Technical Report: Preliminary Development of Computational Sensitivity Analysis

Lorie M. Liebrock† and Radha Krishna Reddy Mudhiganti

†liebrock@cs.nmt.edu

Computer Science Department

New Mexico Institute of Mining and Technology, 801 Leroy Place, Socorro, NM 87801

Keywords: Sensitivity Analysis, Mathematical and Computational Sensitivity Analysis, Sensitivity Analysis Tools

Abstract

This paper presents the use of sensitivity analysis in computer science, aiming to improve the stability in real-time computer applications (Power Grid, Nuclear Reactors or any control systems). Concepts of sensitivity analysis are used to study the uncertainty in computer programs due to small perturbations of the parameters. The ideas of mathematical and computational methods of sensitivity analysis are discussed in detail. Some of the advantages and disadvantages of both these methods are addressed. A user-friendly GUI tool is developed to compute sensitivities of application codes using VB.NET, C, and Visual Basic Applications for Excel macros. The tool was developed based on the computational method of sensitivity analysis. Details of how this tool works, the components of the tool, and the limitations of the tool are also addressed in the later sections of the paper. Sensitivities of some computational problems were computed to validate the tool. Finally, the sensitivities of a dynamic power grid system were computed using the tool.

1. Introduction

Several models have been developed from time to time to perform simulation and/or control tasks. These models can either be mathematical or computational depending on their purpose. The models are used to gain some insight into possible outcomes, which are later used in developing real time applications (Power Grid Systems, Nuclear Reactors, etc). However, before these models are used in the real world, there is a need to study their sensitivity and stability.

There are some real time applications, which are controlled a computer programs and there is a need to study how far these programs are stable, efficient and accurate before they are used. We can use sensitivity analyses to study these issues.

This paper addresses the use of sensitivity analysis to investigate the stability in computer programs, which are used to control real time systems. Stability here refers to the behavior of the program. This behavior could be explained as follows:

If there is a small change in the input, then there are two possible outcomes:

1) *there is a relatively small change in the output, which is acceptable; or*
2) *there is a drastic change in the output, which in turn can lead to two results:*
   a. *this change is acceptable (this could correctly model reality), or*

b. *this change is unacceptable; it does not correctly model reality and this indicates that there is something wrong in the computation.*

Sensitivity analysis is used to examine the behavior of the program. Even if there is a drastic change in the output for a small change in the input, because of the behavior of the system itself, this huge change in the output may correctly model the behavior of the system. Hence, there is a need to study the application to check this behavior. On the other hand, if the behavior of the application corresponds to 2b (as described above), then the program is unstable and the code needs to be revised until the study shows us that it is stable. In the other cases, the code is considered to be stable for the tested input.

Given a model, a mathematical sensitivity analysis study can be done to decide whether the mathematical model is stable or not. For computational procedure, the behavior of the program must be determined and some computations must be performed to decide whether the program is stable or not.

Once the sensitivity analysis is done, we can use the results for various purposes, such as for ranking the inputs and parameters in order of their relative sensitivity in the output, for assessing changes in the output due to parameter and input variations, improving the quality of the computations, or for limiting the use of the program to regions where it is stable.

In the following sections, sensitivity analysis is introduced, three types of sensitivity analysis are discussed, a graphical computational sensitivity analysis tool is presented, validation of the computational sensitivity analysis tool is shown using sample problems, and finally the tool is used to perform the analysis of a dynamic power grid system.

2. Sensitivity Analysis

Sensitivity analysis is an important component in building mathematical, computational, and simulation models. The values of model parameters, the computations, and the input values of variables are prone to many sources of uncertainty. It is necessary to understand the sensitivity of the model outputs to the perturbation in the model inputs [1]. Thus mathematical and computational concepts of sensitivity analysis can be used in the growing field of numerical simulation to study models [2].

Sensitivity analysis experiments may be performed on mathematical and computational models to determine the sensitivity of model outputs to the uncertainty of input variables, computations, and parameter values [1]. By conducting these experiments, we get to know which parameters or input values have the greatest effect on the model outputs.

By performing sensitivity analysis on an application we gain insight into its stability. If the results from the analysis determine that the application is not stable, then there is a need to revise the design of the algorithm and computations involved. There may also be a need to modify the underlying mathematical model, as it may be at fault.

There are different methods of sensitivity analysis, but since none of them is a clear choice for all applications, it is necessary for us to understand each method before applying them to any model. When choosing a method for sensitivity analysis, our main concern is gaining insight into the behavior of the model or computational procedure.

Two methods of sensitivity analysis are briefly described in the following sub-sections.

2.1 Mathematical Sensitivity Analysis

One approach to mathematical sensitivity analysis is to compute partial derivatives of one or more outputs (dependent variables) with respect to one or more inputs (independent variable) in a model. This of course assumes that the partial derivatives exist in some sense. Mathematical models are often expressed as systems of equations, which relate the dependent variables to the independent variables. Hence, mathematical stability analysis can be described as: "Given an equation, the variability of the output (dependent variable) with respect to perturbations in the input (independent or free variable) is measured". Sensitivities for a computer application can be viewed as the partial derivatives of outputs with respect to inputs.

Most computer programs are developed based on some mathematical equations, which can be called the governing equations. These equations relate the outputs to the inputs. Sensitivities of the computer application can be estimated by computing partial derivatives of the governing equations in that application with respect to the inputs. Note that this really is an analysis of the model, not an analysis of the computer program.

Let us consider a mathematical model determined by the following equation to understand the basic idea of mathematical sensitivity analysis:

$$y = F(x,t)$$

Where *y* represents the dependent variable and x represents an independent variable, *t* stands for time and *F* is the governing equation relating *y* to *x* on which the application was developed. We can now compute the sensitivity of x to y by calculating the partial derivative of the governing equation with respect to x during each time step [7, 8]:

$$\frac{\partial y}{\partial x} = \frac{\partial F(x,t)}{\partial x}$$

The differential of y is $\delta y = \frac{\partial y}{\partial x}\Delta x$. So, $\delta y = \frac{\partial F}{\partial x}\Delta x$, where $\frac{\partial F}{\partial x}$ is the amplification factor, i.e., if there is a change, Δx, in x, then the change in y, i.e., δy, will be Δx times the amplification factor.

Here are two examples showing how to obtain the amplification factor.

Example 1: For y = G(x) = 0.4*x + c, the derivative of G with respect to x is:

$$\frac{\partial G}{\partial x} = \frac{dG}{dx} = 0.4,$$

which results in an amplification factor of 0.4.

Example 2: For y = G(x) = 2x + c, the derivative of G with respect to x is:

$$\frac{\partial G}{\partial x} = \frac{dG}{dx} = 2,$$

which results in an amplification factor of 2.

Let's now consider a better example that would illustrate the detailed concepts of mathematical sensitivity analysis:

The fund growth model is expressed by equation (1).

$$f_{n+1} = f_n + \rho f_n + p\,, \qquad (1)$$

where f_0 is the initial fund value, $\rho = r/12$, and r is the annual percentage rate. By replacing $1+\rho$ with R, where R > 1, we get:

$$f_{n+1} = Rf_n + p \Rightarrow f_n = R^n f_0 + p\frac{R^n - 1}{R-1}\,. \qquad (2)$$

We derive equation 2 by the following steps:

$$f_{n+1} = Rf_n + p$$

for all $n \geq 0$ and

$$f_n = Rf_{n-1} + p$$

for all $n \geq 1$. Therefore,

$$f_{n+1} = R(Rf_{n-1} + p) + p = R^2 f_{n-1} + p(1+R)$$

for all $n \geq 1$. Continuing in this manner leads to

$$f_{n+1} = R^{n+1} f_0 + p(1 + R + R^2 + \ldots + R^2)\,.$$

Then sum the geometric series to get (since R>1).

$$f_n = R^n f_0 + p\frac{R^n - 1}{R-1}\,.$$

Next consider the perturbed equation $f_{n+1} + \delta f_{n+1} = R(f_n + \delta f_n) + p$. Subtracting the unperturbed equation (2) from the perturbed equation results in the perturbation propagation equation: $\delta f_{n+1} = R\delta f_n \Rightarrow \delta f_n = R^n f_0$. This model is not absolutely stable because of the exponential form of growth. However, note that we do have relative stability because

$$\frac{\delta f_n}{f_n} = \frac{R^n \delta f_0}{R^n f_0 + p\dfrac{R^n - 1}{R-1}} = \frac{\delta f_0}{f_0 + \dfrac{p}{R-1} - \dfrac{p}{R^n(R-1)}}$$

As $n \to \infty$,

$$\frac{\delta f_n}{f_n} \to \frac{\delta f_0}{f_0 + \dfrac{p}{R-1}} \quad and \quad \left| \frac{\delta f_0}{f_0} \frac{\delta f_0}{f_0 + \dfrac{p}{R-1}} \right| < \left| \frac{\delta f_0}{f_0} \right|.$$

assuming f_0>0, R>1, and p>0, this shows that the fund growth model is relatively stable.

2.1.1 Limitations of Mathematical Analysis

There are computer applications for which we don’t have a simple system of governing equations relating the dependent variables directly to the independent variables. Agent based computation often falls into this category. Variables may be related indirectly and in a very complicated fashion. Further, many computer applications do not have simple mathematical equations. Instead they are based on logical relations and inequalities (e.g., control flow statements). Mathematical sensitivity analysis involving the partial derivatives of the equations with respect to the independent variables may not be suitable for use where the dependent variables are not explicitly related to the independent variables in a simple way.

Most computer applications are developed on a step-by-step basis, where the designer/programmer keeps modifying the algorithm (the computational procedures) until the desired output is obtained. During the course of modification, he (or she) may end up with a complicated program. Applying mathematical analysis to such applications may be expensive and time-consuming and very complex. Further, the computational scientist may not be able to write down the set of governing equations by the time development is complete.

2.2 Automatic Differentiation

To compute sensitivities for a given computer application there are many automated differentiation tools available. These tools take the code of the application as input and produce the derivative of the application; on executing the derivative version of the code we can get the sensitivities of the application. There are some drawbacks associated with those tools. In particular, automatic differentiation tools such as ADIC and ADIFOR can not differentiate agent based computations. In those applications, individual agent behaviors are separately coded and often influence each other only indirectly.

2.3 Computational Sensitivity Analysis

We have already seen that mathematical sensitivity analysis cannot be easily applied directly for some computer applications. These limitations of mathematical sensitivity analysis lead us to another method of sensitivity analysis called computational sensitivity analysis. In this method, the sensitivities are calculated using numerical procedures based on computational approximations, most of which use concepts from the area of finite difference methods. Sensitivities are calculated at one or more points in the parameter space. A small perturbation of the inputs, such as a change of plus or minus one percent, can be used to assess the corresponding change in the model output [7].

Consider the computer application that models the equation given below to understand the idea of computational sensitivity analysis:

$$y = F(x,t)$$

A finite difference equation to model the above equation is:

$$\frac{\partial y}{\partial x} \approx \frac{\Delta y}{\Delta x} = \frac{F(x+\Delta x,t) - F(x,t)}{\Delta x}.$$

The change in input value (Δx) is to be very small. The procedure to compute the sensitivities requires the program to be executed for each input value, i.e., for x and x + Δx. By taking the difference between the base case and a perturbed output, then dividing the result by the change in input, Δx, we get an approximate sensitivity of the application with respect to the input x [7].

Hence, there are three steps in computational sensitivity analysis:

1) Base case: In this case, run the code with the original fixed inputs and record the corresponding outputs.
2) Comparison case: Next, vary selected input parameters one (or more) at a time, rerun the code with the changes, and record the corresponding changes in the outputs calculated by the code.
3) Compute sensitivity: Compute the approximate amplification factor or sensitivity, as discussed next.

Let's consider the same example of the fund growth model to get the idea of how to do computational sensitivity analysis.

The following equation shows fund growth evolution:

$$f_{n+1} = f_n + \rho f_n + p \,. \tag{3}$$

The computational evolution equation can be expressed as:

$$f_{n+1} = f_n + hG_n \,, \text{ where } G_n = G_n(f_n, p) \equiv (\rho f_n + p)/h \,. \tag{4}$$

The perturbed propagation equation is therefore:

$$\delta f_{n+1} = \delta f_n + h\delta G_n \,. \tag{5}$$

Using the chain rule

$$\delta G_n = \frac{\delta G_n}{\delta f}\delta f_n + \frac{\delta G_n}{\delta p}\delta p \tag{6}$$

and combining equations (5) and (6) leads to

$$\delta f_{n+1} = \delta f_n + h\left[\frac{\delta G_n}{\delta f}\delta f_n + \frac{\delta G_n}{\delta p}\delta p\right]. \tag{7}$$

Next apply the least mean square fit method to fit an equation to the output of the code and compute the coefficients of a linear mathematical model. This reduces equation (7) to the following following:

$$\delta f_{n+1} = \delta f_n + h\left(C_f \delta f_n + C_p \delta p\right), \text{ where } C_f \approx \frac{\delta G_n}{\delta f} \text{ and } C_F \approx \frac{\delta G_n}{\delta p}\,. \tag{8}$$

We can simplify equation (8) to:

$$\delta f_{n+1} = \left(1 + hC_f\right)\delta f_n + hC_p \delta p$$

Where $\hat{R} = (1 + hC_f)$ & $\hat{p} = hC_p \delta p$ and

$$\delta f_{n+1} = \hat{R} \delta f_n + \hat{p} . \tag{9}$$

Note that equation (9) is similar to the fund grow equation (2).

Thus, by solving the above linear equations we compute the corresponding coefficients C_f and C_p. These computed values are computational amplification factors that indicate the growth or decay of the model. Finally, the model inputs responsible for the largest changes in the outputs are then classified to be the most sensitive and hence the most important to study.

2.3.1 Limitations of Computational Analysis

To perform the computational sensitivity analysis, efficient computational models are needed, which take the outputs recorded during each run to analyze the application. Developing and implementing such computational models may require substantial time and effort. Thus, the costs associated with sensitivity analysis may sometimes be high [3].

The number of model runs can sometimes be very large, i.e., of the order of many thousands, resulting in substantial computational demands. For complex models the large amount of computing time needed by such runs restricts the scope of this sensitivity analysis procedure. In practice, this means, that the analyzer can investigate only the parameters that he considers to be of greatest importance in specific input ranges.

Computational sensitivity analysis does not guarantee stability for all the values of the inputs. It provides a measure of stability around the values used as inputs for the test experiments of the application. Therefore the range of values tested should include the range of applicability of the application and no conclusions can be made regarding global sensitivity [7].

There are many automatic differentiation tools available in the market, which can be used to compute the sensitivities of an application. Some of them are efficient, can be used to calculate accurate sensitivities and some are easy to work with. However, available automated differentiation tools cannot be used for all kinds of applications. For example, if an application does not have an explicit relationship between the inputs and outputs or if the computations in an application are based on just simple control flow statements. Further, some of the tools can only be used for applications developed using few programming languages like C/C++ [14, 15] or FORTRAN [16].

3. Computational Sensitivity Analyzer

In this section, we discuss a user-friendly GUI tool (see Figure 1), which can be used to compute sensitivities of a given application. Considering the limitations of mathematical sensitivity analysis and automatic differentiation, a more user-friendly computational sensitivity analyzer tool was developed, which can be used over a wide range of applications. The details of the tool are discussed below.

Components of the Analyzer tool

This tool to compute the sensitivities of the given application uses the computational method of sensitivity analysis discussed above. The different components of the tool are listed below and the functionality of each component is explained in the following sections. The main components of the tool are:

1. Tool Setup
2. Input Processing Stage
3. Application Execution Stage
4. Sensitivity Analysis Stage

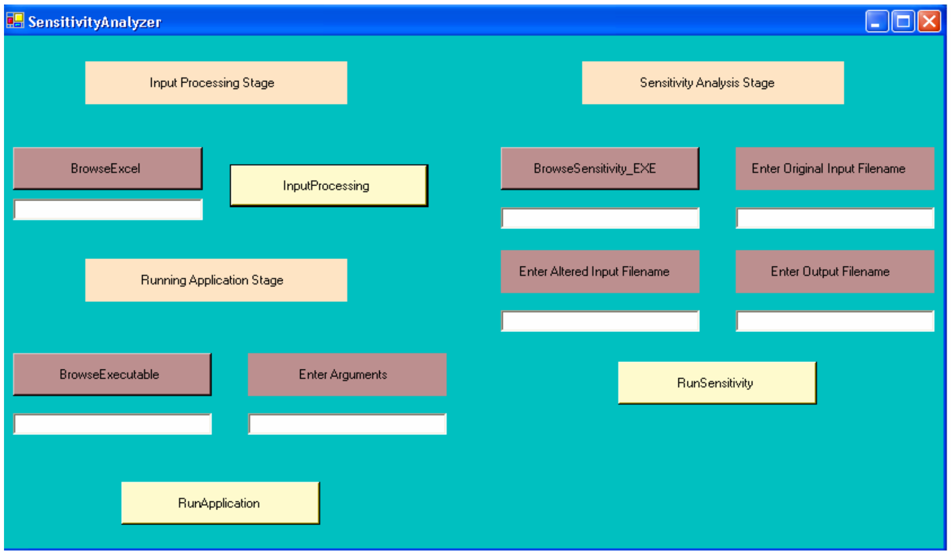


Figure 1: Snapshot of the sensitivity analyzer tool

3.1 Tool Setup

To make use of the tool in an efficient manner, it is recommended that all the files stored in the same folder. The executable for the tool, the executable for the sensitivity analysis, and the executable of the application (for which we want to find the sensitivities) must all are stored in a same folder. The original input file for the given application must be stored in the "bin" folder of the tool. Before starting to use the tool it's necessary to record the macro into the input file; the code for the macro is provided with the tool. Any output generated using the tool is stored in the "bin" folder of the tool. The tool is developed mostly using Visual Basic.NET. The code to compute sensitivities is written using C language.

There are two main functions of the macro which must be included with the Excel file, CreateBooks() and Worksheet_SelectionChange(), the code for these functions is provided with the tool.

The following procedure shows how to save a macro with a given Excel file:

Once the Excel file containing original inputs is open we need to activate the "Visual Basic Editor" window to save the macro with the file. We can open the "Visual Basic Editor" window pressing "Alt+F1" or by going to the menu bar and then selecting

“Tools/Macro” then click “Visual Basic Editor” (see Figure 2), on doing so a new window named Microsoft Visual Basic is opened. The next step is to insert a module in the same window. This can be done by going to the Menu Bar on the “Microsoft Visual Basic” window and select “Insert” from the dropdown menu and click “Module” (see Figure 3). Now a small window called “Project-VBA Project” is displayed. Select Modules, then double-click Module1 and add the code for the CreateBooks() function provided with the tool into that module. Next, copy the code for the Worksheet_SelectionChange() (see Figure 4) function provided into each sheet of the workbook containing input data. Selecting sheets from the “Microsoft Excel Objects” list in the “Project-VBA Project” window can be used to do this.

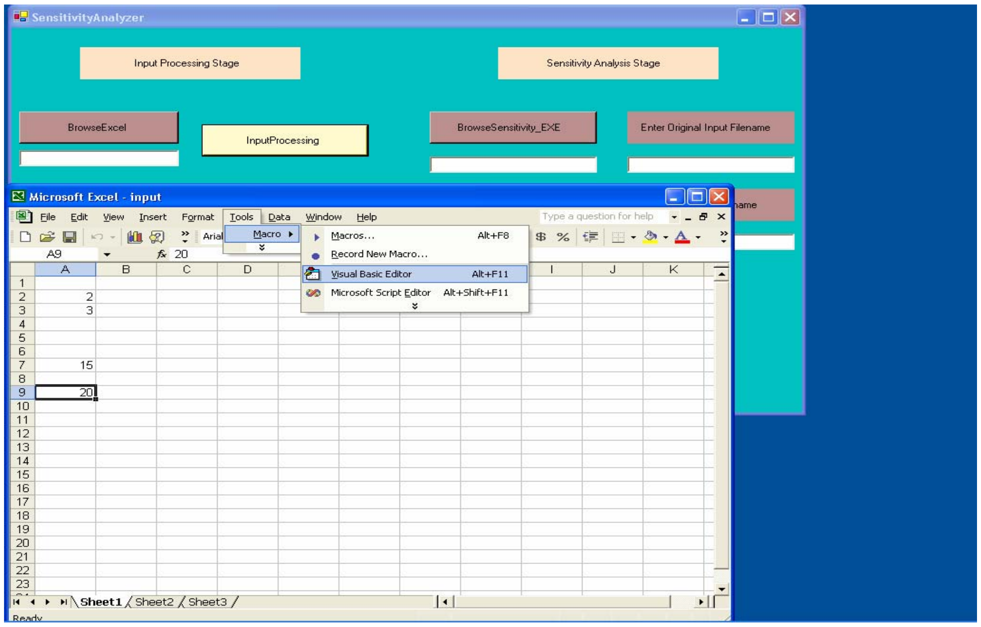
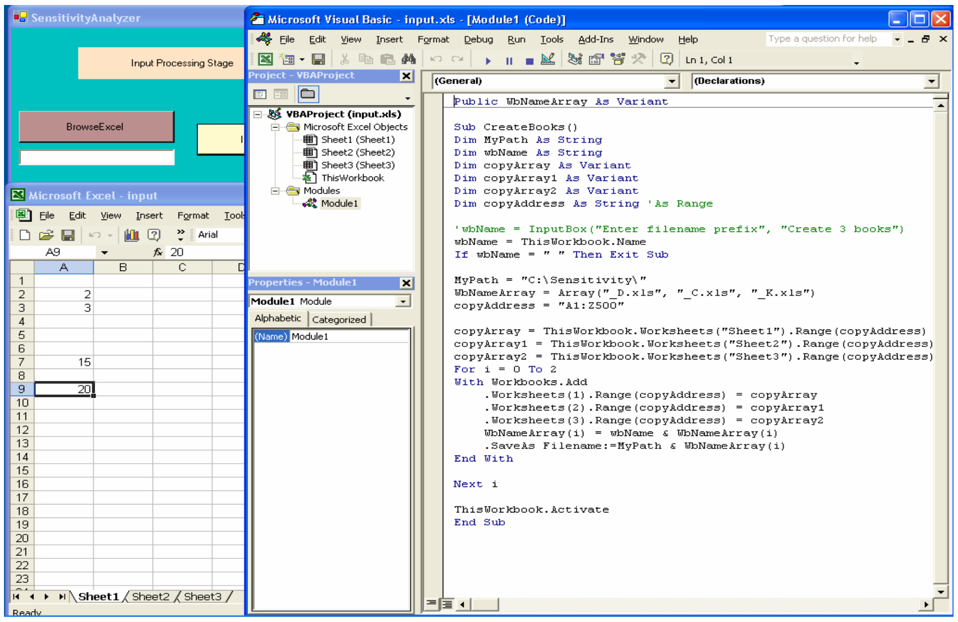
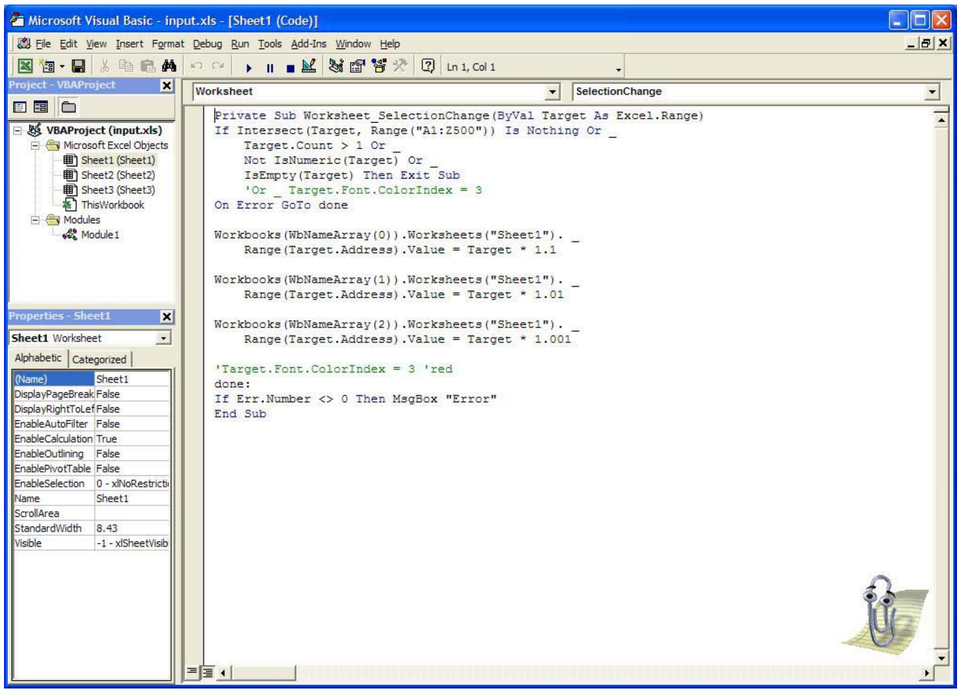


Figure 2: Opening the Visual Basic Editor



```
Public WbNameArray As Variant

Sub CreateBooks()
Dim MyPath As String
Dim wbName As String
Dim copyArray As Variant
Dim copyArray1 As Variant
Dim copyArray2 As Variant
Dim copyAddress As String 'As Range

'wbName = InputBox("Enter filename prefix", "Create 3 books")
wbName = ThisWorkbook.Name
If wbName = " " Then Exit Sub

MyPath = "C:\Sensitivity\"
WbNameArray = Array("_D.xls", "_C.xls", "_K.xls")
copyAddress = "A1:Z500"

copyArray = ThisWorkbook.Worksheets("Sheet1").Range(copyAddress)
copyArray1 = ThisWorkbook.Worksheets("Sheet2").Range(copyAddress)
copyArray2 = ThisWorkbook.Worksheets("Sheet3").Range(copyAddress)
For i = 0 To 2
With Workbooks.Add
    .Worksheets(1).Range(copyAddress) = copyArray
    .Worksheets(2).Range(copyAddress) = copyArray1
    .Worksheets(3).Range(copyAddress) = copyArray2
    WbNameArray(i) = wbName & WbNameArray(i)
    .SaveAs Filename:=MyPath & WbNameArray(i)
End With

Next i

ThisWorkbook.Activate
End Sub
```

Figure 3: Macro for CreateBooks() function



```
Private Sub Worksheet_SelectionChange(ByVal Target As Excel.Range)
If Intersect(Target, Range("A1:Z500")) Is Nothing Or _
    Target.Count > 1 Or _
    Not IsNumeric(Target) Or _
    IsEmpty(Target) Then Exit Sub
    'Or _ Target.Font.ColorIndex = 3
On Error GoTo done

Workbooks(WbNameArray(0)).Worksheets("Sheet1"). _
    Range(Target.Address).Value = Target * 1.1

Workbooks(WbNameArray(1)).Worksheets("Sheet1"). _
    Range(Target.Address).Value = Target * 1.01

Workbooks(WbNameArray(2)).Worksheets("Sheet1"). _
    Range(Target.Address).Value = Target * 1.001

'Target.Font.ColorIndex = 3 'red
done:
If Err.Number <> 0 Then MsgBox "Error"
End Sub
```

Figure 4: Macro for Worksheet_SelectionChange() function

3.2 Input Processing Stage

The initial process of creating three different sets of inputs from the given original input file is done during the input processing stage of the tool. The user can browse for the original input file just by clicking a button called "BrowseExcel" (see Figures 5 and 6). To ease the process of altering the give input, the tool requires the inputs for the application to be read from an Excel spreadsheet. The computational method of sensitivity analysis states that the inputs must be altered by a very small value. To do so, a macro for the Excel spreadsheet is available to modify the original inputs (see Figures 3 and 4). The original input file is opened using the "InputProcessing" button (see Figure 7), then to activate the macro attached to it, the user must run the macro which can be done by pressing "Alt+F8" on the keyboard (see Figure 8). The macro when activated creates three new input files from the original input file. Now, upon clicking a cell in the original input sheet, the value in that cell is altered in three different ways:

1. increase the value in cell by 0.1%
2. increase the value in cell by 1%
3. increase the value in cell by 10%

The computed new values are stored into three different files created earlier by the macro for each variation in the input, the values in the cells, which have not been clicked are just copied into all the new Excel spreadsheets. So finally there are three different files to use as input for the application, in addition to the original input file. Next run the application with one input file at a time and save the results from each run of the application into a new text file. These files will be later used to compute the sensitivities of the application.

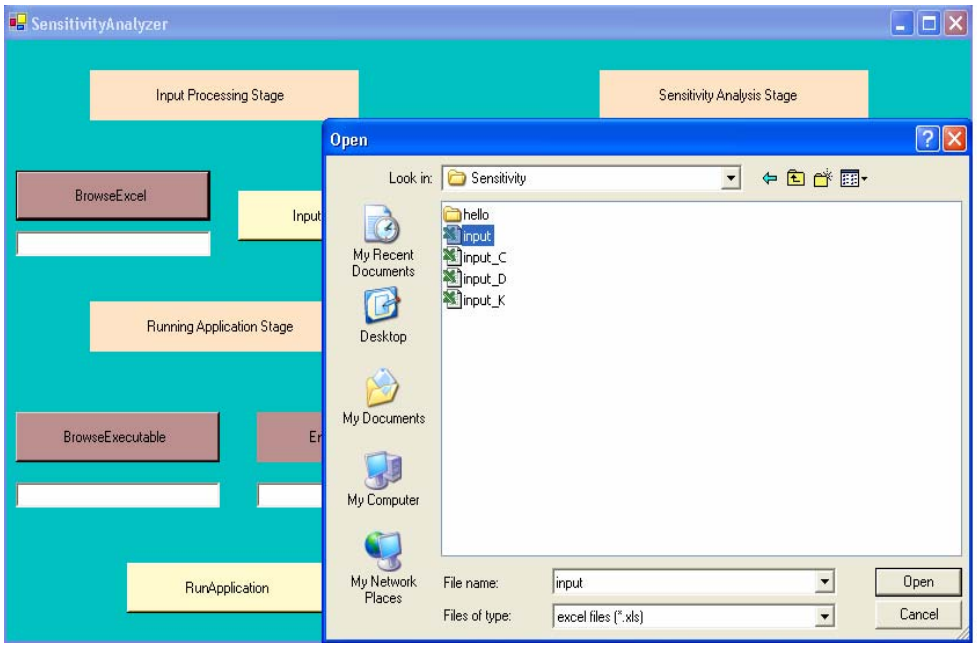
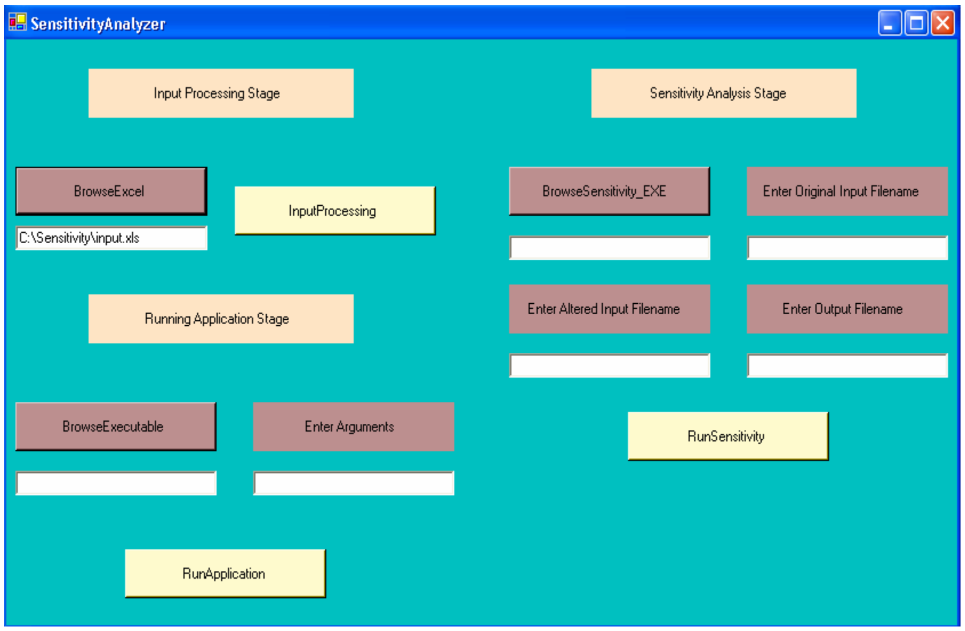
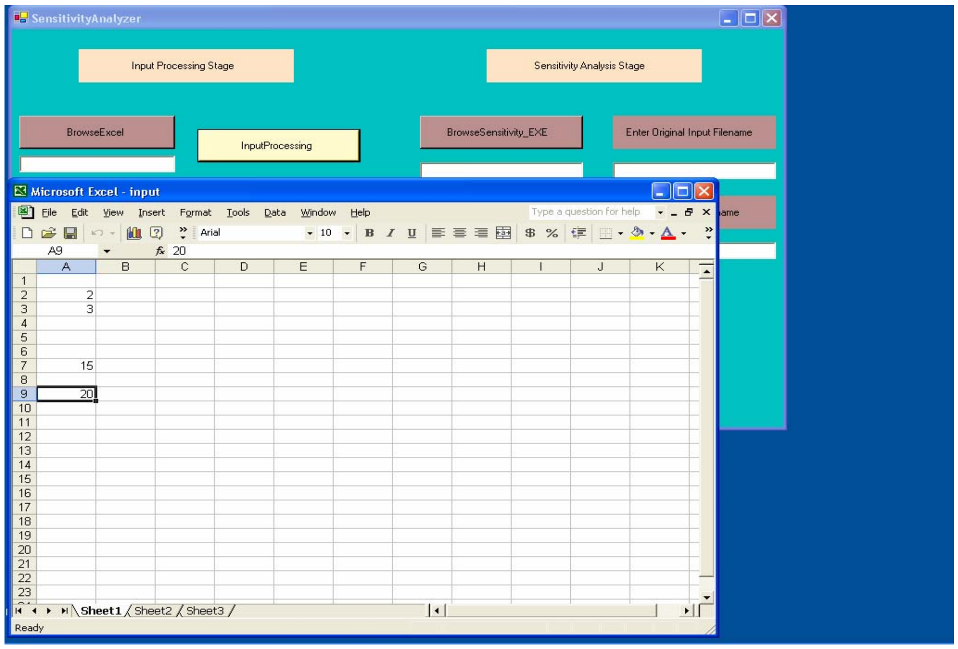


Figure 5: Opening input Excel file



C:\Sensitivity\input.xls

Figure 6: Snapshot of tool after selecting the input Excel file



A9 = 20

	A
1	
2	2
3	3
4	
5	
6	
7	15
8	
9	20

Figure 7: Selected input Excel file

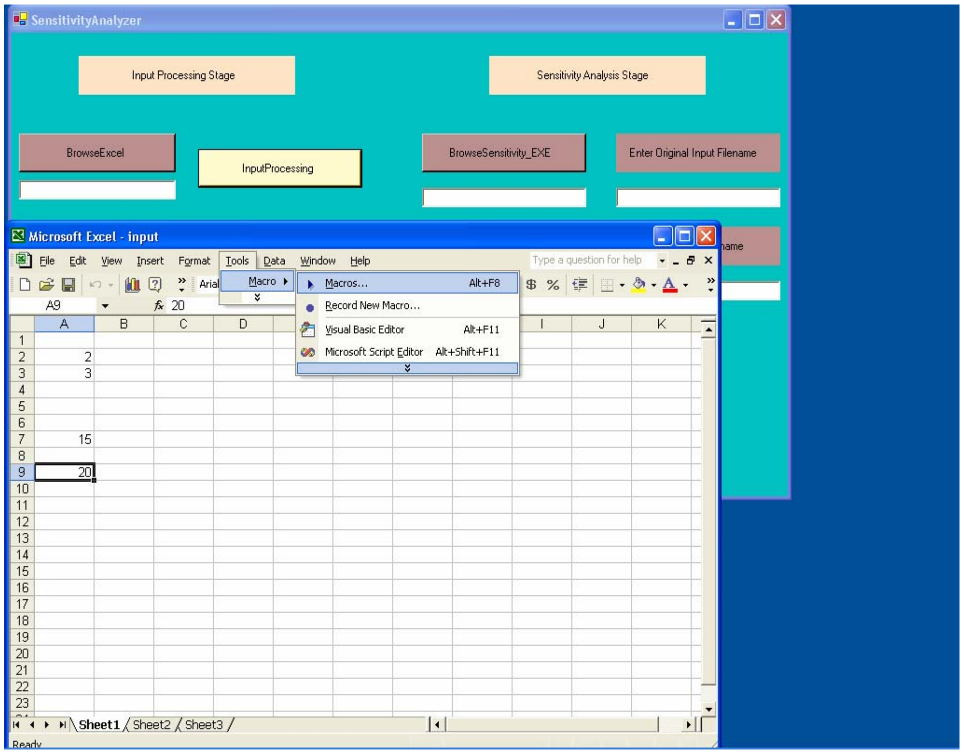
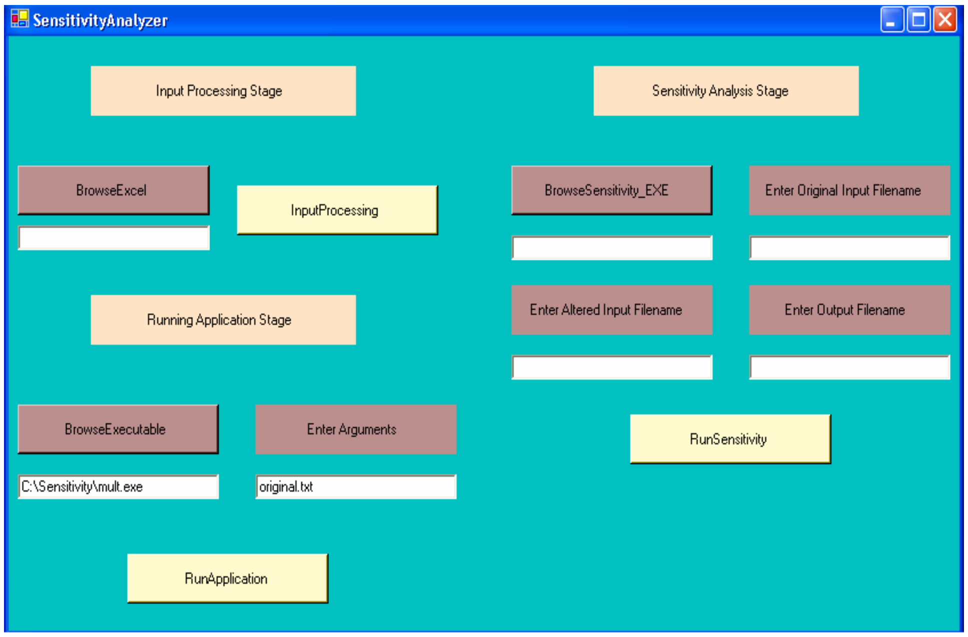


Figure 8: Running the recorded macros

3.3 Running Application Stage

This component of the tool is where the application is executed taking one input file at a time. We have four different input files created by the previous component, i.e., “InputProcessing”. To run the application the user first generates the executable and stores it on the system. The user clicks the “BrowseExecutable” button to search for the executable. Once it is selected, the path is displayed in a text box below the button. If the executable takes arguments, then the user can enter them in the text box labeled “Enter Arguments”. Clicking the “RunApplication” button begins execution of the application (see Figure 9). This procedure mentioned is followed to run the application with each input file. After each run, the results must be saved into text files, which are later used to compute the sensitivities of the application. For further processing, the resulting output files must be saved in the “bin” folder of the tool.



C:\Sensitivity\mult.exe

original.txt

Figure 9: Snapshot showing how to run a given application

3.4 Sensitivity Analysis Stage

This is the part of the tool where the actual sensitivities are computed using the outputs generated during the various runs of the application. To compute the sensitivities, the executable of the sensitivity analysis code takes three arguments. The arguments are as follows:

1. The output file containing the results obtained when the application was executed with the original inputs.
2. The output file containing the results obtained when the application was executed with the altered inputs.
3. The name of the text file where the final sensitivities of the application are to be stored.

The user can select the executable for the sensitivity analysis code by using the "BrowseSensitivity_EXE" button. The name of the files to be passed on to the executable as arguments must be entered into the text boxes with appropriate labels (see Figure 10). Now, by clicking on the "RunSensitivity" button the program to compute the sensitivities of the application is executed and the results are stored in the "bin" folder of the tool. By repeating the process for each perturbed input file, the user can compute the corresponding sensitivities.

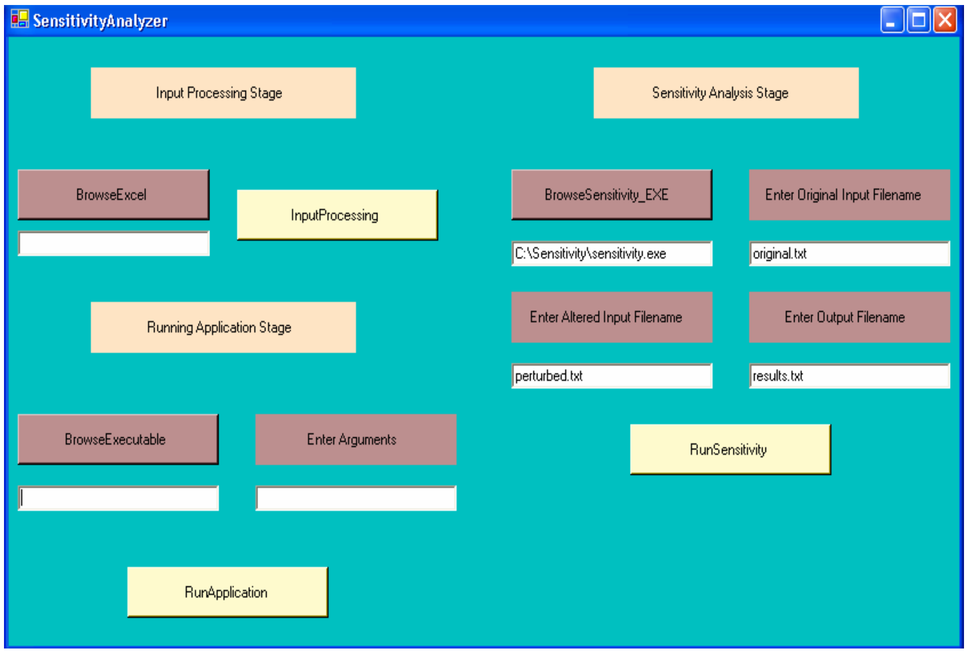


Figure 10: Snapshot showing the sensitivity analysis stage

3.5 Limitations of the Tool

In this section we list some of the limitations of the Sensitivity Analyzer tool described above.

- As discussed in the preceding section, the tool is currently developed to process the input files, which are in Excel format. In the future, it should be extended to process more standard formats (e.g. text, data files).
- The tool computes sensitivities of a given application based on the concepts of computational sensitivity analysis. Different methods of computing sensitivities could be added to the tool and chosen with the browse button from the available list of executables to compute sensitivities. One such extension would involve developing other fits besides the least mean square fit.
- Finally, the limitations mentioned for the computational method of sensitivity analysis are also true for the tool.

4. Sensitivity Analysis of Computational Problems

In this section, sensitivity results for two different computational problems are shown as computed using the tool. The two applications are the fund growth model (to illustrate the comparison to mathematical stability analysis) and the Power System.

4.1 Fund Growth

In this section we provide the sensitivities for the fund growth model computed using the tool. The following equation shows fund growth evolution equation:

$$f_{n+1} = f_n + \rho f_n + p$$

Base Case: Run the fund growth model with original inputs $f = 1000$, $\rho = 0.005$, $p = 100$.

Comparison case 1: Run the model by changing "f" by 1% to use inputs $f = 1010$, $\rho = 0.005$, $p = 100$, $\delta f_0 = 10$. This results in $C_f = 0.005$ and $C_p = 0.0$ (see Equation 8).

Comparison case 2: Run the model by changing "p" by 1% to use inputs $f = 1000$, $\rho = 0.005$, $p = 101$, and $\delta p = 1$. This results in $C_f = 0.005$ and $C_p = 1.0$.

Comparison case 3: Run the model just by changing both "f" and "p" by 1% to use inputs $f = 1010$, $\rho = 0.005$, $p = 101$, and $\delta f_0 = 10$, $\delta p = 1$. This results in $C_f = 0.005$ and $C_p = 1.0$.

Each of these results is in agreement with the mathematical analysis of the growth fund model.

4.2 Power System

This system is a mathematical model developed to model the dynamic behavior of a power system. It is important to understand a system both mathematically and physically to replicate the physical system's actual behavior. A digital model of the power system has been developed to analyze the actual dynamics involved in the system. Dr. Steve Schaffer of the Mathematics Department and his research group at ICASA, New Mexico Tech developed this model at New Mexico Institute of Mining and Technology. In 1990, P. W. Sauer and M. A. Pai published a paper on power system steady-state stability [4]; the paper presents a detailed dynamic model with fundamental features of the system, where the algebraic equations that accompany the model are discussed in detail and are used to develop this digital model. In this paper, the power grid model is used as an application to perform the sensitivity analysis and test the sensitivity analysis tool.

The power system model was developed using MATLAB. The input for this model is read from an Excel spreadsheet. The inputs to the model are the dynamic states of the power system like resistances, torques, etc. The outputs are states of the system, such as voltages, angles, etc. The sensitivities of the voltage parameters with respect to a few critical input parameters were computed using the tool.

By running the power system model using a few test cases and changing one input parameter at a time by a very small value, we computed the sensitivities of the voltages at each bus. On analyzing the sensitivities, the power system model was seen to be stable because the computed sensitivities were all slightly less than 1 and there were no drastic changes in the output voltages with respect to small changes in specific inputs.

The computed sensitivities can be used to rank the critical inputs that are responsible for a drastic change in the outputs. Sensitivities can also be used to list the outputs which were most affected by the change in the input. With this information application programmers can look back into the model and track down any instability if there is any.

5. Conclusion

This paper discussed how the concepts of sensitivity analysis can be used in the field of computer science to improve the stability of computer applications. A distinction is made between mathematical and computational methods of sensitivity analysis. These methods were also categorized as to what kind of sensitivity analysis is most suitable for a given computer application. Both the advantages and disadvantages of the two different methods of sensitivity analysis were discussed in detail. The paper also addressed issues related to available automated tools, which can be used to compute the sensitivities of a computer application, and the drawbacks associated with each type of tool.

A user-friendly semi-automated sensitivity analyzer tool was developed using the concepts of computational method of sensitivity analysis. The tool can be used to compute sensitivities of an application. The tool was tested running some test cases and finally used the same tool to compute the sensitivities of a dynamic power system.

Further extensions in the field of sensitivity analysis and addition of more techniques to compute sensitivities of computer application will be added in the future. Future work will also focus on further standardizing the tool and testing it with more complex applications. Further, to validate the tool, we intend to compare the sensitivities generated by the tool for the power system with the mathematical sensitivities computed internally in the power system code. A research group at ICASA is developing internal mathematical sensitivity analysis for their power system model. Future work will use results generated by that model to compare against the results generated by the tool described in this paper.

6. References

1. Campolongo F., and R. Braddock, The use of graph theory in sensitivity analysis of model output: A second order screening method, Reliability Engineering and System Safety, 64, 1-12, 1999.
2. Saltelli A., Tarantola S., and K. Chan, A quantitative, model independent method for global sensitivity analysis of model output, Technometrics, 41, 39-56, 1999.
3. Sastry S. Isukapalli and Panos G. Georgopoulos, Computational Methods for Sensitivity and Uncertainty Analysis for Environmental and Biological Models, EPA/600/R-01-068, December 2001.
4. Sauer P.W. and Pai M.A, Power system steady-state stability and the load-flow Jacobean, IEEE Transactions on power systems, volume 5, No. 4, 1990.
5. Sauer P.W. and Pai M.A, Power system dynamics and stability, Prentice Hall, 1998.
6. Ian A. Haskins and Magnus Akke, Analysis of the Nordel Power Grid Distribution of January 1, 1997 Using Trajectory Sensitivities, IEEE Transactions on Power Systems, Vol. 14, No. 3, August 1999.
7. H. Christopher Frey, Amirhossein Mokhtari and Tanwir Danish, Evaluation of Selected Sensitivity Analysis Methods Based Upon Applications to Two Food Safety Process Risk Models, Office of Risk Assessment and Cost-Benefit Analysis, U.S. Department of Agriculture, September 2003.
8. Ian A. Haskins, Nonlinear Dynamic Model Evaluation from Disturbance Measurements, IEEE Transactions of Power Systems, Vol.16, No. 4, November 2001.

9. http://www.ubmail.ubalt.edu/~harsham/ref/RefSim.htm
10. http://sensitivity-analysis.jrc.cec.eu.int
11. ATHERTON R.W., Shainker R.B. and Ducot E.R. On the Statistical Sensitivity Analysis of Models for Chemical Kinetics AIChE Journal (1971), 21(3), 441-447.
12. Andres T., Sampling methods and sensitivity analysis for large parameter sets, Journal of Statistics Computation and Simulation, 57, 77-110, 1997.
13. Arsham H., Sensitivity and optimization of computer simulation models, Modeling and Simulation, Instrument Society of America, 19, 1835-1842, 1988.
14. ADOL-C: A Package for the Automatic Differentiation of Algorithms Written in C/C++.
15. http://www.mcs.anl.gov/autodiff/adic.html
16. http://www.mcs.anl.gov/adifor