# Algorithm stabilization techniques and their application to symbolic computation of generalized inverses

Hiroyuki Minakuchi [†] [*], Hiroshi Kai [†] [†],
Kiyoshi Shirayanagi [‡] [‡] and Matu–Tarow Noda[†] [§]

[†] Department of Computer Science, Ehime University, Matsuyama 790–77 Japan ,
[‡] NTT Communication Science Laboratories, 2–2 Hikaridai, Seika–cho, Soraku–gun, Kyoto 619–02 Japan

**Abstract**

We give an algorithm to compute the Moore-Penrose generalized inverse of a matrix, which permits the use of limited precision computation in a convergent fashion, based on Greville's algorithm. This does not mean that Greville's algorithm may be directly computed with limited precision computation in a convergent fashion, but rather the algorithm is amenable to the stabilization techniques proposed by Shirayanagi and Sweedler, and we apply these stabilization techniques to the algorithm. As a further practical application, we consider an associative memory model which plays an important role in pattern recognition. Through some examples, we show the effectiveness of the stabilization techniques. Computations were done on a computer algebra system Risa/Asir.

[*]mina@hpc.cs.ehime–u.ac.jp
[†]kai@cs.ehime–u.ac.jp
[‡]shirayan@progn.kecl.ntt.co.jp
[§]noda@cs.ehime–u.ac.jp

# 1 Introduction

An inverse of a matrix – which is not necessarily square or is square nevertheless singular – is applied to solve ill–conditioned problems such as large sized matrix computation. Such an inverse is referred to as a generalized inverse. Generalized inverses have many applications in engineering problems, such as data analysis, electrical networks, character recognitions, and so on. The most frequently used generalized inverse is a Moore–Penrose type inverse. Symbolic computation of the generalized inverse is one of the interesting application areas of Computer Algebra. Already Frawley implemented the Albert method [1] to obtain the Moore–Penrose type generalized inverse in Macsyma [2]. Gregory and Krishnamurthy intended to apply an error free computation to obtain the generalized inverse [4]. Many algorithms to obtain generalized inverses have been proposed other than the Albert method, and methods used in [4]. Noda and others [8, 9] compared several methods from the viewpoint of computational complexity. They concluded Greville's algorithm is one of the best methods for obtaining a symbolic generalized inverse [3].

In this paper, we apply Greville's algorithm to a method of character recognition. In an analysis of an 'associative memory' model proposed by Kohonen [5, 6], exact calculations of the generalized inverse are required. We show how the symbolic computation works well for the above analysis. However, it is well known that the symbolic computation gives an exact result but it wastes a lot of space and time. Thus it is difficult to compute a generalized inverse for large sized matrices symbolically. Here, we consider how to compute the generalized inverse quickly. A numerical method which is based on the singular value decomposition (SVD) gives results but these results are instable. Then we consider a method of stabilizing an algorithm proposed by Shirayanagi and Sweedler [10].

The aim of this paper is to develop a new application area of symbolic generalized inverse computations and further to show an effective method of stabilizing algorithms. Thus in the following sections, we describe briefly the generalized inverse of matrices ( in **2** ), the associative memory model (in **3** ) and the algorithm stabilization technique ( in **4** ). Finally, Greville's algorithm is stabilized and effectively used to obtain the generalized inverse for associative memory models.

## 2 Algebraic Algorithms of Computing Generalized Inverse

The generalized inverse $G$ of a matrix $A$ is defined as follows:

1. $AGA = A$
2. $GAG = G$
3. $(AG)^T = AG$
4. $(GA)^T = GA$

where $A$ and $G$ are $m \times n$ and $n \times m$ matrices with entries in $\mathbf{R}$, respectively. If a matrix $G$ satisfies all four relations, it is called the *Moore-Penrose type* generalized inverse and is denoted as $A^+$. The generalized inverse $A^+$ also satisfies the following relations:

1. $A^+$ is uniquely determined for $A$
2. if $A$ is regular, $A^+ = A^{-1}$
3. if $A = 0$, $A^+ = 0$
4. if $A$ is an $m \times n$ matrix with rank $m$, then $A^+ = A^T(AA^T)^{-1}$

In this paper, we abbreviate a Moore–Penrose type generalized inverse as simply a 'generalized inverse'. The generalized inverse is approximately computed numerically by a SVD (Singular Value Decomposition) algorithm. The SVD algorithm is simply described as follows: Let the SVD of an $m \times n$ matrix $A$ be

$$A = U\Sigma V^T$$

where $U$ and $V$ are $m \times m$, $n \times n$ matrices, respectively, and $\Sigma$ is an $m \times n$ matrix whose diagonal elements are the singular values $\sigma_i$ of $A$ and off diagonal elements are zeros. Then $A^+$ is computed by

$$A^+ = V\Sigma^+U^T$$

where $\Sigma^+ = (\mathrm{diag}(1/\sigma_i) : 0)^T$ .

The SVD algorithm gives results quickly. However, we must eliminate small singular values. Thus in cases where small elements are meaningful, the results are not reliable especially for ill–conditioned matrices. Further, it is impossible to obtain a generalized inverse of a matrix whose elements are symbols.

Besides the SVD, many numeric algorithms for computing generalized inverse have been proposed. Some of them can be transformed into symbolic algorithms, and it is possible to implement them into computer algebra systems. Comparisons of algorithms were done by Noda and others [8, 9] from the viewpoint of number of operations needed. They conclude that Greville's algorithm [3] is one of the best algorithms. In the Appendix, the details are shown. Thus , in this paper, we compute the generalized inverse by Greville's algorithm. For an input matrix $A$, Greville's algorithm is described as follows:

1. Decompose input $n \times m$ matrix $A$ into row vectors $a_i$
$$A = (a_1^T, a_2^T, \cdots, a_n^T)^T$$
2. Let $i \times n$ matrices $A_i$ be
$$A_1 = a_1, \qquad A_i = \begin{cases} A_{i-1} \\ a_i \end{cases}$$
3. For $i = 1, 2, \cdots$, compute $n \times i$ matrices $A_i^+$ as
$$A_i^+ = (A_{i-1}^+ - b_i^T d_i \mid b_i^T)$$
where
$$\begin{aligned} d_i &= a_i A_{i-1}^+ \\ c_i &= a_i - d_i A_{i-1} \\ b_i &= \begin{cases} \frac{c_i}{c_i c_i^T} & (c_i \neq 0) \\ \frac{d_i (A_{i-1}^+)^T}{1 + d_i d_i^T} & (c_i = 0) \end{cases} \\ A_1^+ &= \begin{cases} \frac{a_1^T}{a_1 a_1^T} & (a_1 \neq 0) \\ a_1^T & (a_1 = 0) \end{cases} \end{aligned}$$
4. After $m$ repetitions, $A_m^+$ gives the Moore–Penrose generalized inverse of $A^+$.

Note that it is necessary to decide precisely whether a value is zero or not in the process of obtaining $b_i$ and $A_1^+$.

# 3 Moore–Penrose generalized inverse and associative memory

Here we apply the exact computation of the Moore-Penrose generalized inverse to an engineering problem. The problem is a computer model of human association and is called *associative memory* . The model is used to feature extractions, character recognitions and so on and was first proposed by Kohonen [6]. The simplest case of the associative memory is a linear association and its mathematical expression is given as follows:

- Let $X$ be an input pattern and $Y$ be a resulting pattern by association
- Further let $X$ and $Y$ be written as rectangular matrices

  $$X = (\vec{x}_1, \vec{x}_2, \cdots, \vec{x}_K) \qquad Y = (\vec{y}_1, \vec{y}_2, \cdots, \vec{y}_K)$$

  where $\vec{x}_i, \vec{y}_i \in \mathbf{R}^n$ for $i = 1, \cdots, K$.
- Linear association is written

  $$Y = MX$$

  for a transformation matrix $M$.
- Since $X$ and $Y$ are rectangular, thus $M$ is written by the generalized inverse as

  $$M = YX^+.$$

The above model shows that when a pattern $X$ is given, we can associate another figure $Y$ which is transformed by $M$. We call the association *mutual* association. The model can be used for *auto* association. In this case, the pattern $X$ is incorrect or noisy and the output pattern $Y$ may be expected to be correct or noise–less. Thus we can write the auto association as

$$M = XX^+.$$

The auto association is used to study of pattern recognition, especially character recognition [7].

Since the size of an input matrix is very large, it is divided and stored in several small distributed memories. Suppose that the number of distributed memories is $S$, and the size of each distributed memory is $n \times n$. In the $i$–th associative memory, $M_i$ is described as

$$M_i = X_i X_i^{+}, \qquad X_i = (\vec{x}_1^i, \vec{x}_2^i, \cdots, \vec{x}_{K_i}^i).$$

The input pattern and its association are reconstructed by $S$ block diagonal matrices where the size of each block is $n \times n$. The problem is how to reduce the size of a matrix $nS \times nS$ to a smaller sized matrix. We must construct the associative memory not by $x_i^k$s but by $M_i$s. We suppose that $S$–associative memories, $M_1, M_2, \cdots, M_S$, are given. We define a generation matrix $\Phi$ whose size is $n \times n$ as

$$\Phi = \sum_{i=1}^{S} m_i M_i$$

where

$$\sum_{i=1}^{S} m_i = 1, \quad 0 < m_i < 1, \quad \text{for } i = 1, 2, \cdots, S.$$

Let $\lambda_i$ and $\varphi_i$ be an eigenvalue and an eigenvector of $\Phi$, respectively. Eigenvectors satisfy an orthogonality condition

$$\varphi_i^T \varphi_i = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}, \qquad 0 \leq \lambda_i \leq 1.$$

Further operations *selection* and *join* are considered. Let $I_i$ be information in the $i$–th memory. Operations *selection* and *join* operate for information in all distributed memories. The former selects commonly stored information and the latter represents all information, respectively. They are written as

$$Selection \quad : \quad I_{sel} = \bigcap_{i=1}^{S} I_i,$$

$$Join \quad : \quad I_{join} = \bigcup_{i=1}^{S} I_i.$$

For an effective realization of the associative memory, we must establish relations between the generation matrix $\Phi$ and operations $I_{sel}$ and $I_{join}$. Murakami and others [7] derived the following relations from Therrien's theorems [11].

- If $n$ vectors $\varphi$ are contained in a space spanned by $I_{sel}$, then $\varphi$ are eigenvectors of $\Phi$ corresponding to **unit** eigenvalues.
- If $n$ vectors $\varphi$ are contained in a space spanned by $I_{join}$, then $\varphi$ are eigenvectors of $\Phi$ corresponding to **positive** eigenvalues.

We show an example of the above processes. Let $\mathrm{P}_1$, $\mathrm{P}_2$, $\mathrm{P}_3$ and $\mathrm{P}_4$ be four patterns where a bit having 1 shows black cell and 0 shows white cell. They represent a box(□) with some other bits and resemble each other but have only minor discrepancies and are shown as:

$$\mathrm{P}_1 = \begin{pmatrix} 1&1&1&1\\ 1&0&0&1\\ 1&0&0&1\\ 1&1&1&1 \end{pmatrix}, \mathrm{P}_2 = \begin{pmatrix} 1&1&1&1\\ 1&0&1&1\\ 1&1&0&1\\ 1&1&1&1 \end{pmatrix}, \mathrm{P}_3 = \begin{pmatrix} 1&1&1&1\\ 1&1&0&1\\ 1&0&1&1\\ 1&1&1&1 \end{pmatrix}, \mathrm{P}_4 = \begin{pmatrix} 1&1&1&1\\ 1&0&0&1\\ 1&0&1&1\\ 1&1&1&1 \end{pmatrix}$$

Vectors $\vec{x}_i$ are obtained by up–to–down and left–to–right scanning of patterns $\mathrm{P}_i$.

$$\begin{aligned} \vec{x}_1 &= (1111100110011111)^T, \\ \vec{x}_2 &= (1111101111011111)^T, \\ \vec{x}_3 &= (1111110110111111)^T, \\ \vec{x}_4 &= (1111100110111111)^T. \end{aligned}$$

Here, we consider three distributed memories $M_1$ by patterns $\mathrm{P}_1$ and $\mathrm{P}_2$, $M_2$ by $\mathrm{P}_1$ and $\mathrm{P}_3$ and $M_3$ by $\mathrm{P}_1$ and $\mathrm{P}_4$. Thus, $M_1$ is obtained as follows:

$$X_1 = (\vec{x}_1, \vec{x}_2) = \begin{pmatrix} 1 & 1 \\ \vdots & \vdots \\ 0 & 1 \\ 1 & 0 \\ \vdots & \vdots \\ 1 & 1 \end{pmatrix},$$

$$M_1 = X_1 X_1^+ = \begin{pmatrix} \frac{1}{12} & \frac{1}{12} & \frac{1}{12} & \frac{1}{12} & \frac{1}{12} & 0 & 0 & \frac{1}{12} & \frac{1}{12} & 0 & 0 & \frac{1}{12} & \frac{1}{12} & \frac{1}{12} & \frac{1}{12} & \frac{1}{12} \\ \frac{1}{12} & \frac{1}{12} & \frac{1}{12} & \frac{1}{12} & \frac{1}{12} & 0 & 0 & \frac{1}{12} & \frac{1}{12} & 0 & 0 & \frac{1}{12} & \frac{1}{12} & \frac{1}{12} & \frac{1}{12} & \frac{1}{12} \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \frac{1}{12} & \frac{1}{12} & \frac{1}{12} & \frac{1}{12} & \frac{1}{12} & 0 & 0 & \frac{1}{12} & \frac{1}{12} & 0 & 0 & \frac{1}{12} & \frac{1}{12} & \frac{1}{12} & \frac{1}{12} & \frac{1}{12} \end{pmatrix}$$

$M_2$ and $M_3$ are obtained similarly. The generation matrix $\Phi$ is obtained by putting $m_1 = m_2 = m_3 = 1/3$. Eigenvalues are obtained by solving an eigenvalue equation of $\Phi$. Desired results by operations selection and join are apparently

$$I_{sel} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}, \qquad I_{join} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}.$$

In an associative memory model, eigenvectors corresponding to unit eigenvalue of $\Phi$ give $I_{sel}$ and to positive eigenvalues of $\Phi$ give $I_{join}$. The eigenvalue equation is obtained

$$\lambda^{16} - 2\lambda^{15} + \frac{23}{18}\lambda^{14} - \frac{8}{27}\lambda^{13} + \frac{1}{54}\lambda^{12} = 0.$$

Eigenvalues of $\Phi$ are

$$1, \frac{1}{3}, \frac{2-\sqrt{2}}{6}, \frac{2+\sqrt{2}}{6}, \quad \text{and others are } 0.$$

An eigenvector $\phi_1$ which corresponds to a unit eigenvalue is represented as

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix},$$

which is the same as the desired $I_{sel}$ . On the other hand, eigenvectors $\phi_P$ which correspond to positive eigenvalues are also the same as desired $I_{join}$.

Especially for applications of the associative memory model to character recognition, there remain difficulties of computations. Since many rows in $X_i$ and $M_i$ are the same as the above example, computations become ill–conditioned. In other words, ranks of vectors or matrices are decrease. This fact makes the exact computation of a generalized inverse difficult. Thus it is necessary to obtain generalized inverses exactly and also to compute eigenvalues of matrices with high accuracy.

# 4 How to stabilize algorithms

Shirayanagi and Sweedler proposed a method of stabilizing algorithms [10]. Their motivation was that computations by symbolic algorithms waste a lot of memories by intermediate swell of coefficients. Thus, if the algorithm is combined with a numeric computation carefully, results may be accurate and stable, and furthermore computations may be done quickly. The result of executing a symbolic algorithm is thought of as being obtained by infinite precision numeric computation. Thus, by limited precision computation, the result must be obtained in a convergent fashion. As a numeric computation, a concept of interval arithmetic is introduced. Coefficients are described by a circular interval number, i.e. a pair of mid–point and small deviation. It is called a Bracket Coefficient. The stabilized algorithm is executed by increasing precisions of inputs, and then the result converges to the true output obtained by symbolic computation. For the convergence to be successful, Zero Rewriting is introduced. Zero rewriting is a rule that if a bracket coefficient contains zero, then the bracket coefficient is rewritten to zero.

Here, we stabilize Greville's Method by the following procedures:

1. Variables take values from **Bracket coefficient**s.
2. **Zero Rewriting** is applied to the steps to obtain $b_i$ and $A_1^+$.
3. Repeat the algorithm by increasing digits used for computations.

For sufficiently large digit computations, the stabilized Greville's algorithm gives a result reasonably approximate to $A^+$.

Further, we consider the following operations between bracket coefficients. Let $A = [a, \alpha]$ and $B = [b, \beta]$ be bracket coefficients. Arithmetic operations are defined as

$$\begin{array}{ll} A + B \Rightarrow & [a + b, \alpha + \beta] \\ A - B \Rightarrow & [a - b, \alpha + \beta] \\ A \times B \Rightarrow & [a \cdot b, (|a| \cdot \beta) + (\alpha \cdot |b|) + (\alpha \cdot \beta)] \\ A/B \Rightarrow & [a, \alpha] \times [b, \beta]^{-1} \end{array}$$

where

$$[b, \beta]^{-1} = [a, \alpha] \times [\frac{a}{(a + \alpha)(a - \alpha)}, \frac{|\alpha|}{(a + \alpha)(a - \alpha)}].$$

Operations are the same as the circular interval arithmetic. The zero rewriting operation is written as

$$[a, \alpha] = \begin{cases} 0 & \text{for } a - \alpha \le 0 \le a + \alpha \\ a & \text{otherwise} \end{cases}$$

# 5 Compare symbolic, numeric and stabilized methods

We show advantages of the stabilized method here. First, computation times of computing a generalized inverse for ill–conditioned matrices by several methods are compared. Generalized inverses of a well known ill–conditioned matrix, Hilbert matrix, whose matrix sizes are $20 \times 20$ and $30 \times 30$ are computed by Greville's method in 1) symbolic , 2) numeric and 3) algorithm stabilization computations. Further, for comparisons, they are also computed by 4) numeric SVD method. Hibert matrix is represented as

$$H = (h_{ij}), \quad h_{ij} = \frac{1}{i+j-1}.$$

Computation times are shown in Table.1 in sec.

| matrix size | Hilbert matrix | | | |
|---|---|---|---|---|
| | symbolic | numeric | | stabilization |
| | | Greville | SVD | |
| $20 \times 20$ | 36.1 | 0.44 | 0.12 | 29.73 |
| $30 \times 30$ | 1192 | 4.82 | 0.14 | 309.1 |

While the results obtained by 1) and 3) satisfy definitions and properties of the generalized inverse, numeric results are far from exact values. In the stabilization method, computations are repeated by increasing digits. In this case, from observation, results converge to exact values after three times repeticions, say 100, 300 and 500 digits computations for $20 \times 20$ case. Computation times 3) show times required to converge. Results in Table 1, further, show the following:

1. Computation time for symbolic computation is huge especially for large sized matrices. Computation time depends on the size of matrices and also the digits of numbers in the computation.

2. Both numeric computations give results quickly. The times depend on the size of the matrices. However, as described above, the results by the numeric computation are not reliable especially for ill–conditioned problems.

3. Computation time by the algorithm stabilization technique is faster than symbolic computation. It also depends on the size of matrices.

Next, the convergence property of the Zero Rewriting in the algorithm stabilization method is examined. We consider an ill–conditioned matrix

$$A = \begin{pmatrix} 3 & \frac{211}{3} & \frac{15}{7} \\ 3 & 70.3333 & 2.14286 \end{pmatrix}.$$

Let the bracket coefficient be $[a, \alpha]$ for $\alpha = 1.0 \times 10^{-k}$ . The convergence property is shown by stabilization computations for several $k$ s. An exact generalized inverse obtained by symbolic computation is

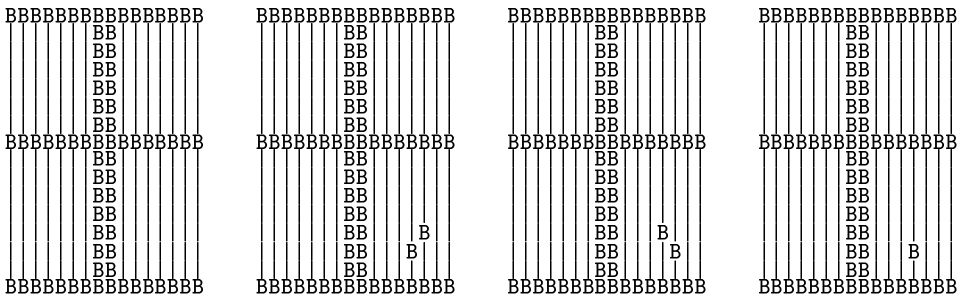
$$A^{+} = \begin{pmatrix} \frac{1323}{4375046} & \frac{1323}{4375046} \\ \frac{31017}{4375046} & \frac{31017}{4375046} \\ \frac{945}{4375046} & \frac{945}{4375046} \end{pmatrix} = \begin{pmatrix} 0.000302397 & 0.000302397 \\ 0.007089525 & 0.007089525 \\ 0.000215998 & 0.000215998 \end{pmatrix}.$$

If we change $k$ from 1 to 5, bracket coefficients converge to exact values. Numerical results of the convergence property are shown as:

$$\overbrace{\begin{pmatrix} 0.000302385 & 0.000302411 \\ 0.007089259 & 0.007089857 \\ 0.000215990 & 0.000216008 \end{pmatrix}}^{k=1} \to \overbrace{\begin{pmatrix} 0.000302397 & 0.000302397 \\ 0.007089523 & 0.007089529 \\ 0.000215998 & 0.000215998 \end{pmatrix}}^{k=2} \to \cdots$$

$$\to \overbrace{\begin{pmatrix} 0.000302397 & 0.000302397 \\ 0.007089525 & 0.007089525 \\ 0.000215998 & 0.000215998 \end{pmatrix}}^{k=4} \to \overbrace{\begin{pmatrix} 0.000302397 & 0.000302397 \\ 0.007089525 & 0.007089525 \\ 0.000215998 & 0.000215998 \end{pmatrix}}^{k=5} \to A^{+}.$$

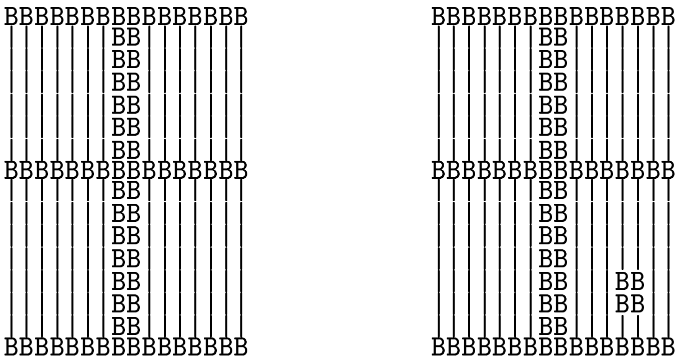
The result by $k = 1$ is erroneous but reslts converge to $A^{+}$ properly for $k = 2, 3, \ldots, 5$. Thus we can say that Greville's method is stabilized by the algorithm stabilization technique. Then we apply the technique to a character recognition problem. We consider a Chinese character which is digitized in $16 \times 16$ dots. As with the problem discussed in **3**, we consider a Chinese character 王 which is digitized in $16 \times 16$ bits and patterns 王 with small noises. $\mathrm{P}_1$ corresponds to a noiseless character and other $\mathrm{P}_2$ , $\mathrm{P}_3$ and $\mathrm{P}_4$ are assumed to be noisy patterns. A black cell is represented by B in figures and takes a value 1 in computations. On the other hand, a white cell corresponds to | in figures and to 0 in computations.



$P_1$ $P_2$ $P_3$ $P_4$

We construct the associative memory by symbolic, numeric and algorithm stabilization methods. By symbolic computation, Greville's method gives exact generalized inverse. The results of $I_{sel}$ and $I_{join}$ are obtained expected, as can be seen in the following pictures:

Expected results of associative memory



$I_{sel}$ $I_{join}$

As explained in **3**, $I_{sel}$ and $I_{join}$ are determined from eigenvectors $\phi_1$ with eigenvalue 1 and $\phi_P$ with eigenvalue positive, respectively. Again, 1 and 0 are represented by B and | in figures. However, in numeric computations, results are changed. If we computes the generalized inverse by numerical Greville's method, results are meaningless because of instability of the method. $I_{join}$ obtained by the SVD computation gives the same as the result obtained by the symbolic method. However, the generalized inverse computed by the SVD has some troubles from floating point error. If we wish to obtain results in high accuracy, we must do high precision computations and eliminate very small singular values. In spite of our expectations, computations often give incorrect results by the error. In our case for $I_{sel}$, four elements which should be zero take small but non zero values. Thus as the result, $I_{sel}$ obtained by the SVD method is the same as exact $I_{join}$. The fact is fatal troble for the associative memory model. A digitization of the Chinese character 王 gives $P_1 = I_{sel}$ , however, a digitized picture $I_{join}$ is the same as the digitized picture of another Chinese character 玉 . They two characters have different meanings and pronounciations. It shows that numerical methods seem to be impossible to apply the associative memory model.

Apparently, in the first step of the stabilization method, say small $k$, results are not satisfactory. By the convergence property of the stabilization method, the stabilized Greville's method gives the same results as the symbolic computation. Thus we can say that the stabilization method can be applied to the associative memory model and gives satisfactory results. Here, the convergence property works well for the method.

# 6 Conclusion

The Moore–Penrose generalized inverse is applied to a kind of character recognition technique, i.e. the associative memory model. Here the exact computation of the generalized inverse is important. Thus we compare several symbolic algorithms for obtaining the Moore-Penrose generalized inverse and mainly discuss Greville's algorithm. Though Greville's algorithm implemented in computer algebra systems gives satisfactory results, computation times and required memories of the algorithm must be reduced. Grevill's algorithm can be computed numerically with limited precision, however, the results are not satisfactory. Thus we apply the stabilization techniques to the algorithm. Through some examples, we obtain satisfactory results both in accuracy and in computation times. Our results are easily applied to problems solving ill–conditioned simultaneous equations.

In most current and future engineering and scientific computations such as the associative memory model, we must treat problems which include large sized matrices. Some of the problems may be ill–conditioned. Thus, techniques just discussed in this paper may be important. We must discuss possiblities of applications of the algorithm stabilization techniques to other algorithms. Then we must establish a computation system suited for the algorithm stabilization method.

Finally, one of the authors, M.T.Noda, would like to thank Fujitsu Lab.

Co. for their financial support. Furthermore, most of the computations were done on a computer algebra system Risa/Asir developed by Fujitsu Lab. Co.

# A Algorithms for computing generalized inverses

Algorithms for computing the generalized inverse are compared from the viewpoint of number of operations by Noda et.al. [8, 9] Here we summarize results briefly.

Symbolic algorithms considered here are Hermite, Albert, Greville, Penrose( or Decell–Leverrier), Glassey and Ben Israel, Wersan algorithms. We show brief introductions of algorithms as:

1. Hermite's Method ( 1976 )
   A kind of extension of Gaussian elimination.

2. Albert's Limit Formula ( 1972 )
   Use the following basic definition of the Moore–Penrose generalized inverse,
   $$A^+ = \lim_{x \to 0} A^T(A^T + x^2 I)^{-1}.$$

3. Greville's Method ( 1960 )
   *see section* **2** *in this paper.*

4. Penrose's ( *or* Decell–Leverrier's ) Method ( 1965 )
   An extension of the Cayley–Hamilton's theorem for a characteristic polynomial of complex $m \times n$ matrix $A$.

5. Method by Ben–Israel, Wersan ( 1963 )
   Let $A^+$ be $A^T(A^T)$. Apply the $LU$ decomposition to $A^T$.

6. Glassey's Method ( 1966 )
   Based on the Gram–Schmidt diagonalization method.

The number of multiplications and additions required to compute the Moore–Penrose generalized inverse $A^+$ for an $m \times n$ matrix $A$ are analyzed. Here, all elements of the matrix are characters or symbols. If the input matrix is regular, its rank is usually defined as $\min(m, n)$. However, especially for the generalized inverse, the rank of the matrix may be less than $\min(m, n)$ in the computation process. We put the rank of matrix as $r \leq \min(m, n)$.

The number of required operations of a method, $\phi$(method) are obtained as follows:

$$\begin{aligned}
\phi(\text{Hermite}) &= m^3 + 3m^2n + 5m^2r - 3mr^2 + \frac{2r^3}{3} \\
\phi(\text{Albert}) &= m^3 + 2m^2n + \text{limit operations} \\
\phi(\text{Greville}) &= 2m^2n - \frac{nr^2}{2} \\
\phi(\text{Penrose}) &= m^3r - m^3 + 2m^2n + \frac{m^2r}{2} \\
\phi(\text{Ben–Israel}) &= 2m^2n + 2m^2r + mr^2 + \frac{4r^3}{3} \\
\phi(\text{Glassey}) &= 5mnr - \frac{mr^2}{2} + nr^2 + r^3
\end{aligned}$$

In the case of square and regular matrices, $m = n = r$, the number of operations of Hermite's, Albert's, Greville's Ben–Israel's and Glassey's methods are $\mathrm{O}(m^3)$ and Penrose's is $\mathrm{O}(m^4)$. All methods are implemented in the computer algebra systems REDUCE and Risa/Asir. Here we use ordinary rational number arithmetic. From most of practical examples, Greville's method gives the fastest computation time. The result is consistent with the above analysis of the number of required operations.

## References

[1] A. Albert : Regression and the Moore–Penrose Pseudoinverse, Academic Press, 1972.

[2] W.J. Frawley, in *Application of Computer Algebra* , ed. R. Pavelle, pp.415–426, 1985.

[3] T. Greville : Some applications of pseudoinverse of a matrix, SIAM Review, 2, pp.15–22, 1960.

[4] R.T. Gregory and E.V. Krishnamurthy : Methods and Applications of Error–Free Computations, Springer–Verlag, 1984.

[5] T. Kohonen : Correlation matrix memories, IEEE Trans. Comput., C–21, pp.353–359, 1972.

[6] T. Kohonen : Associative Memory, Springer–Verlag, 1977.

[7] K. Murakami, H. Takechi and M. Izumida : JIEC, J76–D–II, pp.2607–2614, 1993 (in Japanese).

[8] M.T. Noda, M. Izumida and M. Ochi , Evaluation of exact computation methods of the generalized inverse, Jap. Jour. Inf. Proc., 30, pp.1376–1384 ,1989. (in Japanese)

[9] T. Saito and K. Makino, in RIMS report , 1995 (in Japanese).

[10] K. Shirayanagi and M. Sweedler, A theory of stabilizing algebraic algorithms, MSI Tech. Rep. 95–28, Cornell Univ., 1995.

[11] C.W. Therrien : Eigenvalue properties of projection operators and their application to the subspace method of feature extraction, IEEE Trans.Comput., C–24, pp.944–948, 1975.