Robust SVD Method for Missing Value Estimation of DNA Microarrays

Fen Qin, Joseph Collins, and Jeonghwa Lee
Department of Computer Science, Shippensburg University, Shippensburg, PA, U.S.A.

Abstract—*A majority of DNA microarray datasets contain missing or corrupt values and it is critical to estimate these values accurately. These missing values are most often attributed to insufficient experimental resolution or the presence of foreign objects on the experimental slide's surface. To improve existing missing value estimation algorithms, this paper introduces and investigates the scalable singular value decomposition (SSVD) solver, which is an improvement upon the Jacobi singular value decomposition (SVD) approach. Experiments were conducted on a study comparing SSVD to the Jacobi and QR SVD methods against several legitimate microarray datasets. The robustness of SSVD is verified by subjecting it to several test cases containing 1–20% of missing values. For nearly all of the test cases across all configurations of missing value percentages, SSVD provides more accurate recovery results than Jacobi and SQ SVD. These numerical results strongly suggest SSVD is a robust and scalable solver.*

Keywords: Microarrays, missing value estimation, singular value decomposition

1. Introduction

Deoxyribonucleic acid (DNA) microarray analysis is the study of large scale gene expression experiments, which grants researchers insight into solving many pertinent biological questions [12] including cancer classification, identifying the effects of specific gene therapies and exploring the unknown gene function [11]. The microarray data generated by gene expression experiments is presented as one large matrix consisting of genes ordered by rows and experimental conditions by columns [3]. Even though DNA microarray analysis is an emerging and powerful tool for researchers to utilize, the data produced by microarray experiments is typically not complete. Missing or corrupt data is most commonly attributed to insufficient resolution, image corruption or foreign objects such as dust or scratches on the surface of the experimental slide [8]. Incomplete microarray data is undesirable because complete datasets are a prerequisite for existing gene expression data analysis algorithms. If a microarray dataset contains missing values, then researchers are unable to properly draw conclusions about the gene expression experiments.

There are several gene expression data analysis algorithms available for missing value recovery, including the singular value decomposition imputation (SVDimpute) [7], weighted k-nearest neighbors imputation (KNNimpute) [10], least squares imputation (LSimpute) [2], local least squares imputation (LLSimpute) [9] and dynamic local least squares imputation (DLLSimpute) [6]. Among these methods, LLSimpute has been suggested to be an efficient recovery method for microarray datasets. The solving force behind LLSimpute is the orthogonal-triangular decomposition algorithm powered by QR factorization, denoted the QR singular value decomposition (SVD) method. When the SVD routine of the aforementioned algorithms is changed, the overall accuracy of missing value estimation may improve.

Based on its ease of implementation, QR SVD is typically the solution of choice for software written in MATLAB. Within the MATLAB package, there exists a number of approaches that these gene expression data analysis algorithms commonly utilize to calculate the inverse of a matrix. The approaches for which we are concerned, referred to as the standard MATLAB solvers, are $inv(A)$ and $pinv(A)$, which form the explicit inverse and the Moore-Penrose pseudoinverse of a square matrix, respectively. Algorithms from articles [4], [5], denoted the Jacobi SVD method, propose an improved SVD algorithm driven by an advanced matrix inverse approach.

This paper introduces a new SVD algorithm, referred to as the scalable singular value decomposition (SSVD) solver, which is a further improvement upon the Jacobi SVD implementation, designed to improve the accuracy of missing value estimation methods. In order to compare these solvers in a fair and unbiased manner, each solver was tested with four complete microarray datasets. Completing these microarray datasets was achieved by recovering the missing values in each dataset using LLSimpute with the SSVD solver. Test cases were then created by randomly inducing missing values of various percentages into these artificially completed datasets. The experimental results for each test case were achieved by swapping out the SVD solver within LLSimpute.

This paper is organized as follows: Section 2 gives a concise introduction to the implementation of LLSimpute, the implementation of an SVD solver and the improvements introduced by SSVD. In Section 3, numerical experiments of SSVD versus Jacobi and QR SVD are presented to highlight the improved accuracy and scalability of SSVD. Concluding remarks are made in Section 4.

2. Local Least Squares Imputation

2.1 Selecting the Target Gene

$G \in R^{m\times n}$ denotes a gene expression data matrix with m genes and associated n experiments. Assume $m >> n$. In the matrix G, a row ${g_i}^T \in R^{1\times n}$ represents the i-th gene of n experiments as

$$G = \begin{pmatrix} g_1^T \\ \vdots \\ g_m^T \end{pmatrix} \tag{1}$$

and each missing value location at the i-th gene and j-th experiment will be represented as

$$G(i,j) = g_i(j) = \begin{pmatrix} g_{1,1} & \cdots & g_{1,j} & \cdots & g_{1,n} \\ \vdots & & \vdots & & \vdots \\ g_{i,1} & \cdots & g_{i,j} & \cdots & g_{i,n} \\ \vdots & & \vdots & & \vdots \\ g_{m,1} & \cdots & g_{m,j} & \cdots & g_{m,n} \end{pmatrix},$$

where $i \in (1, 2, \cdots, m)$ and $j \in (1, 2, \cdots, n)$. For consistency, we assume that all missing value estimation algorithms discussed throughout this paper consider the first position of the first gene to be a missing value, i.e.

$$G(1,1) = g_1(1) = \beta,$$

where this first gene is selected as the target gene.

2.2 Missing Value Recovery Using SVD

The k-nearest neighbor genes of the target gene are selected where $1 < k < m$ and $k > n$. The matrix $A \in R^{k\times(n-1)}$ and vector $b \in R^{k\times 1}$ are formed from these k-nearest neighbor genes. The vector $w \in R^{1\times(n-1)}$ is formed from the target gene. Local least squares methods solve the following equation:

$$\min_x \left\| A^T x - w \right\|_2, \tag{2}$$

where solving Eq. (2) is equivalent to solving

$$\min_x \left\| A^T x - w \right\|_2^2. \tag{3}$$

By the definition of the inner product, we have

$$\min_x \left\| A^T x - w \right\|_2^2 = \min_x (A^T x - w)^T (A^T x - w). \tag{4}$$

Eq. (4) is equivalent to

$$\frac{\partial}{\partial x_j} \sum_{i=1}^{n-1} (A^T x - w)_i^2 = 2 \sum_{j=1}^{n-1} (A^T x - w)_j^T A_j^T = 0,$$

$$j = 1, 2, \cdots, k, \tag{5}$$

where $(A^T x - w)_i$ is the i-th component of the column vector. Eq. (5) comes down to the critical point of $\left\| A^T x - w \right\|_2^2$, where

$$\sum_{j=1}^{n-1} A_j (A^T x - w)_j = 0, \ j = 1, 2, \cdots, k, \tag{6}$$

and a vector form

$$\begin{pmatrix} A_1(A^T x - w)_1 \\ A_2(A^T x - w)_2 \\ \cdots \\ A_{n-1}(A^T x - w)_{n-1} \end{pmatrix} = A(A^T x - w) = 0. \tag{7}$$

The right hand side of Eq. (7) transforms into

$$\begin{pmatrix} g_1^T \\ g_{s_1}^T \\ \vdots \\ g_{s_k}^T \end{pmatrix} = \begin{pmatrix} \beta & w_1 & w_2 & \cdots & w_{n-1} \\ b_1 & A_{1,1} & A_{1,2} & \cdots & A_{1,n-1} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ b_k & A_{k,1} & A_{k,2} & \cdots & A_{k,n-1} \end{pmatrix},$$

$$g_{s_i}^T = \begin{pmatrix} b_i & A_{i,1} & A_{i,2} & \cdots & A_{i,n-1} \end{pmatrix},$$

where

$$g_1^T = \begin{pmatrix} \beta & w_1 & w_2 & \cdots & w_{n-1} \end{pmatrix},$$

$$\begin{pmatrix} b_1 \\ \vdots \\ b_k \end{pmatrix} = \begin{pmatrix} g_{s_1}(1) \\ \vdots \\ g_{s_k}(1) \end{pmatrix},$$

$$A = \begin{pmatrix} A_{1,1} & A_{1,2} & \cdots & A_{1,n-1} \\ \vdots & \vdots & \cdots & \vdots \\ A_{k,1} & A_{k,2} & \cdots & A_{k,n-1} \end{pmatrix},$$

and g_1^T is a gene with a missing value (depicted as β in the first location of g_1^T) and $g_{s_i}^T, i = 1, 2, \cdots, k$, are the k-nearest neighbor gene vectors for g_1^T. Solutions of Eq. (7) involve the generalized inverse. If matrix A is invertible, we have

$$AA^T x = Aw \tag{8}$$

and the solution

$$x = (AA^T)^{-1} Aw = (A^T)^\dagger w, \tag{9}$$

where $(A^T)^\dagger = (AA^T)^{-1} A$, and $(A^T)^\dagger$ is the pseudoinverse of A^T. The missing value (β) can then be solved as follows:

$$\beta = \sum_{i=1}^{n-1} x_i b_i = b^T (A^T)^\dagger w. \tag{10}$$

Table 1: Error comparison of the standard MATLAB solvers vs. Jacobi SVD

Test Case	α	$\|\|AA^{\dagger}A-A\|\|_{\infty}$	$\|\|A^{\dagger}AA^{\dagger}-A^{\dagger}\|\|_{\infty}$	$\|\|(AA^{\dagger})^{T}-AA^{\dagger}\|\|_{\infty}$	$\|\|(A^{\dagger}A)^{T}-A^{\dagger}A\|\|_{\infty}$	$\|\|Ax-b\|\|_{\infty}$
$inv(A)$	n/a	5.8272E+00	4.1592E+20	1.2275E+02	3.1764E+13	5.9128E+04
$pinv(A)$	n/a	7.1710E-06	7.5940E+08	4.7020E-04	9.3850E-04	1.6750E-04
Jacobi SVD	1.00	1.1430E-05	3.7480E+07	1.4290E-04	1.0210E-03	2.3480E-04

Table 2: Error comparison of Jacobi SVD vs. SSVD

Test Case	α	$\|\|AA^{\dagger}A-A\|\|_{\infty}$	$\|\|A^{\dagger}AA^{\dagger}-A^{\dagger}\|\|_{\infty}$	$\|\|(AA^{\dagger})^{T}-AA^{\dagger}\|\|_{\infty}$	$\|\|(A^{\dagger}A)^{T}-A^{\dagger}A\|\|_{\infty}$	$\|\|Ax-b\|\|_{\infty}$
Jacobi SVD	0.55	3.6440E-10	2.8770E-02	1.9820E-08	1.5680E-08	9.6300E-05
	0.60	6.5320E-10	2.6639E+00	1.1930E-07	6.9380E-08	4.0300E-05
	0.65	6.6790E-09	2.7378E+02	4.5570E-07	5.0710E-07	1.4780E-05
	0.70	2.3680E-08	1.7679E+03	2.1580E-06	3.4820E-06	5.8480E-06
	0.75	4.1550E-07	1.6235E+04	3.1180E-05	2.6530E-05	1.0030E-05
SSVD	0.55	2.9390E-10	1.9540E-02	3.4340E-09	2.3480E-08	9.6300E-05
	0.60	1.3730E-09	3.9960E-01	4.9080E-09	3.9740E-08	4.0310E-05
	0.65	9.6320E-09	1.6243E+01	2.0710E-07	6.5240E-08	1.4770E-05
	0.70	2.7090E-08	6.9364E+02	1.0470E-06	1.0320E-06	5.9650E-06
	0.75	1.4630E-07	1.5730E+04	2.2050E-06	1.1560E-05	5.0670E-06

2.3 Improvement of the SVD Solver

The result obtained by the previous method to calculate the pseudoinverse of the matrix $A \in R^{m\times n}$,

$$A^{\dagger} = V \begin{bmatrix} \Sigma_{r_A}^{-1} & 0 \\ 0 & 0 \end{bmatrix} U^T = V_{r_A}\Sigma_{r_A}^{-1}U_{r_A}^T, \qquad (11)$$

does not satisfy the 4 Moore-Penrose equations [1]:

$$\begin{gathered} AA^{\dagger}A = A, \\ A^{\dagger}AA^{\dagger} = A^{\dagger}, \\ (AA^{\dagger})^{T} = (AA^{\dagger}), \\ (A^{\dagger}A)^{T} = (A^{\dagger}A). \end{gathered}$$

As a result, if the size of the matrix is increased, the number of computational errors is also increased—that is, the SVD results become less accurate. There are five different ways to test the accuracy of the pseudoinverse:

$$\begin{gathered} ||AA^{\dagger}A-A||_{\infty}, \\ ||A^{\dagger}AA^{\dagger}-A^{\dagger}||_{\infty}, \\ ||(AA^{\dagger})^{T}-AA^{\dagger}||_{\infty}, \\ ||(A^{\dagger}A)^{T}-A^{\dagger}A||_{\infty}, \\ ||Ax-b||_{\infty}. \end{gathered}$$

In this paper, the Hilbert matrix,

$$A = H_{200*200} = (\frac{1}{i+j+1})_{200*200}, \qquad (12)$$

is used to benchmark a solver's robustness, scalability and accuracy. As shown in Table 1, $pinv(A)$ does not satisfy $A^{\dagger}AA^{\dagger} = A^{\dagger}$ and $inv(A)$ does not satisfy any of the 4 Moore-Penrose equations, because—in both cases—their respective errors are too large.

The Jacobi SVD method is an improvement upon the commonly implemented QR SVD solver. The procedure for the Jacobi SVD solver is as follows:

$$A = Q \begin{bmatrix} R_{m\times n} \\ 0_{(m-n)\times n} \end{bmatrix}, \qquad (13)$$

where $R_{m\times n}$ is the upper triangular matrix. If $r_A < n$, $R_{m\times n}^T$ is further decomposed by QR SVD to get

$$\begin{gathered} R_{m\times n}^T = P \begin{bmatrix} \tilde{R}_{r_A\times r_A} \\ 0 \end{bmatrix}, \\ A = Q \begin{bmatrix} \tilde{R}_{r_A\times r_A} & 0 \\ 0 & 0 \end{bmatrix} P^T. \end{gathered} \qquad (14)$$

Jacobi SVD is then used to solve $R_{m\times n}$ or $\tilde{R}_{r_A\times r_A}$. From Eq. (13)-(14) we have

$$R_{m\times n} = U_R\Sigma_{r_A}V_R^T,\ A = Q_{r_A}U_R\Sigma_{r_A}V_{R,r_A}^T, \qquad (15)$$

or

$$\tilde{R}_{m\times n} = U_{\tilde{R}}\Sigma_{r_A}V_{\tilde{R}}^T,\ A = Q_{r_A}U_{\tilde{R}}\Sigma_{r_A}V_{\tilde{R},r_A}^T P_{r_A}^T, \qquad (16)$$

and the pseudoinverse ($A^{\dagger}$) is

$$A^{\dagger} = V_R\Sigma_{r_A}^{-1}U_R^TQ_{r_A}^T \quad \text{or} \quad A^{\dagger} = PV_{\tilde{R}}\Sigma_{r_A}^{-1}U_{\tilde{R}}^TQ_{r_A}^T. \qquad (17)$$

As seen in Table 1, the error associated with the Jacobi SVD solver for $A^{\dagger}AA^{\dagger} = A^{\dagger}$ is smaller than the results produced by the standard MATLAB solutions, yet this value is still too large to be acceptable.

SSVD is a further improvement upon the Jacobi SVD solver, aiming to reduce the overall size of these errors. Since U and V from Eq. (11) are orthogonal matrices, they do not lead to an error, which means the computational errors are

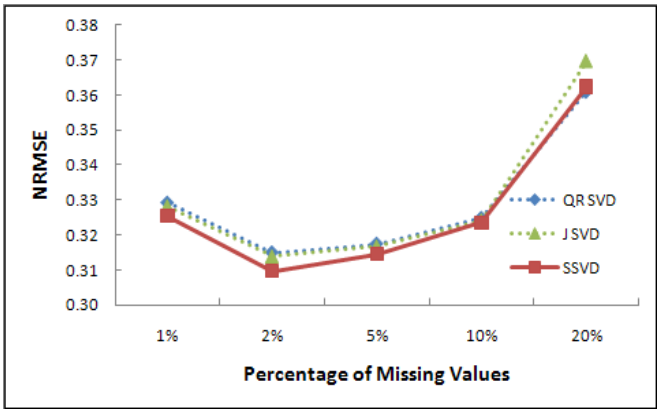

Fig. 1. NRMSE result for the YO microarray dataset

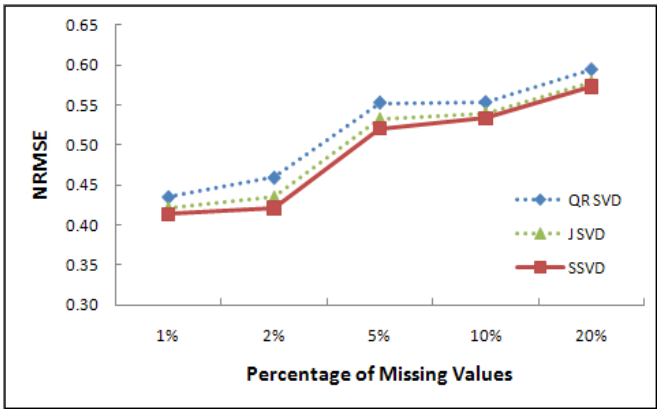

Fig. 2. NRMSE result for the CU microarray dataset

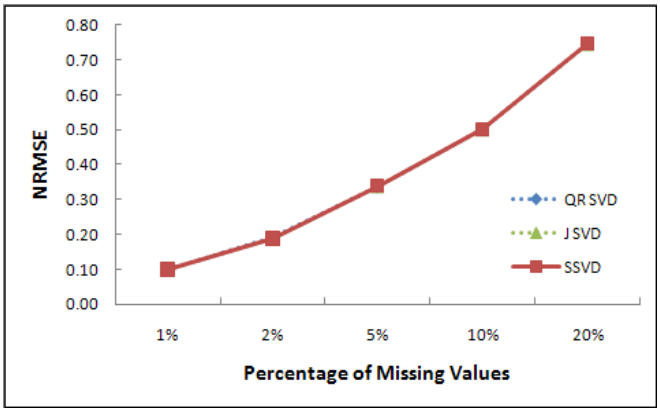

Fig. 3. NRMSE result for the RO microarray dataset

Fig. 4. NRMSE result for the SP microarray dataset

Table 3: NRMSE comparison of QR SVD, Jacobi SVD and SSVD

Test Case	SVD Solver	1%	2%	5%	10%	20%
YO.Calcineurin/Crzlp	QR SVD	0.3292	0.3149	0.3174	0.3248	0.3610
	Jacobi SVD	0.3282	0.3139	0.3167	0.3242	0.3696
	SSVD	0.3254	0.3097	0.3145	0.3235	0.3624
CU.Growth-regulator	QR SVD	0.4349	0.4591	0.5526	0.5531	0.5938
	Jacobi SVD	0.4214	0.4351	0.5318	0.5393	0.5779
	SSVD	0.4134	0.4209	0.5208	0.5334	0.5727
RO.Cellline	QR SVD	0.0988	0.1907	0.3390	0.5021	0.7467
	Jacobi SVD	0.0991	0.1901	0.3391	0.5017	0.7463
	SSVD	0.0988	0.1882	0.3391	0.5010	0.7457
SP.Alpha	QR SVD	0.3186	0.4557	0.5481	0.6064	0.6753
	Jacobi SVD	0.3183	0.4556	0.5398	0.6010	0.6702
	SSVD	0.3175	0.4454	0.5387	0.5974	0.6657

produced by $\Sigma_{r_A}^{-1}$. It is important to note that matrix Σ_{r_A} is dependent on the accuracy of the system executing its implementation; if the singular values $\sigma_i < eps$ (eps is the error bound), the system will set it to zero, then Eq. (11) will contain computational errors.

Suppose the singular values of matrix A are $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{r_A}$, then

$$\Sigma_{r_A}^{-1} = diag(\frac{1}{\sigma_1}, \frac{1}{\sigma_2}, \cdots, \frac{1}{\sigma_{r_A}}, 0, \cdots, 0). \quad (18)$$

Since the cumulative error is $\varepsilon_i, i = 1, 2, \cdots, r_A$, the singu-

lar values of matrix A will become $\sigma_i+\varepsilon_i, i = 1, 2, \cdots, r_A$. Therefore,

$$\Sigma_{r_A}^{-1} = diag(\frac{1}{\sigma_1+\varepsilon_1}, \frac{1}{\sigma_2+\varepsilon_2}, \cdots, \frac{1}{\sigma_{r_A}+\varepsilon_{r_A}}, 0, \cdots, 0), \tag{19}$$

where the maximum computing error is

$$\varepsilon\left(\Sigma_{r_A}^{-1}\right) = diag\left(\begin{array}{c} \frac{|\varepsilon_1|}{\sigma_1|\sigma_1+\varepsilon_1|}, \frac{|\varepsilon_1|}{\sigma_2|\sigma_2+\varepsilon_2|}, \\ \cdots, \frac{|\varepsilon_1|}{\sigma_{r_A}|\sigma_{r_A}+\varepsilon_{r_A}|}, 0, \cdots, 0 \end{array}\right). \tag{20}$$

If $\sigma_i+\varepsilon_i$ is close to zero, the error is significantly magnified, hence it should be set to zero. Eq. (19) becomes as follows:

$$\Sigma_{r_A}^{-1}(\alpha) = diag(\frac{1}{\sigma_1+\varepsilon_1}, \frac{1}{\sigma_2+\varepsilon_2}, \cdots, \frac{1}{\sigma_k+\varepsilon_k}, 0, \cdots, 0),$$
$$\sigma_i+\varepsilon_i \leq eps^{\alpha}, i = k+1, \cdots, r_A. \tag{21}$$

Experimental results using Eq. (21) are shown in Table 2. When α = 0.75, the errors associated with SSVD are much smaller than those presented in Table 1. When α = 0.55, the pseudoinverse of Hilbert matrix ($A = H_{200*200}$) does satisfy the 4 Moore-Penrose equations, yet the solution of $||Ax-b||_{\infty}$ for α = 0.55 is worse than α = 0.75, which is due to the removal of more singular values that are close to zero. Therefore, α cannot be too small, and the empirically chosen value of α = 0.75 is used in this research.

3. Numerical Results

The normalized root mean squared error (NRMSE) was used to measure the accuracy of the results from the test cases.

$$\text{NRMSE} = \sqrt{\frac{mean[(\gamma_{estimated}-\gamma_{known})]^2}{std[\gamma_{known}]}}, \tag{22}$$

where $\gamma_{estimated}$ are the estimations for missing values, and γ_{known} are the known values. The mean and the standard deviation are calculated over missing values in the whole dataset.

The NRMSE test results of Eq. (22) for various percentages (1%, 2%, 5%, 10% and 20%) of missing values for QR SVD, Jacobi SVD and SSVD are presented in Table 3. For an overwhelming majority of test cases, the SSVD method generates more accurate recovery results than QR SVD, and, for all test cases, SSVD consistently performed better than Jacobi SVD. Only in the YO.Calcineurin/Crzlp test case for 20% of missing values and the RO.Cellline test case for 5% of missing values did QR SVD outperform our proposed SSVD solver; however, the difference in performance for the latter test case is so insignificant that it may be regarded as an equal level of performance between the two solvers.

The graphical representations of the results from Table 3 are located in Fig. 1 through Fig. 4. Note that Jacobi SVD is referred to as J SVD within the legend of a figure. Each figure is oriented with the experimental NRMSE results in the y-axis and the various percentages of missing values in the x-axis. A data trend favoring the lower end of the NRMSE scale is favorable because this represents a series of experimental results with smaller levels of erroneous estimations. The scale of each figure is not consistent, thus is insufficient to gauge the performance between datasets based solely on the distance of the separation between their respective data trend lines. These figures further illustrate the improvement in accuracy associated with the SSVD solver when tested with the YO.Calcineurin/Crzlp, CU.Growth-regulator, RO.Cellline and SP.Alpha microarray datasets, respectively.

Fig. 2 depicts the most exciting results of the four microarray datasets. The difference in accuracy between these solvers for these specific test cases is quite substantial, granting significant increases in the accuracy of missing value estimation. Fig 1. and Fig 4. show SSVD's typical outcome, which is a marginal increase in accuracy with respect to Jacobi and QR SVD. Fig. 3 illustrates the worst-case scenario for SSVD—the level of accuracy is nearly equal, yet slightly better, to that of the QR SVD solver.

4. Conclusion

We have successfully developed a scalable solver for estimating the missing values of DNA microarray datasets. For nearly all the test cases across all configurations of missing value percentages, SSVD provides more accurate recovery results than Jacobi and QR SVD. The numerical results presented in this paper strongly suggests that SSVD is a robust, scalable and accurate solver. One would be safe to assume that the benefits from SSVD may be realized in many other disciplines and not those limited to missing value estimation.

References

[1] A. Albert, *Regression and the Moore-Penrose pseudoinverse*, Acdemic Press, INC, New York, 1972.

[2] T. H. BZ, B. Dysvik and I. Jonassen, *LSimpute: accurate estimation of missing values in microarray data with least squares methods*, Nucleic Acids Res 32 (3) (2004) e34.

[3] Z. Cai, M. Heydari and G. Lin, *Iterated local least squares microarray missing value imputation*, Journal of Bioinformatics and Computational Biology 4 (5) (2006) 935–957.

[4] Z. Drmac and K. Veselic, *New Fast and Accurate Jacobi SVD Algorithm I*, SIAM Journal on Matrix Analysis and Applications, 29 (4) (2008) 1322–1342.

[5] Z. Drmac and K. Veselic, *New Fast and Accurate Jacobi SVD Algorithm II*, SIAM Journal on Matrix Analysis and Applications, 29 (4) (2008) 1343–1362.

[6] Q. Fen and J. Lee, *Dynamic Methods for Missing Value Estimation for DNA Sequences*, International Conference on Computational & Information Sciences (ICCIS), 2010 442–445.

[7] T. Hastie, R. Tibshirani, G. Sherlock, M. Eisen, P. Brown and D. Botstein, *Imputing missing data for gene expression arrays*, Technical Report, Division of Biostatistics, Stanford University, 1999.

[8] P. Johansson and J. Hakkinen, *Improving missing value imputation of microarray data by using spot quality weights*, BMC Bioinformatics 7 (2006) 306.

[9] H. Kim, G. H. Golub and H. Park, *Missing value estimation for DNA microarray gene expression data: local least squares imputation*, Bioinformatics 21 (2) (2005) 187–198.
[10] O. Troyanskaya, M. Cantor, G. Sherlock, P. Brown, T. Hastie, R. Tibshirani, D. Botstein and R.B. Altman, *Missing value estimation methods for DNA microarrays*, Bioinformatics 17 (6) (2001) 520–525.
[11] D. Yoon, E. Lee and T. Park, *Robust imputation method for missing values in microarray data*, BMC Bioinformatics 8 (Suppl 2):S6 (2007).
[12] A. Zien, J. Fluck, R. Zimmert and T. Lengauer, *Microarrays: how many do you need?*, Journal of Computational Biology 10 (3-4) (2003) 653–667.