of degree $\leq m-1$ (see, e.g., [9, Lemma 3.1]). This is still valid with the augmented process since it fully retains the original Lanczos elements. Thus we have $p(A)v_1 = \hat{V}_{m+1} p(\hat{T}_{m+1})e_1$ for any polynomial p of degree $\leq m-1$. In fact, we now show that we can extend this property to a higher degree in our context.

Proposition 1. *At the m-th step of the augmented Lanczos process, we have* $u^T p(A)v_1 = u^T \hat{V}_{m+1} p(\hat{T}_{m+1})e_1$ *for any polynomial* p *of degree* $\leq m$.

Proof. Since the result is already true for a degree $\leq m-1$, it remains to establish the degree m. Let $p_m(z) = a_0 + z q_{m-1}(z)$ be a polynomial of degree m in which q_{m-1} is a polynomial of degree $m-1$ and $a_0 = p_m(0)$ is a scalar. We have

$$\begin{aligned}
p_m(A)v_1 &= a_0 v_1 + A q_{m-1}(A)v_1 = a_0 v_1 + A\hat{V}_{m+1} q_{m-1}(\hat{T}_{m+1})e_1 \\
&= a_0 v_1 + \Big(\hat{V}_{m+1}\hat{T}_{m+1} + (I - \hat{v}_{m+1}\hat{v}_{m+1}^T)\Big(\beta_{m+1} v_{m+1} e_m^T + \\
&\quad + (I - v_m v_m^T) A \hat{v}_{m+1} e_{m+1}^T\Big)\Big) q_{m-1}(\hat{T}_{m+1})e_1 \\
&= \hat{V}_{m+1} p_m(\hat{T}_{m+1})e_1 + (I - \hat{v}_{m+1}\hat{v}_{m+1}^T)\Big(\beta_{m+1} v_{m+1} e_m^T + \\
&\quad + (I - v_m v_m^T) A \hat{v}_{m+1} e_{m+1}^T\Big) q_{m-1}(\hat{T}_{m+1})e_1
\end{aligned}$$

Since $\hat{T}_{m+1}$ is a tridiagonal matrix of order $m+1$ and q_{m-1} is a polynomial of degree $m-1$, we have $e_{m+1}^T q_{m-1}(\hat{T}_{m+1})e_1 = 0$. Furthermore $u^T(I - \hat{v}_{m+1}\hat{v}_{m+1}^T)v_{m+1} = 0$.

Proposition 1 means that the approximation is *exact* if the minimal degree of v with respect to the matrix A is $\leq m$. In general, the *a priori* error bounds obtained using functional calculus (as in [7]) for the usual Lanczos scheme for approximating $f(A)v$ can be transposed here. In particular, it is known from the theory of matrix functions [8, Chap.6] that

$$f(A)v = p_{n-1}(A)v \tag{9}$$

where p_{n-1} is the Hermite interpolation polynomial of f at $\lambda(A)$, the spectrum of A. Hence,

$$u^T\hat{w} = \beta u^T\hat{V}_{m+1}f(\hat{T}_{m+1})e_1 = \beta u^T\hat{V}_{m+1}p_m(\hat{T}_{m+1})e_1 = u^T p_m(A)v$$

with p_m being the Hermite interpolation polynomial (of degree m) at the modified 'Ritz' values [1]. The error can be written in a polynomial form as

$$|u^T f(A)v - \beta u^T\hat{V}_{m+1}f(\hat{T}_{m+1})e_1| = |u^T p_{n-1}(A)v - u^T p_m(\hat{T}_{m+1})v|$$

The theory of Krylov subspaces [10] shows that $\lambda(\hat{T}_{m+1})$ not only approximates a larger subset of $\lambda(A)$ as m increases but also approximates it more accurately. The next results shed some light on the behavior of the bi-orthogonal Lanczos process and the block Lanczos process.

Proposition 2. *Let T_m be the resulting tridiagonal matrix at the m-th step of the bi-orthogonal Lanczos algorithm, then $u_1^T p(A)v_1 = e_1^T p(T_m)e_1$ for any polynomial p of degree $\leq 2(m-1)$.*

Proof. Any polynomial p of degree $\leq 2(m-1)$ can be factored as $p(z) = q(z)r(z) + s(z)$ where q, r, s are polynomials of degree $<= m-1$. From this, if follows that $u_1^T p(A)v_1 = (q(A)u_1)^T r(A)v_1 + u_1^T s(A)v_1 = (U_m q(T_m)e_1)^T V_m r(T_m)e_1 + e_1^T U_m^T V_m s_m(T_m)e_1$ and the bi-orthogonality property (5) yields the result.

Proposition 3. *Let T_{2m} be the resulting block-tridiagonal matrix at the m-th step of the block Lanczos algorithm started with MGS(v,u) where MGS stands for the Modified Gram-Schmidt procedure, then $u^T p(A)v = \beta(\omega_1 e_1^T + \omega_2 e_2^T)p(T_{2m})e_1$ for any polynomial p of degree $\leq 2(m-1)$ where ω_1 and ω_2 are the coefficients relative to u from the MGS(v,u) procedure.*

Proof. The block Lanczos procedure will generate an orthonormal basis for $\mathcal{K}_m(A,v) + \mathcal{K}_m(A,u)$, hence the assertion is already true for any polynomial of degree $\leq m-1$. The rest of the proof proceeds as in Proposition 2 using the basis of dimension $2m$ generated by the block Lanczos algorithm.

[1] These are not really the Ritz values since the tridiagonal matrix is augmented in a special way. But since the eigenvalues of $\hat{T}_{m+1}$ embed the Ritz values according to Cauchy's interleaving theorem, they remain close.

We should draw the attention of the reader on the fact that an apparent higher polynomial order of an approach is not synonymous with higher accuracy. A higher degree polynomial does not necessarily convey a better approximation to a *matrix function*. An illustrative case in point is the Taylor series expansion which can still converge slowly even after summing a large number of terms. As relation (9) showed, the underlying approximations are going to depend on how good the spectrum of A is approximated in the respective situations. Since the three approaches usually yield comparable approximations to the spectrum, the augmented approach stands out due to its cheaper cost.

5 Numerical Experiments

There are a number of matrix functions that arise frequently in the literature. In our first experiment, we consider the sine function $f_1(x) = \sin x$. In the second experiment, we consider an example from Druskin and Knizhnerman [4]: the exponential function with square-root $f_2(x) = e^{-\theta\sqrt{x}}$, $\theta \geq 0$. This example results from the boundary value problem

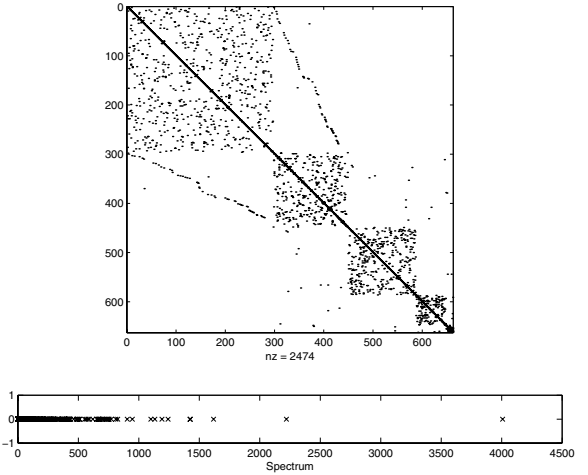
$$Aw - \frac{d^2w}{d\theta^2} = g(\theta)v.$$

The function f_2 is the solution of the boundary value problem $w(0) = v$, $w(+\infty) = 0$ with $g = 0$. References in Druskin and Knizhnerman [4] include applications where this problem arises, for example in geophysical computed tomography. In our experiments, we used the functions on symmetric matrices from the Harwell-Boeing collection.

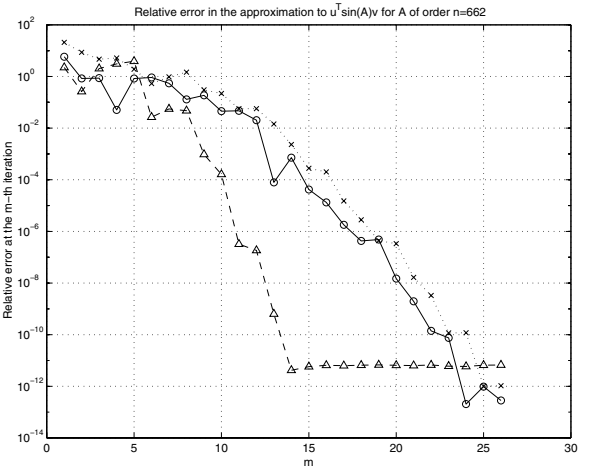
Let f_{exact} be the computed value of $u^T f(tA)v$. For the purpose of the experiments, it was computed by diagonalizing the matrix. Each figure includes three error curves corresponding to $\|f_{\text{exact}} - f_{m^{\text{th}}\text{approx}}\|_2 / \|f_{\text{exact}}\|$, where the m-th approximation, $f_{m^{\text{th}}\text{approx}}$, is either the approximation at the m-th iteration with the proposed augmented scheme (plain curve with circles), the approximation with the bi-orthogonal Lanczos (dashed curved with triangles), or the classical approximation $u^T\tilde{w} = \beta u^T V_m f(tT_m)e_1$ (dotted curve with crosses).

On the reported examples, all methods tend to converge very well as generally observed with Krylov-based approximations to matrix functions. The first example has some interesting observations. The bi-orthogonal Lanczos scheme exhibits faster convergence at the beginning but is subsequently caught up by the other methods. What is even more noticeable is that the convergence of the bi-orthogonal scheme stalls at some point. Further investigation showed us that a marked lost of orthogonality started in the bi-orthogonal process and so the process was contaminated from that point onwards (since we were not dealing with a production code, our testings did not involve re-orthogonalization techniques that may be needed with Lanczos-based algorithms). Since in general, the bi-orthogonal scheme is more sensitive to lost of orthogonality, that is yet another unfavorable argument against using it without cross-checking.

5.1 Example 1



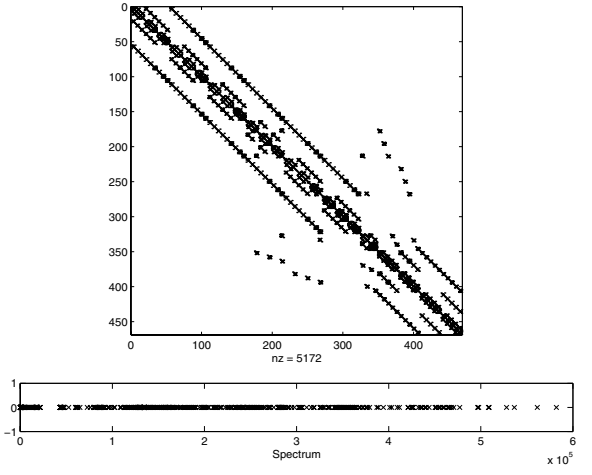
Sparsity pattern and spectrum



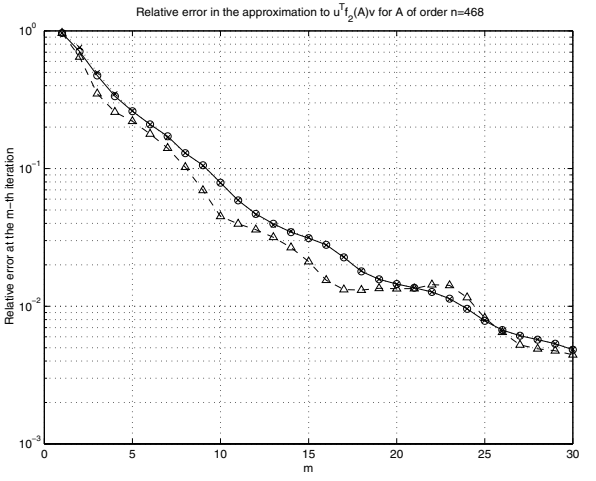
Error curves

Fig. 1. We consider $f_1(x) = \sin x$ and compute $u^T f_1(tA)v$ for random u and v, and a 662-by-662 matrix A known as `662bus` in the Harwell-Boeing collection. The sparsity pattern and spectrum are shown on the left side (the spectrum lies on the real line since A is symmetric). We took $t = 0.01$, for A is of large norm.

5.2 Example 2



Sparsity pattern and spectrum



Error curves

Fig. 2. We consider $f_2(x) = e^{-\theta\sqrt{x}}$, $\theta = 0.01$ and compute $u^T f_2(A)v$ where u and v are randomly generated, and A is the 468-by-468 matrix known as `nos5` in the Harwell-Boeing collection

6 Conclusion

Evaluating $f(A)v$ with the usual Lanczos process has the drawbacks of the storage of the basis vectors or the application of the process twice. Despite the reputed convergence properties of the method, its drawbacks deter some users from using the process in certain situations. We have presented a cost-effective approach for evaluating $u^T f(A)v$ using a suitably augmented Lanczos process that does not have these drawbacks. The proposed method still has the advantage of preserving the original elements, and thus it inherits from the same foundation and offers the recognized quality of Krylov-based methods for approximating matrix functions. Some properties were established and placed in the broader perspective of alternative methods such as the block Lanczos and the bi-orthogonal Lanczos algorithms. Numerical results were made to test the quality of the method on some frequently used matrix functions.

References

1. M. Benzi and G. Golub. Bounds for the entries of matrix functions with applications to preconditioning. *BIT Numerical Mathematics*, 39(3):417–438, 1999.
2. L. Bergamaschi and M. Vianello. Efficient computation of the exponential operator for large, sparse, symmetric matrices. *Numer. Linear Algebra Appl.*, 7:27–45, 2000.
3. D. S. Bernstein and C. F. Van Loan. Rational matrix functions and rank-1 updates. *SIMAX*, 22(1):145–154, 2000.
4. V. Druskin and L. Knizhnerman. Extended krylov subspaces: Approximation of the matrix square root and related functions. *SIMAX*, 19(3):755–771, 1998.
5. G. Golub and G. Meurant. Matrices, moments and quadrature. In D. F. Griffiths and G. A Waston, editors, *Numerical Analysis 93*. Pitman Research Notes in Mathematics 303, Longman Scientific and Technical, 1993.
6. G. Golub and G. Meurant. Matrices, moments and quadrature. ii. how to compute the norm of the error in iterative methods. *BIT*, 37(3):687–705, 1997.
7. M. Hochbruck and Ch. Lubich. On Krylov subspace approximations ot the matrix exponential operator. *SIAM J. Numer. Anal.*, 34(5):1911–1925, October 1997.
8. R. A. Horn and C. R. Johnson. *Topics in Matrix Aanalysis*. Cambridge University Press, Cambridge, 1991.
9. Y. Saad. Analysis of some Krylov subspace approximations to the matrix exponential operator. *SIAM J. Numer. Anal.*, 29(1):208–227, 1992.
10. Y. Saad. *Numerical Methods for Large Eigenvalue Problems*. John Wiley & Sons, Manchester Univ. Press, 1992.
11. Y. Saad and M. H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. and Stat. Comp.*, 3(7):856–869, July 1986.