Optimization of the TabProcs Package

Stephen V. Gliske

May 23, 2006

Abstract

The TabProcs package is a Maple package written by the author, following the procedures developed with his collaborators, Dr. W.H. Klink and Dr. T. Ton-That. This paper describes how the abstract quantities of the procedure are represented and manipulated in the computer implementation, including some of the optimization techniques.

1 Datatype Representations

1.1 Operators

Let the signatures (M) and $(m)_1, (m)_2, \dots (m)_r$ of U(N) be given. As defined previously, elements of the vector space $V^{(m)_1} \otimes V^{(m)_2} \otimes \dots \otimes V^{(m)_r}$are polynomials over a matrix of variables denoted Z. If each signature $(m)_i$ has p_i non-zero entries and $n := \sum_{i=1}^{r} p_i$, then Z is an n by N matrix. All left diferential operators $L_{i,j}$will satisfy $1 \leq i, j \leq n$ while all right operators $R_{i,j}$will satisfy $1 \leq i, j \leq N$.

Let $\tilde{n} = \max(N, n)$. One can enumerate the possible indices according to

$$(i, j) \to (i-1)\tilde{n} + j. \tag{1}$$

This is equivelant to enumerating the elements of a $\tilde{n} \times \tilde{n}$ matrix in row-major order, and the enumeration ranges from 1 to $\tilde{n}^2$.

For example, consider $\tilde{n} = 3$. Then any operator $L_{i,j}$or $R_{i,j}$is enumeratred as the i, jth element of the following matrix:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}. \tag{2}$$

In this manner, only the enumeration index is needed to store the operator.

Products of operators are listed in backwards order from the usual convention. Whereas when one writes a product of operators, the convention is to act first with the operator farthest to the right, the procedures list the operators such that the first operator in the list (the farthest operator to the left) is applied first. Thus, considering the above same example, the product of operators $L_{1,2}L_{2,2}L_{2,1}$ is representated as $[4, 5, 2]$.

1.2 Minors of Determinants

Once (M) and each $(m)_i$ are specified, one immediately knows the structure of all the products of minors of determinants used in a particular example. The procedures explictly determine the set of what sizes of minors will be needed. And since the dimensions of the matrix Z are given, all sets of indices used must include only the integers $1 \ldots \tilde{n}$. The procedures explictly construct (as bit arrays–discussed later) and enumerate all possible minors of determinants, of the sizes needed and of the indices in the correct range. Each minor is then represented by its index in the enumeration.

For example, consider the 8-dimensional irrep of $SU(3)$ sitting the in the three fold product of the 3-dimensional irrep. Then $(M) = (2,1,0)$ and $(m)_1 = (m)_2 = (m)_3 = (1,0,0)$. In this case Z is a 3×3 matrix. The only minors used in this example will be of size 1 or 2, and the indices will range from 1 to 3. Thus an enumeration is

$$Minors = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 1,2 \\ 1,3 \\ 2,3 \end{bmatrix}. \tag{3}$$

The actual order of the enumeration used is discussed later. Note that these could refer to either row or column indices, i.e. $Minors_4$ equivelantly represents $\Delta^{1,2}$ and $\Delta_{1,2}$. Which indices are represented is kept tract of by other means.

A product of minors can then be specified by a list of indices in the enumeration. Since multiplication is commutative, we choose the list to be in increasing order. However, it is much more convinient to represent a product as a single number rather than a list of numbers. Let Λ be the total number of minors in the enumeration. We can then think of the list as being a number, base Λ, which for storage purposes we choose to represent base 10. Thus if $l = [l_1, l_2 \ldots]$ is a list of minors in a product, $(l_1 \geq l_2 \geq \ldots)$, the product is represented as $\sum_i (l_i - 1)\Lambda^{i-1}$ Continuing the above example, $\Lambda = 6$ and the highest weight vector of $(2,1,0)$ is represented

$$\Delta^1 \Delta^{1,2} \rightarrow [1,4] \rightarrow (1-1)6^1 + (4-1)6^0 = 3. \tag{4}$$

In this manner, the spanning sets Δ^r, $\Delta^{c,i}$, are represented as vectors of integers, taking values in the set $\{0 \ldots \Lambda^{\alpha-1}\}$, where α is the number of products of minors respective to each set Δ^r, $\Delta^{c,i}$.

2 Optimization

2.1 Action of Operators

Although the above discussed representations are designed for efficient storage and sorting, they are not the best representations for the application of operators to minors of determinants. Let $L^{(i)}$ represent the ith operator in the enumeration. Note that there exists k such that the action of $L^{(i)}$ on $Minors_j$ yeilds

$$L^{(i)} Minors_j = \pm Minors_k. \tag{5}$$

Thus to speed the application algorithm, a $\tilde{n}^2 \times \Lambda$ array is generated. Denote this array (AT), for application table. Define the elements such that

$$L^{(i)} Minors_j = \operatorname{sign}\left((AT)_{i,j}\right) Minors_{|(AT)_{i,j}|}. \tag{6}$$

The action of differential operators on the polynomials (expressed as minors of determinants) is then represented by the action of the operators, enumerated $1 \ldots \tilde{n}^2$, acting on the space representing the minors, enumerated $1..\Lambda$. The action is already computed in the table, and thus one the table is complete, all computations can be done by considering just the ordinal positions and the action given by the table, without any further consideration to the actual objects they represent.

To compute the table, however, a different representation is needed. For this, a binary representation is quite efficient. The indices of any one minor can be expressed as a $\tilde{n}$-bit array, with the ith bit set to 1 if the minor includes the row (column) index i. This is equivalent to representing the minor as the integer $\sum_i 2^{(i-1)}$ where the sum is over each i included in the minor.

Let x be the bit array representation of a minor, and consider the acton of $L_{i,j}$on the minor. On must first check that the jth bit is 1 and that the ith bit is 0, otherwise the operator anhiliates the minor. In the process, one can easily construct the sum of the bits between ith and jth bit. If the sum is an odd number, then the operator introduces a phase of -1. The jth bit is then changed to the ith bit by adding a factor of $2^{i-1} - 2^{j-1}$.

To further save redundant computations, one can construct an antisymmetric table such that the i, jth component equals $2^{i-1} - 2^{j-1}$. Then after considering whether $L_{i,j}$anhiliates the minor or indroduces a phase, the minor (bit array) x is changed to the minor whose bit array equals x plus the i, jth component of the antisymmetric table.

Once the table is complete, then the bit array representation is no longer useful, except that the actual order of the minors of determinants in their enumeration is by ascending order of their binary representations.

2.2 Spanning Set of Minors

Linear combinations of products of minors of determinants are aways written as a vector of coefficients multiplied by some spanning set of products of minors of determinants. Thus, whenever operators need to be applied to a linear combination of minors of determinants, the action of the operator on the spanning set of products is computed, yielding a transformation matrix times a (possibly new) spanning set. Examples of this are already provided in the other papers on the subject.

Although the spanning sets can be explicitly generated based on the weight structure of the space, in practise is more efficient just to generate the set as it is created. This method, thought, has the disadvantage that the bases set is not generated in any order that can be used to make searching the set more efficient. Sorting the basis at this point would involve re-arranging the matrix of coefficients. Instead, a reverse-lookup table is generated. For example, denote the reverse lookup table of $\vec{\Delta}^{(r)}$ as $\vec{\Delta}^{(r,rev)}$. Then if $\Delta_i^{(r)} = c$, $\Delta_c^{(r,rev)} = i$.

The reverse lookup table is stored sparsely, such that only the non-zero elements are explictly stored. In this case, the memory cost is doubled. However, there is no need to explictly construct $\Delta^{(r)}$ or to ever sort any of these spanning sets. Also no search routine is needed, as the reverse lookup table gives the place of each product of minors in just the amount of time it takes to lookup the sparse table value.

2.3 The Borel Condition

For larger examples, the most time comsuming portion of mapping $f_{\max}$ into the product space is solving the nullspace problem in the Borel Condition. Simply put, one has a matrix $N = MA$ and one needs to determine a matrix B such that $BN = 0$. Furthermore, N often appears to be overdetermined. That is, the column dimension is larger than the row dimension. However, in all cases of interest the number of linearly independant columns (column rank) is less than to the row dimension–otherwise, there exist only the trivial solution and the multiplicity is zero.

Note, the problem can be reduced finding the basis for the set of vectors $\vec{c}$ such that $N^T\vec{c} = \vec{0}$. The usual method would be to compute the LU-decomposition of the matrix N^T, and the author's methods are based on this algorithm. However, the methods compute the decomposition without explicitly transposing the matrix N, and take into consideration the column rank being (assumed) less than the column dimension. The methods are modified to be fraction-free, such that everything is computed in exact integer arithmatic, rather than using Maple's costly fraction datatype or inexact floating point arithmatic. A modified partial pivot is also used.

Consider first the case where pivoting is not needed. N is decomposed as $N = L'R$, where R is a square, invertible, upper-triangular matrix, and L' has the form

$$L' = \begin{bmatrix} L & 0 \\ N' & 0 \end{bmatrix}, \tag{7}$$

where L is a square, invertible, lower-triangular matrix; N' is a matrix; and the zeros represent zero matrices of the appropriate size. The matrix defined by $B = \begin{bmatrix} N'L^{-1}, & -I \end{bmatrix}$ is then a solution since

$$\begin{aligned} BN &= BL'R \\ &= \begin{bmatrix} N'L^{-1}, & -I \end{bmatrix} \begin{bmatrix} L & 0 \\ N' & 0 \end{bmatrix} R \end{aligned}$$

$$
\begin{aligned}
&= \left[\; N'L^{-1}L - N', \quad 0 \;\right] R \\
&= \left[\; 0, \quad 0 \;\right] R \\
&= 0. \qquad (8)
\end{aligned}
$$

The matrix R is not needed in the solution, and thus never actually constructed.

BLAH about the pivot and fraction free

2.4 Change of Spanning Sets

In the procedures for computing Clebsch-Gordan coefficents, the spanning sets of products of minors of determinants eventually reach the end of their usefulness. At this point, one has mapped the highest weight vector of $V^{(M)}$ both into the products space, and broken the degeneracy with a complete set of generalized Casimir operators, yeilding

$$P^{-1}B\,\overrightarrow{(\pi\mathrm{L})}\,f_{\max} = P^{-1}M\vec{\Delta}^{(r)} \qquad (9)$$

The highest weight vector has also been lowered to the basis element specified by a given Gel'fand-Zetlin tableau, yeilding

$$\vec{h}'\cdot\,\overrightarrow{(\pi\mathrm{R})}\,f_{\max}) = \vec{h}\cdot\vec{\Delta}^{(c)} \qquad (10)$$

The next step is to combine $\vec{\Delta}^{(r)}$ with $\vec{h}\cdot\vec{\Delta}^{(c)}$, and rewrite the result as some new matrix $A_{i,j}$ multiplied with a new spanning set $\overrightarrow{(\pi\mathrm{Z})}$. Each element of $\overrightarrow{(\pi\mathrm{Z})}$ is a product of the complex variables $z_{i,j}$. The matrix $A_{i,j}$ is defined such that $\Delta_i^{(r)}$ combined with all of $\vec{h}\cdot\vec{\Delta}^{(c)}$ equals $\sum_j A_{i,j}(\pi Z)_j$.

For each $\Delta_i^{(r)}$ and $\Delta(c)_k$, the procedure seperatly considers minors of each particular size, computes the weighted combination, and generates a vector of coefficients and a vector of products of variables. The procedures then combine the results from each size minors considered, and multipy by h_k. Adding the results for each value of k yeilds the ith row of the matrix A.

To optimize this procedure, the variables $z_{i,j}$ are represented as the number $(w+1)^{(i-1)N+j}$ where w is equal to the sum of the entries of (M), and (M) is a $U(N)$ irrep. In this case, multiplication is replaced by addition. Thus a generic product of elements is represetned as

$$\prod_{k=1}^{w} z_{i_k,j_k} \quad\rightarrow\quad \sum_{k=1}^{w}(w+1)^{(i_k-1)N+j_k}. \qquad (11)$$

This is equivelant to enumerating the variables in row-major order, and if some $z_{i,j}$ happens to be the kth variable in this ordering, then a factor of $(w+1)^k$ is added in for each occurance of $z_{i,j}$ in the product. Alternately, if x is a representation of a product of variables of the above form, the kth digit of x in base $(w+1)$ is the exponent of the kth variable. The exponents are known to be no larger than w by the weight conditions on the $V^{(M)}$ space, and are also non-negative. Thus one needs $(w+1)$ digits for each of the posible exponants, and thus the exponants of the variables can best be represented in base $(w+1)$.

For each particular size of minor of determinant, the procedure can determine what the determinant looks like in this representation, as a formula of the row indices, and then for each determinant of this size just input the list of rows and columsn into the formula to determine the minor. For example,

$$\begin{aligned} \Delta^i_j = z_{i,j} &\quad\rightarrow\quad (w+1)^{(i-1)N+j} = [1]\left[(w+1)^{(i-1)N+j}\right] \\ \Delta^{i,j}_{k,l} = z_{i,k}z_{j,l} - z_{i,l}z_{j,k} &\quad\rightarrow\quad \begin{bmatrix} 1, & -1 \end{bmatrix} \begin{bmatrix} (w+1)^{(i-1)N+k}(w+1)^{(j-1)N+l} \\ (w+1)^{(i-1)N+l}(w+1)^{(j-1)N+k} \end{bmatrix} \end{aligned} \tag{12}$$

3 Examples

3.1 Canonical Example

For example, consider $(3,2,1) \subseteq (2,1,0) \otimes (2,1,0)$. Minors will have sizes 1, 2, and 3. In this case $\tilde{n} = 4$, and the following shows that $\Lambda = 14$. Thus the enumeration of minors are have the following binary and ordinal represention:

Minor	**BitArray**	**Binary**	**Ordinal**
Δ^1	0001	1	1
Δ^2	0010	2	2
$\Delta^{1,2}$	0011	3	3
Δ^3	0100	4	4
$\Delta^{1,3}$	0101	5	5
$\Delta^{2,3}$	0110	6	6
$\Delta^{1,2,3}$	0111	7	7
Δ^4	1000	8	8
$\Delta^{1,4}$	1001	9	9
$\Delta^{2,4}$	1010	10	10
$\Delta^{1,2,4}$	1011	11	11
$\Delta^{3,4}$	1100	12	12
$\Delta^{1,3,4}$	1101	13	13
$\Delta^{2,3,4}$	1110	14	14.

(13)

In this case, it happens that the binary matches the ordinal representation. The highest weight vector is

$$\begin{aligned} \Delta^1\Delta^{1,2}\Delta^{1,2,3} &\quad\rightarrow\quad [1,3,7] \\ &\quad\rightarrow\quad (1-1)14^2 + (3-1)14 + (7-1) = 34. \end{aligned} \tag{14}$$

The spanning set is represented as

$$\begin{bmatrix} \Delta^1\Delta^{1,3}\Delta^{2,3,4} \\ \Delta^1\Delta^{2,3}\Delta^{1,3,4} \\ \Delta^1\Delta^{3,4}\Delta^{1,2,3} \\ \Delta^2\Delta^{1,3}\Delta^{1,3,4} \\ \Delta^3\Delta^{1,2}\Delta^{1,3,4} \\ \Delta^3\Delta^{1,3}\Delta^{1,2,4} \\ \Delta^3\Delta^{1,4}\Delta^{1,2,3} \\ \Delta^4\Delta^{1,3}\Delta^{1,2,3} \end{bmatrix} \rightarrow \begin{bmatrix} [1,5,14] \\ [1,6,13] \\ [1,12,7] \\ [2,5,13] \\ [4,3,13] \\ [4,5,11] \\ [4,9,7] \\ [8,5,7] \end{bmatrix} \rightarrow \begin{bmatrix} 69 \\ 82 \\ 95 \\ 264 \\ 446 \\ 654 \\ 680 \\ 875 \end{bmatrix} \tag{15}$$