MB 1.2

Reference Manual

M. Czakon

Contents

1 Main interface

1.1 MBoptimizedRules

```
MBoptimizedRules[integrand, limit, constraints, fixedVars, options]
```

For a description of the `integrand` and `limit` see `MBcontinue`. The remaining input parameters are as follows

- `constraints:` a list of additional constraints (inequalities) specified by the user. This should usually be left empty, but might be used for experimentation in order to search for contours that might possibly give less residues.
- `fixedVars:` a list of variables, which should be considered fixed during analytic continuation. The integration variables are determined automatically from the arguments of the Γ and ψ functions.
- `options:`
  - `Level:` specifies the level up to which optimization of the contours will be performed. This option should only be used for very large calculations. Since in this case, the contours are only partially tested, the user will have to correct them himself, if poles lying on a contour are encountered. In practice, independent, small shifts should be sufficient for this purpose.

The output matches precisely the form needed in the input of `MBcontinue`, *i.e.*

```
{fixedVarRules, intVarRules}
```

Notice that this procedure not only reduces the number of residues, but also generates such contours that, during analytic continuation, no contours will start or end on a pole.
During the determination of the real parts, warning messages are generated. These can be ignored apart from the case when there is a single message

```
no rules could be found to regulate this integral
```

and the output is an empty list. In this case, the integral cannot be regulated and the user has to provide another one, *e.g.* by introducing a further regulator parameter, for example a propagator power, and performing two subsequent analytic continuations.

1.2 MBcontinue

```
MBcontinue[integrand, limit, {fixedVarRules, intVarRules}, options]
```

where the input arguments are

- `integrand:` any object accepted by `MATHEMATICA`. Notice that the singularities are determined by analyzing Γ and ψ functions only.
- `limit:` a rule, `x -> x0`, which specifies at the same time the variable, `x`, in which the analytic continuation is performed and the point, `x0`, which the user wants to reach.
- `fixedVarRules:` a list of rules giving the values of the real parts of the variables, which are not integrated over. In particular, it must contain the starting value of the variable, in which the analytic continuation is performed.
- `intVarRules:` a list of rules giving the real parts of the integration variables.
- `options:`
  - `Level:` an integer specifying the level at which the recursive analytic continuation will be stopped. By default, it is set to infinity.
  - `Skeleton:` a boolean value. If `True`, the residues will be identified, but not calculated. This is achieved by replacing all Γ and ψ functions by a dummy function `MBgam`. The purpose of this option is to quickly determine the total number of integrals. By default this option is set to `False`.
  - `Residues:` a boolean value. If `True`, the output will also contain the list of Residue points besides the actual values of the residues. This is mainly for internal use and is set by default to `False`.
  - `Verbose:` a boolean value. If `True`, the level is printed as well as the position on the list of the currently continued integral and the residue points together with the signs of the residues. This option is switched on by default.

The output is a nested list obtained by replacing, at every level, the integral to be continued by its residues and the original integral at the limit. The elements are

```
MBint[integrand, {fixedVarRules, intVarRules}]
```

objects, where the integrand can be expanded around the limit, which is placed on the `fixedVarRules` list. If the user specified a finite level, then there might also occur

```
MBitc[integrand, limit, {fixedVarRules, intVarRules}, Options]
```

objects, where “itc” stands for “integral to continue”. These are not yet regular at the limit and require further recursive analytic continuation. Furthermore, if the user set the `Residues` option to `True`, there will also be a list of

```
MBres[sign, var, val],
```

objects, which signal that there was a residue taken in the variable, `var`, at the value, `val`, with `sign`. Restricted input checking has been implemented, and as long as the input is syntactically correct, the only error that may occur is

```
contour starts and/or ends on a pole of Gamma[z]
```

In this case the procedure stops and gives an inequality for an integration variable that is sufficient to remove the problem.

1.3 MBpreselect

`MBpreselect[integrals, {x, x0, n}]`

Rejects those `MBint` objects on the `integrals` list that would vanish after expansion in the variable `x`, around the point `x0`, up to order `n`.

1.4 MBexpand

`MBexpand[integrals, norm, {x, x0, n}]`

Expands `MBint` objects on the `integrals` list around the point `x0`, in the variable `x`, up to order `n`. A normalization factor, `norm`, is included in every integrand.

1.5 MBmerge

`MBmerge[integrals]`

Merges `MBint` objects on the `integrals` list by linearity, if they have the same contours. Vanishing integrals are rejected.

1.6 MBintegrate

`MBintegrate[integrals, kinematics, options]`

where the input arguments are

- `integrals:` a list of integrals as provided by `MBexpand`.
- `kinematics:` a list of rules providing numeric values for all the parameters (usually kinematic invariants) besides the expansion variable and the integration variables. If the user is interested in Minkowski kinematics then a small imaginary part should be added. Even though this is just an approximation, it is justified by the fact, that the final result has usually much lower precision than the error introduced by such a procedure.
- `options:`
    - `NamePrefix:` by default the Fortran programs generated for integrals in more than one variable are called `MBpart1x0`, etc. where the last number is the power of the expansion variable `x` and `part1` denotes the first integral at this order. With this option one can change the prefix `MB`.
    - `PrecisionGoal, AccuracyGoal, MaxPoints, MaxRecursion, Method, WorkingPrecision:` numerical integration options as in `NIntegrate`. The defaults are respectively 4, 12, 10^6, 10^3, `DoubleExponential`, `$MachinePrecision`, and have been tuned to several problems solved with the package.
    - `MaxNIntegrateDim:` dimension threshold, above which Cuhre or Vegas will be used for the evaluation of the integrals. Should be at least 1 (default).
    - `MaxCuhreDim:` dimension threshold, 4 by default, above which Vegas will be used for the evaluation of the integrals instead of Cuhre.
    - `PseudoRandom:` by default Vegas uses Sobol quasi-random numbers. if this option is set to `True`, Mersenne Twister pseudo-random numbers will be used.

- `Complex:` by default, only the real part of the integrals is evaluated, with this option set to `True`, the imaginary part will also be given.
- `FixedContours:` contours will not be shifted if this option is set to `True`. For a detailed explanation, see `MBshiftContours` below.
- `NoHigherDimensional:` by default, the complete integration is performed within `MBintegrate`, however with this option set to `True`, 1-dimensional integrals are evaluated and the Fortran programs are prepared, but not run. This may be used to run them in parallel for example.
- `Debug:` with this option set to `True`, the Fortran programs are kept after evaluation and the value of every integral is given within `MBval[value, error, probability, part]` objects, where `value`, `error` and `probability` are given by `CUBA`, and `part` is the number of the integral. This provides a primitive means of improving the calculation by tuning only specific integrals, since the integration parameters can be easily changed in the Fortran programs.
- `ContourDebug:` with this options set to `True`, `MBshiftContours` will print contour optimization information.
- `Verbose:` by default the progress of the integration is printed to the screen. This can be switched off by setting this option to `False`.

2 Working with contours

2.1 MBcorrectContours

```
MBcorrectContours[{fixedVarRules, intVarRules}, shift]
```

shifts all the `intVarRules` by inverses of primes starting at `Prime[shift]`. This can be used to avoid poles running into contours.

2.2 MBshiftContours

```
MBshiftContours[integrals]
```

shifts the contours such that the arguments of the `Gamma` and `PolyGamma` functions are as far away from the poles as possible. It is assumed that the arguments contain only integration variables, which is the case after expansion in all fixed parameters. This function is used internally by `MBintegrate`.

```
MBshiftContours[integrals, newRules]
```

shifts the contours such that they are at the positions given on the `newRules` list.

2.3 MBshiftVars

```
MBshiftVars[integrals]
```

shifts variables in all integrals such that the exponents have just a single integration variable.

2.4 MBreplaceVar

```
MBreplaceVar[MBint[integrand, rules], var -> var0]
```

makes the replacement in the integrand and the contours, if `var0` contains `var`.

3 Barnes Lemmas

3.1 MBapplyBarnes

```
MBapplyBarnes[integrals]
```

applies Barnes Lemmas recursively.

3.2 Barnes1

```
Barnes1[MBint[integrand, {fixedVarRules, intVarRules}], z]
```

applies the first Barnes lemma to an integral, `MBint`, taking into account the integration contours.

3.3 Barnes2

```
Barnes2[MBint[integrand, {fixedVarRules, intVarRules}], z]
```

applies the second Barnes lemma to an integral, `MBint`, taking into account the integration contours.

3.4 barnes1

```
barnes1[z]
```

applies the first Barnes lemma in the variable `z`.

3.5 barnes2

```
barnes2[z]
```

applies the second Barnes lemma in the variable `z`.

4 Feynman integrals

4.1 FUPolynomials

```
FUPolynomials[integrand, momenta, kinematics]
```

gives the F and U polynomials, as well as the M and Q matrices for a Feynman integrand given by a product of propagators

`DS[k, m, n]` = $1/(k^2 - m^2)^n$

The notation is the same as in

T. Binoth and G. Heinrich, Nucl. Phys. B585 (2000) 741.

4.2 a0

```
a0[ms, l]
```

vacuum graph with one massive line to power `l`.

```
a0[ms, l1, l2]
```

vacuum graph with one massive (power `l1`) and one massless line (power `l2`).

4.3 b0

```
b0[ps, l1, l2]
```

fully massless B_0 function.

4.4 b0OS

```
b0OS[ps, l1, l2]
```

on-shell B_0 function. the power of the massive line is `l2`.

4.5 c0

```
c0[ps, l1, l2, l3]
```

fully massless C_0 function with two massless external lines. the power of the internal line that connects to both of them is `l3`.

5 Low level interface

5.1 MBrules

```
MBrules[integrand, constraints, fixedVars]
```

determines real parts of fixed and integration variables such that the arguments of all `Gamma` and `PolyGamma` functions in the integrand be positive. It is only necessary to specify the fixed variables, `fixedVars`, since the integration variables are determined automatically. The user can also specify further `constraints`.

```
MBrules[integrand, limit, constraints, fixedVars]
```

determines the contours such that during analytic continuation no contour starts or ends on a pole.

5.2 MBresidues

```
MBresidues[integrand, limit, {fixedVarRules, intVarRules}]
```

generates a list of residues obtained by continuing the integral given by the `integrand` and the vertical integration contours in the complex plane whose real parts are specified in `intVarRules`. The continuation is done to the point given in `limit`. The starting point should be given in `fixedVarRules`. The result contains either `MBint` objects where the `limit` has been attained, or

`MBitc` objects, which should be further continued. If one of the integration contours starts or ends on a pole the process is stopped and the user has to give different contours.
If the user is only interested in the number of integrals in the final result, the option `Skeleton -> True` should be set. It might happen that there are cancellations of the singular behaviour between the numerator and the denominator, in which case this number will be overestimated. With the option set, `Gamma` and `PolyGamma` functions are replaced by `MBgam`. With `Residues -> True` two lists will be created. The first contains the details of the residues in `MBres[sign, var, val]`, where `sign` is the sign of the residue, which is taken in `var` at the point `val`, and the second is as before.

5.3 MBresidue

```
MBresidue[expr, {x, x0}]
```

replacement for the `Residue` function, which is correct under the assumption that the residue is generated by `Gamma` and `PolyGamma` functions.

5.4 MBshiftRules

```
MBshiftRules[dx]
```

rules that regularize `Gamma`'s and `PolyGamma`'s with argument `m+a*dx`, where `m` is a nonpositive integer.

5.5 MBexpansionRules

```
MBexpansionRules[dx, order]
```

rules that expand `Gamma`'s and `PolyGamma`'s with argument `m+a*dx` up to given order, under the assumption that `m` is not a nonpositive integer.

5.6 MBfortranForm

```
MBfortranForm[stream, x, expression]
```

writes an `expression` of the form `x = expression` to the `stream` in Fortran form. The output of `FortranForm` is corrected to have the correct syntax of double precision constants and fit into 72 columns as required by the Fortran standard. This function is used internally by `MBintegrate`.

5.7 MBmap

```
MBmap[z, x]
```

is a mapping transformation from the range `[z - I+Infinity, z + I*Infinity]`, to `[0, 1]`, as in `1/(2*Pi*I)*Integrate[f[z + I*x], x, -Infinity, Infinity] = 1/(2*Pi)*Integrate[MBnorm[x]*f[MBmap[z, x]], x, 0, 1]` This transformation is used internally by `MBintegrate`.

5.8 MBnorm

```
MBnorm[x]
```

for a detailed description see MBmap

6 Objects

6.1 MBint

```
MBint[integrand, {fixedVarRules, intVarRules}]
```

is a Mellin-Barnes integral specified by the `integrand` and the values of the real parts of the fixed, `fixedVarRules`, and integration, `intVarRules`, variables.

6.2 MBitc

```
MBitc[integrand, limit, {fixedVarRules, intVarRules}]
```

is a Mellin-Barnes integral specified by the `integrand` and the values of the real parts of the fixed, `fixedVarRules`, and integration, `intVarRules`, variables. The integral still requires analytic continuation to the `limit`.

6.3 MBres

```
MBres[sign, var, val]
```

denotes a residue, which is taken in `var` at the point `val`, with `sign`.

6.4 MBgam

```
MBgam[z]
```

is a replacement for `Gamma` and `PolyGamma` functions used by `MBresidues`, when the `Skeleton` option is set to `True`.

6.5 MBval

```
MBval[value, error, probability, part]
```

is the value of a numerical integral, generated with `MBintegrate`. `part` is the index of the integral on the list of the integrals to evaluate.

6.6 ep

dimensional regularization parameter for the 1-loop integrals