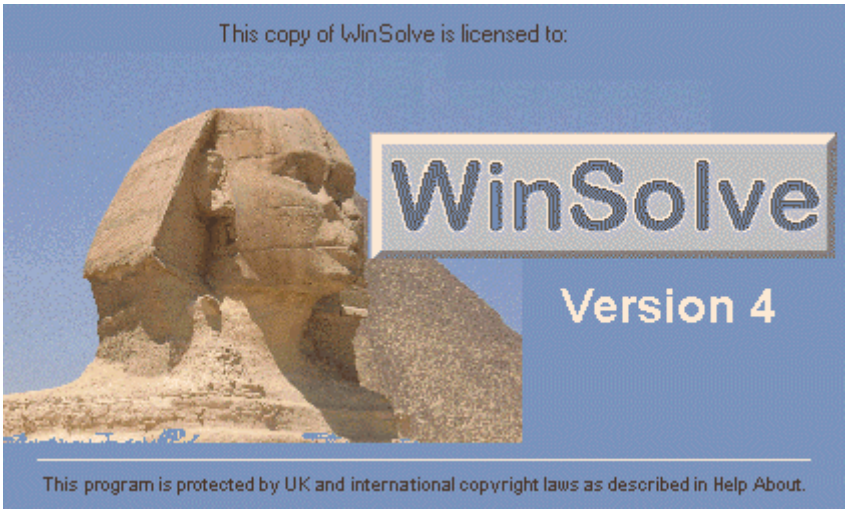
WinSolve Version 4: An Introductory Guide

Richard Pierse

October 2007



WinSolve is a program for solving and simulating non-linear models. It will handle a wide class of models including both theoretical models such as stochastic dynamic general equilibrium (SDGE) models and econometric models such as macroeconomic models. *WinSolve* is a 32 bit Windows application that runs under all current versions of the Windows operating system. There are no explicit hardware requirements; the size of model that can be built and solved is limited solely by the memory available on your computer.

A home page for *WinSolve* exists on the World Wide Web at address: *www.winsolve.co.uk*. This has background information about the *WinSolve* research programme and a browseable version of this guide. Up-to-date information about bug fixes and upgrades is also available from this page. Please send any comments and report any bugs to me at *richard.pierse@yahoo.com*.

What's New in Version 4

Version 4 of *WinSolve* is a major new release and has the following new features:

1 **Linear solution algorithms** New linear solution algorithms have been added. These compute first order perturbation approximations in either the levels or logarithms of the model variables. The model is automatically reformulated in a general state space form and then solved using a generalised real Schur decomposition (*qz algorithm*) that deals automatically with singularities. See *Section 9*.

2 **Stochastic simulation** Stochastic simulation now allows the user to produce quantiles of the distribution as well as a consistent estimate of the mode. The quantiles can be displayed in the form of true fan charts of the empirical distribution of the simulated variables. See *Section 10*.

3 **LQ optimal control** New linear quadratic optimal control methods have been added to augment the non-linear control methods already available. The new methods allow both optimal control and time-consistent control solutions to be computed in non co-operative games involving multiple players. If the model is not already linear, it will be automatically linearised or log-linearised. See *Section 11*.

4 **General non-linear optimal control** A new non-linear control option allows loss functions to be written as user-defined model equations. Arbitrary loss function can thus be specified. See *Section 11*.

5 **Steady state solution mode** A new solution mode is available. This solves numerically for the steady state of a model by collapsing the dynamics and solving the system as if it were static. See *Section 7.1.6*.

6 **Model language extensions** The *WinSolve* model language has been extended to allow the definition of temporary variables. These variables can be used within model equations but are substituted out of the model at the compilation stage and so do not appear in the final model. See *Section 3.3*.

1 Installing *WinSolve*

An automatic installation program is available.

1.1 Installing from CD
Simply insert the installation CD into your computer and follow the instructions on the screen.
1.2 Installing from the Internet
First, download the installation archive from the *WinSolve* web site and save it to a clean directory on your computer. Then, locate the downloaded file in the directory where you saved it and double click on the file icon to run it and extract all the installation files. One of these files will be named **setup.exe**. Double click on this file to start the installation process and follow the instructions on the screen.

2 Getting Started

The following outlines the steps involved in building and solving a new model within *WinSolve*:

1 Select the **File | New** command. This creates a new model object represented by a new window on the *WinSolve* desktop. This window contains an icon representing the equations of the model, which are not yet defined.
2 Double click on the icon within the model object window (or select the **File | Edit** command). This opens a text editor in which the equations of the model should be entered. The notation for writing equations is explained in Section 3.3 of this guide. When finished, quit the editor and save the file. *WinSolve* will know that your model has changed and start to compile the equations. A box shows the progress of the compilation and any errors are flagged. If there are any errors, then select the **File | Edit** command again to return to the editor and correct them. When all errors have been eliminated, *WinSolve* will prompt you to enter an optional description of the model. You now have a complete model but, before it can be solved, it must be associated with a data set.
3 Select the **Data | Open** command to read in a set of data (and adjustments) in one of the supported data formats, or, alternatively, construct a new data file with the **Data | New** command. If the program still does not have all the data it requires, then you will be prompted as to what additional variables are needed. Data observations can be revised in the **Data | Edit** menu (Section 5).
4 The model assumptions can be changed using the options on the Assumptions menu. For example, this menu will allow you to fix or unfix equations, change the solution ordering of equations, or to switch between alternative equations or terminal conditions.
5 You can solve the model using the **Solve | Solve** command. Several types of run are possible and the **Solve | Options** command allows you control over the solution algorithm (Sections 7 and 8). You can also run stochastic simulations (Section 9) or optimal control exercises (Section 10). Each new run is given a name.
6 Results can be viewed as tables, graphs or spreadsheets using the View menu (Section 11). The **Results | Save** command allows runs to be saved to disk.
7 The model itself can be saved using the **File | SaveAs** menu option. This stores the current state of the model in a file with a `.MOD` suffix. In subsequent runs, this model can be retrieved with the **File | Open** command. If the data has been changed, then the updated data should be saved in one of the *WinSolve* supported formats using the **Data | SaveAs** option.

3 Models

3.1 Model Objects

A model object is a collection of information that completely defines a model. This consists of the model equations in a compiled form plus all the assumptions such as choice of solution algorithm, convergence criteria, residual type and equation ordering etc. that are part of the model definition. The model object also has links to the file of model source code and an optional link to a data file associated with the model. The model object is stored on disk in a `.SMF` file.

An open model object is represented on the *WinSolve* desktop by a window containing an icon. Any number of model objects can be open simultaneously. The icon in the model window is a link to the model's source code. Double clicking on this icon (or selecting the **File | Edit** menu command when the model is active) activates the link and opens an editor to view and edit the model equations. When the model equations are changed,

WinSolve receives notification and automatically recompiles the model. This ensures that the model object and its source code form are always synchronised.

3.1.1 Creating a new model object

The **File | New** command creates a new model object. A new window on the *WinSolve* desktop represents this object. An icon within this window represents a link to the model's equations. As yet these are not defined. Double clicking on the icon or selecting the **File | Edit** command from the menu opens a text editor. This enables you to enter the model equations using the model description language described in Section 3.3. Once the equations have been entered, exit the editor and save the file. *WinSolve* will then (re)compile the equations. If any errors are found in compilation, then the equations will need to be corrected by selecting the **File | Edit** command again. When all errors have been eliminated, *WinSolve* will prompt for a description for the model. Your model is now complete and can now be saved as a `.SMF` file using the **File | Save** command.

3.1.2 Opening a model object

An existing model object can be opened using the **File|Open** command. Three types of file can be opened:

1 a compiled model file (`.SMF`) created by a previous run of *WinSolve* containing the compiled model plus links to a source code file and a data file associated with the model
2 a source code file of model equations in simple text (.TXT) format
3 a data file in any one of the formats accepted by *WinSolve*.

When a source code file is opened, *WinSolve* attempts to compile the model equations to create a model object. If any errors are found, the user should double click on the source code icon in the model window (or select the **File | Edit** menu command) to edit the model source and correct the errors.

The **File | Open** command can also be used to open a data file in any of the formats accepted by *WinSolve*. In this case the data file is treated as a special model object. The data file can be viewed and edited using options on the **Data** and **View** menus and can be saved in any of the available formats. It is possible to use this last facility to get *WinSolve* to convert one data format into another.

3.2 Model Equation Files

Model equations are maintained in standard text files (with a `.TXT` suffix). The model object contains a link to the model equation file so that equations can be edited from within *WinSolve* using the **File | Edit** command. When this command is selected, *WinSolve* will open the application associated by Windows with the (`.TXT`) file type to allow the model equations to be edited.

By default Windows associates `.TXT` files with the *Notepad* utility. However, *Notepad* has a limit on the size of file it can handle so that it may be better to switch to an alternative text editor such as *WordPad*. In Windows, file associations can be set up in *My Computer*. Open *My Computer*, choose the **Tools | Folder options** menu option and select the *File Types* tab. Scroll down the list until you find the 'TXT Text File' option and click on the *Change* button to change the program associated with the `.TXT` extension.

3.3 The Model Description 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 operators `+ - * / ^`, and references to functions. 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 that the *WinSolve* compiler will ignore. Two kinds of comment can be used. Either the single quote character '`'`' or the 'at' character '`@`' indicates that the rest of the current line is a comment. Alternatively, any text can be treated as a comment by delimiting it by a pair of brace characters '`{`' and '`}`'. This latter form of comment can be nested and can appear anywhere in the model code.

The algebraic notation for defining equations is best explained through an example:

```
*P    intercept = 100.5 ;
@ Equation 2: Consumption function
*M    log(C) = intercept + 0.15*log(C(-1))
  { This is a comment which is ignored }
           + 0.85*log(Y) ;
@ Equation 3: Income identity
      Y = C + Z;
```

In this code sample there are three equations. The first defines a parameter, '**intercept**', used in the following equation. In the second equation, 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 placed in-between. In the third equation, variable Y is defined as the sum of C and variable Z.

The characters, preceded by an asterisk, that appear immediately before the start of the first and second equations are codes. Their purpose is to give information to *WinSolve*. The code '**M**' tells the program that the equation for C has multiplicative adjustments. The code '**P**' tells the program that '**intercept**' is a parameter and not a variable. Allowable codes are:

A denotes that the equation should have additive adjustments

C denotes that the line is not an equation but an (optional) equation description

M denotes that the equation should have multiplicative adjustments

P denotes that the equation is a parameter definition taking the simple form
***P name = value;**
Parameter names must be unique. Parameters can then appear in subsequent equations in place of coefficients.

T denotes that the equation is a terminal condition.

W denotes that the equation defines a temporary variable. Temporary variables are substituted out of the model by the compiler to be replaced by their definition.

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 Section 3.5 below.

Note that in the second equation in the example, a function appears on the left-hand side of the equation. *WinSolve* automatically re-normalises 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. The argument of such a function must be a single current-dated variable. For example, both the expressions `log(C/Y)` and `log(C(-1))` would be illegal on the left-hand side of an equation.

More examples of models coded for *WinSolve* are provided in the **dat** sub-directory of the main *WinSolve* directory.

In Version 1 of *WinSolve*, all lines were ignored *except* those starting with an asterisk in the first column. This meant that equations running over several lines had to have an asterisk at the start of each line. For backwards compatibility, it is still possible to configure *WinSolve* to use the earlier method by choosing the 'Ignore Unstarred Lines' option on the **Options | Compiler** menu. Note that comments within starred lines will still be ignored. It is recommended that users convert models to the new format and avoid this option.

3.4 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 '`_£$%#`'. By default, names are not case sensitive so that '`ABC`' and '`aBc`' represent the same variable. However, this default setting can be overridden in the **Options | Compile** menu. Names of *WinSolve* functions cannot be used as variable names.

3.5 List of *WinSolve* functions

WinSolve supports the following functions (# indicates an *invertible* function):

Standard mathematical functions

`abs(x)`		absolute value of 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 minus infinity)
`log(x)`	#	natural logarithm of x (x>0); synonyms: `ln` , `alog`
`max(x1,..,xn)`		maximum of n arguments
`min(x1,..,xn)`		minimum of n arguments
`sin(x)`	#	sine of x (x in radians)
`sqrt(x)`	#	square root of x (x non-negative)
`tan(x)`	#	tangent of x (x in radians)

Statistical functions

`cdfchi(x,n)`	cumulative chi-square density from 0 to x (x>0) with degrees of freedom n
`cdff(x,n,m)`	cumulative F density from 0 to x (x>0) with degrees of freedom n and m
`cdfn(x)`	cumulative normal density from minus infinity to x
`cdft(x,n)`	cumulative t density from minus infinity to x with degrees of freedom n
`norm(x)`	normally distributed quasi random number with variance x (x>0)
`unif(x)`	uniformly distributed quasi random number with range [0,x] (x>0)

Variable functions: these take as argument a variable (with optional lag)

`adjust(v)`		Adjustment value associated with variable v
`base(v)`		Base value (or data value) of variable v
`distlag(v,n,w1,..,wn)`		n period distributed lag of variable v with constant weights w1..wn
`distlog(v,n,w1,..,wn)`		n period distributed lag of log of variable v with constant weights w1..wn
`diff(v)`	#	First difference of variable v ; synonym `delta(v)`
`diff4(v)`	#	Fourth difference of variable v ; synonym `delta4(v)`
`dlog(v)`	#	First difference of log of variable v ; synonym `dln(v)`
`d4log(v)`	#	Fourth difference of log of variable v ; synonym `d4ln(v)`
`obs(v,d)`		Data observation of variable v in period d (d must be a valid date)
`ratio(v)`	#	Ratio of variable v to its first difference
`ratio4(v)`	#	Ratio of variable v to its fourth difference

Logical functions

`ifeqz(x)`	= 1 if argument = zero; = 0 otherwise
`ifgtz(x)`	= 1 if argument > zero; = 0 otherwise
`ifltz(x)`	= 1 if argument < zero; = 0 otherwise

Date and time functions: these take as argument a valid date

`ifeq(d)`	= 1 if solution period is equal to argument, = 0 otherwise
`iflt(d)`	= 1 if solution period is less than argument, = 0 otherwise
`ifgt(d)`	= 1 if solution period is greater than argument, = 0 otherwise
`ifne(d)`	= 1 if solution period is not equal to argument, = 0 otherwise
`ifle(d)`	= 1 if solution period is less than or equal to argument, = 0 otherwise
`ifge(d)`	= 1 if solution period is greater or equal to argument, = 0 otherwise

`seas(i)`	= 1 in the ith season of the year; = 0 otherwise
`time(d)`	linear time trend taking the value one in period d

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 1974q1 and minus one in 1974q2 could be coded by the expression:

```
ifeq(197401)-ifeq(197402)
```

For a more complicated example, the expression:

```
5+ifge(198501)*ifle(198504)*time(198501)+ifgt(198504)*5
```

defines a dummy taking the value 5 before 1985q1 then rising by increments of one to the value 10 in 1986q1 and thereafter.

3.6 Special *WinSolve* functions

In addition to the functions listed in the previous section, two special functions are available in *WinSolve*. The first implements the parameterised expectations algorithm of den Haan and Marcet (Journal of Business and Economic Statistics, 1990, Vol. 8, 31-34) and the second a Kalman filter learning algorithm of Hall and Garratt (Economic Modelling, 1995, Vol. 12, 87-95).

`parexp(y,x1,..,xk[,b0,..,bm],n,k)` See *Section 8.2.3* for details

`y`	variable expectation to be approximated (should be a lead variable)
`x1,..,xk`	state variables (should be current or lagged)
`b0,..,bm`	optional initial values for power function parameters if n=1 then m = 1+k or if n=2 then m = 1+k+k*(k+1)/2
n	order of power function approximation (n=1 or n=2)
k	number of state variables

`learn(a1,..,ak,z1,..,zk,zl1,..,zlk,v [,q1,..,qk[,p1,..,pn]],k)`

`a1,..,ak`	Kalman learning coefficients (should be current dated variables)
`z1,..,zk`	Kalman learning variables (should be predetermined)
`zl1,..,zlk`	lagged learning variables (should be predetermined)
v	learning error
`q1,..,qk`	optional values for learning parameter variances (default=1)
`p1,..,pm`	optional initial values for lower triangle of Kalman filter variance m = k*(k+1)/2 (default is that P0 is an identity matrix)
k	number of state (learning) variables

4 Data Files

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

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. The **Data | New** menu command allows the user to create a new data set for all the variables of a model, optionally setting all observations to a fixed value. The observations can subsequently be edited using the **Data | Edit** menu option. Alternatively, data can be read from file in any of the formats supported by *WinSolve* using the **Data | Open** command. Data sets can then be saved in any of these supported formats.

WinSolve reads and writes data files in the following formats:

SDF	Default *WinSolve* binary format used to store data and/or adjustments
BN7, IN7	PcGive, PcFiml, Stamp 5.0 binary format

BNK	*DataView* & *Modler* binary format (read only)
CSV	Comma delimited ASCII spreadsheet format
DAB	*Warwick MMB* data format.
DAT	*Gauss* binary data format
DB	*FAME* data bank format (special build only) or *HMT* binary format
DEM, GEM	*NIMODEL* binary format
FIT	*MicroFit* binary format
FRM	*TROLL* portable data format.
NIM	*NIMODEL* ASCII format
RAT	*RATS* binary data bank format
RUN	*LBS/CEF* binary data bank format
WF1	*EViews* binary work-file format
WK1	*Lotus 123, Excel & Qpro* spreadsheet format
XLS	*Microsoft Excel* Worksheet/Workbook format

In order for *WinSolve* to recognise the data format, the appropriate suffix should be used.

Note that many of these formats have rules on the maximum number of variables, the length of variable names and the characters allowable within names, which are more restrictive than those of *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.

Adjustments
Model adjustments are terms that modify the outcome of the equations of the model, and may be applied either additively or multiplicatively. Their purpose is to proxy information not included in the model, or (as a temporary measure) to correct an equation which has been systematically under- or over-forecasting. They are also used in simulations to shock equations or exogenous variables from their base values. Model adjustments can be read from file in the same way as data files using the **Data | Adjust** option.

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 two-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.

Spreadsheet Formats
In order for *WinSolve* to successfully read data in the spreadsheet formats (WK1, XLS and CSV), the data need to be organised in a particular way. The first column should contain dates in valid *WinSolve* date notation, starting from the second row. Each subsequent column should contain the observations for a variable, with the first row giving the variable name followed by an optional title. Empty cells indicate missing values. *WinSolve* ignores the cell corresponding to the first row and column. *WinSolve* will now read and write multi-sheet Excel Workbooks.

5 Editing Data

The **Data | Edit** dialog box provides the user with several options for editing data or adjustments. A graph displays the current state of the data and is updated as the observations are edited. However, the edit can be cancelled and the original state restored by clicking the cancel button. Only when the update button is selected, will the edit actually be implemented

5.1 Editing observations
This is the most conventional method of editing data. The data are displayed in a scrollable table. By double clicking on the selected observation (or pressing the Enter key) an edit window is opened to allow that observation to be retyped. Pressing the Enter key again returns to the table. When editing is finished, pressing the ESC key or clicking on the close window icon returns to the dialog box.

5.2 Drawing by hand

Selecting this option allows the user to edit the graph of the data freehand using a mouse. Position the mouse over the graph (a pencil cursor appears). Then, with the left-hand button depressed, drag the mouse to redraw the graph. Maximising the graph window may facilitate precise drawing. In freehand drawing mode, the *line draw option* allows a straight line to be drawn between any two points. With the mouse positioned over the graph, click the right-hand button and select the line draw option. The pencil cursor changes to an arrow. Then move the mouse to the point where the line should start. Depress the left-hand mouse button, drag the mouse to the point where the line should end, and release the button. A straight line will be interpolated between the two points.

5.3 Projecting simple processes

This dialog box allows data to be constructed following the simple model:

$$X(t)=a + b*T + c*X(t-1) + d*U(t)$$

where *a, b, c, d* are coefficients to be selected by the user, T is a linear time trend and U is a normally distributed pseudo-random variable with unit variance. Judicious choice of coefficients allows the selection of a variety of models. This option is particularly useful for the projection of adjustments.

5.4 Estimating Time Series Models

This option allows the projection of a data series on the basis of estimated integrated autoregressive models (*ARIMA(p,d,0)* models) fitted over the past to either the level or the logarithm of the series. The appropriate order of differencing (d) can be tested by Augmented Dickey-Fuller tests, given the autoregressive lag length p selected by the user. The estimated model (which can include deterministic components constant, trend and/or seasonal dummies) is then projected forward over the solution period, either deterministically or stochastically. This option is most useful for the projection of exogenous variables.

6 The Assumptions Menu

The Assumptions menu allows the user to change various assumptions about the model.

6.1 Reordering model equations:

By default, the model equations are solved in the order that they appear in the equation listing. The **Assumptions | Reorder** option allows the user to specify a different solution ordering read from a file, or to write the current ordering to a file. The logical dependencies of variables in the model can also be viewed, which can be useful in defining an optimal ordering. The solution ordering can be important if the model is being solved using the Gauss-Seidel solution method or one of its variants. Ordering can affect the speed of convergence, or even in extreme circumstances, whether or not the model converges at all.

6.2 Fixing and Unfixing Variables:

This option allows endogenous variables to be fixed or unfixed over the whole solution period or any sub-sample. When a variable is fixed its equation is ignored and the variable treated as though it were exogenous. A variable may be fixed indirectly using a second variable. In this case the first variable is treated as a target and the second variable as an instrument. The instrument is chosen so that the equation for the target variable tracks the data exactly. This is known as *Type 2 fixing*. Any number of Type 2 fixes may be imposed although no variable may be an instrument for more than one target. Whether it is possible for targets to be achieved depends on the structure of the model and the interdependence of targets and instruments. When specifying Type 2 fixes, a multiplier needs to be specified. This is an approximation to the partial derivative of the target variable with respect to the instrument. The default value of this parameter is one. When the model is solved using Newton's algorithm, *WinSolve* calculates this derivative analytically and ignores the value of this multiplier.

6.3 Changing Adjustment Type:

This option allows the user to select the type of adjustment to be added to each equation. There are two types of adjustment: additive or multiplicative. Generally, the adjustment type for each equation is set in the model definition by using the **`*A`** or **`*M`** code. However, any equation not explicitly typed has adjustments of the default type, which is *additive*, *multiplicative* or *intelligent*. If default type intelligent is selected, then the functional form of the equation determines the adjustment type. When the left-hand side of the equation is a multiplicative function (`log`, `exp`, `dlog`, `dlog4`, `ratio`, *ratio4*) then the adjustment is *multiplicative*, otherwise it is *additive*. Note that if the left-hand side is not a function, then adjustments are assumed to be

additive, whatever the form of the right-hand side. This menu option allows the user to set the default adjustment type and also to over-ride the explicit typing in the model definition.

6.4 Switching between Alternative Equations:
If the model contains more than one equation for any variable, then these equations are treated as alternative equations. Only one of a set of alternative equations will be executed and by default this will be the first equation. This option allows the user to select which one of a set of alternative equations is executed. In order to identify alternative equations, the user should provide equation descriptions ('***C**' lines) in the equation code before each such equation.

6.5 Switching Terminal Conditions:
When the model equations contain any variables with leads, then the model needs to be completed by a definition of the value of these leads *beyond the end of the solution period*. These are known as terminal conditions. Six standard types of terminal condition are available in *WinSolve*:

1 constant level
2 exogenous value
3 constant period-on-period growth rate
4 constant annualised growth rate
5 constant long-run growth rate
6 constant difference

In addition, user-defined terminal conditions can be defined by equations specified as part of the model code using the '***T**' equation type. 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 descriptions in the equation code to identify each terminal condition. In the absence of any user-defined terminal conditions, the default is to use a constant level condition. This option allows the user to over-ride the defaults and select which terminal condition is used for each equation.

6.6 Setting the ragged edge:
Generally the ragged edge in a genuine forecast (the point where the data for a variable ends and the model equation should be used for forecasting) is automatically determined by *WinSolve* (See *Section 7.2*). However, an option allows the user to set up the ragged edge manually or modify the current ragged edge. This is useful when the data set consists of a combination of genuine data and previous model forecasts.

7 Solving the Model

The **Solve | Solve** menu command brings up the Solution Dialog Box and initiates model solution. While a model is being solved, *WinSolve* displays a status box. This gives information on the number of iterations that have elapsed, the elapsed time, the number of unconverged variables etc. There is also a cancel button that allows the user to abort the solution process at any time.

When the model contains forward expectations, then it is possible to display graphs to monitor in real-time the progress of up to four variables as solution progresses. The system menu in the top left-hand corner of the monitor window can be used to change the variables monitored as solution progresses.

7.1 Model Solution Modes
There are seven available modes of solution:

7.1.1 Dynamic model forecast Dynamic model solution is the basic solution mode. The equations of the model are solved simultaneously, period by period, with the solution values for previous periods being used as lagged values in subsequent periods. In this mode no observations are required for the endogenous variables over the solution period. All other solution modes require a full set of observations on all variables for the full solution period, and will therefore not be available unless the data set is complete over the solution period.

7.1.2 Simulation mode: In this mode the model is solved first in *single equation solution* mode. Then the model is solved again in *dynamic model forecast* mode adding back the single equation residuals plus any user-specified adjustments. If no adjustments have been specified, then the model should solve in a single

iteration to reproduce the simulation (data) base. If adjustments have been specified, then the (percentage) differences of the solution from base represent the effects of the adjustments relative to the base. This mode is the simplest way to perform model simulations.

7.1.3 Static model forecast: In this mode the model is solved simultaneously but using lagged actual values in place of lagged forecast values. In consequence, forecast errors will not cumulate dynamically.

7.1.4 Single equation solution: Single equation solution solves each equation singly, using actual rather than forecast values on the right-hand side of each equation. The forecast residuals should reproduce estimation residuals over the estimation period.

7.1.5 Dynamic single equation solution: Dynamic single equation solution solves each equation singly but uses forecast values rather than actual values for lagged dependent variables.

7.1.6 Steady state solution: Steady state solution solves the model simultaneously but collapsing the dynamics so that all lags and leads are set equal to the current period solution. If the model has a flat steady state solution then this mode will try to find it numerically.

7.1.7 Creating a simulation base: This option solves the model in the same way as a dynamic model forecast. However, it *overwrites* data values with the model solution values. It also unfixes each exogenised variable and sets an implicit adjustment for it so that the reinstated equation should reproduce the effect of the fix. The result is to create a simulation base. A subsequent dynamic forecast without any changes to adjustments will reproduce the base and should converge in a single iteration.

7.2 Dealing with the ragged edge in forecasting

When generating a genuine forecast over the future, it is often the case that the data set is incomplete for the current period (and sometimes for the previous few periods). Where data is available, this should over-ride the model equation but otherwise the model needs to be solved in order to generate a forecast for the missing values. *WinSolve* provides an option to deal automatically with this ragged edge problem. When this option is selected, a model equation is ignored (the variable exogenised) for any period where an observation is available. By default, this option is turned on if the data set is incomplete and turned off otherwise. The ragged edge for each variable can also be modified directly by the user. See *Section 6.6*.

7.3 Skipping exogenised equations

By default, when equations in a model have been exogenised (fixed), *WinSolve* still executes these equations in a solution in order to calculate *implicit residuals* (the difference between the exogenised value and the equation prediction). Checking the skip exogenised equations option in the Solution dialog box over-rides this default and instructs *WinSolve* to ignore all equations for exogenised variables. This may be useful in rare circumstances where badly calibrated equations are causing immediate model failure. However, since it prevents *WinSolve* from calculating implicit residuals, this option should in general remain unchecked.

7.4 Monitoring variables

Checking the monitor variables option opens a dialog box to allow the user to select variables to graph in real-time as the model solves. Note that real-time graphing will slow down the iteration process so should be avoided if speed is of paramount importance.

7.5 *Starting from the last solution point*

By default, *WinSolve* starts solution from an initial point determined by the availability of data. If data values exists for the whole of the solution period, then they are used as initial values, otherwise, the solution from last period is used. However, where a solution has been obtained in a previous run, then checking the *Start from last solution* box in the Solution dialog box allows this solution to be used as the initial point in a new run. If the previous run had fully converged and if no changes have subsequently been made to the model or assumptions, then a run starting from the previous solution point should converge in a single iteration. More usefully, if the previous run had run out of iterations without converging, then checking this box would allow the solution to resume from its final point.

8 Solution Options

The Solution Options dialog box, which is accessed by clicking the Options button in the Solution Dialog box, allows the user control over several aspects of the model solution.

8.1 Solution Methods for Models without Leads

WinSolve provides four model solution methods for standard models that do not involve variables with leads: Gauss-Seidel, Jacobi, Fast Gauss-Seidel and Newton's method. All these methods solve the model period by period and forwards in time. Newton's method uses derivatives (automatically computed analytically) and convergence is independent of the equation ordering or normalisation. This method is generally to be preferred for small models but can become expensive for large models since it requires the inversion of a matrix in every iteration. Gauss-Seidel is the default method for large models. However, solution of the Gauss-Seidel method depends on the ordering of the model equations, whereas the Jacobi method, while slower, is not dependent on the model ordering. Hence the Jacobi method can be a useful check on whether a model is badly ordered. A relaxation parameter (alpha, default value 1) can dampen the changes between iterations. When this parameter is not equal to one, Fast Gauss-Seidel (a variant of Gauss-Seidel) can be considerably faster than the plain Gauss-Seidel method.

8.2 Solution Methods for Models involving Leads

For models containing forward-looking expectations, four algorithms are available: Fair-Taylor, stacked Newton, parameterised expectations, and a model specific method supplied by the user in a flexible language.

8.2.1 Fair-Taylor: This is the default method for large models. This algorithm first solves the model over all time periods, using either Gauss-Seidel or Newton's method, taking expectations as fixed. These iterations are known as inner loops. Then the expectations are updated and the process repeated until convergence is achieved. These latter iterations are known as outer loops. Different convergence criteria can be specified for inner and outer loops, to provide so-called incomplete inner iterations. Fair-Taylor is a cheap solution method but is not very robust and does not work well with highly nonlinear models.

8.2.2 Stacked Newton: This algorithm solves the model using Newton's method, treating the system of n equations over T periods as a single simultaneous system of order nT. This makes for a very large problem but *WinSolve* takes advantage of the structure of the matrix of model derivatives to ensure that the system is solved efficiently, avoiding unnecessary storage. Analytic derivatives are used and the method is guaranteed to converge in a single iteration for a linear model. This is a very powerful solution method but is expensive for large models.

8.2.3 Parameterised expectations: This is a method of solving models described by den Haan and Marcet (Journal of Business and Economic Statistics, 1980). It replaces each forward expectation in the model by a non-linear function of a set of k predetermined state variables, which is re-estimated each iteration using non-linear least squares. The functional form is a power function of order p, defined as the antilogarithm of a p-th order polynomial in the logarithms of the k state variables. Note that all state variables have to be strictly positive. In *WinSolve* expectations can be parameterised by replacing each forward expectation in the model by a special function, `parexp`, defined by

```
parexp(y,x1,..,xk,[,b1,..,bn,]p,k)
```

where `y` is the forward expectation to be approximated, `x1` to `xk` are the k predetermined state variables, `p` is the order of the power function and `k` is the number of state variables. The arguments `b1` to `bn` are *optional* initial values to be used for the coefficients of the power function. Note that for $p=1$, $n=1+k$, for $p=2$, $n=1+k+k*(k+1)/2$ etc. The terms are ordered so that the last index varies most rapidly. For example, with $k=3$, and $p=2$, the order is {*1,x1,x2,x3,x1^2,x1*x2,x1*x3,x2^2,x2*x3,x3^2}*. In practice, the n variables in the power function can often be highly collinear. *WinSolve* allows some of the terms to be dropped, which is specified by setting the appropriate element of `b` to zero. Having parameterised each of the expectations, the model can be solved, using the Fair-Taylor solution algorithm. Note that this will not actually use Fair and Taylor's method since the *parexp* function will be doing all the work.

8.2.4 User specified (DIY) algorithm: It is sometimes efficient to solve a model by splitting it into sub-blocks and solving each sub-block separately, iterating around all the blocks until the whole model converges. Equations might appear in more than one block, and some blocks might be more efficiently solved backwards rather than forwards in time. Each block should be allowed to have its own convergence criteria and relaxation

parameters. Such an approach takes account of the dependencies between equations but is obviously model-specific, requiring an expert knowledge of structure of the model. *WinSolve* provides access to such an approach by allowing the user to specify a customised solution algorithm using a general language.

8.3 Solution Errors

In the course of model solution it sometimes happens that an illegal operation will occur, such as taking the logarithm of a negative number or dividing by zero. By default, if such an illegal operation is encountered, *WinSolve* will abandon solution, displaying an error message pinpointing the source of the error. It is possible, however, to make the program continue with solution after an illegal operation, substituting a pre-set value for the offending argument. A maximum number of errors can be allowed before solution is finally abandoned.

8.4 Convergence criteria

The model solution iterations continue until either one of two criteria is satisfied by all the equations: the percentage change between iterations is less than a given tolerance level or the absolute change between iterations is less than a tolerance level. Both these tolerances can be set in the Solution Options dialog box. It is also possible to set specific tolerances for individual equations. The maximum number of iterations can also be set as can the iteration from which all unconverged variables will be listed to a window. This information can be useful in tracking the source of model non-convergence.

For models containing forward expectations, convergence criteria can also be set for the outer loop iterations over the expectational variables.

9 Model Approximation Methods

In addition to the non-linear solution methods considered in the previous section, Version 4 of *WinSolve* also includes alternative solution methods based on approximation of decision rules by a Taylor expansion in terms of the levels or the logarithm of the variables, the approximation taken around the deterministic steady state. These methods are known in general as perturbation methods and in the first order case are more familiar as the linear (or log-linear) approximation of the model. The **Solve | Model approximation** menu command allows choice between several options.

9.1 Computing a perturbation solution

This option computes a perturbation approximation to the solution of a model, around its deterministic steady state. For the first order case, the model is automatically recast into a (possibly singular) state space form in terms of either the levels of the variables or their logarithms. The Blanchard-Kahn conditions for a unique stable equilibrium are then computed and the solution found using a generalised real Schur decomposition (*qz algorithm*). For perturbation orders greater than one, the higher order terms are then computed recursively by solving a set of generalised Sylvester equations at each stage, making use of higher order model derivatives computed using automatic (analytic) differentiation.

9.2 Computing model derivatives

This option computes tensors of model derivatives up to a specified order. The derivatives are computed analytically using automatic differentiation techniques based on a high order multivariate generalisation of the chain rule of elementary calculus (the multivariate Faa di Bruno formula).

9.3 Spectral radii

The convergence of a first-order solution technique such as Gauss-Seidel depends on the spectral radius of a linear function of the model Jacobian matrix. This option computes various spectral radii for the model Jacobian at a selected data point, given the current ordering of model equations.

9.4 Saving model approximation

The model approximation produced by perturbation can be saved in a *WinSolve* file format.

10 Stochastic Simulation

The **Solve | Stoch** menu command opens the Stochastic Simulation Dialog Box and allows the user to select between many options for stochastic simulation of a model. Stochastic simulation involves solving the model a

number of times (called the number of replications), each time adding a pseudo-random shock to each equation, and drawing a new set of shocks for each replication. The average of the resulting simulations is a measure of the expected value of the solution. (In a non-linear model this will be different from the deterministic solution.) The standard deviation of the simulations is a measure of the standard error around this expected value. Both these measures are computed in a stochastic simulation along with higher moments (skew and kurtosis). All are available to graph or save to file. Optionally, quantiles can also be computed to allow the examination of the whole distribution, which can be displayed graphically in the form of fan charts.

WinSolve provides a range of options for generating the stochastic shocks. In models with forward-looking behaviour, shocks are allowed to be either fully anticipated or unanticipated by agents.

10.1 Shock Generation Methods

There are five methods available for generating stochastic shocks. The first three are based on the single equation residuals from the model and therefore require a full data set; these methods are only relevant to a model that has been estimated econometrically. The fourth option allows the user to specify a variance covariance matrix for the shocks while the fifth option allows the user to read shocks directly from a file. In the first three cases, the shocks can be generated using the residuals over a sub-period only: this is called the data period. In all cases, the shocks can be applied over a period that may be only a sub-period of the solution period.

10.1.1 Cholesky method: The maximum likelihood estimate of the variance covariance matrix of the shocks is computed from the model single equation residuals (sometimes known as the Nagar method.) This method requires that the number of endogenous variables is less than the number of observations, so that it may not be feasible in large models. It also assumes a joint normal distribution of the errors. A generalised decomposition is used so that any equations such as identities that have zero single equation residuals will automatically not receive shocks. A degrees of freedom correction can be imposed to correct small sample bias.

10.1.2 McCarthy method: Shocks are generated as a linear combination of the rows of the matrix of single equation residuals, using normally distributed random weights. This method has the advantage over the Cholesky method that it does not require that there are more observations than endogenous variables so that it can be applied in large models.

10.1.3 Bootstrap method: Shocks are generated by repeatedly randomly drawing rows from the matrix of single equation residuals. The shocks drawn will asymptotically have the same distribution as the empirical distribution of the single equation residuals. This method does not assume normality.

10.1.4 User variance matrix: This option allows the user to specify a variance-covariance matrix for the shocks. It is designed for theoretical models or to allow the calculation of impulse responses.

10.1.5 User-specified shock file: This option allows the user to read shocks directly from a data file. This file can be in any supported format in which the data is organised *by observation* e.g. *.csv, .xls, .wk1, .dat*. The rows of the data file should be ordered: observations *n1-n2* for replication *1*, observations *n1-n2* for replication *2* etc. This option provides great flexibility and allows shocks to be generated from any desired distribution.

10.2 The random seed

The user can set the seed used to initialise the random number generator. Initialising the seed to the same value will generate the same sequence of pseudo-random numbers. This enables a stochastic simulation to be replicated at a later date.

10.3 Antithetic shocks

Antithetic shocks are pairs of shocks with the same magnitude and opposite sign. Using antithetic shocks will increase the precision of a simulation. In a linear model, a simulation with antithetic shocks will have a mean exactly equal to the deterministic solution of the model.

10.4 Anticipation of shocks

In a model with forward-looking behaviour, there is an important question of whether shocks are anticipated by expectations-forming agents. *WinSolve* allows two possibilities: either the shocks are fully anticipated from

the start of the solution period, or they are not anticipated until the period in which they occur. The latter possibility involves a considerably larger computational burden and so can be expected to be much slower. Note that if expectations have been parameterised, then the parameterised expectation functions are treated as fixed in stochastic simulation and will not be updated. This implies that, for these expectations, shocks will always be unanticipated, regardless of whether or not this option is selected.

11 Optimal Control

The control menu allows users to define and solve optimal control problems involving one or more players, each player attempting to minimise a loss function of a set of target variables, given a set of variables under its control. Both non-linear and linear control techniques are available, the latter allowing alternative time-consistent control solutions to be found for the case where commitment to stick to the optimal policy cannot be assumed. When there is more than one player, then a non co-operative Nash game solution is sought, by iterating over the players, each player taking as given the reaction functions of the other players.

11.1 Loss functions

WinSolve allows choice between three different types of loss function, each giving rise to a different class of control problem. In all cases, a constant time discount rate can be applied.

11.1.1 Non-linear Quadratic Loss function

This corresponds to the well-known optimal regulator problem in optimal control where the model is non-linear. The loss function for each player is a weighted sum of the squared deviations of each target variable from its desired value, where the weights can be time-varying. Optional terms penalising changes in the control variables can also be included, imposing smoothness on the time path of the instruments. The natural solution algorithm for this control problem is the Gauss-Newton method. Two versions are available, one using numeric derivatives and the other computing derivatives analytically.

11.1.2 Linear Quadratic Loss function

This option is designed for solving standard control problems with a linear model and a quadratic loss function. The weight matrices in the loss function for each player are constant, but need not be non-singular since the efficient linear solution techniques used for solving this problem are inverse-free. As well as optimal control, alternative time-consistent control problems can also be solved for this case.

11.1.3 General Loss function

In this case, the loss function for each player is a discounted sum of the values of an arbitrary non-linear function, specified as a user-defined equation within the model. The other two loss functions are obviously special cases of this general loss function

11.2 Defining a control game

A new control game can be defined using the **Control | Setup** menu option. This initiates a sequence of dialog boxes prompting the user for the relevant information: firstly the number of players and then the target and control variables for each player. For the case of a non-linear quadratic loss function, the user then needs to define desired values for the target values and the weights attached to them. Alternatively, with a linear quadratic loss function, the user needs to specify the non-zero elements of the matrices of weights on targets and controls. Once a game has been defined it can be saved to disk in a `.sgf` file. A saved `.sgf` file can later be reopened with the **Control | Open** menu option, and edited with the **Control | Edit** option.

11.3 Solving a non-linear control game

The **Control | Optimal control** menu command brings up the Solution Dialog Box and initiates control solution. Clicking the Options button in this dialog box opens a Control Parameters dialog box allowing access to the parameters affecting the control algorithm. Four non-linear solution algorithms are available: the *BFGS quasi-Newton* method, , the *Gauss-Newton* method using either numeric or analytic derivatives and the *Powell* conjugate directions algorithm. This algorithm is used to minimise the loss function for each player. The outer loop, over players, uses a simple Gauss-Seidel iterative procedure. The maximum number of iterations in each of these two loops can be set in the dialog box. Three convergence criteria can also be set. These control the tolerances used in the outer game loop, the player optimisation and the line search within the player optimisation respectively. Clearly, the outer loops should not have tighter tolerances than the inner loops and

none should have a tighter tolerance than that used in model solution.

11.4 Solving a linear-quadratic control game

When the game to be solved is linear-quadratic, then the **Control | Optimal control** menu command brings up the Linear-Quadratic Control Dialog Box which allows a choice between optimal control and time-consistent control solution. The optimal control solution uses the generalised real Schur decomposition (*qz algorithm*) on the extended state space representation of the control problem. The time consistent control problem is solved using a Markov perfect dynamic programming algorithm based on Oudiz and Sachs but making use of inverse-free methods. Two versions of this algorithm are available according to whether the players are engaged in a Nash or Stackelberg game with rational agents in their economy..

.

12 Viewing and Saving the Results

The View menu allows the graphing and tabling of results from one or more runs. Base (data) values, simulation (forecast) values, forecast residuals (in levels or in percentages) or model adjustments can all be viewed. Variables can be displayed as annual sums or averages for either calendar or tax years. Graphs, tables or tables of summary statistics can be created interactively through the View Dialog Box, or tables or graphs can be generated from predefined files. The View Dialog box also allows results to be turned into spreadsheet files (`.WK1` files) to be analysed by any spreadsheet package. Once displayed, graphs and tables can be sent to the printer by selecting the **File | Print** option. The full set of results for a run can also be saved to file.

12.1 Switching views

The initial view box in the View Dialog Box specifies whether the initial display mode is as a graph, table, summary statistics or spreadsheet. From the initial view it is possible to switch the mode of display to any of the others by selecting the appropriate option from the window's system menu (in the top left-hand corner of the window.

12.2 Copying to the clipboard

A displayed table or graph can also be copied to the Windows clipboard using the **Edit | Copy** command. The information is written to the clipboard in several different formats including as a spreadsheet (`WK1` and `XLS` format) and as a graph (in Windows metafile `WMF` format). It can then be pasted into any package that accepts one of these file formats.

12.3 Saving the results

The full results of a run can be saved to file using the **Results | Save** command. You can choose to save either solution values, adjustments, changes or percentage changes from base and, in stochastic simulation runs, simulation standard deviations. Files can be saved in any of the supported formats.

13 Saving and Re-executing Command Sequences

WinSolve has a facility whereby the user can start logging the commands executed in a particular session (**File | Open log file**). After this command has been selected, all subsequent *WinSolve* commands will be recorded in the log until the **File | Close log file** command is selected. This prompts the user to save the log of commands to disk as a record of the session. This file is in ASCII format and is fairly self-explanatory.

A command log file can subsequently be executed to replay the commands of the session (**File | Execute log file**). This command switches *WinSolve* into a quasi-batch mode. When *WinSolve* is executing a command log file, it will not respond to user input. This is indicated by the presence of the 'Exec' indicator towards the right of the status bar. (The status bar is at the bottom of the main program window). Note that if the logged commands happened to over-write any files, or if files have subsequently been altered or deleted, then re-executing the commands may well not reproduce the previous results.

14 Customising *WinSolve*

The Options menu allows the user to customise various aspects of the operation of *WinSolve*.

14.1 Compiler options

The **Options | Compile** dialog box allows the user to over-ride some of the default settings for the model compiler.

14.1.1 Case sensitivity

Variable names in *WinSolve* are case insensitive by default, but the user can over-ride this default setting by checking the *Case sensitive* box in the **Options | Compile** dialog box. Note however that function names will always remain case insensitive.

14.1.2 Ignoring unstarred lines

By default, all lines in the model code are treated as part of the model definition and any other information not to be interpreted by the compiler must be explicitly commented out. However, it is also possible to run the compiler in a mode whereby only lines that begin with an asterisk are treated as part of the model. In this mode all other lines are simply ignored. This mode can be selected by checking the *Ignore unstarred lines* box in the **Options | Compile** dialog box.

14.2 Graph colours

The **Options | Graph** command allows the user to choose colours, line styles and fonts to be used for drawing graphs.

14.3 Locating Applications

The **Options | Location** command allows the user to set up paths to the external applications that *WinSolve* calls to perform various tasks. These include the text editor for editing model files with a '`TXT`' suffix and the spreadsheet package for displaying spreadsheet tables created by the **View | Spreadsheet** command. Once these paths have been defined, the information will be saved in the '**winsolve.ini**' file for subsequent use.

14.4 Toolbar

The **Options | Toolbar** command allows the user to customise the icons that appear on the *WinSolve* toolbar.

14.5 Model Help

It is possible for the user to write help files to provide help on a particular model. Such files must be written in either Windows help format (`.HLP`) or hypertext mark-up language.(`.HTML`). These files can then be accessed using the **Help | Model** command. This looks along the path for a `.HLP` or `.HTML` file with the same name as the current model. If this file exists, then it is opened to display the contents page. If not, *WinSolve* opens a dialog box to allow the user to select a path to the find the file.

GUIDE
$ Introductory Guide
# WHATSNEW4
k What's New in Version 4
Sec1
k Installing WinSolve
Sec2
k Getting Started
Sec3
k Models
Sec31
k Model Objects
Sec32
K Model Equation Files
Sec33
K Model Description Language
Sec34
k Variable Names
Sec35
K Functions;List of functions
Sec36
K Special Functions
Sec4
K Data formats;Formats
Sec5
k Editing Data
Sec51
k Editing observations
Sec52
k Drawing by hand
Sec53
k Projecting simple processes
Sec54
k Estimating Time Series Models
Sec6
k Assumptions Menu
Sec61
k Reordering model equations
Sec62
k Fixing and unfixing variables
Sec63
k Changing adjustment type
Sec64
k Switching between alternative equations
Sec65
K Terminal conditions
Sec66
K Setting the ragged edge
Sec7
k Solving the model
Sec71
K Model solution modes;Solution modes
Sec72
$ Dealing with the ragged edge
K Ragged edge
Sec73
k Skipping equations
Sec74
k Monitoring variables
Sec75
k Starting from the last solution point

Sec114
K Solving a linear-quadratic control game
Sec12
K Results: viewing;Viewing results
Sec121
K Switching views;Views: switching
Sec122
K Clipboard: copying to;Copying to the clipboard
Sec123
K Saving results
Sec13
K Command sequences: saving;Log files
Sec14
K Customising
Sec141
K Compiler options
Sec1411
K Case sensitivity
Sec1412
K Starred lines: ignoring; Ignoring starred lines
Sec142
K Graph colours
Sec143
K Locating applications;Applications: locating
Sec144
K Customising toolbar
Sec145
K Model help;Help on models