



IBM ILOG CPLEX V12.1

**IBM ILOG CPLEX for Microsoft
Excel User's Manual**

Copyright information

Copyright notice

© Copyright International Business Machines Corporation 1987, 2009.

US Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Trademarks

IBM, the IBM logo, ibm.com, WebSphere, ILOG, the ILOG design, and CPLEX are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at <http://www.ibm.com/legal/copytrade.shtml>

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

Table of contents

Setting up IBM ILOG CPLEX for Microsoft Excel.....	5
Overview.....	6
Registering the IBM ILOG CPLEX for Microsoft Excel add-in with Excel 2003.....	7
Registering the IBM ILOG CPLEX for Microsoft Excel add-in with Excel 2007.....	9
Defining an optimization model in Excel.....	11
Overview.....	12
Creating an optimization model in an Excel spreadsheet.....	15
The four sections of the spreadsheet.....	16
Data Section.....	18
Decision Variables Section.....	19
Objective Function Section.....	20
Constraints Section.....	21
Specifying the objective function cell and constraints.....	22
Excel formulas and cell reference notation.....	23
Modeling Tips.....	24
IBM ILOG CPLEX for Microsoft Excel user interface.....	25
The CPLEX Menu.....	26
CPLEX Model Information window.....	28
Procedures.....	35

Accessing IBM ILOG CPLEX for Microsoft Excel Help and documentation.....	37
Loading a problem model.....	38
Solving a problem model.....	39
Saving a problem model.....	43
Adding, updating, or removing a constraint.....	44
Stepping through MIP solutions.....	46
Analyzing solutions with IBM ILOG CPLEX for Microsoft Excel.....	47
Resetting your worksheet.....	49
CPLEX Model Parameter Settings.....	51
CPLEX Model Parameter Settings: General.....	52
CPLEX Model Parameter Settings: Algorithmic.....	55
CPLEX Model Parameter Settings: Polishing.....	60
CPLEX Advanced Parameters.....	63
IBM ILOG CPLEX for Microsoft Excel Examples.....	65
blend.xls.....	67
diet.xls.....	68
facility.xls.....	69
fixcost1.xls.....	70
foodmanufact.xls.....	71
inout1.xls.....	72
lpex1.xls.....	73
mipex1.xls.....	74
miqpex1.xls.....	75
qcpex1.xls.....	76
qpex1.xls.....	77
rates.xls.....	78
steel.xls.....	79
Using IBM ILOG CPLEX for Microsoft Excel with VBA.....	80
Glossary.....	81
Index.....	85

Setting up IBM ILOG CPLEX for Microsoft Excel

Presents the procedures for registering the IBM® ILOG® CPLEX® for Microsoft® add-in with both Excel 2003 and Excel 2007.

In this section

Overview

Presents an overview of how to start working with IBM ILOG CPLEX for Microsoft Excel.

Registering the IBM ILOG CPLEX for Microsoft Excel add-in with Excel 2003

Presents the procedure for registering the IBM ILOG CPLEX for Microsoft Excel XLL in Excel 2003.

Registering the IBM ILOG CPLEX for Microsoft Excel add-in with Excel 2007

Presents the procedure for registering the IBM ILOG CPLEX for Microsoft Excel XLL in Excel 2007.

Overview

There is no specific installation procedure for IBM® ILOG® CPLEX® for Microsoft® Excel. The program is contained in an XLL (Excel Add-In) file that is installed on your computer when you install CPLEX. You make the XLL available to Excel by registering it as an add-in program. The following sections show how to do this for both Excel 2003 (as part of Microsoft Office 2003) and Excel 2007 (as part of Microsoft Office 2007).

The XLL and all supporting files are located in the <cpalexinstalldir>\excel directory after you install CPLEX.

Licensing

There is no separate software license for IBM ILOG CPLEX for Microsoft Excel. If you have set up IBM ILOG License Manager with a valid license for CPLEX, the same license will work with CPLEX for Microsoft Excel.

If you have not yet set up your IBM ILOG CPLEX license, please consult the *Setting up licensing* section of the *IBM ILOG CPLEX Getting Started* manual and follow the procedures there.

Registering the IBM ILOG CPLEX for Microsoft Excel add-in with Excel 2003

Installing in Excel 2003

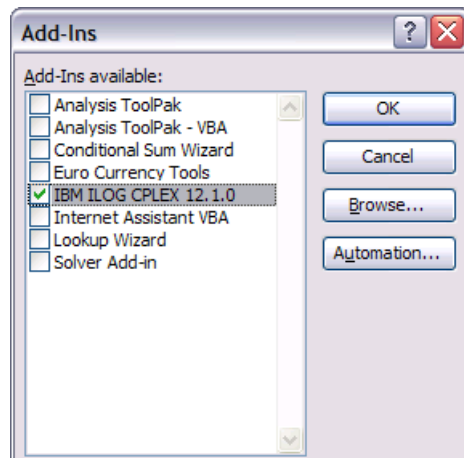
Before you can use IBM® ILOG® CPLEX® for Microsoft® Excel to solve problems, you must first register the XLL (Excel Add-In) file that contains the CPLEX solver as an add-in for Microsoft Excel. This section contains the procedure for doing this with Excel 2003.

1. Locate the `cplex121.xll` file on your computer.

By default this file is installed in `<cplexinstalldir>\excel`.

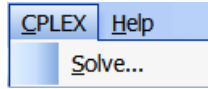
Note: We recommend that you do not move this XLL file from its default location, because there are several support files in the same directory that are required to make context-sensitive Help function properly.

2. Open Excel 2003. In the main menu, choose **Tools > Add-Ins**.
3. On the Add-Ins window, click **Browse** and navigate to the location of the `cplex121.xll` file, select it, and click **OK**.
4. When you return to the Add-Ins window, verify that **IBM ILOG CPLEX 12.1.0** add-in has been added to the list and that its check box is checked.



5. Finally, click **OK** to close the Add-Ins window and return to Excel 2003.

In Excel, verify that a **CPLEX** menu has been added to the items in your main menu bar.



At this point, the XLL for IBM ILOG CPLEX for Microsoft Excel is registered as an add-in of Excel 2003.

Registering the IBM ILOG CPLEX for Microsoft Excel add-in with Excel 2007


Installing in Excel 2007

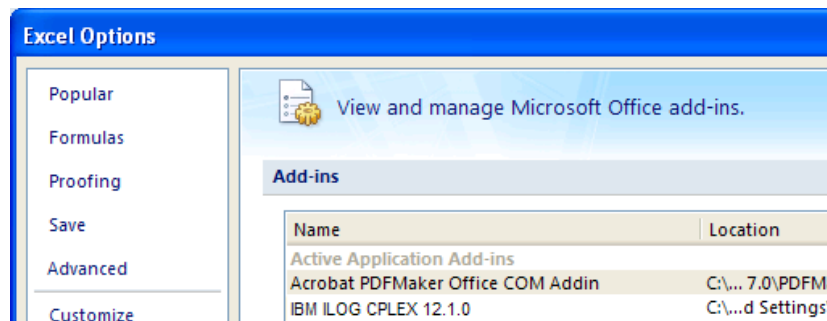
Before you can use IBM ILOG CPLEX for Microsoft Excel to solve problems, you must first register the XLL (Excel Add-In) file that contains the CPLEX solver as an add-in with Microsoft Excel. This section contains the procedure for doing this with Excel 2007.

1. Locate the `cplex121.xll` file on your computer.

By default this file is installed in `<cplexinstalldir>\excel`.

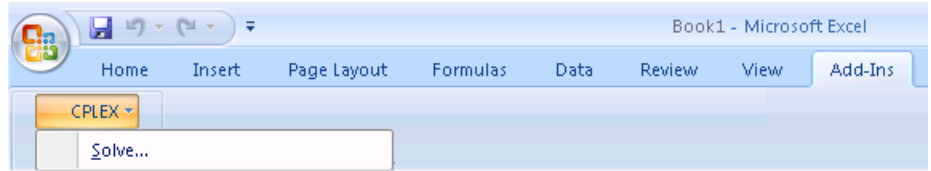
Note: We recommend that you do not move this XLL file from its default location, because there are several “support” files in the same directory that are required to make context-sensitive Help function properly.

2. Open Excel 2007. Click the round **Office** button  in the upper left of the Excel window.
3. In the pop-up menu, click the **Excel Options** button at the bottom right of the window.
4. When the Excel Options window appears, click **Add-Ins** in the left pane to display that wizard in the right pane.
5. At the bottom left of the Add-Ins pane, choose **Excel Add-Ins** from the **Manage** drop-down list and click the **Go** button beside it.
6. On the popup window, click **Browse** and navigate to the location of the `cplex121.xll` file, select it, and click **OK**.
7. When you return to the Add-Ins pane of the Excel Options window, verify that **CPLEX 12.1.0 Excel Solver** has been added to the list of available add-ins.



8. Finally, click **OK** to close the Excel Options window and return to Excel 2007.

In Excel, click the **Add-Ins** tab on the Ribbon Bar to verify that a **CPLEX** menu is displayed below it.



At this point, the XLL for IBM ILOG CPLEX for Microsoft Excel is registered as an add-in of Excel 2007.

Defining an optimization model in Excel

Presents an overview of how to enter an optimization problem into Excel so that it can be solved by IBM ILOG CPLEX for Microsoft Excel.

In this section

Overview

An overview of the types of problems the IBM ILOG CPLEX for Microsoft Excel add-in allows you to solve.

Creating an optimization model in an Excel spreadsheet

Shows how to enter an optimization problem into a spreadsheet so that it can be solved by IBM ILOG CPLEX for Microsoft Excel.

Specifying the objective function cell and constraints

Describes how you define the objective function cell and the constraints for your model before you solve it.

Excel formulas and cell reference notation

Presents a recap of how cell references are used in cells in Excel spreadsheets and in the CPLEX Model Information window.

Modeling Tips

Tips on how to model your optimization problems for IBM ILOG CPLEX for Microsoft Excel.

Overview

IBM® ILOG® CPLEX® for Microsoft® Excel is an extension to IBM ILOG CPLEX that allows you to use Microsoft Excel format to define your optimization problems and solve them. Thus a business user or educator who is already familiar with Excel can enter their optimization problems in that format and solve them, without having to learn a new interface or command language.

CPLEX is a tool for solving linear optimization problems, commonly referred to as Linear Programming (LP) problems, of the form:

$$\begin{aligned} &\text{Maximize (or Minimize)} && c_1 x_1 + c_2 x_2 + \dots + c_n x_n \\ &\text{subject to} && bl_1 \leq a_{11} x_1 + a_{12} x_2 + \dots + a_{1n} x_n \leq bu_1 \\ & && bl_2 \leq a_{21} x_1 + a_{22} x_2 + \dots + a_{2n} x_n \leq bu_2 \\ & && \dots \\ & && bl_m \leq a_{m1} x_1 + a_{m2} x_2 + \dots + a_{mn} x_n \leq bu_m \\ &\text{with these bounds} && l_1 \leq x_1 \leq u_1 \\ & && \dots \\ & && l_n \leq x_n \leq u_n \end{aligned}$$

where the upper bounds bu_i and u_i and the lower bounds bl_i and l_i may be positive infinity, negative infinity, or any real number.

The elements of data you provide as input for this LP problem are:

$$\begin{aligned} \text{Objective function coefficients} &&& c_1, c_2, \dots, c_n \\ \text{Constraint coefficients} &&& a_{11}, a_{12}, \dots, a_{1n} \\ &&& \dots \\ &&& a_{m1}, a_{m2}, \dots, a_{mn} \\ \text{Upper and lower bounds on constraints} &&& bu_1, bu_2, \dots, bu_m \text{ and } bl_1, bl_2, \dots, bl_m \\ \text{Upper and lower bounds on variables} &&& u_1, u_2, \dots, u_n \text{ and } l_1, l_2, \dots, l_n \end{aligned}$$

The optimal solution that CPLEX computes and returns is:

$$\text{Variables } x_1, x_2, \dots, x_n$$

IBM ILOG CPLEX for Microsoft Excel can also solve several extensions to LP:

Creating an optimization model in an Excel spreadsheet

Shows how to enter an optimization problem into a spreadsheet so that it can be solved by IBM ILOG CPLEX for Microsoft Excel.

In this section

The four sections of the spreadsheet

An overview of the process of creating an optimization model in Excel.

Data Section

Describes the Data Section of an optimization model spreadsheet.

Decision Variables Section

Describes the Decision Variables Section of an optimization model spreadsheet.

Objective Function Section

Describes the Objective Function Section of an optimization model spreadsheet.

Constraints Section

Describes the Constraints Section of an optimization model spreadsheet.

The four sections of the spreadsheet

The first step of solving a linear optimization model in a spreadsheet is to convert the formulation of the problem that you would see on paper into an Excel spreadsheet file that can be solved by a suitable add-in solver like IBM® ILOG® CPLEX® for Microsoft® Excel.

The format that we recommend for entering your problems into Excel is that recommended by Dimitris Bertsimas and Robert M. Freund in their book *Data, Models, and Decisions: The Fundamentals of Management Science*. They recommend a standard format that helps you to keep your model organized, logical, transparent, and easy to work with.

Important: All cells used in specifying a model (variables, constraints, bounds, and objective) must belong to the worksheet for which the model is defined.

This is accomplished by organizing the spreadsheet into four discrete areas:

- ◆ The **Data Section**
- ◆ The **Decision Variables Section**
- ◆ The **Objective Function Section**
- ◆ The **Constraints Section**

This format is used in the sample spreadsheets provided as examples in the IBM ILOG CPLEX for Microsoft Excel distribution. A sample spreadsheet (the `diet.xls` example) is shown below, with the four areas marked on it:

Objective Function formula
(shown because cell is selected)

B22		=SUMPRODUCT(B5:J5,B19:J19)									
	A	B	C	D	E	F	G	H	I	J	K
	1 Diet Selection Problem: Choose the amount of foods in a diet to meet nutritional requirements at the lowest cost.										
	2										
	3 Problem Data										
	4 Foods	Food1	Food2	Food3	Food4	Food5	Food6	Food7	Food8	Food9	
	5 Food Cost	1.84	2.19	1.84	1.44	2.29	0.77	1.29	0.6	0.72	
	6 Food Supply	10	10	10	10	10	10	10	10	10	
	7										
	8										
	9 Nutrients	Nutrients per Food Unit									
	10 Carbohydrate	510	370	500	370	400	220	345	110	80	
	11 Fat	34	35	42	38	42	26	27	12	20	
	12 Protein	28	24	25	14	31	3	15	9	1	
	13 Calcium	15	15	6	2	8	0	4	10	2	
	14 Vitamin B	6	10	2	0	15	15	0	4	120	
	15 Vitamin C	30	20	25	15	15	0	20	30	2	
	16 Vitamin D	20	20	20	10	8	2	15	0	2	
	17										
	18 The variables are the amounts of each food to select for the diet.										
	19 Amount of Food										
	20										
	21 The Diet Cost cell contains the objective function, the sum of the cost of each food times the amount of each food.										
	22 Diet cost	0									
	23										
	24 The constraints on the diet are to meet minimum and maximum nutritional requirements. Each constraint states										
	25 that the sum of the amount of the nutrient in the food times the amount of each food must be more than the										
	26 minimum nutritional requirement but less than the maximum requirement.										
	27 Nutrients	Minimum	Provided	Maximum							
	28 Carbohydrate	2000	0	9999							
	29 Fat	350	0	375							
	30 Protein	55	0	9999							
	31 Calcium	100	0	9999							
	32 Vitamin B	100	0	9999							
	33 Vitamin C	100	0	9999							
	34 Vitamin D	100	0	9999							
	35										

Data Section (rows 3-16)

Decision Variables Section (rows 18-19)

Objective Function Section (rows 21-22)

Constraints Section (rows 24-34)

Each of these sections are described in more detail in the following sections.

Data Section

The data section provides the numeric data that is used in the optimization model. The cells in this section can also contain formulas that use raw data to calculate and generate other data that is then used to solve the objective function.

Whenever possible, try to arrange your data into organized tables, because that will make it easier to enter variables that relate to the data in the Decision Variables Section. For example, in the `diet.xls` screenshot shown earlier, each of the variables entered refers to and modifies the data contained in the same column. Thus the decision variable in column **D** of the **Amount of Food** row refers to the **Food3** data in that same column.

Decision Variables Section

This section contains the cells that represent the decision variables used to solve the problem. They are the unknowns to be solved for by CPLEX, and they will be filled with the optimal variable values found for each of them when a solution to the model is found.

Objective Function Section

This section contains the cells and formulas that are required to compute the objective function value. Although several cells may be used to perform intermediate calculations, the final objective function *must be contained in a single cell*.

The formula area of the objective function cell contains an Excel formula that expresses the function of the data in the Data Section and the still unknown values in the Decision Variables Section.

The objective formula or function and the formulas and functions used to compose the objective function may only have terms which are linear or quadratic in the decision variables. In addition, for objective functions containing quadratic terms, the function or formula must be convex if the function is to be minimized or the function must be concave if the function is to be maximized.

See the section *Solving problems with a quadratic objective (QP)* in the *IBM ILOG CPLEX User's Manual*.

Specifying the objective function cell and constraints

After you have finished entering the Data Section, the Decision Variables Section, the Objective Function Section and Constraints Section of your model, the last step that you need to perform before you can solve the model is to specify three additional types of information on the CPLEX **Model Information** window. You access this window by choosing **Solve** from the CPLEX menu.

On the CPLEX **Model Information** window, you:

- ◆ Specify the location of the objective function cell in the **Objective Function** field.
- ◆ Specify the location of the cell or range of cells used as decision variables for the problem in the **Variables** section.
- ◆ Enter the constraints to be used in the solution in the **Constraints** section.

Information on how to do this can be found in the *IBM ILOG CPLEX for Microsoft Excel user interface* and *Procedures* sections of this manual.

Excel formulas and cell reference notation

References to other spreadsheet cells in the formula area of an Excel cell are, by default, *relative*. That is, they identify the position relative to the current cell of the cell or cells being referred to.

For example, if you see **B8** written in the formula area of cell **D7**, this is a reference to the cell that is two columns to the left and one row below the current one (**D7**). If you move or copy this formula one column to the right into cell **E7**, the reference to **B8** automatically becomes a reference to **C8** because this is two cells to the left and one row below the cell **E7**.

However, formulas can also contain *absolute* cell references. If you prefix either the letter or the digit of a cell reference with a dollar sign (\$) it means that this information is absolute and will not change depending on the cell in which it is used.

The possible combinations are

- ◆ cell reference **B8** has relative row and relative column
- ◆ cell reference **\$B8** has absolute column and relative row
- ◆ cell reference **B\$8** has relative column and absolute row
- ◆ cell reference **\$B\$8** has absolute row and absolute column

IBM ILOG CPLEX for Microsoft Excel has no notion of what constitutes the "current cell", so relative cell references do not make sense to it when it solves. **All** references need to be prefixed with a dollar sign (\$) for the column and the row part of each cell reference.

When you type cell references into the CPLEX **Model Information** window, relative cell references are silently converted to absolute cell references by adding the dollar signs (\$).

Named cells and named ranges

IBM ILOG CPLEX for Microsoft Excel supports the convention of *names*, which are a word or string of characters used to represent a cell, range of cells, formula, or constant value, and that can be used in other formulas.

Thus you can use easy-to-understand names, such as **Nutrients**, to refer to hard-to-understand ranges, such as **B4:J15** or **IncreasedProtein** to refer to a constraint. You can then substitute these names in formulas for the range of cells or constraint.

The sample problem `fixcost1.xls` shows an example of a name being used in a formula.

Modeling Tips

The only explicitly supported functions are `SUM`, `SUMPRODUCT`, and `INT`. For all other functions you use on a spreadsheet, IBM® ILOG® CPLEX® for Microsoft® Excel probes whether these functions are linear or quadratic.

Probing means that it evaluates the function at a list of carefully chosen points to calculate coefficients of a linear or quadratic representation of that function. If this probing yields a linear or quadratic function, the calculated function definition is checked against a list of control points to verify the result.

If either coefficients cannot be calculated or the validation of calculated coefficients fails, CPLEX refuses to optimize the model. In particular, indirect referencing using function `INDIRECT()` is not supported.

Excel treats non-numeric values in `SUM` and `SUMPRODUCT` as `NAN` instead of 0. This will produce unexpected results if you rely on these values being interpreted as 0.

Constant logical values in `SUM` or `SUMPRODUCT` are treated as `NAN` instead of 0 and 1. This will produce unexpected results if you rely on these values being interpreted as 0.

CPLEX does not prohibit complex references (unions) in constraints but it does not handle them correctly, so they should not be used.

Integrality or binary constraints on constant cells are not allowed. If you state such a constraint you will see a debug message error and the constraint will be ignored.

Important: All cells used in specifying a model (variables, constraints, bounds, and objective) must belong to the worksheet for which the model is defined.

IBM ILOG CPLEX for Microsoft Excel user interface

Describes the menu items and windows used to solve optimization models with IBM ILOG CPLEX for Microsoft Excel.

In this section

The CPLEX Menu

Describes the CPLEX menu that is added to Excel's main menu when IBM ILOG CPLEX for Microsoft Excel is registered as an add-in program

CPLEX Model Information window

Describes the CPLEX Model Information window and its functions.

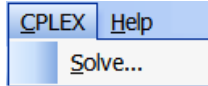
The CPLEX Menu

When you register the IBM® ILOG® CPLEX® for Microsoft® Excel XLL as an add-in program, a new menu is added to Excel's main menu in Excel 2003 or a new CPLEX tab is added to the Add-Ins tab on the Ribbon Bar in Excel 2007. You use the entry on this menu to solve optimization models you have entered into Excel, and to perform other tasks.

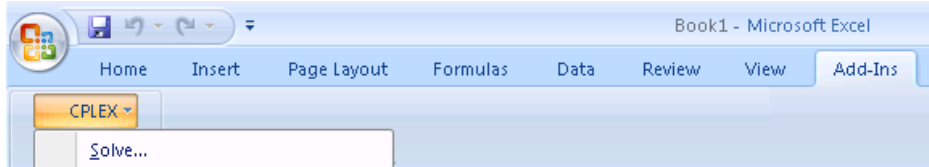
Note: See *Setting up IBM ILOG CPLEX for Microsoft Excel* for information on registering the IBM ILOG CPLEX for Microsoft Excel add-in.

See *Defining an optimization model in Excel* for information on how to prepare your optimization models in spreadsheet format so that they can be solved by IBM ILOG CPLEX for Microsoft Excel.

The CPLEX menu is shown below. Its individual options are covered in the following sections.



Excel menu in Excel 2003

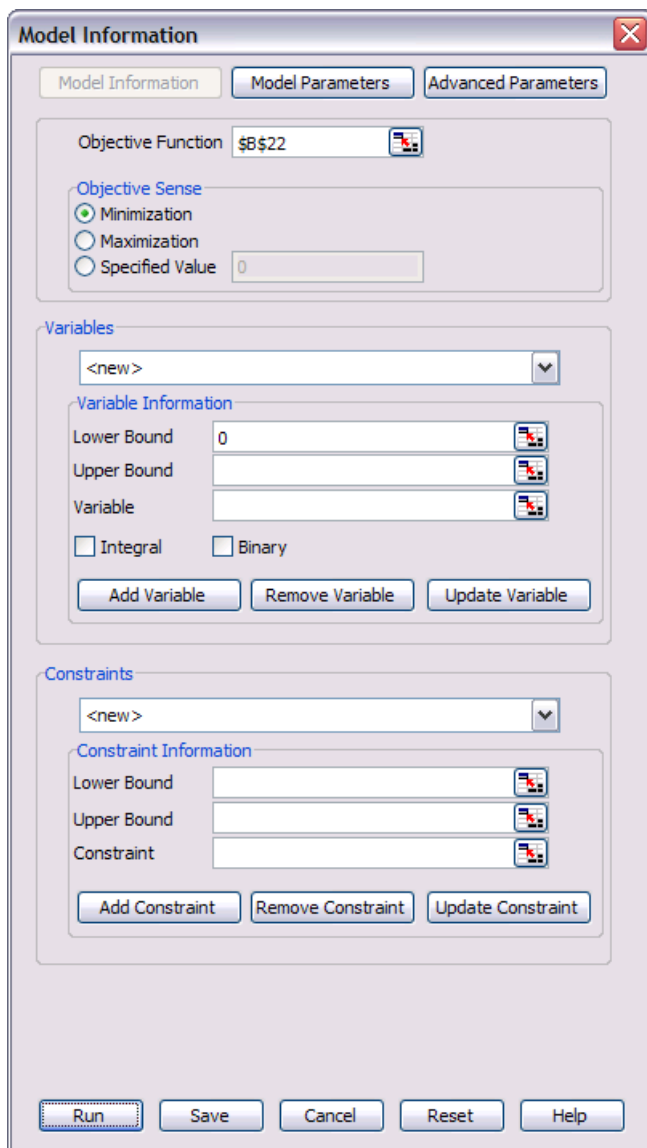


Excel menu in Excel 2007

The instructions in this section will all be presented in the form **CPLEX > menu_option**, to avoid having to repeat the differences in how to access the CPLEX menu in Excel 2003 and Excel 2007.

The 'Solve' option

Choose **CPLEX > Solve** to display the **CPLEX Model Information** window:

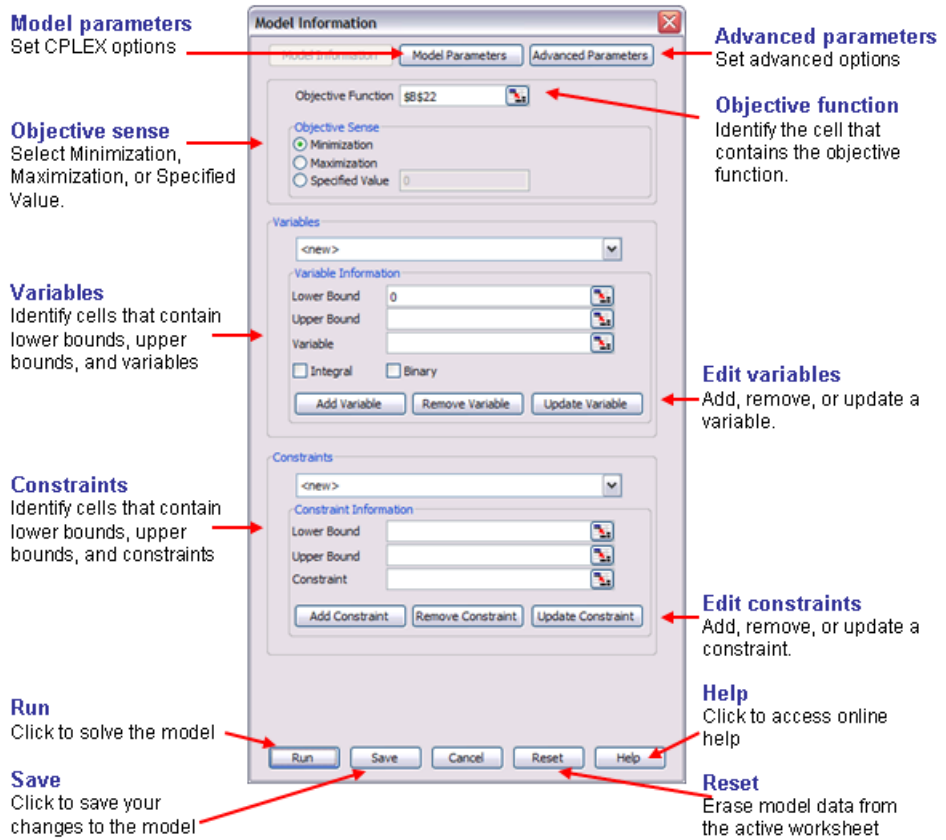


You use this window to solve your optimization model and to enter and edit information when performing “what if” calculations based on the model. More information on this window is presented in the *CPLEX Model Information window* section.

CPLEX Model Information window

Specifying your model


The following figure shows the CPLEX Model Information window, in which you specify the objective function, variables, and constraints in your optimization model:




The different fields and controls on the CPLEX Model Information window are described briefly in the sections that follow, using the example file `diet.xls` as a point of reference. Step-by-step instructions for common tasks are presented in the *Procedures* section.

Objective Function

Use the **Objective Function** field to identify the cell that contains the objective function for the model.

You can type the cell reference directly in the field or click the **Edit** button  to the right of the field and enter your information in the edit window that appears:



You can also select the objective function cell by pointing to it with the mouse. When you have finished, return to the main **Model Information** window by clicking the **Update** button  to the right of the edit window.

In the case of the `diet.xls` example, the cell reference for the Objective Function is **\$B\$22**; the *Diet cost* cell.

Objective Sense

Use the **Objective Sense** radio buttons to specify whether you want to minimize or maximize the objective function by selecting **Minimization**, **Maximization**, or **Specified Value**.

If you select **Specified Value**, enter a numeric value in the field next to it to force a solution that contains the value entered as the result. You can use this feature to calculate the decision variables of the problem “backwards” from the solution value entered, to perform “what-if” calculations.

In the case of the `diet.xls` example, the Objective Sense is **Minimization**, as the goal is to identify a diet that satisfies specific nutritional requirements at the lowest cost.

Set Decision Variables

Use the **Variables** section to define the cells that contain the decision variables or changing data for the model.

To add a variable

1. Enter a cell, a range of cells, or a value in the **Lower Bound** field.
For `diet.xls`, you enter the lower bound of 0, as the diet can contain none of any given food.
2. Enter a cell, a range of cells, or a value in the **Upper Bound** field.
For `diet.xls`, the upper bound is the cell range **Food Supply**, as the diet can contain no more than the available supply of a given food.
3. Enter a cell reference in the **Variable** field.
For `diet.xls`, the **Variable** is the cell range **Amount of Food**, which will contain the quantities of each food in the diet.
4. Click **Add Variable**.

You can add additional variables by following the same steps. As you add variables, they appear in the **Variables** list box. You can remove or modify a variable by selecting its entry in the list.

Specify Constraints

Use the **Constraints** section to define the constraints for the model.

For more information on constraints, see *Adding, updating, or removing a constraint*.

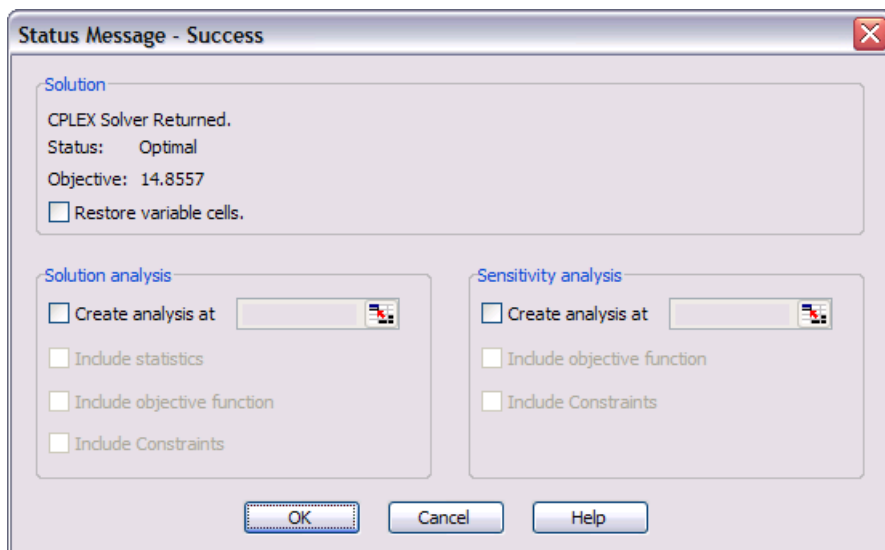
To add a constraint

1. Enter a cell, a range of cells, or a value in the **Lower Bound** field.
For `diet.xls`, the lower bound is the cell range in the **Minimum** column of the **Nutrients** table, representing the lowest acceptable amount of a given nutrient in the diet.
2. Enter a cell, a range of cells, or a value in the **Upper Bound** field.
For `diet.xls`, the upper bound is the cell range in the **Maximum** column of the **Nutrients** table, representing the highest acceptable amount of a given nutrient in the diet.
3. Enter a cell reference in the **Constraint** field.
For `diet.xls`, the **Constraint** is the cell range in the **Provided** column of the **Nutrients** table, which will contain the quantities of each nutrient in the diet.
4. Click **Add Constraint**.

You can add additional constraints by following the same steps. As you add constraints, they appear in the **Constraints** list box. You can remove or modify a constraint by selecting its entry in the list.

Run

Click the **Run** button to solve the problem.



More information on this window and its options is contained in the *Procedures* section, especially the *Solving a problem model* section.

Save button

Click to save any changes you've made on the **CPLEX Model Information** window.

Cancel button

Click this button to close the **CPLEX Model Information** window without solving the problem.

Reset button

Click this button to erase all model data from the active worksheet and close the **CPLEX Model Information** window.

The **Reset** button removes the following information from the **CPLEX Model Information** window:

- ◆ The contents of the **Objective Function** field. The objective function cell must be reentered before the problem model can be solved.
- ◆ The cells specified as the decision variable cells in the **Variables** field. This information must be reentered before the problem model can be solved.
- ◆ Any existing constraints from the **Constraints** area. Any constraints that you wish to apply must be reentered before the problem model can be solved.

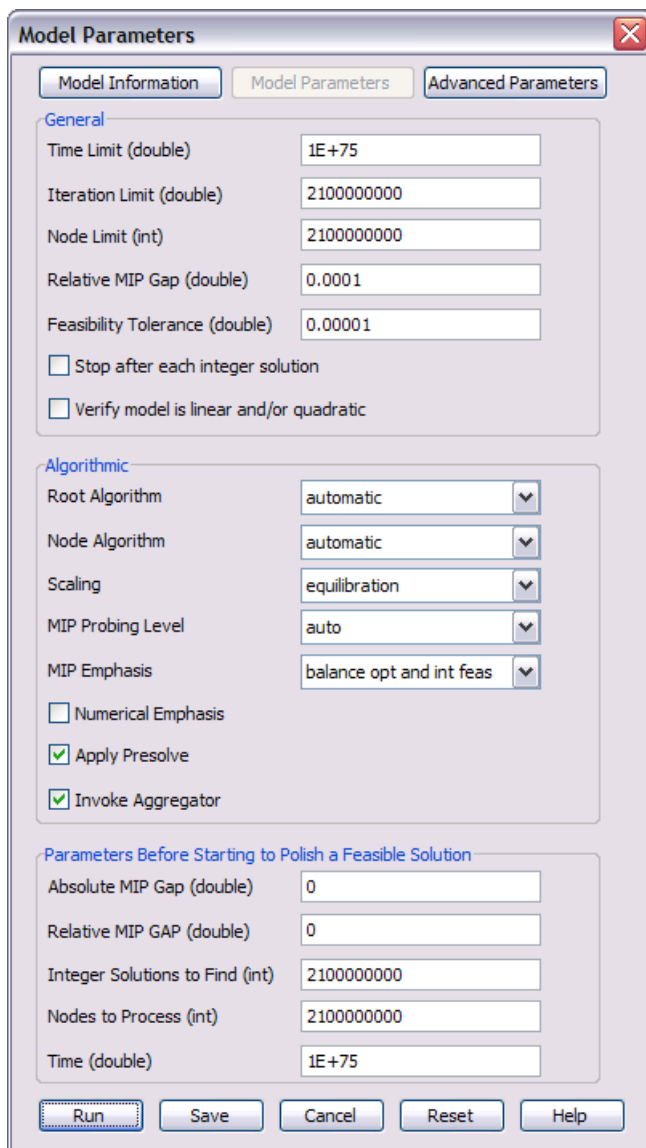
The **Reset** button also:

- ◆ Changes all settings in the **Model Parameters** window to their default values.
- ◆ Removes all parameters specified in the **Advanced Parameters** window. User provided parameters must be specified again to take effect.

Warning: This button removes the objective function cell, information on variables, and information on constraints from the **Model Information** window.

Model parameters button

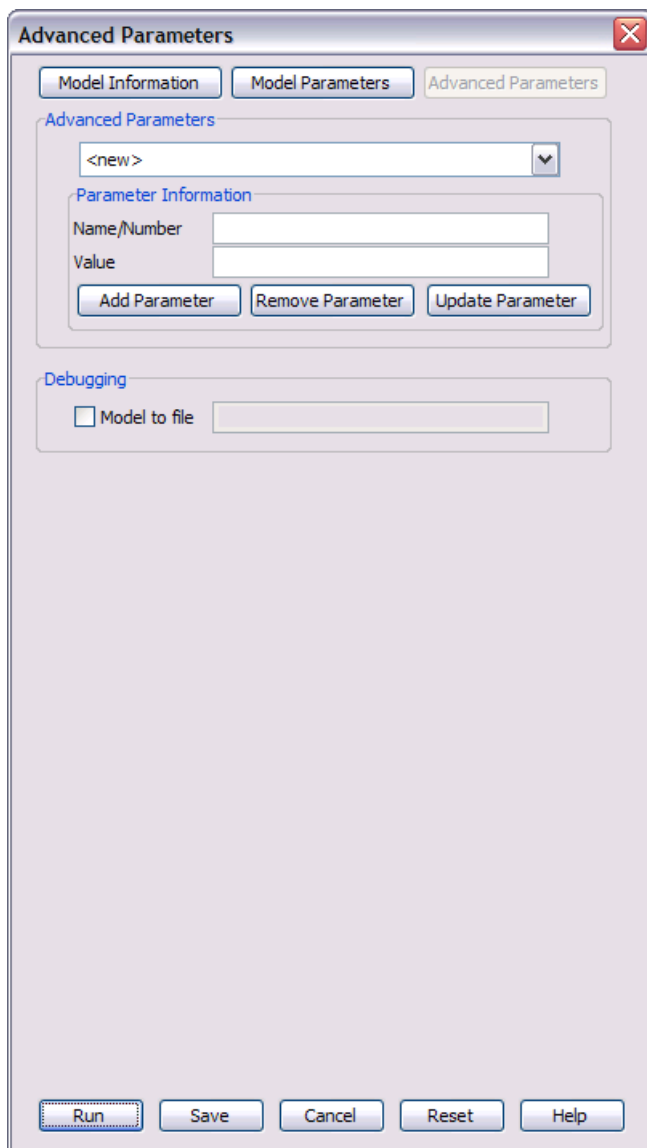
Click this button to display the CPLEX **Model Parameters** window to set CPLEX parameters and other options. The window is displayed below with its **Main** tab active:



For more information on this window and its options, see *CPLEX Model Parameter Settings*.

Advanced parameters button

Click this button to display the CPLEX **Parameter Settings** window to set CPLEX parameters and other options. The window is displayed below with its **Main** tab active:



For more information on this window and its options, see *CPLEX Advanced Parameters*.

Help button

Click this button to display context-sensitive Help for this window and its functions.

Procedures

This section contains step-by-step procedures for common tasks that you perform using IBM ILOG CPLEX for Microsoft Excel.

In this section

Accessing IBM ILOG CPLEX for Microsoft Excel Help and documentation

How to view the IBM ILOG CPLEX for Microsoft Excel online help.

Loading a problem model

Shows how to load a problem model into Excel for solving by IBM ILOG CPLEX for Microsoft Excel.

Solving a problem model

Presents instructions on how to solve a problem model using IBM ILOG CPLEX for Microsoft Excel.

Saving a problem model

How to save a problem model that has been solved using IBM ILOG CPLEX for Microsoft Excel.

Adding, updating, or removing a constraint

Contains procedures for how to add, change, or delete a constraint in IBM ILOG CPLEX for Microsoft Excel.

Stepping through MIP solutions

Instructions for how to step through solutions found when solving MIP problem models.

Analyzing solutions with IBM ILOG CPLEX for Microsoft Excel

Contains procedures for how to analyze your solution.

Resetting your worksheet

How to reset the problem model and remove all model data.

CPLEX Model Parameter Settings

How to set and use CPLEX parameters and other options when solving your problem models.

Accessing IBM ILOG CPLEX for Microsoft Excel Help and documentation

To view online Help

- ◆ In Excel, press the **Help** button on any of the IBM ILOG CPLEX for Microsoft Excel popup windows.

The *IBM ILOG CPLEX for Microsoft Excel Online Help* system will open in its own window, displaying context-sensitive Help for the window you are on.

Note: In Excel 2007, you can navigate to other portions of the Help using the table of contents in the left panel. In Excel 2003 the table of contents does not appear when you invoke Help this way.

To be able to see the entire contents of the IBM ILOG CPLEX for Microsoft Excel Online Help system, launch the `<cplexinstalldir>\excel\cplex121.chm` file using Windows Explorer. You can use the Windows Start Menu command **Programs>IBM ILOG>IBM ILOG CPLEX 12.1>Content Folder** command to navigate to this file.

Loading a problem model

To load an optimization model that has been previously created and saved

- ◆ Double-click on the `xls` file that contains the model to open it in Excel

OR

with Excel already open, select **File > Open** from the main menu, navigate to the file that contains the model, and click **Open**.

Solving a problem model

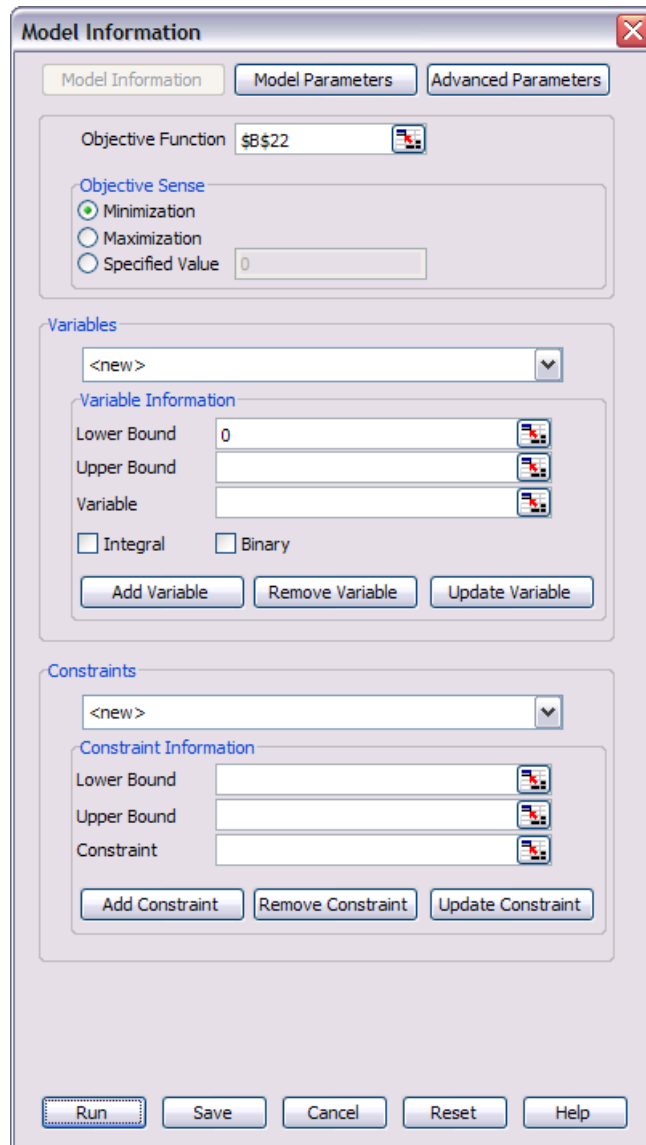
To solve an optimization model that has been loaded into Excel

1. In Excel 2003, select **CPLEX > Solve** from the main menu.


OR

In Excel 2007, select **Add-Ins > CPLEX > Solve**

The CPLEX Model Information window is displayed:



2. Verify that the cell displayed in the **Objective Function** field is correct.

If it is not correct, click the Edit button  to the right of the field and enter the location of the cell that contains the objective function in the edit window that appears.

3. In the **Objective Sense** section, select either the **Maximization** or **Minimization** radio buttons, depending on whether you want to maximize or minimize the formula in the objective function.

Note: Alternatively, you can select **Specified Value** and enter a numeric value in the field next to it to force a solution that contains the value entered as the result. You can use this feature to calculate the decision variables of the problem “backwards” from the solution value entered, to perform “what-if” calculations

4. Verify that the cells or ranges of cells displayed in the **Variable Information** section is correct.

If they are not correct, click the Edit button  to the right of the field and enter the location of the variable cells in the edit window that appears.

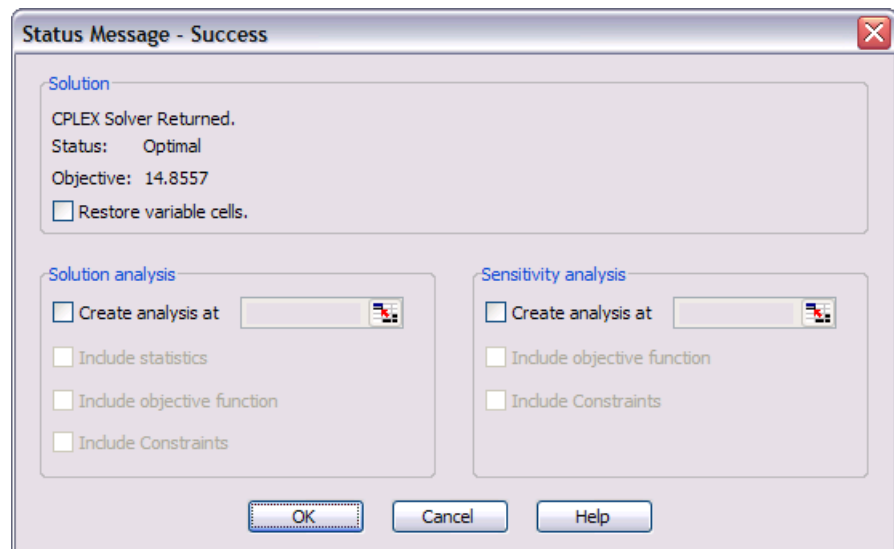
5. Verify that the constraint or constraints displayed in the **Constraints** section are correct, and are the constraints that you want to apply for this solve.

If they are not correct, click the Edit button  to the right of the field and enter the location of the variable cells in the edit window that appears.

6. When all of the information in each of these fields is correct, click the **Run** button.

Note: Depending on the size and complexity of the model, and the **Time Limit** parameter set on the CPLEX Model Parameters window, solving can take some time.


Once the solve process has completed, the Solver Results window is displayed:



The status message at the upper left of the window indicates the status of the solve operation.

On this window, you also have the following options

- ◆ Click **Restore variable cells** to revert the changed values in the spreadsheet to their original values when you click **OK**.
- ◆ Select **Create analysis at** for **Solution analysis** and/or **Sensitivity analysis** to analyze the solution to your problem.

You need to identify where to generate the analysis by clicking the Edit button .

For more information on analysis, see *Analyzing solutions with IBM ILOG CPLEX for Microsoft Excel*.

- ◆ Click **Cancel** to return to your worksheet.

Saving a problem model

To save an optimization model that has been solved

1. **Save A Copy Method** — To save a **copy** of the solved spreadsheet that contains the calculated solution, changed variables, and any generated reports, select **File > Save As** from the Excel main menu and specify a new name for the saved spreadsheet.

This method saves the solved spreadsheet while preserving the original spreadsheet under its original filename.

2. **Overwrite Method** — To **overwrite** the original spreadsheet, saving the calculated solution, changed variables, and any generated reports, select **File > Save** from the Excel main menu.

This method saves the solved spreadsheet under its original filename, but does not preserve the original spreadsheet.

Adding, updating, or removing a constraint

To add a constraint

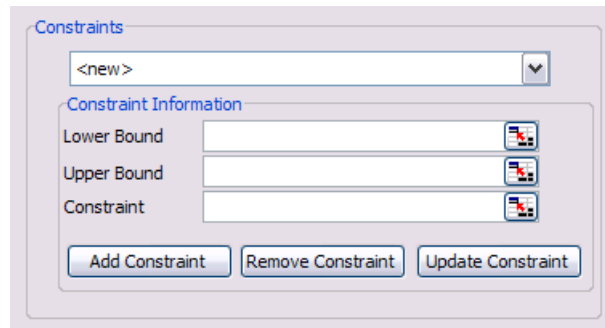
1. With the problem model open in Excel 2003, select **CPLEX > Solve**.

OR

In Excel 2007, select **Add-Ins > CPLEX > Solve**

The CPLEX **Model Information** window is displayed.

2. On the CPLEX **Model Information** window, enter the information for your constraint in the **Constraints** section:



The screenshot shows the 'Constraints' section of the CPLEX Model Information window. At the top, there is a dropdown menu with '<new>' selected. Below this is the 'Constraint Information' section, which contains three input fields: 'Lower Bound', 'Upper Bound', and 'Constraint'. Each field has a small icon to its right. At the bottom of the section are three buttons: 'Add Constraint', 'Remove Constraint', and 'Update Constraint'.


- ◆ Enter a cell, a range of cells, or a value in the **Lower Bound** field.
For `diet.xls`, the lower bound is the cell range in the **Minimum** column of the **Nutrients** table, representing the lowest acceptable amount of a given nutrient in the diet.
- ◆ Enter a cell, a range of cells, or a value in the **Upper Bound** field.
For `diet.xls`, the upper bound is the cell range in the **Maximum** column of the **Nutrients** table, representing the highest acceptable amount of a given nutrient in the diet.
- ◆ Enter a cell reference in the **Constraint** field.
For `diet.xls`, the **Constraint** is the cell range in the **Provided** column of the **Nutrients** table, which will contain the quantities of each nutrient in the diet.

3. Click **Add Constraint**.

To change an existing constraint

1. With the problem model open in Excel, select **CPLEX > Solve** from the main menu to display the CPLEX **Model Information** window.
2. On the CPLEX Model Information window, select the constraint you want to change from the **Constraints** list box.

3. Edit the cell or cells affected by the constraint using the **Lower Bound**, **Upper Bound**, or **Constraint** fields.

You can edit the cell references directly or by clicking the Edit button  to the right of the field

Note: If you have specified a **name** for the cell or range of cells, you can enter that name instead of absolute cell references.

4. Click **Update Constraint**.

To delete an existing constraint

1. With the problem model open in Excel, select **CPLEX > Solve** from the main menu to display the CPLEX Model Information window.
2. On the CPLEX Model Information window, select the constraint you want to delete from the **Constraints** list box.
3. Click **Remove Constraint**.

Stepping through MIP solutions

This option pauses the solver each time a new solution is found for MIP problems. MIP stands for Mixed Integer Program. This is what optimization problems which have integer or binary constraints are called. For continuous models this option is ignored.

MIP problems often take some time to reach an optimal solution, so this option allows you to stop at each iterative solution and determine whether it is acceptable. If so, you can stop the solve at this point and accept the solution.

To step through the feasible solutions to a MIP problem model

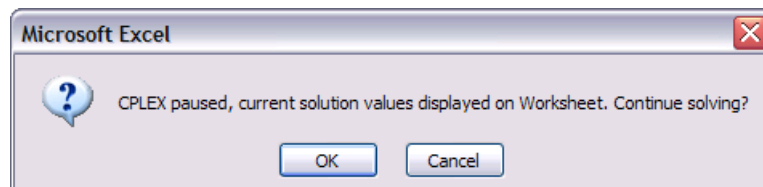
1. On the CPLEX **Model Information** window, click **Model Parameters** to open the CPLEX **Model Parameters** window.
2. In the **General** section, select the **Stop after each integer solution** check box.

This tells the solver to stop each time a new solution is found for MIP problems.

Note: This option is only effective on MIP problems.

3. Click **Run**.

When the CPLEX solver finds a solution, the solver pauses and displays the following dialog:



At this point you can examine the current solution in the spreadsheet.

4. To continue solving, press **OK**.

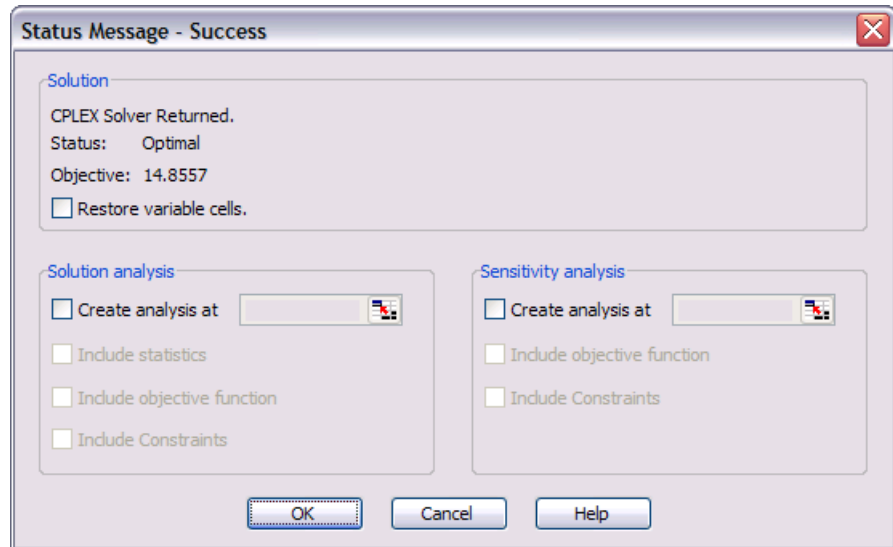
OR



To halt the solve and accept this solution, press **Cancel**.

Analyzing solutions with IBM ILOG CPLEX for Microsoft Excel

To analyze the solution to a problem model

1. After you have solved your problem model using IBM ILOG CPLEX for Microsoft Excel, you can generate analyses for this solution from the **Status** window:



2. In the **Status** window, check the boxes for the type of analysis you want to perform:
 - ◆ In the **Solution analysis section**, check **Create analysis at** and use the Edit button  to identify a location in your worksheet or another worksheet to contain the results.
 - ◆ In the **Sensitivity analysis section**, check **Create analysis at** and use Edit button  to identify a location in your worksheet or another worksheet to contain the sensitivity analysis information.

Note: **Sensitivity** information is not available for models with binary and integer constraints. When you solve such a model, the check box for this analysis will be inactive.

3. Click **OK**.

The information you have selected is generated and displayed in the location you specified.

If you save the spreadsheet at this point, the reports will be saved as well.

The **Status** field in the solution analysis will contain the value #N/A for MIPs and problems with quadratic constraints or objectives. This is because the status value is taken from the basis, a solution artifact that is computed only for LPs, that is, problems without integer or binary variables.

Examples of the analysis are shown in the following sections.

A sample solution analysis, from the diet.xls example

	A	B	C	D	E	F	G
1	Model Sheet		Sheet1				
2	Solution Status		Optimal				
3	Solution Type		basic				
4	Solution Method		dual simplex				
5	Iterations		5				
6							
7							
8	Cell	Objective Name	Value	Sense			
9	\$B\$22		14.8557377	Minimize			
10							
11							
12	Cell	Constraint Name	Lower Bound	Value	Upper Bound	Tight Bound	
13	\$C\$28		2000	3965.368852	9999	#N/A	
14	\$C\$29		350	350	375	Lower	
15	\$C\$30		55	172.0286885	9999	#N/A	
16	\$C\$31		100	100	9999	Lower	
17	\$C\$32		100	132.2131148	9999	#N/A	
18	\$C\$33		100	234.2213115	9999	#N/A	
19	\$C\$34		100	100	9999	Lower	
20							

A sample sensitivity analysis, from the diet.xls example

	A	B	C	D	E	F	G	H	I	J
1				Objective Coefficient Range						
2	Cell	Variable Name	Reduced Cost	Lowest	Current	Highest				
3	\$B\$19		0	1.373846154	1.84	1.860746004				
4	\$C\$19		0	1.867704918	2.19	1E+20				
5	\$D\$19		0	1.820852459	1.84	1E+20				
6	\$E\$19		0	1.354655738	1.44	1E+20				
7	\$F\$19		0	1.576327869	2.29	1E+20				
8	\$G\$19		0	0.094	0.77	0.794851064				
9	\$H\$19		0	1.227590164	1.29	1E+20				
10	\$I\$19		0	0.575590387	0.6	0.910769231				
11	\$J\$19		0	0.657278689	0.72	1E+20				
12										
13										
14				Lower Bound Range			Upper Bound Range			
15	Cell	Constraint Name	Shadow Price	Lowest	Current	Highest	Lowest	Current	Highest	
16	\$C\$28		0	-1E+20	2000	3965.368852	3965.368852	9999	1E+20	
17	\$C\$29		0.027704918	297.6	350	375	350	375	1E+20	
18	\$C\$30		0	-1E+20	55	172.0286885	172.0286885	9999	1E+20	
19	\$C\$31		0.026754098	63.05309735	100	171.0176991	100	9999	1E+20	
20	\$C\$32		0	-1E+20	100	132.2131148	132.2131148	9999	1E+20	
21	\$C\$33		0	-1E+20	100	234.2213115	234.2213115	9999	1E+20	
22	\$C\$34		0.024836066	17.69230769	100	142.8205128	100	9999	1E+20	
23										

CPLEX Model Parameter Settings

How to set and use CPLEX parameters and other options when solving your problem models.

In this section

CPLEX Model Parameter Settings: General

Describes the options that can be set in the General section of the CPLEX Model Parameters window.

CPLEX Model Parameter Settings: Algorithmic

Describes the options that can be set in the Algorithmic section of the CPLEX Model Parameters window.

CPLEX Model Parameter Settings: Polishing

Describes the options that can be set in the Polishing section of the CPLEX Model Parameters window.

CPLEX Advanced Parameters

Describes the options that can be set on the CPLEX Advanced Parameters window.

CPLEX Model Parameter Settings: General

Basic CPLEX parameter settings are accessed by clicking the **Model Parameters** button on the CPLEX Model Information window.

When you do this, the CPLEX Model Parameters window is displayed.

The individual options within the **General** section are discussed in the following sections.

Setting the Time Limit option

- ◆ Enter the maximum time you want (in seconds) the CPLEX optimizer to spend solving this model into the **Time Limit** field.

The default value is **1e+75** seconds.

This option maps to the CPLEX `TiLim C++` parameter.

It sets the maximum time, in seconds, that CPLEX will spend finding a solution to your problem.

Setting the Iteration Limit option

- ◆ Type the value you want for the maximum number of iterations you want the CPLEX optimizer to perform while solving this model into the **Iteration Limit** field.

The default value is **2,100,000,000** iterations.

This option maps to the CPLEX `ItLim` and `BarItLim C++` parameters for problems without integer constraints or binary constraints. For problems with integer constraints or binary constraints this option is ignored.

It sets the maximum number of simplex iterations to be performed before the algorithm terminates without reaching optimality. When set to **0** (zero), no simplex method iteration occurs. However, CPLEX factors the initial basis from which solution routines provide information about the associated initial solution.

Setting the Node Limit option

- ◆ Enter the value you want to set for the maximum number of nodes solved before the algorithm terminates without reaching optimality into the **Node Limit** field.

The default value is **2,100,000,000**. You can enter any nonnegative integer.

Sets the maximum number of nodes solved before the algorithm terminates without reaching optimality.

When this parameter is set to **0** (zero), CPLEX completes processing at the root; that is, it creates cuts and applies heuristics at the root. When this parameter is set to **1** (one), it allows branching from the root; that is, nodes are created but not solved.

This option maps to the CPLEX `NodeLim C++` parameter.

Setting the Relative MIP Gap option

- ◆ Enter the relative MIP gap at which CPLEX should stop searching for improved solutions.

The default value is **0.0001**, which means that CPLEX will stop once it has found a feasible integer solution within 0.01 percent of optimal. The relative MIP gap is calculated like this:

```
|bestnode-bestinteger|/(1e-10+|bestinteger|)
```

You can enter any nonnegative value. For example, to instruct CPLEX to stop as soon as it has found a feasible integer solution proved to be within five percent of optimal, set the **Relative MIP Gap** to 0.05.

This option maps to the CPLEX `EpGap` C++ parameter.

Setting the Feasibility Tolerance option

- ◆ Type the value of feasibility tolerance you want into the **Precision** field.

The default value is **0.000001**.

This option maps to the CPLEX `EpRHS` and `EpInt` C++ parameters.

It specifies the amount by which an integer variable can be different from an integer and still be considered feasible. A value of zero is permitted, and the optimizer will attempt to meet this tolerance.

Feasibility tolerance is the degree to which values of the basic variables calculated by the simplex method may violate their bounds. Feasibility influences the selection of an optimal basis and can be reset to a higher value when a problem is having difficulty maintaining feasibility during optimization. You may also wish to lower this tolerance after finding an optimal solution if there is any doubt that the solution is truly optimal. If the feasibility tolerance is set too low, CPLEX may falsely conclude that a problem is infeasible.

Stop after each integer solution

- ◆ Check the **Stop after each integer solution** checkbox to stop the solver each time a new solution is found for MIP problems with integer variables.

Note: This option is only effective on MIP problems.

This option pauses the solver each time a new solution is found for MIP problems with integer variables. For continuous models this option is ignored.

MIP problems often take some time to reach an optimal solution, so this option allows you to stop at each solution and determine whether it is acceptable. If so, you can stop the solve at this point and accept the solution.

For more information on how to utilize this feature during solving, see *Stepping through MIP solutions*.

Verify model is linear and/or quadratic

- ◆ Check the **Verify model is linear and/or quadratic** checkbox to instruct CPLEX to alert you to functions that it does not recognize.

If a model contains formulas that CPLEX does not know, it will try to compute linear or quadratic coefficients for it by evaluating the expression at different points. CPLEX will warn you about this approximation if this box is checked.

CPLEX Model Parameter Settings: Algorithmic

Basic CPLEX parameter settings are accessed by clicking the **Model Parameters** button on the CPLEX Model Information window.

When you do this, the CPLEX Model Parameters window is displayed.

The individual options within the **Algorithmic** section are discussed in the following sections.

In general, the default settings of the algorithmic options work very well on most problems. You only need to consider changing these options if the performance or accuracy of the results you obtain are not satisfactory.

- ◆ You should consider setting the **Root Algorithm** parameter if you are not satisfied with the speed of CPLEX on models without integer and binary variables.
- ◆ You should consider setting the **MIP Emphasis** parameter if you are not satisfied with the speed of CPLEX on problems with integer and/or binary variables.
- ◆ You should consider setting the **Numerical Emphasis** parameter if your model has both very small and very large numbers or if you believe the problem has a solution but CPLEX says there is none.

Setting the Root Algorithm option

Setting the **Root Algorithm** parameter may be appropriate if you are not satisfied with the speed of CPLEX on models without integer and binary variables.

- ◆ Select which continuous optimizer will be used to solve the initial relaxation of a MIP from the **Root Algorithm** drop-down list.

The possible choices for this option are:

- ◆ **Automatic** — (the default) Let CPLEX choose the optimizer.
- ◆ **Primal Simplex** — Use the Primal Simplex optimizer.
- ◆ **Dual Simplex** — Use the Dual Simplex optimizer.
- ◆ **Network Simplex** — Use the Network Simplex optimizer.
- ◆ **Barrier** — Use the Barrier optimizer.
- ◆ **Sifting** — Use the Sifting optimizer.
- ◆ **Concurrent** — Use the Concurrent optimizer.

This option maps to the CPLEX `RootAlg C++` parameter.

The default **Automatic** setting of this parameter currently selects the Dual Simplex optimizer models with linear constraints and a linear objective, and the Barrier optimizer for models with either quadratic constraints or a quadratic objective. The **Automatic** setting may be expanded in the future so that CPLEX chooses the algorithm based on additional characteristics of the model.

Setting the Node Algorithm option

- ◆ Select which continuous optimizer will be used to solve the subproblems in a MIP after the initial relaxation from the **Node Algorithm** drop-down list.

The possible choices for this option are:

- ◆ **Automatic** — (the default) Let CPLEX choose the optimizer.
- ◆ **Primal Simplex** — Use the Primal Simplex optimizer.
- ◆ **Dual Simplex** — Use the Dual Simplex optimizer.
- ◆ **Network Simplex** — Use the Network Simplex optimizer.
- ◆ **Barrier** — Use the Barrier optimizer.
- ◆ **Sifting** — Use the Sifting optimizer.

This option maps to the CPLEX `NodeAlg C++` parameter.

`NodeAlg` only applies to models with binary and integer variables; that is, MIPs. The default **Automatic** setting of this parameter currently selects the Dual Simplex optimizer for models with linear constraints and the Barrier algorithm for models with quadratic constraints.

Setting the Scaling option

- ◆ Select the scaling appropriate for your model from the **Scaling** drop-down list.

The possible choices for this option are:

- ◆ **None** — No scaling.
- ◆ **Equilibration** — (the default) Equilibration scaling.
- ◆ **Aggressive** — Aggressive scaling.

This option maps to the CPLEX `ScalInd C++` parameter.

Poorly conditioned problems (that is, problems in which even minor changes in data result in major changes in solutions) may benefit from an alternative scaling method. Scaling attempts to rectify poorly conditioned problems by multiplying rows or columns by constants without changing the fundamental sense of the problem. If you observe that your problem has difficulty staying feasible during its solution, then you should consider an alternative scaling method.

Setting the MIP Probing Level option

- ◆ Select your desired level of probing in the solution of your model from the **MIP Probing Level** drop-down list.

The possible choices for this option are:

- ◆ **None** — No probing is performed.
- ◆ **Automatic** — (the default) CPLEX will automatically decide an appropriate level of probing.
- ◆ **Moderate** — A moderate level of probing is performed.

- ◆ **Aggressive** — An aggressive level of probing is performed.
- ◆ **Very Aggressive** — A very aggressive level of probing is performed.

This option maps to the CPLEX `Probe C++` parameter.

The probing feature can help in many different ways on difficult models. Probing is a technique that looks at the logical implications of fixing each binary variable to 0 (zero) or 1 (one). It is performed after preprocessing and before the solution of the root relaxation.

Probing can be expensive, so this parameter should be used selectively. On models that are in some sense easy, the extra time spent probing may not reduce the overall time enough to be worthwhile. On difficult models, probing may incur very large runtime costs at the beginning and yet pay off with shorter overall runtime. When you are tuning performance, it is usually because the model is difficult, and then probing is worth trying.

Setting the MIP Emphasis option

Setting the **MIP Emphasis** parameter may be appropriate when you are not satisfied with the speed of CPLEX on problems with integer and/or binary variables.

- ◆ Select a setting to control trade-offs between speed, feasibility, optimality, and moving bounds in MIP from the **MIP Emphasis** drop-down list.

The possible choices for this option are:

- ◆ **Balance opt and int fea** — (the default) Balance optimality and feasibility.
- ◆ **Integer feasibility** — Emphasize feasibility over optimality.
- ◆ **Optimality** — Emphasize optimality over feasibility.
- ◆ **Moving best bound** — Emphasize moving best bound.
- ◆ **Finding hidden feasibles** — Emphasize finding hidden feasible solutions.

This option maps to the CPLEX `MIPEmphasis C++` parameter.

With the default setting of **Balanced**, CPLEX works toward a rapid proof of an optimal solution, but balances that with effort toward finding high quality feasible solutions early in the optimization.

When this parameter is set to **Feasibility**, CPLEX frequently will generate more feasible solutions as it optimizes the problem, at some sacrifice in the speed to the proof of optimality.

When set to **Optimality**, less effort may be applied to finding feasible solutions early.

With the setting **Best Bound**, even greater emphasis is placed on proving optimality through moving the best bound value, so that the detection of feasible solutions along the way becomes almost incidental.

When the parameter is set to **Hidden Feasibility**, the MIP optimizer works hard to find high quality feasible solutions that are otherwise very difficult to find, so consider this setting when the **Feasibility** setting has difficulty finding solutions of acceptable quality.

Setting the Numerical Emphasis option

Setting the **Numerical Emphasis** parameter may be appropriate when your model has both very small and very large numbers or when you believe the problem to have a solution but CPLEX says there is none.

- ◆ To emphasize precision in numerically unstable or difficult problems, check the **Numerical Emphasis** box.

The possible choices for this option are:

- ◆ Unchecked — (the default) Use CPLEX default numerical precision.
- ◆ Checked — Exercise extreme caution in computation.

This option maps to the CPLEX `NumericalEmphasis C++` parameter.

This parameter lets you indicate to CPLEX that it should emphasize precision in numerically difficult or unstable problems, with consequent performance trade-offs in time and memory.

This parameter is intended as a way to tell CPLEX to exercise more than the usual caution in its computations. When you set it to its nondefault value (Checked), specifying extreme numerical caution, various tactics are invoked internally to try to avoid loss of numerical accuracy in the steps of the barrier algorithm.

Be aware that the nondefault setting may result in slower solution times than usual. The effect of this setting is to shift the emphasis away from fastest solution time and toward numerical caution. On the other hand, if numerical difficulty is causing the barrier algorithm to perform excessive numbers of iterations due to loss of significant digits, it is possible that the setting of extreme numerical caution could actually result in somewhat faster solution times. Overall, it is difficult to project the impact on speed when using this setting.

The purpose of this parameter setting is not to generate "more accurate solutions" particularly where the input data is in some sense unsatisfactory or inaccurate. The numerical caution is applied during the steps taken by the barrier algorithm during its convergence toward the optimum, to help it do its job better. On some models, it may turn out that solution quality measures ($Ax=b$ residuals, variable-bound violations, dual values, etc.) are improved when CPLEX exercises numerical caution, but this would be a secondary outcome from better convergence.

Setting the Apply Presolve option

- ◆ To invoke the presolver to simplify and reduce problems before the problem is solved, check the **Apply Presolve** box.

The possible choices for this option are:

- ◆ Checked — (the default) Apply presolve.
- ◆ Unchecked — Do not apply presolve.

This option maps to the CPLEX `PreInd C++` parameter.

When you invoke the MIP optimizer, CPLEX by default automatically preprocesses your problem. In preprocessing, CPLEX applies its presolver one or more times to reduce the size of the integer program in order to strengthen the initial linear relaxation and to decrease the overall size of the mixed integer program.

Setting the Invoke Aggregator option

- ◆ To invoke the aggregator to use substitution where possible to reduce the number of rows and columns before the problem is solved, check the **Invoke Aggregator** box.

The possible choices for this option are:

- ◆ Checked — (the default) Automatically use the aggregator.

CPLEX Model Parameter Settings: Polishing

Basic CPLEX parameter settings are accessed by clicking the **Model Parameters** button on the CPLEX Model Information window.

When you do this, the CPLEX Model Parameters window is displayed.

The individual options within the **Parameters Before Starting to Polish a Feasible Solution** section are discussed in the following sections.

Solution polishing is heuristic technique which can be applied to problems with integer and binary variables.

Solution polishing can yield better solutions in situations where good solutions are otherwise hard to find. More time-intensive than other heuristics, solution polishing is actually a variety of branch & cut that works after an initial solution is available. In fact, it requires a solution to be available for polishing, either a solution produced by branch & cut, or a MIP start supplied by a user.

Because of the high cost entailed by solution polishing, it is not called throughout branch & cut like other heuristics. Instead, solution polishing works in a second phase after a first phase of conventional branch & cut. As an additional step after branch & cut, solution polishing can improve the best known solution. Polishing will start if there is a solution and any of the multiple starting criteria have been met.

As a kind of branch & cut algorithm itself, solution polishing focuses solely on finding better solutions. Consequently, it may not prove optimality, even if the optimal solution has indeed been found.

The individual options on this window are discussed in the following sections.

Setting the Absolute MIP Gap option

- ◆ Enter the absolute MIP gap at which solution polishing should begin into the **Absolute MIP Gap** field.

The absolute MIP gap is the difference between the best integer objective and the objective of the best node remaining, after which CPLEX stops branch-and-cut and begins polishing a feasible solution.

The default value is **0**, which means that by default CPLEX does not invoke solution polishing by default. You can enter any nonnegative value.

This option maps to the CPLEX `PolishAfterEpAGap C++` parameter.

CPLEX must have a feasible solution in order to start polishing. It must also have certain internal structures in place to support solution polishing.

Consequently, when the criterion specified by this parameter is met, CPLEX begins solution polishing only after these starting conditions are also met. That is, there may be a delay between the moment when the criterion specified by this parameter is met and when solution polishing starts.

Setting the Relative MIP Gap option

- ◆ Enter the relative MIP gap at which solution polishing should begin into the **Relative MIP Gap** field.

The relative MIP gap is the difference between the best integer objective and the objective of the best node remaining, after which CPLEX stops branch-and-cut and begins polishing a feasible solution.

The default value is **0**, which means that by default CPLEX does not invoke solution polishing by default. The relative MIP gap is calculated like this:

```
|bestnode-bestinteger|/(1e-10+|bestinteger|)
```

You can enter any nonnegative value.

This option maps to the CPLEX `PolishAfterEpGap` C++ parameter.

CPLEX must have a feasible solution in order to start polishing. It must also have certain internal structures in place to support solution polishing.

Consequently, when the criterion specified by this parameter is met, CPLEX begins solution polishing only after these starting conditions are also met. That is, there may be a delay between the moment when the criterion specified by this parameter is met and when solution polishing starts.

Setting the Integer Solutions to Find option

- ◆ Type the value you want to set for the number of integer solutions to find before CPLEX stops branch-and-cut and begins to polish a feasible solution into the **MIP Integer Solutions to Find** field.

The default value is **2,100,000,000**, which means that by default CPLEX does not invoke solution polishing by default.

You can enter any positive integer strictly greater than zero; zero is not allowed.

This option maps to the CPLEX `PolishAfterIntSol` C++ parameter.

CPLEX must have a feasible solution in order to start polishing. It must also have certain internal structures in place to support solution polishing.

Consequently, when the criterion specified by this parameter is met, CPLEX begins solution polishing only after these starting conditions are also met. That is, there may be a delay between the moment when the criterion specified by this parameter is met and when solution polishing starts.

Setting the Nodes to Process option

- ◆ Type the value you want to set the number of nodes processed in branch-and-cut before CPLEX begins to polish a feasible solution into the **Nodes to Process** field.

The default value is **2,100,000,000**, which means that by default CPLEX does not invoke solution polishing by default.

You can enter any nonnegative integer.

This option maps to the CPLEX `PolishAfterNode` C++ parameter.

When this parameter is set to **0** (zero), CPLEX completes processing at the root; that is, it creates cuts and applies heuristics at the root.

When this parameter is set to **1** (one), it allows branching from the root; that is, nodes are created but not solved.

When no feasible solution is available yet, CPLEX explores more nodes than the number specified by this parameter.

CPLEX must have a feasible solution in order to start polishing. It must also have certain internal structures in place to support solution polishing.

Consequently, when the criterion specified by this parameter is met, CPLEX begins solution polishing only after these starting conditions are also met. That is, there may be a delay between the moment when the criterion specified by this parameter is met and when solution polishing starts.

Setting the Time option

- ◆ Type the value you want (in seconds) to spend during mixed integer optimization before CPLEX starts polishing a feasible solution into the **Time** field.

The default value is **1E+75** seconds, which means that by default CPLEX does not invoke solution polishing.

You can enter any nonnegative value in seconds.

This option maps to the CPLEX `PolishAfterTime` C++ parameter.

CPLEX must have a feasible solution in order to start polishing. It must also have certain internal structures in place to support solution polishing.

Consequently, when the criterion specified by this parameter is met, CPLEX begins solution polishing only after these starting conditions are also met. That is, there may be a delay between the moment when the criterion specified by this parameter is met and when solution polishing starts.

CPLEX Advanced Parameters

Advanced CPLEX parameter settings are accessed by clicking the **Advanced Parameters** button on the CPLEX Model Information window.

When you do this, the CPLEX Advanced Parameters window is displayed.

You will typically only use these settings with the advice of CPLEX technical support.

The individual options on this window are discussed in the following sections.

Adding CPLEX parameters by reference

The procedure below allows you to specify additional CPLEX parameters using their C++ Parameter Name in the *IBM ILOG CPLEX Parameters Reference Manual*. This option is usually used only if you have been instructed to use it by CPLEX technical support.

To add a CPLEX parameter:

1. Click the **Advanced Parameters** button on the CPLEX Model Information window to access the Advanced Parameters window.
2. In the **Name/Number** field, type the name of the CPLEX parameter you are being instructed to enter.

For example, if CPLEX technical support tells you to enter the `Advanced start switch` parameter, its C++ Name is `AdvInd`, so you type **AdvInd** into this field.

3. If the parameter has a required value, type it in the **Value** field.

In this example, you might be instructed by CPLEX technical support to enter **0**, which means `Do not use advanced start information`.

4. Click the **Add Parameter** button.

When you solve the model, this parameter is invoked, with its supplied value.

To change a CPLEX parameter:

1. On the Advanced Parameters window, select the parameter you want to change from the **Advanced Parameters** list box.
2. Change the **Value** of the selected parameter as required.

For example, CPLEX technical support might tell you to change the **Value** of the `AdvInd` parameter.

3. When you have finished editing the parameter, click the **Update Parameter** button.

When you solve the model, this parameter is invoked, with its changed value.

To remove a CPLEX parameter:

1. On the Advanced Parameters window, select the parameter you want to change from the **Advanced Parameters** list box.
2. Click **Remove Parameter**.

Debugging: Model to File

This option allows you to write the model to an external file in a format which can be read by the CPLEX Interactive System. This is primarily used by CPLEX technical support as a debugging feature. Unusual values in the file can indicate problems in the Excel formulation of your problem.

1. Check the **Model to file** box.
2. Type the name of the file in which you want CPLEX to save a copy of this model when it is solved.

The filename must have one of the following extensions, corresponding to the type of CPLEX file format:

- ◆ .lp
- ◆ .mps
- ◆ .sav

You can specify a path for the file if you want. If you do not, by default it is saved in your `My Documents` folder.

IBM ILOG CPLEX for Microsoft Excel Examples

Contains illustrative examples of linear programming problems that can be solved using IBM® ILOG® CPLEX® for Microsoft® Excel. All examples are located in the <plexinstalldir>\examples\src\excel directory.

In this section

blend.xls

Provides an optimization model of a classic blending problem that involves mixing some metals to form an alloy.

diet.xls

Provides an optimization model of the classic diet problem, a linear program to choose the amount of foods in a diet to meet nutritional requirements at the lowest cost.

facility.xls

Provides an optimization model of a facility location problem that decides which facilities to open in order to supply clients at minimum cost.

fixcost1.xls

Provides an optimization model of a production planning problem with fixed costs that minimizes the sum of fixed and variable costs so that the company exactly meets demand.

foodmanufact.xls

Provides an optimization model of a classic food manufacturing problem in which production is planned over six months to maximize profit when refining and blending several types of oils into a final product which is sold.

inout1.xls

Provides an optimization model of a production planning problem in which a company produces 3 products, using resources with limited capacity. Each product consumes a given number of machines, and has a specified inside cost and outside cost.

lpex1.xls

Provides a translation from an algebraic form of a simple linear program to its spreadsheet form.

mipex1.xls

Provides a translation from the algebraic form of a simple mixed integer program to its spreadsheet form.

miqpex1.xls

Provides a translation from the algebraic form of a mixed integer quadratic program to its spreadsheet form.

qcpex1.xls

Provides a translation from the algebraic form of a simple problem with a quadratic objective and a quadratic constraint to its spreadsheet form.

qpex1.xls

Provides a translation from the algebraic form of a simple problem with a quadratic objective to its spreadsheet form.

rates.xls

Provides an optimization model of a power generation problem to decide which of a set of generators with minimum and maximum operational capacities to use to meet the demand at minimal cost.

steel.xls

Provides an optimization model of a multiperiod production model.

Using IBM ILOG CPLEX for Microsoft Excel with VBA

Describes the modules available for use with Visual Basic.

blend.xls

This is an implementation in Excel spreadsheet form of a classic blending problem. The metal blending example involves mixing some metals to form an alloy.

The metal may come from several sources: in pure form, from raw materials, as scraps from previous mixes, or as ingots. The alloy must contain certain amounts of the various metals, as expressed by a production constraint specifying lower and upper bounds for the quantity of each metal in the alloy. Each source has a cost and the problem consists of blending from the sources while minimizing the cost and satisfying the production constraint. Similar problems arise in other domains, for example, the oil, paint, and food-processing industries.

The problem is located in the `<cpLEXinstallDir>\examples\src\excel` directory and is defined in the spreadsheet as follows:

- ◆ **Data Section** — The knowns in this problem are the component elements of the alloy. Each source material is composed of these elements in varying proportions. The proportion of the elements in the alloy can vary between minimum and maximum proportions. The cost of each component and the amount of alloy to produce are also given.
- ◆ **Variables Section** — The unknowns are the amounts of each source materials to use and the amount of each element in the alloy.
- ◆ **Constraints Section** — The constraints in this problem include:
 - The sum of the elements in the alloy must equal the amount of the alloy.
 - The alloy composition must be in a specific range for each element. These constraints use the variables containing the amount of each element in the alloy and the minimum and maximum proportion from the data table.
- ◆ **Objective Function Section** — The objective is to minimize the cost of producing the alloy. The objective function is the sum of the cost of each source times the amount of that source used in the alloy.

diet.xls

This is an implementation in Excel spreadsheet form of the classic diet problem — a linear program to choose the amount of foods in a diet to meet nutritional requirements at the lowest cost.

The problem is located in the <oplexinstalldir>\examples\src\excel directory and is defined in the spreadsheet as follows:

- ◆ Data Section — The knowns in this problem are the amount of nutrition provided by a given quantity of a given food, the cost per unit of food, and the upper and lower bounds on the foods to be purchased for the diet.
- ◆ Variables Section — The unknowns are the quantities of each food to buy.
- ◆ Constraints Section — The constraints in this problem are that the food bought to consume must satisfy basic nutritional requirements, and that amount of each food purchased must not exceed what is available.
- ◆ Objective Function Section — The objective is to minimize the cost of the food to be bought. (The objective function is the sum of the cost of each food times the amount of each food.)

facility.xls

This is an implementation in Excel spreadsheet form of a facility location problem. The problem is to decide which facilities to open in order to supply clients at minimum cost. Opening a facility has a fixed cost and the cost to supply a client from a facility depends on the facility.

The problem is located in the `<plexinstalldir>\examples\src\excel` directory and is defined in the spreadsheet as follows:

- ◆ Data Section — The knowns in this problem are the names, capacity, and fixed costs of each facility. Also known are the supply costs for each client per facility.
- ◆ Variables Section — The unknowns are binary variables for each facility-client pair (which is **1** if that assignment is used), and a binary variable for each facility (which is **1** if the facility is to be used).
- ◆ Constraints Section — The constraints in this problem include:
 - Number of clients supplied from a facility must not exceed the capacity of the facilities. The capacity is **0** unless the facility is open.
 - The number of facilities that supply a client must be **1**.
- ◆ Objective Function Section — The objective is to minimize the cost of supplying all clients. There is a cost for opening each facility and a cost for using a facility to supply a client.

fixcost1.xls

This is an implementation in Excel spreadsheet form of a production planning problem with fixed costs.

A company must produce a product on a set of machines. Each machine has limited capacity. Producing a product on a machine has both a fixed cost and a cost per unit of production.

The problem is to minimize the sum of fixed and variable costs so that the company exactly meets demand.

The problem is located in the <oplexinstalldir>\examples\src\excel directory and is defined in the spreadsheet as follows:

- ◆ Data Section — The knowns in this problem are the Fixed costs, Variable costs, and Capacity. Also known is the Demand.
- ◆ Variables Section — The unknowns indicate whether each machine is used, and if so, how much it is used.
- ◆ Constraints Section — The constraints in this problem are machine capacity constraints. A machine has capacity 0 if unused and capacity as specified in the table if used.
- ◆ Objective Function Section — The objective is the sum of the fixed costs for machines which are selected and the variable costs for the amount of production on the selected machines, which should be minimized.

Note: `fixcost1.xls` contains an example of using Excel names or named ranges to illustrate the use of this syntax.

foodmanufact.xls

This is an implementation in Excel spreadsheet form of a classic food manufacturing problem — two types of vegetable oils and three types of non-vegetable oils are available to refine and blend into a final product which is sold. The production is planned over six months to maximize profit.

This example was taken from the example in the book *Model Building in Mathematical Programming*, 4th ed. by H. P. Williams. This is the second version of the problem where the additional conditions which make the problem a mixed integer program are imposed.

The problem is located in the `<complexinstalldir>\examples\src\excel` directory and is defined in the spreadsheet as follows:

- ◆ Data Section — The knowns in this problem are data concerning the oils: refining capacity, product price, the maximum oils to be used, and the upper and lower limits of the hardness of the oils. Also known is a forecast of the price of the raw oils over the next six months.
- ◆ Variables Section — The unknowns are the amounts of each oil to be stored and bought over the six months of the planning horizon.
- ◆ Constraints Section — The constraints in this problem include:
 - The final product must have its hardness between minimum and maximum standards. The hardness of the final product is linear in the hardness of the input oils.
 - If a raw oil is used in the final product, at least the specified amount must be used.
 - Linking constraints — the amount of product stored in the previous month plus the amount bought in this month must equal the amount used and stored in this month.
 - Each type of oil is refined on separate equipment which has a limited capacity.
 - The hardness of the final product must meet upper and lower limits.
 - There is a constraint to link the `InUse` variable to the `Use` variable
 - There is a constraint to limit the number of oils in the blend.
 - If `Veg Oil` is used, also use `Oil 3`.
 - Constraint on minimal oil usage if oil is used.
- ◆ Objective Function Section — The objective is to maximize profit: revenue from selling product minus storage expense and the cost of raw oils.

inout1.xls

This is an implementation in Excel spreadsheet form of a production planning problem.

A company has to produce 3 products, using 2 resources. Each resource has a limited capacity. Each product consumes a given number of machines. Each product has a production cost (the inside cost). Both products can also be brought outside the company at a given cost (the outside cost).

The problem is located in the `<oplexinstalldir>\examples\src\excel` directory and is defined in the spreadsheet as follows:

- ◆ Data Section — The knowns in this problem are the resources and capacity for each of the three products. Also known are the demand, insideCost, and outsideCost for each of the three products.
- ◆ Variables Section — The unknowns are the amount of each product produced or bought from outside.
- ◆ Constraints Section — The constraints in this problem include:
 - Demand Satisfaction Constraints — every demand must be satisfied, either by internal production or outside purchase.
 - Resource Capacity Constraints — resource capacity limit must be respected.
- ◆ Objective Function Section — The objective is to minimize the total cost so that the company exactly meets the demand.

lpex1.xls

This is an implementation in Excel spreadsheet form of a linear programming problem. This problem would be expressed mathematically as follows:

```
Maximize
  obj: x1 + 2 x2 + 3 x3
Subject To
  c1: - x1 + x2 + x3 <= 20
  c2: x1 - 3 x2 + x3 <= 30
Bounds
  0 <= x1 <= 40
  0 <= x2 <= infinity
  0 <= x3 <= infinity
```

The problem is located in the <plexinstalldir>\examples\src\excel directory and is defined in the spreadsheet as follows:

- ◆ Data Section — All problem data except constraints' right-hand sides are given in the table in the spreadsheet.
- ◆ Variables Section — Lower and upper bounds are given for each variable. Note that an upper bound of $1e+20$ or above is interpreted as *infinity* (and similarly a lower bound of $-1e+20$ and below as *-infinity*).
- ◆ Constraints Section — The constraints in this problem except the constraints' right-hand sides are contained in the table. Left-hand sides of constraints and objective function are given as matrix A and column vector b so that they can be written as $v^T A v + 1^T v$ where v is the vector of unknowns.
- ◆ Objective Function Section — The objective is a linear function that will be probed by the CPLEX solver.

mipex1.xls

This is an implementation in Excel spreadsheet form of a mixed integer programming (MIP) problem.

This problem would be expressed mathematically as follows:

Maximize

$$\text{obj: } x1 + 2 x2 + 3 x3 + x4$$

Subject To

$$\text{c1: } -x1 + x2 + x3 + 10 x4 \leq 20$$

$$\text{c2: } x1 - 3 x2 + x3 \leq 30$$

$$\text{c3: } x2 - 3.5 x4 = 0$$

Bounds

$$0 \leq x1 \leq 40$$

$$0 \leq x2 \leq \text{infinity}$$

$$0 \leq x3 \leq \text{infinity}$$

The problem is located in the `<plexinstalldir>\examples\src\excel` directory and is defined in the spreadsheet as follows:

- ◆ Data Section — The knowns in this problem are the coefficients for the three constraints, the variable upper and lower bounds, and the objective coefficients.
- ◆ Variables Section — Identifies an explicit lower bound of **0**.
- ◆ Constraints Section — The constraints in this problem except the constraints' right-hand sides are contained in the table. The left-hand sides of constraints and the objective function are specified as row vectors to be multiplied by the vector of unknowns.
- ◆ Objective Function Section — The objective is to solve the problem as defined in the spreadsheet.

miqpex1.xls

This is an implementation in Excel spreadsheet form of a problem in which the objective function includes a quadratic term and the model includes integer constraints.

The problem is located in the `<cpflexinstalldir>\examples\src\excel` directory and is defined in the spreadsheet as follows:

- ◆ Data Section — All problem data except constraints' right-hand sides are given in the table in the spreadsheet.
- ◆ Variables Section — Lower and upper bounds are given for each variable. Note that an upper bound of $1e+20$ or above is interpreted as *infinity* (and similarly a lower bound of $-1e+20$ and below as *-infinity*).
- ◆ Constraints Section — The constraints in this problem except the constraints' right-hand sides are contained in the table. Left-hand sides of constraints are specified as row vectors to be multiplied by the vector of unknowns.
- ◆ Objective Function Section — The objective function is specified as quadratic (a matrix) and linear part so that it can be written as

$$c^T v - 0.5 * v^T Q v$$

where c is the linear part and Q the quadratic matrix.

Note: The objective function is a quadratic function that will be probed by the CPLEX solver. To avoid the information message about this, the **Verify model is linear and/or quadratic** option has been unchecked in the CPLEX Model Parameters window.

qcpex1.xls

This is an implementation in Excel spreadsheet form of a problem that has quadratic constraints as well as a quadratic objective function.

This problem would be expressed mathematically as follows:

Maximize

$$x + 2y + 3z - 0.5 * (33x^2 + 22y^2 + 11z^2 - 12xy - 23yz)$$

Subject To

$$c1: -x + y + z \leq 20$$

$$c2: x - 3y + z \leq 30$$

$$c3: x^2 + y^2 + z^2 \leq 1$$

Bounds:

$$0 \leq x \leq 40$$

$$0 \leq y \leq \text{infinity}$$

$$0 \leq z \leq \text{infinity}$$

The problem is located in the `<cpalexinstalldir>\examples\src\excel` directory and is defined in the spreadsheet as follows:

- ◆ Data Section — All problem data except constraints' right-hand sides are given in the table in the spreadsheet.
- ◆ Variables Section — Lower and upper bounds are given for each variable. Note that an upper bound of `1e+20` or above is interpreted as `infinity` (and similarly a lower bound of `-1e+20` and below as `-infinity`).
- ◆ Constraints Section — The constraints in this problem except the constraints' right-hand sides are contained in the table. Left-hand sides of constraints and objective function are given as matrix `A` and column vector `l` so that they can be written as $v^T A v + l^T v$ where `v` is the vector of unknowns. If either the quadratic or linear part are unused we leave them blank (and don't include them into the respective formula).
- ◆ Objective Function Section — The objective is a quadratic function that will be probed by the CPLEX solver.

Note: The objective function is a quadratic function that will be probed by the CPLEX solver. To avoid the information message about this, the **Verify model is linear and/or quadratic** option has been unchecked in the CPLEX Model Parameters window.

qpex1.xls

This is an implementation in Excel spreadsheet form of a general linear programming problem with quadratic objective.

This problem would be expressed mathematically as follows:

Maximize

$$x + 2y + 3z - 0.5 * (33x^2 + 22y^2 + 11z^2 - 12xy - 23yz)$$

Subject To

$$c1: -x + y + z \leq 20$$

$$c2: x - 3y + z \leq 30$$

Bounds

$$0 \leq x \leq 40$$

$$0 \leq y \leq \text{infinity}$$

$$0 \leq z \leq \text{infinity}$$

The problem is located in the <oplexinstalldir>\examples\src\excel directory and is defined in the spreadsheet as follows:

- ◆ Data Section — All problem data except constraints' right-hand sides are given in the table in the spreadsheet. Constraints' left-hand sides are specified as row vectors to be multiplied by the vector of unknowns.
- ◆ Variables Section — Lower and upper bounds are given for each variable.
- ◆ Constraints Section — The constraints in this problem except the constraints' right-hand sides are contained in the table. The left-hand sides of constraints and the objective function are specified as row vectors to be multiplied by the vector of unknowns.
- ◆ Objective Function Section — The objective is specified as quadratic (a matrix) and linear part so that it can be written as $\mathbf{l}^T \mathbf{v} - 0.5 * \mathbf{v}^T \mathbf{A} \mathbf{v}$ where \mathbf{l} is the linear part and \mathbf{A} the quadratic matrix (both as in the table).

Note: The objective function is a quadratic function that will be probed by the CPLEX solver. To avoid the information message about this, the **Verify model is linear and/or quadratic** option has been unchecked in the CPLEX Model Parameters window.

rates.xls

This is an implementation in Excel spreadsheet form of a power generation problem — given a set of generators with minimum and maximum operational capacities and costs, decide which generators to use to meet the demand at minimal cost.

The problem is located in the `<oplexinstalldir>\examples\src\excel` directory and is defined in the spreadsheet as follows:

- ◆ Data Section — The knowns in this problem are the names of each generator, the minimum and maximum levels at which they can operate, and the cost of operating them.
- ◆ Variables Section — The unknowns are how much power each of the generators can produce, and whether or not they are in use.
- ◆ Constraints Section — The constraints in this problem include:
 - Each generator must be off or be operated between its minimum and maximum levels.
 - The binary `InUse` variables are linked to the `Generate` variables to model this constraint.
- ◆ Objective Function Section — The objective is to determine which generators should be used to meet demand, at the lowest cost.

steel.xls

This is an implementation in Excel spreadsheet form of the model called `steelT.mod` on page 58 of the book *AMPL: A Modeling Language for Mathematical Programming, 2nd. edition* by Robert Fourer, David M. Gay and Brian W. Kernighan (copyright 2003, publisher Thomson Brooks/Cole, ISBN number 0-534-38809-4). In the AMPL example, a multiperiod production model is given, with data for 4 weeks.

The problem is located in the `<cplexinstalldir>\examples\src\excel` directory and is defined in the spreadsheet as follows:

- ◆ Data Section — The knowns in this problem are the two products and their starting inventory and cost and production rate and cost. Also known are the Time Periods for production, the demand for each product per time period, and the revenue per product unit.
- ◆ Variables Section — The unknowns are:
 - The amounts to be produced.
 - The amounts to be inventoried for the next time period.
 - The amounts to be sold.

In each set we have one variable for every product/time period pair.

- ◆ Constraints Section — The constraints in this problem include:
 - Market demand limit — The constraint that we cannot sell more than the market demand is formulated directly on the variables and data and does not use any of the intermediate expressions calculated below.
 - Required production time — The time required to produce the specified amounts of the two products. This must not exceed the working time limits.
 - For each product and time period: the amount sold and inventoried must match the amount produced and received from previous time period. All the terms here must be **0**.
- ◆ Objective Function Section — The objective is to maximize profit — the income from selling products minus the cost for producing and inventorizing it.

Using IBM ILOG CPLEX for Microsoft Excel with VBA

IBM® ILOG® CPLEX® provides functions that allow you to run CPLEX from Visual Basic. There are functions corresponding to the steps listed earlier to specify a model and to set options in solving the model.

To access these functions from a Visual Basic application, you first launch the VBA editor from Microsoft® Excel by choosing **Tools > Macro > Visual Basic Editor**. This brings up a new window for Visual Basic.

CPLEX provides a module for use in Visual Basic:

◆ `cplexvba.bas` — This module has functions names starting with `CPLEX`.

To load the modules, choose **File > Import File** and browse to the location of the modules in the `<cplexinstalldir>\excel` folder and select the `cplexvba.bas` file. You should see the module in the VBA project pane. You can examine the code by clicking on the module name. At this point you are ready to write a VBA application using CPLEX.

Glossary

add-in

A supplemental program that adds custom commands or custom features to Microsoft® Excel.

cell reference

The set of coordinates that indicates the location of a cell on a worksheet. For example, the reference of the cell that appears at the intersection of column B and row 3 is B3. The range of cells in rows 3-7 in column B would be B3:B7. Cell references can be either *relative* or *absolute*.

For example, the B3 example is relative because if you see B3 written in cell D2, this is a reference to the cell that is two columns to the left and one row below the current cell (D2). If you move or copy this formula one column to the right into cell E2 the B3 automatically turns into a C3 since this is still two cells to the left and one row below the new current cell.

If you prefix either the letter or the digit with a "\$" it means that this is an absolute reference and will not change depending on the cell it is used in. Thus B3 has relative row and column, \$B3 has an absolute column and a relative row, B\$8 has a relative column and an absolute row, and \$B\$8 has absolute row and column.

constraint

A constraint is a restriction or rule that must be respected. You can apply constraints to variable cells, to the objective function cell, or to other cells that are directly or indirectly related to the objective function cell.

feasible solution

In IBM® ILOG® CPLEX® for Microsoft® Excel, any specification of the decision variables that satisfies the model's constraints is known as a feasible solution.

In operations research, a solution is feasible if it represents an assignment of values to variables that satisfies the constraints of the problem, even if it does not optimally satisfy the *objective function* of the problem.

formula

A sequence of values, cell references, names, functions, or operators in a cell that performs calculations to produce a new value. An Excel formula always begins with an equal sign (=).

name

A word or string of characters that represents a cell, range of cells, formula, or constant value that can be used in other formulas. Thus you can use easy-to-understand names, such as Products, to refer to hard-to-understand ranges, such as C20:C30. See *cell reference*.

objective function

In operations research, an *objective function* is a mathematical expression to optimize (that is, either to minimize or to maximize) subject to the constraints of the problem.

objective function cell

This cell contains the formula that computes the objective function value. Although multiple cells can be used to perform intermediate calculations, the final objective function calculation must be contained in a single cell.

optimal solution

In IBM ILOG CPLEX for Microsoft Excel, when the model is solved the solver searches all *feasible solutions* and finds the feasible solution that has the "best" objective function value (the largest value for maximum optimization, the smallest for minimum optimization). Such a solution is called an optimal solution.

variable

A variable represents an unknown value in a problem. Each variable may be part of the objective function and/or some number of constraints.

what-if analysis

The process of changing the values in cells to see how those changes affect the solution in the worksheet formulas. For example, you could change the value of a constraint or of input data and re-solve the problem to see if you can achieve a better solution.

Index

A	<ul style="list-style-type: none"> adding constraints 44 Adding CPLEX parameters by Parameter Name 63 analyzing solutions <ul style="list-style-type: none"> sensitivity analysis 47 solution analysis 47 	
B	<ul style="list-style-type: none"> blend.xls example 67 	
C	<ul style="list-style-type: none"> constraints <ul style="list-style-type: none"> adding, changing, or deleting 44 CPLEX Advanced Parameters window 63 CPLEX for Microsoft Excel <ul style="list-style-type: none"> problem types 12 CPLEX Model Parameters window <ul style="list-style-type: none"> Polishing section 60 CPLEX options <ul style="list-style-type: none"> setting 51 CPLEX parameter <ul style="list-style-type: none"> absolute MIP gap before starting to polish a feasible solution 60 AggInd 58 BarItLim 52 clock type for computation time parameter 52 CPX_PARAM_AGGIND 58 CPX_PARAM_BARITLIM 52 CPX_PARAM_EPGAP 52 CPX_PARAM_EPINT 53 CPX_PARAM_EPRHS 53 CPX_PARAM_ITLIM 52 CPX_PARAM_LPMETHOD 55 CPX_PARAM_MIPEMPHASIS 57 CPX_PARAM_NODELIM 52 CPX_PARAM_NUMERICALEMPHASIS 57 CPX_PARAM_POLISHAFTEREPAGAP 60 CPX_PARAM_POLISHAFTEREINTSOL 61 CPX_PARAM_POLISHAFTERNODE 61 CPX_PARAM_POLISHAFTERTIME 62 CPX_PARAM_PREIND 58 CPX_PARAM_PROBE 56 CPX_PARAM_QPMETHOD 55 CPX_PARAM_SCAIND 56 CPX_PARAM_STARTALG 55 CPX_PARAM_SUBALG 56 CPX_PARAM_TILIM 52 EpGap 52 EpInt 53 EpRHS 53 ItLim 52 MIP emphasis switch 57 MIP integer solutions to find before starting to polish a feasible solution 61 MIP probing level 56 MIP starting algorithm 55 MIP subproblem algorithm 56 MIPEmphasis 57 NodeAlg 56 NodeLim 52 nodes to process before starting to polish a feasible solution 61 numerical precision emphasis 57 NumericalEmphasis 57 optimizer time limit 52 PolishAfterEpAGap 60 PolishAfterEpGap 60 PolishAfterIntSol 61 PolishAfterNode 61 PolishAfterTime 62 PreInd 58 presolve switch 58 Probe 56 	

- relative MIP gap before starting to polish a feasible solution **60**
- relative MIP gap General settings **52**
- RootAlg **55**
- Scalnd **56**
- scale parameter **56**
- TiLim **52**
- time before starting to polish a feasible solution **62**
- CPLEX parameter settings
 - Absolute MIP Gap (Polishing) **60**
 - Algorithmic **55**
 - Apply Presolve **58**
 - Assume Linear or Quadratic Model **53**
 - General **52**
 - Invoke Aggregator **58**
 - Iteration Limit **52**
 - MIP Emphasis **57**
 - MIP Integer Solutions to Find **61**
 - MIP Probing Level **56**
 - Node Algorithm **56**
 - Node Limit **52**
 - Nodes to Process **61**
 - Numerical Emphasis **57**
 - Precision **53**
 - Relative MIP Gap **60**
 - Relative MIP Gap (General) **52**
 - Root Algorithm **55**
 - Scaling **56**
 - Stop after each integer solution **53**
 - Stop For Each New MIP Solution **46**
 - Time **62**
 - Time Limit **52**
- CPLEX parameters
 - setting **51**
- cplexvba.bas **80**
- creating an optimization model
 - in Microsoft Excel **15, 20**
 - cell reference notation **23**
 - Constraints Section **21**
 - Data Section **18**
 - Decision Variables Section **19**
 - formulas **23**
 - modeling tips **24**
 - Objective Function Section **20**
 - specifying constraints **22**
 - specifying the objective function cell **22**
 - the main sections of the spreadsheet **16**

D

- Debugging
 - Save model to file **64**
- deleting constraints **44**
- diet.xls example **68**

E

- Examples
 - blend.xls **67**
 - diet.xls **68**
 - facility.xls **69**
 - fixcost1.xls **70**
 - foodmanufact.xls **71**
 - inout1.xls **72**
 - lpex1.xls **73**
 - mipex1.xls **74**
 - qcpex1.xls **76**
 - qpex1.xls **77**
 - rates.xls **78**
 - steel.xls **79**

F

- facility.xls example **69**
- feasible solutions **46, 53**
- fixcost1.xls example **70**
- foodmanufact.xls example **71**

I

- inout1.xls example **72**
- installing CPLEX for Microsoft Excel **7, 9**

L

- licensing **6**
- linear optimization **12**
- loading a problem model **38**
- LP
 - problem format **12**
- lpex1.xls example **73**

M

- MIP
 - description **12**
- mipex1.xls example **74**

O

- online help **37**

P

- problem formulation
 - standard notation for **12**
- problem model
 - loading **38**
 - reset worksheet **49**
 - saving **43**
 - solving **39**

- problem types solved by CPLEX for Microsoft Excel **12**

Q

- QCP
 - description **12**
- qcpex1.xls example **76**
- QP
 - description **12**

qpex1.xls example **77**

R

rates.xls example **78**

registering the CPLEX XLL
with Excel 2003 **7**
with Excel 2007 **9**

resetting the problem model **49**

S

Save model to file **64**

saving a problem model **43**

Setting advanced CPLEX parameters **63**

setting CPLEX options **51**

setting CPLEX parameters **51**

SOCP

description **12**

solving a problem model **39**

steel.xls example **79**

stepping through feasible MIP solutions **46, 53**

U

updating constraints **44**

V

VBA

run CPLEX from Visual Basic **80**

viewing

online help **37**