A Concise Introduction to SVDIFP

1 Introduction

`svdifp` is a Matlab program for computing a few extreme singular values and corresponding singular vectors of an $m \times n$ real matrix C. With $m \geq n$, `svdifp` will return corresponding right singular vectors. Otherwise, it will return corresponding left singular vectors. The underlying algorithm of `svdifp` is an inverse free preconditioned Krylov subspace method for SVD developed in [1].

1.1 Basic Method

Let $\sigma_1 \leq \sigma_2 \leq ... \leq \sigma_n$ be the singular values of C. `svdifp` considers the reformulation of the singular value problem as the symmetric eigenvalue problems:

$$\sigma_1^2 \leq \sigma_2^2 \leq \cdots \leq \sigma_n^2 \text{ are the eigenvalues of } A = C^T C \tag{1}$$

and apply the inverse free preconditioned Krylov subspace projection method of [2] to (1). In an iterative process, assume that x_k is an approximation eigenvector at step k. We construct a new approximation x_{k+1} by the Rayleigh-Ritz projection onto the Krylov subspace

$$K_m(A - \rho_k I, x_k) := \text{span}\{x_k, (A - \rho_k I)x_k, ..., (A - \rho_k)^{m-1} x_k\}$$

where $\rho_k = \rho(x_k) := x_k^T A x_k / x_k^T x_k$ is the Rayleigh quotient and m is a parameter to be chosen. To deal with the issue that there may be a great loss of accuracy with a very small singular value, we construct a two-sided projection of C, from which we compute approximate singular values directly. Specifically, let V_m be the matrix consisting of the basis vectors of $K_m(A - \rho_k I, x_k)$. We then form the matrices B_m by orthogonalize the columns of CV_m, i.e. $W_m B_m = CV_m$ where W_m is orthonormal. After finding the smallest singular pair (μ_1, u_1) of B_m, μ_1 is taken as the new approximate singular value and the corresponding new approximate right singular vector is

$$x_{k+1} = V_m u_1,$$

and $\rho_{k+1} = \rho(x_{k+1})$. See [2] for convergence analysis.

1.2 Preconditioning Technique

According to Corollary 3.5 of [2], we would like to speed up the convergence by increasing the spectral gap by preconditioning. Let L be the factor in the LDL^T factorization of $C^TC - \rho_k I$ with D being a diagonal matrix of 0 and 1. Applying `svdifp` to the preconditioned matrix

$$L^{-1}(A - \rho_k I)L^{-T}$$

which has exactly the same eigenvalues as A will result in a faster convergence. The preconditioning transformation can be carried out implicitly, see [3].

We employ Robust Incomplete Factorization[4] to obtain the preconditioner L in our algortihm. The idea of RIF is to apply Gram-Schmidt process with respect to $\langle x, y\rangle = x^T(C^TC - \mu I)y$ to $Z = I$:

$$z_i = z_i - \frac{\langle Cz_i, Cz_j\rangle - \mu\langle z_i, z_j\rangle}{\langle Cz_j, Cz_j\rangle - \mu\langle z_j, z_j\rangle} z_j \tag{2}$$

for $j = 1, 2, ..., n$ and $i = j+1, ..., n$. The preconditioner $L = [l_{ij}]$ where

$$l_{ij} = \frac{Cz_i, Cz_j\rangle - \mu\langle z_i, z_j\rangle}{\sqrt{\langle Cz_j, Cz_j\rangle - \mu\langle z_j, z_j\rangle}}.$$

2 Implementation Details

2.1 Robust Incomplete Factorization

M. Benzi and M. Tuma provide code of RIF in Fortran in Sparslab. In our implementation we build a C subroutine `RIF` into Matlab function using MEX-functions from Matlab.

By the property of Gram-Schmidt process and the fact that Z is upper triangular, we can simplify (2) to be

$$z_i = z_i - \frac{\langle Ce_i, Cz_j\rangle}{\langle Cz_j, Cz_j\rangle - \mu\langle z_j, z_j\rangle} z_j$$

for $j = 1, 2, ..., n$ and $i = j+1, ..., n$.

To obtain a sparse preconditioner, we set three thresholds in the process of RIF: `rifthresh, zrifthresh, rifnnz`. `rifthresh` controls the sparsity of L. If

$$l_{ij} <= \texttt{rifthresh}\|Ce_i\|$$

we will skip (2) and the corresponding $l_{ij} = 0$. `zrifthresh` and `rifnnz` controls the sparsity of Z in Gram-Schmidt process. In z_i, if $z_{li} <$ `zrifthresh` $\|z_i\|$, set $z_{li} = 0$. `rifnnz` is the number of non zeros allowed in each column of Z. In our implementation, the lowest `rifnnz` non zeros in each column of Z will be stored and others will be disgarded.

2.2 SVDIFP

`svdifp` inherits many implementation techniques from `eigifp`[3], though there are still some differences.

In the case of finding multiple smallest singular values, we use deflation technique. Suppose p singular values have been found, let V_p be the matrix consisting of the p corresponding right singular vectors with $V_p^T V_p = I$ and $\Lambda_p \in \mathbb{R}^{p\times p}$ be the diagonal matrix with $\tilde{\lambda}_i - \sigma_i^2$ as diagonals, where $\tilde{\lambda}_i$ are chosen such that $\tilde{\lambda}_i - \sigma_i^2 >= \sigma_n^2$. Then, we consider

$$C_p = K_1 C + K_2 V_p^T,$$

where $K_1 \in \mathbb{R}^{(m+p)\times m}, K_2 \in \mathbb{R}^{(m+p)\times p}$ and $K_1^T K_1 = I$, $K_2^T K_2 = \Lambda_p, K_1^T K_2 = O$. The singular values of C_p are

$$\sigma_{p+1} \leq \sigma_{p+2} \leq ... \leq \sigma_n \leq \sqrt{\tilde{\lambda}_i - \sigma_i^2}, i = 1, 2, ..., p.$$

To find the largest singular value of C, we take the largest singular value of the projected matrix as the approximate singular value. We can also get multiple largest singular values through deflation by letting

$$C_p = C - C V_p V_p^T$$

whose singular values are

$$0 \leq ... \leq \sigma_1 \leq ... \leq \sigma_{n-p}.$$

3 Usage

Since `svdifp` calls `RIF` which is a MEX function, we have to call

```
>>mex -largeArrayDims RIF.c
```

or

```
>>mex RIF.c
```

in Matlab console first. You probably need to run

```
>>mex -setup
```

to choose an appropriate C compiler before you compile `RIF`.

The most basic call to `svdifp` is

```
>>[S,V] = svdifp(A)
```

where A is an $m \times n$ matrix in sparse format. This returns the smallest singular value of A `S` and its corresponding right singular vector `V` if $m \geq n$ or left singular vector if $m < n$.

To compute the k smallest singular values of the matrix A, one appends the value k to the above call

```
>>[S,V] = svdifp(A,k)
```

where $k \geq 1$ is an integer. Then the return results are a vector of k smallest singular values `S` and an $n \times k$(if $m \geq n$) matrix of the corresponding singular vectors `V`.

To compute the k largest singular values, we call

```
>>[S,V] = svdifp(A,k,'L')
```

where 'L' stands for 'Largest'.

`svdifp` also uses an option struture to provide user a way to specify the parameters of the algorithm. This can be done by setting values in a structure and then pass it calling

```
>>[S,V] = svdifp(A,k,opt)
```

or

```
>>[S,V] = svdifp(A,k,'L',opt)
```

Here is a description of all members in the option structure:

Table 1: Members of Option in svdifp

`initialvec`	`A matrix whose` i`-th column is the` i`-th intial approximate right singular vector.`
`tolerance`	`Termination tolerance for the 2-norm of residual:` $\|C^TCv - \sigma v\|_2$`. Default:` $10\epsilon\sqrt{n}\|C\|_2^2$`.`
`maxit`	`Set the maximum number of outer iteration. Default: 1000.`
`innerit`	`Set a fixed inner iteration to control the memory requirement. Default: between 1 and 128 as adaptively determined.`
`useprecon`	`Set to 0 to disable preconditioning. Default: 1`
`shift`	`Set shift to be an approximated singular value. Default: 0 for smallest singular value.`
`adshift`	`Set adaptiveshift to 1 to choose shift adaptively. Default: 0 for smallest singular value and 1 for largest singular value.`
`rifthresh`	`A thershold between 0 and 1 used in RIF for computing preconditioner. Default: 1e-3.`
`zrifthresh`	`A threshold between 0 and 1 used for dropping the` Z`-factor in RIF. Default:` $10\epsilon\sqrt{n}\|C\|_2^2$`.`
`rifnnz`	`A number between 1 and` n `which preallocates the nonzeros in each column of` Z `in RIF. Defalut: 1000.`
`disp`	`Set to 0 to disable on-screen display of output, and to other numerical value to enable display. Default: 1.`

References

[1] Qiao Liang and Qiang Ye. Computing Singular Values of Large Matrices With Inverse Free Preconditioned Krylov Subspace Method. *Submitted*.

[2] Gene H. Golub and Qiang Ye. An Inverse Free Preconditioned Krylov Subspace Method for Symmetric Generalized Eigenvalue Problem. *SIAM J. Sci. Comp.*, 24:312-334, 2002.

[3] James H. Money and Qiang Ye. Algorithm 845: EIGIFP: A MATLAB Program for Solving Large Symmetric Generalized Eigenvalue Problems. *ACM Trans. Math. Softw.*, 31:270-279, 2005.

[4] Michele Benzi and Miroslav Tuma. A Robust Incomplete Factorization Preconditioner for Positive Definite Matrices. *Num. Lin. Alg. Appl.*, 10:385-400, 2003.