SUMMARY NOTE FOR EUROMOD MARGINAL TAX RATE ADD-ON

Last updated: 11 August 2020

This document provides a short overview of the main features of the Marginal Tax Rate (MTR) Add-on. The description is mainly based on the following publication: Jara and Tumino (2013); as well as the following manual of the EUROMOD help: *EUROMOD add-ons and the special functions AddOn_Applic, AddOn_Pol, AddOn_Func and AddOnPar.*

The MTR Add-on allows computing marginal tax rates (MTRs) for all countries and policy years in EUROMOD. MTRs are "an indicator of the proportion of a marginal increase in earnings that is taxed away due to social insurance contributions, taxes and loss of benefit entitlement" (Jara and Tumino, 2013). They are expressed in percentage and they usually vary between 0 and 100%[1]. They are computed only for individuals declaring positive earnings and, therefore, they can be interpreted as a proxy of labour market incentives at the intensive margin of labour supply[2].

In short, the MTR Add-on complements the countries' spines by adding some extra policies which allow running and storing the results of policy systems twice: first, by running baselines' policies and, second, by rerunning them with increased earnings. Here it must be noted that earnings are increased in turn for each earner in the household. All the above is implemented in EUROMOD through a primary loop of 2 iterations and a secondary loop of as many iterations as members with positive earnings within the household (see section 2 *MTR_PREP* for more details). Results are stored individually and differences in household disposable income between the 2nd and 1st run are computed (see equation 1).

$$(1) \qquad \boldsymbol{MTR} = 1 - \left(\frac{{Y^1}_{HH} - {Y^0}_{HH}}{{E^1}_i - {E^0}_i}\right) = 1 - \left(\frac{\Delta Y_{HH}}{\Delta E_i}\right),$$

[1] Although there might be cases where MTRs can be negative or above 100%. The latter arises from decreases in household disposable income which overcome the increase in earnings, while the former are due to increases of household disposable income of higher magnitude than the increase in earnings (see Jara & Tumino, 2013 for more information).

[2] A marginal increase in earnings can occur through different sources such as working more hours in the same job, moving or promoting to a better-paid job, getting bonus payments on top of the base salary, etc. (Adam and Browne, 2010).

where $Y^1{}_{HH}$ is the household disposable income in the 2nd run after the increase in earnings, $Y^0{}_{HH}$ is the household disposable income in the 1st run and ΔE_i is the individual increase in earnings.

Moreover, given the advantage in EUROMOD of decomposing disposable income into its different detailed components, MTRs are decomposed in the Add-on as follows: original income[3], public pensions (*mtrpc_pen*), means-tested (*mtrpc_benmt*) and non means-tested benefits (*mtrpc_bennt*), taxes (*mtrpc_tax*) and social insurance contributions paid by the employee (*mtrpc_sicee*), self-employed (*mtrpc_sicse*) and others (*mtrpc_sicot*)[4].

SPECIFIC ASPECTS AND ASSUMPTIONS APPLIED IN THE ADD-ON

MTRs are only computed for those individuals with positive earnings (*ils_earns > 0*).
Full take-up and full tax compliance are assumed.
The default marginal increase of earnings is of 3% of gross earnings in the baseline.

STRUCTURE OF THE MTR ADD-ON

EUROMOD add-ons are structured in a similar way as EUROMOD country models, i.e. they have a policy spine consisting of several policies. The policies in the add-ons do not describe the rules of taxes or benefits of the baseline but perform additional operations, i.e. increasing earnings by a specific margin and recalculating tax liabilities and benefit entitlements.

The MTR Add-on is composed of a general system (*MTR*) which applies to some countries (CZ, HU, LU, SE) and several additional country-specific systems (*MTR_=cc=*, where *=cc=* stands for the country acronym) for which some country-specific adjustments are needed. In any case, all systems are composed of 5 policies with different functionalities. They are described below:

1. *ao_control_MTR*

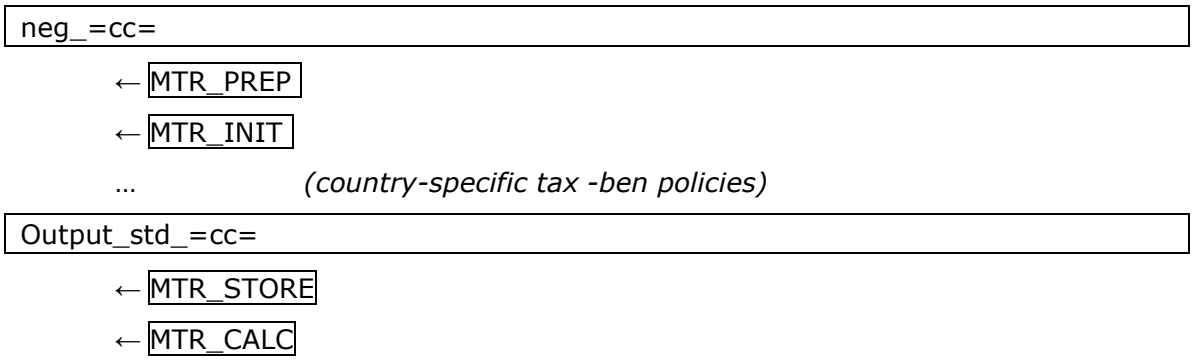
This policy defines which country systems the add-on can be run with (*func_AddOn_Applic*) and which additional MTR policies are added to the policy spine and where (*func_AddOn_Pol*). Note that the policies themselves are defined in separate policies which are described in the subsequent sections. Besides, the MTR Add-on is intended to work with all policy systems-datasets combinations.

[3] This component will be 0 as the change in original income is equal to the marginal increase in earnings.

[4] Paid, for example, by pensioners.

The placement of the Add-on in the country-specific policy spines is described in Figure 1. In general, MTRs calculations starts right after the *neg_=cc=* policy and they mainly involve the running of each country-specific tax and benefit policies without and with the increase in earnings[5]. Tax compliance and benefit take-up adjustments are switched off at this point, i.e. full compliance with the policy rules and full take-up of benefits are assumed (*func_AddOn_Extension*). Moreover, the policy *ao_control_MTR* restricts the standard output policy to the baseline results only (*func_AddOn_Par*) so they are not overwritten after the marginal increase in earnings.

Figure 1. MTR Add-on location in the country spine.



2. *MTR_PREP*

MTR_PREP applies some general adjustments needed before the looping of tax-benefit policies: e.g. define the marginal increase in earnings (3% by default); define new tax units (TUs), income lists (ILs) and variables, etc. Before the loops start, baseline's gross earnings (i.e. *ils_earns* and all its components) are stored at the individual level (*func Store*).

Then, two loops are defined/started in order to calculate tax liabilities and benefit entitlements before and after the increase in earnings:

- Primary loop (named BASE) (*func_Loop*): 1st run is the baseline, 2nd run is with increased earnings. The loop starts right after the policy *MTR_PREP* and finishes before *MTR_CALC.*
- Secondary loop (named MTR) (*func_UnitLoop*) within the 2nd run of the primary loop: earnings are increased for each person in the household in turn, therefore, the number of runs depends how many people with positive earnings there are in the household. The secondary loop goes from *MTR_INIT* to *MTR_STORE*.

Figure 2 in the Appendix depicts graphically the way taxes and benefits are looped so to compute MTRs.

[5] Note here that the location policies are inserted is slightly different for some countries (LT & SI) as some previous calculations are needed before running the MTR Add-on.

3. *MTR_INIT*

This policy allows, mainly, increasing earnings (*ils_earns* and all its components) for one person at a time in each household (*func_IlVarOp*). This is achieved through the use of the parameter *IsCurElig_mtr* which identifies, for each household, all those individuals with positive earnings, i.e. the variable is true for individuals with *ils_earns > 0* and false for all others.

Besides, any country-specific adjustments, e.g. recalculating certain variables, switching off certain policies that should not be rerun for technical reasons (*func_ChangeParam*), etc. are applied in this policy.

Note that in this policy the name of the EUROMOD standard output is renamed to =sys=_base_mtr, where =*sys*= stands for the name of the policy system. This needs to be done to clearly distinguish the standard MTR output from the standard baseline output (=sys=_std), as the former assumes full take-up and full compliance with the policy rules, whereas the latter might include any of the above-mentioned adjustments.

4. *MTR_STORE*

This policy stores the baseline results (*func_Store, postloop=base*); by default all standard income lists (*ils_*)* and their components. Note that all variables are stored at the individual level.

Additionally it stores the results with increased earnings (*func_Store, postloop=mtr*); by default all standard income lists (*ils_* *) and their components:

- note there are two types of IL/variables: (i) for each MTR run (with a suffix _MTR1, _MTR2, etc.) and (ii) 'composite' IL/variables (with a suffix _MTR) containing information only from the relevant run for each person.
- variables for a specific MTR run have missing values (i.e. void) for people from households not included in that run; 'composite' variables have missing values for people with no earnings.
- variables for a specific MTR run are stored at the individual level independent of level-parameters, while all 'composite' variables (except earnings) are stored at the household level (as *level=tu_household_mtr* and *il2_level=tu_individual_=cc=*).

Finally, earnings variables are restored to its original value at the beginning of each iteration (*func_Restore*).

5. *MTR_CALC*

The last policy of the MTR Add-on defines and calculates MTR-related indicators:

- change in earnings (*ils_earns_diff*)
- MTR in percentage (*mtrpc*)
- MTR components in percentage, such that *mtrpc = mtrpc_pen + mtrpc_benmt + mtrpc_bennt + mtrpc_tax + mtrpc_sicee + mtrpc_sicse + mtrpc_sicot*

MTR_CALC also generates a separate output for MTR results *(=sys=_mtr,* where =*sys*= stands for the name of the policy system), outputting, by default, the initial earnings (*ils_earns_backup*), initial employee earnings (*yem_backup*), all (standard) income lists in the baseline and MTR-related indicators, e.g. variables named *mtr**, **mtr* or **diff.*

- note that missing values for variables from the MTR loop (see *MTR_STORE*) are replaced with 9999999.
- Besides, additional variables can be outputted (see *Appendix: adding extra variables to the MTR output* for guidelines).

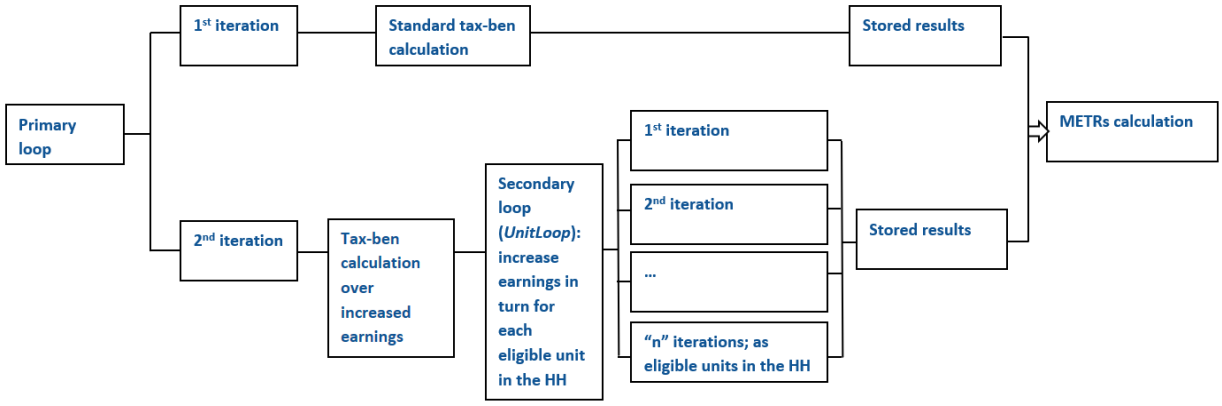
REFERENCES

Adam, S. and Browne, J. (2010) 'Redistribution, Work Incentives and Thirty Years of UK Tax and Benefit Reform', IFS Working Paper 10/24.

Jara, HX and Tumino, A (2013) 'Tax-benefit systems, income distribution and work incentives in the European Union.' The International Journal of Microsimulation, 6. 27 - 62. ISSN 1747-5864.

APPENDIX

Figure 2. Looping of tax-ben policies in the MTR Add-on



Adding extra variables to the MTR output

Note that in general, output function parameters *ilgroup* and *vargroup* do not include stored results (for technical reasons). There are two exceptions which are already included in the default setup: (standard) income lists for the baseline run (*ilgroup=*base1*) and 'composite' IL/variables for the MTR run (both with ***vargroup***=**mtr*).

To output additional information:

- Initial earnings variables
    - If the variable is not a component of *ils_earns* then add to *func_Store* in *MTR_PREP*, e.g. *varX=yse00*; note that if an income list is stored its components are stored as well.
    - Add to *func_DefOutput* in *MTR_CALC*, e.g. *varX=yse00_backup*
- Baseline results[6]

[6] As earning variables should be part of disposable income, this is another way of inputting initial earnings variables (as yem_backup should equal yem_base1). Furthermore, the usual baseline output is written into a separate variable.

- Add the variable/IL to *func_Store* in *MTR_STORE* (*postloop=base*) unless already included, e.g. *varX=temp_rand*, *ilX=il_SA_means*; note that if an income list is stored its components are stored as well.
- Add to *func_DefOutput* in *MTR_CALC*, e.g. *varX=bsa00_s_base1*, *ilX=il_SA_means_base1.*

- Results from a specific MTR run
  - Add the variable/IL to *func_Store* in *MTR_STORE* (*postloop=mtr*) unless already included, e.g. *varX=temp_rand*, *ilX=il_SA_means*; note that if an income list is stored its components are stored as well.
  - Add to func DefOutput in MTR_CALC, e.g. *varX=bsa00_s_mtr2*, *ilX=ils_dispy_mtr3.*

- 'Composite' results from the MTR loop
  - Add the variable/IL to *func_Store* in MTR_STORE (*postloop=mtr*) unless already included, e.g. *varX=temp_rand*, *varX=il_SA_means*; note that (i) both use varX-parameter, (ii) 'composite' results are not constructed for income list components (and hence need to be specified separately).
    - note that no changes are needed for *func_DefOutput* in *MTR_CALC* as all stored variables ending with "_mtr" are already outputted in the default setup.