Invariant analysis of CP violation

F. del Aguila\textsuperscript{a}, J.A. Aguilar-Saavedra\textsuperscript{b}, M. Zralek\textsuperscript{a}

\textsuperscript{a} Departamento de Física Teórica y del Cosmos, Universidad de Granada, 18071 Granada, Spain
\textsuperscript{b} Field Theory and Particle Physics Dept., University of Silesia, 40-007 Katowice, Poland

Received 22 July 1996; revised 6 November 1996

Abstract

The invariant formulation of CP violation involves the generation of sets of invariant constraints for CP conservation, the manipulation of their expressions and the identification of complete and minimal subsets of such constraints. In this paper we present a collection of subroutines to deal with these three tasks in a fast, reliable and systematic way, with examples for the leptonic sector.

Keywords: CP violation; Quark and lepton masses and mixings

PACS: 11.30.Er; 12.15.Ff; 14.60.Pq; 14.80.-j

1. Introduction

The observed CP violation in the $K^0-\bar{K}^0$ system \cite{1} is related to the presence of complex phases in the mixing matrix describing the quark gauge couplings in the mass-eigenstate basis in the Standard Model (SM) \cite{2}. However, not all the phases in the mixing matrices are CP violating. Some of them can be eliminated redefining the fermion phases. In the SM with three quark generations, the 6 phases of the $3 \times 3$ unitary mixing matrix $V$ reduce to 1 after an appropriate field redefinition. For this case we can identify a quantity invariant under weak quark basis transformations, the determinant of the commutator of the products of the up and down quark mass matrices $M_u$ and $M_d$ times their Hermitian conjugate, respectively, whose vanishing characterizes CP conservation \cite{3},

$$
det(M_u M_u^\dagger, M_d M_d^\dagger) = -2i(m_t^2 - m_c^2)(m_t^2 - m_u^2)(m_c^2 - m_u^2)(m_b^2 - m_s^2)$$

$$
\times (m_b^2 - m_s^2)(m_s^2 - m_d^2) \text{Im}(V_{ud} V_{cd}^* V_{us} V_{us}^*) = 0, \tag{1}
$$

where $m_i$ is the mass of the quark $i$ and $V_{ij}$ is the $ij$ entry of the Cabibbo–Kobayashi–Maskawa (CKM) matrix \cite{2}. This invariant formulation of CP violation makes apparent the necessary and sufficient conditions for CP conservation, giving the size of CP violation. It also allows to decide if CP is conserved in any weak basis, motivating model building and helping to understand eventually the origin of CP violation if a definite model (weak basis) is physically distinguished.

In general, i.e. in extended models with extra generations and/or vector-like quarks and/or right-handed (Majorana) neutrinos, it is difficult to find such a minimal set of necessary and sufficient invariant conditions.

0010-4655/97/$17.00$ Copyright © 1997 Elsevier Science B.V. All rights reserved.

PH S0010-4655(96)00159-2
In this paper we present a collection of subroutines for Mathematica [4] which carries out this task in a fast and systematic way. They are used (i) to calculate products of mass (sub)matrices giving rise to expressions invariant under weak fermion basis transformations, (ii) to do the symbolic manipulations and to solve the constraints resulting from equating these invariants to zero, and (iii) to verify that the set of necessary conditions for CP conservation generated in this way is sufficient and minimal (this is to say that this set does not contain any smaller subset of sufficient conditions). Although these subroutines are written for the leptons including Majorana neutrinos, they can be easily adapted to analyze CP violation in the quark sector [5,6]. They can also be generalized to study extended gauge theories as left-right models, and/or (supersymmetric) scalar gauge couplings. On the other hand, the computer calculations use to be more reliable than the manual ones. For example, we have revised the two more complicated cases studied in Ref. [7]. In both cases we find CP violating solutions which were overlooked there. In Section 2 we comment on the main theoretical features and in Section 3 we describe the subroutines and apply them to these two cases.

2. Theory and general features

Let us assume the SM with \( n_L \) left-handed and \( n_R \) right-handed neutrinos and define

\[
\psi_L = \begin{pmatrix} \nu_L \\ (\nu^c)_L \end{pmatrix}, \quad \psi_R = \begin{pmatrix} (\nu^c)_R \\ \nu_R \end{pmatrix},
\]

where \( \nu^c = C\bar{\nu}^T \), with \( C \) the Dirac charge conjugation matrix, and \( \psi_{L,R} \) are \( n_L + n_R \) component vectors in flavour space describing the \( n_L \) left-handed and \( n_R \) right-handed neutrinos. Then the mass term reads

\[
-L_{\text{mass}} = \bar{\psi}_L M_L \psi_R + \frac{1}{2} \bar{\psi}_L M_A \psi_R + \text{h.c.},
\]

where \( I_{L,R} \) are \( n_L \) component vectors describing the \( n_L \) charged leptons and \( M_I \) an arbitrary \( n_L \times n_L \) complex matrix, whereas

\[
M_\nu = \frac{1}{n_L} \begin{pmatrix} M_L & M_D \\ M_D^T & M_R \end{pmatrix}
\]

is an arbitrary \( (n_L + n_R) \times (n_L + n_R) \) complex symmetric matrix. This Lagrangian is invariant under a CP transformation leaving the gauge SU(2)\(_L\) \( \times U(1)_Y \) interactions unchanged [8],

\[
I_L \rightarrow U_L C \nu_L^*, \quad I_R \rightarrow U_R C \nu_R^*, \quad \nu_L \rightarrow U_L C \nu_L^*, \quad \nu_R \rightarrow U_R C \nu_R^*,
\]

where \( U_L, U_R^T \) are \( n_L \times n_L \) \( n_R \times n_R \) unitary matrices acting on flavour indices, if

\[
U_L^T M_L U_R^* = M_L^*, \quad U_L^T M_D U_R^* = M_D^*, \quad U_R^T M_R U_R^* = M_R^*.
\]

Eqs. (6) are necessary and sufficient conditions for CP conservation. From these equations, one can write necessary conditions for CP conservation which do not require the knowledge of the unitary matrices involved in the definition of the CP transformation. Thus, the products of mass matrices \( M_I, M_L, M_D, M_R \) are classified in three classes \( G_1 \equiv G_L, G_2 \equiv G_R, G_3 \equiv G_R^* \), depending under which unitary matrix \( (U_L, U_R^T, U_R^T) \) (respectively) transform,

\[
G_1 = \{ A_{L1} \equiv M_I M_L^*, A_{L2} \equiv M_L M_I^*, A_{L3} \equiv M_D M_D^*, A_{LI} A_{LJ}, M_L M_I^* M_L^*, M_D M_D^* M_D^*, M_R M_R^* M_R^* \},
\]

\[
G_2 = \{ A_1 \equiv M_I^* M_I, A_2^* M_L^* M_I, A_3 M_I^* M_D^* M_I, M_L^* M_D^* M_L^* \}.
\]
G_3 = \{A_{ij} \equiv M_D^i M_D^j, A_{ij} \equiv M_M^i M_M^j, M_D^i M_M^j M_D^k M_D^j, M_M^i M_D^j M_M^k M_M^j, M_M^i M_M^j M_D^k M_D^j, M_M^i M_M^j M_M^k M_M^j, M_D^i M_M^j M_D^k M_D^j, M_M^i M_M^j M_D^k M_D^j, \ldots \}.

The trace and determinant of any of such products or their sums within the same class do not change under unitary transformations due to their cyclic and factorization properties, respectively. Then CP conservation (Eq. (6)) implies that the trace and the determinant of any linear combination \( g \) of elements of the same class is real,

\[ \text{Im} \, \text{tr} \, g = 0, \quad \text{Im} \, \text{det} \, g = 0. \] (8)

To find which subsets of these conditions are also sufficient has to be worked out case by case. In practice, we are interested in identifying minimal subsets with these properties. We proceed as follows.

2.1. Generation of invariance products with the same transformation properties

The elements of \( G_i \) in Eq. (7) must be generated using the mass matrices

\[
\{M_M^i, M_M^j, M_M^i M_M^j, M_M^i M_D^j, M_M^i M_M^j M_M^k M_M^j, M_M^i M_D^j M_M^k M_M^j, M_M^i M_M^j M_M^k M_M^j, \ldots \}.
\] (9)

(Note that \( M_{L,R} \) are symmetric.) A sequence \( S \) of order \( n \) is an allowed product of \( n \) of such matrices. A product of two sequences \( S_a S_b \) is allowed if \( S_a' = V_a S_a V_a^\dagger \), \( S_b' = V_b S_b V_b^\dagger \) and \( V_b' = V_b \). Hence the sequences of order 2 are

\[
\{ M_M^i M_M^j, M_M^i M_D^j, M_M^i M_M^j M_M^k M_M^j, M_M^i M_D^j M_M^k M_M^j, M_M^i M_M^j M_M^k M_M^j, M_M^i M_M^j M_M^k M_M^j, \ldots \}.
\] (10)

A sequence \( S \), with \( S' = VSV^\dagger \), is an element of \( G_{1,2,3} \) in Eq. (7) if \( V = V' = U_L, U_R, V' \), respectively. Then Eq. (6) implies that the order of the elements of \( G_i \) is even and that the sequences in \( G_i \) can be constructed with the biproducts in Eq. (10). These elements are generated order by order. The elements of order \( n \) are the allowed products of the sequences of order \( n - 2 \) which begin with \( M_M, M_L \) or \( M_D \) for \( G_1 \), \( M_M^i \) for \( G_2 \) and \( M_M^j \) or \( M_D \) for \( G_3 \) (see Eq. (6)), times the biproducts in Eq. (10) which end with \( M_M^i, M_L^i \) or \( M_D^i \) for \( G_1 \), \( M_M \) for \( G_2 \) and \( M_D \) or \( M_R \) for \( G_3 \).

2.2. Solution of invariant constraints

Once the elements of \( G_i \) are generated, we have to solve the nontrivial conditions in Eq. (8) up to a given order. This means to find the relations among the parameters fixing \( M_{l,L,D,R} \) implied by these conditions. The order is increased until there is no CP violating solution. In the examples discussed later we need to consider conditions involving sequences of order 4, 6 and 8. These constraints are easier to solve if the mass matrices are conveniently parametrized. Under a change of weak basis,

\[
M_{L}^t = W_L M_L W_R^\dagger, \quad M_{R}^t = W_L M_R W_R^\dagger, \quad M_{D}^t = W_L M_D W_R^\dagger, \quad M_R^t = W_R^\dagger M_R W_R^\dagger.
\] (11)

Hence, as the constraints do not depend on the choice of weak basis, we can choose the unitary matrices \( W_L, W_R \) appropriately and assume \( M_L \) and \( M_R \) diagonal with nonnegative real elements, whereas \( M_M \) is complex and symmetric and \( M_D \) complex and arbitrary. The number and difficulty of the equations grow with the order.
of the sequences. Generically, \textit{Mathematica} uses too much memory and time to solve simultaneously more than 5 conditions. In this case one solves the equations of lowest order first and inserts the solutions in the remaining equations, which can then be solved more easily.

2.3. Minimal subset of invariant constraints

The last subroutine verifies if a solution of a set of conditions in Eq. (8) is CP conserving. A solution \( s \) is a set of relations among the parameters fixing the mass matrices \( M_{1,2,3,4} \). Let \( M^{(s)}_{1,2,3,4} \) be the mass matrices satisfying these relations. We generate random numbers for their independent parameters denoting the corresponding mass matrices \( M^{(s)}_{1,2,3,4} \). These matrices conserve CP if there exists a unitary matrix \( U^{(s)} \) and such that

\[
U^{(s)}M^{(s)}U^{(s)T} = M^{(s)*}.
\]  

If this matrix can be constructed, we say that \( s \) conserves CP. It is highly improbable that there is a relation among the random parameters in \( M^{(s)}_{1,2,3,4} \) such that the matrices \( M^{(s)}_{1,2,3,4} \) violate CP but \( M^{(s)}_{1,2,3,4} \) do not. On the other hand, if Eq. (12) is not satisfied, the solution \( s \) is said to violate CP.

Let us drop the superscripts from now on. The existence of \( U \) is determined in two steps. Step 1 guarantees that if \( U \) exists, it is diagonal, and step 2 checks if there is a diagonal unitary matrix satisfying Eq. (12). In the convenient basis and for a given solution \( s \), \( U \) is block diagonal with diagonal submatrices of dimension the multiplicity \( m_i \) and \( \mu_i \) of the eigenvalues of \( M_i \) and \( M_R \), respectively,

\[
U = \begin{pmatrix}
(m_1 \times m_1) & & \\
& (m_2 \times m_2) & \\
& & \ddots \\
& & & (\mu_1 \times \mu_1) \\
& & & (\mu_2 \times \mu_2) \\
& & & & \ddots
\end{pmatrix}, \tag{13}
\]

with \( \sum m_i = n_L, \sum \mu_i = n_R \).

If there is no degeneracy, \( U \) is diagonal and we go to step 2: Eq. (12) is fulfilled if \( \forall i > j, \)

\[
\Delta_{ij} = \arg M_{ei} - \arg M_{ej} - \arg M_{ji} - \arg M_{ji} = 0, \pi. \tag{14}
\]

If any \( M_{ij} = 0, \Delta_{ij} \) is zero for \( \arg M_{ij} (= \arg M_{ji}) \) is conventionally defined by Eq. (14). On the other hand, if some diagonal elements \( M_{ii} = \cdots = M_{kk} = 0 \), we consider \( \Delta_k = \arg M_{ii}, \ldots, \arg M_{kk} \) arbitrary but fixed. We regard Eq. (14) as a system of equations in the variables \( \delta_i \equiv \arg M_{ii} \). Then CP is conserved if and only if the full system of equations (14) is compatible.

If there is degeneracy we can fix the weak basis to ensure that \( U \) is diagonal and then go to step 2. This involves the most delicate casuistry, and it has to be done and programmed for each dimension \( n_L, n_R \). For small dimensions it can be easier to do it by hand. Let us discuss the simplest case: \( n_L = 1, n_R = 2 \), and the \( 2 \times 2 \) matrix \( M_R \) diagonal, real, nonnegative and degenerate. Then \( U \) has the general form

\[
U = \begin{pmatrix}
e^{-i\delta} & 0 \\
0 & U_R^T
\end{pmatrix}. \tag{15}
\]

- If \( M_R \) is identically zero, \( U_R \) is an arbitrary unitary matrix. There are different ways to fix the weak basis and guarantee that \( U_R \) is diagonal. We choose to diagonalize \( M^T_D M_D \).
- If \( M^T_D M_D \) is nondegenerate, we transform \( M \) accordingly and go to step 2.
If $M^T_D M_D$ is degenerate, it is identically zero because $M_D$ is a $1 \times 2$ matrix. Then $M_D$ is also identically zero and there is an extra flavour symmetry, and Eq. (12) is satisfied taking $U_R$ equal to the identity.

- If $M_R$ is nonzero, $U_R$ is a real orthogonal matrix. In this case we choose to diagonalize $(\text{Im } M_D)^T(\text{Im } M_D)$ or $(\text{Re } M_D)^T(\text{Re } M_D)$ if the former is degenerate.

- If one of them is non-degenerate, we transform $M$ accordingly and go to step 2. (For convenience only, if $M_L = 0$ we also require $M_D M_D^T$ to be real and positive. This fixes $\delta_L$. If $M_D M_D^T = 0$, there is a flavour symmetry.)

- If both are degenerate, as above they are identically zero. $M_D$ is also zero and there is again an extra flavour symmetry. Similarly, Eq. (12) is fulfilled with $U_R$ equal to the identity.

3. Examples

In this section we present a description of the different subroutines in the package CPEP. Some of them require the subroutines DiagonalizeH and DiagonalizeS defined in the package Diagonal [9]. DiagonalizeH is a subroutine which diagonalizes Hermitian matrices, returning orthonormal eigenvectors even in the case of degenerate eigenvalues. DiagonalizeS diagonalizes general complex symmetric matrices $M$, with a congruent transformation $M \rightarrow U M U^T = D$. Let us show how CPEP works in two examples. (Definitions are given in the appendix.)

3.1. Case $n_L = 1$, $n_R = 2$

The list

$$\{a[10], a[11], a[12], a[13], a[20], a[23], a[30], a[31], a[32], a[33], a[40], a[43]\}$$

is the computer representation of the mass matrices in Eq. (9). We define an adequate output format for $a[i]$ so that the expressions can be more easily understood. The rules for the generation of invariants are embodied in the definition of an associative function $f$, with an arbitrary number of arguments $a[i_1], \ldots, a[i_m]$ ordered as the corresponding matrix product $a[i_1] \cdots a[i_m]$. $f$ is taken to be zero if any of the products $a[i_1] a[i_2], \ldots, a[i_{m-1}] a[i_m]$ is not allowed (see Eq. (10)). Then the function inv checks if the product $f[a[i_1], \ldots, a[i_m]]$ transforms with a matrix and its adjoint, and returns again its argument if it does or 0 if it does not. The lists obtained in this way are multiplied with Outer.

```plaintext
<<cpep.m;
12[1]=Union[Flatten[Outer[f,2init,11]]];
13[1]=Union[Flatten[Outer[f,2init,11]]];
...;
16[3]=Union[Flatten[Outer[f,16[3],11]]];

i4[1]=Union[inv[i4[1]]];
i4[2]=Union[inv[i4[2]]];
i4[3]=Union[inv[i4[3]]];
i6[1]=Union[inv[i6[1]]];
```

\footnote{The built-in function Eigensystem may give nonorthogonal eigenvectors.}
In this example it is necessary to consider only sequences up to order 6. In order to obtain the invariant constraints for $nt = 1$, $n_g = 2$, we introduce the convenient parametrization for $M_L$, $M_M$, $M_D$, $M_R$ and define $M_n$, the neutrino mass matrix in Eq. (4). Random numerical values for the parameters are also generated for later use.

\[
M_1 = \{\{e_1\}\}; \quad M_L = \{\{n_1\}\}; \quad M_D = \{\{Re[n_2]+I Im[n_2], Re[n_3]+I Im[n_3]\}\}; \\
M_R = \{\{n_4, 0\}, \{0, n_5\}\}; \quad M_n = Transpose[Join[Transpose[Join[ML, \\
      Transpose[MD]], Transpose[Join[MD, MR]]]]];
\]

\[
ev = \{e_1 -> Random[], n_1 -> Random[], Re[n_2] -> Random[Real, \{-1, 1\}], \\
Im[n_2] -> Random[Real, \{-1, 1\}], Re[n_3] -> Random[Real, \{-1, 1\}], \\
Im[n_3] -> Random[Real, \{-1, 1\}], n_4 -> Random[], n_5 -> Random[]\}
\]

Now to identify those constraints in Eq. (8) which are not identically zero, we define the functions Looktrace and Lookdet. Their first argument $I$ is a list of sequences, whereas the other arguments are the matrices with random entries. These are required to check if the constraint is identically zero. Looktrace looks for the elements $g$ of $I$ which satisfy $\text{Im} \ 	ext{tr} \ g \neq 0$. Lookdet first finds all non-Hermitian combinations $g - g'$ and then returns those with $\text{Im} \ \text{det}(g - g') \neq 0$. Finally, Newecs calculates the nontrivial constraints returned by Looktrace and Lookdet as a function of the convenient parameters (see the appendix for a detailed description of its arguments). Thus

\[
\text{Looktrace}[i4[1], ML/. ev1, ML/. ev1, MD/. ev1, MR/. ev1]; \\
\text{Newecs}[][[], %, ML, MD, MR, \{e_1, n_1, n_2, n_3, n_4, n_5\}, \{n_2, n_3\}]
\]

returns

\[
\text{Im} \ \text{tr}(M_DM_D^TM_D^TM_L^TM_L^T) = 2n_1(n_4 \text{ Im} n_2 \text{ Re} n_2 + n_5 \text{ Im} n_3 \text{ Re} n_3).
\]

The traces of the other elements of $i4[1]$, $i4[2]$ and $i4[3]$ are proportional to the trace in Eq. (16). Similarly,

\[
\text{Looktrace}[i6[1], ML/. ev1, ML/. ev1, MD/. ev1, MR/. ev1]; \\
\text{Newecs}[ecs, %, ML, MD, MR, \{e_1, n_1, n_2, n_3, n_4, n_5\}, \{n_2, n_3\}]
\]

\[
\text{AppendTo}[ecs, %[[3, 2]]];
\]

\[
\text{Im} \ \text{tr}(M_DM_D^TM_D^TM_L^TM_L^T) = 2n_1(n_4 \text{ Im} n_2 \text{ Re} n_2 + n_5 \text{ Im} n_3 \text{ Re} n_3).
\]

The traces of the remaining sequences in $i6[1]$, $i6[2]$ and $i6[3]$ do not give new equations. Finally,

\[
\text{Lookdet}[i4[3], ML/. ev1, ML/. ev1, MD/. ev1, MR/. ev1]; \\
\text{Newecs}[ecs, %, ML, MD, MR, \{e_1, n_1, n_2, n_3, n_4, n_5\}, \{n_2, n_3\}, \text{det} -> \text{True}]
\]

\[
\text{AppendTo}[ecs, %[[3, 3]]];
\]

returns

\[
\text{Im} \ \text{det}(M_D^TM_DM_D^TM_D^TM_R^TM_R^T) = 2n_4n_5(n_4 - n_5)(n_4 + n_5) \\
\times (\text{Im} n_2 \text{ Re} n_3 - \text{Im} n_3 \text{ Re} n_2) \\
\times (\text{Im} n_2 \text{ Im} n_3 + \text{ Re} n_2 \text{ Re} n_3).
\]

(18)
When we consider that the set of constraints may be complete, we use Reduce to find all the solutions. In this case we try with Eqs. (16-18) equal to zero. Reduce gives a lot of redundant solutions, often repeated, for instance \( n_1=0 \) \& \( n_4=0 \) and \( n_1=0 \) \& \( n_4=0 \) \& \( n_5=0 \), and some inconsistent solutions, like \( \text{Re}[n_3]=1 \) \( \text{Re}[n_2] \). The function Eliminatesols gets rid of the redundant solutions; whereas the inconsistent solutions are eliminated by inspection.

\[
\text{sol0} = \text{Reduce}[\text{ecs} = \{0, 0, 0\}];
\]

\[
<<\text{Algebra/ReIm.m};
\]

\[
\text{el}/: \text{Positive}[n_1] = \text{True};\text{nl}/: \text{Positive}[n_1] = \text{True};
\]

\[
\text{n4}/: \text{Positive}[n_4] = \text{True};\text{n5}/: \text{Positive}[n_5] = \text{True};
\]

\[
\text{sol1} = \text{Eliminatesols}[\text{sol0}/._!=0->\text{True}];
\]

\[
\text{sol2} = \text{sol1}/._{\{\text{Re}[\_] == \text{Complex}[\_, \_]\times/;\text{Im}[x] != 0\}}->\text{False},
\]

\[
(\text{Im}[\_] == \text{Complex}[\_, \_]\times/;\text{Im}[x] == 0) -> \text{False};
\]

In this case, Reduce gives us 198 solutions (sol0). Many of them are redundant and we need to keep only 32 (sol1). 6 of them are inconsistent and are eliminated in sol2. To verify whether the former set is complete, we use the subroutines Red12 and LookCP. Red12 writes the mass matrix Mn in the basis in which \( U \) is diagonal, if it exists. LookCP then returns True or False depending on whether \( U \) exists.

Although the whole process has been described in the previous section, we want to point out a little trick used in LookCP. When trying to solve the system of equations \( \Delta_{ij} = 0, \pi \) in Eq. (14), one could consider instead to solve the system of equations \( \sin \Delta_{ij} = 0 \). However, Mathematica does not give correct results in this case. For this reason we look for a subset of linearly independent equations in Eq. (14), and solve these equations equated to zero (it can be done always if the equations are independent, because we can always redefine the variables \( \delta_i = \text{arg} M_{\mu} \) conveniently). Then the solution is substituted in the complete set of equations, checking whether they are equal to 0, \( \pi \).

We check if any of the 26 solutions in sol2 violates CP using a loop. It is initialized with

\[
\text{m[number]} : = \text{Mn}/.\text{ToRules[sol2[[number]]]}/.\text{ev1};
\]

\[
i = 0;
\]

and for each solution we run

\[
++i;\text{LookCP[Red12[m[i]]]}
\]

The result is that none of the solutions in sol2 violates CP. So the set of invariant constraints is complete. We could try to take away any of the equations in ecs and repeat the same process, but we would find CP violating solutions. So the set is also minimal.

3.2. Case \( n_L = 2, n_R = 1 \)

This case is more difficult to solve because more invariant constraints are necessary. As we pointed out in Section 2, the computer memory required to solve all the equations simultaneously is too big, so we have to use a different approach. First we solve the condition

\[
\text{Im} \text{det}(M_L M_L^T M_L^T - M_L M_L^T M_L^T M_L^T M_L^T) = 0.
\]

Then for each solution of Eq. (19) we proceed as in the former example but restricting the general form of the mass matrices to fulfill this particular solution. For this case we have found that

\[
\text{Im} \text{tr}(M_D M_L M_D^T M_R) = 0,
\]

\[
\text{Im} \text{tr}(M_R M_D^T M_L^T M_L M_D^T M_D) = 0.
\]
\[
\begin{align*}
\text{Im tr}(M_L M_L^\dagger M_L M_L^\dagger M_L^\dagger M_D^\dagger M_R M_R^\dagger) &= 0, \\
\text{Im det}(M_L M_L^\dagger M_L M_L^\dagger - M_L M_L^\dagger M_L^\dagger M_L^\dagger) &= 0, \\
\text{Im det}(M_D M_D M_L M_L^\dagger - M_L M_L^\dagger M_D M_D^\dagger) &= 0, \\
\text{Im det}(M_D M_D^\dagger M_L M_L^\dagger - M_L M_L^\dagger M_D^\dagger M_D^\dagger) &= 0, \\
\text{Im det}(M_D M_D^\dagger M_L M_L^\dagger - M_L M_L^\dagger M_D^\dagger M_D^\dagger) &= 0, \\
\text{Im det}(M_L M_D^\dagger M_R M_R^\dagger - M_L M_D^\dagger M_R M_R^\dagger) &= 0.
\end{align*}
\]

form a complete set of constraints. This set is also minimal by construction, as long as Eq. (19) is included.

Acknowledgements

We thank G. Branco for discussions. This work was partially supported by CICYT under contract AEN94-0936, by the Junta de Andalucía and by the European Union under contract CHRX-CT92-0004.

Appendix A

(*)
This package is available by anonymous ftp at deneb.ugr.es in directory pub/packages
*)

BeginPackage["CPlep", "Diagon"]

(*)
This package needs the functions DiagonalizeS, DiagonalizeH defined in the package Diagon, available by anonymous ftp at deneb.ugr.es

DiagonalizeS[m] gives a list \{v,u\} where m is a square matrix, v is a row vector and u is a matrix fulfilling
u.m.Transpose[u]=DiagonalMatrix[v]

DiagonalizeH[m] is a modification of Eigensystem which gives a list \{v,u\}, with the eigenvalues and eigenvectors of the Hermitian matrix m, such that
u.m.Transpose[Conjugate[u]]=DiagonalMatrix[v]. The list of eigenvalues v is ordered by increasing absolute value
*)

c1lini::usage =
"c1lini is a list with the possible first elements of a product in the class GI"
c12ini::usage =
"cl2ini is a list with the possible first elements of a product in the class G2"
cl3ini::usage = "cl3ini is a list with the possible first elements of a product in the class G3"
cl1fin::usage = "cl1fin is a list with the possible last elements of a product in the class G1"
cl2fin::usage = "cl2fin is a list with the possible last elements of a product in the class G2"
cl3fin::usage = "cl3fin is a list with the possible last elements of a product in the class G3"
l1::usage = "l1 is a list with all the possible elements of a product"
f::usage = "f[expr] is 0 if expr is not an allowed product"
inv::usage = "inv[expr] returns expr if it is an invariant product, or 0 if not"
Looktrace::usage = "Looktrace[lf,MI,ML,MD,MR] returns the elements in a list lf of f-products with trace nonzero when evaluated with the numerical matrices MI,ML,MD,MR. The output is in the form of Dot products"
Lookdet::usage = "Lookdet[lf,MI,ML,MD,MR] calculates all possible non-Hermitian combinations x-y of elements of a list lf of f-products and then returns those with nonzero determinant when evaluated with the numerical matrices MI,ML,MD,MR. The output is in the form of Dot products"
Newecs::usage = "Newecs[oldecs,l1d,MI,ML,MD,MR,vars,comp] looks for equations not present in oldecs, generated with a list l1d of Dot products and the matrices MI,ML,MD,MR with variables vars, of which comp are complex. It returns a list containing four lists: the list oldecs, possibly improved with simpler equations, the list of elements of l1d that improve oldecs, a list of new equations not present in oldecs and a list of elements of l1d that generate the new equations"
det::usage = "det is an option for Newecs with default value False which select to calculate the trace or the determinant"
Eliminatesols::usage = "Eliminatesols[oldecs,l1d,MI,ML,MD,MR,vars,comp] look to detect solutions in sel"
Red21[N,Me] reduces a neutrino mass matrix M in the case nL=2, nR=1 by changes of basis to a form in which CP is a diagonal matrix if it exists. Me is the charged lepton mass matrix

LookCP::usage = "LookCP[M] tests if a diagonal unitary matrix U exists such that U.M.Transpose[U]=Conjugate[M] and yields True or False"

Begin["'Private'"]

(* Definitions for internal use only *)

\[ No comments needed *)

trace[x_?MatrixQ] := Sum[x[[i,i]], {i, Length[x]}]/;Length[x]==Length[Transpose[x]];

Adj[m_] := Transpose[Conjugate[m]]

(* Enlarge the dimension of a matrix with the identity *)

IncDim1[m_] := Transpose[Prepend[Transpose[Prepend[m, Table[0, {Length[m]}]]], Join[{1}, Table[0, {Length[m]}]]]]

IncDim[m_, n_] := Nest[IncDim1, m, n]

(* adjoint of a matrix *)

ad1[a[x_]] := a[10 Floor[x/10] + Sqrt[a[x]]];

(* Adjoint of an f-product *)

ad[f[x__]] := Apply[f, Reverse[Thread[ad1[Apply[ff, f[x]]], ff]]];
ad[0] = 0;

(* Take Hermitian elements of a list *)

auto[1_List] := Block[{lt},
  For[lt = {}; n = 1, n <= Length[1], n++, If[1[[n]] == ad1[1[[n]]],
    AppendTo[lt, 1[[n]]]; lt]];

(* Take non-Hermitian elements of a list *)
symbolic expressions of the mass matrices and calculate the
imaginary part of the trace/determinant *)

\[ \text{Imtr}[\text{expr}, x, y, u, v, \text{comp}] := \text{Block}[[\text{sust}], \{
\text{sust} = \{a[10] \rightarrow x, a[20] \rightarrow y, a[30] \rightarrow u, a[40] \rightarrow v,
    a[13] \rightarrow \text{Adj}[x], a[23] \rightarrow \text{Adj}[y], a[33] \rightarrow \text{Adj}[u], a[43] \rightarrow \text{Adj}[v],
    a[11] \rightarrow \text{Transpose}[x], a[31] \rightarrow \text{Transpose}[u],
    a[12] \rightarrow \text{Conjugate}[x], a[32] \rightarrow \text{Conjugate}[u]\};
\text{Factor}[\text{ComplexExpand}[\text{Im}[\text{trace}[\text{expr}/.\text{sust}]], \text{comp}]]\}
\]

\[ \text{Imdet}[\text{expr}, x, y, u, v, \text{comp}] := \text{Block}[[\text{sust}], \{
\text{sust} = \{a[10] \rightarrow x, a[20] \rightarrow y, a[30] \rightarrow u, a[40] \rightarrow v,
    a[13] \rightarrow \text{Adj}[x], a[23] \rightarrow \text{Adj}[y], a[33] \rightarrow \text{Adj}[u], a[43] \rightarrow \text{Adj}[v],
    a[11] \rightarrow \text{Transpose}[x], a[31] \rightarrow \text{Transpose}[u],
    a[12] \rightarrow \text{Conjugate}[x], a[32] \rightarrow \text{Conjugate}[u]\};
\text{Factor}[\text{ComplexExpand}[\text{Im}[\text{Det}[\text{expr}/.\text{sust}]], \text{comp}]]\}
\]

(* minors calculates the set of invariant phases \( \Delta_{ij} \) of a
matrix. Used by LookCP *)

\[ \text{minor}[m, i, j] := m[[i, i]] \cdot m[[j, j]] - m[[i, j]] \cdot m[[j, i]] \]

\[ \text{minors}[m] := \text{Flatten}[\text{Table}[[\text{If}[m[[i, j]] == 0, 0, \text{minor}[m, i, j]],
    \{i, 1, \text{Length}[m]\}, \{j, i + 1, \text{Length}[m]\}]]\]

(* vec and linealindep check if a subset of invariant phases is
linearly independent. Used by LookCP *)

\[ \text{vec}[\text{expr}, d] := \text{Table}[\text{Coefficient}[\text{expr}, \text{delta}[i]], \{i, d\}]\]

\[ \text{linealindep}[\text{exprs}, d] := \text{MemberQ}[\text{Complement}[\text{Flatten}[\text{Minors}[\text{Thread}[\text{vec}[\text{exprs}, d]], \text{Length}[\text{exprs}]]], \{0\}], \_]\]

(* Output format with standard notation *)

\[ \text{Format}[f[x, y]] := \{x, y\}; \]
\[ \text{Format}[a[10]] = \text{ColumnForm}[[", "M", " 1"], \text{Left}, \text{Center}]; \]
\[ \text{Format}[a[13]] = \text{ColumnForm}[[", +", "M", " 1"], \text{Left}, \text{Center}]; \]
\[ \text{Format}[a[12]] = \text{ColumnForm}[[", \", "M", " 1"], \text{Left}, \text{Center}]; \]
\[ \text{Format}[a[11]] = \text{ColumnForm}[[" T", "M", " 1"], \text{Left}, \text{Center}]; \]
\[ \text{Format}[a[23]] = \text{ColumnForm}[[", +", "M", " L"], \text{Left}, \text{Center}]; \]
\[ \text{Format}[a[30]] = \text{ColumnForm}[[", +", "M", " D"], \text{Left}, \text{Center}]; \]
\[ \text{Format}[a[33]] = \text{ColumnForm}[[", +", "M", " D"], \text{Left}, \text{Center}]; \]
\[ \text{Format}[a[32]] = \text{ColumnForm}[[", +", "M", " D"], \text{Left}, \text{Center}]; \]
\[ \text{Format}[a[31]] = \text{ColumnForm}[[" T", "M", " D"], \text{Left}, \text{Center}]; \]
\[ \text{Format}[a[40]] = \text{ColumnForm}[[", +", "M", " R"], \text{Left}, \text{Center}]; \]
(* Definition of f: not allowed products are 0 *)

\[
f[a[10], a[x_]] := 0 /; x != 13;
\]

\[
f[a[13], a[x_]] := 0 /; (x == 10 \&\& x != 20 \&\& x == 30);
\]

\[
f[a[12], a[x_]] := 0 /; x == 11;
\]

\[
f[a[11], a[x_]] := 0 /; (x == 12 \&\& x != 23 \&\& x == 32);
\]

\[
f[a[20], a[x_]] := 0 /; (x == 12 \&\& x != 23 \&\& x == 32);
\]

\[
f[a[23], a[x_]] := 0 /; (x == 10 \&\& x != 20 \&\& x == 30);
\]

\[
f[a[30], a[x_]] := 0 /; (x == 33 \&\& x == 43);
\]

\[
f[a[33], a[x_]] := 0 /; (x == 10 \&\& x != 20 \&\& x == 30);
\]

\[
f[a[32], a[x_]] := 0 /; (x == 31 \&\& x == 40);
\]

\[
f[a[31], a[x_]] := 0 /; (x == 12 \&\& x != 23 \&\& x == 32);
\]

\[
f[a[40], a[x_]] := 0 /; (x == 33 \&\& x == 43);
\]

\[
f[a[43], a[x_]] := 0 /; (x == 31 \&\& x == 40);
\]

\[
f[f[x_, y_], z_] := \text{If}[f[y, z] == 0, 0, f[x, y, z], f[x, y, z]];\]

\[
f[x_, f[y_, z_]] := \text{If}[f[x, y] == 0, 0, f[x, y, z], f[x, y, z]];\]

\[
f[f[x_, y_], f[u_, v_]] := \text{If}[f[y, u] == 0, 0, f[x, y, u, v], f[x, y, u, v]];\]

\[
f[0, x_] := 0; f[x_, 0] := 0;
\]

(* Definition of inv *)

\[
inv[f[x_, y_, z_]] := \text{If}[f[z, x] == 0, 0, f[x, y, z], f[x, y, z]];\]

\[
inv[0] := 0;
\]

SetAttributes[inv, Listable];

(* Definitions of lists *)

11 = {a[10], a[13], a[12], a[11], a[20], a[23], a[30], a[33], a[32], a[31], a[40], a[43]};

c11ini = {a[10], a[20], a[30]};

c12ini = {a[13]};

c13ini = {a[33], a[43]};

c11fin = {a[13], a[23], a[33]};

c12fin = {a[10]};

c13fin = {a[30], a[40]};

(* Definitions of Looktrace and Lookdet *)

Looktrace[l_List, x_, y_, u_, v_] := Block[{n, sust, lev, lt, ls},
  ls = Apply[Dot, 1, 1];
    a[31] -> Transpose[u], a[12] -> Conjugate[x], a[32] -> Conjugate[u]};
  lev = ls /. sust;
  For[lt = {}; n = 1, n <= Length[ls], n++,
  ]
If[Chop[trace[lev[[n]]]]!=Conjugate[Chop[trace[lev[[n]]]]], AppendTo[lts,ls[[n]]]]];
lt=];

Lookdet[l_List, x_, y_, u_, v_, sgn_: -1] := Block[{i, cm, sust, cev, it},
  cm = Complement[Union[Flatten[Outer[Plus, Apply[Dot, autono[1], 1]],
    sgn, Apply[Dot, l, 1]]],{0}];
a[13] -> Adj[x], a[23] -> Adj[y], a[33] -> Adj[u], a[43] -> Adj[v],
a[11] -> Transpose[x], a[31] -> Transpose[u],
a[12] -> Conjugate[x], a[32] -> Conjugate[u];
  cm = cm/. sust;
  For[It=; i=1, i<=Length[cm], i++,
    If[Chop[Det[cm[[i]]]]!=Conjugate[Chop[Det[cm[[i]]]]],
      AppendTo[lts,cm[[i]]]];lt=];

(* Definition of Newecs *)

Newecs[oldecs_,list_,x_,y_,u_,v_,vars_,complexvars_,ops___]:= Block[{ecs2,ecs3,i,j,its,lt,newec,end,what,detyn,trordet,vars2},
  detyn = (det/. {ops})/. Options[Newecs];
  If[detyn, trordet = Imdet[det], trordet = Imtr[det] ];
  trordet[ars___] := Imtr[ars] ;
  ecs2 = oldecs;
  ecs3 = {};
  vars2 = Join[vars, Re[complexvars], Im[complexvars]];
  For[Its=; i=1, i<=Length[list], i++,
    newec = trordet[list[[i]]], x, y, u, v, complexvars];
    For[]=1; end=False; what=0, j=1=Length[ecs2] && ! end, j++,
      If[PolynomialQ[Factor[ecs2[[j]]]/newec], vars2] && ! {NumberQ[Factor[ecs2[[j]]]/newec]},
      what=; end=True] ];
    If[! end, ecs2 = newec; AppendTo[Its, list[[i]]]];lt=];
  For[lt=; i=1, i<=Length[list], i++,
    newec = trordet[list[[i]]], x, y, u, v, complexvars];
    For[]=1; end=False, j=1=Length[ecs2] && ! end, j++,
      If[PolynomialQ[Factor[newec/ecs2[[j]]]], vars2], end=True] ];
    If[! end, AppendTo[ecs3, newec]; AppendTo[lt, list[[i]]]];lt=];
  {ecs2, Its, ecs3, lt} ];

Options[Newecs] = {det -> False}

(* Definition of Eliminatesols *)

Eliminatesols[sols_] := Block[{i, j, sols2, inter},
  sols2 = Union[sols];
  For[i=1, i<=Length[sols2], i++, If[Head[sols2[[i]]] == And,
    sols2[[i]]=And[sols2[[i]], sols2[[i] ]];
    sols2[[i]]=And[sols2[[i]], sols2[[i]] ]];];
i=1;
While[i<Length[sols2],
    j=i+1;
    While[j<=Length[sols2],
        inter=Intersection[sols2[[i]],sols2[[j]]];
        If[inter==Union[sols2[[i]]],sols2=Delete[sols2,j],
            If[inter==Union[sols2[[j]]],
                sols2=Delete[sols2,i];j=i+1,,j++ ] ] ];
    i++)
For[i=1,i<=Length[sols2],i++,sols2[[i]]=Union[sols2[[i]] ]];
sols2 ]);

(* Definitions of Red12 and Red21 *)
Red12[m0_]:=Bloch[{lu,m,m12,m21,m22,w2,w,delta,n},
    lu={};
    m=m0;
    m12:={{m[[1,2]],m[[1,3]]}};
    m21:=Transpose[m12];
    m22:={{m[[2,2]],m[[2,3]]},{m[[3,2]],m[[3,3]]}};
    w2=DiagonalizeS[m22][[2]];
    w=IncDim1[w2];
    m=Chop[w.m.Transpose[w]];
    If[Chop[m[[2,2]]-m[[3,3]]]==0,,
        If[m12=={{0,0}},
            AppendTo[lu,2];AppendTo[lu,3],
        (* else *)
        If[m22=={{0,0},{0,0}},
            w2=DiagonalizeH[m21.Adj[m21]][[2]];
            w=IncDim1[w2];
            m=Chop[w.m.Transpose[w]],
        (* else *)
        If[m[[1,1]]!=0,
            delta=Arg[m[[1,1]]]/2;
            w=DiagonalMatrix[{Exp[I delta],1,1}];
            m=w.m.Transpose[w],
        (* else *)
        n=(Chop[m12.Transpose[m12]])[[1,1]]; 
        If[n==0,
            AppendTo[lu,1],
        (* else *)
            delta=Arg[n]/2;
            w=DiagonalMatrix[{Exp[I delta],1,1}];
            m=w.m.Transpose[w]]];
    ];
    If[Im[m12]=={{0,0}},
        w2=DiagonalizeH[Re[m21].Transpose[Re[m21]]][[2]];
\begin{verbatim}

w = IncDiml[w2];
m = Chop[w.m.Transpose[w]],
(* else *)
w2 = DiagonalizeH[Im[m21].Transpose[Im[m21]]][[2]];
w = IncDiml[w2];
m = Chop[w.m.Transpose[w]]
]
] (* end m2=0 *)
(* else 1st if *) ];
m
]];

Red21[m_, me_] := Block[{m2},
If[Chop[Abs[me[[1, 1]]] - Abs[me[[2, 2]]]] != 0,
m,
(* else *)
m2 = Reverse[Transpose[Reverse[m]]];
Reverse[Transpose[Reverse[Red21[m2]]]]
]];

(* Definition of LookCP *)

LookCP[m_] := Block[{m2, dim, i, ecsdeg0, ecsdeg1, ecsdeg2, allecs, sol},

m2 = Chop[Arg[m]]

dim = Length[m];
For[i = 1, i <= dim, i++,
If[m[[i, i]] == 0, m2[[i, i]] = delta[i] ] ]

cesdeg0 = Part[m2, Flatten[Position[Apply[Plus, Thread[vec[mins[m2], dim]], i], 0]]];
cesdeg1 = Part[m2, Flatten[Position[Apply[Plus, Thread[vec[mins[m2], dim]], i]], 1]]];
cesdeg2 = Part[m2, Flatten[Position[Apply[Plus, Thread[vec[mins[m2], dim]], i]], 1, 2]]];
allecs = Join[ecsdeg1, ecsdeg2];
If[Complement[Chop[Sin[ecsdeg0]], {0}] == {},
If[allecs == {}, True, (* there aren’t equations *)
For[i = 2; indeps = allecs[[1]], i <= dim && i <= Length[allecs], i++,
If[linealindep Append[indeps, allecs[[i]]], i], dim],
AppendTo[indeps, allecs[[i]] ] ] ]
];
sol = Flatten[Solve[indeps == Table[0, {Length[indeps]}]]
Complement[Chop[Chop[Sin[allecs/.sol]]], {0}] == {}, False]
]

End[]

EndPackage[]
\end{verbatim}
References


Package available at anonymous ftp at ftp://deneb.ugr.es/pub/packages/.