Accurate Income Measurement for the Assessment of Public Policies (AIM-AP)
Contract no 028412.

Deliverable 3.7:

Users' guide to simulate indirect taxes in EUROMOD

Andre Decoster[1], Dirk Verwerft[1]

Abstract: **This deliverable contains the guidelines to impute expenditure data in EUROMOD datasets and to perform a joint simulation of direct and indirect taxation. Three do-files will be presented that facilitate the process.**

I. INTRODUCTION

This deliverable contains a detailed modus operandi for the imputation of expenditure data into a EUROMOD dataset as well as the simulation of changes in both direct and indirect taxes. The statistical program STATA is used for the entire process. We start with an overview of the whoel procedure in section II. Section III discusses the necessary input information and the structure of the datasets. The imputation step and the working of the matching do-file are treated in section IV. Section V and VI give an overview of the simulation process.

II. SEQUENCE OF A SIMULATION OF INDIRECT TAXES IN EUROMOD

The calculation of indirect taxes in EUROMOD can be separated in two distinct steps:

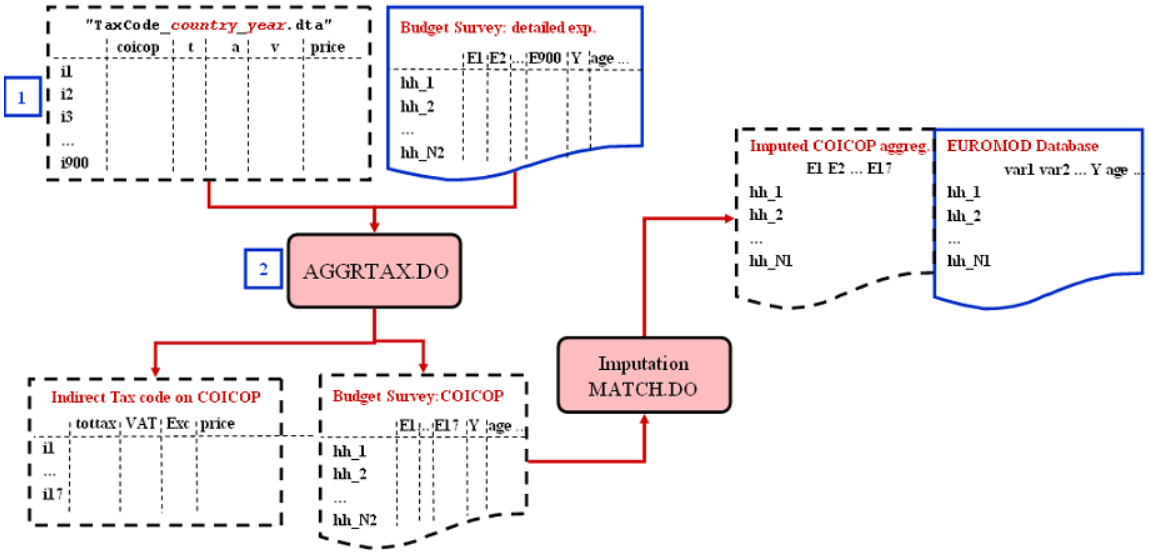
1. We need detailed expenditures for the EUROMOD households; we call this the *imputation* step.
2. We want to calculate indirect tax liabilities for these households. We call this the *simulation* step.

[1] Center for Economic Stduies - University of Leuven, Belgium.

The imputation step obviously precedes the simulation step. It needs an expenditure survey which – in most cases – is not part of EUROMOD. Ideally, once this imputation has been carried out, and we have enriched the EUROMOD dataset with expenditures, the budget survey is no longer needed. Yet, and this is an important remark for future development of the design of EUROMOD, as long as statutory indirect tax rates are defined at the lowest level of the itmeisation in the budget survey, we will also need the budget survey in the simualtion step. We descirbe the data requirements in detail in section III. First we give a schematic overview of the imputation and simulation step in two graphs. We then describe the imputation and simulation separately

The scheme in Figure 1 below illustrates the imputation step, described in detail in section IV. The process starts from a file containing detailed expenditure information per household (the budget survey) and a STATA-file containing indirect tax rates and the COICOP aggregate to which each expenditure item belongs. By making changes in this file one can manipulate the aggregation process of the detailed expenditure items, and introduce chnages in indirect taxes (either the VAT-rate, denoted by *t*, the excise rate, denoted by *a*, or the ad valorem rate *v*). As described in WP 3.1 the consumer price is needed for those commodities where there is an excise rate.

Figure 1: Calculation of aggregate indirect tax rates and imputation step

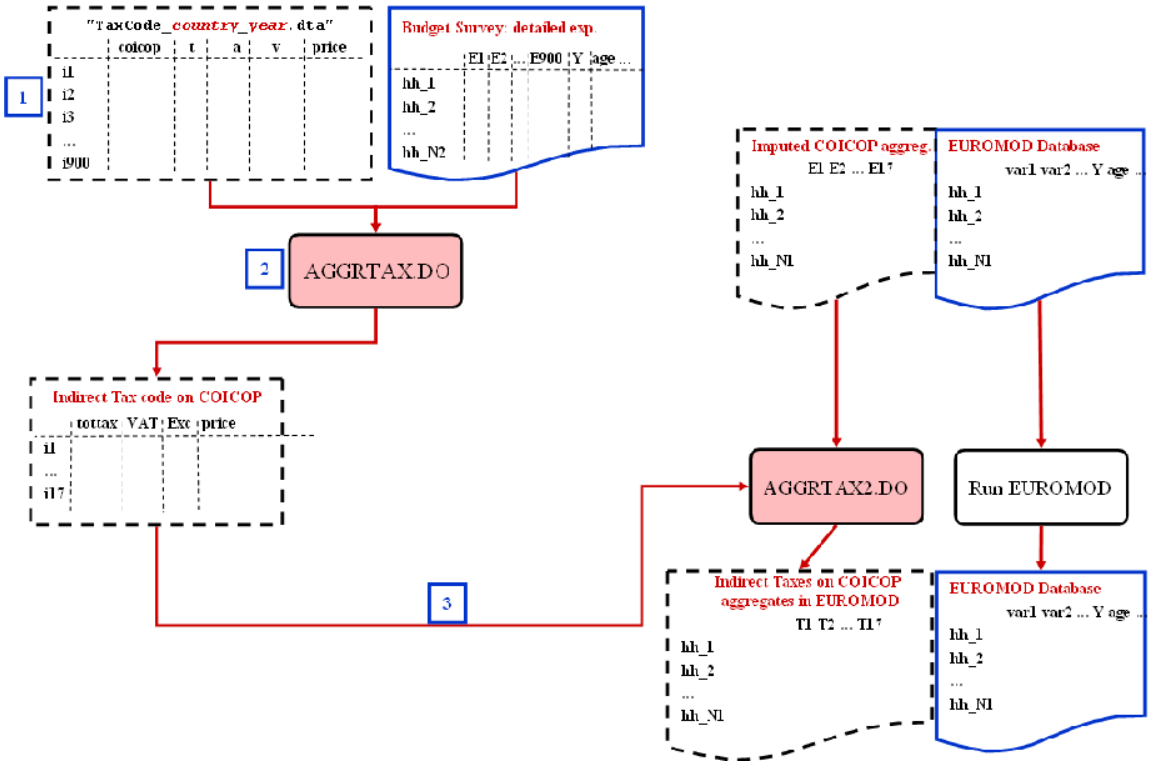


The STATA-program **`aggrtax.do`** transforms the data in the budget survey into expenditure aggregates of the COICOP scheme. It also produces a file with indirect tax rates on these aggregates. The STATA-program **`match.do`** imputes the aggregated expenditure information in the target EUROMOD dataset.

The scheme in Figure 2 below represents the simulation step, explained in more detail in sections V and VI. Note that indirect tax liabilities in EUROMOD are calculated by means of average tax rates on the COICOP aggregates (differentiated of course between VAT and excise). Since policy changes are introduced at the detailed level of statutory indirect tax rates or excise duties in the STAT-file at the upper left corner of the graph, the program **`aggrtax.do`,** and more importanly also the orginal budget survey are also needed in the simulation step. One way to get around this in future developments of EUROMOD, is to replace the complete budget survey in this step by one vector of average expenditures across households (or by groups of households) at the most detailed level of itemisation. This would really decouple indirect tax calculations in EUROMOD from the need to dispose of the budget survey itself.[2]

To obtain government budget neutrality, the simulation depicted in Figure 2 is tried for different indirect tax changes (hence there is no automatic revenue neutrality built in).

Figure 2: Sequence to simulate indirect taxes in EUROMOD



[2] This is the way the Belgian microsimulation model ASTER is built up.

III. DATA REQUIREMENTS

Basically, there are three different data sources necessary for the process.

- The first source of data contains a **link between the tax code and the detailed expenditure aggregates**, used to calculate indirect taxes. The dataset has to contain a variable "itemid" that links the line to the corresponding variable in the detailed expenditure dataset, so that itemid equal to 1 means that the line pertains to the variable "e_1" (cf. infra). Furthermore, there has to be a variable containing the COICOP aggregate (coicop1) to which the user wants to assign this item, the VAT rate (t), the excise rate (a) and the ad valorem tax rate (v). Finally, a variable containing consumer prices (q) has to be included: this variable equals 1 for the consumption items without excises, but for the items with excises it specifies on which original price the excise is levied.

- The **expenditure data** have to be transformed into two different datasets using `aggrtax.do`.

  - The first one has to contain a household identification variable, a weight variable and the **expenditures in detailed categories**. The expenditure variables have to be named "e_1", "e_2" etc. This is in most cases just the original dataset.

  - The second one has to contain the **variables in common** with the EUROMOD dataset (cf. supra), as well as the **expenditures grouped per COICOP aggregate**. This dataset is created from the first one using `aggrtax.do`.

- A third source is the **EUROMOD dataset** that will be used for the simulation. This has to be transformed into STATA, and a harmonization process has to take place between its variables common with that of the expenditure dataset. Qualitative variables have to contain the same categories and quantitative variables have to be expressed in the same basic unit. It is recommended to inspect especially the disposable income variable for equality between the datasets. A visual comparison with QQ-plots is often the most instructive (see our description in WP 3.4). The method described here has been devised to be more or less robust with respect to differences in the tails of the distributions, as long as those are not too heavy.

IV. IMPUTATION OF EXPENDITURE INFORMATION

- The imputation step uses as only **input the EUROMOD and aggregate expenditure datasets** where corresponding variables have the same name. This step performs an Engel curve based imputation for different subgroups of the population based on their zero expenditure items (details in workpackage 3.5). Appendix 1 contains the do-file **`match.do`** that can be used as an example of the imputation. This do-file does not work for datasets with more than 11000 households due to STATA's calculation constraints.

- There are two main **output files** to this program.

  - The first is the **EUROMOD dataset household id's with the imputed COICOP expenditure data**. It is on this file that indirect taxes are calculated for the EUROMOD households by using the output file of `aggrtax.do` containing the average tax rate per COICOP-aggregate. The do-file in appendix 2 (`aggrtax2.do`) is a slimmed down version of `aggrtax.do` which only serves this purpose.

  - The second output file (not in the grpah) contains the **parameters of the Engel curves**: observations 1 and 2 contain the estimated parameters for total nondurable and durable expenditures respectively and the next observations give the parameters for the nondurable budget shares of the other aggregates per subgroup (the budget shares for the first subgroup, then for the second etc.). The variables are ordered as follows: the first four variables are the parameters for the dummies for the zero expenditure categories: smoker, renter, public transport and education. The next variables are the parameters for the regressors given by the user in the input part. The last three variables are the parameters for 1) disposable income in the case of durable and nondurable expenditures and the logarithm of nondurable expenditures for the budget shares, 2) the square of the latter, and 3) the constant term. See WP 3.5 for details.

V. PREPARING THE SIMULATION STEP

As already suggested above, a pure indirect tax simulation exercise can be carried out straightforwardly by using the do-file `aggrtax2.do` (in appendix 2). Therefore we do not describe this kind of simulation in further detail here. We do however describe more fully the simulations of WP 3.6 where a change in direct taxation or social security contributions is implemented and the government wishes to ensure revenue neutrality through taxation on

consumption. This necessiates a **stepwise procedur**e in the sense that first the change in labour taxation and social security contributions is simulated with EUROMOD. The compensating change in indirect taxes can then be calculated conditional upon the result.

- Hence, as primary input in the simulation step, **monetary variables for all households under the baseline and the reform** have to be stored in a joint dataset (**merge EUROMOD baseline and reform output**). Wherever necessary, they have to be put in annual figures. The dataset also has to include the imputed expenditure and indirect tax information, together with the earlier mentioned **common variables**.

- The datasets with the **detailed expenditure information and the tax code** of the first step are also required in order to calculate the new indirect tax rates. A last necessary dataset consists of the **Engel curve parameter estimates** derived in the previous step. This dataset is an output of the imputation program.

VI. SIMULATION

The two simulation do-files calculate the rise in standard VAT rate necessary to obtain government neutrality. The difference between them is that **`simulation1.do`** (see appendix 3) does not include behavioural changes to the increase in consumer prices but merely **assumes constant budget shares**, while the second do-file (appendix 4) allows for **Engel curve driven changes** in budget shares as a result of changes in total nondurable expenditures. It is only for the latter program that the parameter data file is used.

- The **input** consists of the four datasets described in the previous paragraph. The programs iteratively increment the standard VAT rate by a certain amount that can be defined by the user, until the extra revenue generated by the tax raise exceeds the amount lost by the decrease in labour taxation. An increase in labour taxation can also be implemented with a few minor adaptations.

- The **influence of the increased VAT rate on the total indirect tax rate**s is calculated with an aggrtax-like module in the program. Then the **new expenditures** are calculated: savings stay constant in nominal terms, durables in real terms, and nondurable expenditure shares are kept constant in the first simulation program and changed according to the estimated Engel curves in the second program. Negative budget shares are set to zero and the shares are standardized so that they sum back to one. Afterwards, the **new taxes** are calculated on the new expenditures with the new rates. If the extra revenue generated does not exceed the lost amount of money, these steps are repeated with a higher VAT rate.

During the process, **extra diagnostic variables** are created, such as WG (the welfare effect for the household), the price effect and the change in total nondurable expenditures.

APPENDIX 1: STATA-PROGRAM MATCH.DO

```
*imputes expenditure data from a source file into a target file
*in a first step total expenditure and durable consumption are imputed
*(using common variables, log(inc) and log^2(inc))
*in a second step nondurable budget shares are estimated (on total
*nondurable expenditure) and imputed
*creates file with parameters (line1 total nondurable expenditure,
*line2 durable expenditure, then budget shares per subgroup)
*zeroes on tobacco, public transport and education consumption and on
*rents are replicated using probit estimations

*********************************************************************
*Input part
*********************************************************************

*specify country
local country = "BE"

*specify tax system year
local year = "2003"

*is homeproduction (coicop aggregate 99) present in the dataset? (if
*yes, type "y")
local homeprod = "n"

*specify dataset paths
*source and target are obvious, parset is the dataset where engel curve
*coefficients will be stored
local source = "C:\Documents and
Settings\n07072\Desktop\wp3.5\\`country'\input\source_`country'_`year'.
dta"
local target = "C:\Documents and
Settings\n07072\Desktop\wp3.5\\`country'\input\target_`country'_`year'.
dta"
local parset = "C:\Documents and
Settings\n07072\Desktop\wp3.5\\`country'\output\parameters_`country'_`y
ear'.dta"
local out = "C:\Documents and
Settings\n07072\Desktop\wp3.5\\`country'\output\match_`country'_`year'.
dta"

*specify non-income matching variables and give their number
if "`country'" == "HU" {
      local ncoef = 12
      local varlist = "male selfemp emp unemp pens higheduc secondary
age age2 np na nch"
}

if "`country'" == "IE" {
      local ncoef = 12
      local varlist = "male selfemp emp unemp pens higheduc secondary
age age2 np na nch"
}
```

```
if "`country'" == "BE" {
      local ncoef = 14
      local varlist = "male wa br selfemp emp unemp pens higheduc
secondary age age2 np na nch"
}

*specify an upper boundary for max(number of variables, number of
*observations)
*if matsize > 11000, e.g. for the UK, another version of this program
is needed
set matsize 11000

*********************************************************************
*Generic part
*********************************************************************

*create common variables in source
use "`source'"
local ncoef1 = `ncoef' + 7
capture drop age2
gen age2 = age^2
capture drop inc
gen inc = disp_inc
capture drop inc2
gen inc2 = inc^2
capture drop totexpnondur
egen totexpnondur = rowtotal(exp*)
replace totexpnondur = totexpnondur - exp98
foreach var of varlist exp* {
      local i = substr("`var'", 4,.)
      capture drop w_`i'
      gen w_`i' = exp`i'/totexpnondur
}
drop w_98
capture drop lntotexp
gen lntotexp = log(totexpnondur)
capture drop lntotexp2
gen lntotexp2 = lntotexp^2

*create category dummies for zero expenditures and initialize weight
*variable for subgroup differencing
capture drop smoker
gen smoker = w_3>0
capture drop renter
gen renter = w_6>0
capture drop pubtr
gen pubtr = w_10>0
capture drop edu
gen edu = w_13>0
capture drop weights
gen weights = 0

*probit regression step
local nvars = `ncoef' + 3

probit smoker `varlist' inc inc2
matrix Asm = e(b)
```

```
matrix Asm = Asm[1,1..`nvars']
qui su smoker
local prop_smoker = r(mean)

probit renter `varlist' inc inc2
matrix Are = e(b)
matrix Are = Are[1,1..`nvars']
qui su renter
local prop_renter = r(mean)

probit pubtr `varlist' inc inc2
matrix Apu = e(b)
matrix Apu = Apu[1,1..`nvars']
qui su pubtr
local prop_pubtr = r(mean)

probit edu `varlist' inc inc2
matrix Aed = e(b)
matrix Aed = Aed[1,1..`nvars']
qui su edu
local prop_edu = r(mean)

*create group variable and sum of groups dummies variable (degree of
*similarity)
capture drop som
gen som = smoker + renter + pubtr + edu
capture drop group
gen group = smoker + 2*renter + 4*pubtr + 8*edu

save "`source'", replace

*regress total nondurable expenditure and initialise parameter matrix
regress totexpnondur `varlist' inc inc2
matrix A = e(b)
matrix param = [0,0,0,0,A]

*regress durable expenditure
regress exp98 `varlist' inc inc2
matrix B = e(b)
matrix C = [0,0,0,0,B]
matrix param = [param\C]

use "`target'", clear

*create common variables in target
capture drop constant
gen constant = 1
capture drop age2
gen age2 = age^2
capture drop inc
gen inc = disp_inc
capture drop inc2
gen inc2 = inc^2
capture drop lninc
gen lninc = log(disp_inc)
capture drop lninc2
gen lninc2 = lninc^2
```

```
mkmat `varlist' inc inc2 constant, matrix(X)

*impute total nondurable expenditure and durable expenditure, delete
*observation if total nondurable expenditure are smaller than 1
matrix b = A'
matrix Y = X*b
capture drop totexpnondur
svmat Y
rename Y1 totexpnondur
matrix b = B'
matrix Y = X*b
capture drop durables
svmat Y
rename Y1 durables
replace durables = 0 if durables < 0
drop if totexpnondur <=0
mkmat `varlist' inc inc2 constant, matrix(X)

*create expenditure variables in target
capture drop lntotexp
gen lntotexp = log(totexpnondur)
capture drop lntotexp2
gen lntotexp2 = lntotexp^2

*impute subgroups and set proportion of smokers, renters, ... in target
*equal to proportion in source by changing cut-off value

matrix b = Asm'
matrix Y = X*b
capture drop smoker
svmat Y
rename Y1 smoker
replace smoker = normal(smoker)
qui su smoker
local gemid = r(mean)
replace smoker = (smoker/`gemid')*`prop_smoker'
capture drop smoker1
gen smoker1 = uniform()
replace smoker = (smoker1 < smoker)
drop smoker1

matrix b = Are'
matrix Y = X*b
capture drop renter
svmat Y
rename Y1 renter
replace renter = normal(renter)
qui su renter
local gemid = r(mean)
replace renter = (renter/`gemid')*`prop_renter'
capture drop renter1
gen renter1 = uniform()
replace renter = (renter1 < renter)
drop renter1

matrix b = Apu'
matrix Y = X*b
```

```
capture drop pubtr
svmat Y
rename Y1 pubtr
replace pubtr = normal(pubtr)
qui su pubtr
local gemid = r(mean)
replace pubtr = (pubtr/`gemid')*`prop_pubtr'
capture drop pubtr1
gen pubtr1 = uniform()
replace pubtr = (pubtr1 < pubtr)
drop pubtr1

matrix b = Aed'
matrix Y = X*b
capture drop edu
svmat Y
rename Y1 edu
replace edu = normal(edu)
qui su edu
local gemid = r(mean)
replace edu = (edu/`gemid')*`prop_edu'
capture drop edu1
gen edu1 = uniform()
replace edu = (edu1 < edu)
drop edu1

*create group variable in target and determine number of groups
capture drop group
gen group = smoker + 2*renter + 4*pubtr + 8*edu
qui su group
local maximum = r(max)

*initialise budget share variables in target
forv i = 1/15 {
      capture drop w_`i'
      gen w_`i' =0
}
if "`homeprod'" == "y" {
      capture drop w_99
      gen w_99 = 0
}
save "`out'", replace

*estimate and impute budget shares for each subgroup
forv j = 0/`maximum' {
      *determine subgroup
      local edu1 = int(`j'/8)
      local hulp = mod(`j',8)
      local pubtr1 = int(`hulp'/4)
      local hulp = mod(`hulp',4)
      local renter1 = int(`hulp'/2)
      local hulp = mod(`hulp',2)
      local smoker1 = `hulp'
      local som1 = `edu1' + `pubtr1' + `renter1' + `smoker1'

      *attribute weights to observations in source set wrt their
subgroup
```

```
      use "`source'", clear
      replace weights = (smoker <= `smoker1')*(renter <=
`renter1')*(pubtr <= `pubtr1')*(edu <= `edu1')*(2^(som))

      *take dummy variables in regression according to subgroup
      local varlist1 = ""
      if `smoker1' == 1 {
            local varlist1 = "`varlist1' smoker"
      }
      if `renter1' == 1 {
            local varlist1 = "`varlist1' renter"
      }
      if `pubtr1' == 1 {
            local varlist1 = "`varlist1' pubtr"
      }
      if `edu1' == 1 {
            local varlist1 = "`varlist1' edu"
      }
      save "`source'", replace

      *regression and imputation step
      local teller = 0
      foreach var of varlist w_* {
            local i = substr("`var'",3,.)
            if((`i'!=3 | `smoker1'==1)&(`i'!=6 | `renter1'==1)&(`i'!=10
| `pubtr1'==1)&(`i'!=13 | `edu1'==1)){
                  use "`source'", clear
                  qui regress `var' `varlist1' `varlist' lntotexp
lntotexp2 [fw=weights]
                  matrix A = e(b)
                  mat param1 = J(1,4,0)
                  local optellen = 1
                  if `smoker1'==1 {
                        mat param1[1,1] = A[1,`optellen']
                        local optellen = `optellen' +1
                  }
                  if `renter1' ==1 {
                        mat param1[1,2] = A[1,`optellen']
                        local optellen = `optellen' +1
                  }
                  if `pubtr1' ==1 {
                        mat param1[1,3] = A[1,`optellen']
                        local optellen = `optellen' +1
                  }
                  if `edu1' ==1 {
                        mat param1[1,4] = A[1,`optellen']
                        local optellen = `optellen' +1
                  }
                  local begin = `optellen'
                  local einde = `optellen'+`ncoef' + 2
                  mat param1= [param1, A[1,`begin'..`einde']]
            }
            else {
                  local ncoef1 = `ncoef' + 7
                  mat param1 = J(1,`ncoef1',0)
            }
            mat param = [param\param1]
```

```
            use "`out'", clear
            if((`i'!=3 | `smoker1'==1)&(`i'!=6 | `renter1'==1)&(`i'!=10
| `pubtr1'==1)&(`i'!=13 | `edu1'==1)){
                  mkmat `varlist1' `varlist' lntotexp lntotexp2
constant, matrix(X)
                  matrix b = A'
                  matrix Y = X*b
                  svmat Y
                  replace w_`i' = Y1 if group == `j'
                  replace w_`i' = 0 if w_`i' == .
                  drop Y1
            }
            save "`out'", replace
      }
}

*create parameter datafile
tempfile parameters
svmat param
keep param*
drop if param1 ==.
save "`parset'", replace

use "`out'", clear
/*
*normalise imputed budget shares
capture drop total
egen total = rowtotal(w_*)
foreach var of varlist w_* {
      replace `var' = `var'/total
}
drop total
*/
*create expenditures
foreach var of varlist w_* {
      local i = substr("`var'",3,.)
      capture drop exp`i'
      gen exp`i' = `var'*totexpnondur
}
drop if totexpnondur < 1
capture drop exp98
rename durables exp98
capture drop Y1
capture drop total

save "`out'", replace
```

APPENDIX 2: STATA-PROGRAM AGGRTAX2.DO

```
************************************************
* Input part                                   *
************************************************

*define country
local country = "UK"

*country specific information
if "`country'" == "UK" {
      *year of budget survey and taxcode
      local year = 2003
      *number of aggregates
      local nagg = 16
      *number of durable aggregates
      local ndur = 1
      *number of homeproduction aggregates
      local nhome = 0
}
if "`country'" == "BE" {
      *year of budget survey and taxcode
      local year = 2003
      *number of aggregates
      local nagg = 16
      *number of durable aggregates
      local ndur = 1
      *number of homeproduction aggregates
      local nhome = 0
}
if "`country'" == "HU" {
      *year of budget survey and taxcode
      local year = 2005
      *number of aggregates
      local nagg = 17
      *number of durable aggregates
      local ndur = 1
      *number of homeproduction aggregates
      local nhome = 1
}

*path to matched data
local match = "C:\Documents and
Settings\n07072\Desktop\wp3.5\\`country'\output\match_`country'_`year'.
dta"
*path to detailed taxcode
local taxcode = "C:\Documents and
Settings\n07072\Desktop\wp3.6\\`country'\input\taxcode_`country'_`year'
.dta"
*variable containing coicop aggregates in taxcode
local varcat = "coicop1"
*path to detailed expenditure dataset
```

```
local exp = "C:\Documents and
Settings\n07072\Desktop\wp3.6\\`country'\input\HBSdetail_`country'_`yea
r'.dta"

*!!!!!!!!!!!!!! detailed items in taxcode file should have the same
order as the expenditure variables in the detailed expenditure dataset

local outputtax = "C:\Documents and Settings\n07072\Desktop\aggregates
prices and taxes_`country'.dta"
local expandtax = "C:\Documents and Settings\n07072\Desktop\exp and
taxes_`country'.dta"
local output = "C:\Documents and
Settings\n07072\Desktop\match_`country'_`year'_taxes.dta"


*set max matrix size
set matsize 2000

************************************************
* Generic part                                 *
************************************************

*define local parameters concerning number of aggregates
local hp = "n"
if `nhome' > 0 {
      local hp = "y"
}
local nndagg = `nagg' - `ndur'
local nh = `nndagg' - `nhome'

*construct variable lists for nondurables-nonhomeproduction, for
durables and for home production variables
*construct vector of exp-subscripts
matrix subscripts = J(`nagg',1,0)
use "`match'", clear
local varlist1 = ""
local varlist2 = ""
local varlist3 = ""
local count = 1
foreach var of varlist exp* {
      if (substr("`var'",4,2) == "98") {
            local varlist2 = "`varlist2' `var'"
            local i = substr("`var'",4,.)
            mat subscripts[`count',1] = `i'
            local count = `count' + 1
      }
      else {
            if (substr("`var'",4,2) == "99") {
                  local varlist3 = "`varlist3' `var'"
                  local i = substr("`var'",4,.)
                  mat subscripts[`count',1] = `i'
                  local count = `count' + 1
            }
            else {
                  local varlist1 = "`varlist1' `var'"
                  local i = substr("`var'",4,.)
                  mat subscripts[`count',1] = `i'
```

```
                local count = `count' + 1
            }
      }
}

*retrieve rates and prices on desaggregate level
use "`taxcode'", clear
sort itemid
local n_items = _N
mkmat `varcat', matrix(coicop)
mkmat t, matrix(t)
mkmat v, matrix(v)
mkmat q, matrix(q)
mkmat a, matrix(a)

**calculate producer price
capture drop p
gen p = ((1-(1+t)*v)/(1+t))*q-a
mkmat p, matrix(p)

*calculate aggregate prices and tax rates

** initialisatie variabelen
use "`exp'", clear
forv i = 1/`nagg' {
      local j = subscripts[`i',1]
      capture drop exp_`j'
      capture drop VAT_`j'
      capture drop EXC_`j'
      gen exp_`j' = 0
      gen VAT_`j' = 0
      gen EXC_`j' = 0

}
save "`expandtax'", replace


** calculate total expenditures and taxes
quietly {
forv i = 1/`n_items' {
      local k = coicop[`i',1]
      local alpha = a[`i',1]/p[`i',1]
      local denom0 = 1-(1+t[`i',1])*v[`i',1]
      local tau_t0 = (t[`i',1]*(1+`alpha'+v[`i',1])+v[`i',1])/`denom0'
      local tau_a0 = `alpha'/`denom0'
      local tau0 = `tau_t0'+`tau_a0'
      replace exp_`k'=exp_`k'+e_`i'
      replace VAT_`k'=VAT_`k'+(`tau_t0'/(1+`tau0'))*e_`i'
      replace EXC_`k'=EXC_`k'+(`tau_a0'/(1+`tau0'))*e_`i'
}
}
keep hhid weight exp* VAT* EXC*
save "`expandtax'", replace

collapse (sum) exp* VAT* EXC* [pw=weight]
mkmat exp_*, matrix(EXP0)
mkmat VAT_*, matrix(VAT0)
```

```
mkmat EXC_*, matrix(EXC0)
mat TAX0 = VAT0+EXC0
drop exp* VAT* EXC*

tempfile taxes
svmat subscripts
rename subscripts1 coicop
gen tau0=0
gen tau_t0=0
gen tau_a0=0
gen q0=0
gen taxcons0 = 0
gen taxcons_t0=0
gen taxcons_a0=0
quietly {
forv i = 1/`nagg' {
      replace tau_t0 = VAT0[1,`i']/(EXP0[1,`i']-TAX0[1,`i']) in `i'
      replace tau_a0 = EXC0[1,`i']/(EXP0[1,`i']-TAX0[1,`i']) in `i'
      replace tau0 = tau_t0+tau_a0 in `i'
      replace q0 = 1 + tau0 in `i'
      replace taxcons_t0 = tau_t0/(1+tau0) in `i'
      replace taxcons_a0 = tau_a0/(1+tau0) in `i'
      replace taxcons0 = tau0/(1+tau0) in `i'
}
}
save "`outputtax'", replace
**********aggrtax_slim

mkmat taxcons_t0, matrix(VAT)
mkmat taxcons_a0, matrix(EXC)
mkmat taxcons0, matrix(TAX)

use "`match'", clear
forv i = 1/`nagg' {
      local j = subscripts[`i',1]
      capture drop VAT`j'
      capture drop EXC`j'
      capture drop TAX`j'
      gen VAT`j' = exp`j'*VAT[`i',1]
      gen EXC`j' = exp`j'*EXC[`i',1]
      gen TAX`j' = exp`j'*TAX[`i',1]
}
save "`output'", replace
```

APPENDIX 3: STATA-PROGRAM SIMULATION1.DO

```
*simulates change in standard VAT rates to compensate for change in
direct taxation (government budget neutrality)
*budget shares are kept constant

************************************************
* Input part                                   *
************************************************

*define country
local country = "UK"

*country specific information
if "`country'" == "UK" {
      local standardrate = 0.175
      *year of budget survey and taxcode
      local year = 2003
      *number of aggregates
      local nagg = 16
      *number of durable aggregates
      local ndur = 1
      *number of homeproduction aggregates
      local nhome = 0
      *independent variables (without total nondurable expenditures),
order must be the same as in parameters_`country'.dta
      local varlist = "smoker renter pubtr edu male north mid east west
scotland nireland selfemp emp unemp pens age age2 np na nch"
      *define consecutive rise in standard VAT rate per loop
      local step = 0.005
      *define initial rise in standard VAT rate
      local t = 0.035
}
if "`country'" == "BE" {
      local standardrate = 0.21
      *year of budget survey and taxcode
      local year = 2003
      *number of aggregates
      local nagg = 16
      *number of durable aggregates
      local ndur = 1
      *number of homeproduction aggregates
      local nhome = 0
      *independent variables (without total nondurable expenditures),
order must be the same as in parameters_`country'.dta
      local varlist = "smoker renter pubtr edu male wa br selfemp emp
unemp pens higheduc secondary age age2 np na nch"
      *define consecutive rise in standard VAT rate per loop
      local step = 0.01
      *define initial rise in standard VAT rate
      local t = 0.03
}
if "`country'" == "HU" {
      local standardrate = 0.25
      *year of budget survey and taxcode
```

```
      local year = 2005
      *number of aggregates
      local nagg = 17
      *number of durable aggregates
      local ndur = 1
      *number of homeproduction aggregates
      local nhome = 1
      *independent variables (without total nondurable expenditures),
order must be the same as in parameters_`country'.dta
      local varlist = "smoker renter pubtr edu male selfemp emp unemp
pens higheduc secondary age age2 np na nch"
      *define consecutive rise in standard VAT rate per loop
      local step = 0.01
      *define initial rise in standard VAT rate
      local t = 0.06
}


*paths to input and output datasets
local data = "C:\Documents and
Settings\n07072\Desktop\wp3.6\\`country'\input\outputparstata.dta"
local taxcode = "C:\Documents and
Settings\n07072\Desktop\wp3.6\\`country'\input\taxcode_`country'_`year'
.dta"
*variable containing coicop aggregates in taxcode
local varcat = "coicop1"
local exp = "C:\Documents and
Settings\n07072\Desktop\wp3.6\\`country'\input\HBSdetail_`country'_`yea
r'.dta"
local param = "C:\Documents and
Settings\n07072\Desktop\wp3.6\\`country'\input\parameters_`country'_`ye
ar'.dta"
local output = "C:\Documents and
Settings\n07072\Desktop\wp3.6\\`country'\output\simulatieparstata1.dta"
local outputtax = "C:\Documents and
Settings\n07072\Desktop\wp3.6\\`country'\output\aggregates prices and
taxes1.dta"
local expandtax = "C:\Documents and
Settings\n07072\Desktop\wp3.6\\`country'\output\exp and taxes1.dta"


*set max matrix size
set matsize 2000

************************************************
* Generic part                                 *
************************************************

*define local parameters concerning number of aggregates
local hp = "n"
if `nhome' > 0 {
      local hp = "y"
}
local nndagg = `nagg' - `ndur'
local nh = `nndagg' - `nhome'
```

```
*construct variable lists for nondurables-nonhomeproduction, for
durables and for home production variables
*construct vector of exp-subscripts
matrix subscripts = J(`nagg',1,0)
use "`data'", clear
local varlist1 = ""
local varlist2 = ""
local varlist3 = ""
local count = 1
foreach var of varlist exp* {
      if (substr("`var'",4,2) == "98") {
            local varlist2 = "`varlist2' `var'"
            local i = substr("`var'",4,.)
            mat subscripts[`count',1] = `i'
            local count = `count' + 1
      }
      else {
            if (substr("`var'",4,2) == "99") {
                  local varlist3 = "`varlist3' `var'"
                  local i = substr("`var'",4,.)
                  mat subscripts[`count',1] = `i'
                  local count = `count' + 1
            }
            else {
                  local varlist1 = "`varlist1' `var'"
                  local i = substr("`var'",4,.)
                  mat subscripts[`count',1] = `i'
                  local count = `count' + 1
            }
      }
}

*calculate change in government revenue and create output file
capture drop Dy
gen Dy = (std_dispy_1 - std_dispy)*coweight
qui su Dy
local average = r(mean)
local n = _N
local DG = `n'*`average'
save "`output'", replace

*calculate new total expenditures: savings constant
use "`output'", clear
capture drop totexpnondur
egen totexpnondur = rowtotal(`varlist1' `varlist3')
capture drop totexp
egen totexp = rowtotal(`varlist1' `varlist2' `varlist3')
capture drop savings
gen savings = std_dispy - totexp
capture drop totexp_1
gen totexp_1 = std_dispy_1 - savings
save "`output'", replace

**********aggrtax_slim
*retrieving rates and prices on desaggregate level
use "`taxcode'", clear
sort itemid
```

```
local n_items = _N
capture drop t1
gen t1 = t
replace t1 = t1 + `t' if round(1000*t1) == 1000*`standardrate'
mkmat `varcat', matrix(coicop)
mkmat t, matrix(t)
mkmat t1, matrix(t1)
mkmat v, matrix(v)
mkmat q, matrix(q)
mkmat a, matrix(a)

**calculate producer price
capture drop p
gen p = ((1-(1+t)*v)/(1+t))*q-a
mkmat p, matrix(p)

**calculate new consumer price
capture drop q1
gen q1 =((1+t1)*(p+a))/(1-(1+t1)*v)
mkmat q1, matrix(q1)

*calculate old and new aggregate prices and tax rates

** initialisatie variabelen
use "`exp'", clear
forv i = 1/`nagg' {
      local j = subscripts[`i',1]
      capture drop exp_`j'
      capture drop VAT_`j'
      capture drop EXC_`j'
      gen exp_`j' = 0
      gen VAT_`j' = 0
      gen EXC_`j' = 0
      capture drop exp1_`j'
      capture drop VAT1_`j'
      capture drop EXC1_`j'
      gen exp1_`j' = 0
      gen VAT1_`j' = 0
      gen EXC1_`j' = 0

}
save "`expandtax'", replace


** calculate total expenditures and taxes
quietly {
forv i = 1/`n_items' {
      local k = coicop[`i',1]
      capture drop e1
      gen e1=e_`i'*(q1[`i',1]/q[`i',1])
      local alpha = a[`i',1]/p[`i',1]
      local denom0 = 1-(1+t[`i',1])*v[`i',1]
      local denom1 = 1-(1+t1[`i',1])*v[`i',1]
      local tau_t0 = (t[`i',1]*(1+`alpha'+v[`i',1])+v[`i',1])/`denom0'
      local tau_t1= (t1[`i',1]*(1+`alpha'+v[`i',1])+v[`i',1])/`denom1'
      local tau_a0 = `alpha'/`denom0'
      local tau_a1 = `alpha'/`denom1'
```

```
      local tau0 = `tau_t0'+`tau_a0'
      local tau1 = `tau_t1'+`tau_a1'
      replace exp_`k'=exp_`k'+e_`i'
      replace exp1_`k'=exp1_`k'+e1
      replace VAT_`k'=VAT_`k'+(`tau_t0'/(1+`tau0'))*e_`i'
      replace VAT1_`k'=VAT1_`k'+(`tau_t1'/(1+`tau1'))*e1
      replace EXC_`k'=EXC_`k'+(`tau_a0'/(1+`tau0'))*e_`i'
      replace EXC1_`k'=EXC1_`k'+(`tau_a1'/(1+`tau1'))*e1
}
}
keep hhid weight exp* VAT* EXC*
save "`expandtax'", replace

collapse (sum) exp* VAT* EXC* [pw=weight]
mkmat exp_*, matrix(EXP0)
mkmat exp1_*, matrix(EXP1)
mkmat VAT_*, matrix(VAT0)
mkmat VAT1_*, matrix(VAT1)
mkmat EXC_*, matrix(EXC0)
mkmat EXC1_*, matrix(EXC1)
mat TAX0 = VAT0+EXC0
mat TAX1 = VAT1+EXC1
drop exp* VAT* EXC*

tempfile taxes
svmat subscripts
rename subscripts1 coicop
gen tau0=0
gen tau_t0=0
gen tau_a0=0
gen q0=0
gen tau1=0
gen tau_t1=0
gen tau_a1=0
gen q1=0
gen tax0 = 0
gen tax1 = 0
quietly {
forv i = 1/`nagg' {
      replace tau_t0 = VAT0[1,`i']/(EXP0[1,`i']-TAX0[1,`i']) in `i'
      replace tau_a0 = EXC0[1,`i']/(EXP0[1,`i']-TAX0[1,`i']) in `i'
      replace tau0 = tau_t0+tau_a0 in `i'
      replace q0 = 1 + tau0 in `i'
      replace tax0 = tau0/(1+tau0) in `i'

      replace tau_t1 = VAT1[1,`i']/(EXP1[1,`i']-TAX1[1,`i']) in `i'
      replace tau_a1 = EXC1[1,`i']/(EXP1[1,`i']-TAX1[1,`i']) in `i'
      replace tau1 = tau_t1+tau_a1 in `i'
      replace q1 = 1 + tau1 in `i'
      replace tax1 = tau1/(1+tau1) in `i'
}
}
save "`outputtax'", replace
**********aggrtax_slim

*calculate total indirect taxes per household and over population,
baseline
```

```
mkmat q0, matrix(prices0)
mkmat tau0, matrix(tau0)
mkmat tax0, matrix(taxes0)
mkmat q1, matrix(prices1)
mkmat tau1, matrix(tau1)
mkmat tax1, matrix(taxes1)

*calculate new durable expenditures
use "`output'", clear
foreach var of varlist `varlist2' {
      local i = substr("`var'",4,.)
      local count = 0
      forv j = 1/`nagg' {
            if subscripts[`j',1] == `i' {
                  local count = `j'
            }
      }
      capture drop uitg_`i'
      gen uitg_`i' =(`var'/(1+tau0[`count',1]))*(1+tau1[`count',1])
}
capture drop durables_1
egen durables_1 = rowtotal(uitg_98*)
*calculate new total nondurable expenditures
capture drop totexpnondur_1
gen totexpnondur_1 = totexp_1 - durables_1
drop if totexpnondur_1 <= 0

*calculate original indirect tax revenue
capture drop indtax0
gen indtax0 = 0
foreach var of varlist `varlist1' `varlist2' `varlist3' {
      local i = substr("`var'",4,.)
      local count = 0
      forv j = 1/`nagg' {
            if subscripts[`j',1] == `i' {
                  local count = `j'
            }
      }
      replace indtax0 = indtax0 + taxes0[`count',1]*`var'
}
capture drop weightedindtax0
gen weightedindtax0 = indtax0*coweight
qui su weightedindtax0
local tottax_0 = r(mean)*_N

*construct welfare matrix containing price effect for nondurable goods
matrix welfare = J(`nagg',1,0)
forv i = 1/`nagg' {
      matrix welfare[`i',1] = (prices1[`i',1]-
prices0[`i',1])/prices1[`i',1]
}
foreach var of varlist `varlist2' {
      local i = substr("`var'",4,.)
      forv j = 1/`nagg' {
            if subscripts[`j',1] == `i' {
                  mat welfare[`j',1] = 0
            }
```

```
	}
}

*calculate new nondurable expenditures
foreach var of varlist `varlist1' `varlist3' {
	local i = substr("`var'",4,.)
	capture drop uitg_`i'
	gen uitg_`i' = w_`i'*totexpnondur_1
}

*calculate welfare changes
capture drop pe
gen pe = 0
foreach var of varlist uitg_* {
	local i = substr("`var'",6,.)
	forv j = 1/`nagg' {
		if `i' == subscripts[`j',1] {
			local pos = `j'
		}
	}
	replace pe = pe + welfare[`pos',1]*`var'

}
replace pe = - pe
capture drop verschil_exp
gen verschil_exp = totexp_1-totexp
capture drop WG
gen WG = verschil_exp+pe

*calculate new indirect taxes
capture drop indtax1
gen indtax1 = 0
foreach var of varlist `varlist1' `varlist2' `varlist3' {
	local i = substr("`var'",4,.)
	local count = 0
	forv j = 1/`nagg' {
		if subscripts[`j',1] == `i' {
			local count = `j'
		}
	}
	replace indtax1 = indtax1 + taxes1[`count',1]*uitg_`i'
}
capture drop weightedindtax1
gen weightedindtax1 = indtax1*coweight
qui su weightedindtax1
local tottax_1 = r(mean)*_N
save "`output'", replace

*loop
local verschil = `tottax_1' - `tottax_0'
local teller = 0
while(`verschil'<`DG') {

	local t = `t' + `step'
	local teller = `teller' + 1

**********aggrtax_slim
```

```
*retrieving rates and prices on desaggregate level
use "`taxcode'", clear
sort itemid
local n_items = _N
capture drop t1
gen t1 = t
replace t1 = t1 + `t' if round(1000*t1) == 1000*`standardrate'
mkmat `varcat', matrix(coicop)
mkmat t, matrix(t)
mkmat t1, matrix(t1)
mkmat v, matrix(v)
mkmat q, matrix(q)
mkmat a, matrix(a)

**calculate producer price
capture drop p
gen p = ((1-(1+t)*v)/(1+t))*q-a
mkmat p, matrix(p)

**calculate new consumer price
capture drop q1
gen q1 =((1+t1)*(p+a))/(1-(1+t1)*v)
mkmat q1, matrix(q1)

*calculate old and new aggregate prices and tax rates

** initialisatie variabelen
use "`exp'", clear
forv i = 1/`nagg' {
      local j = subscripts[`i',1]
      capture drop exp_`j'
      capture drop VAT_`j'
      capture drop EXC_`j'
      gen exp_`j' = 0
      gen VAT_`j' = 0
      gen EXC_`j' = 0
      capture drop exp1_`j'
      capture drop VAT1_`j'
      capture drop EXC1_`j'
      gen exp1_`j' = 0
      gen VAT1_`j' = 0
      gen EXC1_`j' = 0

}
save "`expandtax'", replace


** calculate total expenditures and taxes
quietly {
forv i = 1/`n_items' {
      local k = coicop[`i',1]
      capture drop e1
      gen e1=e_`i'*(q1[`i',1]/q[`i',1])
      local alpha = a[`i',1]/p[`i',1]
      local denom0 = 1-(1+t[`i',1])*v[`i',1]
      local denom1 = 1-(1+t1[`i',1])*v[`i',1]
      local tau_t0 = (t[`i',1]*(1+`alpha'+v[`i',1])+v[`i',1])/`denom0'
```

```
      local tau_t1= (t1[`i',1]*(1+`alpha'+v[`i',1])+v[`i',1])/`denom1'
      local tau_a0 = `alpha'/`denom0'
      local tau_a1 = `alpha'/`denom1'
      local tau0 = `tau_t0'+`tau_a0'
      local tau1 = `tau_t1'+`tau_a1'
      replace exp_`k'=exp_`k'+e_`i'
      replace exp1_`k'=exp1_`k'+e1
      replace VAT_`k'=VAT_`k'+(`tau_t0'/(1+`tau0'))*e_`i'
      replace VAT1_`k'=VAT1_`k'+(`tau_t1'/(1+`tau1'))*e1
      replace EXC_`k'=EXC_`k'+(`tau_a0'/(1+`tau0'))*e_`i'
      replace EXC1_`k'=EXC1_`k'+(`tau_a1'/(1+`tau1'))*e1
}
}
keep hhid weight exp* VAT* EXC*
save "`expandtax'", replace

collapse (sum) exp* VAT* EXC* [pw=weight]
mkmat exp_*, matrix(EXP0)
mkmat exp1_*, matrix(EXP1)
mkmat VAT_*, matrix(VAT0)
mkmat VAT1_*, matrix(VAT1)
mkmat EXC_*, matrix(EXC0)
mkmat EXC1_*, matrix(EXC1)
mat TAX0 = VAT0+EXC0
mat TAX1 = VAT1+EXC1
drop exp* VAT* EXC*

tempfile taxes
svmat subscripts
rename subscripts1 coicop
gen tau0=0
gen tau_t0=0
gen tau_a0=0
gen q0=0
gen tau1=0
gen tau_t1=0
gen tau_a1=0
gen q1=0
gen tax0 = 0
gen tax1 = 0
quietly {
forv i = 1/`nagg' {
      replace tau_t0 = VAT0[1,`i']/(EXP0[1,`i']-TAX0[1,`i']) in `i'
      replace tau_a0 = EXC0[1,`i']/(EXP0[1,`i']-TAX0[1,`i']) in `i'
      replace tau0 = tau_t0+tau_a0 in `i'
      replace q0 = 1 + tau0 in `i'
      replace tax0 = tau0/(1+tau0) in `i'

      replace tau_t1 = VAT1[1,`i']/(EXP1[1,`i']-TAX1[1,`i']) in `i'
      replace tau_a1 = EXC1[1,`i']/(EXP1[1,`i']-TAX1[1,`i']) in `i'
      replace tau1 = tau_t1+tau_a1 in `i'
      replace q1 = 1 + tau1 in `i'
      replace tax1 = tau1/(1+tau1) in `i'
}
}
save "`outputtax'", replace
**********aggrtax_slim
```

```
*calculate total indirect taxes per household and over population,
baseline
mkmat q0, matrix(prices0)
mkmat tau0, matrix(tau0)
mkmat tax0, matrix(taxes0)
mkmat q1, matrix(prices1)
mkmat tau1, matrix(tau1)
mkmat tax1, matrix(taxes1)

*calculate new durable expenditures
use "`output'", clear
foreach var of varlist `varlist2' {
      local i = substr("`var'",4,.)
      local count = 0
      forv j = 1/`nagg' {
            if subscripts[`j',1] == `i' {
                  local count = `j'
            }
      }
      capture drop uitg_`i'
      gen uitg_`i' =(`var'/(1+tau0[`count',1]))*(1+tau1[`count',1])
}
capture drop durables_1
egen durables_1 = rowtotal(uitg_98*)
*calculate new total nondurable expenditures
capture drop totexpnondur_1
gen totexpnondur_1 = totexp_1 - durables_1
drop if totexpnondur_1 <= 0

*calculate original indirect tax revenue
capture drop indtax0
gen indtax0 = 0
foreach var of varlist `varlist1' `varlist2' `varlist3' {
      local i = substr("`var'",4,.)
      local count = 0
      forv j = 1/`nagg' {
            if subscripts[`j',1] == `i' {
                  local count = `j'
            }
      }
      replace indtax0 = indtax0 + taxes0[`count',1]*`var'
}
capture drop weightedindtax0
gen weightedindtax0 = indtax0*coweight
qui su weightedindtax0
local tottax_0 = r(mean)*_N

*construct welfare matrix containing price effect for nondurable goods
matrix welfare = J(`nagg',1,0)
forv i = 1/`nagg' {
      matrix welfare[`i',1] = (prices1[`i',1]-
prices0[`i',1])/prices1[`i',1]
}
foreach var of varlist `varlist2' {
      local i = substr("`var'",4,.)
      forv j = 1/`nagg' {
```

```
            if subscripts[`j',1] == `i' {
                  mat welfare[`j',1] = 0
            }
      }
}

*calculate new nondurable expenditures
foreach var of varlist `varlist1' `varlist3' {
      local i = substr("`var'",4,.)
      capture drop uitg_`i'
      gen uitg_`i' = w_`i'*totexpnondur_1
}

*calculate welfare changes
capture drop pe
gen pe = 0
foreach var of varlist uitg_* {
      local i = substr("`var'",6,.)
      forv j = 1/`nagg' {
            if `i' == subscripts[`j',1] {
                  local pos = `j'
            }
      }
      replace pe = pe + welfare[`pos',1]*`var'
}
replace pe = - pe
capture drop verschil_exp
gen verschil_exp = totexp_1-totexp
capture drop WG
gen WG = verschil_exp+pe

*calculate new indirect taxes
capture drop indtax1
gen indtax1 = 0
foreach var of varlist `varlist1' `varlist2' `varlist3' {
      local i = substr("`var'",4,.)
      local count = 0
      forv j = 1/`nagg' {
            if subscripts[`j',1] == `i' {
                  local count = `j'
            }
      }
      replace indtax1 = indtax1 + taxes1[`count',1]*uitg_`i'
}
capture drop weightedindtax1
gen weightedindtax1 = indtax1*coweight
qui su weightedindtax1
local tottax_1 = r(mean)*_N
save "`output'", replace
local verschil = `tottax_1' - `tottax_0'
}

capture drop agegroup
gen agegroup = 0
replace agegroup = 1 if age>30 & age <=50
replace agegroup = 2 if age > 50
capture drop socstatb
```

```
gen socstatb = 0
replace socstatb =1 if emp ==1
replace socstatb =2 if unemp ==1
replace socstatb =3 if pens ==1
replace socstatb =4 if other ==1
capture drop eqinco
gen eqinco = inc/sqrt(np)
xtile dec = eqinco, nq(10)
capture drop t
gen t = `t'
save "`output'", replace


di `verschil'
di `DG'
local maladj = `verschil'-`DG'
di `maladj'
di `t'
```

APPENDIX 4: STATA-PROGRAM SIMULATION2.DO

```
*simulates change in standard VAT rates to compensate for change in
direct taxation (government budget neutrality)
*new budget shares determined by Engel curves

************************************************
* Input part                                   *
************************************************

*define country
local country = "UK"

*country specific information
if "`country'" == "UK" {
      local standardrate = 0.175
      *year of budget survey and taxcode
      local year = 2003
      *number of aggregates
      local nagg = 16
      *number of durable aggregates
      local ndur = 1
      *number of homeproduction aggregates
      local nhome = 0
      *independent variables (without total nondurable expenditures),
order must be the same as in parameters_`country'.dta
      local varlist = "smoker renter pubtr edu male north mid east west
scotland nireland selfemp emp unemp pens age age2 np na nch"
      *define consecutive rise in standard VAT rate per loop
      local step = 0.005
      *define initial rise in standard VAT rate
      local t = 0.035
}
if "`country'" == "BE" {
      local standardrate = 0.21
      *year of budget survey and taxcode
      local year = 2003
      *number of aggregates
      local nagg = 16
      *number of durable aggregates
      local ndur = 1
      *number of homeproduction aggregates
      local nhome = 0
      *independent variables (without total nondurable expenditures),
order must be the same as in parameters_`country'.dta
      local varlist = "smoker renter pubtr edu male wa br selfemp emp
unemp pens higheduc secondary age age2 np na nch"
      *define consecutive rise in standard VAT rate per loop
      local step = 0.01
      *define initial rise in standard VAT rate
      local t = 0.04
}
if "`country'" == "HU" {
      local standardrate = 0.25
```

```
      *year of budget survey and taxcode
      local year = 2005
      *number of aggregates
      local nagg = 17
      *number of durable aggregates
      local ndur = 1
      *number of homeproduction aggregates
      local nhome = 1
      *independent variables (without total nondurable expenditures),
order must be the same as in parameters_`country'.dta
      local varlist = "smoker renter pubtr edu male selfemp emp unemp
pens higheduc secondary age age2 np na nch"
      *define consecutive rise in standard VAT rate per loop
      local step = 0.01
      *define initial rise in standard VAT rate
      local t = 0.08
}


*paths to input and output datasets
local data = "C:\Documents and
Settings\n07072\Desktop\wp3.6\\`country'\input\outputparstata.dta"
local taxcode = "C:\Documents and
Settings\n07072\Desktop\wp3.6\\`country'\input\taxcode_`country'_`year'
.dta"
*variable containing coicop aggregates in taxcode
local varcat = "coicop1"
local exp = "C:\Documents and
Settings\n07072\Desktop\wp3.6\\`country'\input\HBSdetail_`country'_`yea
r'.dta"
local param = "C:\Documents and
Settings\n07072\Desktop\wp3.6\\`country'\input\parameters_`country'_`ye
ar'.dta"
local output = "C:\Documents and
Settings\n07072\Desktop\wp3.6\\`country'\output\simulatieparstata2.dta"
local outputtax = "C:\Documents and
Settings\n07072\Desktop\wp3.6\\`country'\output\aggregates prices and
taxes2.dta"
local expandtax = "C:\Documents and
Settings\n07072\Desktop\wp3.6\\`country'\output\exp and taxes2.dta"


*set max matrix size
set matsize 2000

************************************************
* Generic part                                 *
************************************************

*define local parameters concerning number of aggregates
local hp = "n"
if `nhome' > 0 {
      local hp = "y"
}
local nndagg = `nagg' - `ndur'
local nh = `nndagg' - `nhome'
```

```
*construct variable lists for nondurables-nonhomeproduction, for
durables and for home production variables
*construct vector of exp-subscripts
matrix subscripts = J(`nagg',1,0)
use "`data'", clear
local varlist1 = ""
local varlist2 = ""
local varlist3 = ""
local count = 1
foreach var of varlist exp* {
      if (substr("`var'",4,2) == "98") {
            local varlist2 = "`varlist2' `var'"
            local i = substr("`var'",4,.)
            mat subscripts[`count',1] = `i'
            local count = `count' + 1
      }
      else {
            if (substr("`var'",4,2) == "99") {
                  local varlist3 = "`varlist3' `var'"
                  local i = substr("`var'",4,.)
                  mat subscripts[`count',1] = `i'
                  local count = `count' + 1
            }
            else {
                  local varlist1 = "`varlist1' `var'"
                  local i = substr("`var'",4,.)
                  mat subscripts[`count',1] = `i'
                  local count = `count' + 1
            }
      }
}

*calculate change in government revenue and create output file
capture drop Dy
gen Dy = (std_dispy_1 - std_dispy)*coweight
qui su Dy
local average = r(mean)
local n = _N
local DG = `n'*`average'
save "`output'", replace

*calculate new total expenditures: savings constant
use "`output'", clear
capture drop totexpnondur
egen totexpnondur = rowtotal(`varlist1' `varlist3')
capture drop totexp
egen totexp = rowtotal(`varlist1' `varlist2' `varlist3')
capture drop savings
gen savings = std_dispy - totexp
capture drop totexp_1
gen totexp_1 = std_dispy_1 - savings
save "`output'", replace

**********aggrtax_slim
*retrieving rates and prices on desaggregate level
use "`taxcode'", clear
sort itemid
```

```
local n_items = _N
capture drop t1
gen t1 = t
replace t1 = t1 + `t' if round(1000*t1) == 1000*`standardrate'
mkmat `varcat', matrix(coicop)
mkmat t, matrix(t)
mkmat t1, matrix(t1)
mkmat v, matrix(v)
mkmat q, matrix(q)
mkmat a, matrix(a)

**calculate producer price
capture drop p
gen p = ((1-(1+t)*v)/(1+t))*q-a
mkmat p, matrix(p)

**calculate new consumer price
capture drop q1
gen q1 =((1+t1)*(p+a))/(1-(1+t1)*v)
mkmat q1, matrix(q1)

*calculate old and new aggregate prices and tax rates

** initialisatie variabelen
use "`exp'", clear
forv i = 1/`nagg' {
      local j = subscripts[`i',1]
      capture drop exp_`j'
      capture drop VAT_`j'
      capture drop EXC_`j'
      gen exp_`j' = 0
      gen VAT_`j' = 0
      gen EXC_`j' = 0
      capture drop exp1_`j'
      capture drop VAT1_`j'
      capture drop EXC1_`j'
      gen exp1_`j' = 0
      gen VAT1_`j' = 0
      gen EXC1_`j' = 0

}
save "`expandtax'", replace


** calculate total expenditures and taxes
quietly {
forv i = 1/`n_items' {
      local k = coicop[`i',1]
      capture drop e1
      gen e1=e_`i'*(q1[`i',1]/q[`i',1])
      local alpha = a[`i',1]/p[`i',1]
      local denom0 = 1-(1+t[`i',1])*v[`i',1]
      local denom1 = 1-(1+t1[`i',1])*v[`i',1]
      local tau_t0 = (t[`i',1]*(1+`alpha'+v[`i',1])+v[`i',1])/`denom0'
      local tau_t1= (t1[`i',1]*(1+`alpha'+v[`i',1])+v[`i',1])/`denom1'
      local tau_a0 = `alpha'/`denom0'
      local tau_a1 = `alpha'/`denom1'
```

```
      local tau0 = `tau_t0'+`tau_a0'
      local tau1 = `tau_t1'+`tau_a1'
      replace exp_`k'=exp_`k'+e_`i'
      replace exp1_`k'=exp1_`k'+e1
      replace VAT_`k'=VAT_`k'+(`tau_t0'/(1+`tau0'))*e_`i'
      replace VAT1_`k'=VAT1_`k'+(`tau_t1'/(1+`tau1'))*e1
      replace EXC_`k'=EXC_`k'+(`tau_a0'/(1+`tau0'))*e_`i'
      replace EXC1_`k'=EXC1_`k'+(`tau_a1'/(1+`tau1'))*e1
}
}
keep hhid weight exp* VAT* EXC*
save "`expandtax'", replace

collapse (sum) exp* VAT* EXC* [pw=weight]
mkmat exp_*, matrix(EXP0)
mkmat exp1_*, matrix(EXP1)
mkmat VAT_*, matrix(VAT0)
mkmat VAT1_*, matrix(VAT1)
mkmat EXC_*, matrix(EXC0)
mkmat EXC1_*, matrix(EXC1)
mat TAX0 = VAT0+EXC0
mat TAX1 = VAT1+EXC1
drop exp* VAT* EXC*

tempfile taxes
svmat subscripts
rename subscripts1 coicop
gen tau0=0
gen tau_t0=0
gen tau_a0=0
gen q0=0
gen tau1=0
gen tau_t1=0
gen tau_a1=0
gen q1=0
gen tax0 = 0
gen tax1 = 0
quietly {
forv i = 1/`nagg' {
      replace tau_t0 = VAT0[1,`i']/(EXP0[1,`i']-TAX0[1,`i']) in `i'
      replace tau_a0 = EXC0[1,`i']/(EXP0[1,`i']-TAX0[1,`i']) in `i'
      replace tau0 = tau_t0+tau_a0 in `i'
      replace q0 = 1 + tau0 in `i'
      replace tax0 = tau0/(1+tau0) in `i'

      replace tau_t1 = VAT1[1,`i']/(EXP1[1,`i']-TAX1[1,`i']) in `i'
      replace tau_a1 = EXC1[1,`i']/(EXP1[1,`i']-TAX1[1,`i']) in `i'
      replace tau1 = tau_t1+tau_a1 in `i'
      replace q1 = 1 + tau1 in `i'
      replace tax1 = tau1/(1+tau1) in `i'
}
}
save "`outputtax'", replace
**********aggrtax_slim

*calculate total indirect taxes per household and over population,
baseline
```

```
mkmat q0, matrix(prices0)
mkmat tau0, matrix(tau0)
mkmat tax0, matrix(taxes0)
mkmat q1, matrix(prices1)
mkmat tau1, matrix(tau1)
mkmat tax1, matrix(taxes1)

*calculate new durable expenditures
use "`output'", clear
foreach var of varlist `varlist2' {
      local i = substr("`var'",4,.)
      local count = 0
      forv j = 1/`nagg' {
            if subscripts[`j',1] == `i' {
                  local count = `j'
            }
      }
      capture drop uitg_`i'
      gen uitg_`i' =(`var'/(1+tau0[`count',1]))*(1+tau1[`count',1])
}
capture drop durables_1
egen durables_1 = rowtotal(uitg_98*)
*calculate new total nondurable expenditures
capture drop totexpnondur_1
gen totexpnondur_1 = totexp_1 - durables_1
drop if totexpnondur_1 <= 0
save "`output'", replace

*engel
use "`param'", clear
local neq = 1+`ndur'
drop in 1/`neq'
mkmat param*, matrix(parameters)
*engel

use "`output'", clear
capture drop indtax0
gen indtax0 = 0
foreach var of varlist `varlist1' `varlist2' `varlist3' {
      local i = substr("`var'",4,.)
      local count = 0
      forv j = 1/`nagg' {
            if subscripts[`j',1] == `i' {
                  local count = `j'
            }
      }
      replace indtax0 = indtax0 + taxes0[`count',1]*`var'
}

capture drop weightedindtax0
gen weightedindtax0 = indtax0*coweight
qui su weightedindtax0
local tottax_0 = r(mean)*_N

*construct welfare matrix containing price effect for nondurable goods
matrix welfare = J(`nagg',1,0)
forv i = 1/`nagg' {
```

```
      matrix welfare[`i',1] = (prices1[`i',1]-
prices0[`i',1])/prices1[`i',1]
}
foreach var of varlist `varlist2' {
      local i = substr("`var'",4,.)
      forv j = 1/`nagg' {
            if subscripts[`j',1] == `i' {
                  mat welfare[`j',1] = 0
            }
      }
}

*calculate new nondurable budget shares with Engel curves
*engel
capture drop lntotexp_1
gen lntotexp_1 = log(totexpnondur_1*matcherror)
capture drop lntotexp_12
gen lntotexp_12=lntotexp_1^2
capture drop een
gen een = 1
capture drop group
gen group = smoker + 2*renter + 4*pubtr + 8*edu
capture drop bs_*
local count = 1
foreach var of varlist exp* {
      if (substr("`var'",1,5) != "exp98") {
            local i = substr("`var'",4,.)
            gen bs_`i' = 0
            forv j = 0/15 {
                  local position = `count' + `j'*`nndagg'
                  local count2 = 1
                  foreach var of varlist `varlist' lntotexp_1
lntotexp_12 een {
                        replace bs_`i' = bs_`i' +
parameters[`position',`count2']*`var' if group == `j'
                        local count2 = `count2' + 1
                  }
            }
            local count = `count' + 1
      }
}
foreach var of varlist bs_* {
      replace `var' = 0 if `var' < 0
}
capture drop total
egen total = rowtotal(bs_*)
foreach var of varlist bs_* {
      local i = substr("`var'",4,.)
      replace `var' = `var'/total
      capture drop uitg_`i'
      gen uitg_`i' = bs_`i'*totexpnondur_1
}
* engel

*calculate welfare changes
capture drop pe
gen pe = 0
```

```
foreach var of varlist uitg_* {
      local i = substr("`var'",6,.)
      forv j = 1/`nagg' {
            if `i' == subscripts[`j',1] {
                  local pos = `j'
            }
      }
      replace pe = pe + welfare[`pos',1]*`var'

}
replace pe = - pe
capture drop verschil_exp
gen verschil_exp = totexp_1-totexp
capture drop WG
gen WG = verschil_exp+pe

*calculate new indirect taxes
capture drop indtax1
gen indtax1 = 0
foreach var of varlist `varlist1' `varlist2' `varlist3' {
      local i = substr("`var'",4,.)
      local count = 0
      forv j = 1/`nagg' {
            if subscripts[`j',1] == `i' {
                  local count = `j'
            }
      }
      replace indtax1 = indtax1 + taxes1[`count',1]*uitg_`i'
}
capture drop weightedindtax1
gen weightedindtax1 = indtax1*coweight
qui su weightedindtax1
local tottax_1 = r(mean)*_N
save "`output'", replace

*loop
local verschil = `tottax_1' - `tottax_0'
local teller = 0
while(`verschil'<`DG') {

      local t = `t' + `step'
      local teller = `teller' + 1

**********aggrtax_slim
*retrieving rates and prices on desaggregate level
use "`taxcode'", clear
sort itemid
local n_items = _N
capture drop t1
gen t1 = t
replace t1 = t1 + `t' if round(1000*t1) == 1000*`standardrate'
mkmat `varcat', matrix(coicop)
mkmat t, matrix(t)
mkmat t1, matrix(t1)
mkmat v, matrix(v)
mkmat q, matrix(q)
mkmat a, matrix(a)
```

```
**calculate producer price
capture drop p
gen p = ((1-(1+t)*v)/(1+t))*q-a
mkmat p, matrix(p)

**calculate new consumer price
capture drop q1
gen q1 =((1+t1)*(p+a))/(1-(1+t1)*v)
mkmat q1, matrix(q1)

*calculate old and new aggregate prices and tax rates

** initialisatie variabelen
use "`exp'", clear
forv i = 1/`nagg' {
      local j = subscripts[`i',1]
      capture drop exp_`j'
      capture drop VAT_`j'
      capture drop EXC_`j'
      gen exp_`j' = 0
      gen VAT_`j' = 0
      gen EXC_`j' = 0
      capture drop exp1_`j'
      capture drop VAT1_`j'
      capture drop EXC1_`j'
      gen exp1_`j' = 0
      gen VAT1_`j' = 0
      gen EXC1_`j' = 0

}
save "`expandtax'", replace


** calculate total expenditures and taxes
quietly {
forv i = 1/`n_items' {
      local k = coicop[`i',1]
      capture drop e1
      gen e1=e_`i'*(q1[`i',1]/q[`i',1])
      local alpha = a[`i',1]/p[`i',1]
      local denom0 = 1-(1+t[`i',1])*v[`i',1]
      local denom1 = 1-(1+t1[`i',1])*v[`i',1]
      local tau_t0 = (t[`i',1]*(1+`alpha'+v[`i',1])+v[`i',1])/`denom0'
      local tau_t1= (t1[`i',1]*(1+`alpha'+v[`i',1])+v[`i',1])/`denom1'
      local tau_a0 = `alpha'/`denom0'
      local tau_a1 = `alpha'/`denom1'
      local tau0 = `tau_t0'+`tau_a0'
      local tau1 = `tau_t1'+`tau_a1'
      replace exp_`k'=exp_`k'+e_`i'
      replace exp1_`k'=exp1_`k'+e1
      replace VAT_`k'=VAT_`k'+(`tau_t0'/(1+`tau0'))*e_`i'
      replace VAT1_`k'=VAT1_`k'+(`tau_t1'/(1+`tau1'))*e1
      replace EXC_`k'=EXC_`k'+(`tau_a0'/(1+`tau0'))*e_`i'
      replace EXC1_`k'=EXC1_`k'+(`tau_a1'/(1+`tau1'))*e1
}
}
```

```
keep hhid weight exp* VAT* EXC*
save "`expandtax'", replace

collapse (sum) exp* VAT* EXC* [pw=weight]
mkmat exp_*, matrix(EXP0)
mkmat exp1_*, matrix(EXP1)
mkmat VAT_*, matrix(VAT0)
mkmat VAT1_*, matrix(VAT1)
mkmat EXC_*, matrix(EXC0)
mkmat EXC1_*, matrix(EXC1)
mat TAX0 = VAT0+EXC0
mat TAX1 = VAT1+EXC1
drop exp* VAT* EXC*

tempfile taxes
svmat subscripts
rename subscripts1 coicop
gen tau0=0
gen tau_t0=0
gen tau_a0=0
gen q0=0
gen tau1=0
gen tau_t1=0
gen tau_a1=0
gen q1=0
gen tax0 = 0
gen tax1 = 0
quietly {
forv i = 1/`nagg' {
      replace tau_t0 = VAT0[1,`i']/(EXP0[1,`i']-TAX0[1,`i']) in `i'
      replace tau_a0 = EXC0[1,`i']/(EXP0[1,`i']-TAX0[1,`i']) in `i'
      replace tau0 = tau_t0+tau_a0 in `i'
      replace q0 = 1 + tau0 in `i'
      replace tax0 = tau0/(1+tau0) in `i'

      replace tau_t1 = VAT1[1,`i']/(EXP1[1,`i']-TAX1[1,`i']) in `i'
      replace tau_a1 = EXC1[1,`i']/(EXP1[1,`i']-TAX1[1,`i']) in `i'
      replace tau1 = tau_t1+tau_a1 in `i'
      replace q1 = 1 + tau1 in `i'
      replace tax1 = tau1/(1+tau1) in `i'
}
}
save "`outputtax'", replace
**********aggrtax_slim

*calculate total indirect taxes per household and over population,
baseline
mkmat q0, matrix(prices0)
mkmat tau0, matrix(tau0)
mkmat tax0, matrix(taxes0)
mkmat q1, matrix(prices1)
mkmat tau1, matrix(tau1)
mkmat tax1, matrix(taxes1)

*calculate new durable expenditures
use "`output'", clear
foreach var of varlist `varlist2' {
```

```
      local i = substr("`var'",4,.)
      local count = 0
      forv j = 1/`nagg' {
            if subscripts[`j',1] == `i' {
                  local count = `j'
            }
      }
      capture drop uitg_`i'
      gen uitg_`i' =(`var'/(1+tau0[`count',1]))*(1+tau1[`count',1])
}
capture drop durables_1
egen durables_1 = rowtotal(uitg_98*)
*calculate new total nondurable expenditures
capture drop totexpnondur_1
gen totexpnondur_1 = totexp_1 - durables_1
drop if totexpnondur_1 <= 0
save "`output'", replace

*engel
use "`param'", clear
local neq = 1+`ndur'
drop in 1/`neq'
mkmat param*, matrix(parameters)
*engel

use "`output'", clear
capture drop indtax0
gen indtax0 = 0
foreach var of varlist `varlist1' `varlist2' `varlist3' {
      local i = substr("`var'",4,.)
      local count = 0
      forv j = 1/`nagg' {
            if subscripts[`j',1] == `i' {
                  local count = `j'
            }
      }
      replace indtax0 = indtax0 + taxes0[`count',1]*`var'
}

capture drop weightedindtax0
gen weightedindtax0 = indtax0*coweight
qui su weightedindtax0
local tottax_0 = r(mean)*_N

*construct welfare matrix containing price effect for nondurable goods
matrix welfare = J(`nagg',1,0)
forv i = 1/`nagg' {
      matrix welfare[`i',1] = (prices1[`i',1]-
prices0[`i',1])/prices1[`i',1]
}
foreach var of varlist `varlist2' {
      local i = substr("`var'",4,.)
      forv j = 1/`nagg' {
            if subscripts[`j',1] == `i' {
                  mat welfare[`j',1] = 0
            }
      }
```

```
}

*calculate new nondurable budget shares with Engel curves
*engel
capture drop lntotexp_1
gen lntotexp_1 = log(totexpnondur_1*matcherror)
capture drop lntotexp_12
gen lntotexp_12=lntotexp_1^2
capture drop een
gen een = 1
capture drop group
gen group = smoker + 2*renter + 4*pubtr + 8*edu
capture drop bs_*
local count = 1
foreach var of varlist exp* {
      if (substr("`var'",1,5) != "exp98") {
            local i = substr("`var'",4,.)
            gen bs_`i' = 0
            forv j = 0/15 {
                  local position = `count' + `j'*`nndagg'
                  local count2 = 1
                  foreach var of varlist `varlist' lntotexp_1
lntotexp_12 een {
                        replace bs_`i' = bs_`i' +
parameters[`position',`count2']*`var' if group == `j'
                        local count2 = `count2' + 1
                  }
            }
            local count = `count' + 1
      }
}
foreach var of varlist bs_* {
      replace `var' = 0 if `var' < 0
}
capture drop total
egen total = rowtotal(bs_*)
foreach var of varlist bs_* {
      local i = substr("`var'",4,.)
      replace `var' = `var'/total
      capture drop uitg_`i'
      gen uitg_`i' = bs_`i'*totexpnondur_1
}
* engel

*calculate welfare changes
capture drop pe
gen pe = 0
foreach var of varlist uitg_* {
      local i = substr("`var'",6,.)
      forv j = 1/`nagg' {
            if `i' == subscripts[`j',1] {
                  local pos = `j'
            }
      }
      replace pe = pe + welfare[`pos',1]*`var'
}
replace pe = - pe
```

```
capture drop verschil_exp
gen verschil_exp = totexp_1-totexp
capture drop WG
gen WG = verschil_exp+pe

*calculate new indirect taxes
capture drop indtax1
gen indtax1 = 0
foreach var of varlist `varlist1' `varlist2' `varlist3' {
      local i = substr("`var'",4,.)
      local count = 0
      forv j = 1/`nagg' {
            if subscripts[`j',1] == `i' {
                  local count = `j'
            }
      }
      replace indtax1 = indtax1 + taxes1[`count',1]*uitg_`i'
}
capture drop weightedindtax1
gen weightedindtax1 = indtax1*coweight
qui su weightedindtax1
local tottax_1 = r(mean)*_N
save "`output'", replace
local verschil = `tottax_1' - `tottax_0'
}

capture drop agegroup
gen agegroup = 0
replace agegroup = 1 if age>30 & age <=50
replace agegroup = 2 if age > 50
capture drop socstatb
gen socstatb = 0
replace socstatb =1 if emp ==1
replace socstatb =2 if unemp ==1
replace socstatb =3 if pens ==1
replace socstatb =4 if other ==1
capture drop eqinco
gen eqinco = inc/sqrt(np)
xtile dec = eqinco, nq(10)
capture drop t
gen t = `t'
save "`output'", replace


di `verschil'
di `DG'
local maladj = `verschil'-`DG'
di `maladj'
di `t'
```