

# MassToMI - a Mathematica package for an automatic Mass Insertion expansion

Janusz Rosiek<sup>a,\*</sup>

<sup>a</sup>*Institute of Theoretical Physics, Physics Department, Warsaw University, Pasteura 5, 02-093  
Warsaw, Poland*

---

## Abstract

We present a Mathematica package designed to automatize the expansion of transition amplitudes calculated in the mass eigenstates basis (i.e. expressed in terms of physical masses and mixing matrices) into series of “mass insertions”, defined as off-diagonal entries of mass matrices in Lagrangian before diagonalization and identification of the physical states. The algorithm implemented in this package is based on the general “Flavor Expansion Theorem” proven in Ref. [1]. The supplied routines are able to automatically analyze the structure of the amplitude, identify the parts which could be expanded and expand them to any required order. They are capable of dealing with amplitudes depending on both scalar or vector (Hermitian) and Dirac or Majorana fermion (complex) mass matrices. The package can be downloaded from the address [www.fuw.edu.pl/masstomi](http://www.fuw.edu.pl/masstomi).

*Keywords:* Mass Insertion Expansion, Flavor Violation, Mass Eigenstates vs. Interaction Basis

---

---

\*Corresponding author. *E-mail address:* [janusz.rosiek@fuw.edu.pl](mailto:janusz.rosiek@fuw.edu.pl)

## PROGRAM SUMMARY

*Manuscript Title:*

MassToMI - a Mathematica package for an automatic Mass Insertion expansion

*Authors:* Janusz Rosiek

*Program Title:* MassToMI v1.0

*Journal Reference:*

*Catalogue identifier:*

*Licensing provisions:* None

*Programming language:* Mathematica 10.2 (earlier versions should work as well)

*Computer:* any running Mathematica

*Operating system:* any running Mathematica

*RAM:* allocated dynamically by Mathematica, at least 4GB total RAM suggested

*Number of processors used:* allocated dynamically by Mathematica

*Supplementary material:* None

*Keywords:* Mass Insertion Expansion, Flavor Violation, Mass Eigenstates vs. Interaction Basis

*Classification:*

- 11.1 General, High Energy Physics and Computing,
- 5 Computer Algebra.

*External routines/libraries:* Wolfram Mathematica program

*Subprograms used:* None

*Nature of problem:*

Automatized expansion of QFT transition amplitude calculated in mass eigenstates basis into power series of off-diagonal elements of mass matrices of the interaction basis Lagrangian.

*Solution method:*

Implementation (as the Mathematica package) of the algebraic algorithm “Flavor Expansion Theorem”, formulated and proven in Ref. [1] given below.

*Restrictions:* None

*Unusual features:* None

*Additional comments:* None

*Running time:* depending on complexity of the analyzed expression, from seconds for simple problems to hours for complicated amplitudes expanded to high order (using Mathematica 10.2 running on a personal computer)

- [1] A. Dedes, M. Paraskevas, J. Rosiek, K. Suxho, K. Tamvakis, Mass Insertions vs. Mass Eigenstates calculations in Flavour Physics, JHEP 1506 (2015) 151 [arXiv:1504.00960 [hep-ph]].

## 1. Introduction

Quantum Field Theory (QFT) models are usually defined by specifying the Lagrangian of the theory. Such definition is not unique, in the sense that it allows for transformations of the field basis, leading to equivalent descriptions of the model, with different degrees of freedom. The calculation of transition amplitudes can be performed in any basis, however two special choices are most convenient from the practical point of view.

In many cases the Lagrangian of the model is initially constructed in terms of fields having definite charges under some symmetry groups - we call it *symmetry* or *interaction basis*. However, in general, fields defined in such a way do not correspond to physical degrees of freedom of the theory, and some redefinitions have to be performed in order to identify the physical fields (with the spontaneous symmetry breaking being the typical example).

Another possibility is to use *mass eigenbasis*, in which bilinear terms (kinetic and mass matrices) in the Lagrangian have been diagonalized and states of the theory correspond to physical particles with definite mass. The transformation from the initial basis to the mass eigenstates basis is performed by unitary rotations (“mixing matrices”) in the field space. Perturbative calculations of the amplitudes in the mass eigenstates basis lead to results expressed in terms of physically measurable quantities, i.e. physical masses and the elements of the mixing matrices. They are usually more compact as compared to those obtained in any other basis and best suited for numerical computations. However, the analytical dependence of such amplitudes on the initial interaction basis is typically complicated and difficult to use for qualitative interpretation.

For the latter purpose, it is often useful to have analytic expressions for the transition amplitudes calculated directly in the interaction basis. They can be obtained using two different methods. Firstly, one can perform an independent diagrammatic calculation of the amplitude using the approximation commonly referred to as the *Mass Insertion Approximation* (MIA) [2, 3, 4]. In this approach, diagonal elements of the mass matrices are absorbed into the definition of (unphysical) massive propagators and the amplitude is, at every loop order, expanded in an infinite series of non-diagonal elements of mass matrices, commonly referred to as *mass insertions* (MI). Alternatively, as proven in Ref. [1], the MIA result can be obtained directly from the mass eigenstates amplitude employing the purely algebraic technique coined in Ref [1] as the “Flavor Expansion Theorem” (FET)<sup>1</sup>. The last method, the FET expansion, has two important advantages. Firstly, it allows to avoid the Feynman diagram calculation with mass insertions, which is usually tedious and prone to errors or omissions of important terms. Secondly, it can become, to large extent, automatized.

In this paper we describe the **MassToMI** package, written with the use of Mathemat-

---

<sup>1</sup>The first non-trivial order of the FET expansion, with applications in MSSM flavor physics, have been presented in refs. [5]. Higher orders could be also obtained using the standard quantum mechanic perturbation theory, applied to mass matrix eigenstates problem (see e.g. [6]), but in this case it is very difficult to get the simple closed expressions.

ica [7] symbolic manipulation language and designed to perform automatically the MI expansion of QFT amplitudes evaluated in mass eigenstates basis, provided that they are coded using a specific format which could be parsed by the **MassToMI** routines. The results are given in terms of MI powers and the so-called *divided differences* [8] of the loop functions. The package is able to expand any type of amplitude for Hermitian (scalar or vector) or general complex (fermion) mass matrices, to any requested MI order.

The paper is organized as follows. In Sec. 2 we formulate the algorithm used for the expansion. In Sec. 3 we present the syntax for defining the amplitudes in the **MassToMI** package, the routines provided for the users and the output format. In Section 4 we illustrate the applications of the **MassToMI** package with several examples, finally we conclude in Sec. 5. The **MassToMI** Mathematica code can be downloaded from the address

[www.fuw.edu.pl/masstomi](http://www.fuw.edu.pl/masstomi)

## 2. Flavor and mass amplitudes in QFT calculations

### 2.1. Mass matrices in physical QFT models

The physical spectrum of any QFT model is defined by the structure of the quadratic terms in the Lagrangian. Assuming that quadratic kinetic (momentum-dependent) terms have been transformed to the canonical form by appropriate field and coupling redefinitions and, if necessary, the Spontaneous Symmetry Breaking (SSB) mechanism has been used to identify physical degrees of freedom, three type of mass matrices can appear in the Lagrangian:

1. Hermitian (squared) mass matrices for scalar and vectors fields,  $\mathbf{M}_S^2 = (\mathbf{M}_S^2)^\dagger$ , diagonalized by a unitary transformation  $\mathbf{Z}$ :

$$\mathbf{Z}^\dagger \mathbf{M}_S^2 \mathbf{Z} = \mathbf{m}_S^2 = \text{diag}(m_1^2, \dots, m_n^2). \quad (2.1)$$

2. General complex mass matrices for Dirac fermions  $\mathbf{M}_D$ , diagonalized by two unitary transformations  $\mathbf{U}, \mathbf{V}$ :

$$\mathbf{V}^\dagger \mathbf{M}_D \mathbf{U} = \mathbf{m}_D = \text{diag}(m_1, \dots, m_n). \quad (2.2)$$

The matrices  $\mathbf{V}$  and  $\mathbf{U}$  diagonalize also the Hermitian matrices  $\mathbf{M}_D \mathbf{M}_D^\dagger$  and  $\mathbf{M}_D^\dagger \mathbf{M}_D$ , through the transformations

$$\mathbf{V}^\dagger \mathbf{M}_D \mathbf{M}_D^\dagger \mathbf{V} = \mathbf{U}^\dagger \mathbf{M}_D^\dagger \mathbf{M}_D \mathbf{U} = \mathbf{m}_D^2. \quad (2.3)$$

3. Symmetric complex mass matrices for Majorana fermions,  $\mathbf{M}_N = \mathbf{M}_N^T$ . In such case one can assume  $\mathbf{U} = \mathbf{V}^* = \mathbf{O}$  in eq. (2.3), so that the mass matrix is diagonalized by a single unitary transformation  $\mathbf{O}$ :

$$\mathbf{O}^T \mathbf{M}_N \mathbf{O} = \mathbf{m}_N = \text{diag}(m_1, \dots, m_n). \quad (2.4)$$

The matrix  $\mathbf{O}$  diagonalizes also the Hermitian matrix  $\mathbf{M}_N^\dagger \mathbf{M}_N$ ,

$$\mathbf{O}^\dagger \mathbf{M}_N^\dagger \mathbf{M}_N \mathbf{O} = \mathbf{m}_N^2. \quad (2.5)$$

Note that the squared mass matrices of physical particles, i.e.  $\mathbf{M}_S^2, \mathbf{M}_D \mathbf{M}_D^\dagger, \mathbf{M}_D^\dagger \mathbf{M}_D$  and  $\mathbf{M}_N^\dagger \mathbf{M}_N$  must be (semi-) positive-definite for well-defined QFT theories.

## 2.2. Structure of the transition amplitudes

Applying the rotations  $\mathbf{Z}, \mathbf{U}, \mathbf{V}, \mathbf{O}$  to the field multiplets one gets the Lagrangian of the theory in the basis of physical (mass eigenstates) fields. The tree level vertices and Feynman rules in such basis depend on the elements of mixing matrices and on the physical particle masses. Consequently, all transition amplitudes (to any loop order) are linear combinations of products of mixing matrices and propagators, or loop integrals being the functions of physical particle masses. The dependence of such amplitudes on the initial Lagrangian parameters is very complicated and non-linear. Thus, as mentioned in the introduction, although they are compact in form and better suited for numerical computations, it is often practical to use the Mass Insertion expansion to recover an approximate, but simpler direct dependence on the interaction basis Lagrangian parameters.

Such an expansion can be done with the use of Flavor Expansion Theorem, formulated and proven in Ref. [1]. As argued in [1], the mixing matrices of the internal particles of Feynman diagrams can appear in amplitudes only in some specific combinations, namely

$$\begin{aligned}
\text{Scalars, vector bosons :} & \quad Z_{Bi} f(m_i^2) Z_{Ai}^* , \\
\text{Dirac fermions :} & \quad U_{Bi} f(m_i^2) U_{Ai}^*, \quad V_{Bi} f(m_i^2) V_{Ai}^* , \\
& \quad U_{Bi} m_i f(m_i^2) V_{Ai}^*, \quad V_{Bi} m_i f(m_i^2) U_{Ai}^* , \\
\text{Majorana fermions :} & \quad O_{Bi} f(m_i^2) O_{Ai}^*, \quad O_{Bi} m_i f(m_i^2) O_{Ai} , \tag{2.6}
\end{aligned}$$

where  $f(m_i^2)$  represents symbolically the dependence of the amplitude on the physical masses, at tree or loop level. In the expression above, and similarly for other equations in the rest of the paper, we assume a summation over the repeating indices (even if they appear more than twice).

Assuming that the function  $f$  is analytical, the combinations listed above can be

formally expressed as matrix elements being functions of the squared mass matrices:

$$\begin{aligned}
Z_{Bi} f(m_i^2) Z_{Ai}^* &= f(\mathbf{M}_S^2)_{BA} , \\
U_{Bi} f(m_i^2) U_{Ai}^* &= f(\mathbf{M}_D^\dagger \mathbf{M}_D)_{BA} , \\
V_{Bi} f(m_i^2) V_{Ai}^* &= f(\mathbf{M}_D \mathbf{M}_D^\dagger)_{BA} , \\
V_{Bi} m_i f(m_i^2) U_{Ai}^* &= \left( \mathbf{M}_D f(\mathbf{M}_D^\dagger \mathbf{M}_D) \right)_{BA} = \left( f(\mathbf{M}_D \mathbf{M}_D^\dagger) \mathbf{M}_D \right)_{BA} , \\
U_{Bi} m_i f(m_i^2) V_{Ai}^* &= \left( \mathbf{M}_D^\dagger f(\mathbf{M}_D \mathbf{M}_D^\dagger) \right)_{BA} = \left( f(\mathbf{M}_D^\dagger \mathbf{M}_D) \mathbf{M}_D^\dagger \right)_{BA} , \\
O_{Bi} f(m_i^2) O_{Ai}^* &= f(\mathbf{M}_N^\dagger \mathbf{M}_N)_{BA} = f(\mathbf{M}_N \mathbf{M}_N^\dagger)_{AB} , \\
O_{Bi} m_i f(m_i^2) O_{Ai}^* &= \left( \mathbf{M}_N^\dagger f(\mathbf{M}_N \mathbf{M}_N^\dagger) \right)_{BA} = \left( f(\mathbf{M}_N^\dagger \mathbf{M}_N) \mathbf{M}_N^\dagger \right)_{BA} , \\
O_{Bi}^* m_i f(m_i^2) O_{Ai}^* &= \left( \mathbf{M}_N f(\mathbf{M}_N^\dagger \mathbf{M}_N) \right)_{BA} = \left( f(\mathbf{M}_N \mathbf{M}_N^\dagger) \mathbf{M}_N \right)_{BA} . \quad (2.7)
\end{aligned}$$

RHS of all of the expressions above depends on the matrix elements of functions of the hermitian matrices. As proven in Ref. [1], such elements can be expanded in terms of “divided differences”  $f^{[k]}$  of the function  $f$ , defined recursively as

$$\begin{aligned}
f^{[0]}(x) &\equiv f(x) , \\
f^{[1]}(x_0, x_1) &\equiv \frac{f(x_0) - f(x_1)}{x_0 - x_1} , \\
&\dots \\
f^{[k+1]}(x_0, \dots, x_k, x_{k+1}) &\equiv \frac{f^{[k]}(x_0, \dots, x_{k-1}, x_k) - f^{[k]}(x_0, \dots, x_{k-1}, x_{k+1})}{x_k - x_{k+1}} \quad (2.8)
\end{aligned}$$

or, for degenerate case (and analytical  $f$ ),

$$\lim_{\{x_0, \dots, x_m\} \rightarrow \{\xi, \dots, \xi\}} f^{[k]}(x_0, \dots, x_k) = \frac{1}{m!} \frac{\partial^m}{\partial \xi^m} f^{[k-m]}(\xi, x_{m+1}, \dots, x_k) . \quad (2.9)$$

The expansion can be done by splitting the mass matrices into diagonal parts and off-diagonal mass insertions,

$$\mathbf{M}^2 = \mathbf{M}_0^2 + \hat{\mathbf{M}}^2 , \quad (2.10)$$

where, by definition,

$$\begin{aligned}
(M_0^2)_I &\equiv M_{II}^2 , \\
\hat{M}_{IJ}^2 &\equiv M_{IJ}^2, \quad \hat{M}_{II}^2 = 0 , \quad (I, J = 1, \dots, n) . \quad (2.11)
\end{aligned}$$

Then, for any analytical Hermitian matrix function  $f(\mathbf{M}^2)$  (see [1] for detailed assump-

tions), a matrix element will be given by the expansion

$$\begin{aligned}
f(\mathbf{M}^2)_{IJ} &= \delta_{IJ} f((M_0^2)_I) + f^{[1]}((M_0^2)_I, (M_0^2)_J) \hat{M}_{IJ}^2 \\
&+ \sum_{K_1} f^{[2]}((M_0^2)_I, (M_0^2)_J, (M_0^2)_{K_1}) \hat{M}_{IK_1}^2 \hat{M}_{K_1J}^2 \\
&+ \sum_{K_1, K_2} f^{[3]}((M_0^2)_I, (M_0^2)_J, (M_0^2)_{K_1}, (M_0^2)_{K_2}) \hat{M}_{IK_1}^2 \hat{M}_{K_1K_2}^2 \hat{M}_{K_2J}^2 + \dots,
\end{aligned} \tag{2.12}$$

Notice that due to the definition (2.11) terms proportional to  $\hat{M}_{II}^2$  will always vanish in the summation.

### 3. Expanding amplitudes with `MassToMI` package

Eqs. (2.7, 2.12) allow to expand any transition amplitude calculated in the mass eigenstates basis as a series in mass insertions powers. However, for realistic models with many fields it may lead to lengthy expressions, requiring tedious and error-prone calculations. To facilitate the problem, the `MassToMI` package automatizes such calculations.

#### 3.1. Syntax for constructing amplitudes in `MassToMI`

In general, any amplitude, 1PI irreducible or reducible, can be constructed as a sum of products of mixing matrices of scalars, fermions or vector bosons, physical particle masses, loop integrals and other factors which are not affected by the FET expansion. To identify the combinations of objects relevant for the expansion procedure, `MassToMI` package require them to be denoted using the special syntax, listed in Table 1 (other symbolic or numerical factors in the amplitude, not listed in Table 1, are treated by program as constants).

The sub-expressions in the amplitude which could be expanded in terms of MIs need to have one of the forms listed on the LHS of eq. (2.7). Some further remarks are in order here:

- As in the analytical expressions of eq. (2.7), it is assumed that indices of particles appearing more than once in the amplitude, written as Mathematica code, are summed over without the need of specifying the sum explicitly.
- The amplitude does not need to be 1PI, i.e. products of loop functions are allowed.
- Only linear fermion mass powers (or multi-linear for amplitudes involving more than one fermion) can appear as factors in the amplitude, all even powers of masses should be included in the definitions of the loop functions.
- Eq. (2.8) defines the divided differences for functions of one variable. The generalization to many variables is obvious: one should calculate the divided differences

| Object and its syntax in <b>MassToMI</b>                        | Arguments & examples  |
|---|---|
| Scalar (vector) mixing matrix $Z$                               |   |
| $\text{SMIX}[P, i, j]$  | $P$ - particle symbol<br>$i, j$ - particle indices  |
| Example:  | $Z_D^{Ij} = \text{SMIX}[D, I, j]$<br>$Z_U^{Ij*} = \text{Conjugate}[\text{SMIX}[U, I, j]]$   |
| Left Dirac fermion mixing matrix $V$                            |   |
| $\text{FMIXL}[P, i, j]$   | $P$ - particle symbol<br>$i, j$ - particle indices  |
| Example:  | $V_D^{Ij} = \text{FMIXL}[D, I, j]$  |
| Right Dirac fermion mixing matrix $U$                           |   |
| $\text{FMIXR}[P, i, j]$   | $P$ - particle symbol<br>$i, j$ - particle indices  |
| Example:  | $U_Q^{Ij*} = \text{Conjugate}[\text{FMIXR}[Q, I, j]]$   |
| Majorana fermion mixing matrix $O$                              |   |
| $\text{NMIX}[P, i, j]$  | $P$ - particle symbol<br>$i, j$ - particle indices  |
| Example:  | $O_Q^{Ij} = \text{NMIX}[Q, I, j]$   |
| physical particle mass  |   |
| $\text{MASS}[P, i]$   | $P$ - particle symbol<br>$i$ - particle index   |
| Example:  | $m_Q^I = \text{MASS}[Q, I]$   |
| general loop function   |   |
| $\text{LOOP}[\text{name}, \{\{P1, i1\}, \dots\}, \{a, \dots\}]$ | $\text{name}$ - name of function<br>$\{\{P1, i1\}, \dots\}$ - physical loop masses<br>given as the list of (particle,index) pairs<br>$\{a, \dots\}$ - optional other arguments  |
| Examples:   | $b_{21}(p, m_{U_i}^2, m_{D_j}^2) =$<br>$\text{LOOP}[\text{b21}, \{\{U, i\}, \{D, j\}\}, \{p\}]$<br>$c_0(p, q, m_{A_i}^2, m_{B_j}^2, m_{C_k}^2) =$<br>$\text{LOOP}[\text{c0}, \{\{A, i\}, \{B, j\}, \{C, k\}\}, \{p, q\}]$ |

Table 1: Syntax of objects used to constructing amplitudes in **MassToMI** package



of the required order with respect to each variable separately. However, complications arise when some of the particles in the loop have equal masses (as for example in the case of flavor-diagonal photon couplings) and several arguments of loop function are identical. Then, the generalization of eq. 2.8 is less trivial and leads to some form of “Leibniz-like” rule, which combinatorial complications growing quickly with the number of identical arguments and the order of divided difference. **MassToMI** package does not allow for repeating arguments of the loop functions, assuming if necessary they have been appropriately renamed, e.g.:

$$c0(m_1^2, m_2^2, m_1^2) \rightarrow c0new(m_1^2, m_2^2) \quad (3.1)$$

To give an example, let's consider the 3-point scalar-fermion-fermion 1-loop amplitude, the triangle diagram with Dirac fermions  $C_n$ , Majorana fermions  $N_j$  and scalars  $D_i$  circulating in the loop. In general such an amplitude is the sum of terms depending on mixing matrices and 3-point loop function, one of them could look like

$$Amplitude = Z_D^{Ii} Z_D^{Ji*} O_N^{Kj} O_N^{Lj*} V_C^{Mn*} U_C^{Nn} m_{C_n} c_{21}(p, q, m_{C_n}^2, m_{D_i}^2, m_{N_j}^2) \quad (3.2)$$

To expand this in terms of mass insertions with the use of **MassToMI** routines, one must rewrite it into a Mathematica expression as:

```
Amplitude = SMIX[D,I,i] Conjugate[SMIX[D,J,i]]
            NMIX[N,K,j] Conjugate[NMIX[N,L,j]]
            Conjugate[FMIXL[C,M,n]] FMIXR[C,N,n] MASS[C,n]
            LOOP[c21,{{C,n},{D,i},{N,j}},{{p,q}}];
```

### 3.2. Control variables in MassToMI

**MassToMI** package uses the following control variables, defining which combinations of mixing matrices should be expanded to which order in terms of mass insertions, and in which form the final result should be presented.

- **FetScalarList** =  $\{\{P1, o1\}, \dots, \{Pn, on\}\}$ . In this list the user can specify symbols for the scalar and vector particles ( $P1, \dots, Pn$ ) for which the mixing matrices are expanded in MI powers to orders  $o1, \dots, on$ , respectively.
- **FetFermionList** =  $\{\{P1, o1, M1\}, \dots, \{Pn, on, Mn\}\}$ . Specifies list of Dirac or Majorana fermions  $P1, \dots, Pn$  for which the mixing matrices are expanded to MI orders  $o1, \dots, on$ . Each of the arguments  $M1, \dots, Mn$  can take one of two values, **MHM** or **MMH**, deciding which form of the RHS in the last 3 lines of eq. (2.7) is used, depending on  $M^\dagger M$  or on  $MM^\dagger$ , respectively (see discussion below).
- **FetMaxOrder** =  $n$ . Only mass insertion products of the total order **FetMaxOrder** or lower are kept in the final result.

Particles with symbols not specified on `FetScalarList` or `FetFermionList` lists are left unexpanded.

It is important to note that although the alternative forms of the RHS in the last 3 lines of eq. (2.7) are order-by-order equivalent when the function  $f$  is expanded in a Taylor series, they are *not* order-by-order equivalent when FET expansion of eq. (2.12) for  $f$  is used. To illustrate this, let's consider the lowest order FET expansion of the last line of eq. (2.7). It would lead to (for Majorana fermions  $\mathbf{M}_N = \mathbf{M}_N^T$ ):

$$\begin{aligned} \left( \mathbf{M}_N f(\mathbf{M}_N^\dagger \mathbf{M}_N) \right)_{BA} &\rightarrow (\mathbf{M}_N)_{BA} f \left( \left( \mathbf{M}_N^\dagger \mathbf{M}_N \right)_{AA} \right) \\ \left( f(\mathbf{M}_N \mathbf{M}_N^\dagger) \mathbf{M}_N \right)_{BA} &\rightarrow (\mathbf{M}_N)_{BA} f \left( \left( \mathbf{M}_N \mathbf{M}_N^\dagger \right)_{BB} \right) \end{aligned} \quad (3.3)$$

with both forms obviously different for  $A \neq B$ .

The choice of the correct form of the RHS of eq. (2.7) depends, for a given amplitude, on the physical context and can be determined by adjusting the value of the 3rd parameter in each entry of the `FetFermionList`, as described above.

### 3.3. The main expansion routine and output syntax

The actual MI expansion is done by a call to the function

**FetExpand[ Amplitude ]**

The `FetExpand` routine performs the following actions:

1. It checks first the structure of the amplitude for possible syntax problems, displaying if necessary the relevant error or warning messages.
2. It finds pairs of mixing matrices for which the *second* index is identical (like e.g. in  $X_Q^{Ai*} X_Q^{Bi}$  product), and the particle symbol ( $Q$  in this case) appears on `FetScalarList` or on `FetFermionList` (as mentioned above it is assumed that the repeating index denotes summation over it). For pairs of fermionic matrices, the program also searches if they are associated with any linear fermion mass factors with the same summation index.
3. The routine finds the loop functions which arguments have indices identical to those in the matching pairs of mixing matrices and performs the FET expansion for each of such pairs, using the formula of eq. (2.7) and eq. (2.12). Some further syntax checks are performed at this stage and reported if necessary.
4. Finally the result is displayed in terms of the objects used to denote diagonal and off-diagonal entries of the mass matrices listed in Table 2.

The expanded amplitude depends on the divided differences of the loop functions appearing in the original unexpanded expression. The notation for the divided differences is as follows:

| Object  | Syntax and arguments   | Example   |
|---|--|---|
| Diagonal entry of scalar squared mass matrix                | $\text{MS2}[P, i, i]$<br>$P$ - particle symbol<br>$i$ - particle index       | $(M_P^2)_{ii} = \text{MS2}[P, i, i]$            |
| Fermion mass matrix   | $\text{MF}[P, i, j]$<br>$P$ - particle symbol<br>$i, j$ - particle indices   | $(M_P)_{ij} = \text{MF}[P, i, j]$               |
| Diagonal entry of squared $M^\dagger M$ fermion mass matrix | $\text{MHM}[P, i, i]$<br>$P$ - particle symbol<br>$i$ - particle index       | $(M_P^\dagger M_P)_{ii} = \text{MHM}[P, i, i]$  |
| Diagonal entry of squared $MM^\dagger$ fermion mass matrix  | $\text{MMH}[P, i, i]$<br>$P$ - particle symbol<br>$i$ - particle index       | $(M_P M_P^\dagger)_{ii} = \text{MMH}[P, i, i]$  |
| Scalar MI matrix  | $\text{MS2I}[P, i, j]$<br>$P$ - particle symbol<br>$i, j$ - particle indices | $(M_P^2)_{ij} = \text{MS2I}[P, i, j]$           |
| Fermion squared $M^\dagger M$ MI matrix                     | $\text{MHMI}[P, i, j]$<br>$P$ - particle symbol<br>$i, j$ - particle index   | $(M_P^\dagger M_P)_{ij} = \text{MHMI}[P, i, j]$ |
| Fermion squared $MM^\dagger$ MI matrix                      | $\text{MMHI}[P, i, j]$<br>$P$ - particle symbol<br>$i, j$ - particle index   | $(M_P M_P^\dagger)_{ij} = \text{MMHI}[P, i, j]$ |

Table 2: Notation for the mass matrices in the interaction basis appearing in the expanded amplitudes. The objects denoting off-diagonal mass insertions,  $\text{MS2I}$ ,  $\text{MHMI}$  and  $\text{MMHI}$ , are assumed to have 0's on diagonal.

- The loop functions are renamed using the first argument of the original expression; the new first argument (non-negative integer) is added to denote the order of divided difference. The list containing the other arguments is flattened, so that the arguments are no longer splitted into masses and optional variables. This is illustrated by the following example:

$$\text{LOOP}[\text{b21}, \{\{\text{U}, i\}, \{\text{C}, j\}\}, \{\text{p}\}] \rightarrow \text{b21}[0, \{\text{U}, i\}, \{\text{C}, j\}, \text{p}]$$

- The physical mass arguments of the loop functions undergoing the MI expansion are replaced by a list containing the diagonal entries of mass matrices (boson or fermion squared masses) representing the variables of divided difference in the given argument.
- The divided differences of the loop functions are multiplied by the relevant mass insertion factors. Again, it is assumed that all repeated indices named **fetQxx**

should be summed over.

Continuing the example given above, let's assume that MI expansion has been performed to 2nd order for a fermion  $C$ . Then the highest order term in the expansion is proportional to the function denoted as:

$$\begin{aligned} & \text{Conjugate}[\text{FMIXL}[C, c, j]] \text{ FMIXL}[C, d, j] \text{ LOOP}[\text{b21}, \{\{U, i\}, \{C, j\}\}, \{p\}] \rightarrow \\ & \text{MMHI}[C, d, \text{fetC1}] \text{ MMHI}[C, \text{fetC1}, c] \times \\ & \times \text{b21}[2, \{U, i\}, \{\text{MMH}[C, d, d], \text{MMH}[C, \text{fetC1}, \text{fetC1}], \text{MMH}[C, c, c]\}, p] \end{aligned}$$

where 2 being used as the 1st argument of the `b21` function is the order of the divided difference,  $\text{MMH}[C, i, i] = (M_C M_C^\dagger)_{ii}$ ,  $\text{MMHI}[C, i, j] = (M_C M_C^\dagger)_{ij}$  for  $i \neq j$  and the summation over the internal index `fetC1` is assumed.

- If the FET expansion is performed in several indices, the first argument of the expanded loop function denotes the total order (the sum of all orders) of the divided differences in all arguments.
- If the final result still depends on some of the physical masses (not expanded into MI series), they are now denoted as `MASS[P, i]`, e.g.

$$\begin{aligned} & \text{b21}[2, \{U, i\}, \{\text{MMH}[C, d, d], \text{MMH}[C, \text{fetC1}, \text{fetC1}], \text{MMH}[C, c, c]\}, p] \rightarrow \\ & \text{b21}[2, \text{MASS}[U, i]^2, \{\text{MMH}[C, d, d], \text{MMH}[C, \text{fetC1}, \text{fetC1}], \text{MMH}[C, c, c]\}, p] \end{aligned}$$

- If necessary, the divided differences can be re-expressed as the explicit combinations of the original loop functions with the use of `FetExpandDividedDifferences` functions described in next section.

### 3.4. Auxiliary functions

In addition to the main `FetExpand` routine, the `MassToMI` package provides several auxiliary functions. For consistency and to distinguish them from other routines provided by Mathematica, all their names, by convention, also start from the prefix `Fet`.

One of the auxiliary functions, `FetExpandDividedDifferences`, allows to re-express divided differences appearing in the `FetExpand` output as the combinations of the initial loop functions. Other functions help to manipulate expressions with repeating indices, assumed to be implicitly summed over. They naturally appear in the higher order terms of the FET expansion (see eq. (2.12)) in sums over mass matrix indices. In many cases, the summation convention can be also used to define the initial (unexpanded) amplitude in a compact form. However, for the analysis of the physical effects it may be useful to expand some (or all) repeated indices as explicit sums. For that purpose, `MassToMI` package provides 3 routines, `FetSumParticle`, `FetSumFactor` and `FetSumIndex`. In addition, Mathematica, at least in version 10, does not have summation convention rules with Kronecker  $\delta$ -symbol implemented. Resumming Kronecker symbols can be done using the `FetSumWithDelta` function.

| Function  | Arguments  |
|---|--|
| <code>FetExpandDividedDifferences[ DivFunction, (<i>ReplacementRule</i>) ]</code> |  |
| DivFunction   | divided difference   |
| ReplacementRule   | rule how to redefine Function (optional)                             |
| <code>FetSumWithDelta[ Expression, (<i>ExcludeList</i>) ]</code>                  |  |
| Expression  | expression containing Kronecker $\delta$ -symbols                    |
| ExcludeList   | index or list of indices excluded from summation (optional)          |
| <code>FetSumParticle[ Expression, Particle, Range, (<i>ExcludeList</i>) ]</code>  |  |
| Expression  | expression to expand repeating indices                               |
| Particle  | symbol of particle (or list of particles) which indices are expanded |
| Range={Low,Up}  | summation range for repeating indices                                |
| ExcludeList   | index or list of indices excluded from summation (optional)          |
| <code>FetSumFactor[ Expression, Factor, Range, (<i>ExcludeList</i>) ]</code>      |  |
| Expression  | expression to expand repeating indices                               |
| Factor  | factor name (or list of factors) which indices are expanded          |
| Range={Low,Up}  | summation range for repeating indices                                |
| ExcludeList   | index or list of indices excluded from summation (optional)          |
| <code>FetSumIndex[ Expression, Index, Range ]</code>                              |  |
| Expression  | expression to expand   |
| Index   | name of index to expand  |
| Range={Low,Up}  | summation range for Index  |

Table 3: List and arguments of auxiliary functions provided by `MassToMI` package.

Below we describe in details the syntax for using the auxiliary functions (their compact list is collected in Table 3). One should note that some of their arguments, given in parenthesis with slanted font, are optional.

#### 3.4.1. `FetExpandDividedDifferences[ DivFunction, (ReplacementRule) ]`

`FetExpandDividedDifferences` expands the divided difference of a function, given as

`DivFunction = fname[(n + ..., {M_1, ..., M_(n+1)}), ..., optional arguments]`

into an explicit combination of the initial functions with different mass arguments (the first argument, order of divided difference, is truncated in the expanded form). For the degenerate mass arguments relevant derivatives (in the standard Mathematica notation) of the initial function are used. Examples are given below.

Non-degenerated arguments:

$$\begin{aligned}
&\text{FetExpandDividedDifferences[ b21[2, \{m1, m2, m3\}, M1, p] ]} = \\
&\quad \text{b21[m1, M1, p]/(m1 - m2)/(m1 - m3) - b21[m2, M1, p]/(m1 - m2)/(m2 - m3)} \\
&\quad + \text{b21[m3, M1, p]/(m1 - m3)/(m2 - m3)}
\end{aligned}$$

Degenerated arguments:

$$\begin{aligned} \text{FetExpandDividedDifferences}[b21[2, \{m1, m2, m1\}, M1, p]] &= \\ (b21[m2, M1, p] - b21[m1, M1, p]) / (m1 - m2)^2 + b21^{(1,0,0)}[m2, M1, p] / (m1 - m2) \\ \text{FetExpandDividedDifferences}[b21[2, \{m1, m1, m1\}, M1, p]] &= b21^{(2,0,0)}[m1, M1, p] / 2 \end{aligned}$$

Divided difference of multiple arguments, of total order  $2 + 1 = 3$ :

$$\begin{aligned} \text{FetExpandDividedDifferences}[b21[3, \{m1, m1, m1\}, \{M1, M2\}, p]] &= \\ (b21^{(2,0,0)}[m1, M1, p] - b21^{(2,0,0)}[m1, M2, p]) / 2 / (M1 - M2) \end{aligned}$$

The second optional argument of `FetExpandDividedDifferences` should be specified if the function used as the first argument was renamed to avoid multiple identical mass arguments, as in the example given in eq. (3.1). Specifying the replacement rule as the second argument allows to revert the redefinition and to go back to the original function name, argument list and, for the degenerate mass arguments, to calculate correctly the function derivatives. Example:

$$\begin{aligned} \text{FetExpandDividedDifferences}[c0new[2, \{m1, m2, m1\}, M1]] &= \\ (c0new[m2, M1] - c0new[m1, M1]) / (m1 - m2)^2 \\ + c0new^{(1,0)}[m1, M1] / (m1 - m2) \end{aligned}$$

but

$$\begin{aligned} \text{FetExpandDividedDifferences}[c0new[2, \{m1, m2, m1\}, M1], c0new[a_, b_] \rightarrow c0[a, b, a]] &= \\ (c0[m2, M1, m2] - c0[m1, M1, m1]) / (m1 - m2)^2 \\ + (c0^{(1,0,0)}[m1, M1, m1] + c0^{(0,0,1)}[m1, M1, m1]) / (m1 - m2) \end{aligned}$$

### 3.4.2. `FetSumWithDelta[ Expression, (ExcludeList) ]`

`FetSumWithDelta` function searches for occurrences of the factor `KroneckerDelta[a,b]` in `Expression` and if possible replaces them using the standard rule

$$\text{KroneckerDelta}[a, b] X[a] \rightarrow X[b],$$

excluding the cases when summation index is specified on the (optional) `ExcludeList`. This option allows e.g. to exclude external particle indices from summation. Example:

$$\begin{aligned} \text{exp} &= \text{KroneckerDelta}[I, a] \text{KroneckerDelta}[J, b] \text{Fun}[a, b] \\ \text{FetSumWithDelta}[\text{exp}] &= \text{Fun}[I, J] \\ \text{FetSumWithDelta}[\text{exp}, \{b\}] &= \text{KroneckerDelta}[J, b] \text{Fun}[I, b] \end{aligned}$$

Few additional remarks are in order:

- `FetSumWithDelta` does not perform summation if the repeating index is not a simple variable but an expression by itself, so terms of the form

`KroneckerDelta[a+1,b] f[a]`

are left unchanged. However the argument of `f` is evaluated properly:

`FetSumWithDelta[ KroneckerDelta[a,b] f[a+1] ] = f[b+1].`

- If the second `KroneckerDelta` argument, complementary to the summation index, is an expression, replacement is done:

`FetSumWithDelta[ KroneckerDelta[a,b+3] f[a] ] = f[b+3]`

However, `FetSumWithDelta` does not check for the allowed range for summation indices - if in the example above `a,b = 1...3`, then `KroneckerDelta[a,b+3] ≡ 0` and the result given by `FetSumWithDelta` is not correct. Expressions of that type must be simplified manually by the user.

### 3.4.3. `FetSumParticle[ Expression, Particle, Range, (ExcludeList) ]`

`FetSumParticle` expands the summation convention for the repeating indices of mass insertion matrices for given set of particles specified in the argument `Particle` (it could be a single particle symbol or list of symbols). More specifically, it searches for the occurrence of factors of the form `X[Particle,a,b]`, where `X` is one of the following matrices: `X=MF,MS2I,MHMI,MMHI`. If such a factor is found, and at least one of its indices (not specified on the optional argument `ExcludeList`) is present also in the term multiplying it, the repeating index is replaced by the explicit sum in the range `Range={Low,Up}`. Examples:

`exp = MF[Q,A,K] MF[Q,A,J] + MS2I[P,K,A] MA[A,J] + MF[P,C,K] MB[C,J]`

`FetSumParticle[ exp,P,{1,2} ] = MB[1,J] MF[P,1,K] + MB[2,J] MF[P,2,K]  
+ MF[Q,A,J] MF[Q,A,K] + MA[1,J] MS2I[P,K,1] + MA[2,J] MS2I[P,K,2]`

`FetSumParticle[ exp,{P,Q},{1,2},C ] = MB[C,J] MF[P,C,K] + MF[Q,1,J] MF[Q,1,K]  
+ MF[Q,2,J] MF[Q,2,K] + MA[1,J] MS2I[P,K,1] + MA[2,J] MS2I[P,K,2]`

### 3.4.4. `FetSumFactor[ Expression, Factor, Range, (ExcludeList) ]`

`FetSumFactor` routine can be used to expand summation over indices of objects not related to MI expansion (like CKM matrix). It searches for the repeating indices of factor(s) specified as the argument `Factor`. Factors can have any number of indices, but they are all summed in the same `Range={Low,Up}`. If various indices have different summation ranges, routine should be called several times, specifying optional `ExcludeList` argument to avoid summation over indices not belonging to correct `Range`. Examples:

`exp = MA[A,K] CKM[A,J]`

`FetSumFactor[ exp,CKM,{1,3} ] = CKM[1,J] MA[1,K] + CKM[2,J] MA[2,K]  
+ CKM[3,J] MA[3,K]`

`FetSumFactor[ exp,CKM,{1,3}, A ] = CKM[A,J] MA[A,K]`

### 3.4.5. FetSumIndex[ Expression, Index, Range ]

**FetSumIndex** routine can be used to expand summation over single index directly specified by user. Example:

`exp = 1 + A[J,K] B[J,L]`

`FetSumIndex[ exp,J,{1,3} ] = 1 + A[1,K] B[1,L] + A[2,K] B[2,L] + A[3,K] B[3,L]`

### 3.4.6. Special cases and limitations

Syntax of **MassToMI** package assumes in general that repeating indices are implicitly summed over. However, this is not a summation convention in a “classical” sense, as it allows for indices repeating more than twice. Such definition is convenient for applications of FET expansion, but, used without proper care, may lead to various ambiguities. Thus, it is advisable to check the structure of the intermediate expressions appearing during calculations, particularly before applying to them auxiliary functions designed for manipulating objects with repeating indices.

Few additional remarks may be helpful to avoid problems with the incorrect usage of routines provided by **MassToMI** package:

- It is highly recommended to use the **ExcludeList** argument of **FetSumWithDelta**, **FetSumParticle** and **FetSumFactor** functions for specifying which indices correspond to external particles in given amplitude, to avoid resummation over them accidentally.
- Notice that the functions **FetSumFactor**, **FetSumParticle** and **FetSumWithDelta** search only for repeating indices in simple factors in the analyzed expression, not checking for their appearance in lower level sub-expression (those are left unchanged). For example:

`FetSumWithDelta[ KroneckerDelta[a,b] f[a] ] = f[b]`

but

`FetSumWithDelta[ Fun[KroneckerDelta[a,b] f[a]] ] =  
Fun[f[a] KroneckerDelta[a,b]]`

and similarly for **FetSumParticle** and **FetSumFactor**. **FetSumIndex** will work even for indices hidden in sub-expressions, assuming that they appear in at least two different factors.

- Functions **FetSumParticle** and **FetSumFactor** treat higher powers of matrix elements as products with repeating indices, and perform appropriate resummations, as illustrated below:

`FetSumFactor[ M[a,b]^2,M,{1,2} ] ≡ FetSumFactor[ M[a,b]M[a,b],M,{1,2} ] =  
M[1,1]^2 + M[1,2]^2 + M[2,1]^2 + M[2,2]^2`



- One should strictly avoid using the same names for various types of quantities appearing in the transition amplitudes or other expressions used as arguments of `MassToMI` functions. In particular, functions dealing with summation convention cannot distinguish what is really an “index” and try to sum over all symbols in given expression with the names identical to repeating indices. This may lead to strange bugs and obviously incorrect or nonsensical results, like in the examples given below:

```
FetSumWithDelta[ KroneckerDelta[A,B] A[A] ] = B[B]
FetSumFactor[ M[A,B] A[A],A,{1,2} ] = 1[1] M[1,B] + 2[2] M[2,B]
```

### 3.5. `MassToMI` installation

The `MassToMI` package does not require any special installation procedures. It should be unpacked to the directory accessible to Mathematica, or installed system-wide using *Install* command from Mathematica menu. Then the package can be loaded using the commands

```
Needs["MassToMI`"];
or
<< MassToMI.m
```

## 4. Examples of the `MassToMI` applications

We collect below several examples illustrating the functionality of the `MassToMI` package, from simple test amplitudes to the realistic case of the expansion of one of the diagrams contributing to the flavor violating decay of the Higgs boson to leptons,  $h \rightarrow \tau\mu$ , in the MSSM.

### 4.1. Expansion of simple amplitudes

The two basic examples of `MassToMI` usage presented below are included in the file `mmi_example.m` attached to the `MassToMI` distribution.

**Example 1.** The simplest case involves a pair of mixing matrices of a single scalar particle multiplying loop function  $a(m^2)$ . Lets assume that amplitude should be expanded to 2nd order in MI powers.

```
Ampl = Conjugate[SMIX[P,a,i]] SMIX[P,b,i] LOOP[A0,{P,i}];
FetScalarList = {{P,2}};
FetMaxOrder = 2;
FetAmpl = FetExpand[Ampl];
Print["Expanded amplitude = ", FetAmpl];
```

The result is:

```
Expanded amplitude = A0[0,MS2[P,b,b]] KroneckerDelta[a,b] +
A0[1,{MS2[P,b,b],MS2[P,a,a]}] MS2I[P,b,a] +
A0[2,{MS2[P,b,b],MS2[P,fetP1,fetP1],MS2[P,a,a]}] MS2I[P,b,fetP1] MS2I[P,fetP1,a]
```

Note that only amplitudes of a specific structure (see eq. (2.7)) are allowed and `FetExpand` reports errors if incorrect combinations of mixing matrices are used. For example, if c.c. is removed from one of the scalar matrices in the example above,

```
Ampl = SMIX[P,a,i] SMIX[P,b,i] LOOP[A0,{P,i}];
```

program reports:

```
MassToMI::cmplx2: ERROR: Incorrect matching of c.c.'s in product of SMIX[P,a,i]
and SMIX[P,b,i]?
```

and aborts the execution.

In other cases only warnings or informational messages are displayed but program still makes an attempt to calculate the result, like for instance the mismatch in the summation index of scalar matrices:

```
Ampl = Conjugate[SMIX[P,a,j]] SMIX[P,b,i] LOOP[A0,{P,i}];
```

leads to

```
MassToMI::war1: WARNING: unmatched index of scalar on FetExpandList in
SMIX[P,a,j]?
```

```
MassToMI::war1: WARNING: unmatched index of scalar on FetExpandList in
SMIX[P,b,i]?
```

```
Expanded amplitude = A0[0,MASS[P,i]^2] Conjugate[SMIX[P,a,j]] SMIX[P,b,i]
```

meaning that no pair of scalar mixing matrices with matching 2nd index was found, so the result is simply equal to the initial amplitude rewritten using the syntax rules for the `MassToMI` output.

The full list of errors, warnings and informational messages can be found in the header of the `MassToMI.m` file.

**Example 2.** Let us now consider a more complicated case of an amplitude depending on scalar  $P$  and Dirac fermion  $F$  masses. We request 1st order of MI expansion for both particles and set `FetMaxOrder = 1` as a maximum total MI order. The preferred form of the squared fermion mass in output is  $M^\dagger M$ .

```
Ampl = Conjugate[SMIX[P,a,i]] SMIX[P,b,i] Conjugate[FMIXL[F,c,j]] FMIXR[F,d,j]
MASS[F,j] LOOP[B0,{P,i},{F,j}];
FetScalarList = {{P,1}};
FetFermionList = {{F,1,MHM}};
FetMaxOrder = 1;
```

```
FetAmpl = FetExpand[Ampl];
Print["Expanded amplitude = ", FetAmpl];
```

The result is

```
Expanded amplitude = B0[0,MS2[P,b,b],MHM[F,d,d]] Conjugate[MF[F,c,fetF1]]
KroneckerDelta[a,b] KroneckerDelta[d,fetF1]
+ B0[1,MS2[P,b,b],{MHM[F,d,d],MHM[F,fetF1,fetF1]] Conjugate[MF[F,c,fetF1]]
KroneckerDelta[a,b] MHMI[F,d,fetF1]
+ B0[1,{MS2[P,b,b],MS2[P,a,a]},MHM[F,d,d]] Conjugate[MF[F,c,fetF1]]
KroneckerDelta[d,fetF1] MS2I[P,b,a]
```

where again summation over the repeating index `fetF1` is assumed. The result contains some Kronecker  $\delta$ -symbols with fermion indices which can be resummed

```
FetAmpl = FetSumWithDelta[FetAmpl,{a,b,c,d}];
```

giving the simpler expression

```
Expanded amplitude = B0[0,MS2[P,b,b],MHM[F,d,d]] Conjugate[MF[F,c,d]]
KroneckerDelta[a,b]
+ B0[1,MS2[P,b,b],{MHM[F,d,d],MHM[F,fetF1,fetF1]] Conjugate[MF[F,c,fetF1]]
KroneckerDelta[a,b] MHMI[F,d,fetF1]
+ B0[1,{MS2[P,b,b],MS2[P,a,a]},MHM[F,d,d]] Conjugate[MF[F,c,d]] MS2I[P,b,a]
```

#### 4.2. $h \rightarrow \tau\mu$ decay in the MSSM.

As a more realistic and complicated example we consider the flavor violating decay of the Higgs boson to leptons,  $h \rightarrow \tau\mu$ , in the Minimal Supersymmetric Standard Model (MSSM). This decay was recently measured by ATLAS [9] and CMS [10] Collaborations, and reported to be more frequent than predicted within the Standard Model. It was also theoretically recently reanalyzed within the MSSM [11] and claimed to be enhanced.

For simplicity, for the purpose of our example we expand the contribution from one diagram only, shown in Fig. 1 (results for MI expansion of sum of all diagrams contributing to the effective Higgs-fermion vertex in the MSSM can be found in Refs. [12, 13, 14]). It depends on the scalar (slepton) and fermionic (neutralino) mixing matrices.

In the approximation of the vanishing external momenta, the amplitude corresponding to this diagram reads as:

$$\mathcal{A} = \Gamma_L^{ABK} P_L + \Gamma_R^{ABK} P_R . \quad (4.1)$$

where

$$\Gamma_{L(R)}^{ABK} = - \sum_{i,j,l} m_{\chi_j} (V_{HLL}^{Kli} V_{iL\chi,R(L)}^{Alj*} V_{iL\chi,L(R)}^{Bij}) C_0[m_{L_i}, m_{L_l}, m_{\chi_j}] , \quad (4.2)$$

and  $A, B$  are lepton generation indices ( $B = 3$  and  $A = 2$  for  $\tau\mu$  in the final state,  $K$  is the Higgs index,  $K = 2$  for the “little  $h$ ” Higgs boson and  $K = 1$  for the “big  $H$ ” and  $C_0$  is the standard 3-point scalar loop integral for vanishing external momenta.

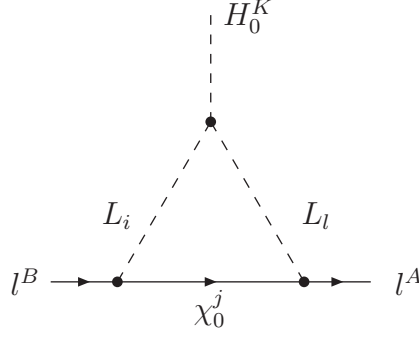


Figure 1: Slepton-neutralino diagram contributing to the  $h \rightarrow \tau\mu$  decay in the MSSM

Following strictly the notation and conventions of Ref. [15, 16] (we do not repeat them here explicitly), the Higgs-slepton and lepton-slepton-neutralino vertices read as

$$V_{lL\chi,L}^{Aij} = \frac{e}{\sqrt{2}s_W c_W} Z_L^{Ai} (Z_N^{1j} s_W + Z_N^{2j} c_W) + Y_l^A Z_L^{(A+3)i} Z_N^{3j} \quad (4.3)$$

$$V_{lL\chi,R}^{Aij} = \frac{-e\sqrt{2}}{c_W} Z_L^{(A+3)i} Z_N^{1j*} + Y_l^A Z_L^{Ai} Z_N^{3j*} \quad (4.4)$$

$$\begin{aligned} V_{HLL}^{Kli} &= \sum_{C=1}^3 \left( \frac{e^2}{2c_W^2} (v_1 Z_R^{1K} - v_2 Z_R^{2K}) \left( \delta^{il} + \frac{1-4s_W^2}{2s_W^2} Z_L^{Ci*} Z_L^{Cl} \right) \right. \\ &\quad - (Y_l^C)^2 v_1 Z_R^{1K} (Z_L^{Ci*} Z_L^{Cl} + Z_L^{(C+3)i*} Z_L^{(C+3)l}) \\ &\quad - \left. \frac{Z_R^{2K}}{\sqrt{2}} Y_l^C (\mu^* Z_L^{Ci*} Z_L^{(C+3)l} + \mu Z_L^{Cl} Z_L^{(C+3)i*}) \right) \\ &\quad - \frac{1}{\sqrt{2}} \sum_{C,D=1}^3 \left( Z_R^{1K} (A_l^{CD*} Z_L^{Cl} Z_L^{(D+3)i*} + A_l^{CD} Z_L^{Ci*} Z_L^{(D+3)l}) \right. \\ &\quad - \left. Z_R^{2K} (A_l'^{CD*} Z_L^{Cl} Z_L^{(D+3)i*} + A_l'^{CD} Z_L^{Ci*} Z_L^{(D+3)l}) \right) \end{aligned} \quad (4.5)$$

Neglecting the small terms proportional to lepton Yukawa couplings and, for simplicity, the “non-holomorphic”  $A_l'$  trilinear soft terms, the vertices and amplitude for  $\Gamma_L^{ABK}$  are coded as (summation over repeating generation indices  $C, D$  in HLL is assumed)<sup>2</sup>:

```
HLL[K_,l_,i_] = - e^2/2/cw^2 (v1 ZR[1,K] - v2 ZR[2,K]) (KroneckerDelta[i,1]
+ (1 - 4 sw^2)/2/sw^2 Conjugate[SMIX[L,C,i]] SMIX[L,C,1])
- 1/Sqrt[2] ZR[1,K] (Conjugate[AL[C,D]] SMIX[L,C,1] Conjugate[SMIX[L,D+3,i]] +
AL[C,D] Conjugate[SMIX[L,C,i]] SMIX[L,D+3,1]);

LSNL[J_,j_,i_] = e/(Sqrt[2] sw cw) SMIX[L,J,j] (NMIX[N,1,i] sw + NMIX[N,2,i] cw);
LSNR[J_,j_,i_] = - e Sqrt[2]/cw SMIX[L,J+3,j] Conjugate[NMIX[N,1,i]];
```

<sup>2</sup>The Mathematica code for the example discussed in this section is attached to the `MassToMI` distribution as the file `htaumu_decay.m`

```
(* amplitude *)
ampl = - MASS[N,j] HLL[K,l,i] Conjugate[LSNR[A,l,j]] LSNL[B,i,j] *
LOOP[c0,{{L,i},{L,l},{N,j}}] // Expand;

(* simplification: sum over KroneckerDelta[i,l] *)
ampl = FetSumWithDelta[ampl,{A,B,K}];
(* give new name for loop function with repeating arguments *)
ampl = ampl /. LOOP[c0,{{L,i-},{L,i-},{N,j}}] -> LOOP[c0sq,{{L,i},{N,j}}];
```

At this stage the amplitude is correctly defined in terms of `MassToMI` objects (expressions for  $\Gamma_R^{ABK}$  can be obtained by replacing  $L \leftrightarrow R$ ). For our purpose we expand it up to the lowest non-trivial order, i.e. 1st order in the slepton mass insertion and 0th order in the neutralino mass insertion (neutralino mass matrix entries are not related to flavor violation, so neglecting higher orders is equivalent to skipping  $\mathcal{O}(M_W^2/M_{SUSY}^2)$  suppressed terms). The actual expansion is done by:

```
(* control variables *)
FetMaxOrder = 1;
FetScalarList = {{L,1}};
FetFermionList = {{N,0,MHM}};

(* MAIN EXPANSION ROUTINE *)
fetampl = FetExpand[ ampl ];
```

The expanded amplitude is a lengthy expression build from objects defined in Section 3.3. It could be further significantly simplified by substituting the explicit form for the neutralino and slepton mass matrices.

```
(* simplify KroneckerDelta terms, excluding external indices *)
fetampl = fetampl /. KroneckerDelta[3 + a_, 3 + b_] -> KroneckerDelta[a, b];
fetampl = fetampl /. KroneckerDelta[3 + a_, b_] -> 0;
fetampl = fetampl /. KroneckerDelta[a_, 3 + b_] -> 0;
fetampl = FetSumWithDelta[ fetampl, {A,B,K} ];
```

```
(* perform explicit sum over AL indices (expand summation convention) *)
fetampl = FetSumFactor[ fetampl, AL, {1, 3} ];
```

```
(* Neutralino mass matrix MN and MHMN = MN^+MN *)
MN = Table[0,{i,1,4},{j,1,4}];
```

```
MN[[1,1]] = M1;
MN[[2,2]] = M2;
MN[[3,4]] = MN[[4,3]] = mu;
MN[[1,3]] = MN[[3,1]] = - e v1/2/cw;
MN[[1,4]] = MN[[4,1]] = e v2/2/cw;
MN[[2,3]] = MN[[3,2]] = e v1/2/sw;
MN[[2,4]] = MN[[4,2]] = - e v2/2/sw;
```

```

MHMN = ConjugateTranspose[MN].MN;

(* Slepton mass matrix ML2 (neglecting small Yukawa contributions) *)
ML2 = Table[0,{i,1,6},{j,1,6}];

For[i=1,i<4,i++, For[j=1,j<4,j++,
ML2[[i,j]] = e^2 (v1^2-v2^2) (1-2 cw^2)/(8 sw^2 cw^2) KroneckerDelta[i,j] + MLL[j,i];
ML2[[i+3,j+3]] = - e^2 (v1^2-v2^2) /(4 cw^2) KroneckerDelta[i,j] + MRR[i,j];
ML2[[i,j+3]] = v1/Sqrt[2] AL[i,j];
ML2[[i+3,j]] = Conjugate[ML2[[i,j+3]]];
] ];

(* slepton MI matrix *)
ML2I = ML2;
For[i=1,i<7,i++, ML2I[[i,i]]=0 ];

(* choose indices for h->tau mu decay *)
fetampl = fetampl /. A->2 /. B->3 /. K->2;

(* substitute real form of neutralino and slepton mass matrices *)
For[i=1,i<5,i++, For[j=1,j<5,j++,
fetampl = fetampl /. MF[N,i,j] -> MN[[i,j]] /. MHM[N,i,j] -> MHMN[[i,j]];
]; ];

For[i=1,i<7,i++, For[j=1,j<7,j++,
fetampl = fetampl /. MS2[L,i,j] -> ML2[[i,j]] /. MS2I [L,i,j] -> ML2I[[i,j]];
]; ];

(* remove unnecessary c.c. from real parameters *)
fetampl = fetampl // FunctionExpand;
fetampl = fetampl /. Conjugate[e] -> e /. Conjugate[cw] -> cw
/. Conjugate[sw] -> sw /. Conjugate[v1] -> v1
/. Conjugate[v2] -> v2 // Expand;

(* for simplicity assume real M1 *)
fetampl = fetampl /. Conjugate[M1] -> M1;

```

At this stage the flavor violating terms could come from the FET expansion or directly from the trilinear  $A_i$  terms explicitly present in the Higgs-slepton vertex of eq. (4.5), so their product could be of a order higher then 1. In addition, leaving the  $\mathcal{O}(M_W^2/M_{SUSY}^2)$  terms in the arguments of loop functions is inconsistent, as similar terms have been neglected in expanding neutralino mass matrices to 0th order only. Thus, further simplifications should be done:

```
(* kill higher order MI terms *)
```

```

fetampl = fetampl /. AL[A_,B_] -> eps AL[A,B] /. ALP[A_,B_] -> eps ALP[A,B] /.
MLL[A_,B_] -> eps MLL[A,B] /. MRR[A_,B_] -> eps MRR[A,B];
fetampl = fetampl /. AL[A_,A_] -> AL[A,A]/eps /. ALP[A_,A_] -> ALP[A,A]/eps /.
MLL[A_,A_] -> MLL[A,A]/eps /. MRR[A_,A_] -> MRR[A,A]/eps;
fetampl = Normal[Series[fetampl,{eps,0,1}]] /. eps->1;

```

```

(* neglect terms  $O(MW^2/MSUSY^2)$  in loop function arguments *)
fetampl = Simplify[fetampl] /. e^2(v1^2 + v2^2) -> 0
/. e^2(v1^2 - v2^2) -> 0 /. e^2(-v1^2 + v2^2) -> 0;

```

```

(* define and substitute explicit Higgs mixing matrix and vev's *)
ZH = Table[0,{i,1,2},{j,1,2}];
ZH[[1,1]] = ZH[[2,2]] = Cos[alpha];
ZH[[1,2]] = - Sin[alpha];
ZH[[2,1]] = Sin[alpha];
v1 = 2 MW sw/e Cos[beta];
v2 = 2 MW sw/e Sin[beta];
For[i=1,i<3,i++, For[j=1,j<3,j++, fetampl = fetampl /. ZR[i,j]->ZH[[i,j]] ] ];

```

In spite of lengthy and complicated intermediate expressions, difficult to obtain without the use of computer, the final result is compact and simple:

```

fetampl = 3 e^2 M1/(2 Sqrt[2] cw^4) (
+ 2/3 cw^2 AL[2,2] c0[1,MLL[3,3],MLL[2,2],MRR[2,2],M1^2] MLL[2,3] Sin[alpha]
+ 2/3 cw^2 AL[3,3] c0[1,MLL[3,3],MRR[3,3],MRR[2,2],M1^2] MRR[3,2] Sin[alpha]
+ (2 cw^2 c0[0,MLL[3,3],MRR[2,2],M1^2] Sin[alpha]
+ 2 MW^2 ((4 cw^2 - 3) c0[1,MLL[3,3],MLL[3,3],MRR[2,2],M1^2]
+ 2 sw^2 c0sq[1,MLL[3,3],MRR[2,2],M1^2]) Cos[beta] Sin[alpha+beta])/3 AL[3,2] )

```

As expected, it is linear in the three flavor violating slepton mass insertions,  $MLL[2,3]$ ,  $MRR[2,3]$  and  $AL[3,2]$ , with the coefficients given in a simple analytical form.

The divided differences of loop functions can be further expanded with the use of `FetExpandDividedDifferences` routine. The latter step may be in particular required if the diagonal slepton mass terms are degenerated and divided differences should be replaced by relevant derivatives. This may require some care: e.g. the  $c0sq$  function has been earlier defined in the code as  $c0sq[m1^2, m2^2] \equiv c0[m1^2, m1^2, m2^2]$ , so the divided differences expansion should be done reverting the redefinition:

```

FetExpandDividedDifferences[c0sq[ 1, {MLL[3,3],MRR[2,2]},M1^2],
c0sq[a_,b_] -> c0[a,a,b] ] =
(c0[MLL[3,3],MLL[3,3],M1^2] - c0[MRR[2,2],MRR[2,2],M1^2])/(MLL[3,3] - MRR[2,2])

```

Obviously, the same program could be easily adapted to expand other diagrams for this process and finally to calculate the decay branching ratio directly in terms of parameters of the initial MSSM Lagrangian. If necessary, with minimal modifications, it could be

also used to obtain higher order terms, both in flavor violating mass insertions and in  $M_W^2/M_{SUSY}^2$  powers. Such higher order terms may appear important, especially when lower order MI powers cancel out, as it often happens for rare decays in the MSSM - see e.g. discussion of  $t \rightarrow ch, uh$  decays in Ref. [17], where the accuracy of the results expanded by FET technique has been compared vs. the exact numerical calculations performed in mass eigenstates basis with the use of `SUSY_FLAVOR` library [18, 19, 20].

## 5. Summary

We presented `MassToMI` v1.0, a Mathematica package for an automatic expansion of transition amplitudes calculated in the mass eigenstates basis in terms of series of mass insertions. The expressions for the mass eigenstates amplitudes are usually more easier to calculate diagrammatically and better suited for the numerical computations but depend on the initial (interaction basis) Lagrangian parameters in a complicated way. The `MassToMI` routines allow to obtain an analytical approximation for the transition amplitudes directly as a power series in terms of the off-diagonal entries of mass matrices, without the need of a separate calculation of the Feynman diagrams with mass insertions as vertices.

The package is general enough to expand any amplitude, of any loop order, in any model involving scalar, fermion (Dirac or Majorana) or vector particles, expressing the result in terms of the divided differences of the loop functions. It can perform the Mass Insertion expansion to high orders, limited only by the combinatorial complication of the result and growing computational time.

In addition, `MassToMI` provides several auxiliary functions allowing to express divided differences of any order in terms of the initial function used to generate them, and to manipulate and expand expressions containing factors with repeating indices, assumed to be implicitly summed over.

The current version of `MassToMI` Mathematica code and its manual can be downloaded from the address

[www.fuw.edu.pl/masstomi](http://www.fuw.edu.pl/masstomi)

## Acknowledgments

This work was supported in part by Polish National Science Centre under research grants DEC-2012/05/B/ST2/02597 and DEC-2014/15/B/ST2/02157. The author would like to thank University of Ioannina and CERN for the hospitality during his stays there. I would also like to express my gratitude to Michalis Paraskevas for thorough testing the code and proofreading the manuscript.



- [1] A. Dedes, M. Paraskevas, J. Rosiek, K. Suxho, K. Tamvakis, Mass Insertions vs. Mass Eigenstates calculations in Flavour Physics, JHEP 1506 (2015) 151. [arXiv:1504.00960](#), [doi:10.1007/JHEP06\(2015\)151](#).
- [2] J. S. Hagelin, S. Kelley, T. Tanaka, Supersymmetric flavor changing neutral currents: Exact amplitudes and phenomenological analysis, Nucl. Phys. B415 (1994) 293–331. [doi:10.1016/0550-3213\(94\)90113-9](#).
- [3] F. Gabbiani, E. Gabrielli, A. Masiero, L. Silvestrini, A Complete analysis of FCNC and CP constraints in general SUSY extensions of the standard model, Nucl.Phys. B477 (1996) 321–352. [arXiv:hep-ph/9604387](#), [doi:10.1016/0550-3213\(96\)00390-2](#).
- [4] M. Misiak, S. Pokorski, J. Rosiek, Supersymmetry and FCNC effects, Adv.Ser.Direct.High Energy Phys. 15 (1998) 795–828. [arXiv:hep-ph/9703442](#).
- [5] A. J. Buras, A. Romanino, L. Silvestrini, K to pi neutrino anti-neutrino: A Model independent analysis and supersymmetry, Nucl.Phys. B520 (1998) 3–30. [arXiv:hep-ph/9712398](#), [doi:10.1016/S0550-3213\(98\)00169-2](#).
- [6] A. Crivellin, J. Girrbach, Constraining the MSSM sfermion mass matrices with light fermion masses, Phys.Rev. D81 (2010) 076001. [arXiv:1002.0227](#), [doi:10.1103/PhysRevD.81.076001](#).
- [7] Wolfram Research, Inc., Mathematica 10.2 (2015).
- [8] C. de Boor, Divided differences, Surv. Approx. Theory 1 (2005) 46–69. [arXiv:math/0502036](#).
- [9] G. Aad, et al., Search for lepton-flavour-violating  $H \rightarrow \mu\tau$  decays of the Higgs boson with the ATLAS detector [arXiv:1508.03372](#).
- [10] V. Khachatryan, et al., Search for lepton-flavour-violating decays of the Higgs boson, Phys. Lett. B749 (2015) 337–362. [arXiv:1502.07400](#), [doi:10.1016/j.physletb.2015.07.053](#).
- [11] M. Arana-Catania, E. Arganda, M. J. Herrero, Non-decoupling SUSY in LFV Higgs decays: a window to new physics at the LHC, JHEP 09 (2013) 160. [arXiv:1304.3371](#), [doi:10.1007/JHEP09\(2013\)160](#).
- [12] A. Crivellin, Effective Higgs Vertices in the generic MSSM, Phys. Rev. D83 (2011) 056001. [arXiv:1012.4840](#), [doi:10.1103/PhysRevD.83.056001](#).
- [13] A. Crivellin, L. Hofer, J. Rosiek, Complete resummation of chirally-enhanced loop-effects in the MSSM with non-minimal sources of flavor-violation, JHEP 07 (2011) 017. [arXiv:1103.4272](#), [doi:10.1007/JHEP07\(2011\)017](#).

- [14] A. Crivellin, C. Greub, Two-loop supersymmetric QCD corrections to Higgs-quark-quark couplings in the generic MSSM, *Phys. Rev. D* **87** (2013) 015013, [Erratum: *Phys. Rev. D* **87**, 079901(2013)]. [arXiv:1210.7453](#), [doi:10.1103/PhysRevD.87.015013](#), [10.1103/PhysRevD.87.079901](#).
- [15] J. Rosiek, Complete Set of Feynman Rules for the Minimal Supersymmetric Extension of the Standard Model, *Phys. Rev. D* **41** (1990) 3464. [doi:10.1103/PhysRevD.41.3464](#).
- [16] J. Rosiek, Complete set of Feynman rules for the MSSM: Erratum [arXiv:hep-ph/9511250](#).
- [17] A. Dedes, M. Paraskevas, J. Rosiek, K. Suxho, K. Tamvakis, Rare Top-quark Decays to Higgs boson in MSSM, *JHEP* **1411** (2014) 137. [arXiv:1409.6546](#), [doi:10.1007/JHEP11\(2014\)137](#).
- [18] J. Rosiek, P. Chankowski, A. Dedes, S. Jager, P. Tanedo, **SUSY\_FLAVOR**: A Computational Tool for FCNC and CP-Violating Processes in the MSSM, *Comput. Phys. Commun.* **181** (2010) 2180–2205. [arXiv:1003.4260](#), [doi:10.1016/j.cpc.2010.07.047](#).
- [19] A. Crivellin, J. Rosiek, P. Chankowski, A. Dedes, S. Jaeger, et al., **SUSY\_FLAVOR v2**: A Computational tool for FCNC and CP-violating processes in the MSSM, *Comput. Phys. Commun.* **184** (2013) 1004–1032. [arXiv:1203.5023](#), [doi:10.1016/j.cpc.2012.11.007](#).
- [20] J. Rosiek, **SUSY\_FLAVOR v2.5**: a computational tool for FCNC and CP-violating processes in the MSSM, *Comput. Phys. Commun.* **188** (2014) 208–210. [arXiv:1410.0606](#), [doi:10.1016/j.cpc.2014.10.003](#).