**Summary**

Many applications require the computation of a few singular values and vectors of a large, sparse matrix. We present a polynomial filtering technique for accelerating such computations. Our method is competitive with existing algorithms and is particularly effective when many singular values are required.

**Background**

Lanczos bidiagonalization is an efficient and scalable method for computing a few leading singular values of a large matrix.

- Using the two-step recurrence

\[
A v_j = \alpha_j u_j + \beta_{j-1} u_{j-1},
\]

\[
A^* u_j = \alpha_j v_j + \beta_{j-1} v_{j+1},
\]

compute matrices \(U_k\) and \(V_k\) with orthonormal columns \(u_1, \ldots, u_k\) and \(v_1, \ldots, v_k\) (Lanczos vectors) such that

\[
B_k = U_k^* A V_k = \begin{bmatrix}
\alpha_1 & \beta_1 \\
\alpha_2 & \beta_2 \\
& \ddots \\
\alpha_k & \beta_{k-1} \\
& & \ldots \\
& & & \alpha_k
\end{bmatrix}
\]

- The singular values of \(B_k\) approximate those of \(A\).
- As \(k\) increases, the largest singular values converge first.
- The better-separated these values are from the rest, the faster the convergence.
- The method engages \(A\) only through matrix-vector products.

**The Problem**

What if we need many singular values, possibly not the leading ones?

- One approach: Keep taking Lanczos steps until all desired values converge.
  - The number of steps needed may be quite large.
  - Each step produces two new Lanczos vectors, increasing memory usage and orthogonalization costs.
- Better idea: Apply a spectral transformation.
  - Move the desired values to the high end of the spectrum.
  - The classic choice is the shift-and-invert transformation—quite effective but expensive for large matrices (must solve linear systems).

**Our Method**

We propose using a polynomial filter to accelerate the computation.

- If \(P\) is a polynomial and

\[
A = U \Sigma V^* \text{ is the SVD of } A,
\]

then

\[
A P(A^* A) U \Sigma P(V^* V) = U \Sigma q(\Sigma) V^*, \quad q(x) = xp(x^2).
\]

By selecting \(p\) so that \(q\) is large on the singular values of interest and small on the rest, the Lanczos algorithm applied to \(A P(A^* A)\) will rapidly pick out singular vectors corresponding to the desired values. We choose \(p\) so that \(q\) is a Chebyshev least-squares approximation to the characteristic function of the interval containing the singular values of interest. Note that \(q\) always has odd symmetry.

**Benefits:**

- Better isolation of the wanted singular values means fewer Lanczos steps are needed for convergence.
- Fewer steps means fewer Lanczos vectors, saving memory and effort spent on orthogonalization.
- Since the filter is a polynomial, the method engages \(A\) only via matrix-vector products—superior scaling to shift-and-invert.
- Interior singular values can be easily computed by choosing a filter that de-emphasizes the extreme ones.
- If many interior singular values are required, multiple search intervals can be processed in parallel.
- The method is easier to implement than restarted Lanczos.

**Numerical Results**

We implemented our method on top of the Lanczos bidiagonalization routines available in the SLEPc library. Some practical details:

- We scale the matrix so that its singular values lie in \([0, 1]\) using an initial estimate of the leading singular value, which we get from a few (\(\approx 10\)) steps of unfiltered Lanczos.
- We employ full reorthogonalization to ensure orthogonality of the computed singular vectors.

Example: We compute the leading 100 singular values of the “dawson5” matrix from the UF Sparse Matrix Collection.

- Problem size: 51,537 × 51,537 with 4,653,901 nonzeros.
- We use our method with degree-11 and degree-17 filters on \([0.87, 1]\).
- We compare with unfiltered Lanczos, the thick-restart Lanczos solver in SLEPc, and svds in MATLAB.

The results are summarized in the following pair of tables. The filtered Lanczos methods use more matrix-vector products but spend far less time on orthogonalization, leading to significant computational savings.

<table>
<thead>
<tr>
<th>Method</th>
<th>Time (s)</th>
<th>Lanczos Steps</th>
<th>Mat-vecs</th>
<th>% of Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unfiltered Lanczos</td>
<td>83</td>
<td>831</td>
<td>1,962</td>
<td>100</td>
</tr>
<tr>
<td>Filtered Lanczos</td>
<td>18</td>
<td>256</td>
<td>5,961</td>
<td>100</td>
</tr>
<tr>
<td>Filtered Lanczos (17)</td>
<td>20</td>
<td>228</td>
<td>8,081</td>
<td>100</td>
</tr>
<tr>
<td>Thick-restart Lanczos</td>
<td>37</td>
<td>228 (6 restarts)</td>
<td>1,872</td>
<td>100</td>
</tr>
<tr>
<td>MATLAB svds</td>
<td>30</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Mat-vecs</th>
<th>Orthogonalization</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unfiltered Lanczos</td>
<td>4</td>
<td>93</td>
<td>3</td>
</tr>
<tr>
<td>Filtered Lanczos</td>
<td>54</td>
<td>41</td>
<td>5</td>
</tr>
<tr>
<td>Filtered Lanczos (17)</td>
<td>67</td>
<td>29</td>
<td>4</td>
</tr>
<tr>
<td>Thick-restart Lanczos</td>
<td>7</td>
<td>84</td>
<td>9</td>
</tr>
</tbody>
</table>

**References**