THE NATIONAL INSTITUTE MODEL FOR LIFETIME INCOME DISTRIBUTIONAL ANALYSIS, LINDA

This manual provides practical guidance in the use of LINDA, a dynamic microsimulation model that projects a reference population cross-section through time, subject to endogenous savings and labour supply decisions.

User Manual

J. van de Ven
November, 2013

The National Institute Lifetime INcome Distributional Analysis model LINDA

User Manual

1. Table of Contents

1. Introduction

This manual describes use of the National Institute's *Lifetime INcome Distributional Analysis* model, *LINDA*, which is designed to explore the effects of changes to the tax and benefits structure on household circumstances through time. The model generates panel data for the entire life course of a reference population cross-section, and a series of summary statistics for each considered policy environment.

LINDA is complementary to other analytical approaches that are currently in use. Current large scale microsimulation models are able to provide detailed information regarding the immediate financial implications of policy change. However, such models are not well adapted to consider how savings, employment and consumption can be expected to adapt to altered financial incentives. Econometric analyses can go some way to filling in this missing detail, but not where uncertainty is likely to influence decision making. LINDA is specifically designed to explore savings and employment responses to policy change in context of important aspects of uncertainty that individuals face. The cost of this approach is that it is unable to reflect the degree of detail that is commonly taken into account by the two alternative analytical approaches that are referred to above. Thus, a thorough basis for balanced policy advice is best achieved at the present time by considering the same issue from alternative analytical perspectives.

LINDA is comprised of a series of Excel files that describe model parameters, and a central executable program that undertakes all of the requested analyses. Alteration of the model parameters is facilitated through an Excel "front-end", via a series of "user forms". One purpose of this manual is to describe how to use this Excel front-end. But before we move on to that, it is worth describing at a very high level of detail how the model works.

The model starts from cross-sectional data for the nuclear families of a sample of reference adults drawn from the Wealth and Assets Survey. The user is first directed to run a simulation that projects the evolving circumstances for the population cross-section forward and back through time, to build up a complete life-history for each reference adult. These lifetime data are saved by the model, and used as the "base" data from which subsequent policy-specific projections are made. It is possible to update the "base" data used by the model at any time, as is described later in this manual.

A simulation for a given policy environment typically involves 3 stages. 1) Specify model parameters through an Excel spread sheet. 2) Run the executable program, which automatically loads the model parameters, projects associated panel data staring from the prevailing simulation "base" (output in a standard format, csv), and calculates a set of associated summary statistics (output to Excel). 3) Analyse the model output.

Although users are unable to access the source code of the main executable program, they are able to alter in any way that they like the auxilliary files that implement taxes and benefits in the model. This manual also provides a brief description of how the tax and benefit code considered by the model can be altered.

The manual is divided into four sections. Section 2 describes how the model should be set-up for the first time on a computer, which includes use of the Excel front-end. Section 3 describes how to use the Excel front end, which guides a user through adjustment of selected model parameters. Section

4 describes the output generated by the model, and Section 5 provides some pointers for those interested in altering the tax and benefit programing code.

2. Set-up

System requirements

LINDA is designed to operate on desktop workstations that use Intel processors and the Microsoft "Windows" operating system. We recommend minimum system specifications of a 64 bit operating system, computing processor(s) with at least 6 physical cores, 8GB of RAM, and 500 GB of hard disk space.[1] Microsoft Excel is required to analyse summary statistics reported by the model. Furthermore, users who intend to alter the tax and benefits structure beyond simple parameter adjustments, or who wish to write their own analysis routines for integration with the model, will also require *Intel Fortran Studio XE* or *Intel Parallel Studio XE*.

Loading the model onto a new computer

The model is delivered as a single zip folder. The folder includes two subdirectories: FORTRAN, and MODEL. The FORTRAN subdirectory includes the programming code for the ANALYSIS and TAX routines that are provided with the model. The MODEL subdirectory includes all of the files that are required to run the model.

The MODEL subdirectory contains two subdirectories in addition to a set of model files. The subdirectory BASE_FILES contains a separate subdirectory for each "base" specification that you create with the model in which files that are required for the respective base specification are stored (as discussed in the section concerned with "FORM 0" that we return to below). The subdirectory SIMULATIONS will contain a separate subdirectory for each simulation that you run, in which are stored the panel data generated by the model, model parameters, and excel simulation output.

Please follow these steps when installing the model on a new computer:

1. Extract the zipped files from the compressed folder to a subdirectory of your choosing, maintaining the directory structure that we have included with the zipped file

[USERS WITH FORTRAN – ALL OTHERS SKIP TO STEP 19]

2. In the FORTRAN subdirectory, open up the TAXES subdirectory, and double-click on TAXES.sln
    a. This should open the Visual Studio program environment
3. If you can see the "solution explorer" window, then select the purple box
    a. If you cannot see the window, then open it through the "View" drop-down menu
4. In the "Project" drop-down menu at the top of Visual Studio, select "Properties"
5. In the "Configuration" drop-down menu select "All configurations"
6. In the "Platform" drop-down menu select "All platforms"
7. Under the "Configuration Properties", select the "General" category

[1] The model is delivered on the assumption that you have a 64 bit operating system. Please contact the NIESR if you require files to run on a 32 bit system.

8. Against the “Output Directory”, replace “C:\MyFiles\MODEL_LAB\MODEL\” with the file location that you have saved the model into
9. Under the “Configuration Properties”, select the “Debugging” category
10. Against “Command”, enter the location of the file “SIDD.exe”; eg: “C:\MyFiles\MODEL_LAB\MODEL\SIDD.EXE”
11. Against “Working Directory”, enter the same text as under (8)
12. Press the “Apply” button, and then the “Ok” button
13. Under the “File” drop-down menu select “Save All”
14. Under the “Build” drop-down menu select “Configuration Manager”
15. Under the “Active Solution Configuration” select “complete”
16. Under the “Active Solution Platform” select “x64” and press the “Close” button[2]
17. Under the “Build” drop-down menu select “Rebuild Solution”

You should then see some text like:

```
1>------ Rebuild All started: Project: TAXES, Configuration: complete x64 ------
1>Deleting intermediate files and output files for project 'TAXES', configuration
'Complete_taxes|x64'.
1>Compiling with Intel(R) Visual Fortran Compiler XE 13.0.1.119 [Intel(R) 64]...
1>2_global.F90
1>UK_1.F90
1>IE_1.F90
1>AUS_1.f90
1>UK_2.F90
1>3_NewTemplate.f90
1>1_TaxTools.f90
1>0_app_taxes.f90
1>Linking...
1>   Creating library C:\temp\MODEL\TAXES.lib and object C:\temp\MODEL\TAXES.exp
1>
1>Build log written to  "file://C:\temp\FORTRAN\TAXES\x64\Complete_taxes\BuildLog.htm"
1>TAXES - 0 error(s), 0 warning(s)
========== Rebuild All: 1 succeeded, 0 failed, 0 skipped ==========
```

18. Redo steps (2) to (17) for the “ANALYSIS” program solution

The above ensures that the programming files are all present and work

Creating a simulation base

As noted in the introduction, the model starts with data reported by the Wealth and Assets Survey for a population cross-section of reference adults. The model parameters have been calibrated to match the model to a wide range of summary statistics calculated from survey data sources, with the calibration structured around the year in which the reference population was observed (see van de Ven and Lucchino, 2013, for details). The model comes packaged, ready to project the circumstances of the population cross-section forward and backward through time, to build up a complete life history for each reference individual. It is recommended that this be done, and that the associated data should be defined as the “base” for subsequent simulations. This can be done by following the steps 19 to 29 below

19. Open the MODEL subdirectory, and then open “job file.xls”
20. Ensure that you allow macros to work in Excel
    a. please ask your system administrator if you require assistance with this

[2] If running a 32 bit environment then select “Win32” here.

21. Press ALT+F8
22. Select “SIDD” and press the RUN button
23. Press the “RUN EXISTING JOB FILE” button

This version of the model currently runs in around 9 hours – the associated simulation creates a new set of base data for analysis, using the full model specification.

24. Re-open “job file.xls”
25. Press ALT+F8
26. Select “SIDD” and press the RUN button
27. Enter “age18_all” into the text-box with the title “name of run to adopt as new base”
28. Enter “base_2006_age18_all” into the text-box with the title “directory name for new base”
29. Press the “CONVERT RUN TO NEW BASE” button

Excel will then work away for a short while, after which you should receive a message confirming that the new base has been created. If you look in the “base_files” subdirectory, you should now see a new subdirectory with the name “base_2006_age18_all”, which includes all of the files defining the base simulation specification.

To test that the model set-up has been successfully completed:

30. Re-open “job file.xls”
    a. This file has been changed since step (24), so that it now references the new base model directory by default.
31. Press ALT+F8
32. Select “SIDD” and press the RUN button
33. Press the “SET UP NEW SIMULATION” button
34. Type “test” in the text-box with the title “Simulation Name”
35. Press the “ENTER” button
36. Tick the box to direct the model to calculate “statistics for equivalised income deciles”
37. Tick the box to indicate that “comparative statistics with the population base” should be evaluated
38. Press the “ENTER AND RUN” button
39. Press the “LAUNCH MODEL” button

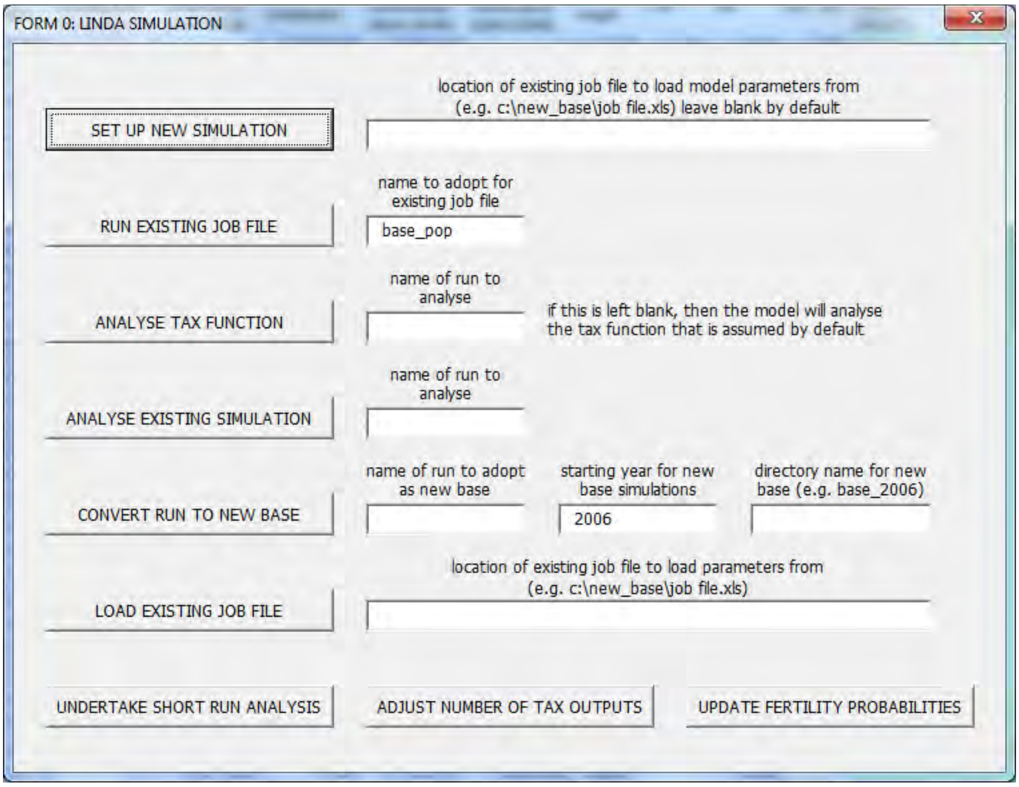
The model should then run through once again, in around 6-7 hours. This time, however, the simulations will project only forward through time, taking the population characteristics back in time from the base specification”. When the model is complete, please open the analysis_dec.xls file that is created in the “test” simulation directory, and check that all of the statistics reported in the “differences with base” sheet are close to zero.

3. Altering Parameters

Most of the key model parameters are stored in the spread sheet "job file.xls". Altering this file name may prevent the main executable file SIDD.exe from locating the model parameters.[3] "Job file.xls" is comprised of a number of worksheets. The parameters which drive the model are present in the worksheet *input*. Parameter values of the base simulation are stored in the sheet i*nputA.* Expert users can make changes directly to the parameters described in the *input* worksheet, without recourse to the user front-end. Alternatively the front end system of forms that is included with the spread sheet can be used to alter a selected set of model parameters. Differences between the data stored in the *input* and *inputA* sheets are identified by 1s in the *check* sheet. Any other sheets included in "job file.xls" are beyond the scope of this manual.

To use the front end, users should open "job file.xls" and run the macro *"SIDD"*, visible in the Tools/Macros window (Alt+F8). This displays Form 0.

FORM 0: LINDA SIMULATION



Form 0 offers the user a series of alternative options for running LINDA.

Setting up a New Simulation

[3] The only exception is in the case when the user would like to "stack" a series of simulations for consecutive (automated) execution. In this case, the "job file" associated with each alternative simulation should be numbered in their order or execution – e.g. "job file1.xls", "job file2.xls", "job file3.xls"…

This function allows the user to define and run a new simulation from scratch. Pressing the “SET UP NEW SIMULATION” button will lead to a series of alternative user forms, which are designed to guide the user through the process of selecting new model parameters. The related forms are described at length below.

Run Existing Job File

An existing job specification (as described by the parameters on the existing *input* sheet) can be run by clicking the “RUN EXISTING JOB FILE” button. The name of the simulation will be auto-populated with whatever name exists in cell A2 of the current *input* sheet. The user may alter this name to an alpha-numeric combination of their choosing. The model will then run, and save all associated results into a subdirectory with the name given to the simulation.

Analysing the Tax Function

It is often useful to “eyeball” the influence of simulated taxes and benefits on households. The model includes two analysis routines for this purpose, which can be accessed by pressing the “ANALYSE TAX FUNCTION” button. The associated analysis routines are described under “Form D4” later in this manual.

Analyse Existing Simulation

The model permits a series of supplementary analyses to be run after a given simulation is complete. Pressing the “ANALYSE EXISTING SIMULATION” button will open a new form that allows the user to choose which additional analyses are performed. This alternative is returned to under “Form 2” below.

Convert Run to New Base

As discussed in the introduction, the model projects a population through time assuming a series of “base” parameters. The base from which model projections are made can be re-specified to reflect any simulation that the user has previously run by listing the simulation name in the “NAME OF EXISING RUN TO ADOPT AS BASE” text box, and pressing the “CONVERT RUN TO NEW BASE” button. A name for the new base must also be entered, and all associated files will subsequently be stored under the given name in the subdirectory “base_files” of the main model directory. The model will subsequently use the specified simulation as its base for simulating a population forward through time.

Load Existing Job File

It will sometimes be useful to load in parameters from an existing job file for analysis. This can be achieved using the “LOAD EXISTING JOB FILE BUTTON”.

Undertake Short-run Analysis

The model will usually project the circumstances of a population forward through time from the base year, assuming a single policy environment. The “UNDERTAKE SHORT RUN ANALYSIS” button allows the population to be simulated forward assuming multiple policy environments. We return to describe this option further under “Form D1” below.

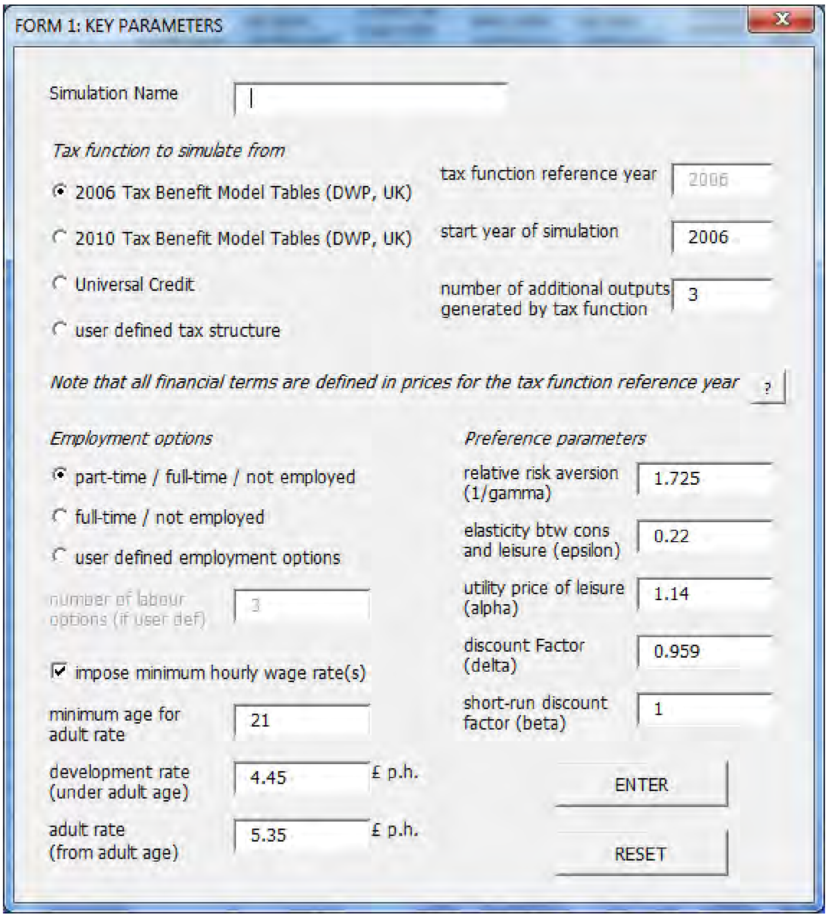
Adjust Number of Tax Outputs

The user can define a series of outputs relating to the tax and benefits structure that the model will save by default. We describe how to define additional transfer statistics in Section 5 of this manual. Where new outputs are desired, then the model parameters can be adjusted to accommodate these by pressing the "ADJUST NUMBER OF TAX OUTPUTS" button.

Update Fertility Probabilities

This is only necessary when the terms of fertility, as assumed in the model, are altered. If this is the case, then please contact the NIESR for further assistance.

FORM 1: KEY PARAMETERS



TIP: Explanatory notes can be found by pressing the "?" buttons.

Simulation Name.

The user must provide an alpha-numeric name for each simulated scenario. The results of the simulation are stored in a sub-directory with this name.

Preference Parameters

The preference relation assumed for the model is described in Appendix A of this manual. Key parameters governing the nature of the preference relation can be altered within this sheet.

Employment options

Here the user can choose between alternative specifications for the simulated labour supply decision of each adult: full-time/not employed; full-time/part-time/not employed; and an option that allows the user to define an arbitrary number of labour supply alternatives. When the third of these options is chosen, then the model assumes that the same hourly wage rate applies to all labour alternatives. If the full-time/part-time option is chosen, then hourly wage rates may vary between by the labour decision.

National Minimum Wage - Tick this box to apply the NWM, and set the adult level for the year from which the model projects a population cross-section through time. The model applies the NMW by assuming that any individual who's underlying productivity implies a lower hourly wage rate at a given employment option (e.g. full-time / part-time) than the National Minimum, cannot find work at that employment option.

Tax function to simulate from

LINDA is currently coded to simulate the population cross-section's lifetime under three alternative assumptions concerning the prevailing tax system. The model includes the policy environments described by the 2006 and 2010 DWP Tax and Benefit Tables, and the tax and benefit system that is expected to be in place following introduction of the Universal Credit. Users should use the radio buttons in Form 1 to select the tax system that they wish to apply. Specific parameters characterising these systems are specified in later forms. It is also possible to direct the model to use an alternative tax schedule that they have written themselves. Details about how to specify an entire tax schedule are discussed in Section 5.

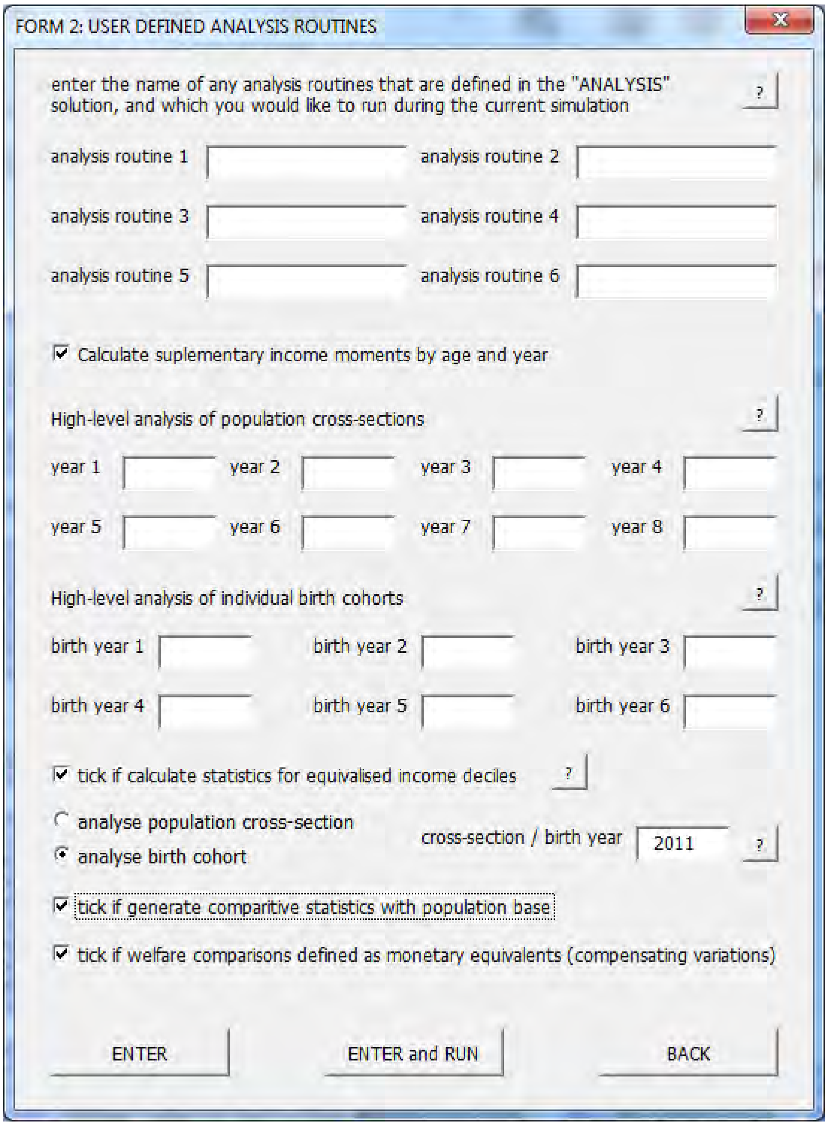
User-defined tax output

The model is set-up to permit the user to define specific tax and benefits statistics that the model will generate for each simulated individual. How these statistics are defined is discussed in Section 5. The number of statistics that the user has defined should be provided in the associated text box on this form.

Start year of simulation

The model will project forward, starting from data described for the year defined in this form.

FORM 2: ANALYSE EXISTING SIMULATION



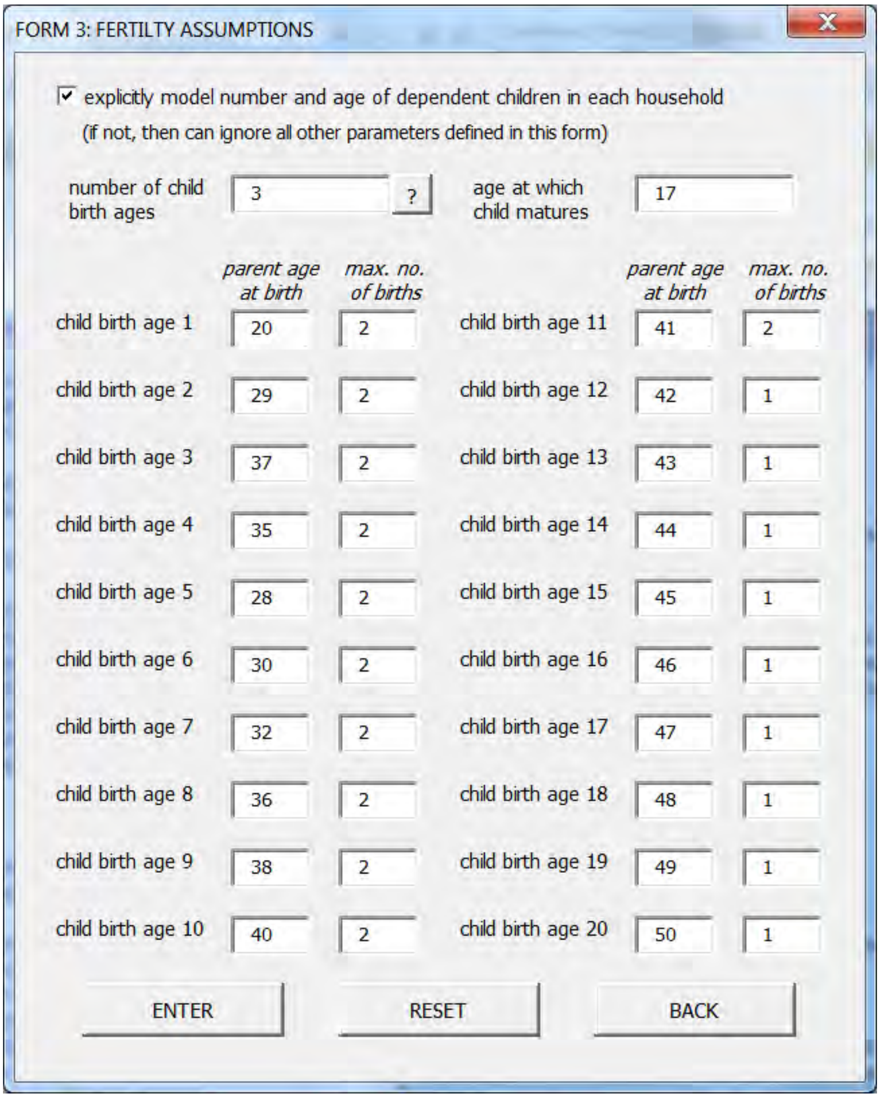
The user can request that the model run a series of alternative analysis routines, in addition to those that it runs for each simulation by default. The top panel of this form allows the user to request that the model run analysis routines that they have programmed themselves. How to program up this type of routine is discussed in Section 5. The name of the respective routine should be entered into one of the text boxes provided, bearing in mind that these names are case-specific.

The model can also be requested to run a series of pre-packaged routines. The first reports means and variances of income by age and year. The second set reports a selected summary statistics for given population cross-sections, and the third reports selected summary statistics for given birth cohorts. All of these statistics are reported in Excel output files (“income_moments.xls”, files ending

“XXXXcs.xls” for cross-sectional statistics, and files ending “XXXXby.xls” for birth cohorts, where “XXXX” refers to the relevant year).

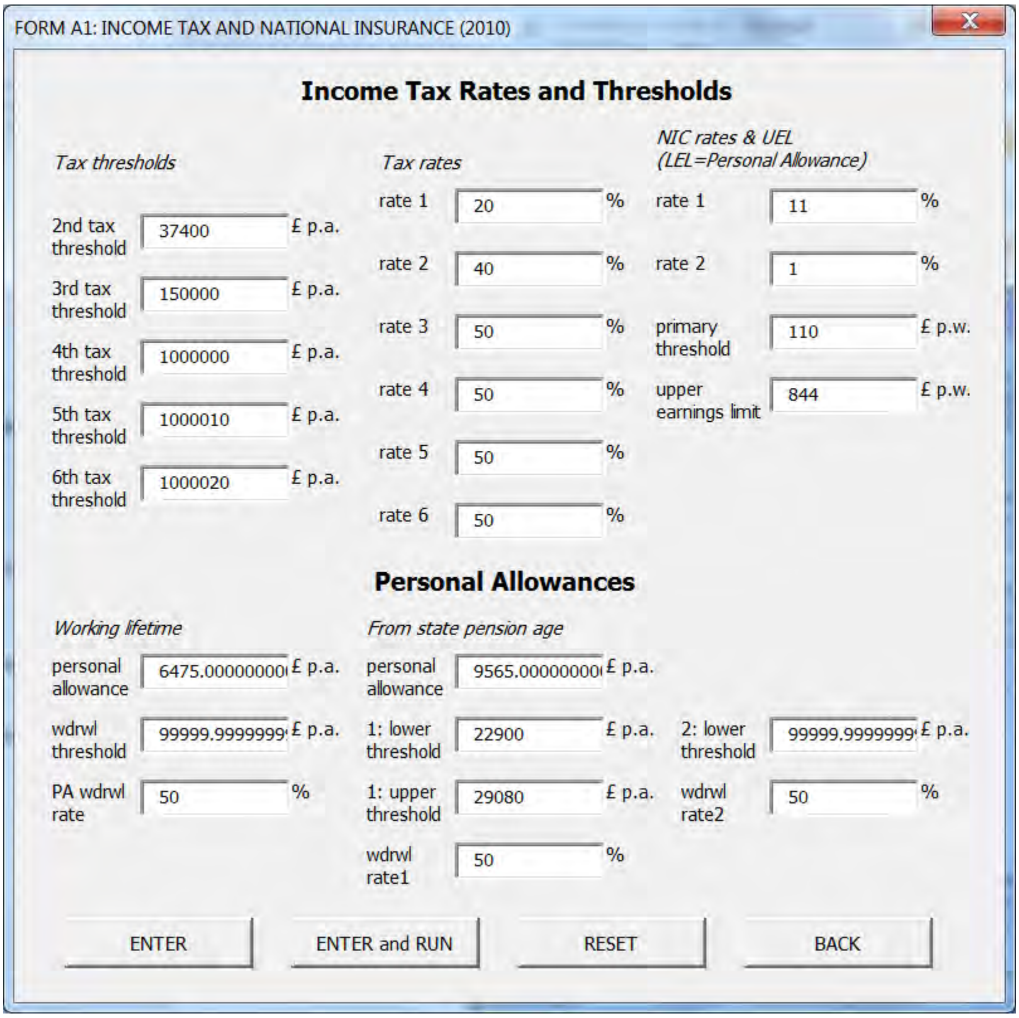
Finally, the model can be asked to produce a series of simulated averages for population deciles, specified by equivalised disposable family income. The revised OECD equivalence scale is used to adjust disposable income of families for size. If the model is directed to analyse data for a population cross-section, then the relevant year should be included in the form as directed. Otherwise, the birth year of the cohort of interest should be provided. The model can also be directed to report differences with the assumed base simulation. In this case, welfare comparisons between simulations can be specified either in the form of percentage changes or monetary equivalents. The second of these two is given the technical term “compensating variation”, and the associated box should be ticked in this user form if this is the format that welfare effects should be expressed in.

FORM 3: FERTILITY ASSUMPTIONS



To ensure that the model will solve within the desired timeframe, it is currently necessary to restrict child "births" to a small set of "child birth ages" (e.g. 3). To offset this stylisation, the model allows for multiple births at each child birth age. The parameters in this form allow the number and timing of child birth ages to be defined, the number of children that can be born at each age, and the number of years that children are considered to remain dependents. It is also possible to suppress explicit consideration of children through this user form, which will allow the model to complete a simulation in appreciably less time.

FORM A1: INCOME TAX AND NATIONAL INSTURANCE (2010)

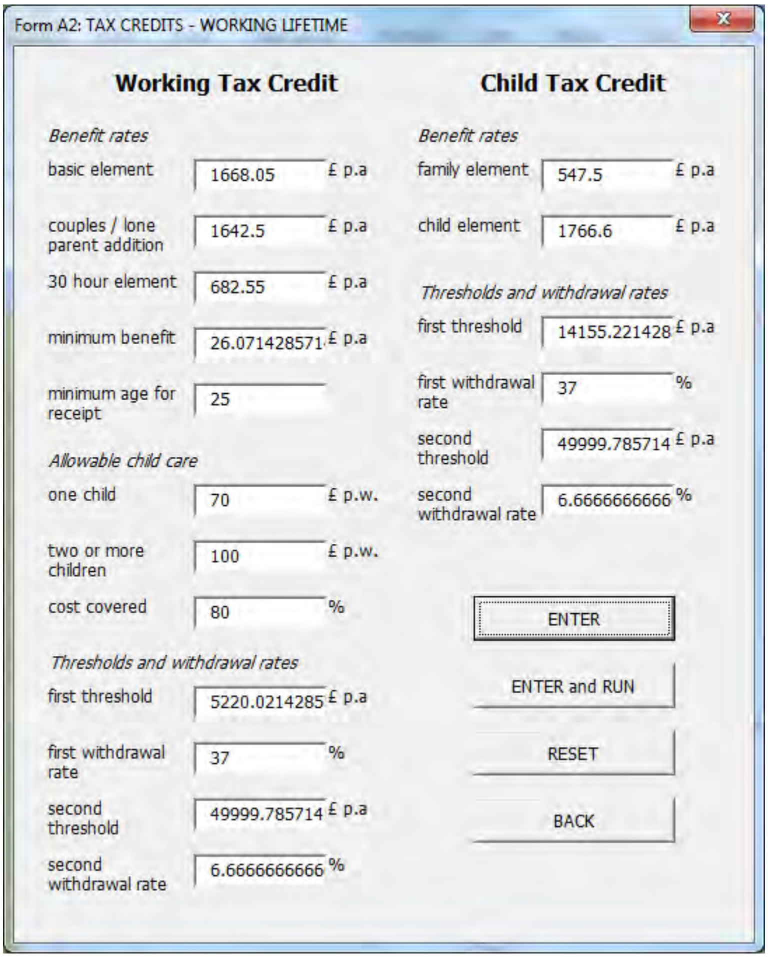


The form reported here is for the 2010 tax structure, and a slightly different form is presented for the 2006 tax structure. This form allows the user to specify the structure of income tax and national insurance. Consistent with contemporary tax policy in the UK, income taxes are calculated on individual specific "taxable income", obtained by subtracting an individual's Personal Allowance from their gross income. There are six possible tax rates and two national insurance rates. The first tax rate applies to taxable income up to the "2nd Tax Threshold", the second tax rate to the "3rd Tax Threshold", and so on. Similarly, "Rate 1" NICs are applied to taxable income between the "Primary Threshold" and the "Upper Earnings Limit", and "Rate 2" applied to taxable income in excess of the Upper Earnings Limit.

Individuals under state pension age and with a taxable income in excess of a "Wdrwl Threshold" have their Personal Allowance reduced at the rate "PA Wdrwl Rate". The adjustment of the Personal Allowance for people over state pension age is somewhat more complex. In this case, the Personal Allowance is withdrawn at "Wdrwl Rate1" on taxable income between "1: lower threshold", and "1:

upper threshold”, and is withdrawn at “Wdrwl Rate2” on taxable income in excess of “2: lower threshold”.

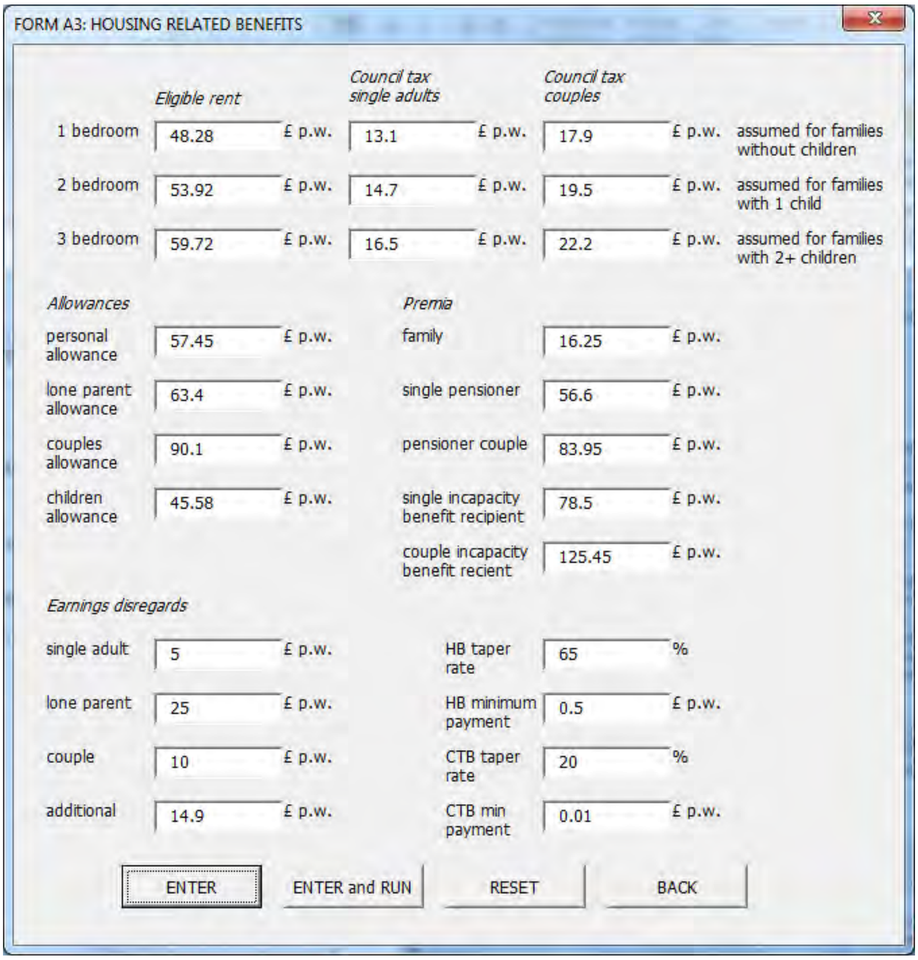
FORM A2: TAX CREDITS – WORKING LIFETIME



The parameters of the Working Tax Credit and the Child Tax Credit can be altered via Form A2. The 30 hour element of the WTC is considered to be awarded in respect of full-time employment of at least one adult household member. In the case of both the WTC and the CTC, the first withdrawal rate applies to gross income earned between the first and second thresholds, and the second withdrawal rate to gross income earned in excess of the second threshold until the respective benefit is exhausted.

If Universal Credit is selected for analysis in Form 1, then only those parameters that are relevant to the simplified tax structure are displayed.

FORM A3: HOUSING RELATED BENEFITS



FORM A3: HOUSING RELATED BENEFITS

	Eligible rent		Council tax single adults		Council tax couples		
1 bedroom	48.28	£ p.w.	13.1	£ p.w.	17.9	£ p.w.	assumed for families without children
2 bedroom	53.92	£ p.w.	14.7	£ p.w.	19.5	£ p.w.	assumed for families with 1 child
3 bedroom	59.72	£ p.w.	16.5	£ p.w.	22.2	£ p.w.	assumed for families with 2+ children

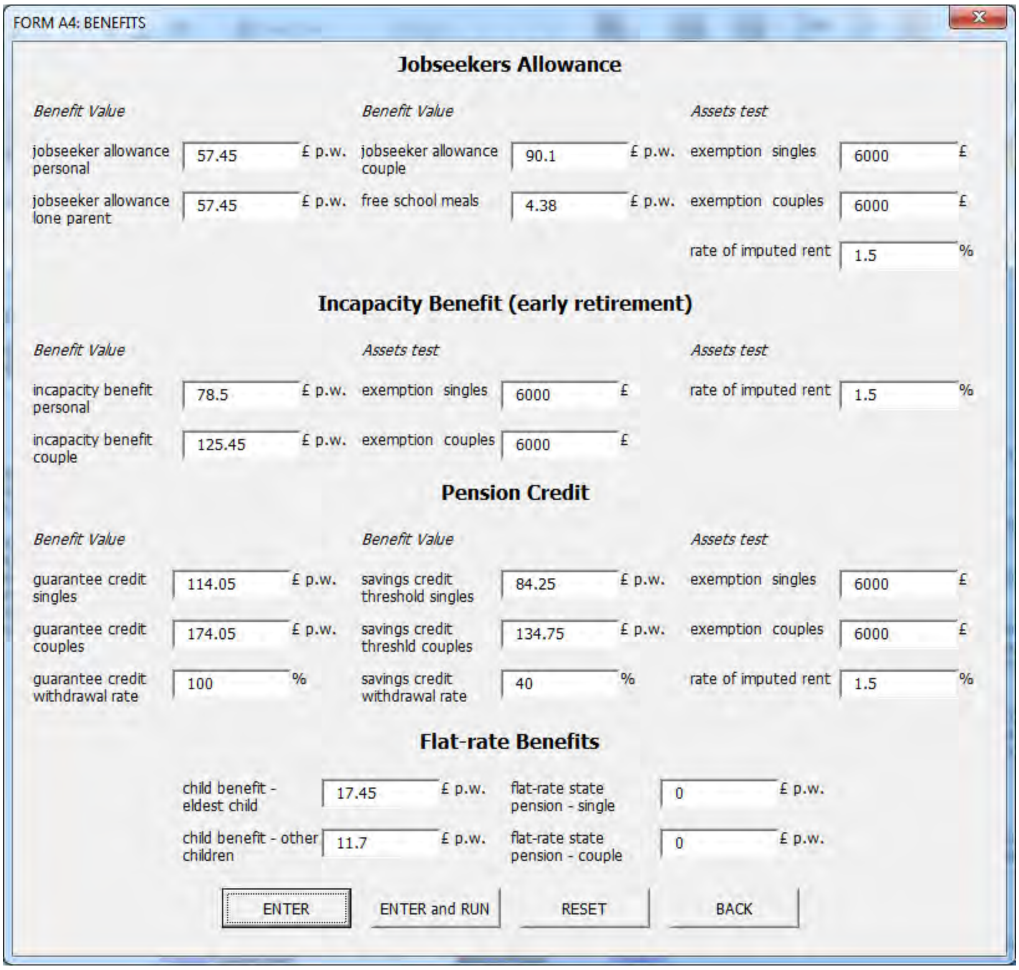
Allowances			*Premia*		
personal allowance	57.45	£ p.w.	family	16.25	£ p.w.
lone parent allowance	63.4	£ p.w.	single pensioner	56.6	£ p.w.
couples allowance	90.1	£ p.w.	pensioner couple	83.95	£ p.w.
children allowance	45.58	£ p.w.	single incapacity benefit recipient	78.5	£ p.w.
			couple incapacity benefit recient	125.45	£ p.w.

Earnings disregards					
single adult	5	£ p.w.	HB taper rate	65	%
lone parent	25	£ p.w.	HB minimum payment	0.5	£ p.w.
couple	10	£ p.w.	CTB taper rate	20	%
additional	14.9	£ p.w.	CTB min payment	0.01	£ p.w.

ENTER | ENTER and RUN | RESET | BACK

Form A3 defines the housing costs and related benefits assumed by the model. Rental costs are based upon the number of children in a household, and Council Tax varies by household relationship status. Matching to survey data suggests that these housing costs should be based on TBMT assumptions for local authority tenants and not private tenants (as had been assumed in the past). This seems a more sensible assumption for those toward the bottom of the distribution. The "Allowances", "Premia", "Earnings Disregards" and taper rates assumed for Housing Benefit and Council Tax Benefit can all be varied here.

FORM A4: BENEFITS



Form A4 allows the user to set the terms for means-tested benefits such as Jobseekers Allowance, Incapacity Benefit, and the Pension Credit, all of which are withdrawn in response to private income.

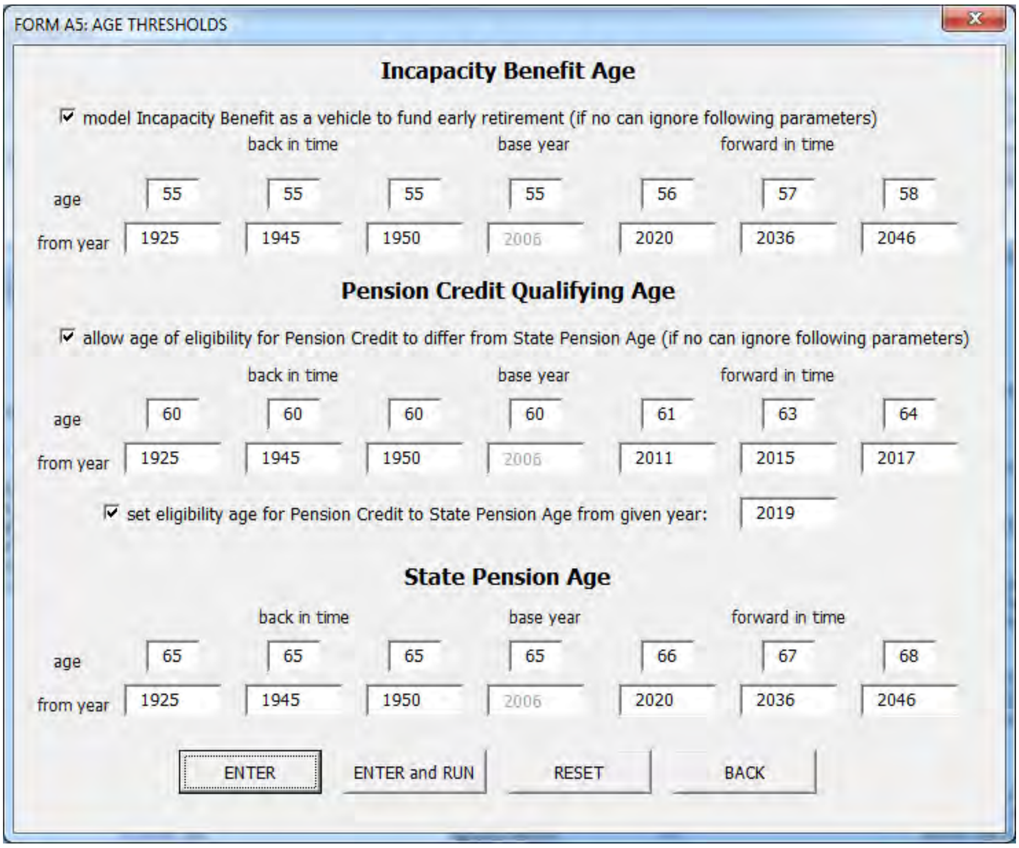
Universal Credit will replace Jobseeker's Allowance. However, the full form will continue to appear when Universal Credit is selected, as Universal Credit is based on some of the same parameters as Jobseeker's Allowance.

Note here that the model only considers Incapacity Benefit as a vehicle to fund early retirement. The model does not allow for heterogeneous health status of the population.

In additional to income tests, assets tests can be applied to means tested benefits. In the pre-packaged tax and benefit schemes, assets tests are accommodated by: 1) calculating an implicit rent flowing from household assets; 2) adding the implicit rent to other household income; 3) applying the aggregate income to the relevant income test.

This form also allows the user to input parameters in relation to flat-rate benefits, namely Child Benefit and the State Pension.

FORM A5: AGE THRESHOLDS

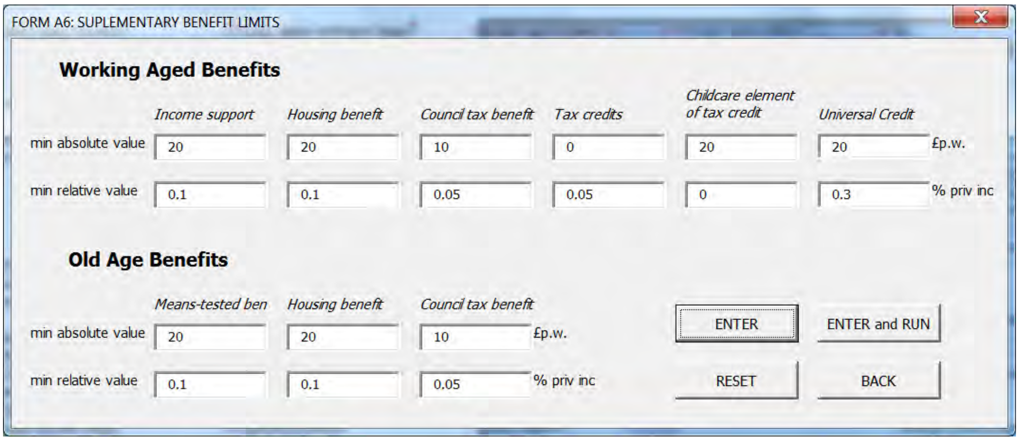


Incapacity Benefit Age: This is the lowest age at which incapacity benefit is assumed to be available as a form of early retirement support. This age can be allowed to vary through time, as indicated by the auto-populated figures.

Pension Credit Qualifying Age: This is the lowest age at which the Pension Credit is assumed to be available. The model allows the qualifying age for Pension Credit to differ from State Pension Age, consistent with current legislation. The Pension Credit Qualifying Age can be allowed to vary through time, and any disparity with State Pension Age can be suppressed from an exogenously specified year.

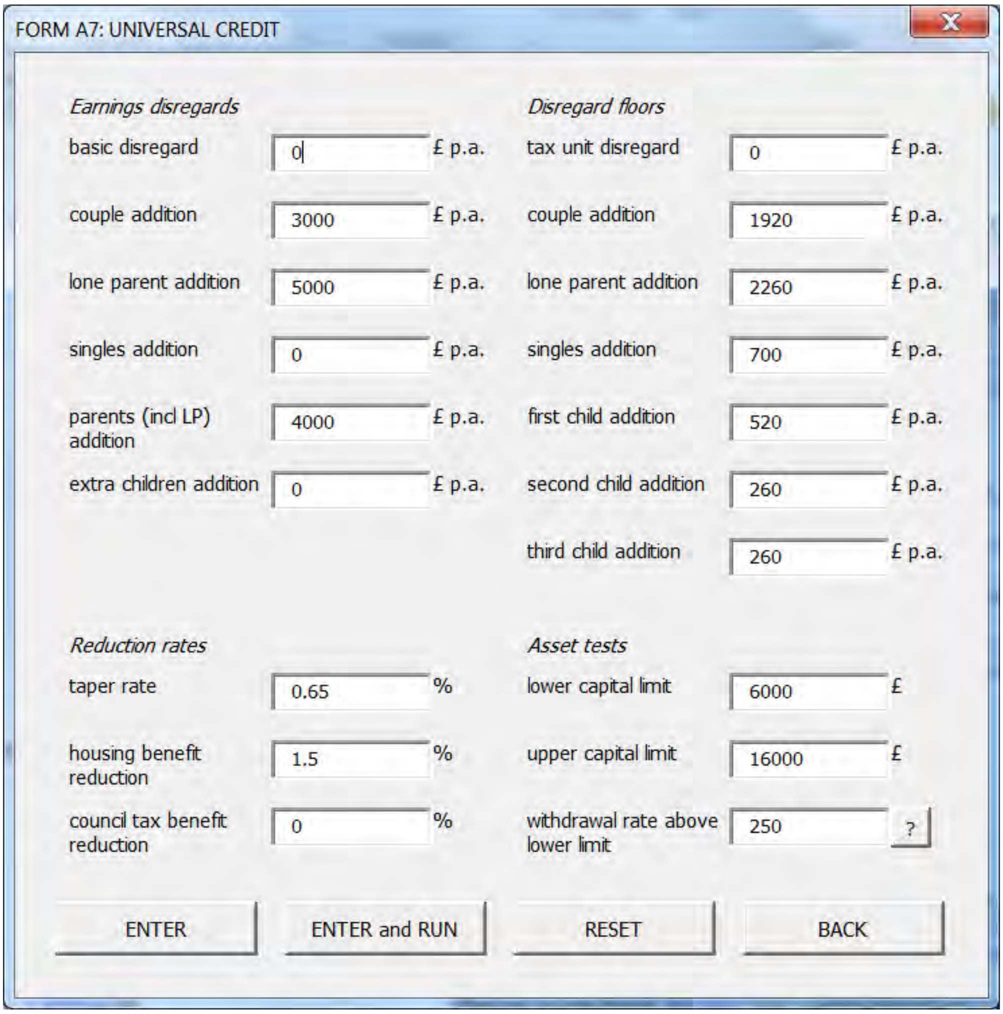
State Pension Age: This is the age from which state pensions are taken. Any employment income after State Pension Age is not subject to NICs and all individuals are assumed to be eligible to the Pension Credit from State Pension Age under the pre-programmed tax structures.

FORM A6: SUPLEMENTARY BENEFIT LIMITS



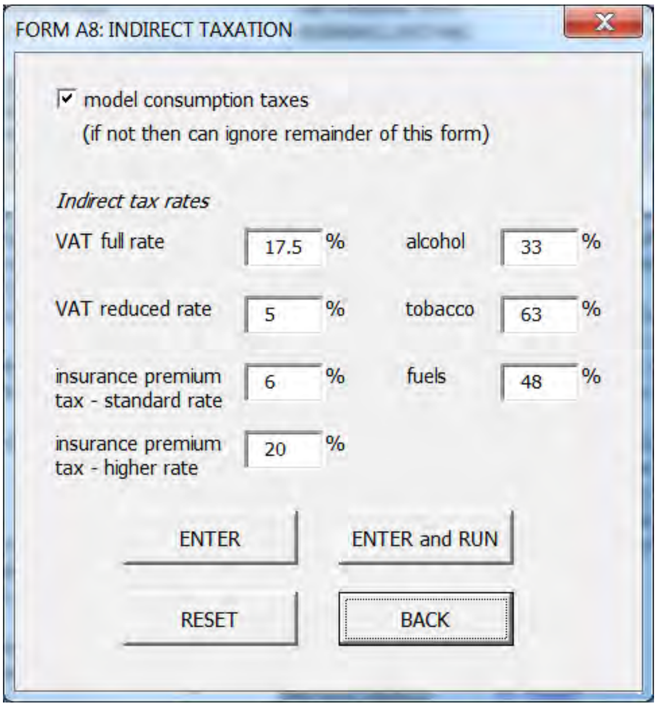
This form is designed to provide a stylised account of incomplete benefits take-up. Two thresholds are applied to each of the schemes that are referred to in this form; an absolute minimum benefit threshold (specified in £ per week), and a minimum relative value specified as a percentage of private income. If the benefit to which a given family fails to exceed either of these thresholds, then the family is assumed not to apply for the associated benefit.

FORM A7: UNIVERSAL CREDIT



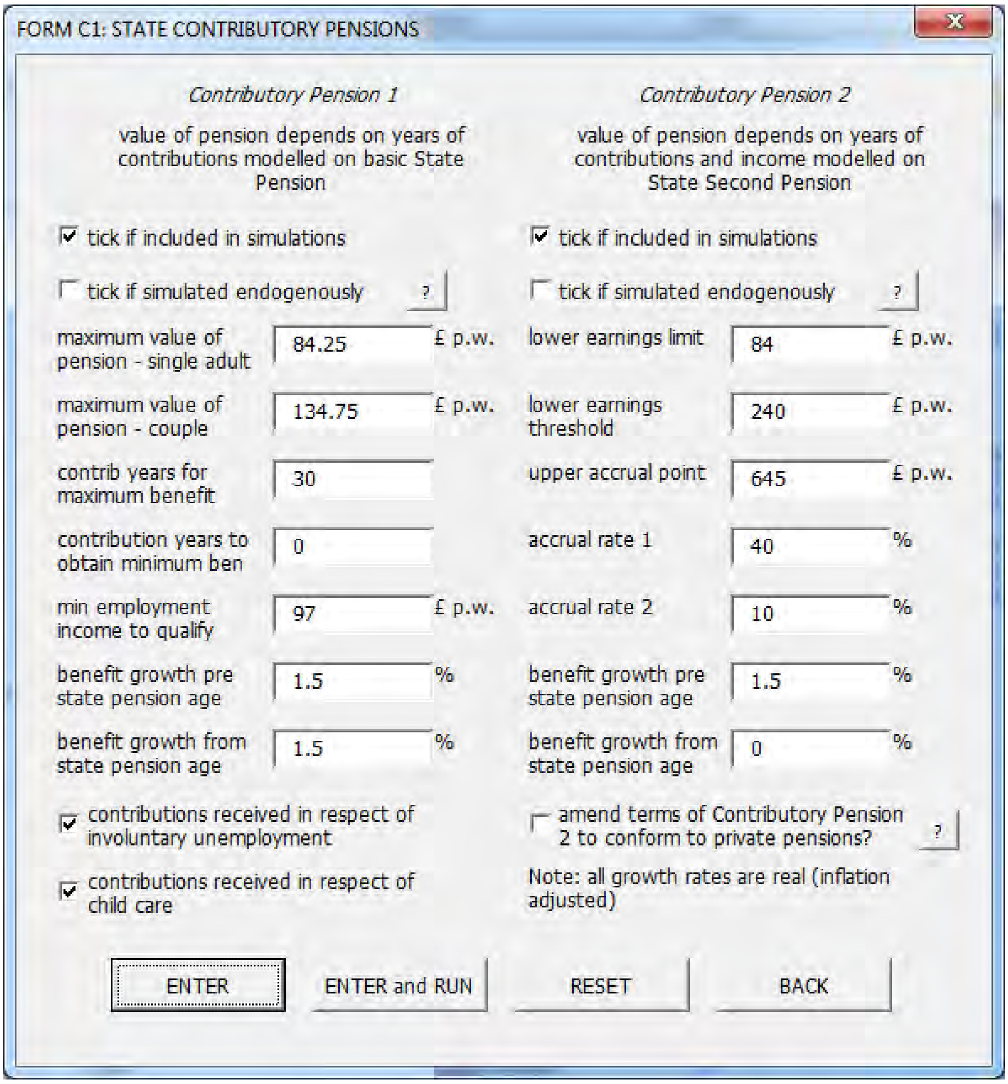
This form allows the user to input parameters defining Universal Credit. It only appears if Universal Credit is selected in Form 1.

FORM A8: INDIRECT TAXATION



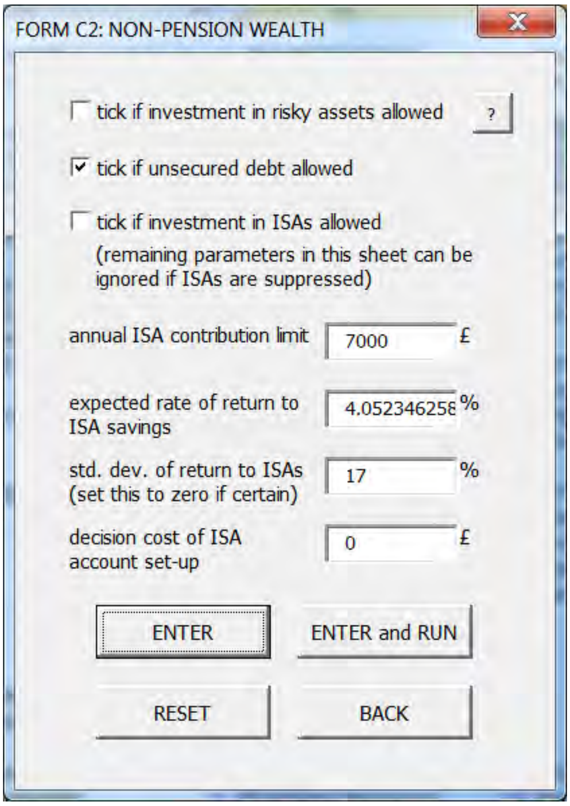
LINDA can be directed to include an allowance for indirect taxes in the simulated analysis. In this case, the model uses reduced form regression equations to disaggregate aggregate consumption (which is simulated endogenously) into the consumption categories that are subject to alternative tax rates. The model therefore accounts for income effects associated with indirect taxes (ie the reduction in aggregate purchasing power), but not price effects (ie the influence of indirect taxes on relative prices of alternative consumption subgroups). Please contact the NIESR for further details.

FORM C1: STATE CONTRIBUTORY PENSIONS



Form C1 allows two forms of State contributory pension to be included in the analysis. Contributory Pension 1 offers a flat-rate increase in the pension payable from state pension age for each year that contributions are accredited during the working lifetime, and is designed to reflect the basic State Pension. Contributory Pension 2 provides pension benefits from state pension age that can increase with earnings during the working lifetime, and is designed to reflect the State Second Pension.

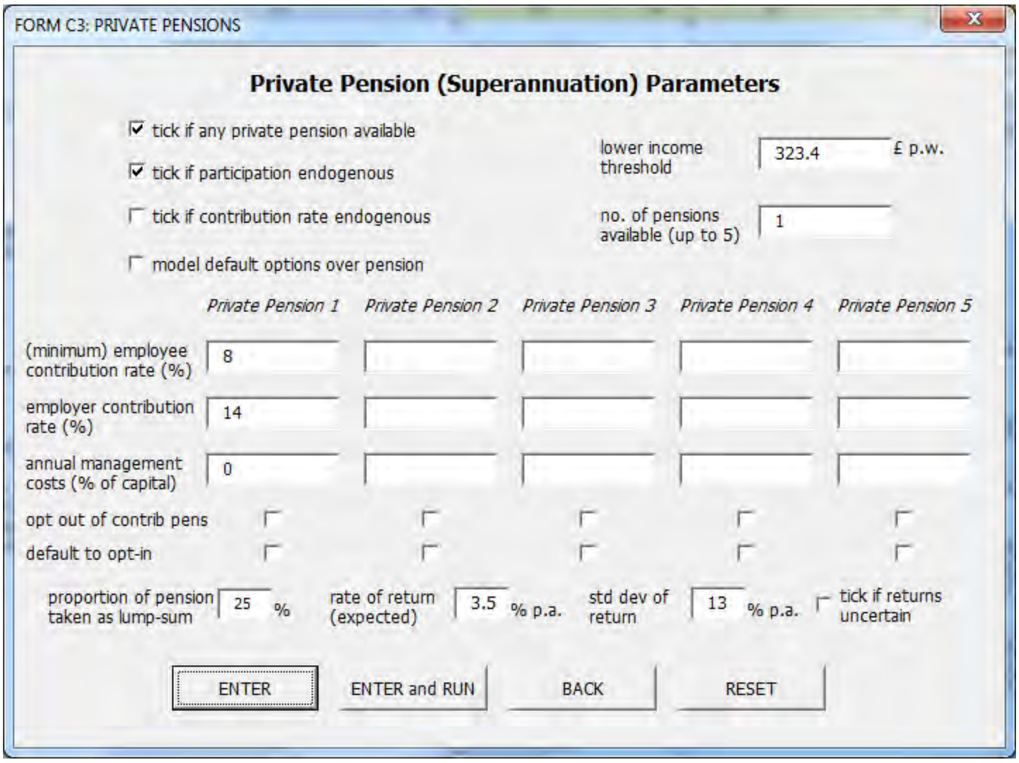
FORM C2: NON-PENSION WEALTH



This form defines parameters that determine the types of non-pension assets that the simulated population has access to. Save liquid assets are allowed for by default. The user can also specify that the simulated population can allocate some of their liquid wealth to a risky investment asset, whether the simulated population has access to (unsecured) credit, or whether they are able to invest in an Individual Savings Account (ISA).

If ISAs are included for analysis, then parameters governing the terms of this asset class can also be amended.

FORM C3: PRIVATE PENSIONS

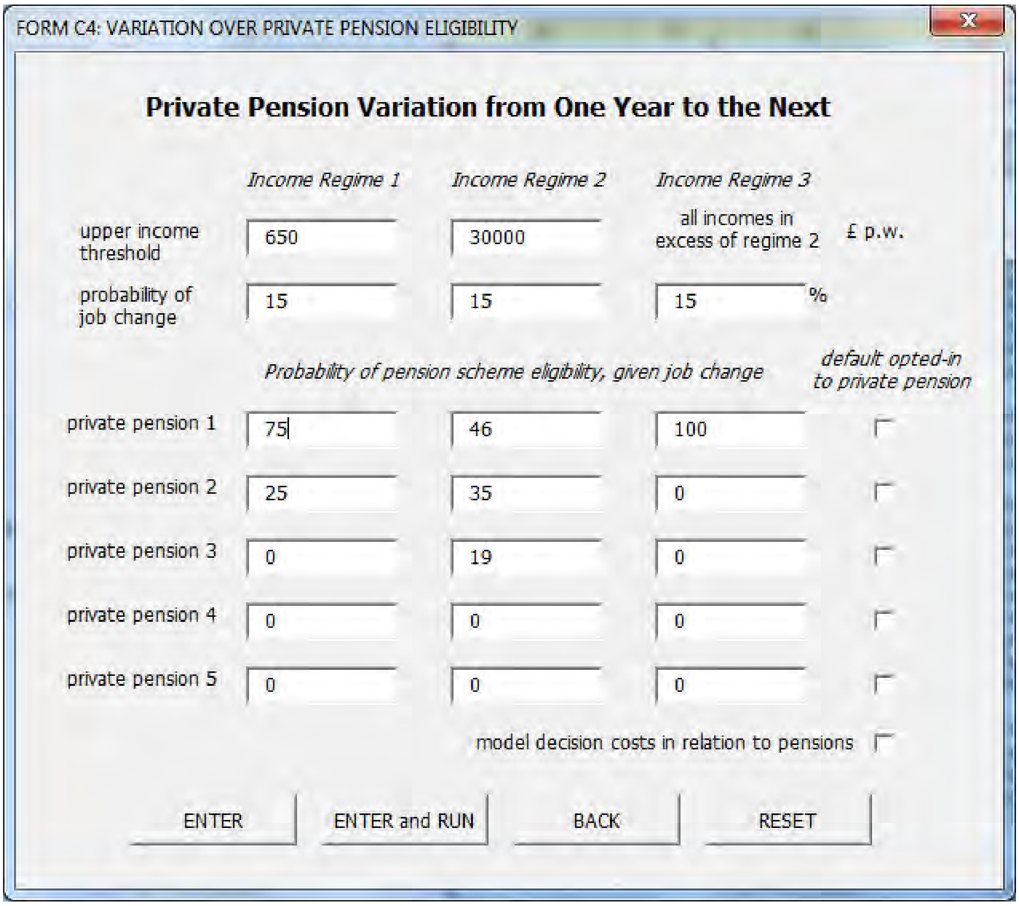


Form C3 allows the parameters of personal pension schemes available in the model to be set. It is necessary to ensure that the check-box “tick if any private pension available” is set to allow for private pensions in the model simulations.

If private pensions are accommodated in the simulations, then it is possible to allow for endogenous decisions regarding pension take-up (participation endogenous), and contribution rates. It is also possible to allow for multiple private pension types in the simulation, which differ from one another over a range of details including (minimum) employee and (fixed) employer contribution rates, and management investment charges (that reduce total annual returns). Pension contributions are specified as a percentage of labour income, and only if labour income exceeds the lower threshold defined in this form.

Parameters defining the rate of return to private pensions, and whether these returns are uncertain can also be specified. If uncertain returns are considered the box should be ticked and the standard deviations of the rates of return set.

FORM C4: VARIATION OVER PRIVATE PENSION ELIGIBILITY

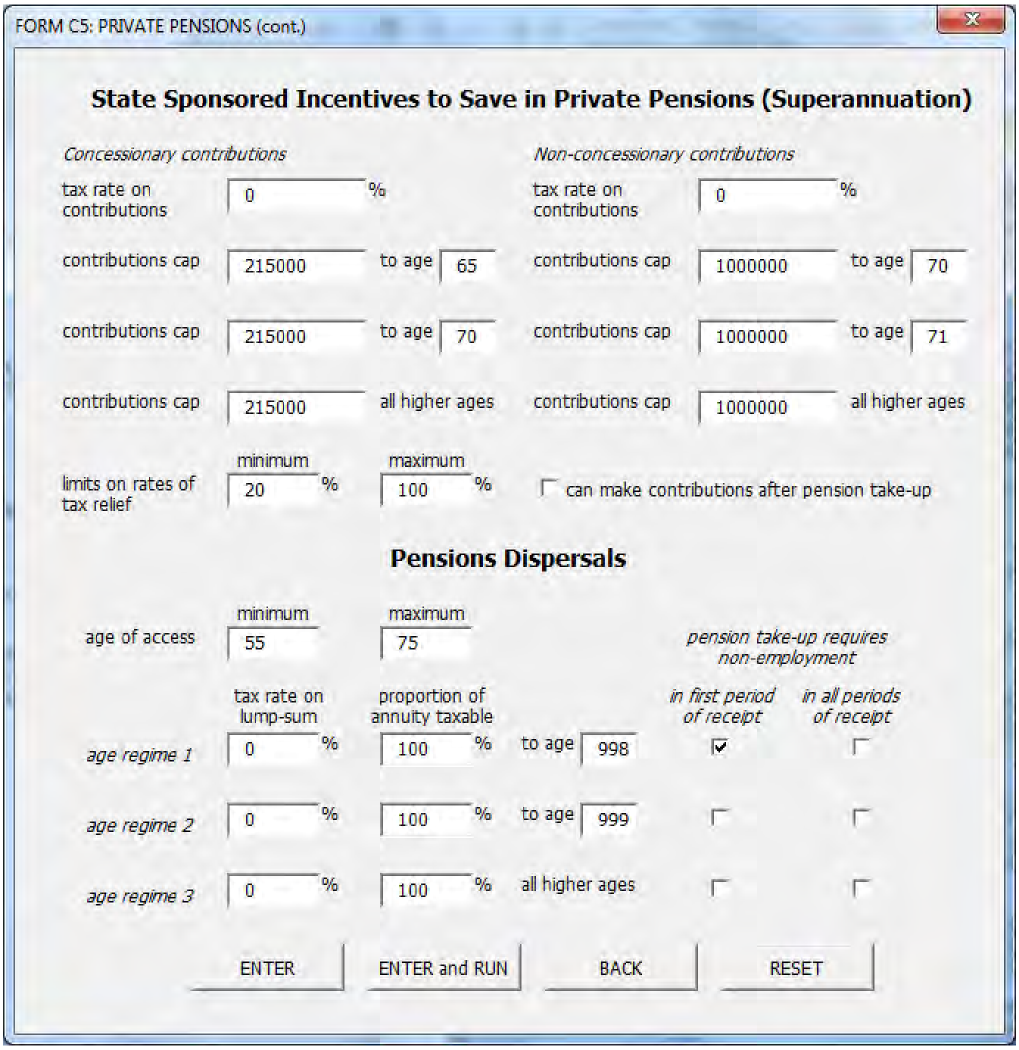


Households are considered to be eligible to participate in only one of the above-defined private pension scheme in any one year. Eligibility to each scheme is identified stochastically with reference to income-dependent probabilities defined in Form C3.

The likelihood that a family is eligible to a given pension scheme in each year depends upon whether they chose to participate (contribute to) their eligible pension in the preceding year and their income. Individuals who chose not to participate in a given scheme receive a new random draw from the available pensions in the immediately succeeding year. In this case, the probability of drawing a scheme varies over three alternative income regimes. Individuals who chose to participate in a given scheme are automatically assumed to be eligible to the same scheme in the immediately succeeding year, unless they experience a job change. A job change has no influence on the family's circumstances, other than to indicate that the pension to which they are eligible is taken as a new random draw (in the same way as it would if they chose not to participate in the preceding year).

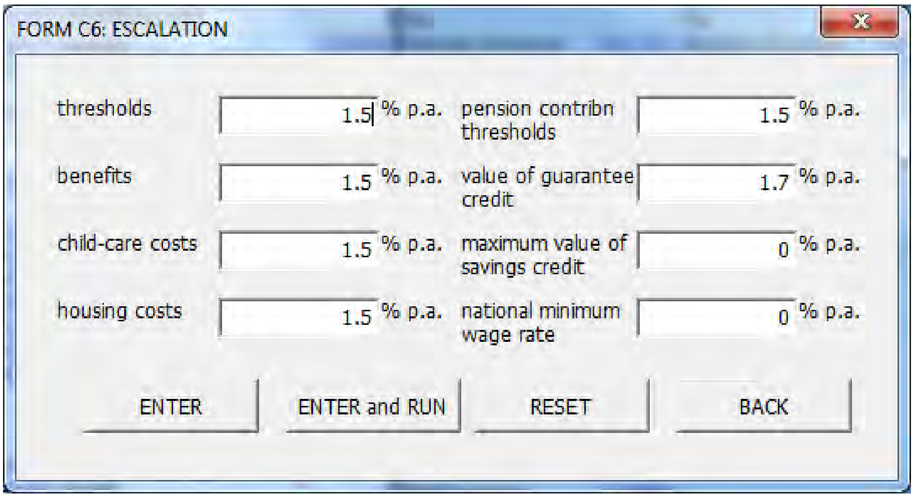
The form further includes tick boxes to impose opting-in as the default decision on selected pension schemes (if the decision to participate is endogenous) and whether decision costs are incurred when deciding against the default option.

FORM C5: PRIVATE PENSIONS (cont.)



This form sets the tax treatment of pension contributions and dispersals. Concessionary contributions refer to pension contributions that receive some tax relief, in contrast to Non-concenssionary contributions. Caps on contributions can be administed within three mutually exclusive age bands, and the tax relief given to concenssionary contributions can also be subject to limits. The bottom panel of this form also allows the terms of pension dispersals to be defined.

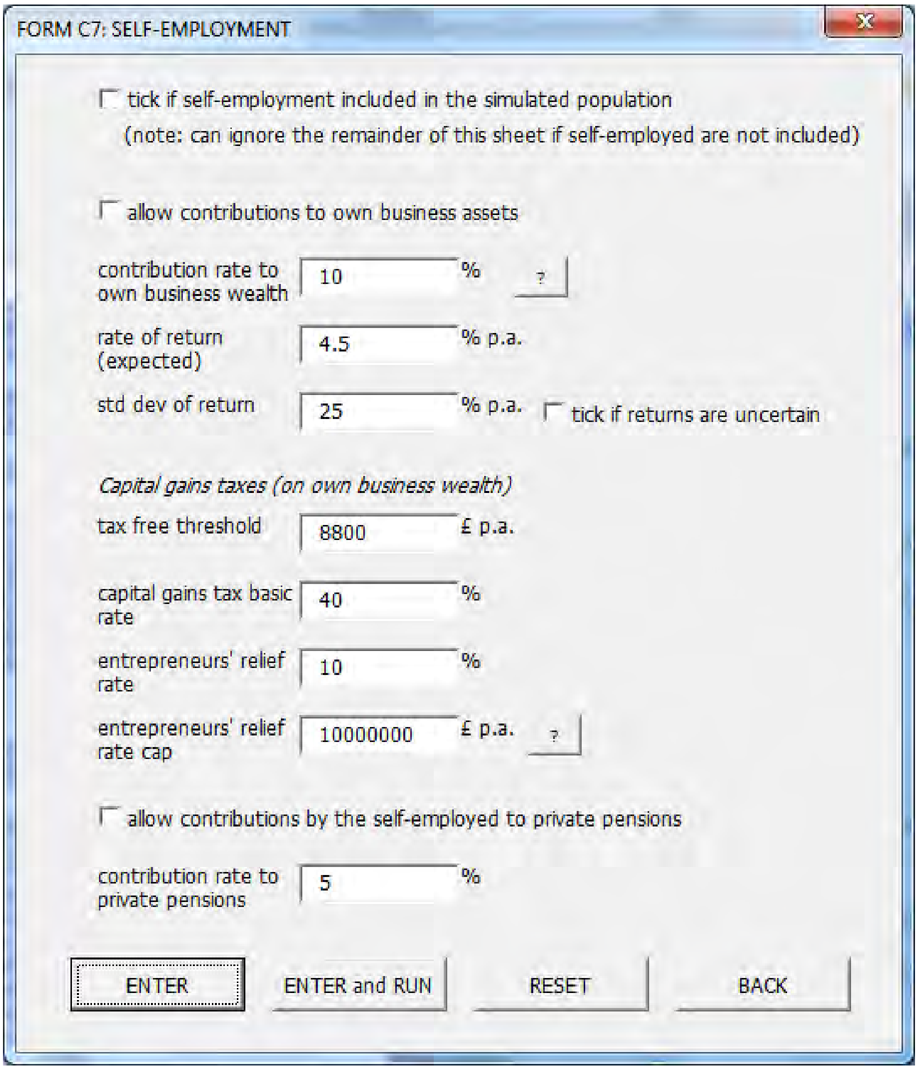
FORM C6: ESCALATION



Here annual growth rates can be defined for tax thresholds, benefit rates, child-care costs, housing costs, and the thresholds used for determining income eligibility of pension contributions**.** It should be noted that there is trend growth of wage incomes of 2.5 % p.a. assumed in the model.

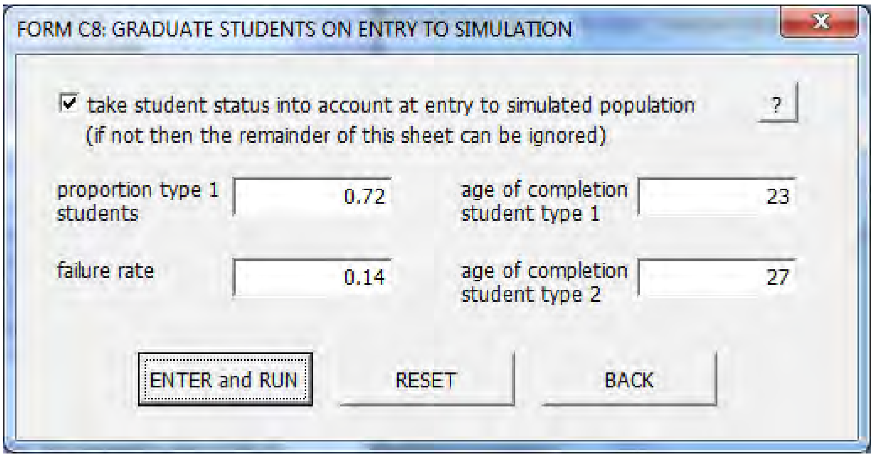
When the form is complete the ENTER button is pressed. The RESET button restores the initial values of all of the parameters.

FORM C7: SELF-EMPLOYMENT



This form allows the user to indicate whether the self-employed should be included in the analysis. If the self-employed are included, then the user can also define whether the model should account for own-business assets, and private pension contributions.

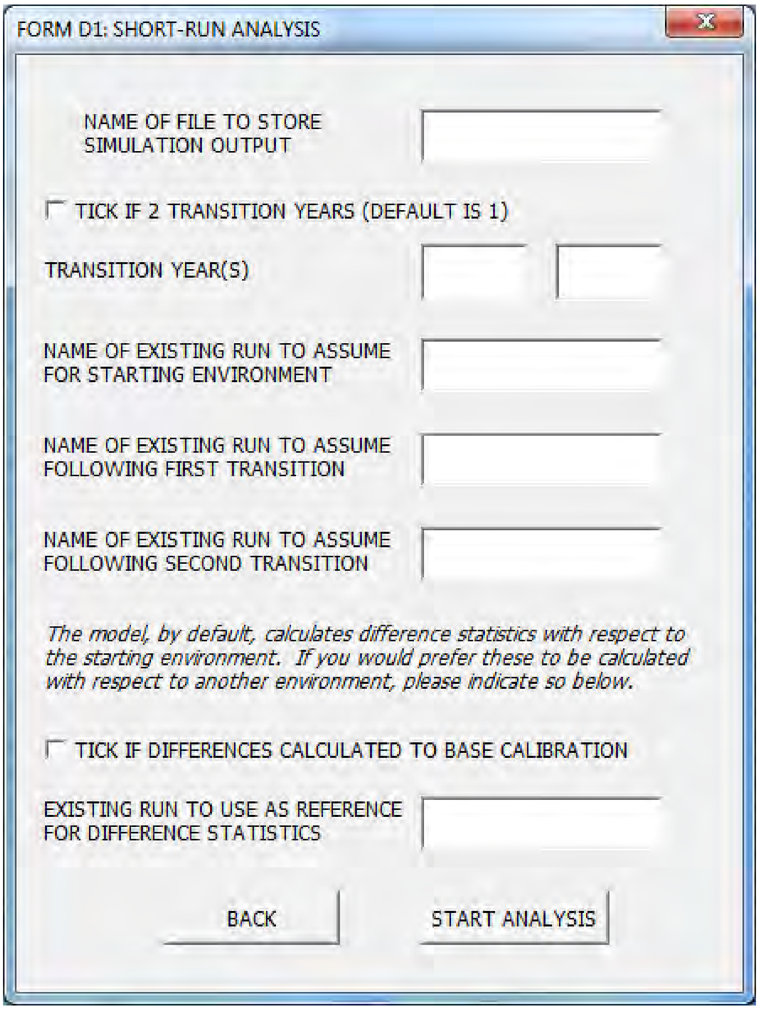
FROM C8: GRADUATE STUDENTS ON ENTRY TO SAMPLE



For the greater part of the simulated lifetime the qualifications of each reference adult remain time-invariant. The sole exception occurs toward the beginning of the simulated lifetime, when the model can be defined to consider the circumstances of students in tertiary education

A student at entry to the sample is considered to remain a student until their respective graduation age. Graduation ages can take one of two values, where type 1 students are considered to graduate before type 2. All individuals identified as students at entry to the simulated sample, and over the graduation age for type 1, but not type 2, are identified as type 2 students. An exogenously defined fraction of individuals identified as students and under the graduation age for type 1 students are defined as type 1 and the remainder as type 2 students. All individuals identified as students and over graduation age 2 are ignored. Achieving graduate status is uncertain and depends upon an exogenously defined failure rate. These parameters can be set in the top half of form C8.

FORM D1: SHORT-RUN ANALYSIS



The LINDA model can be used to gain an appreciation of the medium term behavioural implications of a policy change. Specifically, it is possible to consider how a population cross-section will evolve through time if they were confronted by a series of unexpected changes to the policy environment. Up to two policy changes can be considered during any given model run.

To undertake this type of analysis the user should do the following:

1) Run each of the policy scenarios that are of interest (as described in “Setting up a New Simulation” below), choosing any run names that you prefer.

2) After completing the simulations in (1), re-open “job_file.xls”, start the *SIDD* macro, and press the “UNDERTAKE SHORT RUN ANALYSIS” button.

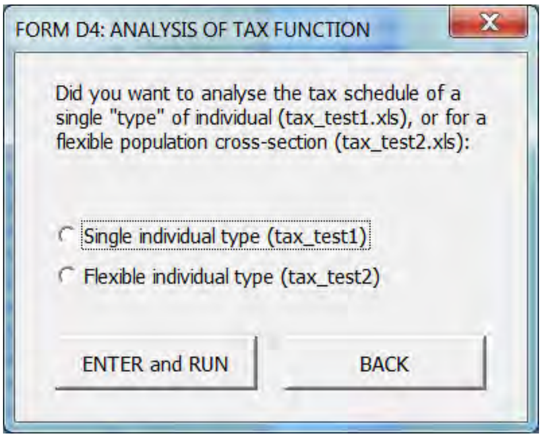
3) Enter the name of the folder in which to store the short-run output in the top text box of Form D1

4) Enter the transition year(s) in the text boxes as indicated. The second (right-most) box should only be used if two transition years are indicated, and should be greater than the first.

5) Enter the run names defined in (1) in the remaining text boxes.

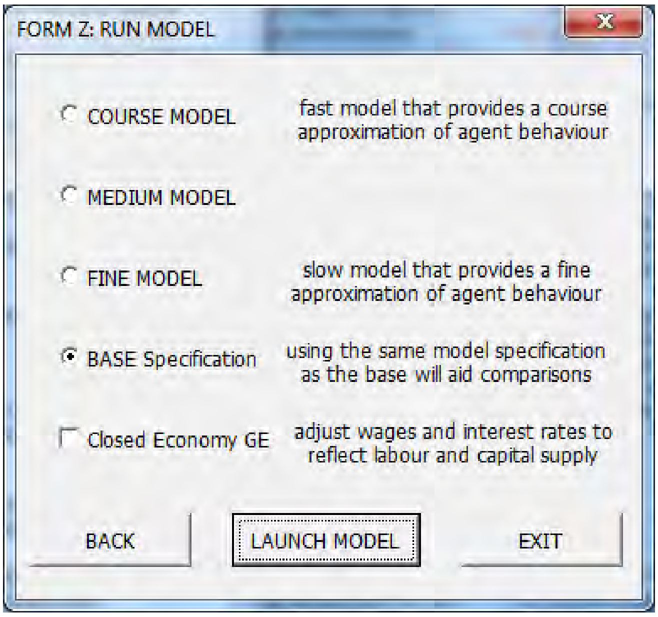
6) Press the "START ANALYSIS" button.

FORM D4: ANALYSIS OF TAX FUNCTION



The model includes two methods for analysing the tax function. The first focuses upon a user-defined household type, with specific measures of private earnings, and the second considers a set of broad circumstances that are designed to capture a population cross-section. Input statistics, and analytical results for each of these two routines are communicated, respectively, through the files "tax_test.xls" and "tax_test2.xls", located in the assumed base directory (defined in cell E27 of the *input* sheet of "job file.xls").

FORM Z: RUN MODEL



This final form allows the user to run the model, to exit the model, or to go back to Form C8. The analyst can choose one of four options for running the model. Running the "course model" achieves a fast run-time, but at the cost of numerical accuracy. This option should be used for exploratory analyses only. At the other end of the scale, running the "fine model" implies a relatively long run-time, to obtain a high degree of numerical accuracy. An intermediate option between these two extremes, and the option to adopt the same specification as the base simulation are also available.[4]

Finally, the user can direct the model to adjust capital and labour prices to reflect a General Equilibrium in a closed economy. This option is discussed at further length in the following section.

4. Simulation Output

The model generates by default two levels of statistic for each simulation and saves these in the sub-directory "simulations\xxx\", where "xxx" refers to the name given to the specific simulation (see discussion on "form 1" above). The model produces, for each family in the reference population cross-section, simulated panel data for a range of characteristics over the life-course. These "micro-data" are reported in a standard format (csv), and can be analysed by widely available statistical packages.

Secondly, the model also generates a core set of summary statistics for each simulation, which are reported in two excel files: "DA1.xls", and "hi_level_statistics.xls".

Each of these respective model outputs is discussed separately below.

[4] All three execution modes of the model are based upon linear interpolation methods, solve using linearlised Euler conditions, and omit non-concavity checks. The modes differ in the number of grid points that are assumed for liquid wealth, labour income, and pension state variables.

Household level micro-data

The model stores the following data in the form of Comma Separated Variable files:

age: age of reference adult
alc_dy: alcohol duty paid (£ per week)
bcohort: birth cohort
ben: aggregate state welfare benefits received (excluding state pensions, £ per week)
ben_unit: benefit unit number in household (imputed into WAS)
cons: non-durable discretionary consumption (£ per week)
costs: non-discretionary housing and childcare costs (£ per week)
cp1: first state contributory pension (BSP), fraction of full-basic state pension
cp2: second state contributory pension (S2P) (£ per week)
cpinc: state pension income (cp1/2) received (£ per week)
death_age: age at which reference adult dies
dppart: flag = 1 if default is to participate in personal pensions / 0 to not participate
education: highest education level achieved during the simulated lifetime
emp1/2: employment status of reference adult / spouse
eqs: equivalence scale
ful_dy: fuel duty paid (£ per week)
hhno: household number as reported in WAS
hsgret: unrealised capital gain on housing wealth (£ per week)
humcap: household human capital (wage potential) (£ per week)
humcap2: NOT APPLICABLE
ihr_dy: insurance, higher rate duty paid (£ per week)
inherit_age: age at which inheritance received
inherit_val: value of inheritance (£)
isa: wealth held in Individual Savings Accounts (ISAs, £)
isa_cont: contributions to ISAs (£ per week)
isr_dy: insurance, standard rate duty paid (£ per week)
labinc: labour income of household (£ per week)
leis: proportion of time spent in leisure
mdr: marginal deduction (tax) rate
na: number of adults in household (1=singles 2=couples)
net: household net (disposable) income (£ per week), excluding returns to housing
nic: National Insurance Contributions of household (£ per week)
nk: aggregate number of dependent children
nk_allX: number of children, in birth age X
obw: own business wealth (£)
OPcont: NOT APPLICABLE
OPpen: NOT APPLICABLE
OPpenb: NOT APPLICABLE
parttr1/2: participation tax rates of reference person / spouse
peninc: private pension (PP+OP) income received (£ per week)
ppc: private pension contributions (£ p.w.)
PPcont: aggregate contributions to private pension (including employer contribution) (£ p.w.)
PPcr: contribution rate to private pension
PPpen: aggregate accrued rights to private pension, defined as an annuity stream (£ p.w.)
PPpenb: aggregate accrued rights to private pension, defined as a wealth equivalent (£)
prec: age at which pension income first received
prett: pre-tax and benefit household income (£ per week)
ret: whether household defined as retired for pension purposes

ri: proportion of wealth invested in risky assets
risky_r: return to risky assets
semp: self-employment flag (=1 implies self-employed, if working)
sim_weight: household weighting variable
student: student status
taxX: household tax burden paid in tax band X (£ per week)
taxagg: aggregate household tax burden (£ per week)
tbc_dy: tobacco duty paid (£ per week)
train_decis: NOT APPLICABLE
training: NOT APPLICABLE
traintime: NOT APPLICABLE
user_toX: user defined output X from tax routines
Val: measure of expected lifetime utility
vat_rr: reduced rate VAT paid (£ per week)
vat_sr: standard rate VAT paid (£ per week)
w: liquid wealth of household (£)
w2: aggregate household net worth (£)
wage_offer: flag = 1 if reference adult receives wage offer, 0 if they do not
wage_offer2: flag = 1 if spouse receives wage offer, 0 if they do not

The following relationships exist between simulated variables:

$$w_{t+1} = w_t + net_t + hsgret_t - cons_t - costs_t - isa\_cont_t - sum(indirect\ taxes)_t \quad (1)$$

$$net_t = prett_t + ben_t - taxagg_t - nic_t - ppc_t \quad (2)$$

$$investment\ income_t = prett_t - labinc_t - peninc_t + hsgret_t \quad (3)$$

Note: only realised housing returns are included in the reported measure of disposable income

Note: whereas equation (2) holds throughout the simulated lifetime, (1) is subject to variation in respect of relationship transitions, liquidity constraints, and about the age of pensions take-up.

Default summary statistics

High-level summary statistics - hi_level_statistics.xls

The high level statistics that are reported by the model fall into two broad categories, which are each represented by a separate table. Simulated population averages for key household income, consumption, and balance sheet items are reported in one table, and statistics relating to the macro-economy are reported in the other. The population averages that are reported in the first table are reasonably self-explanatory. The statistics relating to the macro-economy, however, warrant future comment.

Economic analyses of the type for which LINDA has been devised can typically be distinguished in relation to their treatment of factor prices. On the one hand, *partial equilibrium* analyses assume that factor prices are fixed; this is also commonly referred to as the small open economy assumption. On the other hand, the alternative assumption is that factor prices adjust endogenously to reflect changes in demand and supply. In the case of LINDA, the analyst can choose between these two basic frameworks, as noted in relation to Form Z. Here, directing the model to generate results for the General Equilibrium in a closed economy results in the model adjusting interest rates and wage rates to reflect changes in the supply of capital and labour that are implied by the respective simulation.

The basic idea is that, if saving increases, then this raises the supply of capital in the economy. Increasing the supply of capital in the economy, all else held fixed, should reduce the rate of return paid to capital. This is simulated by LINDA on the assumption that aggregate production, *Y*, combines aggregate capital, *K*, and labour, *L*, in the form of a CES function:

$$Y = A\left(aK^{1-1/\varepsilon} + (1-a)L^{1-1/\varepsilon}\right)^{\frac{1}{1-1/\varepsilon}}$$

Assuming that factor markets are perfectly competitive implies that capital and labour are paid their respective marginal products, so that the interest rate, *r*, and wage rate, *w*, are given by:

$$r + \delta = \frac{\partial Y}{\partial K} = AaK^{-1/\varepsilon}\left(aK^{1-1/\varepsilon} + (1-a)L^{1-1/\varepsilon}\right)^{\frac{1/\varepsilon}{1-1/\varepsilon}}$$
$$w = \frac{\partial Y}{\partial L} = A(1-a)L^{-1/\varepsilon}\left(aK^{1-1/\varepsilon} + (1-a)L^{1-1/\varepsilon}\right)^{\frac{1/\varepsilon}{1-1/\varepsilon}}$$

where δ is the rate of depreciation of capital. The aggregate capital stock is set equal to aggregate household wealth less government debt, where age specific averages generated by the model are weighted to match household numbers by age described by 2001 census data. Similarly, aggregate labour supply is set equal to the aggregate wage bill, and these two (capital and labour) are combined to give aggregate production as described by the CES function referred to above. Investment is $I = (\delta + g).K$, where *g* is the assumed growth of the economy. Government consumption, *G*, is then calculated to equate aggregate income to expenditure:

$$G = Y - C - I$$

where *C* is aggregate household consumption. It should be noted that we do not consider the issue of the government budget balance because we do not cover all of the forms of taxation that are applied in practice (for example, LINDA currently omits taxation of firms).

When directed to generate results for the General Equilibrium in a closed economy, the model iteratively adjusts interest rates, *r*, and wage rates, *w*, until these are consistent with the associated aggregates generated for output, capital and labour supply. Note, however, there is no guarantee that the model will converge. The analyst is given a warning if the model has failed to find a solution after searching over 25 alternative parameter combinations, and can choose to continue with the analysis or cancel out in that case.

A broad selection of summary statistics – DA1.xls

The spreadsheet DA1.xls that is generated by default for each simulation reports a series of summary statistics that have been identified as useful by policy makers. A wide range of statistics are reported, and the excel file is designed to be fairly self-explanatory.

The code that is used to calculate the statistics reported in DA1.xls is provided in the “ANALYSIS solution” that is provided with the model (see Section 5 for details). Please contact the NIESR if in doubt concerning definition of the statistics reported in the excel file.

Optional summary statistics

High level analysis of population cross-sections and birth cohorts

The model can be directed to report age specific summary statistics for selected population cross-sections and birth cohorts. Where the model generates statistics for a cross-section, then it will save results into a file named "NNN YYYYcs.xls", where NNN is replaced by the simulation name, and YYYY by the year of the relevant cross-section. Similarly, birth year statistics are saved into a file named "NNN YYYYby.xls". This analysis routine delivers age and relationship specific moments for labour supply, employment income, disposable income, consumption, private pension participation, and wealth, each in a separate worksheet. Statistics are reported in both tabular, and graphical form.

Decile-level summary statistics for population cross-sections and birth cohorts

A workbook analysis_dec.xls is generated by the programme and is placed in the sub-directory given by the model name. Users are likely to be interested in both the absolute values and the differences from the base. The differences are generated in differences_dec.xls. Users have two options with regard to the way in which decile groups are defined: they can choose to define deciles on characteristics associated with a prevailing simulation, or they can choose to maintain the population groups defined by the base simulation. When the latter of these two options is selected, then the letter "b" is appended to the end of the associated output file names. The default for the model is to group households on the basis of simulation specific characteristics. To apply population subgroups defined by the base simulation, the analyst should enter the value 1 into Cell Y58 of the "input" sheet on the associated job file.

Distributional measures depend on the variable used to rank households. The output files associated with the decile analysis each include two work sheets. "sim_output_all" ranks individuals by total wealth, and "sim_output_lifetime" ranks by average net income earned during the entire simulated life-course. The threshold cut-offs that define respective deciles are also reported in the right-most columns of each sheet.

The decile level analysis is carried out for the following variables:

Table 1 Data in Analysis_Dec.xls and Differences_Dec.xls

Variable	Notes and Definitions.
number of adults	Shows how household size changes with age of family reference person
consumption	Consumption, including non-discretionary costs met by the household for housing and child care. (£ per week)
leisure	This is the proportion of the leisure available to someone who does not work. Someone who works full-time is defined as having leisure of 0.5. The ratio of the part-time working week to the full-time working week is user-determined in Form 1 as Part-time Ratio (1) for single people and the Lab Ratio variables for couples.
proportion employed	Proportion of population with any employment
gross labour income	Average income from employment (£ per week)

average contribution to private pension	Average employee and employer contributions to private pensions over decile (£ per week)
propn contributing to private pension	Proportion of decile making any contributions to private pensions
private income	private income = (gross labour income) + (investment income on non pension wealth) – (interest on debt) – (private pension contributions) (£ per week)
disposable income	disposable income = private income + taxes – benefits (£ per week)
unsecured debt	Average value of unsecured debts by population decile (£)
propn of population with unsecured debts	Proportion of decile with negative net liquid wealth (cash on hand)
net non-pension assets	Average value of net non-pension assets by population decile (£). Based only on data for households without unsecured debt.
pension wealth	Average value of assets held in pensions (£)
total wealth	total wealth = (net non-pension assets) – (unsecured debt) + (pension wealth) (£)
value function	Remaining life-time welfare. Utility Units. Percentage rather than absolute differences should be considered.
upper threshold for decile	Threshold used to allocate households to deciles as considered in relevant worksheet

Comment [P1]: y - t + b ?

As is implicit in the above discussion, the sheet shows arithmetic differences for all variables except the value function. Differences in expected lifetime utility between a given simulation and the associated simulation base are expressed either as compensating variations (£ of liquid wealth equivalents) or percentage differences, as little meaning can be given to absolute differences.

5. Adjusting the Program Code

The model is programmed in Intel Visual Fortran, and two aspects of the model code can be amended by the user: the routines that calculate taxes and benefits in the model, and routines that undertake analysis of simulated output. These routines can be edited through Microsoft Visual Studio 2010 at the time of writing, and the steps involved in setting-up this environment for use are provided in the second section of this manual ("Setting-up the Model"). In this section we describe the basic programming structure, outline how to create a new analysis routine, and provide an overview of how the existing code is organised to calculate taxes and benefits.

Basic Code Structure

- The Fortran code that is provided with the model is organised into four broad structures, which can usefully be thought of as containers.

- The largest container is the “solution”, which is a set of files that comprise the basic building blocks that Fortran uses to generate program files
    - The “ANALYSIS solution” can be opened by double-clicking on the file “analysis.sln” in the “FORTRAN\ANALYSIS\” subdirectory.
- The second largest container is the “source file”, into which code is written.
    - You can browse through the source files of a solution via the “Source Files” folder of the “Solution Explorer” (which can be seen by selecting “Solution Explorer” from the “View” menu of Visual Studio)
    - To add a new source file to a solution:
        - right click on the Source Files folder
            - select “add”, “new item”
            - select “Fortran Free-form File (.f90)”
            - make up a name for the file toward the bottom
            - and press the “Add” button
- Each source file can contain one, or a number, of MODULEs.
    - A MODULE is predominantly a container to organise a number of SUBROUTINEs.
        - e.g. MODULE AA might contain SUBROUTINEs AA1 and AA2; and MODULE BB might contain SUBROUTINEs BB1 and BB2
        - a slight complication arises in relation to “global variables”, which is returned to below.
- Most MODULEs are organised as follows:
    - MODULE AA
        - This line denotes the start of the MODULE with the name AA
    - IMPLICIT NONE
        - This line is necessary to avoid easy programming errors – do a google search on it for further detail
    - CONTAINS
        - This line notes that the SUBROUTINES that follow are contained within the MODULE
    - **** SUBROUTINEs then appear here ****
    - END MODULE AA
        - This line denotes the end of the module
- All of the program computations are undertaken by code that is organised within a series of SUBROUTINEs.
    - e.g. SUBROUTINE AA1(x, y) takes a series of inputs x, performs a number of calculations, and then returns a series of outputs y.
        - We would execute this subroutine by entering the following code: call AA1(x,y)
- To use (call) SUBROUTINE XX from within SUBROUTINE YY, either:
    - the two SUBROUTINEs must be organised within the same MODULE, or SUBROUTINE YY must be given access to the MODULE containing SUBROUTINE XX
        - e.g. in the above example SUBROUTINE AA1 could call AA2 by default, but would need to be given access to MODULE BB to call BB1 (or BB2)
- Each SUBROUTINE must be organised as follows (based on the above example):
    - SUBROUTINE AA1(x, y)
        - This line denotes the start of the subroutine, and the variables that are used as inputs and outputs – if the subroutine takes in no explicit inputs, and produces no explicit outputs, then we write “SUBROUTINE AA1()”
    - USE BB
        - This line gives SUBROUTINE AA1 access to the SUBROUTINEs contained in MODULE BB

        - This line is only required if you want to access SUBROUTINES (or global variables) stored in MODULE BB
    - IMPLICIT NONE
        - This line is necessary to avoid easy programming errors – do a google search on it for further detail
    - real (8) :: x, y
        - This line of code refers to the "type definitions", and usually covers a number of lines.
        - A "type definition" tells fortran the explicit nature of the data that each variable contains.
        - The variables that you will most commonly require will be limited to real(8) (a number with a decimal point) and integer(4) (a whole number without any decimals) types.
            - A variable cc is assigned a real type by; real(8) :: cc
            - A variable cc is assigned an integer type by; integer(4) :: cc
            - If cc is a matrix of real numbers with dimension (5,4) (5 rows and 4 columns), then it is assigned by; real(8) :: cc(5,4)
        - You must assign types to all of the variables that are included as inputs and outputs to a given subroutine (e.g. x and y in the example here)
        - You must also assign types to all of the variables that you use within the subroutine, and which are discarded after the subroutine is complete
            - Variables discarded after a subroutine is complete are commonly referred to as "local variables".
    - **** You then add in programming code to undertake your desired calculations here ****
    - END SUBROUTINE AA1
        - This line denotes the end of your subroutine

The above covers just about everything you will need in relation to program structure. There is, however, one final complication. Fortran requires each variable that is used in any subroutine to be assigned a type (real / integer above). In most cases, the variables that you use will either be explicit inputs / outputs of a subroutine, or will be "local variables" that you don't mind discarding after your desired computations within the subroutine are complete. Nevertheless, there are a number of variables that you might want to make common to a range of subroutines, without needing to repeatedly pass these variables as explicit inputs to each subroutine. Examples in relation to tax and benefits calculations include the number of adults and children in a household, the employment status of adult household members, measures of gross income, and so on. This is achieved in the code using "global variables".

- You will find in the set of source files included with the TAX program, one called "2_global.F90".
- If you open this file, then you will see that it includes a module named "global_tax".
    - The MODULE global_tax includes a series of variable type definitions, and no subroutines.
    - The variables defined within this module are referred to as "global variables". This is because it is possible to share them between alternative subroutines without the need for explicit declarations.
- The global variables defined in MODULE global_tax are assigned values within the SUBROUTINE initialise_taxinputs (see Figure 1), found in the source file "1_TaxTools.f90"

    - Note that SUBROUTINE initialise_taxinputs is given access to the global variables by the "USE global_tax" declaration at line 23
- Any SUBROUTINE that subsequently requires access to the global variables need only include the declaration "USE global_tax" in its second line of code (as outlined above for SUBROUTINE structure)

Programming ANALAYSIS Routines

In this section we provide an outline of the structure of the "analysis solution" that is provided with the model. We then work through a practical programming example that new users may find useful, before describing how a new analysis routine should be included in the solution.

Outline of the "analysis solution"

The "analysis solution" can be opened in the VS 2010 environment by double-clicking the file "analysis.sln". The program files of this solution are organised as follows:

1) The file "0_entry.f90" is the point of entry into the program structure.
2) A series of global variables are initialised, which describe simulated micro-data and high-level characteristics.
    a. you can take a look at what the characteristics are by:
        i. open file "1b_simdata.f90": the population characteristics are then listed in the comment immediately under the start of the module.
        ii. Open file "1_GlobalParam.f90": the high-level population characteristics are listed in the comment under the start of the module
        iii. A full list of the all global variables is provided in the table at the end of this section
3) Each requested analysis is then run
    a. Developer routines, which we may alter from time-to-time are provided in the file "DeveloperRoutines.f90".
    b. If you decide to include a new analysis routine of your own, then you should add this to the file "UserAnalysisRoutines.f90". We will not amend this file, and you should take care to preserve your own version of this file whenever you update your version of the model.
4) Problems are then reported
5) And the routine is exited

Common programming steps

It is beyond the scope of this manual to provide detailed advice concerning programming of new analysis procedures, which is best understood by obtaining hands-on experience. Here we provide a brief over-view of the steps involved in editing Fortran code, and some advice concerning use of the debugging environment.

The following steps are usually involved when editing the Fortran code:

1) Open a "program solution" (see the section on "Basic Code Structure" for a description of a "solution").
2) Edit the program text
3) Save the revised text

4) “Re-build” the solution in a “debug configuration”
    a. This essentially tells Fortran to use the new building blocks to create new program files
    b. The debug configuration is designed to enable you to check that your revised program works as intended
5) Run some test analyses to make sure that the revised code works as intended
6) “Re-build” the solution in a “release configuration”
    a. The release configuration is designed to omit the checks implemented in the debug configuration, and typically works much faster as a result
7) Run your desired analyses

The following will walk you through how to use the debugger, essentially addressing steps (1), (4), and (5) listed above.

Opening a “Solution”

1) Double click on the file “ANALYSIS.sln” in the subdirectory “FORTRAN\analysis\”
    a. This should open up the Visual Studio (VS) programming environment
    b. If you followed the steps set out in the Section 2, this solution is now ready for editing (please ensure that this is the case).
2) In the “solution explorer” window, expand the “source files” folder
    a. If the “solution explorer” window is not open, then you can open it by selecting the “View” menu at the top of VS, and then “Solution Explorer”
    b. The following source files should be listed under the “source files” folder:
        i. 0_entry.f90
        ii. 1_GlobalParam.f90
        iii. 1b_simdata.f90
        iv. 2_sorting.f90
        v. DeveloperRoutines.f90
        vi. UserAnalysisRoutines.f90
3) Double click on the file “DeveloperRoutines.f90”
    a. This should open a text file in the main window of Visual Studio
4) Press CNTRL+HOME
    a. This should move the cursor to the top of the text file (if it wasn’t already).
5) Toward the top of the text window, you should see a grey bar that is split into two parts. Click on the right part (which should have “DA1()” listed in it)
6) Select “DA1()” from the drop down menu
    a. This will move the cursor to the beginning of the SUBROUTINE DA1()

Re-build the Solution in a “Debug Configuration”

1. Select the appropriate compile settings
    a. In the “solution explorer”, highlight the purple icon “ANALYSIS”
    b. Under the view menu of VS, select “Property Pages”
    c. From the “Property Pages” window, press the “Configuration Manager” button
    d. Under “Active solution configuration:”, select “debug”
    e. Under “Active solution platform:”, selection “x64”

f. Press the “Close” button
g. Press the “OK” button
  - You can also do the above more easily by selecting the “Active solution configuration”, and “Active solution platform” from the drop-down windows of VS, which may be visible as toolbars in the VS environment.

2. Build the solution
   a. From the “Build” menu of VS, select “Clean Solution”
   b. From the “Build” menu of VS, select “Build Solution”
      - You should see the following output:

```
1>------ Rebuild All started: Project: ANALYSIS, Configuration: debug x64 ------
1>Deleting intermediate files and output files for project 'ANALYSIS', configuration 'debug|x64'.
1>Compiling with Intel(R) Visual Fortran Compiler XE 14.0.1.139 [Intel(R) 64]...
1>2_sorting.f90
1>1b_simdata.f90
1>1_GlobalParam.f90
1>UserAnalysisRoutines.f90
1>C:\MyFiles\MODEL_LAB\FORTRAN\ANALYSIS\UserAnalysisRoutines.f90(68): remark #7712: This variable has
not been used.   [II]
1>C:\MyFiles\MODEL_LAB\FORTRAN\ANALYSIS\UserAnalysisRoutines.f90(69): remark #7712: This variable has
not been used.   [TEST]
1>DeveloperRoutines.f90
1>0_entry.f90
1>Compiling manifest to resources...
1>Microsoft (R) Windows (R) Resource Compiler Version 6.1.7600.16385
1>Copyright (C) Microsoft Corporation.  All rights reserved.
1>Linking...
1>Creating library C:\MyFiles\MODEL_LAB\MODEL\ANALYSIS.lib and object
C:\MyFiles\MODEL_LAB\MODEL\ANALYSIS.exp
1>Embedding manifest...
1>
1>Build log written to  "file://C:\MyFiles\MODEL_LAB\FORTRAN\ANALYSIS\x64\debug\BuildLog.htm"
1>ANALYSIS - 0 error(s), 0 warning(s)
========== Rebuild All: 1 succeeded, 0 failed, 0 skipped ==========
```

   c. Check that the files have built to the correct location
      - Look in the “\model\” subdirectory, and check that 8 new files have been generated by the compiler, all of which start with “ANALYSIS.*”

Run a Test Analysis

The VS program environment provides sophisticated debugging tools to help you identify problems with your code. We will run through some of the basics here, which will help you get started.

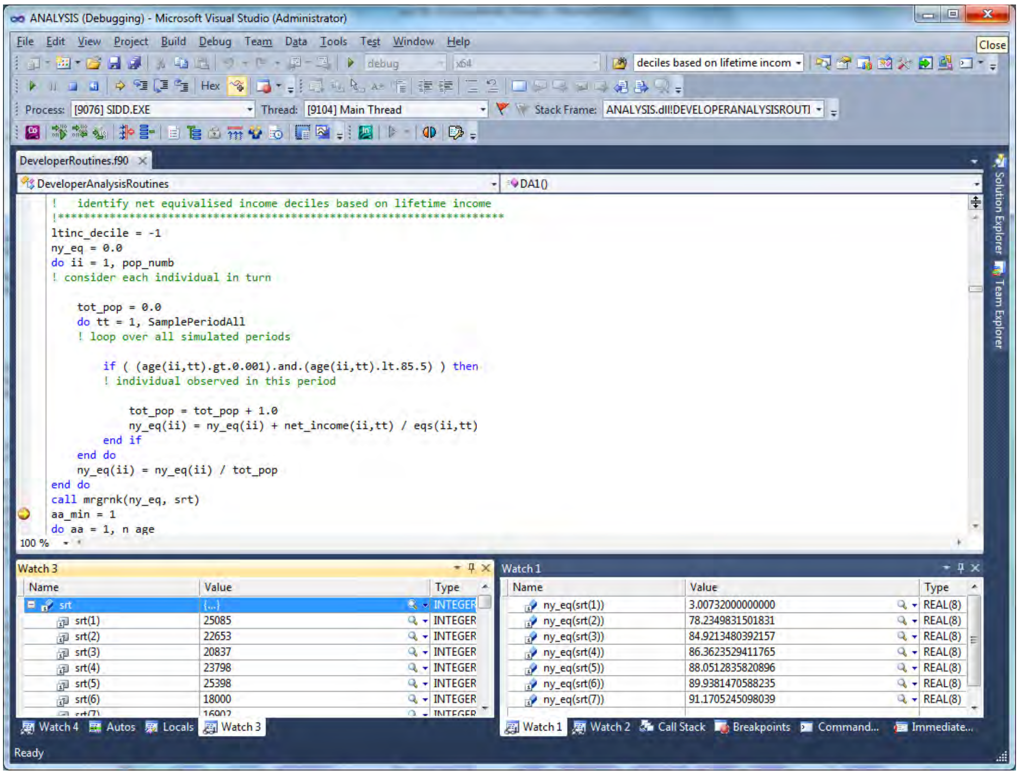
1. If you have not done so already, run the model through for the first time – see “creating a simulation base” under Section 2.
2. Set-up LINDA to make use of your new analysis routine
   a. Open “job file.xls”
   b. run macro “SIDD” (Alt F8, select SIDD, and run)
   c. Type the name of an existing simulation into the text box under the title “NAME OF EXISTING RUN TO ANALYSE”
   d. Press the “ANALYSIS OF EXISTING SIMULATION” button
   e. Type “DA1” into the text box alongside the title “Analysis routine 1”
   f. Press the “ENTER AND RUN” button
   g. Press the “EXIT” button
   h. Save “job file.xls”

2. Add a "break-point" to your subroutine. A break point is a marker, which tells Fortran to pause and wait for instructions if it is ever asked to execute a specific line of code. We will add a break-point where the analysis routine identifies population ranking based on lifetime equivalised income.
    a. Clear all break-points
        - Select the "Debug" menu from VS, and select "Delete all breakpoints"
    b. Search for (CNTRL+F): `deciles based on lifetime income`
    c. Put the cursor at the beginning of the line: `aa_min = 1`
    d. press F9 or press the left mouse button while holding the mouse pointer over the grey bar.
        - A red dot indicating the location of the break point should appear where the red break point should appear.
3. Launch the debugger
    a. Select the "Debug" menu from VS, and select "Start Debugging", or press "F5"
        - The model should start to run, and will pause for a while (while it loads the simulated population characteristics), before the code stops at the break-point that you have set (indicated by a yellow arrow).
4. Start analysing the code
    a. You should feel free to play around with the debug environment, which has a large number of very useful tools. Some of the most important of these include:
        - the "watch windows", accessed through the "Debug – Windows – Watch" menu options. These windows allow you to add in variables that you are interested in seeing the current value of. Adding the value "pop_numb" in one of these windows, for example, should indicate the population size.
        - walking through the code, achieved by pressing "F10"
        - skipping to the next line of code by pressing "F5"
        - you can also drag the yellow arrow and drop it at any point in the code that you like. The program will then proceed forward from that point.

A screenshot of what you might see is provided below. Here the debugger has been organised with two "Watch" windows side-by-side at the bottom. On the left, we have srt, which lists the index numbers of family units, ranked in ascending order by equivalised net lifetime income. On the right, we have the seven lowest values of ny_eq simulated by the model.

Re-build the Solution in a "Release Configuration"

1. Select the appropriate compile settings
    d. In the "solution explorer", highlight the purple icon "ANALYSIS"
    e. Under the view menu of VS, select "Property Pages"
    f. From the "Property Pages" window, press the "Configuration Manager" button
    g. Under "Active solution configuration:", select "release"
    h. Under "Active solution platform:", selection "x64"
    i. Press the "Close" button
    j. Press the "OK" button
        - You can also do the above more easily by selecting the "Active solution configuration", and "Active solution platform" from the drop-down windows of VS, which may be visible as toolbars in the VS environment.
2. Build the solution
    k. From the "Build" menu of VS, select "Clean Solution"
    l. From the "Build" menu of VS, select "Build Solution"



Introducing a new analysis routine

There are four key steps that you will need to take to run a new analysis routine for an existing simulation.

1) Create a new subroutine, with a name of your own choosing
    a. We have included an example "template_eg()" in file "UserAnalysisRoutines.f90" for you to work from
2) Include the name of your new subroutine in the list of the "UserEntry" subroutine
3) Save and "Build" your new analysis routine
    a. From the "Build" menu of VS, select "Build Solution"
4) Tell the model to run your new routine
    a. Open "job file.xls"
    b. run the macro "SIDD" (Alt F8, select "SIDD", and run)
    c. Type the name of your existing simulation into the text box under the title "NAME OF EXISTING RUN TO ANALYSE"
        i. if it is a new simulation, then work through the macro as usual, until you get to Form 2
    d. Press the "ANALYSIS OF EXISTING SIMULATION" button
    e. Type the name of your new analysis routine into the text box alongside the title "Analysis routine 1"
    f. Press the "ENTER AND RUN" button

List of global simulation variables

VARIABLE	DIMENSION	FILE (*.csv)	DEFINITION
pop_numb (N)	scalar	NA	number of families in simulated population cross-section
brth_ages (K)	scalar	NA	number of periods at which children can be born into households in simulation
brth_agesi	K	NA	vector that reports precise ages at which children can be born into households
min_age	scalar	NA	minimum age of the simulated sample
max_age	scalar	NA	maximum age of the simulated sample
SamplePeriodAll (T)	scalar	NA	number of periods that model projection covers
n_userto (U)	scalar	NA	number of user requested outputs from the tax and benefits routine
age	NxT	age	age of reference adult (years)
alc_dy	NxT	alc_dy	simulated alcohol duties (£ per week)
bcohort	N	bcohort	birth cohort index, higher number indicates younger cohort
benefit	NxT	ben	aggregate state welfare benefits received (£ per week)
cons	NxT	cons	discretionary consumption (£ per week)
costs	NxT	costs	non-discretionary housing and childcare costs (£ per week)
cp1	NxT	cp1	rights to first state contributory pension (BSP) (proportion of full pension, can be >1 for couples)
cp2	NxT	cp2	second state contributory pension (S2P) (£ per week)
death_age	N	death_age	age at which reference adult dies (years)
default_pp	NxT	dppart	FLAG = 0/1 if default is to opt-out / participate in pensions
education	N	education	FLAG = 0/1 for non-graduates / graduates NOTE: this flag indicates highest qualification achieved during simulated lifetime; students at entry to sample are not identified as graduates until they complete their studies
emp1/2	NxT	emp1/2	employment status of reference adult / spouse NOTE: emp = 0 if not employed, and higher numbers indicate higher labour supply.
eqs	NxT	eqs	equivalence scale of reference adult's household
ful_dy	NxT	ful_dy	fuel duty paid (£ per week)
hum_cap	NxT	hum_cap	household human capital (wage potential) (£ per week)
isa	NxT	isa	savings held in an Individual Savings Account (£)
isr_dy	NxT	isr_dy	insurance standard rate duty paid (£ per week)
ihr_dy	NxT	ihr_dy	insurance higher rate duty paid (£ per week)
isa_cont	NxT	isa_cont	net contributions to individual savings account (£ per week)
lab_inc	NxT	labinc	labour income of household (£ per week)
leis	NxT	leis	proportion of time spent in leisure (proportion of time)
mdr	NxT	mdr	marginal deduction (tax) rate (fraction)
na	NxT	na	number of adults in household (1=singles 2=couples)
net_income	NxT	net	household net income (£ per week)
nic	NxT	nic	National Insurance Contributions of household (£ per week)
nk	NxT	nk	aggregate number of dependent children

nk_all	NxTxK	nk_all	number of children, by childbirth year NOTE: number of children from child birth age 2, at time t0 of household ii = nk_all(ii,t0,2)
obw	NxT	obw	own business wealth (£)
OP_cont	NxT	OPcont	contributions to personal pension 1 (£ per week)
OPpen	NxT	OPpenb	aggregate accrued wealth in personal pension 1 pre pension take-up, personal pension income post pension take-up (£ / £ per week)
parttr1/2	NxT	parttr1/2	participation tax rates of reference person / spouse
pen_inc	NxT	pen_inc	pension income received (£ per week)
pcr	NxT	PPcr	personal pension to which individual is eligible
PP_cont	NxT	PPcont	aggregate contributions to personal pension 2 (£ per week)
PPpen	NxT	PPpen	aggregate accrued rights to private pension, defined as an annuity stream (£ per week)
PPpenb	NxT	PPpenb	aggregate accrued rights to private pension, defined as a wealth equivalent (£)
prec	N	prec	age at which pension taken up (years)
prett_inc	NxT	prett	pre-tax and benefit household income (£ per week)
priv_penc	NxT	ppc	private pension contributions (£ per week)
ri	NxT	ri	proportion of wealth invested in risky assets
semp	NxT	semp	self-employment flag (1= self-employed if supply labour, 0=employee if supply labour)
sim_weight	NxT	sim_weight	household weighting variable (index based on weighting variable reported in WAS)
student	NxT	student	student status (0 = non-student, 1 = student graduating at age 23, 2 = student graduating at age 27)
tax	Nx7xT	tax	household tax burden by tax band (£ per week)
taxAgg	NxT	taxagg	aggregate household tax burden (£ per week)
tbc_dy	NxT	tbc_dy	tobacco duties paid (£ per week)
ret	NxT	ret	retirement status (0=not yet defined as retired, 1=retired)
user_to	NxUxT	user_taxoutput	user defined output from tax routines (user defined)
Val	NxT	Val	measure of expected lifetime utility (utility index)
vat_sr	NxT	vat_sr	standard rate VAT paid (£ per week)
vat_rr	NxT	vat_rr	reduced rate VAT paid (£ per week)
w	NxT	w	liquid wealth of household (£)
w2	NxT	w2	aggregate household net worth, including value of pension rights (£)
wage_offer	NxT	wage_offer	flag = 1 if reference adult receives wage offer, 0 otherwise
wage_offer2	NxT	wage_offer2	flag = 1 if spouse receives wage offer, 0 otherwise

Organisation of the Tax and Benefits Code

A stylised schematic of how the tax and benefit calculations are organised is provided in Figure 3.

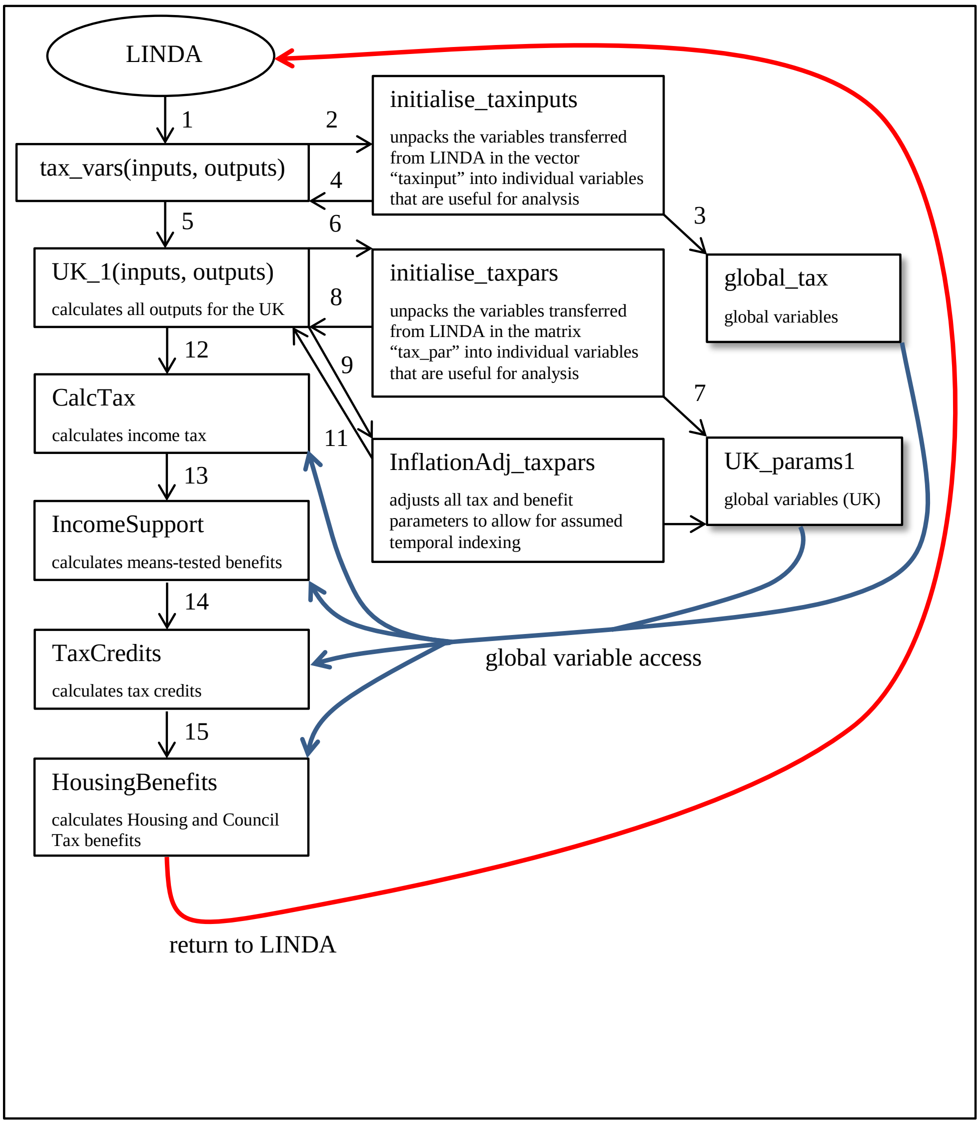


FIGURE 1: Stylised Schematic of the Calculation of Taxes and Benefits in LINDA

Adjusting number of tax outputs

It is possible to alter the tax code of the model, so that it will generate additional summary statistics to those that are produced by default. This involves the following steps:

1) Define the number of additional statistics in the model parameters.
    a. Add the number of additional tax statistics that you would like to generate with the model to the number currently defined in cell AO14 of sheets "input" and "inputA" in "job file.xls".

        i. E.g. if the pre-existing value in cell AO14 is 17, and you would like to add an additional three outputs, then alter this cell to 20.

    b. Save “job file.xls”.

2) Alter the Fortran code in the solution “TAXES.sln” to generate the statistics of interest

    a. Note that these statistics need to be included as output variables in the subroutines that calculate taxes and benefits.

    e.g. you might alter the subroutine:

```
call UK_1(age, time, n_taxp4, tax_par4, ndim_tax, tax_par, yy1,
Rpstt, ptbmr, mcost_cpc1, mcost_cpc2, costs, benefit_s,
tax_s(1:8))
```

    so that it includes an additional output xx, by including xx at the end of the variable list as in:

```
call UK_1(age, time, n_taxp4, tax_par4, ndim_tax, tax_par, yy1,
Rpstt, ptbmr, mcost_cpc1, mcost_cpc2, costs, benefit_s,
tax_s(1:8), xx)
```

3) Alter the output vector to return the statistics to the model.

    a. The additional statistics need to be sent to the subroutine pack_taxoutputs, and then added to the end of the output vector as directed in that subroutine.

    e.g. in the above example, we would alter the code as indicated here:

```
call pack_taxoutputs(taxoutput, n_taxoutput, yy, Rpstt, ptbmr,
m_pens_cost, costs, prett_inc, benefit_s, tax_s, xx)
```

    and then include xx in taxoutput, just after the line:

```
! user output add to the bottom here
```

4) The model will then include the associated statistics in the panel data generated for each simulated population, under the name user_toX, where X = 1, 2, 3, etc, for the first second and third output that has been added to the list of statistics reported by the tax function.

6. Best Practice Methods of Use

This section provides a brief step-by-step guide concerning how the model should be used to explore the effects of policy alternatives. The guide has been written so that it will be applicable to a wide range of alternative subjects of interest, from studies concerned solely with the distributional implications of policy, to those that focus on behavioural responses to policy counterfactuals. Each step of a stylised analytical problem is described under separate section heading, in approximate chronological order.

TIP: it is incredibly important that you allow ample time to conduct your analysis.

Step 1: specify policy parameters

There are two alternative ways to alter the policy environment in the model. The first is to alter the parameters of existing model structure. New users of the model are encouraged to select policy parameters using the Excel front-end that is supplied with the file "job file.xls", as described in Section 3. Advanced users may find it easier to alter model parameters directly through the cells on the "input" sheet of "job file.xls". The second is to alter the tax code directly; see Section 5 for details.

Having adjusted the model parameters, it is often advisable to check that the revised structure conforms to that which is desired. A number of tools are provided with the model to facilitate such checks.

Checking differences with base simulation

The "check" tab in "job file.xls" allows you to identify which parameters in the "input" sheet are different to those of the base simulation (stored in the "inputA" sheet). You can run this check after working through the front-end forms in "job file.xls" in one of two ways. If you press the "LAUNCH MODEL" button on Form Z, then you can look at the "check" tab after the fortran program has started. Alternatively, you can choose the "EXIT" button on Form Z, saving "job file.xls", and then look at the "check" tab. In this second case, after you have confirmed that you are happy with the differences described between the prevailing model parameters (sheet "input") and the model base (sheet "inputA"), you can launch the Fortran code manually by double-clicking on the program SIDD.exe.

Validating the tax function

As noted in Section 3 (Form D4), there are two routines that are provided with the model that produce trial statistics for the simulated tax function. The terms of both test routines are defined by Excel files stored in the "\base_files\analysis_files\" subdirectory. "tax_test.xls" allows you to generate test statistics for a range of alternative pre-tax incomes, for a given family type. "tax_test2.xls" is similar to "tax_test.xls", but allows you to vary both pre-tax income and family demographic characteristics.

These test routines can be run at any time after you have started running a given simulation.

1) Open the relevant excel file ("tax_test.xls" or "tax_test2.xls").
2) Define the alternative test cases that you would like to consider
3) Save the excel file.
4) Open "job file.xls"

5) Run the SIDD macro
6) Enter the relevant simulation name in the text box adjacent to the button "ANALYSE TAX FUNCTION"
7) Press the "ANALYSE TAX FUNCTION" button.
8) Select the relevant analysis routine using the radial button.
9) Press the "ENTER and RUN" button.

A new simulation window will open, and will indicate when the analysis routine is complete. You can then re-open the relevant excel file ("tax_test.xls" or "tax_test2.xls") to check the results obtained.

Step 2: run model and generate output

As noted in Section 4, two types of output can be generated by the model. The model will solve simulated panel data for each family unit, in the form of *.csv files (that can be opened by most statistical packages). In this case, each variable is saved in a separate file, each row represents a different family unit, and each column a different time period.

The model will generate a series of summary statistics for each simulation by default, which are reported in Excel files. The model can also be directed to generate a range of additional summary statistics; see Section 4 and "Form 2" in Section 3 for further details.

Step 3: think about likely incentive effects

An important feature of the model is that savings and labour supply decisions are endogenous. This means that these decisions react to the incentives that are embodied by the assumed policy environment. Explaining the results that are reported by the model consequently depends in part on understanding the drivers underlying the simulated behaviour. It is useful to think about what behaviour you expect to see before the model completes its analysis, as this will help you to identify where to begin looking once the simulated results are in. It is easiest to think about incentives relative to the base simulation, as the model can be asked to generate comparisons with the base simulation by default.

Step 4: analyse model output

This stage of the analysis involves comparing the model output against the expectations that you formed in Stage 3 (described above). It is not uncommon to find that behaviour deviates from what you had initially expected, as the incentives embodied by policy can often be quite difficult to infer.

High level analysis

The model produces a large number of high-level summary statistics by default in the file DA1.xls. It can also be directed to report a series of age specific statistics for population cross-sections and individual birth cohorts (see Section 4). Through repeated use, you are likely to settle upon some subset of these results as an appropriate place to start your analysis.

Detailed statistical analysis

The model can be used to generate decile level statistics, and to produce associated comparisons with the base population. These statistics will often be sufficient to identify the key margins of behavioural variation simulated under a given policy environment, and allow you to formulate a "story" underlying the results obtained.

What to do when results do not conform to expectations

If the above statistics do not enable you to devise a compelling account of the results generated by the model, then it is often useful to focus upon the micro-data generated by for individual family units. Suppose, for example, that you cannot understand why labour supply falls under the policy environment of interest, relative to the base simulation. In this case, you could look for units that are simulated to reduce their labour supply substantively under the new policy environment. Bearing in mind that the model is designed so that the only differences between alternative simulations are due to the considered policy environments, this approach will hopefully provide clues as to the incentives that underlie the unanticipated behavioural responses. As a last resort, staff and the NIESR will attempt to provide technical support (subject to availability).

Step 5: package output

Having identified a behavioural "story" underlying simulated results, the packaging exercise is usually fairly straight-forward. As this exercise will vary, depending upon the subject of concern, we do not discuss it further here.

Appendix A: The Utility Function

This has two components to it. Within-period utility u is a function of total household consumption $c_{i,t}$ adjusted for effective household size $\theta_{i,t}$ and leisure time represented by $l_{i,t}$.

α represents the consumption-equivalent of leisure and ε the elasticity of substitution between consumption and leisure.

$$u\left(\frac{c_{i,j}}{\theta_{i,j}}, l_{i,t}\right) = \left(\left(\frac{c_{i,j}}{\theta_{i,j}}\right)^{(1-1/\varepsilon)} + \alpha^{1/\varepsilon} l_{i,t}^{(1-1/\varepsilon)}\right)^{\frac{1}{1-1/\varepsilon}} \quad \textbf{(1)}$$

Within-period utility enters into an intertemporal utility function in the manner represented below. Intertemporal discounting takes a quasi-hyperbolic form, where δ is the long-run discount factor, and β is the excess short-run discount factor. When $\beta = 1$, preferences are time consistent, which implies that – for any given set of circumstances – the same decisions will maximise expected lifetime utility, regardless of when the decisions are made. That is, if an individual could commit to savings and employment decisions that take their evolving circumstances into account for any future age, then they will make the same decisions regardless of their current age. With $0 < \beta < 1$, intertemporal preferences exhibit myopia, which means that people would like to be more patient in the future than will actually be the case. The model assumes that people are 'sophisticatedly' myopic, in the sense that they are aware of their own self-control problems and react to them. This can result, for example, in a preference to lock savings away in a pension rather than a bank account, to avoid the temptation of spending the savings prematurely.

γ is relative risk aversion, and $\phi_{j-t,t}$ is the probability of surviving j years, given survival to age t. ζ_a and ζ_b represent the warm glow utility derived from leaving a positive bequest $w^{+}_{i,t+1}$.

$$U_{i,t} = \left[u\left(\frac{c_{i,t}}{\theta_{i,t}}, l_{i,t}\right)^{1-\gamma} + \beta E_t \left\{ \sum_{j=t+1}^{T} \delta^{j-t} \left(\phi_{j-t,t} u\left(\frac{c_{i,j}}{\theta_{i,j}}, l_{i,j}\right)^{1-\gamma} + (1-\phi_{j-t,t})(\zeta_a + \zeta_b w^{+}_{i,t+1})^{1-1/\gamma} \right) \right\} \right]^{\frac{1}{1-\gamma}} \quad \textbf{(2)}$$