Matrix Representations of su(3) and sl(3,C) Lie algebras via Fortran 90

Richard Shurtleff *

December 18, 2025

Abstract

The Fortran 90 program included in this article calculates eight matrices that form a basis of the sl(3,C) Lie algebra in an irreducible representation of the user's choice. A quick linear transformation yields a basis for the su(3) Lie algebra. The program checks that the generators satisfy the necessary commutation relations and saves the matrix generators to data files.

Keywords: SU(3); SL(3,C); Fortran; Lie algebra; matrix representation

1 Introduction

The special unitary group SU(3) is exemplified by its prototype, the group of 3x3 unitary matrices with unit determinant with elements combined by matrix multiplication. Here, we calculate irreducible matrix representations (irreps) of SU(3) whose elements behave under matrix multiplication the same way as those 3x3 unitary matrices. There is one irrep for each pair of nonnegative integers (p,q).[1, 2, 3, 4]

The group has the algebra su(3). Each element g of the SU(3) Lie group is generated by an element F of the su(3) Lie algebra, $g = \exp iF$, where the 'generator' F can be expressed as a linear combination of a basis of eight generators F_j, j = 1,...,8. We have $F = \sum \theta_j F_j$ for a suitable choice of coefficients with real value θ_j.

*affiliation and mailing address: Department of Applied Mathematics and Sciences, Wentworth Institute of Technology, 550 Huntington Avenue, Boston, MA, USA, ZIP 02115, e-mail addresses: shurtleffr@wit.edu, momentum.matrix@gmail.com

The special unitary group SU(3) has extensive applications in physics, from particle physics[3, 5, 6, 7, 8, 9] and nuclear physics[10, 11] to the ubiquitous harmonic oscillator, specifically the 3D isotropic harmonic oscillator [12].

Recently, formulas became available to calculate matrix bases for irreducible representations (irreps) of the $\mathfrak{su}(3)$ Lie algebra and the closely related $\mathfrak{sl}(3,\mathbb{C})$ Lie algebra.[13]

The Fortran program in Section 5 calculates a set of matrices that form a basis of generators for the (p,q) irrep of the su(3) algebra. The program first calculates matrices for the basis $TYUV$ of the $\mathfrak{sl}(3,\mathbb{C})$ Lie algebra, which is the complexification of su(3). The components of the $TYUV$ matrices are real-valued, while the components of the F_j basis are complex-valued. The terminology has twisted irony since the real-valued $TYUV$ basis is a complexification of the F_j basis.[1]

Sections 2, 3, 4 provide background. Consult [13] for more details. In Section 2, we develop the commutation relations of the $\mathfrak{sl}(3,\mathbb{C})$ Lie algebra for the basis $TYUV$. These are the quadratic equations that must be satisfied by the $TYUV$ matrices.

The matrix generators T^3 and Y in the basis $TYUV$ are special because they are represented as diagonal matrices. Their diagonal components α and y are simultaneous eigenvalues of the eigenvectors $|\alpha, y\rangle$.

Section 3 describes a function n that produces a sequence of integers. The scheme defines two integer parameters a, b in terms of the irrep identifiers p, q and the eigenvalues α and y. The sequence function $n(a, b, \alpha) = 1, ..., d$ assigns a place to each of the d eigenvectors $|\alpha, y\rangle$.

The function $n(a, b, \alpha)$ is related to (i) the row and (ii) the column indices, and (iii) the value of a matrix component. Thus, the component M^{rc} of a matrix $TYUV$ has a row index r determined by the parameters a_r, b_r, α_r and a column index determined by a_c, b_c, α_c.

Then the value of the component M^{rc} is a function of the six parameters a_r, b_r, α_r, a_c, b_c, α_c. In reality, the six parameters are constrained, so only three are free for any component M^{rc}.

Section 4 displays the recently obtained formulas for the components M^{rc}, where M represents one of the eight matrices in the $TYUV$ basis. The basis F_j for the associated (p,q) irrep of $\mathfrak{su}(3)$ is implicitly given as linear combinations of the $TYUV$ matrices.

A listing of the Fortran program is copied in Section 5. The program calculates the matrices for the basis $TYUV$ of the (p,q) irrep of the $\mathfrak{sl}(3,\mathbb{C})$ Lie algebra and the matrices for the basis F_j of the (p,q) irrep of the $\mathfrak{su}(3)$ Lie algebra. The values of p and q are set by the user and the matrices are saved in data files. The program checks that the matrices satisfy the appropriate commutation relations before they are saved.

A currently viable link to the program is provided in the references.[17, 18] Or one can cut the program from Section 5 and paste it into a Fortran 90 compiler.

2 Lie algebras

One basis F_j of the fundamental rep of the $\mathfrak{su}(3)$ Lie algebra consists of the following eight matrices,[3, 7, 14]

$$F_1 = \frac{1}{2}\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad F_2 = \frac{1}{2}\begin{pmatrix} 0 & -i & 0 \\ i & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad F_3 = \frac{1}{2}\begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad , \qquad (1)$$

$$F_4 = \frac{1}{2}\begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} \quad F_5 = \frac{1}{2}\begin{pmatrix} 0 & 0 & -i \\ 0 & 0 & 0 \\ i & 0 & 0 \end{pmatrix} \quad F_6 = \frac{1}{2}\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \quad ,$$

$$F_7 = \frac{1}{2}\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -i \\ 0 & i & 0 \end{pmatrix} \quad F_8 = \frac{1}{2\sqrt{3}}\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -2 \end{pmatrix} \quad .$$

By inspection, the F_js are hermitian and traceless.

A group element g of SU(3) can be expressed as the matrix exponent of an element of the $\mathfrak{su}(3)$ Lie algebra,[1, 4]

$$g = e^{i\sum_j^8 \theta_j F_j} \quad , \qquad (2)$$

The eight coefficients θ_j are real valued numbers.

We choose the basis $TYUV$ of $\mathfrak{sl}(3,\mathbb{C})$ to be [3, 4]

$$T^3 = F_3 \; ; \; Y = \frac{2}{\sqrt{3}}F_8 \; ; \; T^{\pm} = F_1 \pm iF_2 \; ; \; U^{\pm} = F_6 \pm iF_7 \; ; \; V^{\pm} = F_4 \pm iF_5 \quad . \qquad (3)$$

By (1) to (3), one finds

$$T^3 = \frac{1}{2}\begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad , \quad Y = \frac{1}{3}\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -2 \end{pmatrix} \; ; \; T^+ = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \qquad (4)$$

$$T^- = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad U^+ = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} \quad , U^- = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \quad ,$$

$$V^+ = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad V^- = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} \quad .$$

The transformation is invertible, so one can determine the basis of F_js from the $TYUV$ basis.

There are $28 = 8 * 7/2$ comutation relations (CRs) determined by the eight $TYUV$ generators in the basis of the $\mathfrak{sl}(3,\mathbb{C})$ Lie algebra. The CRs can be grouped into CRs that are not dependent on U, V matrices, CRs that are linear in the U, V matrices, and CRs that are quadratic in the U, V matrices.

- CRs not dependent on U, V matrices:

$$[T^+, T^-] = 2\,T^3 \quad ; \quad [T^3, T^\pm] = \pm\, T^\pm \quad ; \tag{5}$$

$$[Y, T^\pm] = 0 \quad ; \quad [Y, T^3] = 0 \quad . \tag{6}$$

- CRs linear in U, V matrices:

$$[T^3, U^\pm] = \mp\,\frac{1}{2} U^\pm \; ; \; [T^3, V^\pm] = \pm\,\frac{1}{2} V^\pm \; ; \; [Y, U^\pm] = \pm\, U^\pm \; ; \; [Y, V^\pm] = \pm\, V^\pm \tag{7}$$

$$[T^\pm, U^\mp] = [T^\pm, V^\pm] = 0 \quad ; \quad [T^\pm, U^\pm] = \pm\, V^\pm \quad ; \quad [T^\pm, V^\mp] = \mp\, U^\mp \quad . \tag{8}$$

- CRs quadratic in U, V matrices:

$$[U^+, U^-] = \frac{3}{2}\, Y - T^3 \quad ; \quad [V^+, V^-] = \frac{3}{2}\, Y + T^3 \quad ; \tag{9}$$

$$[U^\pm, V^\mp] = \pm\, T^\mp \quad ; \quad [U^\pm, V^\pm] = 0 \quad , \tag{10}$$

  where the commutator $[A, B]$ of two matrices is the difference of their dot products, $[A, B] \equiv AB - BA$.

By (5) and (6), the four generators T^3, $T^\pm$ and Y form a basis of the subalgebra $\mathfrak{u}(2)$. Hence, each of the four generators can be completely reduced to a direct sum of irreducible representations (irreps) $\mathfrak{u}(2)$. Thus, we give the matrices for T^3, $T^\pm$ and Y a block-diagonal structure, with an irrep of the subalgebra $\mathfrak{u}(2)$ in each block.[4]

The sequence of the $\mathfrak{u}(2)$ irreps for the blocks along the diagonals of T^3, $T^\pm$, and Y is arbitrary. In the next section, order is imposed, and the blocks and their contents are assigned to specific locations in the matrices.

3 The Eigenvector Sequence

The system of $\mathfrak{u}(2)$ subalgebras is a multiplet. The subalgebra $\mathfrak{u}(2)$ has its own subalgebra, $\mathfrak{su}(2)$, with the matrices T forming the basis for $\mathfrak{su}(2)$. Familiarity with $\mathfrak{su}(2)$ is assumed. We show how the properties of multiplets lead to the adopted sequence of eigenvectors and their eigenvalues.

In Figure 1, we represent each subalgebra multiplet $\mathfrak{u}(2)$ by plotting its eigenvalues of the eigenvectors of T^3 and Y. The multiplicity of the eigenvector with eigenvalues α, y marks the point (α, y^0). We shift y by $2(p-q)/3$, so $y^0 = y - 2(p-q)/3$ is plotted on the vertical axis. The eigenvalue y is not always an integer, while y^0 is an integer for all choices of (p, q). The boundary of the six-sided pattern has $p = 5$ spaces on three sides and $q = 3$ spaces between points on the other three sides.

The graph displays the structure of the subalgebra blocks. For example, the row $y^0 = 3$ has 18 eigenvectors that can be collected into three irreducible representations (irreps) $\mathfrak{u}(2)$. The $(t, y^0) = (5/2, 3)$ $\mathfrak{u}(2)$ irrep has one eigenvector for each pair of eigenvalues $(\alpha, y^0) = (5/2, 3)$, $(3/2, 3)$, $(1/2, 3)$, $(-1/2, 3)$, $(-3/2, 3)$, $(-5/2, 3)$. Thus, the row $y^0 = 3$ has 3 irreps with $(t, y^0) = (7/2, 3)$, $(5/2, 3)$ and $(3/2, 3)$ each having 8, 6 and 4 eigenvectors.

For any (p, q) the eigenvalues α, y at the point labeled "Max α" in Figure 1 are known. One has

$$\alpha = (p+q)/2 \quad ; \quad y^0 = p - q \quad ; \quad y = (p-q)/3 \quad (\text{at Max } \alpha) \tag{11}$$

Note that the eigenvalue y is not an integer for $p, q = 5, 3$ and for many other values of (p, q).

For each (t, y^0) irrep, we introduce the integers a and b,

$$a \equiv \left(2t - y^0\right)/2 \quad ; \quad b \equiv \left(2t + y^0\right)/2 \quad . \tag{12}$$

Solving the equations for t and y^0 gives

$$t = (a+b)/2 \quad ; \quad y^0 = b - a \quad . \tag{13}$$

We can identify a $\mathfrak{u}(2)$ subalgebra irrep as either the (t, y^0) $\mathfrak{u}(2)$ irrep or as the (a, b) $\mathfrak{u}(2)$ irrep.

When plotted in Figure 2, the points (a, b) for the multiplet in Figure 1, form a welcome image, a rectangle nicely aligned with the axes a and b. For each point (a, b), there is exactly one (a, b) $\mathfrak{u}(2)$ irrep. The number of (a, b) $\mathfrak{u}(2)$ irreps is $(p+1)(q+1)$.

Let us order the subalgebra irreps row-by-row. In Figure 2, the points (a, b) are each marked by their place in the sequence of irreps. The first irrep has $(a, b) = (0, 0)$ and the last, 24^{th}, has $(a, b) = (q, p)$.

For any (p, q), the (a, b) $\mathfrak{u}(2)$ irrep is the kth irrep, with

$$k = 1 + a + b\,(q+1) \quad . \tag{14}$$

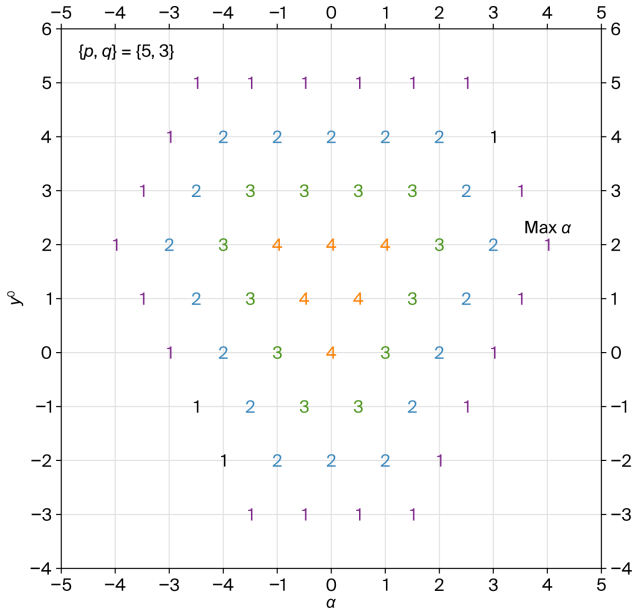

Figure 1: *The* $(p, q) = (5, 3)$ *multiplet.* The values α and $y^0 = y + 2(p - q)/3 = y + 4/3$ for the simultaneous eigenvectors $|\alpha, y^0\rangle$ of T^3 and Y make a six-sided figure whose sides have p or q spaces. The point (α, y^0) is marked with the number of eigenvectors, the 'multiplicity,' that share the values α and y^0. The multiplicity is one on the rim and increases inward on smaller six-sided figures until reaching the central triangle with multiplicity 4. The point on the rim with the maximum T^3 has values $(\alpha, y^0) = (4, 2)$ and is labeled 'Max α'.

The index k runs through successive whole numbers from 1 to $(q + 1)(p + 1)$. We call k the 'single index' for the (a, b) $\mathfrak{u}(2)$ irrep. The pair of integers (a, b) is called the 'double index' of the irrep in the $\mathfrak{u}(2)$ irrep list.

By (13), each (a, b) $\mathfrak{u}(2)$ irrep has spin $t = (a + b)/2$ and it consists of $2t + 1 = a + b + 1$ eigenvectors $|\alpha, y^0\rangle$. These eigenvectors are placed in descending order of the spin component α, largest first, $\alpha = t, ..., -t$. Which makes $|\alpha, y^0\rangle$ the m^{th} eigenvector in the irrep (a, b)-$\mathfrak{u}(2)$, where

$$m = t - \alpha + 1 = (a + b)/2 - \alpha + 1 \quad . \tag{15}$$

The place number $m = 1, ..., a + b + 1$ and depends on a, b, α.

Combine the numbers of eigenvectors in the complete rows of the rectangle below (a, b) and the number in the partial row b to the left of (a, b). Then consider that the eigenvector $|\alpha, y^0\rangle$ is the m^{th} eigenvector in the (a, b) $\mathfrak{u}(2)$ irrep, by (15). It follows that the assigned

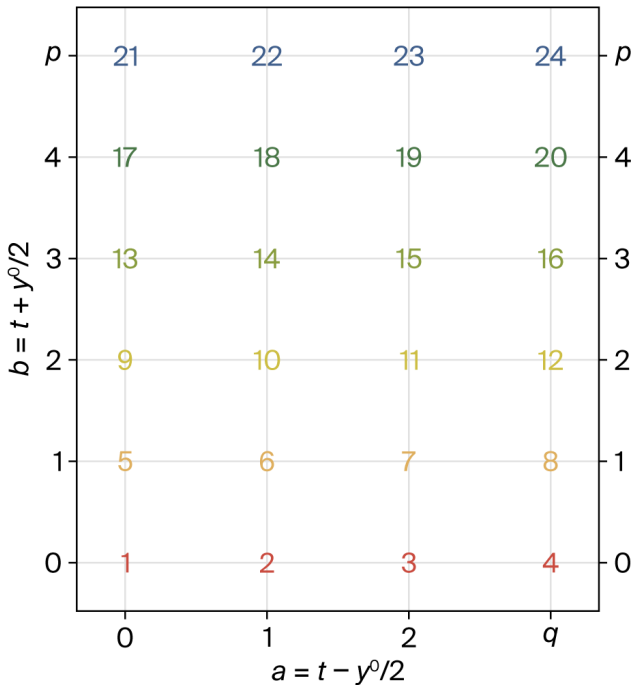

Figure 2: *The sequence of* $\mathfrak{u}(2)$ *irreps in the direct sum of* $\mathfrak{u}(2)$ *irreps making the* T*-matrices.* For the $(p, q) = (5, 3)$ multiplet in Figure 1, simultaneous eigenvectors $|\alpha, y^0\rangle$ are collected in $(t, y^0/2)$-$\mathfrak{u}(2)$ irreps. By (12), transforming $(t, y^0/2)$ to (a, b), renames the $\mathfrak{u}(2)$ irreps as (a, b)-$\mathfrak{u}(2)$ irreps. Each of the $(q + 1)(p + 1)$ points of the rectangle represents exactly one (a, b)-$\mathfrak{u}(2)$ irrep. On the plot, each point (a, b) is marked by its place in the sequence of $\mathfrak{sl}(3, \mathbb{C})$ irreps.

place n for the eigenvector $|\alpha, y^0\rangle$ in the sequence of eigenvectors is given by [13]

$$n(a, b, \alpha) = \frac{1}{2}\left[1 + (a + b + 1)^2 + qb\,(q + b + 2) - 2\alpha\right] \tag{16}$$

where $a = 0, ..., q$, $b = 0, ..., p$, and $\alpha = (a + b)/2, (a + b)/2 - 1, ..., -(a + b)/2$.

The first eigenvector has $a = b = \alpha = 0$ and (16) gives $n(0, 0, 0) = 1$. The last eigenvector has $a = q$, $b = p$, and $\alpha = -(p + q)/2$. Its place $n = n(q, p, -(p + q)/2)$ is the number of eigenvectors d, which is the dimension of the matrix generators. One can show that $n(q, p, -(p + q)/2)$ has the value d,

$$d = \frac{1}{2}(p + 1)(q + 1)(p + q + 2) \quad , \tag{17}$$

which is the well-known value for the dimension of the irrep,[1, 3, 7, 13]

The sequence of eigenvectors and their eigenvalues is determined. The eigenvector $|\alpha, b-a\rangle$ in the (a,b)-$\mathfrak{u}(2)$ irrep is the $n(a,b,\alpha)^{\text{th}}$ in the sequence of d eigenvectors. This is a one-to-one correspondence of the integers n, where $1 \leq n \leq d$, with the d eigenvectors in the collection of simultaneous eigenvectors $|\alpha, y\rangle$ of T^3 and Y.

Consider a component M^{rc} of one of the basis $TYUV$ matrices that we are constructing. Since the indices r and c are a pair of integers in the range 1,...,d, we must have

$$r = n(a_r, b_r, \alpha) \quad ; \quad c = n(a_c, b_c, \beta) \quad , \tag{18}$$

for some values of the six parameters. The values of the six parameters are restricted, $0 \leq a_r, a_c \leq q$, $0 \leq b_r, b_c \leq p$ and the spin indices have ranges $-t_r \leq \alpha \leq t_r$, $-t_c \leq \beta \leq t_c$, with $\mathfrak{u}(2)$-irrep spins $t_r = (a_r + b_r)/2$ and $t_c = (a_c + b_c)/2$. In the following section, formulas for the various components M^{rc} are presented as functions of the two sets of a, b, α parameters.

4 Matrix generator formulas

This section contains the formulas for the nonzero components of the basis generators' matrices for the Lie algebra $\mathfrak{sl}(3,\mathbb{C})$. The formulas for the generators T^3 and $T^{\pm}$ can be found in the references [1, 2, 13, 15, 16]. Those for Y, $U^{\pm}$ and $V^{\pm}$ are found in Ref. [13].

T^3. For the diagonal matrix T^3, we have

$$^3T^{rc} = \alpha \quad ; \quad r = n(a,b,\alpha) \quad ; \quad c = n(a,b,\alpha) \tag{19}$$

where $a = 0, ..., q$, $b = 0, 1, ..., p$, $t = (a+b)/2$, $\alpha = t, ..., -t$.

Y. By (13), we have, for the other diagonal matrix,

$$Y^{rc} = b - a - 2\,(p-q)\,/3 \quad ; \quad r = n(a,b,\alpha) \quad ; \quad c = n(a,b,\alpha) \tag{20}$$

where $a = 0, ..., q$, $b = 0, 1, ..., p$, $t = (a+b)/2$, $\alpha = t, ..., -t$.

T^+. The components of T^+ are given by

$$^+T^{rc} = [(t+\alpha)(1+t-\alpha)]^{1/2} \quad ; \quad r = n(a,b,\alpha) \quad ; \quad c = n(a,b,\alpha-1) \tag{21}$$

where $a = 0, ..., q$, $b = 0, 1, ..., p$, $t = (a+b)/2$, $\alpha = t, ..., -t+1$.

T^-. The components of T^- are given by

$$^-T^{rc} = [(t-\alpha)(1+t+\alpha)]^{1/2} \quad ; \quad r = n(a,b,\alpha) \quad ; \quad c = n(a,b,\alpha+1) \tag{22}$$

where $a = 0, ..., q$, $b = 0, 1, ..., p$, $t = (a+b)/2$, $\alpha = t-1, ..., -t$.

We write the components of the matrices for the generators U, V in terms of two functions g and h, which are defined as

$$g(a,b) \equiv [a(p+a+1)(q-a+1)] \,/\, [(a+b)(a+b+1)] \tag{23}$$

$$h(a,b) \equiv [b\,(p-b+1)\,(q+b+1) \,/\, ((a+b)\,(a+b+1))] \quad .$$

Note that g and h share the same denominator, $(a+b)(a+b+1)$ and the numerator of g depends on a, not b. In contrast, the numerator of h is a function of b, not a. The numerator functions of g and h are very similar.

The four matrix generators U^+, U^-, V^+, V^-, each have upper components above the diagonal with $i < j$ and lower components with $i > j$. There are two formulas, upper and lower, for the components of each of these generators.

U^+, upper components. We get

$${}^+U^{rc} = [g(a,b)\,(t+\beta)]^{1/2} \quad ; \quad r = n(a-1,b,\beta-1/2) \quad ; \quad c = n(a,b,\beta) \tag{24}$$

where $a = 1, ..., q$, $b = 0, 1, ..., p$, $t = (a+b)/2$, $\beta = t, ..., -t+1$.

U^+, lower components. We find

$${}^+U^{rc} = [h(a,b)\,(t-\alpha)]^{1/2} \quad ; \quad r = n(a,b,\alpha) \quad ; \quad c = n(a,b-1,\alpha+1/2) \tag{25}$$

where $a = 0, ..., q$, $b = 1, ..., p$, $t = (a+b)/2$, $\alpha = t-1, ..., -t$.

U^-, upper components. One has

$${}^-U^{rc} = [h(a,b)\,(t-\beta)]^{1/2} \quad ; \quad r = n(a,b-1,\beta+1/2) \quad ; \quad c = n(a,b,\beta) \tag{26}$$

where $a = 0, ..., q$, $b = 1, ..., p$, $t = (a+b)/2$, $\beta = t-1, ..., -t$.

U^-, lower components. One gets the following

$${}^-U^{rc} = [g(a,b)\,(t+\alpha)]^{1/2} \quad ; \quad r = n(a,b,\alpha) \quad ; \quad c = n(a-1,b,\alpha-1/2) \tag{27}$$

where $a = 1, ..., q$, $b = 0, 1, ..., p$, $t = (a+b)/2$, $\alpha = t, ..., -t+1$.

V^+, upper components. We find

$${}^+V^{rc} = -\,[g(a,b)\,(t-\beta)]^{1/2} \quad ; \quad r = n(a-1,b,\beta+1/2) \quad ; \quad c = n(a,b,\beta) \tag{28}$$

where $a = 1, ..., q$, $b = 0, 1, ..., p$, $t = (a+b)/2$, $\beta = t-1, ..., -t$.

V^+, lower components. One gets the following

$${}^+V^{rc} = [h(a,b)\,(t+\alpha)]^{1/2} \quad ; \quad r = n(a,b,\alpha) \quad ; \quad c = n(a,b-1,\alpha-1/2) \tag{29}$$

where $a = 0, ..., q$, $b = 1, ..., p$, $t = (a+b)/2$, $\alpha = t, ..., -t+1$.

V^-, upper components. The formula is

$$ {}^-V^{rc} = \left[h(a,b)\,(t+\beta)\right]^{1/2} \quad ; \quad r = n(a, b-1, \beta - 1/2) \quad ; \quad c = n(a,b,\beta) \tag{30} $$

where $a = 0, ..., q$, $b = 1, ..., p$, $t = (a+b)/2$, $\beta = t, ..., -t+1$.

V^-, lower components. One gets

$$ {}^-V^{rc} = -\left[g(a,b)\,(t-\alpha)\right]^{1/2} \quad ; \quad r = n(a,b,\alpha) \quad ; \quad c = n(a-1, b, \alpha + 1/2) \tag{31} $$

where $a = 1, ..., q$, $b = 0, 1, ..., p$, $t = (a+b)/2$, $\alpha = t-1, ..., -t$.

A basis F_j for the (p,q) irreducible representation (irrep) of $\mathfrak{su}(3)$ follows by inverting the transformation (3).

$$ F_1 = (T^+ + T^-)/2\ ;\ F_2 = -i(T^+ - T^-)/2\ ;\ F_3 = T^3\ ;\ F_4 = (V^+ + V^-)/2\ ; $$
$$ F_5 = -i(V^+ - V^-)/2\ ;\ F_6 = (U^+ + U^-)/2\ ;\ F_7 = -i(U^+ - U^-)/2\ ;\ F_8 = \sqrt{3}Y/2\ . \tag{32} $$

Because they are Hermitian and traceless, these F_j matrices generate unitary matrices of dimension d with a determinant equal to one.

The derivation of the formulas for the basis matrices $TYUV$ appears separately. That omnibus article, Ref. [13], has more detailed discussions and numerical examples explaining the process.

5 Computer Program

The following Fortran-90 program calculates a $TYUV$ basis of eight matrix generators for the (p,q) $\mathfrak{sl}(3,\mathbb{C})$ irrep. The program also calculates the eight matrix basis F_j for the (p,q) $\mathfrak{su}(3)$ irrep. A ready-to-run copy can be accessed by following the link in [17, 18] Users should refer to Section 1 of the program for technical details and directions.

```
!
! This Fortran 90 program calculates the TYUV basis of matrix generators for
! an irreducible representation of the Lie algebra of the SL(3,C) Lie
! group, [Note: SL(3,C) is called SU(3) for some purposes.]
!
! This Fortran program also calculates the Fj basis for su(3). There are
! checks for both bases against the respective commutation relations (CR).
! The two Casimir operators are also calculated, the quadratic Casimir C1
! and the cubic Casimir C2.
```

```
!
! Read Section 1.
!
!------------ TABLE OF CONTENTS  ----------------------
!---0. Preamble
!---1.  Readme
!---2.  Program Start, Interface Blocks
!---3.  Type declarations
!---4.  Set (p,q), MaxErrLimit
!---5.  Preliminaries
!---6.  Make T3
!---7.  Make Y
!---8.  Make Tp
!---9.  Make Tm
!--10.  Make Up
!--11.  Make Um
!--12.  Make Vp
!--13.  Make Vm
!--14.  Check 28 commutation relations and the Casimir equation
!--15. Save the TYUV matrices to a file
!--16. The Fj basis for su(3)
!--17. Save the Fj matrices to a file
!--18. Quadratic Casimir operator
!--19. Cubic Casimir operator, end program
!--20. External functions, end-of-file
!
!----------------------------0. Preamble-----------------------------------
! This program and similar may appear in my articles, an Appendix or Link.
!
! One such article: Formulas for SU(3) Matrix Generators,
! by Richard Shurtleff, Wentworth Institute of Technology, Boston,
! MA, USA - retired
! email: shurtleffr(at)wit.edu, momentummatrix(at)yahoo.com
!
! A basis for sl(3,C):
! T3, Y, Tp, Tm, Up, Um, Vp, Vm,
! where 'p' and 'm' stand for + plus and - minus.
!
! The TYUV matrices are traceless and have real components.
!
```

```
! A basis for su(3):
!
! F1 = (Tp+Tm)/2 ,  F2 = -i(Tp-Tm)/2 , F3 = T3 , F4 = (Vp+Vm)/2 ,
! F5 = -i(Vp-Vm)/2 , F6 = (Up+Um)/2 , F7 = -i(Up-Um)/2 , F8 = \sqrt{3}Y/2
!
! The Fj matrices are hermitian and traceless.
!
!
!   Acknowledgment
! I would like to express my appreciation for Patrick Koh's debugging of a
! previous program which guided the writing of this program.
!
!   Copyright: CC BY-SA. Public use and modification of this code are
! allowed provided that the preprint[1] or any subsequent published
! version is cited.
!
!   References
!
!   [1] R. Shurtleff, "Formulas for SU(3) Matrix Generators",
!       viXra:2409.0060, 2025, and references therein.
!
!   [2] This program can be downloaded by following the link:

!   https://www.dropbox.com/scl/fi/0a1j2fh4mw6dsbiipytc6/SL3C_SU3pqSHO
!   RT.f90?rlkey=s9ifpokx02blvsh67uhpuhb3i&dl=0
!
!   Be sure to make the word "SHORT" appear in the url because the link
!   is on two lines. Something like "!___" needs to be removed
!   to make a valid url. [Although, I tried the url keeping "!   " and
!   it worked.]
!
!-------------------------1. Readme-----------------------------------
!
! This program calculates the 8 TYUV matrices that form a set of basis
! generators for the (p,q) irrep of the sl(3,C) Lie algebra.
! The TYUV basis for sl(3,C) is then transformed to an Fj basis for su(3).
!
! This Fortran 90 program ran successfully on a Windows 11 computer with a
!  GNU fortran compiler Code::Blocks 20.03, Created: 2010/25/05 11:52
!  Updated: 2010/25/05 11:52, Author: HighTec EDV-Systeme GmbH,
```

```
!  Copyright 2010 HighTec EDV-Systeme GmbH, available at
!  https://gcc.gnu.org/fortran/ and http://www.codeblocks.org
!
! INSTRUCTIONS FOR USING THE PROGRAM:
! 1. Input your value of p and q in Sec. 4
! 2. Set the tolerance for error, MaxErrLimit, if you wish.
!    I set the MaxErrLimit to 1.D-10.
!
! OUTPUT to the standard screen:
! Some meta-data is sent to the standard output, including the integers
! p and q that identify the irrep, the dimension of the irrep's
! matrices, the error tolerance, the maximum error found in 29 equations
! that the generators are required to solve and much more.
!
! OUTPUT to an sl(3,C) DATA File named "'p#q#sl3Cgen.dat'" where #s are
! the values of p,q. The DATA File's records are
! Record 1: the integers p, q, and the max error in the 28 commutation
! relations of the sl(3,C) Lie algebra plus the Casimir expression.
! The Format of Record 1: FORMAT(2I3,1ES16.7).
!
! Records 2,3,...: the components of the eight matrices, a total
! of 8n**2 real numbers, where n = dimREP, the matrix dimension.
! Each of the 8 TYUV matrices X has n**2 components arranged by rows, i.e.
!   X(1,1), X(1,2),X(1,3),...,X(1,n),X(2,1),X(2,2),X(2,3),...,X(n,n).
! The sequence of TYUV matrices is X = T3, Y, Tp, Tm, Up, Um, Vp, Vm.
! The Format of Records 2,3,...: FORMAT(5F16.12).
! Thus the matrix components appear 5 real numbers per line until the
! last line.
!
! Sample records from (p,q) = (3,1) irrep file "p3q1SU3GenII.dat"
! Record #1:
! 3  1   3.5527137E-15
! Record #12:
! -0.500000000000  0.000000000000  0.000000000000 0.000000000000 0.0000000
! Record #541:
! 0.000000000000  1.118033988750  0.000000000000  0.000000000000 0.0000000
! Record #923
! 0.000000000000  0.000000000000  0.000000000000
!
! Given n = dimREP = 24 for (p,q) = (3,1), the four Records shown tell
```

```
! us that
! Record 1: p = 3, q = 1, MaxErr = 3.5527137E-15
! Record 12: 5*(11-1)+1 = 51 = 2*n + 3 implies T3(3,3) = -0.5
! Record 541: 5*(540-1)+2 = 4*24**2 + 16*24 + 9 implies that
! Up(17,9) = 1.11803399
! Record 923: 5*(922-1) + 3 = 4608 = 8*24**2 = 8*n**2,
! which is the expected total # of components.
!
!
! OUTPUT to an su(3)) DATA File named "p#q#su3gen.dat" where #s are
! the values of p,q. The DATA File's records are
! Record 1: the integers p, q, and the max error in the 28 commutation
! relations of the su(3) Lie algebra.
! The Format of Record 1: FORMAT(2I3,1ES16.7).
!
! Records 2,3,...: the components of the eight matrices, a total
! of 8n**2 complex numbers, where n = dimREP, the matrix dimension.
! Each of the 8 Fj matrices X has n**2 components arranged by rows, i.e.
!   X(1,1), X(1,2),X(1,3),...,X(1,n),X(2,1),X(2,2),X(2,3),...,X(n,n).
! The sequence of Fj matrices is j = 1,...,8.
! The Format of Records 2,3,...: FORMAT((2F16.12,2F16.12)).
! Thus the matrix components appear 2 complex numbers per line.
!
PROGRAM SL3CSU3Generators
!
    IMPLICIT NONE
!---------------------2. Program Start, Interface Blocks -----------------
!
INTERFACE
    FUNCTION nSEQUENCE(p,q,a,b,alpha) RESULT  (w)    !
    IMPLICIT NONE        ! The nth place in the sequence belongs to
        INTEGER :: p,q     ! the T3 eigenvector with eigenvalue alpha
        INTEGER :: a,b,m    ! in the (a,b)-SU2 irrep
        REAL(8) :: alpha,w !the T3 eigenvector has the mth place in the
!                           ! irrep, where m = (a+b)/2 -alpha + 1
    END FUNCTION
END INTERFACE
INTERFACE
    FUNCTION g(p,q,a,b) RESULT  (w)    !
    IMPLICIT NONE        ! Four of the U,V formulas have
```

```
        INTEGER :: p,q     ! the factor SQRT(g), where
        INTEGER :: a,b    !g = a*(p+a+1)*(q-a+1)/[(a+b)(a+b+1)]
        REAL(8) :: w      ! w = g
    END FUNCTION
END INTERFACE
INTERFACE
    FUNCTION h(p,q,a,b) RESULT  (w)    !
    IMPLICIT NONE        ! Four of the U,V formulas have
        INTEGER :: p,q     ! the factor SQRT(h), where
        INTEGER :: a,b    !h = b*(p-b+1)*(q+b+1)/[(a+b)(a+b+1)]
        REAL(8) :: w      ! w = h
    END FUNCTION
END INTERFACE
!
!------------------------3. Type declarations -----------------------
!
INTEGER ::  p,q         ! integers p,q identify the (p,q) SU(3) irrep
INTEGER ::  a,b,c !(a,b) identifies an SU2-irrep in the reduced T-matrices
INTEGER ::  ai,bi,aj,bj ! i for row and j for column; used with blocks
!  ALLOCATABLE allows an altered code to calculate more pairs of (p,q)
REAL(8), ALLOCATABLE :: unitMatrix(:,:)
REAL(8) :: fijk(8,8,8),dijk(8,8,8)
REAL(8),ALLOCATABLE :: T3(:,:),Y(:,:),Tp(:,:),Tm(:,:),Up(:,:),Um(:,:)
REAL(8),ALLOCATABLE :: Vp(:,:),Vm(:,:)
!
REAL(8) :: MaxErr,MaxErrLimit !Max error found in the equations, tolerance
INTEGER ::  dimREP    !  TYUV matrix dimension; dimREP = #rows = #cols
REAL(8) ::  alpha,beta ! spin indices of components in a block
REAL(8) ::  t,trTYUV(8) ! spin of an SU2 irrep,trace of a TYUV matrix
INTEGER  :: i,j,k,m,n,r,w   ! dummy indices; r row, w column, n dimREP
!
REAL(8),ALLOCATABLE:: T3Tpcomm(:,:),T3Tmcomm(:,:),T3Upcomm(:,:)
REAL(8),ALLOCATABLE:: T3Umcomm(:,:),T3Ycomm(:,:),T3Vpcomm(:,:)
REAL(8),ALLOCATABLE:: T3Vmcomm(:,:),TpTmcomm(:,:),TpUpcomm(:,:)
REAL(8),ALLOCATABLE:: TpUmcomm(:,:),TpYcomm(:,:),TpVpcomm(:,:)
REAL(8),ALLOCATABLE:: TpVmcomm(:,:),TmUpcomm(:,:),TmUmcomm(:,:)
REAL(8),ALLOCATABLE:: TmYcomm(:,:),TmVpcomm(:,:),TmVmcomm(:,:)
REAL(8),ALLOCATABLE:: YUpcomm(:,:),YUmcomm(:,:),YVpcomm(:,:)
REAL(8),ALLOCATABLE:: YVmcomm(:,:),UpUmcomm(:,:),UpVpcomm(:,:)
REAL(8),ALLOCATABLE:: UpVmcomm(:,:),UmVpcomm(:,:),UmVmcomm(:,:)
```

```
REAL(8),ALLOCATABLE:: VpVmcomm(:,:),Casimir(:,:)
!
LOGICAL:: MaxErrNotTooBig,traceTYUVok  ! error within tolerance? T-Yes
CHARACTER (len=20) :: file_name    ! file name for output data file
!
COMPLEX(8),ALLOCATABLE :: Fj(:,:,:),CRerror(:,:,:,:),hConj(:,:,:)
COMPLEX(8),ALLOCATABLE :: C1matrix(:,:),C2matrix(:,:),traceCMPLX(:)
COMPLEX(8) :: traceFj(8)
REAL(8) :: maxCRerror,maxC1error,maxC2error,hermTEST
LOGICAL :: maxCRerrOK,traceFjOK  !  error within tolerance? T-Yes
!
!
!---------------------4. Set (p,q), MaxErrLimit --------------------------
!
! Set p and q so that the (p,q)-irrep is calculated.
p = 2        ! the irrep's identifiers (p,q)
q = 1       !
!  Set the tolerance:
   MaxErrLimit = 1.D-12 ! the largest allowed error in
   !                        any component of the 29 eqns
!
!   Calculate the dimension of the irrep:
dimREP = (p+1)*(q+1)*(p+q+2)/2 !Matrices are nxn square, n = dimREP
!
!WRITE(*,*) 'p,q = ', p,q
WRITE(*,*)  'For p,q = ',p,q,', '
WRITE(*,*)  ' the program calculates the TYUV basis generators for ', &
            'the p,q ','irrep of sl(3,C) '
WRITE(*,*)  'and the Fj basis generators for the p,q irrep of su(3).'
WRITE(*,*) 'For each matrix, # of rows = # of cols = dimension = dimREP &
            = ', dimREP
!
!-----------------------5. Preliminaries ----------------------
!
!            unit matrix, a.k.a. identity matrix

 ALLOCATE( unitMatrix(dimREP,dimREP))
    DO  i = 1,dimREP
        DO  j = 1,dimREP
    unitMatrix(i,j) = 0.0_8       ! The unit matrix has zeros everywhere,
```

```
        END DO
            unitMatrix(i,i) = 1.0_8    ! except ones along the diagonal.
    END DO
!
!           Structure constants fijk
!
Do i=1,8
    Do j=1,8            !Initial fijk = 0
        Do k=1,8
            fijk(i,j,k)=0._8
        End Do
    End Do
End Do
fijk(1,2,3) = 1._8          !Select Nonzero fijk
fijk(1,4,7) = 1._8/2._8     !('Seed' values with i<j<k)
fijk(1,5,6) = -1._8/2._8
fijk(2,4,6) = 1._8/2._8
fijk(2,5,7) = 1._8/2._8
fijk(3,4,5) = 1._8/2._8
fijk(3,6,7) = -1._8/2._8
fijk(4,5,8) = DSQRT(3._8)/2._8
fijk(6,7,8) = DSQRT(3._8)/2._8
!
Do i=1,6
    Do j=2,7    ! Antisymmetrize the select nonzero fijk
        Do k=3,8
            fijk(i,k,j)=-fijk(i,j,k)
            fijk(k,j,i)=-fijk(i,j,k)
            fijk(k,i,j)=+fijk(i,j,k)
            fijk(j,i,k)=-fijk(i,j,k)
            fijk(j,k,i)=+fijk(i,j,k)
        End Do
    End Do
End Do
!
!           Symmetric coefficient constants dijk
!
Do i=1,8
    Do j=1,8            !Initial dijk = 0
        Do k=1,8
```

```
            dijk(i,j,k)=0._8
        End Do
    End Do
End Do
dijk(1,1,8) = 1._8/DSQRT(3._8)        !Select Nonzero dijk
dijk(1,4,6) = 1._8/2._8             !('Seed' values with i<j<k)
dijk(1,5,7) = 1._8/2._8
dijk(2,2,8) = 1._8/DSQRT(3._8)
dijk(2,4,7) = -1._8/2._8
dijk(2,5,6) = 1._8/2._8
dijk(3,3,8) = 1._8/DSQRT(3._8)
dijk(3,4,4) = 1._8/2._8
dijk(3,5,5) = 1._8/2._8
dijk(3,6,6) = -1._8/2._8
dijk(3,7,7) = -1._8/2._8
dijk(4,4,8) = -1._8/(2._8*DSQRT(3._8))
dijk(5,5,8) = -1._8/(2._8*DSQRT(3._8))
dijk(6,6,8) = -1._8/(2._8*DSQRT(3._8))
dijk(7,7,8) = -1._8/(2._8*DSQRT(3._8))
dijk(8,8,8) = -1._8/DSQRT(3._8)
Do i=1,8
    Do j=1,8    ! Symmetrize the select nonzero dijk
        Do k=1,8
            dijk(i,k,j)=dijk(i,j,k)
            dijk(j,i,k)=dijk(i,j,k)
            dijk(j,k,i)=dijk(i,j,k)
            dijk(k,j,i)=dijk(i,j,k)
            dijk(k,i,j)=dijk(i,j,k)
        End Do
    End Do
End Do
!
!
!-----------------6. Make T3 ----------------------
!
ALLOCATE( T3(dimREP,dimREP))
!
DO  i = 1,dimREP ! Initially null
    DO  j = 1,dimREP
        T3(i,j) = 0._8
```

```
    END DO
END DO
!
DO b = 0,p
    DO a = 0,q
        DO m = 1,a+b+1
            alpha = DBLE((a+b)/2._8 - m + 1._8)
            r = nSEQUENCE(p,q,a,b,alpha)
            T3(r,r) = alpha
        END DO
    END DO
END DO
!   trace
trTYUV(1) = 0._8    ! initialize the trace
Do r = 1,dimREP
    trTYUV(1) = trTYUV(1) + T3(r,r)
End Do
!WRITE(*,*) "tr(T3) = ",trTYUV(1)
!
!-----------------7. Make Y ----------------------
!
ALLOCATE( Y(dimREP,dimREP))
!
DO  i = 1,dimREP    ! Initially null
    DO  j = 1,dimREP
        Y(i,j) = 0._8
    END DO
END DO
!
DO b = 0,p
    DO a = 0,q
        DO m = 1,a+b+1
            alpha = DBLE((a+b)/2._8 - m + 1._8)
            r = nSEQUENCE(p,q,a,b,alpha)
            Y(r,r) = DBLE(b-a-2._8*(p-q)/3._8)
        END DO
    END DO
END DO
!   trace
trTYUV(2) = 0._8    ! initialize the trace
```

```
Do r = 1,dimREP
    trTYUV(2) = trTYUV(2) + Y(r,r)
End Do
!WRITE(*,*) "tr(Y) = ",trTYUV(2)
!
!-----------------8. Make Tp ----------------------
!
ALLOCATE( Tp(dimREP,dimREP))
!
DO  i = 1,dimREP    ! Initially null
    DO  j = 1,dimREP
        Tp(i,j) = 0._8
    END DO
END DO
!
DO b = 0,p
    DO a = 0,q
        DO m = 1,a+b
            alpha = DBLE((a+b)/2._8 - m + 1._8)
            beta = alpha - 1._8
             r = nSEQUENCE(p,q,a,b,alpha)
             c = nSEQUENCE(p,q,a,b,beta)
              t = (a+b)/2._8
        Tp(r,c) = DSQRT((t+alpha)*(1._8+t-alpha))
        END DO
    END DO
END DO
!   trace
trTYUV(3) = 0._8    ! initialize the trace
Do r = 1,dimREP
    trTYUV(3) = trTYUV(3) + Tp(r,r)
End Do
!WRITE(*,*) "tr(Tp) = ",trTYUV(3)
!
!-----------------9. Make Tm ----------------------
!
ALLOCATE( Tm(dimREP,dimREP))
!
DO  i = 1,dimREP    ! Initially null
    DO  j = 1,dimREP
```

```
        Tm(i,j) = 0._8
    END DO
END DO
!
DO b = 0,p
    DO a = 0,q
        DO m = 2,a+b+1
            alpha = DBLE((a+b)/2._8 - m + 1._8)
            beta = alpha + 1._8
             r = nSEQUENCE(p,q,a,b,alpha)
             c = nSEQUENCE(p,q,a,b,beta)
              t = (a+b)/2._8
        Tm(r,c) = DSQRT((t-alpha)*(1._8+t+alpha))
        END DO
    END DO
END DO
!   trace
trTYUV(4) = 0._8    ! initialize the trace
Do r = 1,dimREP
    trTYUV(4) = trTYUV(4) + Tm(r,r)
End Do
!WRITE(*,*) "tr(Tm) = ",trTYUV(4)
!
!
!-----------------10. Make Up -----------------------
!
ALLOCATE( Up(dimREP,dimREP))
!
! Make the upper blocks of Up
DO  i = 1,dimREP    ! Initially null
    DO  j = 1,dimREP
        Up(i,j) = 0._8
    END DO
END DO
!
DO b = 0,p                                  !upper block
    DO a = 1,q
        DO m = 1,a+b+1
            ai=a-1
            beta = DBLE((a+b)/2._8 - m + 1._8)
```

```
            alpha = beta - 0.5_8
    IF ((alpha .LT. -(ai+b)/2._8).OR.(alpha .GT. (ai+b)/2._8)) THEN
    CYCLE
    END IF
             r = nSEQUENCE(p,q,ai,b,alpha)
             c = nSEQUENCE(p,q,a,b,beta)
            t = (a+b)/2._8
         Up(r,c) = DSQRT(g(p,q,a,b)*(t+beta))
        END DO
    END DO
END DO
! Make the lower blocks of Up
DO b = 1,p                                !lower block
    DO a = 0,q
        DO m = 1,a+b+1  ! replace 1 by 2 and drop IF
            bj = b - 1
            alpha = DBLE((a+b)/2._8 - m + 1._8)
            beta = alpha + 0.5_8
    IF ((beta .LT. -(a+bj)/2._8).OR.(beta .GT. (a+bj)/2._8)) THEN
    CYCLE
    END IF
             r = nSEQUENCE(p,q,a,b,alpha)
             c = nSEQUENCE(p,q,a,bj,beta)
            t = (a+b)/2._8
        Up(r,c) = DSQRT(h(p,q,a,b)*(t-alpha))
        END DO
    END DO
END DO
!   trace
trTYUV(5) = 0._8    ! initialize the trace
Do r = 1,dimREP
    trTYUV(5) = trTYUV(5) + Up(r,r)
End Do
!WRITE(*,*) "tr(Up) = ",trTYUV(5)
!
!
!-----------------11. Make Um ----------------------
!
ALLOCATE( Um(dimREP,dimREP))
!
```

```
! Make the upper blocks of Um
DO  i = 1,dimREP    ! Initially null
    DO  j = 1,dimREP
        Um(i,j) = 0._8
    END DO
END DO
!
! Make the upper blocks of Um
DO b = 1,p                                !upper block
    DO a = 0,q
        DO m = 2,a+b+1
            bi = b-1
            beta = DBLE((a+b)/2._8 - m + 1._8)
            alpha = beta + 0.5_8
    IF ((alpha .LT. -(a+bi)/2._8).OR.(alpha .GT. (a+bi)/2._8)) THEN
    CYCLE
    END IF
             r = nSEQUENCE(p,q,a,bi,alpha)
             c = nSEQUENCE(p,q,a,b,beta)
            t = (a+b)/2._8
        Um(r,c) = DSQRT(h(p,q,a,b)*(t-beta))
        END DO
    END DO
END DO
! Make the lower blocks of Um
DO b = 0,p                                !lower block
    DO a = 1,q
        DO m = 1,a+b+1  !for m = 1, alpha = t = (a+b)/2 MOVE TO Up
            aj = a - 1  ! tj = (a+b-1)/2
            alpha = DBLE((a+b)/2._8 - m + 1._8)
            beta = alpha - 0.5_8 !For m = 1, beta = (a+b-1)/2
    IF ((beta .LT. -(aj+b)/2._8).OR.(beta .GT. (aj+b)/2._8)) THEN
    CYCLE
    END IF
             r = nSEQUENCE(p,q,a,b,alpha)
             c = nSEQUENCE(p,q,aj,b,beta)
            t = (a+b)/2._8
        Um(r,c) = DSQRT(g(p,q,a,b)*(t+alpha))
        END DO
    END DO
```

```
END DO
!   trace
trTYUV(6) = 0._8    ! initialize the trace
Do r = 1,dimREP
    trTYUV(6) = trTYUV(6) + Um(r,r)
End Do
!WRITE(*,*) "tr(Um) = ",trTYUV(6)
!
!
!-----------------12. Make Vp ----------------------
!
ALLOCATE( Vp(dimREP,dimREP))
!
! Make the upper blocks of Vp
DO  i = 1,dimREP    ! Initially null
    DO  j = 1,dimREP
        Vp(i,j) = 0._8
    END DO
END DO
!
DO b = 0,p                                !upper block
    DO a = 1,q
        DO m = 1,a+b+1
            ai = a-1
            beta = DBLE((a+b)/2._8 - m + 1._8)
            alpha = beta + 0.5_8
    IF ((alpha .LT. -(ai+b)/2._8).OR.(alpha .GT. (ai+b)/2._8)) THEN
    CYCLE
    END IF
             r = nSEQUENCE(p,q,ai,b,alpha)
             c = nSEQUENCE(p,q,a,b,beta)
            t = (a+b)/2._8
        Vp(r,c) = -DSQRT(g(p,q,a,b)*(t-beta))
        END DO
    END DO
END DO
! Make the lower blocks of Vp
DO b = 1,p                                !lower block
    DO a = 0,q
        t = DBLE((a+b)/2._8)
```

```
        DO m = 1,a+b+1  ! 1 replaced by 2
            bj = b - 1
            alpha = DBLE(t - m + 1._8)
            beta = alpha - 0.5_8
    IF ((beta .LT. -(a+bj)/2._8).OR.(beta .GT. (a+bj)/2._8)) THEN
    CYCLE
    END IF
             r = nSEQUENCE(p,q,a,b,alpha)
             c = nSEQUENCE(p,q,a,bj,beta)
            t = (a+b)/2._8
        Vp(r,c) = DSQRT(h(p,q,a,b)*(t+alpha))
        END DO
    END DO
END DO
!   trace
trTYUV(7) = 0._8    ! initialize the trace
Do r = 1,dimREP
    trTYUV(7) = trTYUV(7) + Vp(r,r)
End Do
!WRITE(*,*) "tr(Vp) = ",trTYUV(7)
!
!
!-----------------13. Make Vm -----------------------
!
ALLOCATE( Vm(dimREP,dimREP))
!
! Make the upper blocks of Vm
DO  i = 1,dimREP    ! Initially null
    DO  j = 1,dimREP
        Vm(i,j) = 0._8
    END DO
END DO
DO b = 1,p                                  !upper block
    DO a = 0,q
        t = (a+b)/2._8
        DO m = 1,a+b+1
            bi = b-1
            beta = DBLE(t - m + 1._8)
            alpha = beta - 0.5_8
    IF ((alpha .LT. -(a+bi)/2._8).OR.(alpha .GT. (a+bi)/2._8)) THEN
```

```
    !WRITE(*,*)  "a,bi,ti,r,alpha = ", a,bi,(a+bi)/2._8,r,alpha
    CYCLE
    END IF
             r = nSEQUENCE(p,q,a,bi,alpha)
             c = nSEQUENCE(p,q,a,b,beta)
        Vm(r,c) = DSQRT(h(p,q,a,b)*(t+beta))
        END DO
    END DO
END DO
! Make the lower block of Vm
!
DO b = 0,p                                !lower block
    DO a = 1,q
        t = (a+b)/2._8
        DO m = 1,a+b+1
            aj = a - 1
            alpha = DBLE(t - m + 1._8)
            beta = alpha + 0.5_8
    IF ((beta .LT. -(aj+b)/2._8).OR.(beta .GT. (aj+b)/2._8)) THEN
    CYCLE
    END IF
               r = nSEQUENCE(p,q,a,b,alpha)
             c = nSEQUENCE(p,q,aj,b,beta)
        Vm(r,c) = -DSQRT(g(p,q,a,b)*(t-alpha))
        END DO
    END DO
END DO
!   trace
trTYUV(8) = 0._8    ! initialize the trace
Do r = 1,dimREP
    trTYUV(8) = trTYUV(8) + Vm(r,r)
End Do
traceTYUVok = (MAX(ABS(trTYUV(1)),ABS(trTYUV(2)),ABS(trTYUV(3)),&
            ABS(trTYUV(4)),ABS(trTYUV(5)),ABS(trTYUV(6)),&
            ABS(trTYUV(7)),ABS(trTYUV(8))) )<MaxErrLimit
!WRITE(*,*) "tr(Vm) = ",trTYUV(8)
!
! The 8 matrices T3,Y,Tp,Tm,Up,Um,Vp,Vm have been calculated. Check them.
!
!
```

```
! ----14. Check 28 commutator relations and the Casimir equation --------
!
n = dimREP ! Use n to abbreviate dimREP; a single character versus six
!
ALLOCATE(T3Tpcomm(n,n),T3Tmcomm(n,n),T3Upcomm(n,n),T3Umcomm(n,n))
ALLOCATE(T3Ycomm(n,n),T3Vpcomm(n,n),T3Vmcomm(n,n),TpTmcomm(n,n))
ALLOCATE(TpUpcomm(n,n),TpUmcomm(n,n),TpYcomm(n,n),TpVpcomm(n,n))
ALLOCATE(TpVmcomm(n,n),TmUpcomm(n,n),TmUmcomm(n,n),TmYcomm(n,n))
ALLOCATE(TmVpcomm(n,n),TmVmcomm(n,n),YUpcomm(n,n),YUmcomm(n,n))
ALLOCATE(YVpcomm(n,n),YVmcomm(n,n),UpUmcomm(n,n),UpVpcomm(n,n))
ALLOCATE(UpVmcomm(n,n),UmVpcomm(n,n),UmVmcomm(n,n),VpVmcomm(n,n))
ALLOCATE(Casimir(n,n))
!
    T3Tpcomm = MATMUL(T3,Tp) - MATMUL(Tp,T3) - Tp                    ! 1
    MaxErr = MAXVAL(ABS( T3Tpcomm ))
        T3Tmcomm = MATMUL(T3,Tm) - MATMUL(Tm,T3) + Tm
        MaxErr = MAXVAL(ABS( (/MaxErr, T3Tmcomm/) ))
            T3Upcomm = MATMUL(T3,Up) - MATMUL(Up,T3) + Up/2._8
            MaxErr = MAXVAL(ABS( (/MaxErr, T3Upcomm/) ))
                T3Umcomm = MATMUL(T3,Um) - MATMUL(Um,T3) - Um/2._8
                MaxErr = MAXVAL(ABS( (/MaxErr, T3Umcomm/) ))
                    T3Ycomm = MATMUL(T3,Y) - MATMUL(Y,T3)            ! 5
                    MaxErr = MAXVAL(ABS( (/MaxErr, T3Ycomm/) ))
                        T3Vpcomm = MATMUL(T3,Vp) - MATMUL(Vp,T3) - Vp/2.
                        MaxErr = MAXVAL(ABS( (/MaxErr, T3Vpcomm/) ))
                          T3Vmcomm = MATMUL(T3,Vm) - MATMUL(Vm,T3) + Vm/2.
                          MaxErr = MAXVAL(ABS( (/MaxErr, T3Vmcomm/) ))
 !
    TpTmcomm = MATMUL(Tp,Tm) - MATMUL(Tm,Tp) - 2._8*T3
    MaxErr = MAXVAL(ABS( (/MaxErr, TpTmcomm/) ))
        TpUpcomm = MATMUL(Tp,Up) - MATMUL(Up,Tp) - Vp
        MaxErr = MAXVAL(ABS( (/MaxErr, TpUpcomm/) ))
            TpUmcomm = MATMUL(Tp,Um) - MATMUL(Um,Tp)                 ! 10
            MaxErr = MAXVAL(ABS( (/MaxErr, TpUmcomm/) ))
                TpYcomm = MATMUL(Tp,Y) - MATMUL(Y,Tp)
                MaxErr = MAXVAL(ABS( (/MaxErr, TpYcomm/) ))
                    TpVpcomm = MATMUL(Tp,Vp) - MATMUL(Vp,Tp)
                    MaxErr = MAXVAL(ABS( (/MaxErr, TpVpcomm/) ))
                        TpVmcomm = MATMUL(Tp,Vm) - MATMUL(Vm,Tp) + Um
                        MaxErr = MAXVAL(ABS( (/MaxErr, TpVmcomm/) ))
```

```
        TmUpcomm = MATMUL(Tm,Up) - MATMUL(Up,Tm)
        MaxErr = MAXVAL(ABS( (/MaxErr, TmUpcomm/) ))
            TmUmcomm = MATMUL(Tm,Um) - MATMUL(Um,Tm) + Vm            ! 15
            MaxErr = MAXVAL(ABS( (/MaxErr, TmUmcomm/) ))
                TmYcomm = MATMUL(Tm,Y) - MATMUL(Y,Tm)
                MaxErr = MAXVAL(ABS( (/MaxErr, TmYcomm/) ))
                    TmVpcomm = MATMUL(Tm,Vp) - MATMUL(Vp,Tm) - Up
                    MaxErr = MAXVAL(ABS( (/MaxErr, TmVpcomm/) ))
                        TmVmcomm = MATMUL(Tm,Vm) - MATMUL(Vm,Tm)
                        MaxErr = MAXVAL(ABS( (/MaxErr, TmVmcomm/) ))
!
        YUpcomm = MATMUL(Y,Up) - MATMUL(Up,Y) - Up
        MaxErr = MAXVAL(ABS( (/MaxErr, YUpcomm/) ))
            YUmcomm = MATMUL(Y,Um) - MATMUL(Um,Y) + Um               ! 20
            MaxErr = MAXVAL(ABS( (/MaxErr, YUmcomm/) ))
                YVpcomm = MATMUL(Y,Vp) - MATMUL(Vp,Y) - Vp
                MaxErr = MAXVAL(ABS( (/MaxErr, YVpcomm/) ))
                    YVmcomm = MATMUL(Y,Vm) - MATMUL(Vm,Y) + Vm
                    MaxErr = MAXVAL(ABS( (/MaxErr, YVmcomm/) ))
        UpUmcomm = MATMUL(Up,Um) - MATMUL(Um,Up) - (3._8*Y/2._8 - T3)
        MaxErr = MAXVAL(ABS( (/MaxErr, UpUmcomm/) ))
            UpVpcomm = MATMUL(Up,Vp) - MATMUL(Vp,Up)
            MaxErr = MAXVAL(ABS( (/MaxErr, UpVpcomm/) ))
                UpVmcomm = MATMUL(Up,Vm) - MATMUL(Vm,Up) - Tm        ! 25
                MaxErr = MAXVAL(ABS( (/MaxErr, UpVmcomm/) ))
        UmVpcomm = MATMUL(Um,Vp) - MATMUL(Vp,Um) + Tp
        MaxErr = MAXVAL(ABS( (/MaxErr, UmVpcomm/) ))
            UmVmcomm = MATMUL(Um,Vm) - MATMUL(Vm,Um)
            MaxErr = MAXVAL(ABS( (/MaxErr, UmVmcomm/) ))
        VpVmcomm = MATMUL(Vp,Vm) - MATMUL(Vm,Vp) - (3._8*Y/2._8 + T3) ! 28
        MaxErr = MAXVAL(ABS( (/MaxErr, VpVmcomm/) ))
!
        Casimir = (MATMUL(Tp,Tm)+MATMUL(Tm,Tp)+MATMUL(Up,Um)+ &
            MATMUL(Um,Up)+MATMUL(Vp,Vm)+MATMUL(Vm,Vp))/2.+MATMUL(T3,T3) &
        +3._8*MATMUL(Y,Y)/4._8 - (dble(p**2+p*q+q**2)/3._8+ &
         dble(p + q))*unitMatrix                                     ! 29
        MaxErr = MAXVAL(ABS( (/MaxErr, Casimir/) ))
!
!
!
```

```
    IF(MaxErr.LE.MaxErrLimit) THEN !Is the largest error within tolerance?
        MaxErrNotTooBig = .TRUE.        ! TRUE means Yes.
    ELSE IF (MaxErr > MaxErrLimit)    THEN
        MaxErrNotTooBig = .FALSE.  ! FALSE means No, not within tolerance.
    END IF
!
!----------15. Save the TYUV matrices to a file--------------------
!
!MaxErrNotTooBig = .FALSE. ! Use this statement to test the failure option
!MaxErrNotTooBig = .TRUE.    ! Use this statement to make an output
!
    IF (MaxErrNotTooBig.AND.traceTYUVok) THEN     ! No Failure(s) found
        WRITE (file_name,"('p',i0,'q',i0,'sl3Cgen.dat')")p,q
        OPEN(Unit=5,file=file_name)
        WRITE(5,3000) p,q, MaxErr   ! Start with p,q,MaxErr.
        CLOSE(5)
    OPEN(Unit=5,file=file_name,STATUS='OLD', POSITION='APPEND') !
!    Next, upload the 8*dimREP**2 components of the 8 basis matrices:
   WRITE(5,4000) TRANSPOSE(T3),TRANSPOSE(Y),TRANSPOSE(Tp), TRANSPOSE(Tm),&
                 TRANSPOSE(Up), TRANSPOSE(Um), TRANSPOSE(Vp), TRANSPOSE(Vm)
      CLOSE(5)
    END IF
!
3000 FORMAT(2I3,1ES16.7)  ! Record 1 has two integers and a real number
4000 FORMAT(5F16.12)  !Components of the 8 matrices, 5 numbers per line
WRITE(*,*)      ! a blank line
WRITE(*,*) '        TYUV basis of sl(3,C)'
WRITE(*,*)  'The TYUV basis satisfies 28 commutation relations (CR) &
             plus the quadratic Casimir:',(MaxErr.LE.MaxErrLimit)
WRITE(*,*) ' For the TYUV basis, Max Error in 29 eqns = ',MaxErr
WRITE(*,*)  'Tolerance = ',MaxErrLimit
WRITE(*,*) "All TYUV have |trace| < Tolerance: ", traceTYUVok
WRITE(*,*) "The TYUV output file name, p#q#sl3Cgen.dat, has &
            the #s replaced by the integers p = ",p, &
            " and q = ",q,"."
!
!
!----------16. The Fj basis for su(3)  ------------------
!
WRITE(*,*)      ! a blank line
```

```
WRITE(*,*)  "         The Fj basis of su(3)"
! A basis for su(3):
!
! F1 = (Tp+Tm)/2 ,  F2 = -i(Tp-Tm)/2 , F3 = T3 , F4 = (Vp+Vm)/2 ,
! F5 = -i(Vp-Vm)/2 , F6 = (Up+Um)/2 , F7 = -i(Up-Um)/2 , F8 = \sqrt{3}Y/2
!
! The Fj matrices are hermitian and traceless.
!
!        Vm(r,c) = -DSQRT(g(p,q,a,b)*(t-alpha))
!
n = dimREP ! Use n to abbreviate dimREP; a single character versus six
!
ALLOCATE(Fj(8,n,n),CRerror(8,8,n,n),C1matrix(n,n),C2matrix(n,n))
ALLOCATE(hConj(8,n,n),traceCMPLX(n))
DO r = 1,dimREP                                !lower block
    DO c = 1,dimREP
        Fj(1,r,c) = (Tp(r,c)+Tm(r,c))/2._8              !F1(r,c)
        Fj(2,r,c) = -(0,1._8)*(Tp(r,c)-Tm(r,c))/2._8    !F2(r,c)
        Fj(3,r,c) = T3(r,c)                             !F3(r,c)
        Fj(4,r,c) = (Vp(r,c)+Vm(r,c))/2._8              !F4(r,c)
        Fj(5,r,c) = -(0,1._8)*(Vp(r,c)-Vm(r,c))/2._8    !F5(r,c)
        Fj(6,r,c) = (Up(r,c)+Um(r,c))/2                 !F6(r,c)
        Fj(7,r,c) = -(0,1._8)*(Up(r,c)-Um(r,c))/2._8    !F7(r,c)
        Fj(8,r,c) =  DSQRT(3._8)*Y(r,c)/2._8            !F8(r,c)
    END DO
END DO
!
!
!               Find the errors in the commutation relations (CR).
!
maxCRerror=0._8
Do i=1,7
    Do j=2,8 !Start with commutator [F1,F2] of CR
          CRerror(i,j,:,:) = MATMUL(Fj(i,:,:),Fj(j,:,:)) - &
                             MATMUL(Fj(j,:,:),Fj(i,:,:))
        Do k=1,8 !Subtract right side of CR, I*fijk*Fk one k at a time
          CRerror(i,j,:,:) = CRerror(i,j,:,:) - &
                             (0,1._8)*fijk(i,j,k)*Fj(k,:,:)
        End Do
      maxCRerror = MAX(maxCRerror,MAXVAL(ABS( (/(CRerror(i,j,:,:))/) )))
```

```
    End Do
End Do
!                   Is the largest error within tolerance?
    IF(maxCRerror.LE.MaxErrLimit) THEN
        maxCRerrOK = .TRUE.        ! TRUE means Yes.
    ELSE IF (maxCRerror > MaxErrLimit)    THEN
        maxCRerrOK = .FALSE.  ! FALSE means No, not within tolerance.
    END IF
!
WRITE(*,*)  'The Fj basis satisfies the 28 commutation relations (CR):',&
                maxCRerrOK
WRITE(*,*) "For the Fj basis, max CR error = ",maxCRerror
WRITE(*,*)  'Tolerance = ',MaxErrLimit
!
Do j=1,8                          !TRACE
!WRITE(*,*) "Fj(j,:,:) = ",Fj(j,:,:)
    traceFj(j) =DCMPLX(0._8)
    Do k=1,n
    traceFj(j) = traceFj(j)+Fj(j,k,k)
    End Do
End Do
traceFjOK = (MAX(ABS(traceFj(1)),ABS(traceFj(2)),ABS(traceFj(3)),&
            ABS(traceFj(4)),ABS(traceFj(5)),ABS(traceFj(6)),&
            ABS(traceFj(7)),ABS(traceFj(8))) )<MaxErrLimit
!
!WRITE(*,*) "trace Fk: ",ABS(traceFj(:))
!WRITE(*,*) "For all j, trace Fj = 0: ",(traceFj(1).EQ.0).AND.&
! (traceFj(2).EQ.0).AND.(traceFj(3).EQ.0).AND.(traceFj(4).EQ.0).AND.&
!            (traceFj(5).EQ.0).AND.(traceFj(6).EQ.0).AND.&
!            (traceFj(7).EQ.0).AND.(traceFj(8).EQ.0)
WRITE(*,*) "All Fj have |trace| < Tolerance: ", traceFjOK
!
hermTEST = 0._8
!WRITE(*,*) 'hermTEST = ', hermTEST
Do j=1,8
    Do r=1,dimREP
        Do c=1,dimREP
!WRITE(*,*) "j,r,c = ", j, r, c
!WRITE(*,*) "DCONJG(Fj(j,c,r)) = ",DCONJG(Fj(j,c,r))
            hConj(j,r,c) = DCONJG(Fj(j,c,r))
```

```
!WRITE(*,*) "hConj(j,r,c) = ", hConj(j,r,c)
             hermTEST=hermTEST + ABS(hConj(j,r,c)-Fj(j,r,c))
        End Do
    End Do
End Do
WRITE(*,*) 'All 8 Fj are hermitian: ', hermTEST<MaxErrLimit
!
!----------17. Save the Fj matrices to a file--------------------
!
!maxCRerrOK = .FALSE. ! Use this statement to test the failure option
!maxCRerrOK = .TRUE.    ! Use this statement to make an output
!WRITE(*,*) maxCRerrOK ! The value of maxCRerrOK determined by the program.
!                   Save the Fj matrices if there are no failures found.
    IF (maxCRerrOK.AND.traceFjOK.AND.(hermTEST<MaxErrLimit)) THEN
        WRITE (file_name,"('p',i0,'q',i0,'su3gen.dat')")p,q
        OPEN(Unit=5,file=file_name)
        WRITE(5,3050) p,q, maxCRerror   ! Start with p,q,maxCRerror.
        CLOSE(5)
    OPEN(Unit=5,file=file_name,STATUS='OLD', POSITION='APPEND') !
!    Next, upload the 8*dimREP**2 components of the 8 basis matrices:
   WRITE(5,4050) TRANSPOSE(Fj(1,:,:)),TRANSPOSE(Fj(2,:,:)),&
                 TRANSPOSE(Fj(3,:,:)), TRANSPOSE(Fj(4,:,:)),&
                 TRANSPOSE(Fj(5,:,:)), TRANSPOSE(Fj(6,:,:)),&
                 TRANSPOSE(Fj(7,:,:)), TRANSPOSE(Fj(8,:,:))
      CLOSE(5)
    END IF
!
3050 FORMAT(2I3,1ES16.7)  ! Record 1 has two integers and a real number
4050 FORMAT((2F16.12,2F16.12))  !Complex matrix components, 2 per line
!
WRITE(*,*) "The Fj output file name, p#q#su3gen.dat, has &
            the #s replaced by the integers p = ",p, &
            " and q = ",q
!
!----------18. Quadratic Casimir operator  ------------------
!
!
WRITE(*,*)      ! a blank line
WRITE(*,*)  "         Quadratic Casimir operator"
C1matrix(:,:)=MATMUL(Fj(1,:,:),Fj(1,:,:))! Initially, C1 = F1.F1
```

```
Do i=2,8
    C1matrix(:,:)=C1matrix(:,:)+MATMUL(Fj(i,:,:),Fj(i,:,:))
End Do
maxC1error = MAXVAL(ABS( (/C1matrix(:,:)-&
 DCMPLX(((p**2 + q**2 + 3._8*p + 3._8*q + p*q)/3._8)*unitMatrix(:,:))/) ))
!WRITE(*,*) !space
WRITE(*,*) 'C1 = FjFj = (p^2 + q^2 + 3 p + 3 q + p q)/3 = ',&
            (p**2 + q**2 + 3._8*p + 3._8*q + p*q)/3._8
WRITE(*,*) 'max error in C1 equation =',maxC1error," , &
        so the max C1 error is within tolerance: ",(maxC1error<MaxErrLimit)
WRITE(*,*)  'Tolerance = ',MaxErrLimit
!
!
!----------19. Cubic Casimir operator, end program  ------------------
!
!
WRITE(*,*)      ! a blank line
WRITE(*,*)  "         Cubic Casimir operator"
! C2 = dijk FiFjFk = (p-q)(3+p+2q)(3+q+2p)/18)
!
C2matrix(:,:)= DCMPLX(unitMatrix(:,:)-unitMatrix(:,:)) ! Initially, C2 = 0
Do i=1,8
    Do j=1,8
        Do k=1,8
   C2matrix(:,:) = C2matrix(:,:)+&
                dijk(i,j,k)*MATMUL(MATMUL(Fj(i,:,:),Fj(j,:,:)),Fj(k,:,:))
        End Do
    End Do
End Do
maxC2error = MAXVAL(ABS( (/C2matrix(:,:)-&
            DCMPLX((p-q)*((3._8)+p+(2._8)*q)*((3._8)+q+(2._8)*p)/(18._8)*&
                            unitMatrix(:,:))/) ))
WRITE(*,*) "C2 = dijk FiFjFk = (p-q)(3+p+2q)(3+q+2p)/18) = ",&
                (p-q)*((3._8)+p+(2._8)*q)*((3._8)+q+(2._8)*p)/(18._8)
WRITE(*,*) 'max C2 error = ',maxC2error ," , &
        so the max C2 error is within tolerance: ",(maxC2error<MaxErrLimit)
WRITE(*,*)  'Tolerance = ',MaxErrLimit
WRITE(*,*)
!
!
```

```
!
                        END PROGRAM SL3CSu3Generators
!
!-----------------20. External functions, end-of-file--------------------
!
FUNCTION nSEQUENCE(p,q,a,b,alpha) RESULT  (w)
    IMPLICIT NONE        ! The nth place in the sequence belongs to
        INTEGER :: p,q     ! the T3 eigenvector with eigenvalue alpha
        INTEGER :: a,b,m    ! in the (a,b)-SU2 irrep
        REAL(8) :: alpha,w !the T3 eigenvector has the mth place in the
!                           ! irrep, where m = (a+b)/2 -alpha + 1
    w = ((a+b)*(a+b+1._8)+q*b*(q+b+2._8))/2._8 + ((a+b)/2._8-alpha+1._8)
    END FUNCTION
FUNCTION g(p,q,a,b) RESULT  (w)
    IMPLICIT NONE        ! Four of the U,V formulas have
    INTEGER :: p,q     ! the factor SQRT(g), where
        INTEGER :: a,b    !g = a*(p+a+1)*(q-a+1)/[(a+b)(a+b+1)]
        REAL(8) :: w      ! w = g
    w = a*(p+a+1._8)*(q-a+1._8)/((a+b)*(a+b+1._8))
    END FUNCTION
FUNCTION h(p,q,a,b) RESULT  (w)
    IMPLICIT NONE        ! Four of the U,V formulas have
INTEGER :: p,q           ! the factor SQRT(h), where
        INTEGER :: a,b   ! h = b*(p-b+1)*(q+b+1)/[(a+b)(a+b+1)]
        REAL(8) :: w     ! w = h
    w = b*(p-b+1._8)*(q+b+1._8)/((a+b)*(a+b+1._8))
    END FUNCTION
! end-of-file
```

Acknowledgments

Funding: This research received no external funding.
Conflicts of Interest: The author declares that he has no conflicts of interest.

References

[1] Brian C. Hall, *Lie Groups, Lie Algebras, and Representations*, 2nd ed.; Springer: Champlain, N.Y., USA, 2015.

[2] Pfeifer, W. *The Lie Algebras su(N)*, 1st ed.; Birkhäuser Verlag: Basel, Switzerland, 2003.

[3] Greiner, W.; Müller, B., *Quantum Mechanics, Symmetries*, 2nd ed.; Springer-Verlag, Berlin, Germany, 1994.

[4] Zee, A. *Group Theory in a Nutshell for Physicists*; Princeton University Press: Princeton, NJ, USA, 2016.

[5] Georgi, S.H., *Lie Algebras in Particle Physics*; CRC Press: Boca Raton, Florida, USA 1999.

[6] Weinberg, S., *The Quantum Theory of Fields, Vol. I*; Cambridge University Press: Cambridge, UK, 1995; pp. 229-233.

[7] Gasiorowicz, S., *Elementary Particle Physics*; John Wiley & Sons: New York, USA 1966); pp. 257-275.

[8] Berganholi, B.; Dorsch, G.C.; Sena, B.M.D.; Valle, G.F. do. Symmetries in particle physics: from nuclear isospin to the quark model. *Eur. J. Phys.* **2024**, *45(6)*, 065402.

[9] Fuchs, J.; Schweigert, C., *Symmetries, Lie Algebras and Representations*, 1st ed.; Cambridge University Press: Cambridge, UK, 1997; especially Chapter 3.

[10] Bonatsos, D.; Assimakis, I.E.; Minkov, N.; Martinou, A.; Cakirli, R.B.; Casten, R.F.; Blaum, K. Proxy-SU(3) symmetry in heavy deformed nuclei. *Phys. Rev. C* **2017**, *95(6)*, 064325.

[11] Elliott, J.P. *The nuclear shell model and its relation with other nuclear models*. In *Selected Topics in Nuclear Theory*; Editor F. Janouch; International Atomic Energy Agency: Vienna, Austria, 1963; 157-208.

[12] Fradkin, D.M. Three-Dimensional Isotropic Harmonic Oscillator and SU(3). *Am. J. Phys.* **1965**; *33(3)*, 207–211.

[13] Shurtleff, R. Formulas for SU(3) Matrix Generators. online: viXra:2409.0060, the omnibus presentation, 2025.

[14] Gell-Mann, M.; Ne'eman, Y. *The Eightfold Way*, 1st ed.; Benjamin: New York, USA 2003.

[15] Edmonds, A.R. *Angular Momentum in Quantum Mechanics*, 2nd ed.; Princeton University Press: Princeton, N.J., USA 1960.

[16] Rose, M.E. *Elementary Theory of Angular Momentum*, 1st ed.; John Wiley & Sons: New York, NY, USA, 1957.

[17] Shurtleff, R. The Fortran program; online, `https://www.dropbox.com/scl/fi/0a1j2fh4mw6dsbiipytc6/SL3C_SU3pqSHORT.f90?rlkey=s9ifpokx02blvsh67uhpuhb3i&dl=0` (accessed on 13 November 2025).

[18] The author ran a Gfortran compiler, the Gnu compiler collection (gcc), on Code::Blocks 20.03, available at `https://gcc.gnu.org/fortran/` and `http://www.codeblocks.org` .