2950 Niles Road, St. Joseph, MI 49085-9659, USA
269.429.0300 fax 269.429.3852 hq@asabe.org www.asabe.org

An ASABE Meeting Presentation
Paper Number: 1009201

Detecting and categorizing hard-coding errors in Excel Spreadsheets using Visual Basic for Applications (VBA)

Vertika Rawat, Graduate Research Assistant

Iowa State University, Agricultural & Biosystems Engineering Department, vrawat@iastate.edu.

D. Raj Raman, Associate Professor, P.E.

Iowa State University, Agricultural & Biosystems Engineering Department, rajraman@iastate.edu.

Robert P. Anex, Professor

Iowa State University, Agricultural & Biosystems Engineering Department, rpanex@iastate.edu.

Written for presentation at the
2010 ASABE Annual International Meeting
Sponsored by ASABE
David L. Lawrence Convention Center
Pittsburgh, Pennsylvania
June 20 – June 23, 2010

The authors are solely responsible for the content of this technical presentation. The technical presentation does not necessarily reflect the official position of the American Society of Agricultural and Biological Engineers (ASABE), and its printing and distribution does not constitute an endorsement of views which may be expressed. Technical presentations are not subject to the formal peer review process by ASABE editorial committees; therefore, they are not to be presented as refereed publications. Citation of this work should state that it is from an ASABE meeting paper. EXAMPLE: Author's Last Name, Initials. 2010. Title of Presentation. ASABE Paper No. 10----. St. Joseph, Mich.: ASABE. For information about securing permission to reprint or reproduce a technical presentation, please contact ASABE at rutter@asabe.org or 269-429-0300 (2950 Niles Road, St. Joseph, MI 49085-9659 USA).

Abstract. Electronic spreadsheets play an indispensable role in the simulation, modeling, and analysis of bioenergy systems, and their results have the ability to affect decision-making significantly. Prior research has shown that spreadsheets are highly error-prone, and that a large percentage of these errors are difficult to detect. To that end, we developed computer code (implemented in Visual Basic for Applications, running under Microsoft Excel) to detect a particularly insidious form of spreadsheet error: the hard-coding error. These errors are defined as the presence of one or more unreferenced numerical values in a cell formula. Hard-coding errors are dangerous because they are a likely source of erroneous constants and/or non-updating assumptions. The code was used to audit six spreadsheets relevant to bioenergy systems, three developed in our lab (and reported on in other sessions at the AIM), and three in the public domain. The preliminary audit results were analyzed to understand the nature and distribution of hard-coding errors. The preponderance and diversity of hard-coding errors in these spreadsheets motivated us to subcategorize them. Together, the hard-coding error detection program and sub-categorization program provide a robust and rapid means of detecting and categorizing multiple types of hard-coding errors. Use of these programs could increase the reliability of spreadsheet software used in simulation, modeling, and analysis of bioenergy systems.

Keywords. Hard-coding errors, non-updating assumptions, bioenergy, Visual Basic for Applications, VBA, Excel, error detection

1. INTRODUCTION

The versatility of spreadsheets has led to their extensive application at all levels of organizations. Because of their wide use, concerns have been raised about the integrity and validity of spreadsheets (Galletta et al., 1997), and other authors have shown that spreadsheets are highly error-prone (Powell et al., 2009). Approximately 80% of errors documented by EuSpRiG (European Spreadsheet Risks Interest Group) were in a formula (Powell et al., 2008). Users cannot readily detect the majority of such errors, which could result in potentially devastating miscalculations in many settings. The typical approach to debugging spreadsheets involves doing hand calculations to verify the results – unfortunately, this approach is time consuming and is frequently skipped or done cursorily. Furthermore, even if the spreadsheet is providing correct results with one set of input data, hidden errors can mean that when inputs change, incorrect values result. A systematic and automated method of error detection could serve to reduce error rates and make spreadsheets more reliable.

A first step in developing any type of automated error detection system is to characterize the types of errors that can occur. To this end, Rajalingham et al. (2000) have proposed the most elaborate taxonomy for spreadsheet errors. In their taxonomy, errors are broadly categorized as system-generated and user-generated. User-generated errors are further decomposed into qualitative and quantitative errors. Quantitative errors are numerical errors that lead to incorrect bottom-line values, as opposed to qualitative errors, which do not immediately produce incorrect numeric values but degrade the quality of the model.

Quantitative errors are further subdivided into Accidental errors (due to typing errors), Omission errors (failure to consider one or more important parameters), Alteration errors (making changes to the model) and Duplication errors (re-creating elements of the model). They could also fall

into the categories of Domain Knowledge errors (stemming from a lack of knowledge) or Mathematical Representation errors (due to inaccurate construction of a formula) or Logic/Syntax errors (due to erroneous logic or syntax).

Qualitative errors are generally trifurcated into Structural errors (resulting from flaws in the design or lay-out of the model), Temporal errors (from the use of data which has not been updated), and Maintainability errors (from spreadsheet features which make it difficult to be modified). An extremely common maintainability error is the so-called **hard-coding error**. Hard-coding errors occur when raw numerical values are used in a formula instead of referenced variables. The term "hard-coded" applies because it renders the formula, and hence the whole spreadsheet model, less flexible to changing values in future scenarios.

Hard-coding errors, which are qualitative errors, can also result in duplication errors, which are quantitative errors. The typical hard-coding duplication error occurs when a constant assumed value is hard-coded. This causes two possible negative consequences. First, the constant buried in a cell formula is easily forgotten when updating a spreadsheet. For example, if current diesel prices are hard-coded into a cell, this cell will contain a non-updating assumption value and will cause the workbook to provide erroneous results if a user forgets to update this value. Second, users may use different numerical values for the same constant in different cells. For example, 3.8 L/gal in one cell and 3.785 L/gal in another. These sorts of inconsistencies lead to bottom-line errors in the workbook.

Powell et al., (2009) applied a spreadsheet auditing protocol to 50 diverse operational spreadsheets, and reported that hard-coding errors were the most common (43.5 % of erroneous cells), followed by logic errors (28.6% of erroneous cells). The remaining categories including omission, duplication, and accidental errors together accounted for less than 5% of erroneous cells.

In this paper we report on the results of a spreadsheet auditing effort in which hard-coding errors were automatically identified and subcategorized, thus addressing a need for such information identified by prior workers in the field (Powell et al., 2008). We were initially unaware of the auditing and error-checking tools available in Excel (e.g., XL Analyst, http://www.codematic.net/, Spreadsheet Professional, http://www.spreadsheetinno vations.com/), and developed the code for hard-coding error detection in-house. As we used our code, we realized the importance of sub-categorizing hard-coding errors when dealing with engineering spreadsheets, and added a second program with sub-categorization capabilities. These capabilities extend beyond what is available commercially, to our knowledge. We then applied the pair of programs to multiple bioenergy-relevant spreadsheets.

2. MATERIALS & METHODS

2.A OVERVIEW

Both programs developed in this project used Microsoft Excel's built-in visual basic for applications (VBA) programming capability. The first program identified hard-coding errors and presented a summary of error statistics along with a new worksheet tab containing a detailed error report. This tab was labeled HCER (Hard-Coding Error Report). The first program also flagged error cells using shading and font bolding to make it easy for users to locate them. The second program scanned the HCER summary, and categorized the errors into four unique types.

2.B ALGORITHMS

The first program stores all worksheet names in the workbook in a string array. Worksheets that are strictly charts/graphs are automatically skipped. The program displays the worksheet count and queries the user to see if there are any protected sheets in the workbook. If there are any protected worksheets, the cell shading and bolding functions are disabled. Because the detection algorithm can be misled by worksheet names containing numbers (e.g., *"TAB_44"*), the user is prompted to enter new names for any such worksheets, and the program assigns the new names.

Any excel spreadsheet is made up of a thousands of rows and columns, with most going unused. The program uses built-in functions to find row and column bounds of data for each worksheet, thus reducing runtime significantly.

On each worksheet, the program loops through all cells in within the data bounds. Once a formula cell is found, the formula is stored in a string and parsed. If a number is encountered as the string is parsed, a check is made on the preceding element. If the predecessor turns out to be a letter, the program assumes that a cell address is specified, not an unreferenced numerical value. If this is the case, the program checks the successor string element too, skipping the successive string elements, as long as they are numbers. However, if the predecessor to a number was not a letter, a hard-coding error is flagged. A counter variable keeps track of the number of such instances. If a hard-coding error has been detected, the numerical value is checked to see if it is equal to one. If this numeral is "1" there is another counter variable that keeps track of the number of unity occurrences. The "unity-error" distinction is made because in certain formulae – such as when converting from moisture content to total solids of a biomass feedstock – the use of a numerical one is justified and not indicative of a typical hard-coding error. For example, Powell et al. (2008) state "the following formula includes the constant 1, but few spreadsheet users would consider it erroneous: Sales_06 = Sales_05*(1+growth_rate)." The loop continues until the last element of the formula string of the concerned cell. At the end of this, the two counter variables are compared. If they are equal, the cell is solely suffering from the "unity" error.

The second algorithm looks only at the faulty cells, i.e. the cells with hard-coding errors in the report produced. The second program places the formula of the faulty cell into a string, and then parses it. When it runs into a number, it employs the same strategy described earlier to distinguish valid cell addresses from unreferenced numerical values. In the latter case, the program checks for multiple possibilities, including: (1) unity errors, (2) power of 10 conversions, (3) commonly used unit conversion, or (4) other unidentified numerals. The program thus isolates any unreferenced numerical values from the formula string and places them into the relevant subcategories. Multiple instances of different sub-categories in a single cell are captured and reported.

2.C INTERFACE

A series of dialog boxes are used for the primary user interface for the first program (the second program does not require any such dialog boxes). Message boxes, input boxes and radio buttons are used as follows: (a) To display the total number of worksheets in the workbook. (b) To

respond to whether there are any protected worksheets in the workbook. (c) To display the tab names of worksheets which contain numbers. (d) To enter the new tab names for worksheets with numbers. (e) To choose the background color and font of the cells to be flagged.

2.D ERROR STATISTICS/OUTPUT

After the first program is finished running on all the selected worksheets, it displays the total number of cells checked and the number of cells with hard-coding errors in a popup box. The same error statistics, along with a list of all the faulty cells, their cell addresses and the formulae can be viewed on the Hard-Coding Error Report tab (denoted by "HCER"), which is placed as the last worksheet of the workbook. The HCER also displays the frequency of cells with hard-coding errors and the number of cells with hard-coding errors that are uniquely unity errors.

The HCER worksheet also presents a list of each error detected, sorted by worksheet, indicating both cell reference and the equation in the cell. This makes it easy for the user to assess the errors. If the only hard-coding error in a cell is a unity error, HCER displays them with a grey fill to be easily distinguished from the non-unity errors. In our experience, cells with only unity errors are less dangerous than other hard-coding errors, and graying the unity errors enables the user to rapidly scan though the report and select non-unity errors for further examination.

The second program creates a sub-categorization table at the upper right of the HCER worksheet. The sub-categorization statistics include the frequency of (1) unity errors, (2) power of 10 conversions, (3) commonly used unit conversion, and (4) other unidentified numerals. The table shows the number of instances of the different kinds of hard-coding errors for each faulty cell of the complete workbook.

2.E AUDIT OF BIOENERGY-RELEVANT SPREADSHEETS

The working code was used on the following six workbooks related to bioenergy simulation, modeling and analysis to better understand the distribution of hard-coding errors: The **Cob-Cost** workbook designed by Carol Faulhaber (MS student, Iowa State University) computes amortized grassroots capital cost of corn-cobs storage systems. The **Simple Framework for Analyzing Anaerobic Digestion (S-FAAD)** workbook, also by Faulhaber, evaluates the economic viability of anaerobic digestion using a set of operating parameters and scale factors. The **Framework for the Evaluation of Bioenergy Feedstocks (FEBEF)** was developed by Raj Raman and Katrina Christiansen (Ph.D. student, Iowa State University)to provide insight into the relative costs and lifecycle impacts of algae, switchgrass, Miscanthus, and corn. The **GREET-BESS Analysis Meta-Model (GBAMM)** prepared by Richard Plevin (Energy and Resources Group, University of California, Berkeley) compares life cycle global warming intensity estimates for corn ethanol as computed in BESS (Biofuel Energy Systems Simulator) and GREET (Greenhouse Gases, Regulated Emissions, and Energy Use in Transportation) to understand why the results from the two models are so disparate. The **Ethanol-Profitability D1-10** workbook prepared by Don Hofstrand (Extension Farm Management Specialist, Iowa State University) presents an economic model of a typical northern Iowa corn ethanol plant to help track its profitability of corn ethanol production. The **GREET Model** (Argonne National Laboratory, Argonne, IL) is a comprehensive model evaluating the energy use and emissions for diverse scenarios.

3. RESULTS & DISCUSSIONS

3.A SAMPLE RESULTS

Figures 1 and 2 illustrate output from running HCER on a sample spreadsheet. Figure 3 illustrates the output from running the sub-categorization program on a sample HCER.

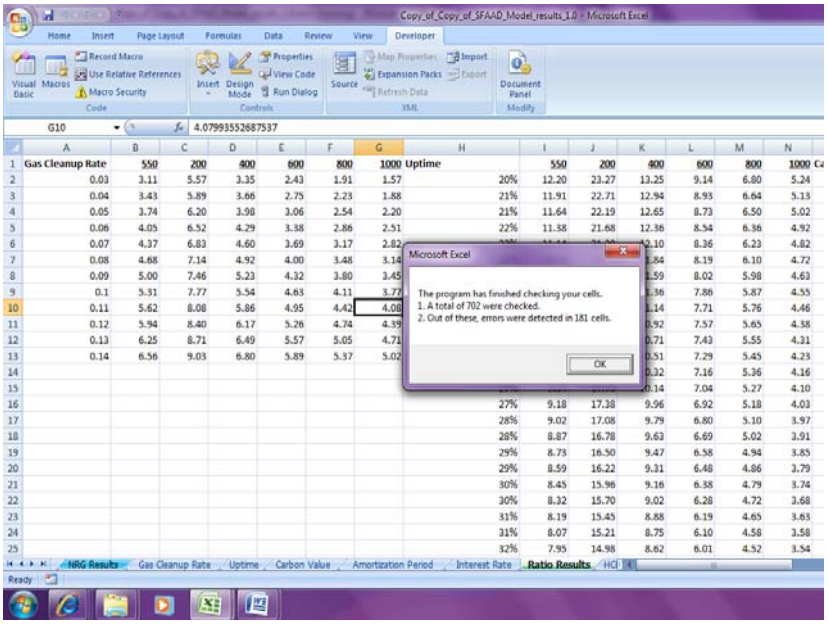
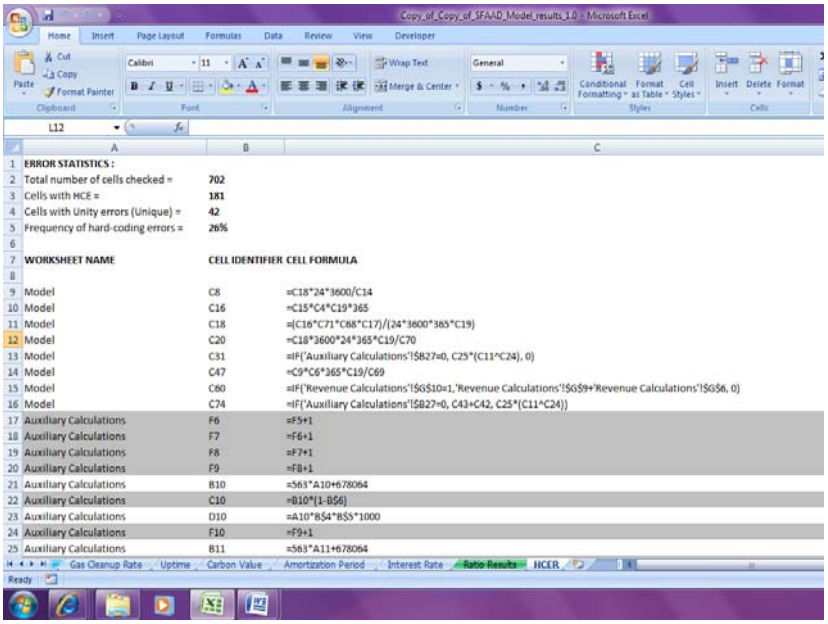


Figure 1: Screenshot of the final pop-up message box



ERROR STATISTICS :		
Total number of cells checked =	702	
Cells with HCE =	181	
Cells with Unity errors (Unique) =	42	
Frequency of hard-coding errors =	26%	
WORKSHEET NAME	CELL IDENTIFIER	CELL FORMULA
Model	C8	=C18*24*3600/C14
Model	C16	=C15*C4*C19*365
Model	C18	=(C16*C71*C68*C17)/(24*3600*365*C19)
Model	C20	=C18*3600*24*365*C19/C70
Model	C31	=IF('Auxiliary Calculations'!$B27=0, C25*(C11^C24), 0)
Model	C47	=C9*C6*365*C19/C69
Model	C60	=IF('Revenue Calculations'!$G$10=1,'Revenue Calculations'!$G$9+'Revenue Calculations'!$G$6, 0)
Model	C74	=IF('Auxiliary Calculations'!$B27=0, C43+C42, C25*(C11^C24))
Auxiliary Calculations	F6	=F5+1
Auxiliary Calculations	F7	=F6+1
Auxiliary Calculations	F8	=F7+1
Auxiliary Calculations	F9	=F8+1
Auxiliary Calculations	B10	=563*A10+678064
Auxiliary Calculations	C10	=B10*(1-B$6)
Auxiliary Calculations	D10	=A10*B$4*B$5*1000
Auxiliary Calculations	F10	=F9+1
Auxiliary Calculations	B11	=563*A11+678064

Figure 2: Screenshot of the Hard-Coding Error Report (HCER) with the error statistics at the top

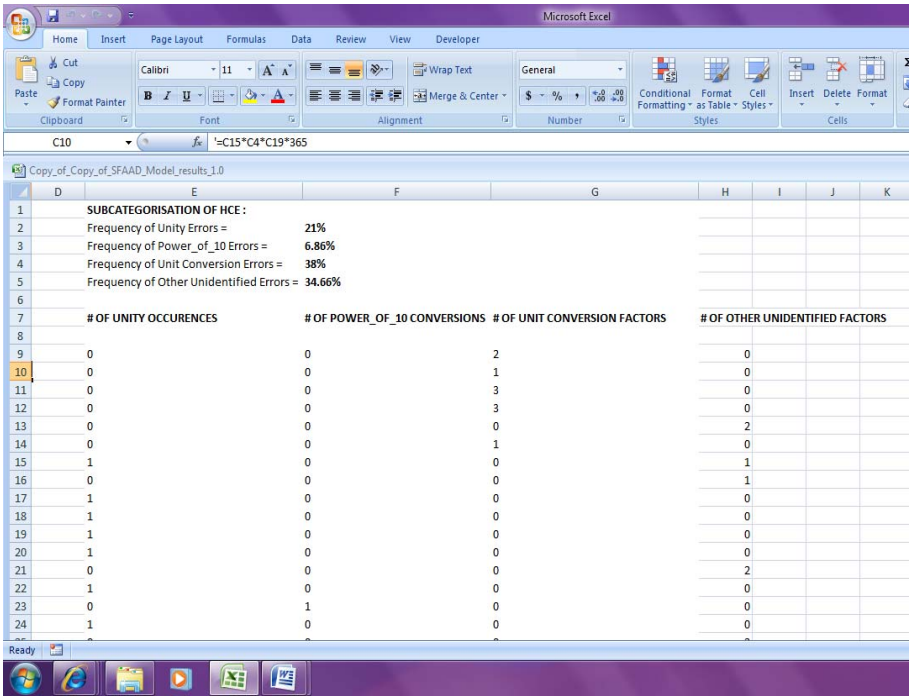


Figure 3: Screenshot of the results of sub-categorization of hard-coding errors

3.B SUB-CATEGORIZATION OF HARD-CODING ERRORS

In light of the large number of hard-coding errors detected in the sample spreadsheets, it appeared useful to further subcategorize them into the following:

- Unity errors
- Power of 10 conversions
- Commonly used unit conversion factors
- Other unidentified numerals

Although all power of 10 conversions are unit conversion factors, they form a class of their own and have an overwhelming occurrence rate compared to the other commonly used unit conversion errors. For this reason, we chose to separate them from the other commonly used unit conversion errors.

3.C AUDIT RESULTS FROM BIOENERGY-RELEVANT SPREADSHEETS

The results of the audits are shown in Table 1 & Table 2. Table 1 shows the frequency of hard-coding errors in the tested spreadsheets ranged from 11- 44%. We note that the low scoring workbook in this regard – FEBEF – originally had well over 45% of its cells as hard-coding errors; the 11% reported reflected a major effort to remove hundreds of instances of hard-coding errors. During this removal process (by one of the co-authors of this paper, Raman), a hard-coded cell was found to also contain a serious Mathematical Representation error that caused

significant errors in the bottom-line values in that spreadsheet. If we had not actively improved FEBEF based on the audit, the lowest error rate would have been 22%.

Table 1: Frequency of hard-coding errors (HCE) in the six tested spreadsheets

Workbook tested	Total number of cells checked	Number of cells with HCE	Frequency (%)
Cob Cost	203	89	44
GBAMM	462	100	22
S-FAAD	702	181	26
FEBEF	844	90	11
Ethanol Profitability	2757	608	22
GREET	66945	26867	40

Table 2 provides distribution statistics on the hard-coding errors in the six spreadsheets. The values in Table 2 are the frequencies of each type as a percentage of the total number of hard-coding errors in the respective workbooks.

Table 2: Sub-categorization of hard-coding errors from six tested spreadsheets (Each of the percentages is specific to the spreadsheet, i.e., 14% unity errors mean 14% of the total number of HCE instances in Cob Cost were unity errors)

Workbook tested	Unity errors (%)	Power of 10 (%)	Commonly used Unit Conversions (%)	Other Unidentified Numerals (%)
Cob Cost	14	14	18	54
GBAMM	7	69	0	24
S-FAAD	21	7	38	34
FEBEF	94	0	0	6
Ethanol Profitability	8	77	0	15
GREET	47	25	0	28

Both GBAMM and Ethanol-Profitability workbooks suffered from high rates of Power of 10 conversions (69% and 77% respectively). Interestingly, these two workbooks had no “other commonly used unit conversion” type errors. Reflecting the effort to rid FEBEF of power of ten and unit conversion errors, unity errors predominate in FEBEF (at a rate of 94%). The Cob Cost, S-FAAD and GREET had unity errors exceeding 14%, while the Ethanol-Profitability Workbook had fewer than 10%, perhaps because it is a financially oriented spreadsheet where the unity errors that typify engineering equations are not as prevalent. Other unidentified numeral hard-coding errors formed a significant mass of errors with a frequency ranging from 6% to 54%. Other unidentified numeral errors are hard-coding errors that contain numerical values other than unity, factors of 10, and the commonly identified unit conversion factors in the program. In some instances, their frequency even exceeds those of the other commonly used unit conversion factors put together. Future versions of this code should have an option for users to specify additional conversion factors heavily used in their spreadsheets.

Conclusion

The frequency of cells with hard-coding errors in the bioenergy-related spreadsheets ranged from 11 – 44%. This is a high error rate, especially since each occurrence is an opportunity for more serious numerical errors. To reduce the frequency of hard-coding errors, spreadsheet authors can create an "Assumptions" tab listing necessary conversions and naming each conversion with a brief but descriptive moniker (e.g., "Acresperha"). By systematically using these named factors in equations, the vast majority of hard-coding errors can be eliminated. Odd factors, such as molecular weights, can similarly be removed, but the cost-benefit ratio is questionable. Having a small fraction (e.g., less than 1%) of cells with such errors is probably not a major problem for most spreadsheets. Along with structuring spreadsheets to make computations easy to follow, and clearly listing units on all quantities, elimination (or at least minimization) of hard-coding errors must be considered another fundamental part of good spreadsheet practices.

Acknowledgements

This material is based upon work supported by the USDA Higher Education Challenge Grant Award #2006-38411-17034.

References

Galletta, D.F., K.S. Hartzel, S.E. Johnson, J.L. Joseph, and S. Rustagi. 1997. Spreadsheet Presentation and Error Detection: An Experimental Study. *Journal of Management Information Systems.* 13(3): 45-63.

Powell, S.G., K.R. Baker, and B. Lawson. 2008. A critical review of the literature on spreadsheet errors. *Decision Support Systems*. 46: 128-138.

Powell, S.G., K.R. Baker, and B. Lawson. 2009. Errors in operational spreadsheets. *Journal of Organizational and End User Computing*. 21(3): 24-36.

Rajalingham, K., D.R. Chadwick, and B. Knight. 2008. Classification of Spreadsheet Errors. *Computing Research Repository*. abs/0805.4224.