Mass Diagonalization

MadGraph meeting, Natal
7th of October 2012

Adam Alloul
Karen De Causmaecker
Benjamin Fuks

How to diagonalize numerically in FeynRules

- Implement the model + mixings in FeynRules
- Compute the tree level mass matrices
- Generate the numerical code
- Diagonalize the matrices and generate output

Example: The Tutorial

- SM + 2 real scalar fields with Lagrangian

$$
\begin{aligned}
L_{scalar} =& \frac{1}{2}\partial_\mu \phi_1 \partial^\mu \phi_1 + \frac{1}{2}\partial_\mu \phi_2 \partial^\mu \phi_2 \\
& - \frac{m_1^2}{2}\phi_1^2 - \frac{m_2^2}{2}\phi_2^2 - m_{12}^2 \phi_1 \phi_2
\end{aligned}
$$

- The rotation to the mass eigenstates had to be calculated analytically

Parameter declaration

```
MM1 == {ParameterType -> External, Value -> 1},
MM2 == {ParameterType -> External, Value -> 100},
MM12 == {ParameterType -> External, Value -> 0.5},

MPe1 == {
       ParameterType -> Internal,
       Value         -> Sqrt[(MM1^2 + MM2^2 - Sqrt[MM1^4 + 4*MM12^4 -
       2*MM1^2*MM2^2 + MM2^4]])/2]
      },

MPe2 == {
       ParameterType -> Internal,
       Value         -> Sqrt[(MM1^2 + MM2^2 + Sqrt[MM1^4 + 4*MM12^4 -
       2*MM1^2*MM2^2 + MM2^4]])/2]
      },

th == {
      ParameterType -> Internal,
      Value         -> ArcCos[1/Sqrt[1 + (-MM1^2 + MM2^2 + Sqrt[4*MM12^4 +
      (MM1^2 - MM2^2)^2])^2/(4*MM12^4)]]
     },
```

Particle declaration

```
S[100] == {
      ClassName      -> pi1,
      SelfConjugate -> True,
      Unphysical     -> True,
      Definitions    -> {pi1   ->   - Sin[th] p1 + Cos[th] p2}
    },

S[101] == {
      ClassName      -> pi2,
      SelfConjugate -> True,
      Unphysical     -> True,
      Definitions    -> {pi2   ->   Cos[th] p1 + Sin[th] p2}
    },

 S[102] == {
      ClassName      -> p1,
      SelfConjugate -> True,
      Mass           -> {MPe1, Internal},
      Width          -> {Wpe1, 1}
    },

 S[103] == {
      ClassName      -> p2,
      SelfConjugate -> True,
      Mass           -> {MPe2, Internal},
      Width          -> {Wpe2, 1}
    },
```

Let's do it numerically instead

Mixing declaration

```
M$MixingsDescription = {
    Mix["ls"] == {
            MassBasis -> {p1, p2},
            GaugeBasis -> {pi1, pi2},
            MixingMatrix -> SM,
            BlockName -> SCALARMIXING
    }
};
```

Parameter declaration

```
MM1 == {
        BlockName     -> SCALMASSES,
        OrderBlock    -> 1,
        ParameterType -> External,
        Value         -> 1
},

MM2 == {
        BlockName     -> SCALMASSES,
        OrderBlock    -> 2,
        ParameterType -> External,
        Value         -> 100
},

MM12 == {
        BlockName     -> SCALMASSES,
        OrderBlock    -> 3,
        ParameterType -> External,
        Value         -> 0.5
},
```

No physical masses
No mixing angle

Particle declaration

```
S[100] == {
      ClassName     -> pi1,
      SelfConjugate -> True,
      Unphysical    -> True
     },

S[101] == {
      ClassName     -> pi2,
      SelfConjugate -> True,
      Unphysical    -> True
     },
```

No definitions

```
S[102] == {
      ClassName     -> p1,
      SelfConjugate -> True,
      PDG           -> 100,
      Mass          -> MPe1,
      Width         -> {Wpe1, 1}
     },

S[103] == {
      ClassName     -> p2,
      SelfConjugate -> True,
      PDG           -> 101,
      Mass          -> MPe2,
      Width         -> {Wpe2, 1}
     },
```

Physical masses
can just be defined here

Summary

Define the parameters and particles in both basises

Analytically

- Calculate the relation between both
- Implement them yourself in the definitions

Numerically

- Mixing declaration
- Numerical code calculates the relations for you

Prospects

Short term

- Further testing
- Full integration of the output in FeynRules
- Find a name

Longer term

- One-loop mass matrices
- Integrate in a mass spectrum generator