

Building and Solving Macroeconomic Models using WinSolve: Introduction to WinSolve*

Richard G. Pierse
University of Surrey, Guildford GU2 7XH, U.K.

April 20006

1 Introduction

This paper illustrates the building of a small model using *WinSolve*. Section 2 has a brief general description of the WinSolve model definition language and the data file formats supported by the program.

2 Getting Started with WinSolve

WinSolve (Pierse 2000) is a computer package for solving and simulating (nonlinear) models. It is a 32 bit Windows program that runs under all Windows platforms. To start the program, click on the *WinSolve* icon on your computer desktop. A banner screen is displayed briefly. Then the main window opens (Figure 1). This is the *WinSolve* desktop in which you work. At the top of the screen is the menu bar from which commands can be selected, either by clicking with the mouse on the appropriate menu name, or by pressing the key of the first letter of the menu name *while holding down the the Alt key*. Below the menu bar is the tool bar. This contains icons that are shortcuts to some of the more popular commands on the menu bar; double clicking on the icon is equivalent to selecting the corresponding command from the menu bar. When an icon is greyed, then that command cannot be selected. On first starting *WinSolve*, only two icons are ungreyed.

*This paper was prepared for a HKIMR Workshop held at HKMA, Hong Kong, April 11-12 2006.

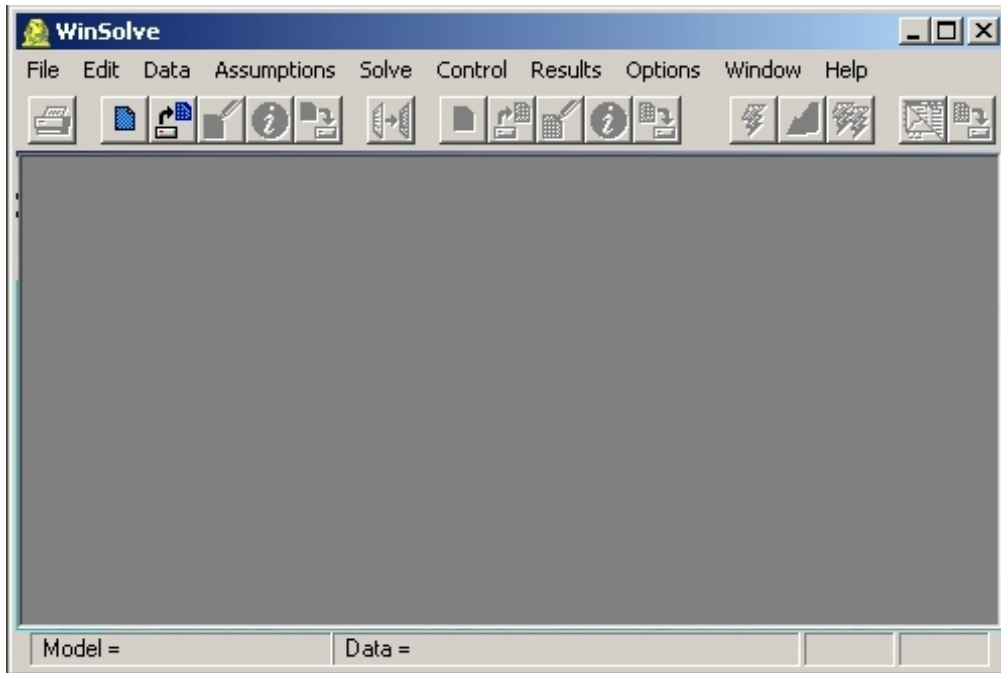




Figure 1: WinSolve main window

These are the icons to create a new model  and to open an existing model . At the bottom of the screen is the status bar that displays information about the current state of the program. The first item lists the currently selected model and the second, a description of the data set associated with that model. When the program is first started, no model has been opened and so both boxes are empty.

2.1 The Model Definition Language

Model equations in *WinSolve* are defined in a simple algebraic language. Equations are made up of a combination of variable names and numerical constants, the five arithmetic operators $+$ $-$ $*$ $/$ and $^$, and function references. The order of evaluation can be altered by the use of parentheses $()$. Equations can run over more than one line and each equation is terminated by the semicolon character $;$. An equation may be preceded by optional codes (sequences of characters starting with an asterisk) specifying information to *WinSolve*.

Equations can be interspersed with comments or other information to be ignored by the *WinSolve* compiler. Two kinds of comment are possible. The

single quote character ' or the 'at' character @ indicate that the rest of the current line is a comment. Alternatively, any section of text can be marked as a comment by delimiting it with a pair of brace characters '{' and '}'. This latter form of comment can be nested and can appear anywhere in the model code.

2.1.1 An example

The algebraic notation for defining equations is best explained by means of a simple example:

```
@ Equation 1: Consumption function
*M log(C) = 100.5 + 0.15 * log(C(-1))
{ This is a comment which is ignored }
+ 0.85 * log(Y) ;
@ Equation 2: Income identity
Y = C + Z ;
```

There are two equations in the example. In the first, the logarithm of variable C is defined to be a linear function of the logarithm of variable Y and the logarithm of C lagged one period. This equation is split over two lines with a comment in between. In the second equation, variable Y is defined as the sum of C and variable Z .

The notation $C(-1)$ denotes that the variable C is lagged by one period. By analogy, $Y(+3)$ or equivalently, $Y(3)$ would denote a lead of three periods on the variable Y .

\log represents the logarithmic function. This is one of many mathematical functions available in *WinSolve*. Several of these have synonyms for the convenience of those used to other computer languages. For a complete list see Table 1.

2.2 Invertible Functions

Note that in the first equation in the example, a function appears on the left-hand side of the equation. *WinSolve* automatically renormalises this equation in terms of the level of the variable. The functions that can legally appear on the left side of an equation are termed invertible and are marked in the function list in Table 1. The argument of such a function must be a single current-dated variable. For example, the expressions $\log(C/Y)$ or $\log(C(-1))$ would both be illegal on the left-hand side of an equation.

Table 1: WinSolve functions

Mathematical and statistical functions	
$abs(x)$	absolute value of x
$bound(x)$	x bounded to the unit interval $=min(max(x,0),1)$
$cdfn(x)$	cumulative normal density from $-\infty$ to x
$cos(x)$	# cosine of x (x in radians)
$exp(x)$	# exponential operator (e to the power of x)
$int(x)$	integer part of x (rounded towards $-\infty$)
$learn(\dots)$	Kalman learning function
$log(x)$	# natural logarithm of x ; synonymns: $ln(x)$
$max(x1, \dots, xn)$	maximum of n arguments: $x1, \dots, xn$
$min(x1, \dots, xn)$	minimum of n arguments: $x1, \dots, xn$
$norm(x)$	normal random number with variance x
$parexp(\dots)$	Parameterised expectations function
$sin(x)$	# sine of x (x in radians)
$sqrt(x)$	# square root of x ($x \geq 0$)
$tan(x)$	# tangent of x (x in radians)
$unif(x)$	uniform random number on interval $(0, x)$
Logical functions	
$ifeqz(x)$	= 1 if $x = 0$; = 0 otherwise
$ifgtz(x)$	= 1 if $x > 0$; = 0 otherwise
$ifltz(x)$	= 1 if $x < 0$; = 0 otherwise
Date & time functions: (<i>argument d must be a valid date</i>)	
$ifeq(d)$	= 1 in period d , = 0 otherwise
$iflt(d)$	= 1 in periods $< d$, = 0 otherwise
$ifgt(d)$	= 1 in periods $> d$, = 0 otherwise
$ifne(d)$	= 1 in periods $\neq d$, = 0 otherwise
$ifle(d)$	= 1 in periods $\leq d$, = 0 otherwise
$ifge(d)$	= 1 in periods $\geq d$, = 0 otherwise
$seas(i)$	= 1 in the i th season of the year; = 0 otherwise
$time(d)$	time trend taking the value one in period d
Special functions: <i>argument must be variable (with optional lag)</i>	
$adjust(v)$	Adjustment associated with variable v
$base(v)$	Data base value of variable v
$diff(v)$	# First difference of variable v
$diff4(v)$	# Fourth difference of variable v
$distlag(v,n,x1, \dots, xn)$	Distributed lag of v : $= x1*v + \dots + xn*v(-n+1)$
$distlog(v,n,x1, \dots, xn)$	Distributed lag of $log(v)$
$dlog(v)$	# First difference of log of v
$d4log(v)$	# Fourth difference of log of v
$ratio(v)$	# Ratio of variable v to its first difference
$ratio4(v)$	# Ratio of variable v to its fourth difference
# denotes function is invertible	4

Table 2: Equation codes

<i>A</i>	equation has additive adjustments
<i>C</i>	line is not an equation but an (optional) equation description
<i>M</i>	equation has multiplicative adjustments
<i>P</i>	equation is a parameter definition
<i>T</i>	equation is a terminal condition

2.2.1 Variable names

Variable names in *WinSolve* can be up to 16 characters long. Legal characters within names are the upper and lower case letters $A - Z$ and $a - z$, the numbers $0 - 9$, and the five special characters $_ \backslash \$ \% \#$. By default, names are not case sensitive so that ABC and aBc represent the same variable, although it is possible to override this default setting. Names of *WinSolve* functions cannot be used as variable names.

2.2.2 Equation codes

A set of characters preceded by an asterisk appearing immediately before the start of an equation is treated by *WinSolve* as a code. Codes are used to pass information to the program. The code M in the first equation in the example tells the program that the equation for C has multiplicative adjustments. Allowable codes are given in Table 2.

2.2.3 Switching equations and dummies

The date and time functions provide a simple way to code up the dummy variables that typically appear in model equations. For example a variable taking the value plus one in 2004:1 and minus one in 2004:2 could be coded by the expression:

$$ifeq(200401) - ifeq(200402)$$

For a more complicated example, the expression:

$$5 + ifge(200501) * ifle(200504) * time(200501) + ifgt(200504) * 5$$

represents a dummy taking the value 5 before 2005:1 then rising by increments of one to the value 10 in 2006:1 and thereafter.

The date functions also make it possible to code equations that switch at different points in time. For example, an equation for variable Y , taking the form

$$\log(Y) = 100 + 0.5 * \log(Z)$$

up until the period 2005:3, and thereafter taking the form

$$\log(Y) = 150 + 0.4 * \log(W)$$

can be written in *WinSolve* as

$$\begin{aligned} \log(Y) = & 100 + \text{ifgt}(200503) * 50 + \text{ifle}(200503) * 0.5 * \log(Z) \\ & + \text{ifgt}(200503) * 0.4 * \log(W) ; \end{aligned}$$

2.2.4 Using Parameters

As well as hard-coding equation coefficients into equations, as illustrated in the examples above, it is also possible to define names for coefficients in *parameter* statements (prefixed by the **P* code) and then use these parameter names in subsequent equations. For example, the consumption function used above could also be rewritten as:

$$\begin{aligned} *P \quad c1 &= 100.5; \\ *P \quad c2 &= 0.15; \\ *M \quad \log(C) &= c1 + c2 * \log(C(-1)) + (1-c2) * \log(Y) ; \end{aligned}$$

The parameter definition takes the form **P name=value*; where *name* is the parameter name (which must be unique and cannot be the same as the name of a variable within the model) and *value* is a numerical value. The parameter definition must appear in the model code before the parameter is actually used in an equation. To define a parameter as a function of other parameters, use the **W* definition as in the following code fragment:

$$\begin{aligned} *P \quad c2 &= 0.15; \\ *W \quad c3 &= 1-c2; \\ *M \quad \log(C) &= c1 + c2 * \log(C(-1)) + c3 * \log(Y) ; \end{aligned}$$

The **W* command defines a temporary name that is replaced by its definition, wherever it appears within the equation code. The right-hand side of a **W* definition can include parameter names, variable names, constant and *WinSolve* functions.

2.2.5 Alternative equations

A model may include more than one equation for an endogenous variable. In this case, *WinSolve* treats these as alternative equations and, by default, will execute the first such equation in each case and ignore the others. However, it is possible to select any one of the alternative equations through the *Switch alternate equations* option on the *Assumptions* menu. In this way, one of several variants of a model can be chosen without having to edit the equations or recompile the model. It is advisable to prefix each alternative equation definition by an equation description code. This description is then used to identify the equation variant within *WinSolve*.

The following example illustrates the use of alternative equations for the exchange rate xr . Note the use of description code lines before each equation. The default is the fixed real exchange rate equation which says that the exchange rate grows as the ratio of growth in domestic prices p to growth in foreign prices pf .

**C Fixed real exchange rate*

$$ratio(xr) = ratio(p) / ratio(pf) ;$$

**C Backward looking UIP*

$$dlog(xr) = 0.25 * log((1 + r(-1)) / (1 + rf(-1))) ;$$

**C Forward looking UIP*

$$log(xr) = log(xr(+1)) + 0.25 * log((1 + r) / (1 + rf)) ;$$

**C Fixed nominal exchange rate*

$$xr = xr(-1) ;$$

2.2.6 Terminal condition equations

WinSolve can be used to solve models with model consistent expectations. Such models require terminal conditions on those variables that appear in the model with expectational leads. The terminal conditions define the value of these variables for periods beyond the final solution period. Five standard types of terminal condition are available in *WinSolve*. These are: constant level, exogenous value, constant period-on-period growth rate, constant annualised growth rate, and constant long run growth rate.

Additional terminal conditions can be defined by users in equations as part of the model code. These special equations are flagged as terminal conditions by using the **T* equation code. If more than one terminal condition is defined for any variable, then these are treated as alternative terminal conditions, and by default the first is used. As with alternative equations, the

user should provide a description code line before each terminal condition in order to identify it.

The following example shows a possible user-defined terminal condition for a forward looking exchange rate equation.



```
*C Equilibrium net asset ratio condition
*T       $diff(xr) = diff(nar) - diff(narf) ;$ 
```

2.3 Model Equation Files

Model equations are maintained in standard text files (with a .txt suffix). The model object contains an *OLE* link to the model equation file so that equations can be edited from within *WinSolve* by double clicking on the model icon in the model object window or by selecting the *Edit model* option on the *File* menu. When this command is selected, *WinSolve* will open the editor associated by Windows with text files.

2.4 Data Files

In addition to a set of equations, a model normally needs to have an associated data set. This must include initial conditions on all the endogenous variables in the model that appear with lags, as well as observations over the full solution period for all exogenous variables. Endogenous variables that are defined before they appear on the right-hand side of any equation, and which never appear with a lead or lag, need not have any observations in the data set. If not, then they are termed working variables.

There are two ways to read data into *WinSolve*: either a new empty data set can be created, or an existing data set can be opened and read. Clicking the new data button  , or selecting the *Create new data file* command from the *Data* menu opens a dialog box allowing the user to create a new data set for all the variables of a model over a specified period, setting all observations to a fixed value. The observations can subsequently be changed by selecting the *Edit Data/Adjustments* option on the *Data* menu or clicking on the  button.


Alternatively, clicking the  button or selecting the *Open data file* command from the *Data* menu, allows a data set to be read from file in any of the supported formats defined in Table 3. The name of the data file should have the appropriate suffix to ensure that *WinSolve* recognises the correct format. Data sets can subsequently be saved in the same or a different format.

Table 3: WinSolve supported data formats

SDF	Default binary format used by <i>WinSolve</i>
BN7, IN7	PcGive, PcFiml, Stamp 5.0 binary format
CSV	Comma delimited format
FIT	MicroFit binary format
RAT	RATS binary data format
WF1	EViews workfile format
WK1	Lotus 123 etc.binary spreadsheet format
XLS	Excel spreadsheet formats
BNK	DataView and Modler binary format (read only)
DAB	Warwick MacroModelling Bureau binary format
DEM	NIESR binary format
FRM	TROLL portable data format
NIM	NIMODEL program ascii format
RUN	LBS binary data bank format

Note that several of the data formats in Table 3 have restrictions on the maximum number of variables, the length of variable names and the characters allowable within names, that are more restrictive than those required by *WinSolve*. In order to ensure that data files saved by *WinSolve* in these formats will be readable by other programs, the user should make sure that variable names conform to the rules of the chosen data format.

2.4.1 Date Notation in WinSolve

Observations may be annual, quarterly, monthly, or undated. The standard notation for dates within *WinSolve* is a four figure integer for annual or undated observations and, for other frequencies, a six figure integer comprising a four figure year followed by a 2 figure within-year period. For example, the date 199602 represents the second quarter of 1996 if data is quarterly or February 1996 if data is monthly.

3 Tutorial on Model Building

In this tutorial we look at a very simple linear rational expectations model derived from Muth (1961) and analysed in Chapter 4 of Fisher (1992). This may be defined by the equation

$$p_t = \alpha p_{t+1}^e + x_t$$

where p_{t+1}^e is the expected value of p_{t+1} formed in period t . On the assumption of model consistent expectations,

$$p_{t+1}^e = p_{t+1} .$$

x_t is an exogenous forcing variable which can be defined by the autoregressive process

$$x_t = \mu + \rho x_{t-1} + \varepsilon_t$$

where ε_t is a white noise stochastic shock.

Neglecting the stochastic shock, it can be shown that this model has an analytic solution given by

$$x_t = \rho^t x_0 + \frac{\mu}{1 - \rho} (1 - \rho^t)$$

and

$$p_t = \frac{\rho^t}{(1 - \alpha\rho)} \left(x_0 - \frac{\mu}{1 - \rho} \right) + \frac{\mu}{(1 - \rho)(1 - \alpha)}$$

so that, as long as $|\rho| < 1$, asymptotically, $\rho^t = 0$ and so

$$x_t = \frac{\mu}{1 - \rho} \quad \text{and} \quad p_t = \frac{\mu}{(1 - \rho)(1 - \alpha)} .$$

Choosing parameter values $\mu = 5$, $\rho = 0.5$ and $\alpha = 0.8$, asymptotically we have $x_t = 10$ and $p_t = 50$.


This model may be defined in *WinSolve* by the equations

$$\begin{aligned} x &= 5 + 0.5*x(-1) + norm(0.1) ; \\ p &= 0.8*p(1) + x ; \end{aligned}$$

Note that the first equation here includes the stochastic shock ε_t , represented by the *WinSolve* function *norm(0.1)* which generates a normally distributed pseudo-random variable with variance of *0.1*.

3.1 Creating a new model object

The first step is to create a new model object in which we will define the equations of the simple model considered above. If *WinSolve* is not already running then you need to click on the *WinSolve* icon on your Windows desktop to start the program.

Click on the new model icon  or select the *New Model* option from the *File* menu. A new window will appear on the desktop representing the model object (Figure 2). The icon in this window is a link to the model equations and double clicking the icon will open an editor to view the equations. The file containing the equations is given the default name of *Model1.txt*. You will be able to change this later.

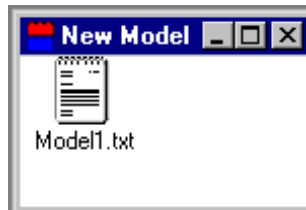


Figure 2: The new model object

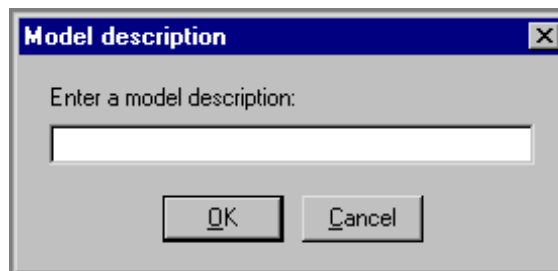


Figure 3: Model description box

3.2 Entering the model equations

Double click on the *Model1.txt* icon in the model object window. This will open the editor you have set up to view files with suffix TXT. The equation file should contain a single comment line:

@WinSolve code

which is there purely as a marker and may be deleted.

Now you may enter the equations of your model. Type in the lines:

$$\begin{aligned}x &= 5 + 0.5*x(-1) + norm(0.1) ; \\ p &= 0.8*p(1) + x ;\end{aligned}$$

Then save the file and close the editor to return to *WinSolve*. *WinSolve* will know that the equations of your model have been changed and will start to compile them and check whether there are any errors.

If not, then another box will be displayed, Figure 3, which allows you to enter a description to give to your model. You can type anything here but a concise description such as *Linear RE Model* is best. Then click the *OK* button to exit.

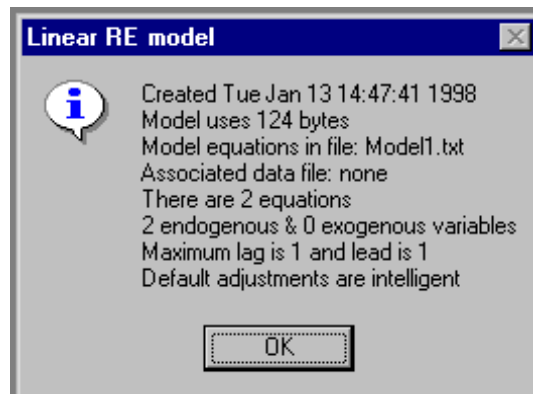



Figure 4: Model information box

You now have a model. You can find out some of its properties by clicking on the  icon or selecting the *Model info* option from the *File* menu. This opens an information box as in Figure 4. You will see that the model contains 2 equations, with a maximum lead on any equation of 1 and a maximum lag of 1. It uses 124 bytes of storage. Click the *OK* button to exit.

3.3 Correcting mistakes

Suppose that you made a typing mistake in defining the model and entered the function name *norn* instead of *norm* in the first equation. Then when *WinSolve* compiled your model it would have found an error and displayed the status box in Figure 5. Clicking the *OK* button reveals the error box Figure 6 which displays the lines where *WinSolve* has detected errors. In this case, since *norm* is not a function name, *WinSolve* assumes that it is a variable and treats the argument 0.1 as a lead on that variable. However, leads and lags must take integer values and so the decimal point generates a syntax error of an unexpected character. To correct the error, close the error box and then double click on the model icon to re-enter the editor and fix the mistake.

Note that if the argument to *norm* had been an integer value, then *WinSolve* would not have detected any error but would have assumed that *norm* was an exogenous variable in the model. *WinSolve* is unable to detect all model mistakes and you should check carefully the list of model variables to check that there are no unexpected variables.

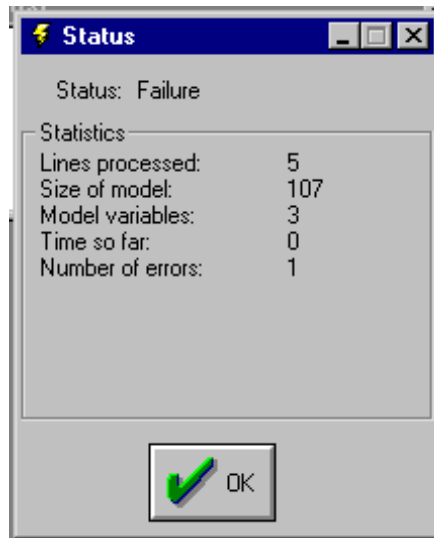


Figure 5: Status box

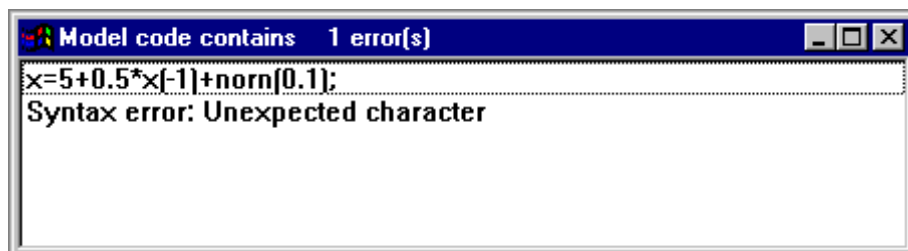


Figure 6: Error box

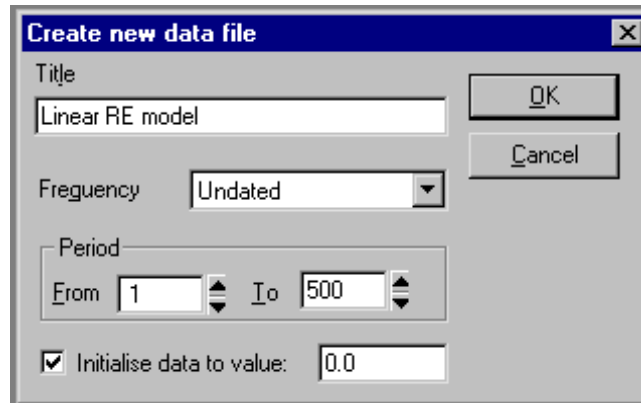


Figure 7: Create a data file dialog box

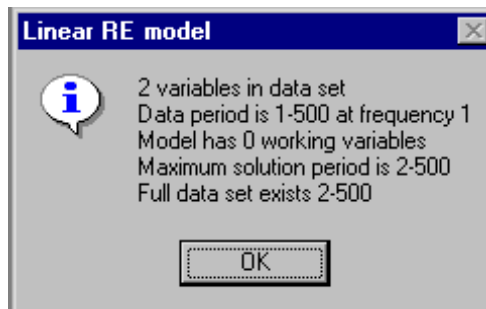



Figure 8: Data information box

3.4 Adding Data

In order to solve the model, a data set needs to be defined. This sets up the maximum solution period and gives an initial condition for x_0 . In addition an exogenous terminal value for p_{T+1} can be defined.


Click on the new data button , or select the *Create new data file* command from the *Data* menu. This opens the dialog box in Figure 7. Select *Undated* from the frequency box and choose the periods to run from 1 to 500. This generates a data set of 500 undated observations, as the maximum solution period for the model. Check the initialise data box and set the value to zero. This initialises observations over the full solution period. Strictly, this is not necessary but it is convenient. Finally, click *OK* to exit the dialog box. A box appears with information about the data set, Figure 8. Click *OK* to close it. You now have a data set and are now almost in a position to solve the model.


3.5 Choosing terminal conditions

Before commencing model solution, some consideration should be given to the matter of terminal conditions. In order to solve a model including leads of the variables, some condition is needed to tie down the value of these variables for periods beyond the final solution period. *WinSolve* provides five standard types of terminal condition: constant level, exogenous value, constant period-on-period growth rate, constant annualised growth rate, and constant long run growth rate. In addition, users may define their own terminal conditions as special types of model equation. Choosing a sensible terminal condition is important since imposing an inappropriate condition may make it impossible for *WinSolve* to find a model solution.

The default terminal condition is constant level which implies that values of the variable beyond the terminal date T are equal to the solution value in period T . In the model under consideration, we know that p asymptotically attains a steady state. In this case, a constant level terminal condition is sensible so that we do not need to override the default value. However, in other models it may be necessary to change the terminal conditions for one or more equations. This can be done through the *Change terminal conditions* assumption on the *Assumptions* menu.

3.6 Solving the model

Click on the Solve model icon  or select the *Solve model* option from the *Solve* menu. Select the *Dynamic model solution* option from the Solution mode listbox and click *OK*. This initiates solution. The Solution status box displays information as the model solves. When model solution is complete, the status box can be closed by clicking on the *OK* button.

The results can now be viewed in the *Results* menu. Click on the graph icon  or select the *New table/graph* option in the *Results* menu. Choose the variable p and the variable type *Simulation values*. Click the *Add* button to add this item to the graph and click *OK* to finish. The solution values will be graphed (Figure 9). It can be seen that p quickly tends to its asymptotic value of 50 although the stochastic shock in the model gives some noise around this value.

References

- [1] Fisher, P.G. (1992), *Rational Expectations in Macroeconomic Models*, Kluwer Academic Publishers, Dordrecht, Netherlands.

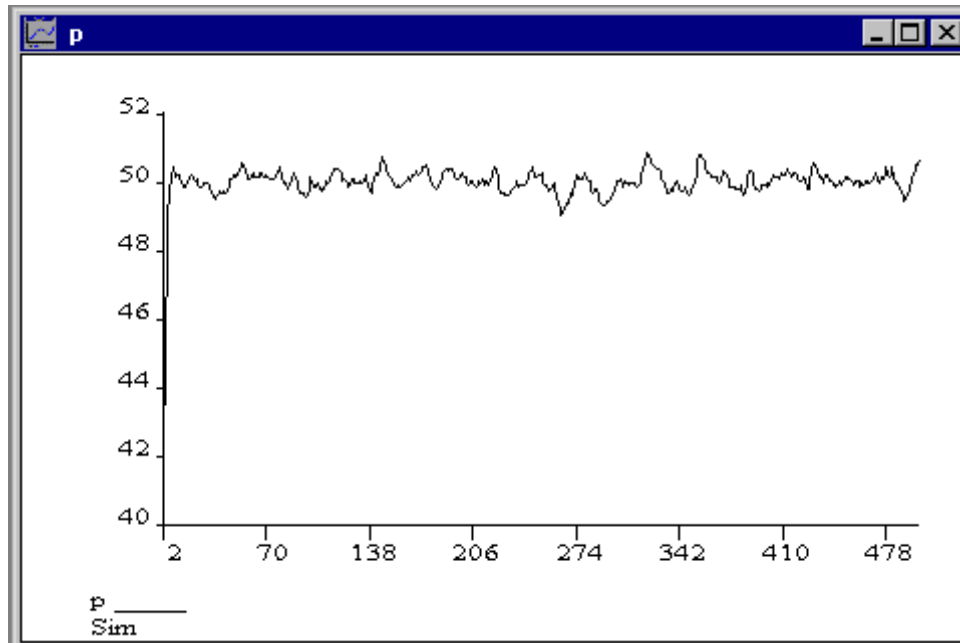


Figure 9: Dynamic solution values for p

- [2] Muth, J.F. (1961), 'Rational expectations and the theory of price movements', *Econometrica*, 29, 315–335.
- [3] Pierse, R. G. (2000), 'WinSolve Version 3: An introductory Guide', Department of Economics, University of Surrey.