United States Environmental Protection Agency | National Center for Environmental Assessment Research Triangle Park, NC 27711 | EPA/ September, 2008

Research and Development

Ten Berge CxT Models

External Draft Version 2.0

Prepared by

National Center for Environmental Assessment
Office of Research and Development
U. S. Environmental Protection Agency
Research Triangle Park, N C 27711

NOTICE

The U.S. Environmental Protection Agency, through its National Center for Environmental Assessment at Research Triangle Park, produced this report. This document is a preliminary draft. It has not been formally released by the U.S. Environmental Protection Agency and should not be construed to represent Agency policy. It is circulated as a guide for reviewers of the model.

TABLE OF CONTENTS

AUTHORS

Qun He, Lockheed Martin Information Technology, RTP, NC 27711

Jeff Gift, National Center for Environmental Assessment, RTP, U.S. EPA, NC 27711

Bruce Allen, Lockheed Martin Information Technology, RTP, NC 27711

SECTION 1: INTRODUCTION

This report is intended to provide an overview of beta version 1.0 of the implementation of a concentration-time (CxT) model originally programmed and provided by Wil ten Berge (referred to hereafter as the ten Berge model). The recoding and development described here represent the first steps towards integration of the ten Berge model into the EPA benchmark dose software (BMDS). At present the software can be run from a Windows Command Prompt; at a later date it will be possible to invoke the ten Berge model from the beta interface for BMDS version 2.0.

This introduction provides the "high level" background for the ten Berge model and the current implementation. It discusses the perceived needs and requirements for this implementation. . This is followed by a description of the modeling framework and criteria for implementing the ten Berge model within BMDS, i.e., it is discussed in the context of other dichotomous models that are part of the BMDS software. These discussions are intended to satisfy the reporting requirements of the Council for Regulatory Environmental Modeling (CREM, 2003) concerning the development, evaluation, and application of environmental models.

Specific ten Berge model design issues (theoretical development, mathematical formulation and identification of data and parameter value inputs and constraints) and model coding are discussed in Section 2. In subsequent sections, the method of running the software is described, with some sample input and output explained as an example of its use in a risk assessment (Section 3). The user who simply wants to understand how to get the model running can skip ahead to Section 3, perhaps returning to Section 2 for information about the options that must be specified in the input file and the various choices available for those options.

Finally, and the main purpose of the current iteration of this document, in Section 4 is a report on the testing completed to date. That testing has focused on the translation of the original ten Berge model (written in Visual Basic and running through spreadsheets) to the compiled C version that is its current form. Thus the testing has focused on ensuring that the same results are obtained in the new version as were obtained in the previous version. One of the main purposes of the document is to provide reviewers information about the testing that was done; that information, with preliminary test results, is provided in Section 4.

1.1 MODELING NEEDS AND REQUIREMENTS

CxT modeling has a long history of use in the assessment of acute exposure risks. CxT modeling is used primarily in the context of short-term exposures where both the concentration and the duration of exposure are considered important and relevant for estimating risk. Haber's "Law," which states that risk is related to CxT has been a motivation for development of CxT models, although such models (including the current implementation) have generalized the simple CxT relationships somewhat. Thus, the software under consideration here is intended to be applied to data that presents both concentration (or dose) values and durations of exposure (the time component, typically shorter-term exposure durations), as well as responses (dichotomous response rates) to estimate a concentration-time-response relationship.

The ten Berge version of CxT modeling was initially developed as a program coded in Visual Basic by Dr. Wil ten Berge. Dr. ten Berge graciously provided the source code, as well as the executable programs, to EPA so that they could be evaluated for possible inclusion in BMDS.

It was determined that the first step towards possible inclusion in BMDS would be a new version of the ten Berge model written in the C language, the language in which the other BMDS model code is written. Thus, the purpose of the task described herein was to develop models comparable to the ten Berge model, written in C, and suitable for integration into BMDS. The products of this task include the C source code, an executable program implementing that source code and yielding results that have been verified, and this manual documenting the achievement of those goals and specifying how to run the new program.

1.2 MODELING FRAMEWORK -- GENERAL OVERVIEW AND COMPARISON TO BMDS DICHOTMOUS MODELS

When the ten Berge model is incorporated into BMDS, it will represent a type of model unlike any other model currently in BMDS. That is because it incorporates more than one explanatory (independent) variable to define the dependent variable, which in this instance is the probability of some response of interest. In fact, the motivation for a model like the ten Berge model is to specifically allow both concentration (exposure level) and duration of exposure (time) to be determinants of the probability of response, hence the rubric "CxT model."

Because the response variable is probability of response, the ten Berge is most similar to the dichotomous models that are included in BMDS. Those models also predict probability of response as a function of the explanatory variable (dose only in those models). In fact, the logit and probit link functions that are options for defining the relationship between concentration and time, on the one hand, and probability of response, on the other hand, in the ten Berge model are also options for BMDS modeling of dichotomous responses (albeit without the capability to factor in the duration of exposure). Section 2 describes the mathematical relationships in greater detail.

Because the response of interest is a probability (represented by observations consisting of the number of individuals who have the response of interest out of the total number of individuals examined) just like the BMDS dichotomous models, the following description of the likelihood approach to parameter estimation, which is true of the BMDS dichotomous models is also relevant to the ten Berge model.

Model Equations

BMDS dichotomous models can all be written in the form:

$$p(dose) = g(dose; \alpha, \beta, \ldots)$$

where p(dose) is the probability of response when the exposure level is equal to "dose." The extension to the ten Berge model is that additional explanatory variables are included:

p(concentration, time, x) = h(concentration, time, x; α, β, γ, …)

where "x" represents some other explanatory variable(s) that the investigator might be interested in including. Here g(dose**;** α, β , . . .) or h(concentration, time, x; α, β, γ, …) is some function requiring the independent variables (dose or concentration, time, and x) and α, β , . . ., as inputs; α, β, … are parameters to be estimated using maximum likelihood methods. Depending on the model function chosen, the parameter values may be constrained. The options for the h(·) function for the ten Berge model are described in greater detail in Section 2.

Likelihood Function

All models in the current version of BMDS are fit using maximum likelihood methods. This is true of the ten Berge model as well. This section describes the likelihood function used to fit the dichotomous and ten Berge models.

Suppose the data set to be fit has k groups, each may have a dose level:

dose(1), dose(2), …, dose(k)

or, for CxT modeling, a set of explanatory variables such as

[conc(1), time(1), x(1)], [conc(2), time(2), x(2)], …, [conc(k), time(k), x(k)]

Suppose the total numbers of individuals in each of the groups are:

N(1), N(2), …, N(k)

If the observed numbers of individuals with the response of interest are:

n**(1), n(2), …, n(k)**

then, assuming the distribution of the n(i) is binomial, the log-likelihood of the data for a given dichotomous model is

$$L = \sum_{i=1}^{k} \{n(i) \cdot \ln(p(i)) + (N(i) - n(i)) \cdot \ln(1 - p(i))\}$$

where p(i) = g(dose(i); α, β, …) or p(i) = h(conc(i), time(i), x(i); α, β, γ, …). In the ten Berge model and in the other dichotomous models, the parameters α, β, … are estimated by finding the values that maximize that log-likelihood (perhaps subject to certain constraints on those parameters). Note that the above formulation of the log-likelihood function ignores a term that is constant (for any given data set) because it does not depend on the model parameters. Log-likelihoods reported by all BMDS models (including the current implementation of the ten Berge model) do not include that "binomial" constant.

BMD Computation

The ten Berge model allows the user to estimate concentrations associated with specified probabilities of response, for specified durations of exposure. In the context of the BMDS software, a BMD is a dose corresponding to some specified additional or extra risk value. Whenever the background rate of response is zero (either by design – e.g., using ln(conc) in the $h(\cdot)$ function – or when the background is *estimated* to be zero), additional or extra risk equal the probability of response for any concentration and time. In those instances the ten Berge model provides estimates of BMDs. In the context of the analysis of many acute exposure experiments the assumption of a zero background response is typical and appropriate and use of ln(conc) is standard.

The ten Berge model also has the capability to calculate the estimated response probability for specified concentration, time, (and perhaps other parameter inputs). Additional details about the computation of those values and how the user may request the program to do so are provided in Section 2 and 3.

SECTION 2: TEN BERGE MODEL: DESIGN AND CODING

This section provides information related to the technical and mathematical details underlying the ten Berge model. It does provide some useful information for more casual users (e.g., with respect to the transformations of the explanatory variables) but users/reviewers may want to skip forward to Section 3, which provides information on the "how-to" of running the model. Then, if one needs more information about the details of the choices outlined there, the user can obtain them from this section.

2.1 THEORETICAL DEVELOPMENT

The idea that both concentration and duration of exposure may affect the likelihood of a toxic response, especially for short-term or acute exposures, has been common for some time and can be traced to Haber's "Law." In fact, EPA's software, CatReg, is a very flexible program for modeling the probability of response as a function of concentration and durations of exposure. CatReg can handle responses that have a graded severity scale, not just the 0/1 dichotomous responses that are treated in the ten Berge model.

2.2 MATHEMATICAL FORMULATION

In the discussion above (Section 1.2), the explanatory variables for the ten Berge model were "linked" to the probability of response via a function h(·). In this implementation of the ten Berge model, that function can take two forms:

Logistic:	$h(z) = \exp(z) / (1 + \exp(z))$
Probit:	$h(z) = \Phi(z-5)$

where $\Phi(y)$ is the cumulative standard normal distribution function evaluated at y. In generalized linear models terminology, the term link function is used (e.g., McCullagh and Nelder, 1989) to denote the inverse of the h functions shown above; the **logit** link function $[\ln(p/(1-p))]$ is the inverse of the logistic function and the **probit** link function $[\Phi^{-1} + 5]$ is the inverse of the cumulative normal function with the "+5" component. The term "z-5" takes the place of the now-standard "z" in the cumulative normal function defining the probit model.[1]

The variable z in the equations above is actually a function of the explanatory variables of interest. If those explanatory variables are concentration (C), duration of exposure (T), and some other parameter (x), then

$$z = B_0 + B_1*f_c(C) + B_2*f_t(T) + B_3*f_x(x) + B_4*r_4(C, T, x) + B_5*r_5(C, T, x) + \ldots$$

1 In the original definition of the probits, the addition of 5 was a convenience, giving positive values for the probits and facilitating computation before the days of electronic computers; that addition has been retained in this implementation (because it was done so by ten Berge, following Finney, 1971, in his software and we are, at this point, attempting to replicate his calculations). It seems that there is no toxicological or other reason for probits to be positive in value.

The terms B_0, B_1, B_2, … correspond to the α, β, γ, parameters discussed above (Section 1.2), the ones that are estimated via maximum likelihood methods and which determine the magnitude of the contribution of the explanatory variables for the probability of response. The functions f_c, f_t, and f_x are chosen by the user to be one of the following:

$f_i(u) =$	u	(identity transformation)
$f_i(u) =$	ln(u)	(logarithmic transformation)
$f_i(u) =$	1/u	(reciprocal transformation)

The user may pick f_c independently of f_t and f_x; that is, the user may pick the logarithmic transformation for f_c, for example, without constraining f_t and f_x, which can then be any of the three possible transformations and might differ from one another. The $r_i(C, T, x)$ terms represent products of a pair of the values $f_c(C)$, $f_t(T)$, and $f_x(x)$; for example $r_4(C, T, x)$ might be $f_c(C)*f_t(T)$. If, for example, the logarithmic transformation of C was chosen, then the product term(s) that involved C would also have to be the logarithm of C times the chosen transformation of another parameter.

Note that the parameter "x" is actually a place holder for any number of possible explanatory variables of interest (think of x as a vector of variables). Thus, there may be many terms that could be added to the model at the user's option, including all the individual variables represented by x, the products of pairs of terms in x, and the product of those terms with C and/or T. Currently, the one limitation on the number of product terms in the model is that it cannot exceed the total number of explanatory variables divided by 2 (rounded down). Thus, if C, T and x were in the model, and x consisted of a single term, there could be no more than floor(3/2) = 1 product terms. This limitation is a carry over from the original ten Berge model coding.

Note: the user should be wary of "over-parameterizing" the CxT models by inclusion of product terms. Moreover, their interpretation may be problematic, especially in combination with some transformations of the explanatory variables. At the very least, users might want to start with models that include no product terms, and consider the ability of such models to fit the data and provide interpretable results. Caution should always be exercised when adding more complex terms to these and other models.

One particular model formulation may be of interest for acute exposure modeling, where often a $C^N xT$ relationship is postulated, where n is some unknown power on concentration (as a generalization of Haber's "Law"). If the logarithmic transformation is chosen for C and for T, then the term inside h(·) is (ignoring any contributions from other parameters represented in x)

$$
\begin{aligned}
& B_0 + B_1*\ln(C) + B_2*\ln(T) \\
= \quad & B_0 + B_2*((B_1/B_2)*\ln(C) + \ln(T)) \\
= \quad & B_0 + B_2*\ln(C^{B1/B2} * T) \\
= \quad & B_0 + B_2*\ln(C^N * T)
\end{aligned}
$$

where $N = B_1/B_2$. Thus, a model that includes log-transformed C and T will have a term that can

be equated to $C^N xT$ (log-transformed) where N can be estimated by the ratio of the linear coefficients of ln(C) and ln(T). As discussed in the next section, the BMDS coding of the ten Berge model allows the user to request that such ratios be estimated.

2.3 PARAMETER ESTIMATION

As indicated above, for the selected model, the linear coefficients (the B_i terms) are estimated by maximum likelihood techniques. The computation of such estimates requires an iterative numerical optimization technique. There are no pre-defined constraints on the parameters; the user should be careful in interpreting the outputs of any given model run to ensure that the concentration-time-response relationship entailed by the parameter estimates makes sense toxicologically (dependent on the values and transformations of the explanatory variables). For example, in the particular formulation shown above using ln(C) and ln(T), then positive values for B_1 and B_2 would be expected.

The maximization of the likelihood is done via a Taylor-MacLaurin expansion following Finney (1971), which is also known as the Fisher scoring method (Collette, 1991; Morgan, 1992). For that method, the inverse of the expected Fisher information matrix and the vector of derivatives of the log-likelihood with respect to the model parameters are multiplied and added to the previous guesses for the parameter values to determine the next step to the next iteration, i.e., new parameter estimates. The expected Fisher information matrix is defined as the negative of the expectation of the Hessian matrix of second derivatives of the log-likelihood with respect to the model parameters. The values for the vector of derivatives of the log-likelihood with respect to the model parameters at any iteration are computed using the previous parameter estimates. The procedure iterates from a starting set of parameter estimates (not under the control of the user) until the change in parameter values from one iteration to the next are sufficiently small (close to zero, where the degree of closeness is also not under the control of the user but is set equal to 10^{-6}).

Tests of the significance of the B_i coefficients (so-called Student T values) are estimated based on the variances and covariances of the B_i's. Following Finney (1971), the expected information approach is used (the variance-covariance matrix is set equal to the negative of the inverse of the expected value of the Hessian of second derivatives of the log-likelihood with respect to the model parameters). The ten Berge model output includes a chi-square value (assessing lack of fit) and a degrees of freedom associated with that chi-square value. Those values allow calculation of what is known in other contexts as an over-dispersion or heterogeneity factor. Although that factor is calculated by the ten Berge program when estimates are requested (see the following paragraph), it is *not* applied automatically, either in a "correction" of the variance and covariance estimates of the model parameters or in the calculation of the Student T values for those parameters. The user who wishes to correct the variances and covariances needs to multiply them by the heterogeneity factor (chi-square value divided by its degrees of freedom); with those corrected variances, adjusted Student T values can be computed (the parameter estimate divided by the square root of the corrected variance).

The user may also request estimates and confidence limits for the following:

A value of one of the explanatory variables, given the values of the other explanatory variables and a probability of response (e.g., a concentration that, for a given duration of exposure, yields a probability of response of interest);
The probability of response, given the values of the all the explanatory variables (e.g., the probability of response for a specified concentration and duration);
The ratio between the coefficients of two explanatory variables (e.g., the ratio of the coefficients for ln(C) and ln(T), which was shown above to give an estimate of the parameter N in the $C^N xT$ formulation).

The confidence limits in all of these cases are based on the variances estimated for the model parameters (as described above). The method follows Fieller (1944) as reported in Finney (1971): a confidence interval for an estimate, Y, is defined as $Y \pm t*sqrt(Var(Y))$. Var(Y), the variance of Y, is estimated using the Wald method for approximating the variances of functions of the model parameters.

The user is required to input the value of t for the confidence limit calculations shown above. It is not the confidence level itself, but rather is a deviate corresponding to the confidence level of interest. It is recommended that the user start with the deviates based on the standard normal distribution (see Table 1 below). That recommendation is based on the fact that use of other deviates (derived from a Student T distribution) have been justified on the basis of the heterogeneity factor discussed above (see Morgan, 1992). But in the instances where Student T-based deviates would be recommended (significant chi-squared values), one would typically scale up the variances by the heterogeneity factor as well. Since the ten Berge program does not automatically do that scaling (see discussion above), it appears more consistent to retain the use of standard normal-based deviates.

Table 1: Deviates Corresponding to Confidence Levels of Interest for Confidence Interval Estimation (from Standard Normal Distribution)

α	**Confidence Level**	**Deviate**
0.2	80%	1.282
0.1	90%	1.645
0.05	95%	1.960
0.01	99%	2.576

The user can determine the deviate to use for other confidence levels of interest from a table of quantiles of a standard normal distribution, available in many elementary statistics books (or s/he may compute them using Microsoft Excel function "NORMSINV" and putting in (1-α/2) as the argument to that function when interest is in the 100*(1-α) confidence interval).

Note: The Wald approximation that is used in the confidence limit calculations is recognized to be much less reliable than other methods such as profile likelihood and bootstrapping (Crump and Howe, 1984; Moerbeek et al, 2004; Nitcheva et al., 2007). The Wald approximation is vulnerable not only because it is a large-sample method (true also for profile method) but also because it is a second-order approximation and vulnerable to degree of curvature (see Seber & Wild, 1989). For estimation of BMDs especially, it will ultimately be desirable to implement methods that are more reliable (other BMDS models use the profile likelihood method). But, for the purposes of testing the translation of the previous code to the C version, the Wald-based method used in the original ten Berge code has been retained. The user is warned to be aware

that there may be issues and faulty lower bound estimates produced by the method currently implemented.

2.4 MODEL CODING

C code was developed that implements the ten Berge model and computes the estimates of interest to the user. That code was a translation of the Visual Basic code originally developed by Wil ten Berge and graciously shared with EPA. The entire source code is presented in Attachment A.

The C coding was accomplished within a Windows XP environment. All tests of the code have been carried out with that operating system. The C code was compiled with the Gnu GCC compiler.

SECTION 3: RUNNING THE MODEL

The current version of the ten Berge model software was built to run on a Windows platform, and at present the ten Berge model can only be run from a Command Prompt window. To run the ten Berge model, the user should do the following. The code has been run and tested in using the Windows XP operating system.

Prior to running the model, the executable (tenberge.exe) and data file(s) must be saved in a directory (folder) of the user's choice. They should be stored in the same directory. Further information about the data (input) files is presented below.

Once the files have been saved to the directory of the user's choice, open a Windows Command Prompt window (e.g., under the "Run" option of the start menu, type "cmd.exe"). Change the directory shown on the prompt line to the directory that contains exponential.exe by using the "cd" (change directories) command and being sure to include the entire path name to the directory containing tenberge.exe.

When the Command Prompt window shows that you are in the directory containing tenberge.exe, type the following to execute the program on a data set of your choosing and produce an output file of the results:

tenberge input.(d)

where input.(d) can be the name of any suitably defined (d) file. The input file must have the (d) extension, but the file name (to the left of the period) can be specified as desired by the user as long as there are no spaces in the name. The output of the run will be contained in a file called "input.out" where, again, the actual name of the (d) file used will precede the ".out" extension. For example, typing "tenberge test1.(d)" will run the exponential model program using an input file called test1. It will create an output file called test1.out which will reside in the same directory as the input file and the executable program.

Once created and stored in the directory of the user's choice, the output files can be opened, examined, and edited using Notepad or WordPad, for example. The input files can be opened and edited using Notepad and WordPad as well.

3.1 THE (d) FILE FORMAT

As a demonstration of how to create an input file for the ten Berge model, we reference an example data set. The following dose-response data summary is an example of a data set that might be obtained from a set of experiments in several species, where the animals were exposed to a concentration of a compound (in mg/m^3) for variable durations of time (in minutes) (Darmer et al, 1972). The species are "identified" by their body weights (in grams). In this example, the body weights are the same within each species (e.g., they do not vary by concentration-time group within any one species); they might be replaced by a simple "1," "2," "3," "4" identification flag, but are given as "average" species weights to indicate the magnitude of any

average weight contribution to response across species.

In this example the toxic response of interest was death, as indicated by the response parameter name. There were a total of 68 observations (data records). This is the data set that was used to create a file called example.(d) shown below.

Table 2: Example Dose-Response Data Set

Conc mg/m3	Minutes	BW grams	Exposed	Dead
952	15	200	10	1
1278	15	200	10	4
1403	15	200	10	6
1631	15	200	10	7
1767	15	200	10	9
2028	15	200	10	6
2349	15	200	10	9
653	30	200	10	0
886	30	200	10	0
1006	30	200	10	3
1033	30	200	10	6
1267	30	200	10	9
1359	30	200	10	10
435	60	200	10	0
544	60	200	10	1
653	60	200	10	4
740	60	200	10	8
544	15	23	10	2
707	15	23	10	4
903	15	23	10	7
946	15	23	10	7
1060	15	23	10	6
1153	15	23	10	9
1256	15	23	10	8
1658	15	23	10	9
1958	15	23	15	15
381	30	23	10	2
489	30	23	10	3
636	30	23	10	6
653	30	23	10	5
761	30	23	10	8
788	30	23	10	8
903	30	23	10	9
952	30	23	10	10
190	60	23	10	1
256	60	23	10	2
337	60	23	10	5
408	60	23	10	9
914	15	10000	4	0
1098	15	10000	4	1
1631	15	10000	4	2

1958	15	10000	4	2
2409	15	10000	4	4
555	30	10000	4	1
816	30	10000	4	2
1033	30	10000	4	2
1213	30	10000	4	3
1370	30	10000	4	3
1490	30	10000	4	4
343	60	10000	4	0
598	60	10000	4	1
696	60	10000	4	2
778	60	10000	4	4
924	60	10000	4	4
897	15	3700	4	0
1049	15	3700	4	1
1223	15	3700	4	3
1822	15	3700	4	3
2148	15	3700	4	3
1077	30	3700	4	0
1185	30	3700	4	2
1283	30	3700	4	4
631	60	3700	4	0
663	60	3700	4	1
761	60	3700	4	1
1028	60	3700	4	2
1169	60	3700	4	2
1213	60	3700	4	4

The example (d) file with the appropriate format is given below, based on the above data. Only the values to the left are in the file; the numbers to the right are annotations so that a more complete explanation of the required inputs can be provided below the sample file. An unannotated version of this file is contained in Appendix A, which can be copied and pasted into an empty txt file and then edited (with Notepad for example) to create data sets for your particular application. [Note: except as noted specifically below (e.g., see item 8 in the annotated file below), leading and trailing tabs or spaces on the lines of the input file, or extra tabs or spaces between entries within a line are ignored when the program reads the input file, so when the user edits Appendix A or any other suitable (d) file, s/he may space entries as desired for clarity.]

```
3                          ..... 1
68                         ..... 2
Conc mg/m3                 ..... 3
Minutes                    ..... 4
BW grams                   ..... 5
Exposed                    ..... 6
Dead                       ..... 7
 952 15 200 10 1           ..... etc
 1278 15 200 10 4
 1403 15 200 10 6
 1631 15 200 10 7
```

```
1767 15 200 10 9
2028 15 200 10 6
2349 15 200 10 9
653 30 200 10 0
886 30 200 10 0
1006 30 200 10 3
1033 30 200 10 6
1267 30 200 10 9
1359 30 200 10 10
435 60 200 10 0
544 60 200 10 1
653 60 200 10 4
740 60 200 10 8
544 15 23 10 2
707 15 23 10 4
903 15 23 10 7
946 15 23 10 7
1060 15 23 10 6
1153 15 23 10 9
1256 15 23 10 8
1658 15 23 10 9
1958 15 23 15 15
381 30 23 10 2
489 30 23 10 3
636 30 23 10 6
653 30 23 10 5
761 30 23 10 8
788 30 23 10 8
903 30 23 10 9
952 30 23 10 10
190 60 23 10 1
256 60 23 10 2
337 60 23 10 5
408 60 23 10 9
914 15 10000 4 0
1098 15 10000 4 1
1631 15 10000 4 2
1958 15 10000 4 2
2409 15 10000 4 4
555 30 10000 4 1
816 30 10000 4 2
1033 30 10000 4 2
1213 30 10000 4 3
1370 30 10000 4 3
1490 30 10000 4 4
343 60 10000 4 0
598 60 10000 4 1
696 60 10000 4 2
778 60 10000 4 4
924 60 10000 4 4
897 15 3700 4 0
1049 15 3700 4 1
1223 15 3700 4 3
1822 15 3700 4 3
2148 15 3700 4 3
1077 30 3700 4 0
1185 30 3700 4 2
1283 30 3700 4 4
631 60 3700 4 0
663 60 3700 4 1
761 60 3700 4 1
```

```
1028 60 3700 4 2
1169 60 3700 4 2
1213 60 3700 4 4
                          ..... 8
modeling                  ..... 9
1 1 1 1 1                 ..... 10, 11, 12, 13, 14
3                         ..... 15
1 2 3                     ..... 16,17,18
1                         ..... 19
1 2                       ..... 20, 21
1 68                      ..... 22, 23
dose                      ..... 24
1 1.998 95 1 20 200       ..... 25,26,27,28,29,30
response                  ..... 31
1 1.998 500 20 200        ..... 32,33,34,35,36
ratio                     ..... 37
1 1.998 2 1 2             ..... 38,39,40,41,42
graph/response            ..... 43
1 0.95 1 0 1000 2 20 3 200 ..... 44-52
end                       ..... 53
```

Annotation Notes:

1. Number of input parameters (possible explanatory variables). This number will be equal to the number of fields in an input record minus 2 (the last 2 being for the sample size and number responding).
2. Number of observations/data records. This number tells the program how many lines of data read (see the lines below labeled "etc."
3. Name of the first input parameter. This name will be used to identify the variable considered to be input parameter 1.
4. Name of the second input parameter. This name will be used to identify the variable considered to be input parameter 2.
5. Name of the third input parameter. This name will be used to identify the variable considered to be input parameter 3. *NOTE: there will be as many of these "name" lines as the number on line 1 ("Number of input parameters").*
6. Name of the parameter corresponding to the sample size. This name should indicate the total number of individuals examined for any given data record.
7. Name of the parameter corresponding to the number of individuals having the response of interest. This name may be used to indicate what that response is (e.g., "Dead" in this example).

etc. The data record lines. There should be exactly the same number of lines as the number on the second line of the input file ("Number of observations/data records"). The order of the fields must be the same on each line and must be exactly in the order given in the "Name" lines immediately preceding. And the last two fields on each line must correspond to the total number of individuals examined and the number responding, in that order.

8. Blank line. Include a blank line, with no spaces, tabs, or any other delimiter following the lines of data. This separates the data section from the user-specified modeling control sections.
9. The string "modeling." Just the one string "modeling" should be on this line to indicate the

section that defines the parameters and link functions that are to be included in the model.

10. Type of link function (see Section 2.2 above):
    =1: Probit link function
    = 2: Logit link function
11. Background response correction:
    =1: No background response correction
    =2: Background response correction
    Currently, it is recommended that the user set this control parameter to 1 (no background response correction), as the other option has not been fully tested.
12. Transformation for input parameter 1. Even if the model subsequently defined (see items 15-21) does not include the first input parameter, a transformation for the parameter must be chosen. The identifiers for the 3 transformations included in this software are:
    =1: logarithmic
    =2: reciprocal
    =3: identity (none)
13. Transformation for input parameter 2. Even if the model subsequently defined (see items 15-21) does not include the second input parameter, a transformation for the parameter must be chosen. The identifiers for the transformations are the same as for the first input variable (see item 12 above).
14. Transformation for input parameter 3. Even if the model subsequently defined (see items 15-21) does not include the third input parameter, a transformation for the parameter must be chosen. The identifiers for the transformations are the same as for the first input variable (see item 12 above). *NOTE: It is possible to have more transformation identifiers on this line. There should be one for each of the input parameters and so the number of them should be equal to the number on the first line of the input file ("Number of input parameters").*
15. Number of transformed input parameters to include in the model. This number must be less than or equal to the "Number of input parameters" field (item #1). It indicates the number of single (not product) terms to be included in the model. This value can be zero, but in that case the next line should be skipped altogether

16-18. The numbers corresponding to the input variables to be included in the model. These numbers correspond to the order in which those parameters were entered on the data record lines and to the order of the names given by items #2, #3, #4 (and possibly more) specified above. Remember, these input parameters will be entered into the model transformed however that parameter was specified to be transformed (see items #12-14). The total number of entries on this line will equal the number on the immediately preceding line (unless that number was zero in which case this line will be skipped entirely).

19. Number of product terms to include in the model. This integer must be less than or equal to the number of input parameters (item #1) divided by 2 (rounded down). It indicates the number of product terms to be included in the model. This number can equal zero, in which case the next line is skipped entirely.

20-21. The numbers corresponding to the input parameters included as product terms in the model. The numbers must be entered in pairs, the number of such pairs equaling the integer on the previous line (item #19) (unless the preceding number was zero in which case this line is skipped entirely). These numbers correspond to the order in which those

parameters were entered on the data record lines and to the order of the names given by items #2, #3, #4 (and possibly more) specified above. Remember, these input parameters will be entered into the model transformed however that parameter was specified to be transformed (see items #12-14), even for the product terms. The input parameter identifier numbers used in the product terms need *not* be restricted to those used in the single-parameter terms of the model.

22. The first record number to be included in the analysis. This number must be greater than or equal to 1 and less than or equal to item #2 (number of observations in the data set). Together with the next item (#23), this allows the user to restrict attention to (model only) observations in a certain range of the entire data set.
23. The last record number to be included in the analysis. This number must be greater than or equal to the number given by item #22, and less than or equal to item #2 (number of observations in the data set). Together with the previous item (#22), this allows the user to restrict attention to (model only) observations in a certain range of the entire data set.
24. The string "dose." Only the string "dose" should appear on the next line to indicate that the user wishes to calculate the value of one input parameter that, for specified values of the other input parameters gives a user-specified response value. If the user does not desire to do such a calculation, this line and the following line should be skipped entirely.
25. Confidence intervals calculated?
    = 0 for no (even if this value =0, the remaining items on this line, items #26-30, should still be entered in the appropriate order)
    = 1 for yes
26. Deviate corresponding to the confidence level of choice. See Section 2.2 for a more detailed description of how to select this deviate. Enter this value even if item #25 equals zero.
27. The response of interest. This is given as a percent and so can have values between 0 and 100. X percent response corresponds to a probability of response of X/100.
28. The number identifier corresponding to the input parameter the user wishes to estimate. That number identifier must be one of those included in the list given by items #16-18.

29-30. The values for the other input parameters. These are the values assumed to be known (fixed). For the response of interest (item #27), and for the fixed values, one can determine what value of the input parameter given by the identifier in item #28 gives that response. The values entered here must be in the same order as given in items #16-18, except that the value for the parameter corresponding to the identifier given in item #28 will not be given (because that is the value to be estimated). For example, if parameters 1, 2, and 3 are entered in the model, and we wish to estimate the value of parameter 1 (Conc mg/m3 in our example data set) that give a response of 95% when Minutes is equal to 20 and BW grams is equal to 200, then the line in question will appear as shown above, with the value for Minutes (20) preceding the value for BW grams (200) because Minutes is the second input parameter and BW grams is the third input parameter.

31. The string "response." Only the string "response" should appear on this line to indicate that the user wishes to calculate the response percentage for user-specified values of all the selected input parameters. If the user does not desire to do such a calculation, this line and the following line should be skipped entirely.
32. Confidence intervals calculated?
    = 0 for no (even if this value =0, the remaining items on this line, items #33-36, should still be entered in the appropriate order)

= 1 for yes

33. Deviate corresponding to the confidence level of choice. See Section 2.2 for a more detailed description of how to select this deviate. Enter this value even if item #32 equals zero.
34-36. Values for the all the input parameters included in the model. These values should be entered in the order corresponding to their number identifier (e.g., in this example, the value for Conc mg/m3 first, followed by a value for Minutes, followed by a value for BW grams).
37. The string "ratio." Only the string "ratio" should appear on this line to indicate that the user wishes to calculate the ratio of specified B_i coefficients estimated in the model. If the user does not desire to do such a calculation, this line and the following line should be skipped entirely.
38. Confidence intervals calculated?
    = 0 for no (even if this value =0, the remaining items on this line, items #39-42, should still be entered in the appropriate order)
    = 1 for yes
39. Deviate corresponding to the confidence level of choice. See Section 2.2 for a more detailed description of how to select this deviate. Enter this value even if item #38 equals zero.
40. Number of variables selected. This value should always be set equal to 2, as the ratio will be of the coefficients for two input parameters.
41-42. The number identifiers of the pair of coefficients for which the ratio is desired. As above, these number identifiers correspond to the order that the input parameters are entered in a data record. The identifiers must be from among the set of identifiers entered in items #16-18 (and any additional identifiers on that line) as input parameters used as single terms in the model.
43. The string "graph/response." Only the string "graph/response" should appear on this line to indicate that the user wishes to produce plots of the model results. *Currently, the graphical functions of this software are in development, so this line and the next merely indicate right now how such plotting will be done.* If the user does not desire to do such plotting, this line and the following line should be skipped entirely.
44. Confidence intervals calculated?
    = 0 for no (even if this value =0, the remaining items on this line, items #45-52, should still be entered in the appropriate order)
    = 1 for yes
45. Deviate corresponding to the confidence level of choice. See Section 2.2 for a more detailed description of how to select this deviate. Enter this value even if item #44 equals zero.
46. X-axis variable. This is identified by identifier number as in the previous references to parameters.
47-48. The range for the x-axis. The response will be calculated and plotted for each value of the x-axis variable between item #47 and item #48.
49-52. In pairs, the identifier number and value to assume for the remaining input parameters included in the model. Each pair will have the identifier number for the parameter and the value to assign to that parameter when the plots are created. There will be as many pairs as there are input parameters included in the model.
53. The string "end." Only the string "end" should appear on this line to indicate that the program has reached the end of the input file. This must be the last line of each input file.

3.2 INTERPRETATION OF OUTPUT FILES

Appendix B contains the output file produced by running the ten Berge model on this data set. That is, Appendix B is the result of typing "tenberge example.(d)" at a Windows Command Prompt if the dose-response data shown in the above table are represented in the file called "example.(d)," which is created in accordance with the instructions shown above, and is in fact the very input file that is shown in Appendix A.

The output file shown in Appendix B has been color coded for ease of reference and explanation of the various sections of that output. Details related to that output are provided here, referenced by color-coded section.

Reference Model Information: This section merely gives information about the version number and build date of the program, as well as identifying the input data set used to create the output and the name of the file ("example.plt" in this example) that has the information needed to later produce graphics. One of the most important pieces of information in this section may be the date and time at which the model was run. This may be important to keep track of the latest version of the output, should changes and corrections be made. At present, if the model is run using an input file having the same name as an input file previously run, the earlier output file will be over-written.

Model Specifications: This section provides the overview of the framework for the model, including the reference for Finney (1977) from which Wil ten Berge identified the probit analysis approach. The general form of the model as it is now implemented is presented here as is a basic summary of the number of input parameters (possible explanatory variables) and the number of observations in the data set.

Input Data Set Echo: This section should contain exactly the data that the user has included in the input file. If there are any errors here, the user should go back to the input file and correct the input values, and then rerun the analysis.

Modeling Choices: In this section, the following information is provided: the choices for the range of observations to analyze, the transformations of the input parameters to use, the link function (logit or probit), and the variable identifier numbers associated with the selected explanatory variables (single input parameters or products of pairs of input parameters). If any of this information does not correspond to the desired analysis, the user must go back to the input file to make corrections to the coding in the modeling section.

Fit and parameter estimates: The chi-square evaluation of fit and the degrees of freedom associated with the model fit to the selected data are given. The maximum likelihood estimates of the B_i coefficients are shown as is a Student t value that can be used to determine whether each of those terms is "statistically significant." Variance and covariance estimates for each of the B_i terms are provided.

Notes: When the model fails, one or more of the parameter values may have a value of the form "-1.#J" or a similar non-number. This is an indication that the model has not converged to an answer. The user should check the input file (also reflected in the data echo section of the output file) to see if some data entry errors are contributing to that problem.

If there are no errors in the input data, it is entirely possible that there is no solution for some data; this happens, for example, when there are only groups with either 0% or 100% response. In such cases, the model cannot determine a maximum likelihood and returns values for one or more of the parameters (and estimates depending on those parameters) that are of the form "-1.#IOe+000," "1.#R," "1.#QOe+000," "1.#QNAN0," or similar indications that no numerical answer was available. It is known, for example, that probit slope estimates can be infinite in some situations where concentrations with and without response are not suitably intermingled. A sufficient (but not necessary) condition to avoid this is to have two distinct doses with partial response.

"Dose" estimation: In this section one finds the estimate of an input parameter value that, for a given response and for specified values of the other input parameters, gives that response rate. In this example, the response was set to 95% and the Minutes and BW grams variables were set to 20 and 200, respectively. For those Minutes and BW grams values, the model estimates that that the Conc mg/m3 needed to get 95% response is 2228. Because a deviate corresponding to a confidence level was given (1.998, which is the student-t deviate associated with 95% confidence level, where there are 63 degrees of freedom) the lower and upper bounds on that Conc mg/m3 estimate are also shown). The values (1865 and 3029 mg/m^3, respectively) can be taken as the 95% confidence interval for the concentration giving 95% chance of death when the exposure duration is 20 minutes and the BW of the animal is 200 grams.

Notes: The use of the terminology "probability of correct model" in the output file is not a good choice for describing the results of the chi-squared goodness-of-fit test that is the basis for the reported p-value. Subsequent versions of this software will replace that terminology with a statement like "The p-value associated with the chi-square goodness of fit test equals x" where x is the calculated p-value. The terminology as shown in the example output file has been retained so that comparisons between the new version and the original version of the ten Berge software could be more easily made (the same description has been retained in both cases). Similarly, the statement that the "prediction of the model is not sufficient" will be modified to simply indicate whether or not the p-value is greater than or less than 0.05, with the appropriate statement regarding adequate fit of the model or not (similar to the evaluations of fit in other BMDS models) and a suggestion that (if the model is not fitting the data well) the correction factor be applied to the variance and covariance estimates as well as selecting the deviate from the Student T distribution rather than the standard normal distribution. Currently, neither the variance-covariance correction nor the choice of the deviate are done automatically for the calculation of confidence limits.

Response Estimation: Much like the previous section, this section provides estimates of the response associated with specified values of the input variables used in the model. When a deviate is given, the corresponding confidence interval is also calculated for that estimate. Note that the deviate supplied need not be the same as the one provided for the "dose" estimation, in

case different confidence levels may be desired for dose and response estimates. In this example, the response estimated to be associated with 500 mg/m^3 exposure, lasting 20 minutes, in an animal weighing 200 grams is 9.52%, and the 95% confidence interval for that estimate is 3.46% to 21.2%.
Notes: See notes in previous section about terminology related to the model fit.

Ratio Estimation: In this section the ratio of the Bi coefficients request by the user will be reported. As in the previous sections the confidence interval is also shown, if requested, at a level consistent with the specified deviate. Here the ratio of the coefficients for Conc mg/m3 and Minutes is estimated to be approximately 1.50, with 95% confidence interval extending from -1.38 to 4.39. Recall from the above presentation (Section 2.2) that the ratio of the coefficient for the concentration term to the coefficient for the duration term is an estimate of the exponent N in the $C^N xT$ relationship, when ln(C) and ln(T) are the transformations used in the model (as they are here). In this particular example, the model also includes the term ln(C)*ln(T) (the product specified and identified with variable 4). Thus, the interpretation of the ratio B_1/B_2 as an estimate of N might be problematic and might account for the fact that the lower bound on B_1/B_2 is negative. Of course, the user could always re-run the model opting not to include the product of transformed concentration and transformed duration. It is left as an exercise for the reader to see what happens in that.

Notes: See notes in previous section about terminology related to the model fit.

3.3 ADDITIONAL CONSIDERATIONS AND ISSUES FOR MODEL RUNS

At present, the ten Berge software does not allow the user to fix parameters at specific values nor does it allow the user to specify initial (starting) values for the estimation of maximum likelihood parameter values. Both of those options are available in other BMDS models. It is anticipated that future versions of the ten Berge program may also include those options. However, as discussed previously and in the next section, the focus for this iteration was to produce a program that could reproduce the estimates and calculations obtained by the original ten Berge software.

Also, for the present implementation, we have not exercised the background correction option. As noted in the above description of the input file items, the user/reviewer should keep the flag set so that no background correction is attempted. For some older probit programs, the approach for handling background mortality was to adjust the observed responses based on mortality in the control group (using Abbott's formula), and fit a two-parameter probit model. More modern probit programs are like BMDS in estimating the background response as a third parameter. The ten Berge program already contains enough parameters so that it can fit a model that estimates a background response rate (when the identity transformation is used with concentration, but not when the logarithmic transformation is used with concentration). When this option is implemented, its parameterization will be completely specified.

SECTION 4. MODEL TESTING

This section reports on the latest set of testing that has been conducted for the software. This section can be considered almost a separate module that is independent of the previous sections (which describe the motivation for, the background of, and the "how-to" for running and interpreting the model. Indeed, with additional testing, this section may change and be updated while the previous sections may remain the same .

4.1 CURRENT TESTING METHODS

The basis for testing the C-code implementation of the ten Berge model was the Visual Basic version of that model provided by Wil ten Berge. Executable version number 4 of his model was used to derive estimates of the **parameter values (including Student-t values, variances, and covariances)**, **chi-squared fit estimates**, **degrees of freedom**, **dose estimates for a given response**, **response estimates for a given set of input parameters**, and **coefficient ratio estimates**, for 3 data sets. The values produced by his program were then used as the standard against which the C-code implementation was compared.

Note that this test merely confirms that the new model code yields the same estimates as the old ten Berge model from which it was translated. Since the first task towards getting code that can be integrated into BMDS is to successfully translate existing working models into code that can be used for that integration, such a test is adequate for our purposes. The new program has not, however, been tested against an independent program that could be configured to run the same models. At some point, such a test, perhaps using EPA's CatReg software as the independent implementation might be desirable. One small exception to the restrictions on the testing just described is presented below, based on analysis provided by an internal EPA reviewer of a data set run using the C-code implementation of the ten Berge model and an R program he wrote.

The approach used for the production of the current implementation was to use an automated translator of Visual Basic code into C code, with additional programming input needed to fix identified problems. Problems were identified only when the Visual Basic and C versions produced different output values. Thus, the main goal of testing for successful translation is accomplished when and if the outputs from the two versions agree on the test sets.

All runs of the original and C versions of the program were done using the Windows XP operating system.

4.2 TESTING RESULTS

Figure 1a shows the screen shot of the original (Visual Basic) ten Berge model parameter estimates for the example.(d) data set. This is the screen obtained when the "calculate" option is chosen from the drop down menu under "Estimation" on the toolbar, after having made the selections that match those shown in example.(d) and echoed in example.out. For example, the

selections for that data set include picking the probit link function, logarithmically transforming all the parameters, and selecting all of them to be in the model. Moreover, Figures 1b, 1c, and 1d show, respectively, the screen shots for that run corresponding to "dose at response," "response at dose," and "ratio regression coeff." choices under the "Evaluation" option on the toolbar. The corresponding set of Figures (2a-2d, and 3a-3d) are shown for runs on data sets example2.(d) and example3.(d), respectively. These are the standards against which the new code outputs were compared.

Appendix B displays the output file produced by the new code for the first data set (shown above in Table 3). Appendix C shows the data set (as part of the input file) and an output file produced by the new code when run on the example2.(d) data set; Appendix D shows a third data set, example3.(d) and the corresponding output file. Thus, Appendix B outputs correspond to those shown in Figures 1a-1d. Appendix C outputs correspond to those shown in Figures 2a-2d. Appendix D outputs correspond to those shown in Figures 3a-3d. The choice of link function, the parameters included in the model, the transformations, and the responses and doses estimated were varied from one data set to another (see descriptions in the corresponding output files) to ensure that the desired options were operating correctly.

Comparison of the corresponding figures and output files from the appendices shows that the executable from the new code and the executable obtained from Wil ten Berge match exactly (to the number of digits shown, which is up to 4 significant digits for the parameter estimates) on all of the parameters listed in Section 5.1. There is no instance where a mismatch was detected.

As mentioned above, an internal EPA reviewer ran a version of the ten Berge model that he coded using the R language. He used the data set of example.(d) but included in the model only conc mg/m3, minutes, and BW, all logarithmically transformed. No product terms were included.

A portion of the ten Berge output that he obtained is shown here:

```
Transformation of input parameters
   Conc mg/m3          is transformed logaritmically!
   Minutes             is transformed logaritmically!
   BW grams            is transformed logaritmically!

   Probit link used without background response correction!

   Variable  1  =  Transformed Conc mg/m3
   Variable  2  =  Transformed Minutes
   Variable  3  =  Transformed BW grams

   Chi-Square          =  114.87
   Degrees of Freedom  =  64

   B0 = -1.008e+001    Student t  for B0 = -5.88
   B1 = 1.950e+000     Student t  for B1 = 9.81
   B2 = 9.754e-001     Student t  for B2 = 5.67
   B3 = -2.340e-001    Student t  for B3 = -6.76

     variance  B00 = 2.935e+000
   covariance  B01 = -3.315e-001
   covariance  B02 = -2.562e-001
   covariance  B03 = 3.128e-002
     variance  B11 = 3.947e-002
   covariance  B12 = 2.571e-002
   covariance  B13 = -4.035e-003
     variance  B22 = 2.963e-002
   covariance  B23 = -3.082e-003
     variance  B33 = 1.197e-003
```

The corresponding R code output was reported as follows:

```
Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -15.0814     1.7132  -8.803  < 2e-16 ***
log(C)        1.9498     0.1987   9.814  < 2e-16 ***
log(T)        0.9754     0.1721   5.667 1.46e-08 ***
log(BW)      -0.2340     0.0346  -6.762 1.36e-11 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 269.09  on 67  degrees of freedom
Residual deviance: 137.65  on 64  degrees of freedom
AIC: 257.01

Number of Fisher Scoring iterations: 4

> print(vcov(model))
            (Intercept)       log(C)       log(T)      log(BW)
(Intercept)  2.93514424 -0.331546093 -0.256184219  0.031281131
log(C)      -0.33154609  0.039472909  0.025712671 -0.004034729
log(T)      -0.25618422  0.025712671  0.029628403 -0.003082273
log(BW)      0.03128113 -0.004034729 -0.003082273  0.001197344
```

Despite the slightly different way of presenting the information, all the printed estimates agree to 4 significant digits. [The estimate and Student T values for B0 from the ten Berge program do not match the estimate of the "intercept" or "z value" from the R program because the ten Berge program has an "intercept" equal to B0 – 5; see the discussion in Section 2.2. Once that correction is made, the agreement noted is complete.]

Thus, the comparisons with the original Visual Basic program and a simple and small independent test illustrated here suggest that the conversion of the ten Berge model to C code that can eventually be used to incorporate CxT modeling into BMDS has been successfully accomplished.

ACKNOWLEDGEMENT

The authors would like to acknowledge the special assistance and original work of Dr. Wil Ten Berge. The code developed here is a direct "translation" of the program Dr. ten Berge wrote to implement the CxT methodology. Dr. ten Berge was extremely helpful in resolving issues that arose in the course of that development.

REFERENCES

Berge WF ten (1985). The toxicity of methylisocyanate for rats. Journal of Hazardous Materials; 12:309-311.

Berge WF ten, Zwart A, Appelman LM (1986). Concentration-time mortality response relationship of irritant and systemically acting vapors and gases. Journal of Hazardous Materials; 13:301- 309.

Berge WF ten; Zwart A (1989). More efficient use of animals in acute inhalation toxicity testing. Journal of Hazardous Materials; 21:65-71.

Collett, D. (1991). *Modelling Binary Data*. Chapman and Hall/CRC, New York.

CREM, 2003. Draft Guidance on the development, evaluation, and application of regulatory environmental models. Prepared by the Council for Regulatory Environmental Modeling. Novemeber, 2003.

Crump, K. amd Howe, R., 1984. A review of methods for calculating statistical confidence limits in low dose extrapolation. In: Clayson et al., eds. Toxicological Risk Assessment, Volume I: Biological and Statistical Criteria. CRC Press, New York.

Darmer, K.I. Jr, Haun, C.C., MacEwen, J.D., 1972. The acute inhalation toxicology of chlorine pentafluoride. Am Ind Hyg Assoc J. 1972 Oct;33(10):661-8.

Fieller, E. C., 1944. Fundamental formula in the statistics of biological assay and some applications. Quarterly Journal of Pharmacology 17: 117-123.

Finney, D.J. (1971). *Probit Analysis*. (3rd ed.) Cambridge University Press, Cambridge, UK.

McCullagh and Nelder (1989). *Generalized Linear Models*. Chapman and Hall, New York.

Moerbeek et al. 2004. Risk Analysis 24:31-40

Morgan, B.J.T. (1992). *Analysis of Quantal Response Data*. Chapman and Hall, New York.

Nitcheva et al. 2007. Reg. Tox. Pharm. 48: 135-147

Seber, G.A.F., and Wild, C.J. (1989). *Nonlinear Regression*. Wiley and Sons, New York.

FIGURES

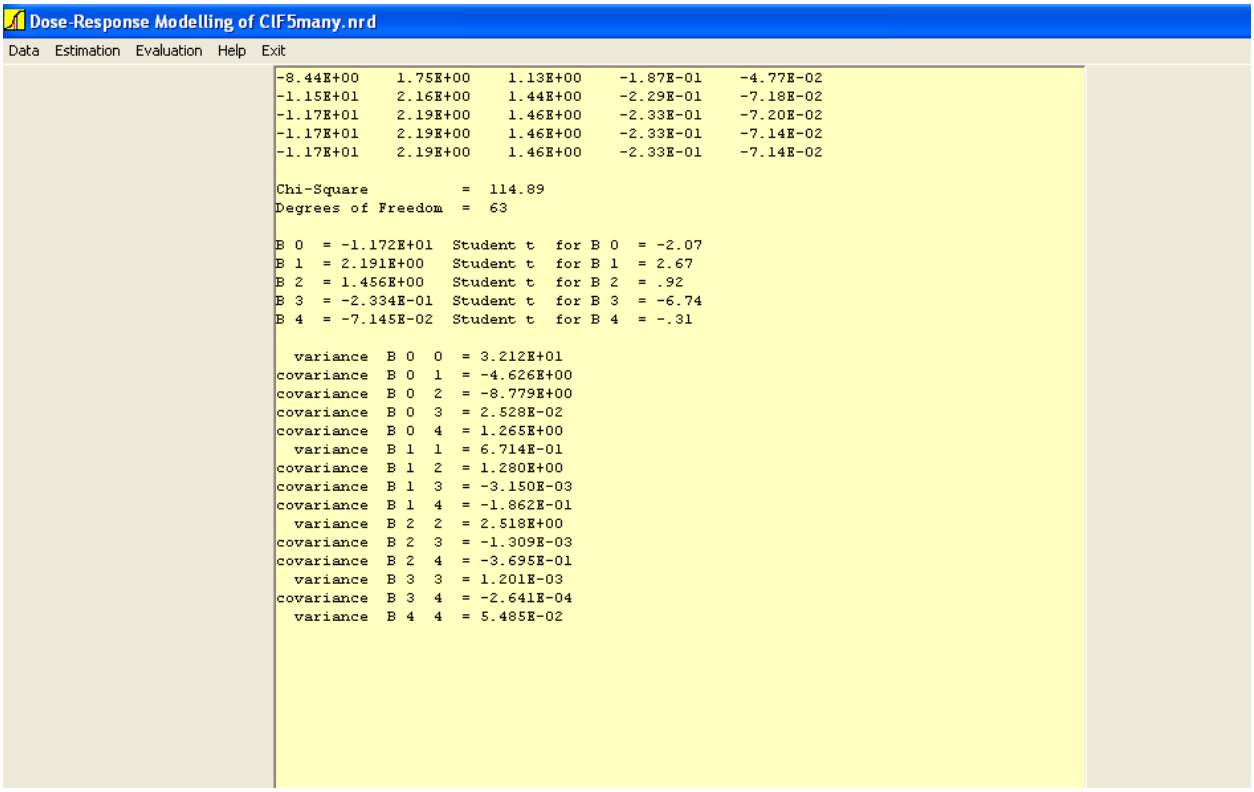

Figure 1a: ten Berge Visual Basic run on example.(d) data set. Coefficient estimates and chi-square.

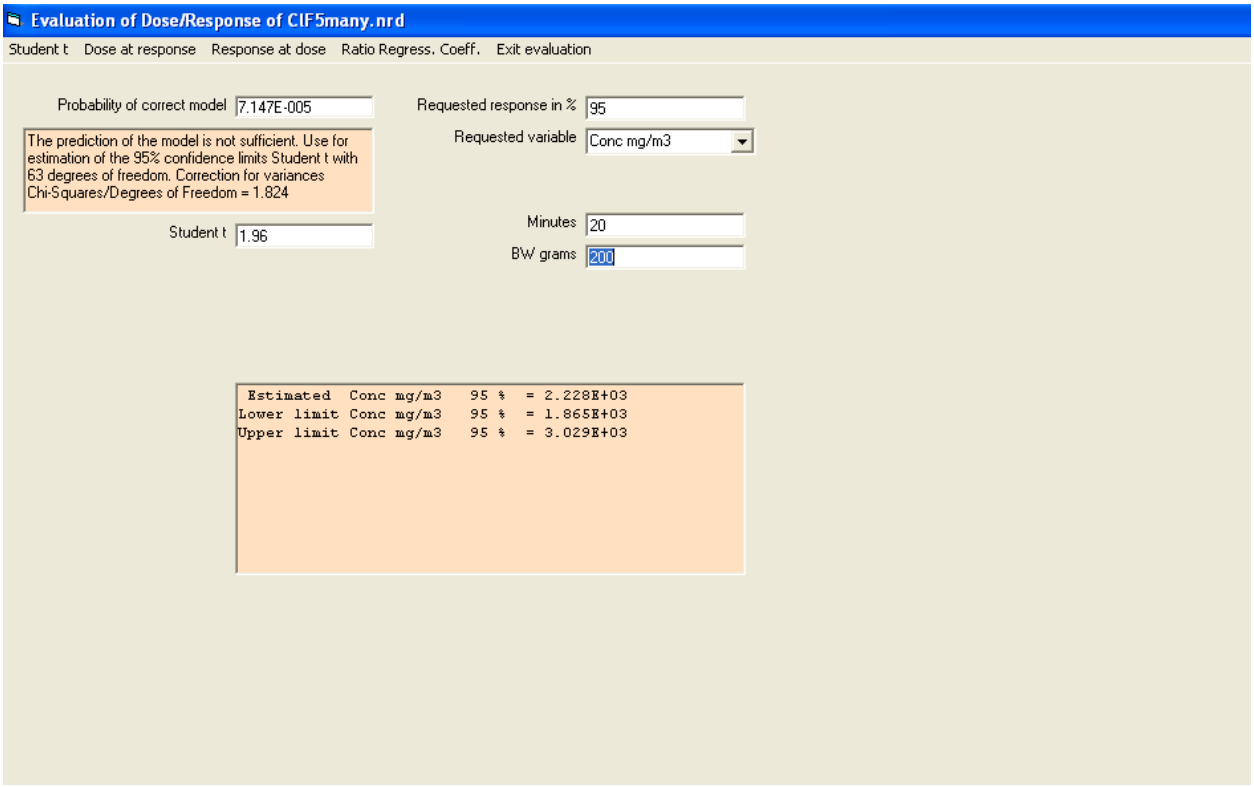

Figure 1b: ten Berge Visual Basic run on example.(d) data set. Concentration estimate corresponding to 95% response rate, for selected values of Minutes and BW grams.

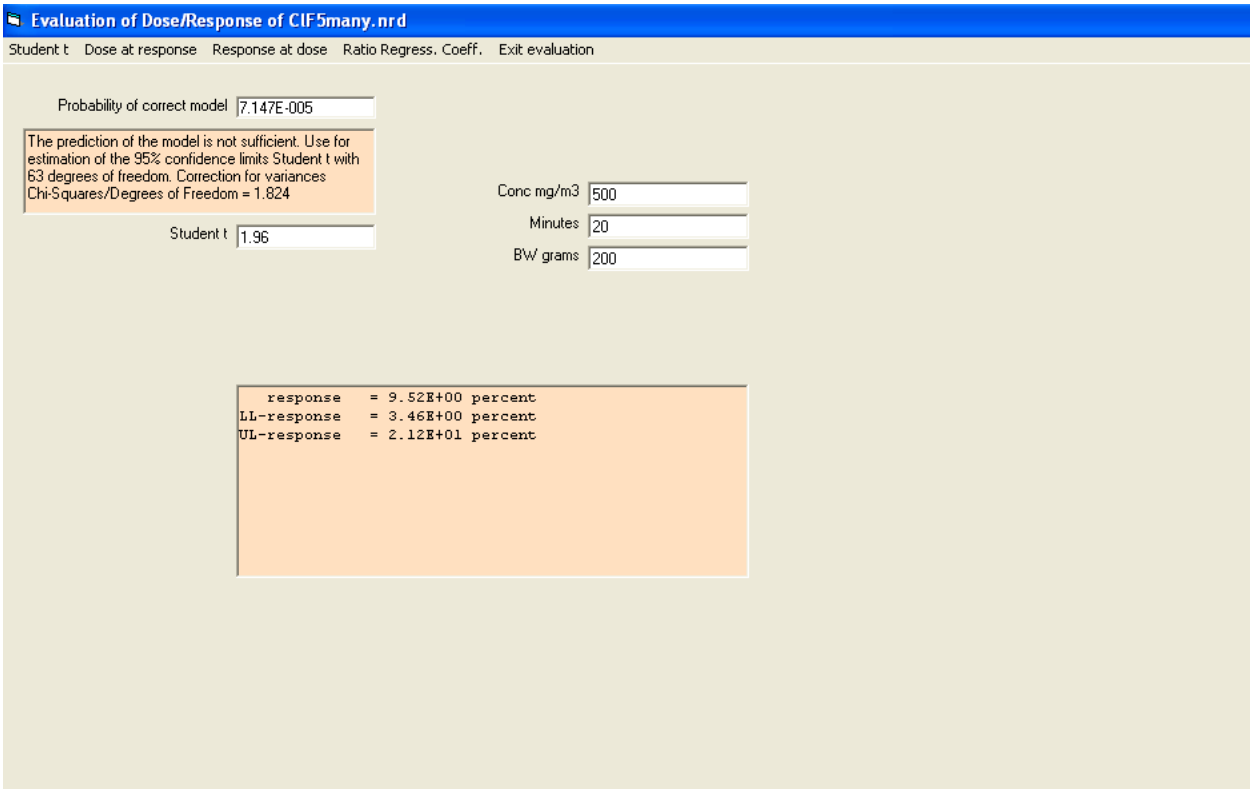

Figure 1c: ten Berge Visual Basic run on example.(d) data set. Response estimate for selected values of Conc mg/m3, Minutes, and BW grams.

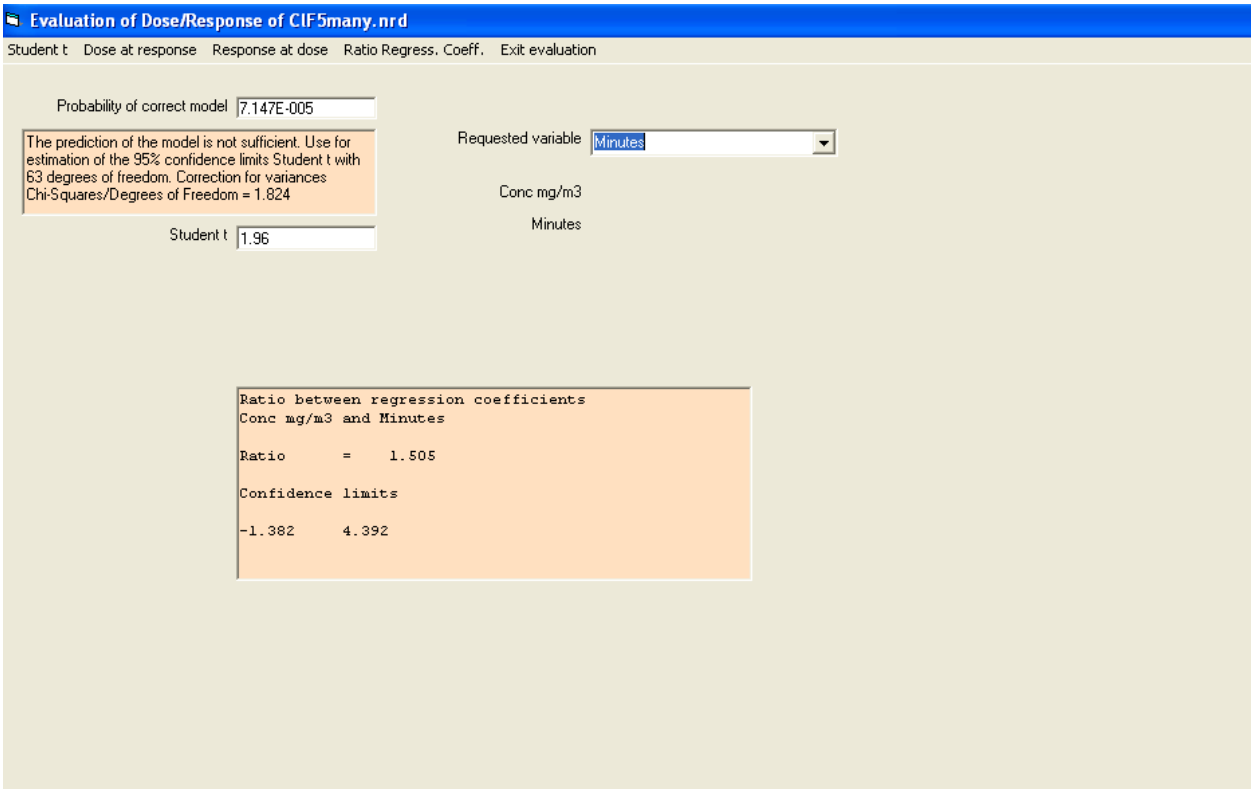

Figure 1d: ten Berge Visual Basic run on example1.(d) data set. Ratio of coefficients of Conc mg/m3 and Minutes.

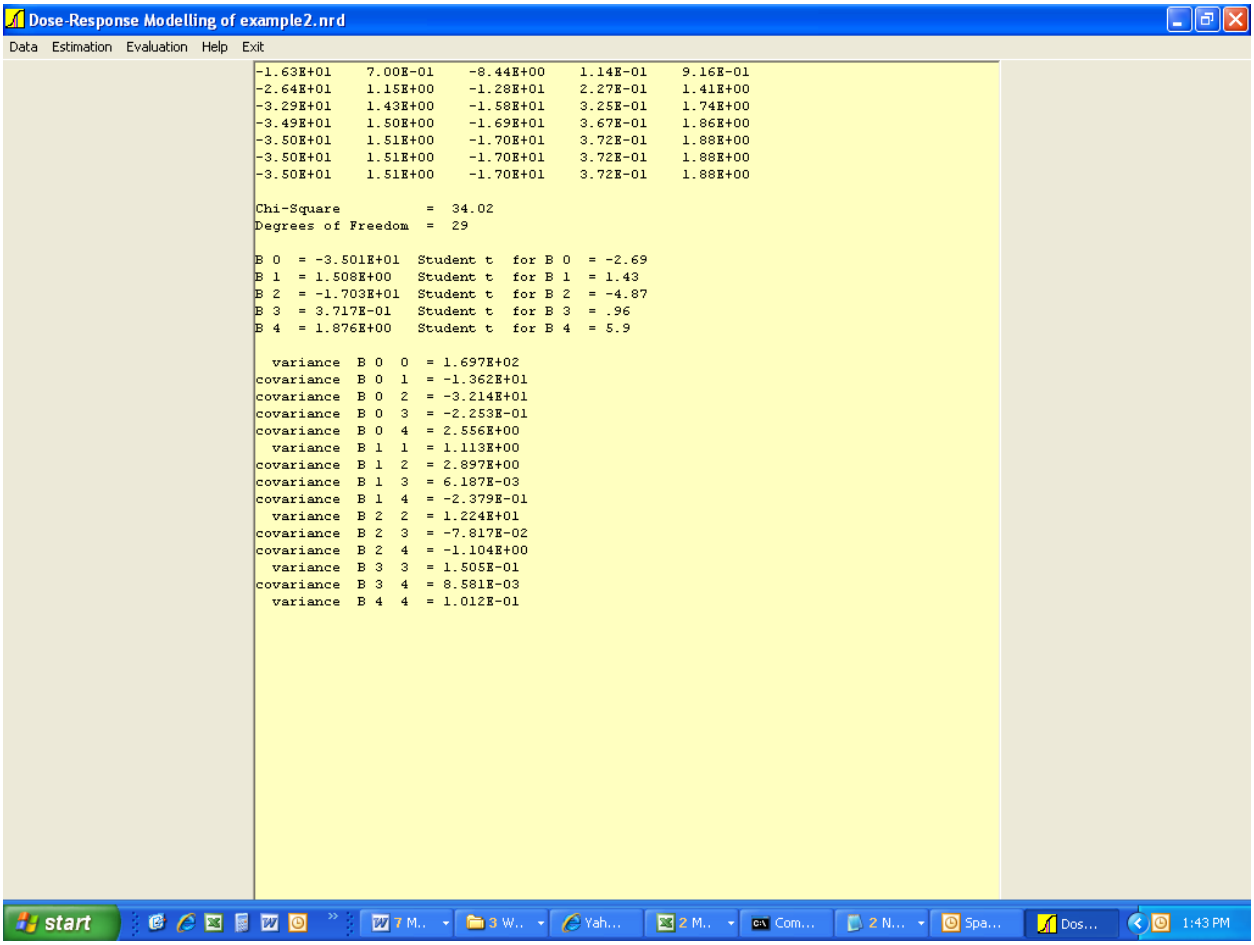

Figure 2a: ten Berge Visual Basic run on example2.(d) data set. Coefficient estimates and chi-square.

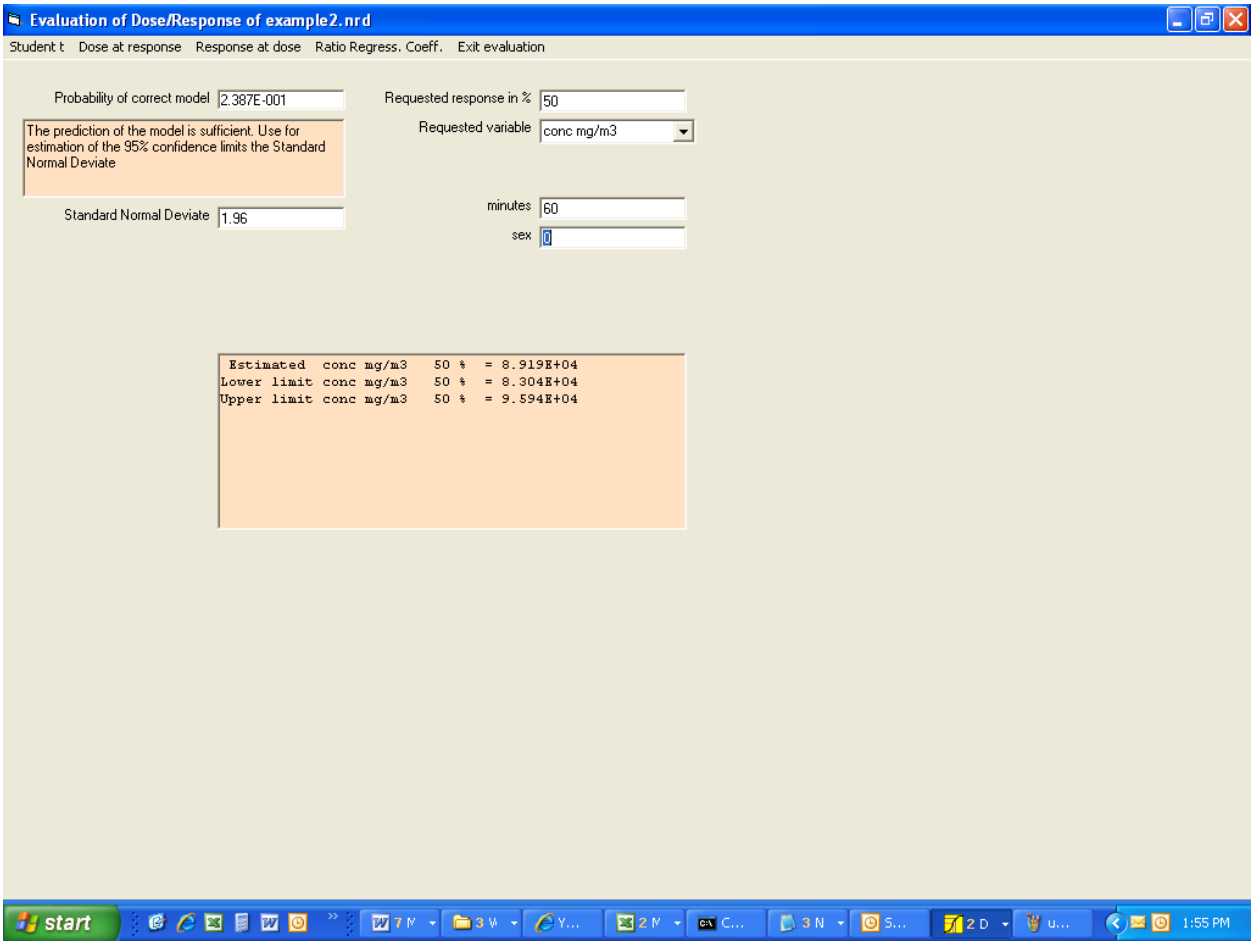

Figure 2b: ten Berge Visual Basic run on example2.(d) data set. Concentration estimate corresponding to 50% response rate, for selected values of minutes and sex.

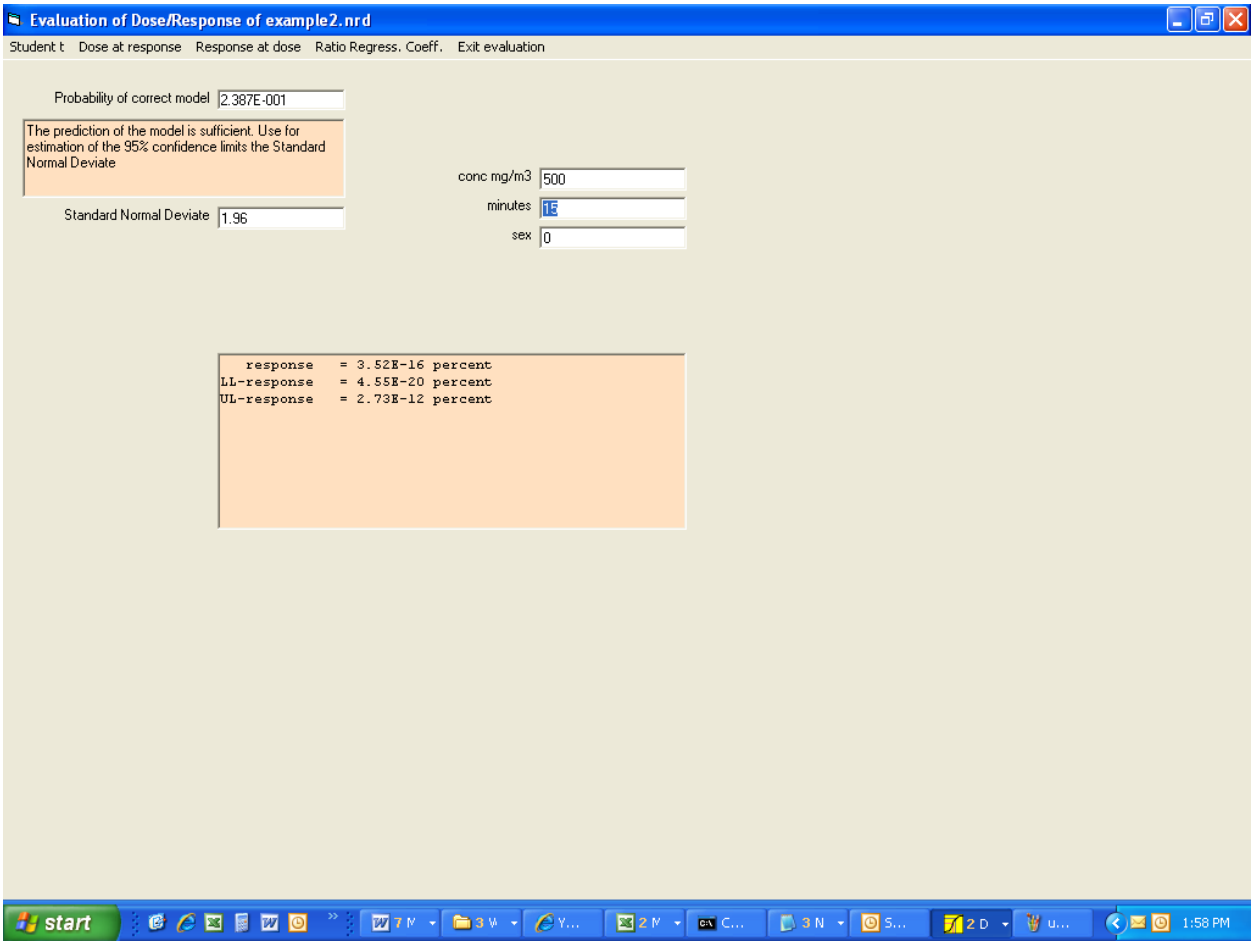

Figure 2c: ten Berge Visual Basic run on example2.(d) data set. Response estimate for selected values of conc mg/m3, minutes, and sex.

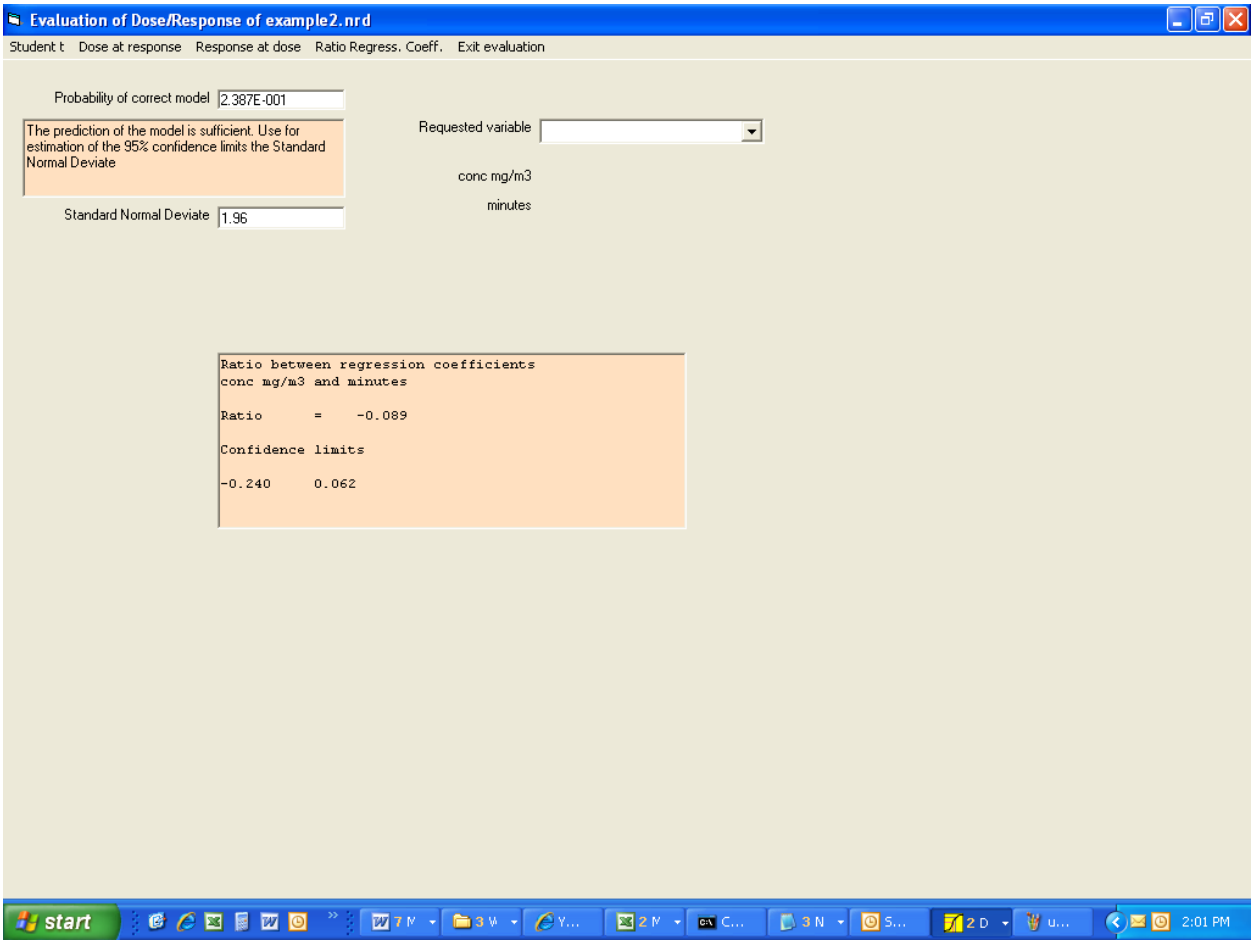

Figure 2d: ten Berge Visual Basic run on example2.(d) data set. Ratio of coefficients of conc mg/m3 and minutes.

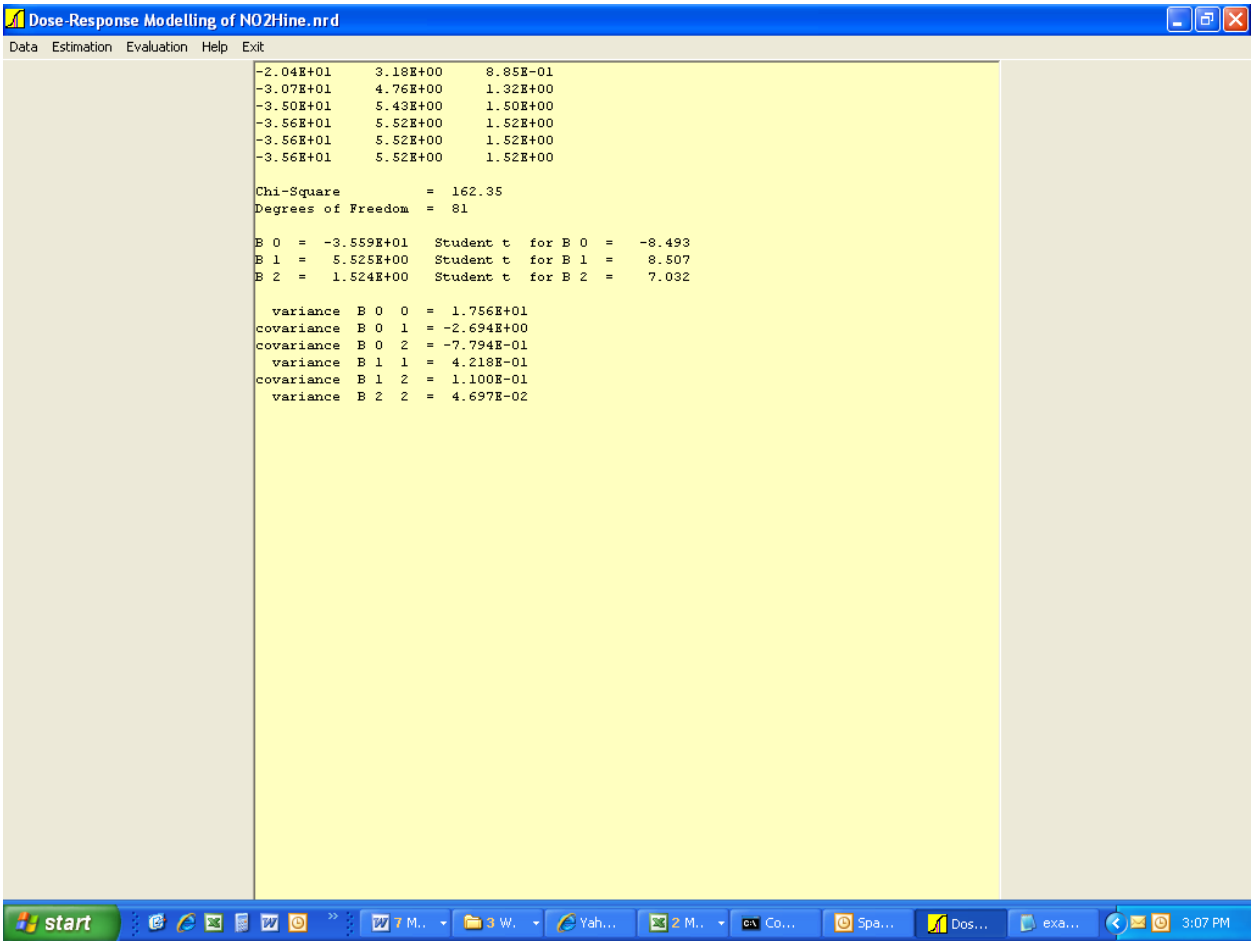

Figure 3a: ten Berge Visual Basic run on example3.(d) data set. Coefficient estimates and chi-square.

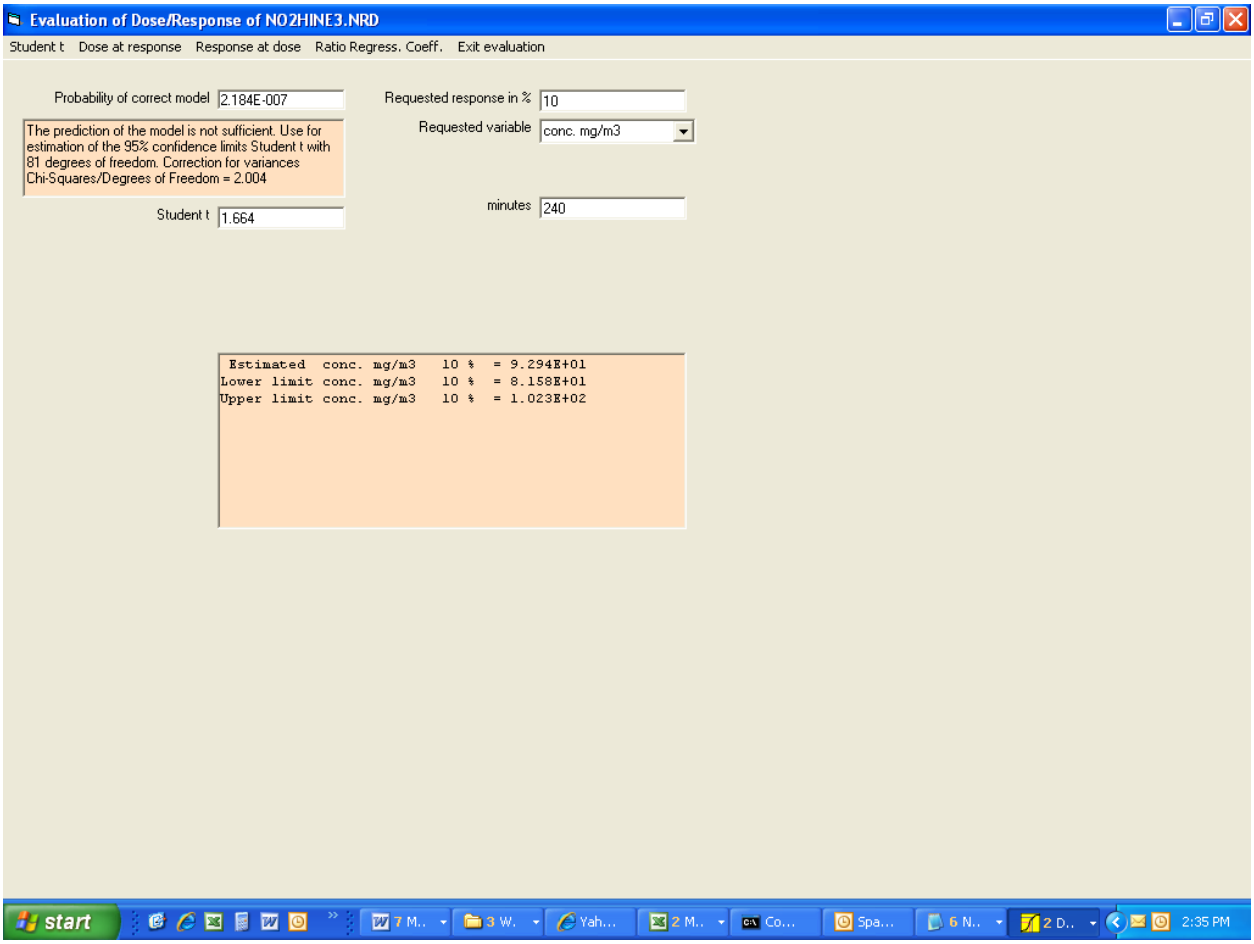

Figure 3b: ten Berge Visual Basic run on example3.(d) data set. Concentration estimate corresponding to 10% response rate, for selected value of minutes.

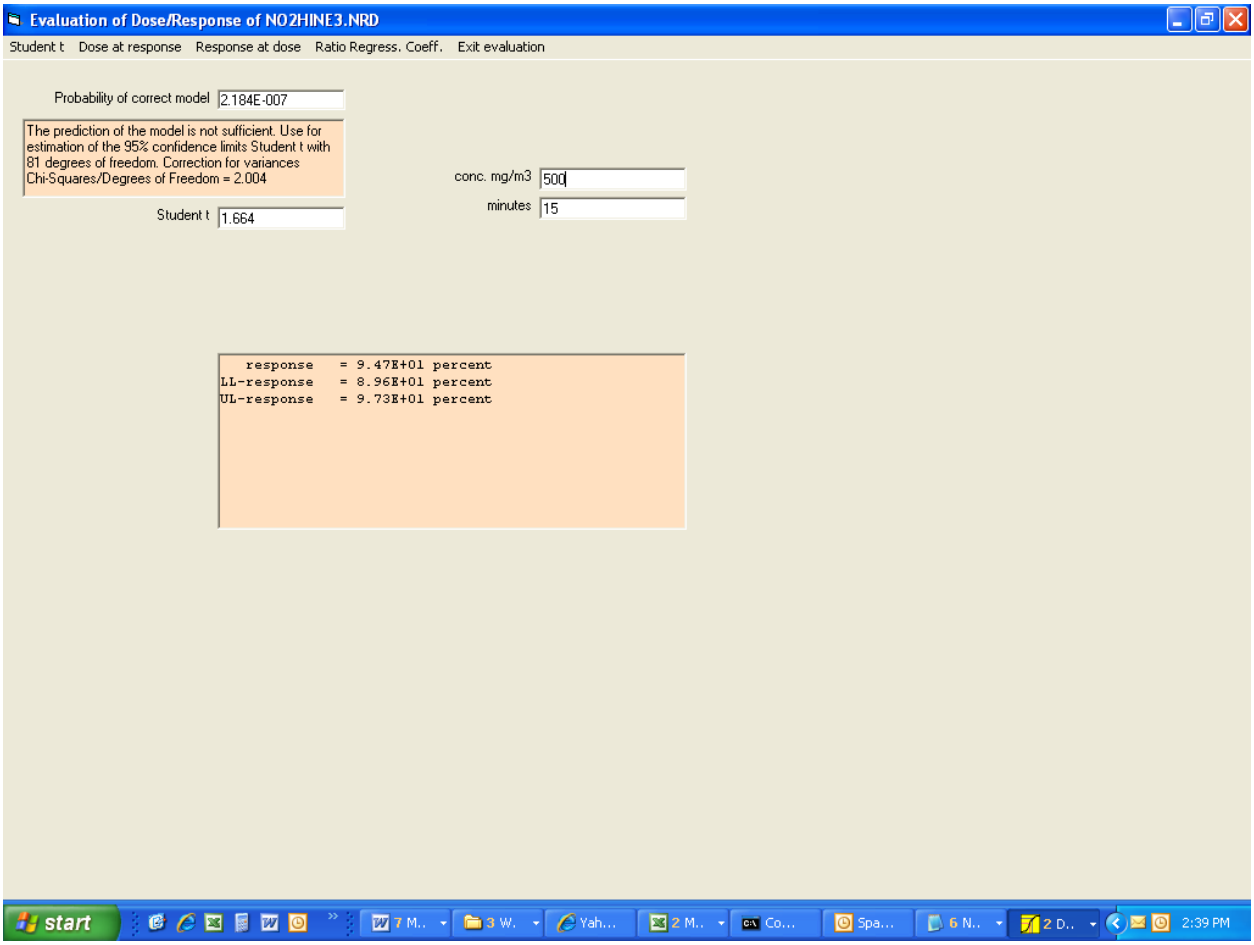

Figure 3c: ten Berge Visual Basic run on example3.(d) data set. Response estimate for selected values of conc mg/m3 and minutes.

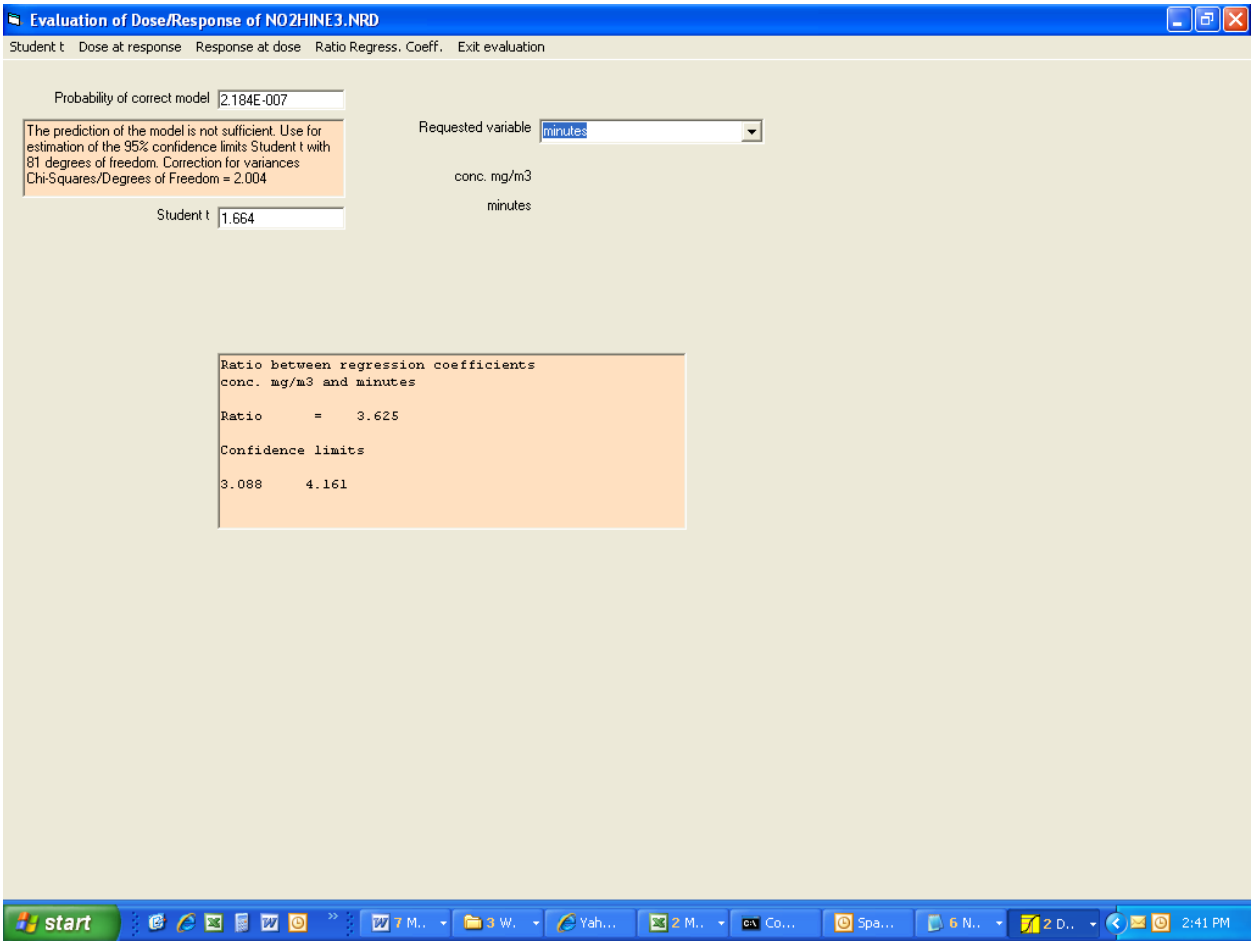

Figure 3d: ten Berge Visual Basic run on example3.(d) data set. Ratio of coefficients of conc mg/m3 and minutes.

Appendix A: Unannotated Sample (d) File

```
3
68
Conc mg/m3
Minutes
BW grams
Exposed
Dead
 952 15 200 10 1
 1278 15 200 10 4
 1403 15 200 10 6
 1631 15 200 10 7
 1767 15 200 10 9
 2028 15 200 10 6
 2349 15 200 10 9
 653 30 200 10 0
 886 30 200 10 0
 1006 30 200 10 3
 1033 30 200 10 6
 1267 30 200 10 9
 1359 30 200 10 10
 435 60 200 10 0
 544 60 200 10 1
 653 60 200 10 4
 740 60 200 10 8
 544 15 23 10 2
 707 15 23 10 4
 903 15 23 10 7
 946 15 23 10 7
 1060 15 23 10 6
 1153 15 23 10 9
 1256 15 23 10 8
 1658 15 23 10 9
 1958 15 23 15 15
 381 30 23 10 2
 489 30 23 10 3
 636 30 23 10 6
 653 30 23 10 5
 761 30 23 10 8
 788 30 23 10 8
 903 30 23 10 9
 952 30 23 10 10
 190 60 23 10 1
 256 60 23 10 2
 337 60 23 10 5
 408 60 23 10 9
 914 15 10000 4 0
 1098 15 10000 4 1
 1631 15 10000 4 2
 1958 15 10000 4 2
 2409 15 10000 4 4
 555 30 10000 4 1
 816 30 10000 4 2
 1033 30 10000 4 2
 1213 30 10000 4 3
 1370 30 10000 4 3
 1490 30 10000 4 4
 343 60 10000 4 0
```

```
598 60 10000 4 1
696 60 10000 4 2
778 60 10000 4 4
924 60 10000 4 4
897 15 3700 4 0
1049 15 3700 4 1
1223 15 3700 4 3
1822 15 3700 4 3
2148 15 3700 4 3
1077 30 3700 4 0
1185 30 3700 4 2
1283 30 3700 4 4
631 60 3700 4 0
663 60 3700 4 1
761 60 3700 4 1
1028 60 3700 4 2
1169 60 3700 4 2
1213 60 3700 4 4

modeling
1 1 1 1 1
3
1 2 3
1
1 2
1 68
dose
1 1.96 95 1 20 200
response
1 1.96 500 20 200
ratio
1 1.96 2 1 2
graph/response
1 0.95 1 0 1000 2 20 3 200
end
```

Appendix B: Color Coded Example Output File

```
 ================================================================
       Ten Berge Model. (Version: 1.0; Date: 12/26/2006)
       Input Data File: example.(d)
       Gnuplot Plotting File:  example.plt
                                          Mon Apr 02 11:32:50 2007
 ================================================================

 Dose-Response Analysis

 Method of Maximum Likelihood according to:
 D.J. Finney, 1977. Probit Analysis. Cambridge University Press.
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

   Model: P(v1, v2, ...) = Link(B0 + B1*v1 + B2*v2 + ...)

   Link is either Logit or Probit
   v1, v2, ... are the variables (transformations of the input parameters)


   Number of input parameters = 3
   Total number of observations = 68
   Total number of records with missing values = 0
```

Conc mg/m3	Minutes	BW grams	Exposed	Dead
952.00	15.00	200.00	10.	1.
1278.00	15.00	200.00	10.	4.
1403.00	15.00	200.00	10.	6.
1631.00	15.00	200.00	10.	7.
1767.00	15.00	200.00	10.	9.
2028.00	15.00	200.00	10.	6.
2349.00	15.00	200.00	10.	9.
653.00	30.00	200.00	10.	0.
886.00	30.00	200.00	10.	0.
1006.00	30.00	200.00	10.	3.
1033.00	30.00	200.00	10.	6.
1267.00	30.00	200.00	10.	9.
1359.00	30.00	200.00	10.	10.
435.00	60.00	200.00	10.	0.
544.00	60.00	200.00	10.	1.
653.00	60.00	200.00	10.	4.
740.00	60.00	200.00	10.	8.
544.00	15.00	23.00	10.	2.
707.00	15.00	23.00	10.	4.
903.00	15.00	23.00	10.	7.
946.00	15.00	23.00	10.	7.
1060.00	15.00	23.00	10.	6.
1153.00	15.00	23.00	10.	9.
1256.00	15.00	23.00	10.	8.

```
      1658.00          15.00          23.00          10.           9.

      1958.00          15.00          23.00          15.          15.
       381.00          30.00          23.00          10.           2.
       489.00          30.00          23.00          10.           3.
       636.00          30.00          23.00          10.           6.
       653.00          30.00          23.00          10.           5.

       761.00          30.00          23.00          10.           8.
       788.00          30.00          23.00          10.           8.
       903.00          30.00          23.00          10.           9.
       952.00          30.00          23.00          10.          10.
       190.00          60.00          23.00          10.           1.

       256.00          60.00          23.00          10.           2.
       337.00          60.00          23.00          10.           5.
       408.00          60.00          23.00          10.           9.
       914.00          15.00       10000.00           4.           0.
      1098.00          15.00       10000.00           4.           1.

      1631.00          15.00       10000.00           4.           2.
      1958.00          15.00       10000.00           4.           2.
      2409.00          15.00       10000.00           4.           4.
       555.00          30.00       10000.00           4.           1.
       816.00          30.00       10000.00           4.           2.

      1033.00          30.00       10000.00           4.           2.
      1213.00          30.00       10000.00           4.           3.
      1370.00          30.00       10000.00           4.           3.
      1490.00          30.00       10000.00           4.           4.
       343.00          60.00       10000.00           4.           0.

       598.00          60.00       10000.00           4.           1.
       696.00          60.00       10000.00           4.           2.
       778.00          60.00       10000.00           4.           4.
       924.00          60.00       10000.00           4.           4.
       897.00          15.00        3700.00           4.           0.

      1049.00          15.00        3700.00           4.           1.
      1223.00          15.00        3700.00           4.           3.
      1822.00          15.00        3700.00           4.           3.
      2148.00          15.00        3700.00           4.           3.
      1077.00          30.00        3700.00           4.           0.

      1185.00          30.00        3700.00           4.           2.
      1283.00          30.00        3700.00           4.           4.
       631.00          60.00        3700.00           4.           0.
       663.00          60.00        3700.00           4.           1.
       761.00          60.00        3700.00           4.           1.

      1028.00          60.00        3700.00           4.           2.
      1169.00          60.00        3700.00           4.           2.
      1213.00          60.00        3700.00           4.           4.

   Selection of observations from number  1 through  68

   Transformation of input parameters
   Conc mg/m3          is transformed logaritmically!
   Minutes             is transformed logaritmically!
   BW grams            is transformed logaritmically!
```

```
   Probit link used without background response correction!

   Variable  1  =  Transformed Conc mg/m3
   Variable  2  =  Transformed Minutes
   Variable  3  =  Transformed BW grams
   Variable  4  =  Product of transformed Conc mg/m3 and transformed Minutes


   Chi-Square          =  114.89
   Degrees of Freedom  =  63

   B0 = -1.172e+001        Student t  for B0 = -2.07
   B1 = 2.191e+000         Student t  for B1 = 2.67
   B2 = 1.456e+000         Student t  for B2 = 0.92
   B3 = -2.334e-001        Student t  for B3 = -6.74
   B4 = -7.145e-002        Student t  for B4 = -0.31

    variance  B00 = 3.212e+001
   covariance  B01 = -4.626e+000
   covariance  B02 = -8.779e+000
   covariance  B03 = 2.528e-002
   covariance  B04 = 1.265e+000
     variance  B11 = 6.714e-001
   covariance  B12 = 1.280e+000
   covariance  B13 = -3.150e-003
   covariance  B14 = -1.862e-001
     variance  B22 = 2.518e+000
   covariance  B23 = -1.309e-003
   covariance  B24 = -3.695e-001
     variance  B33 = 1.201e-003
   covariance  B34 = -2.641e-004
     variance  B44 = 5.485e-002

   Probability of correct model(p-value) is 0.000071
   The prediction of the model is not sufficient. Use for estimation of the
   95% confidence limits Student t with  63 degrees of freedom

   Correction for variances Chi-Squares/Degrees of Freedom = 1.824


   Estimation of Conc mg/m3
   Response  = 95.000000  percent
   Minutes   = 20.000000
   BW grams  = 200.000000

   Estimated Conc mg/m3  95.000000 percent = 2.228e+003
   Deviate Corresponding to Confidence Level of Interest = 1.960000
   Lower limit Conc mg/m3  95.000000 percent  = 1.865e+003
   Upper limit Conc mg/m3  95.000000 percent  = 3.029e+003


   Probability of correct model(p-value) is 0.000071
   The prediction of the model is not sufficient. Use for estimation of the
   95% confidence limits Student t with  63 degrees of freedom

   Correction for variances Chi-Squares/Degrees of Freedom = 1.824


   Estimation of response
   Conc mg/m3      = 500.000000
   Minutes   = 20.000000
```

```
   BW grams  = 200.000000

   Response   = 9.52e+000 percent
   Deviate Corresponding to Confidence Level of Interest = 1.960000
   LL-response   = 3.46e+000 percent
   UL-response   = 2.12e+001 percent


   Probability of correct model(p-value) is 0.000071
   The prediction of the model is not sufficient. Use for estimation of the
   95% confidence limits Student t with  63 degrees of freedom

   Correction for variances Chi-Squares/Degrees of Freedom = 1.824


   Estimation of ratio between regression coefficients
   Ratio between regression coefficients
   Conc mg/m3 and Minutes

   Deviate Corresponding to Confidence Level of Interest = 1.960000

   Ratio      =    1.504850

   Confidence limits
     -1.382078     4.391779
```

Appendix C: Second Example Input File (Example2.(d)) and Associated Output File

Input File:

```
3
 34
conc mg/m3
minutes
sex
exposed
responded
424000      12    1     10    10
424000      6     1     10    0
212000      36    1     10    10
212000      24    1     10    10
212000      18    1     10    6
212000      12    1     10    3
212000      6     1     15    0
106000      120   1     10    10
106000      90    1     10    10
106000      60    1     10    9
106000      30    1     10    0
106000      24    1     10    2
106000      18    1     10    1
53000       240   1     10    6
53000       120   1     10    1
53000       90    1     10    1
53000       60    1     10    0
424000      12    0     10    10
424000      6     0     10    0
212000      36    0     10    10
212000      24    0     10    10
212000      18    0     10    8
212000      12    0     10    3
212000      6     0     15    0
106000      120   0     10    10
106000      90    0     10    9
106000      60    0     10    6
106000      30    0     10    1
106000      24    0     10    0
106000      18    0     10    0
53000       240   0     10    5
53000       120   0     10    1
53000       90    0     10    0
53000       60    0     10    1

 modeling
 2 1 1 1 3
 3
 1 2 3
 1
 1 2
 1 34
 dose
 1 1.96 50 1 60 0
 response
 1 1.96 500 15 0
```

```
 ratio
 1 1.96 2 1 2
 graph/response
 1 1.96 1 0 1000 2 20 3 200
 end
```

Output File:

```
 ========================================================================
       Ten Berge Model. (Version: 1.0; Date: 12/26/2006)
       Input Data File: example2.(d)
       Gnuplot Plotting File:  example2.plt
                                        Mon Apr 02 13:25:48 2007
 ========================================================================

 Dose-Response Analysis

 Method of Maximum Likelihood according to:
 D.J. Finney, 1977. Probit Analysis. Cambridge University Press.
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

   Model: P(v1, v2, ...) = Link(B0 + B1*v1 + B2*v2 + ...)

   Link is either Logit or Probit
   v1, v2, ... are the variables (transformations of the input parameters)


   Number of input parameters = 3
   Total number of observations = 34
   Total number of records with missing values = 0


     conc mg/m3        minutes            sex        exposed      responded

      424000.00          12.00           1.00            10.            10.
      424000.00           6.00           1.00            10.             0.
      212000.00          36.00           1.00            10.            10.
      212000.00          24.00           1.00            10.            10.
      212000.00          18.00           1.00            10.             6.

      212000.00          12.00           1.00            10.             3.
      212000.00           6.00           1.00            15.             0.
      106000.00         120.00           1.00            10.            10.
      106000.00          90.00           1.00            10.            10.
      106000.00          60.00           1.00            10.             9.

      106000.00          30.00           1.00            10.             0.
      106000.00          24.00           1.00            10.             2.
      106000.00          18.00           1.00            10.             1.
       53000.00         240.00           1.00            10.             6.
       53000.00         120.00           1.00            10.             1.

       53000.00          90.00           1.00            10.             1.
       53000.00          60.00           1.00            10.             0.
      424000.00          12.00           0.00            10.            10.
      424000.00           6.00           0.00            10.             0.
      212000.00          36.00           0.00            10.            10.

      212000.00          24.00           0.00            10.            10.
```

```
   212000.00          18.00           0.00            10.             8.
   212000.00          12.00           0.00            10.             3.
   212000.00           6.00           0.00            15.             0.
   106000.00         120.00           0.00            10.            10.

   106000.00          90.00           0.00            10.             9.
   106000.00          60.00           0.00            10.             6.
   106000.00          30.00           0.00            10.             1.
   106000.00          24.00           0.00            10.             0.
   106000.00          18.00           0.00            10.             0.

    53000.00         240.00           0.00            10.             5.
    53000.00         120.00           0.00            10.             1.
    53000.00          90.00           0.00            10.             0.
    53000.00          60.00           0.00            10.             1.

Selection of observations from number  1 through  34

Transformation of input parameters
conc mg/m3          is transformed logaritmically!
minutes             is transformed logaritmically!
sex                 is not transformed at all!

Logit link used without background response correction!

Variable  1  =  Transformed conc mg/m3
Variable  2  =  Transformed minutes
Variable  3  =  sex
Variable  4  =  Product of transformed conc mg/m3 and transformed minutes


Chi-Square          =  34.02
Degrees of Freedom  =  29

B0 = -3.501e+001        Student t  for B0 = -2.69
B1 = 1.508e+000         Student t  for B1 = 1.43
B2 = -1.703e+001        Student t  for B2 = -4.87
B3 = 3.717e-001         Student t  for B3 = 0.96
B4 = 1.876e+000         Student t  for B4 = 5.90

  variance  B00 = 1.697e+002
covariance  B01 = -1.362e+001
covariance  B02 = -3.214e+001
covariance  B03 = -2.253e-001
covariance  B04 = 2.556e+000
  variance  B11 = 1.113e+000
covariance  B12 = 2.897e+000
covariance  B13 = 6.187e-003
covariance  B14 = -2.379e-001
  variance  B22 = 1.224e+001
covariance  B23 = -7.817e-002
covariance  B24 = -1.104e+000
  variance  B33 = 1.505e-001
covariance  B34 = 8.581e-003
  variance  B44 = 1.012e-001


Probability of correct model (p-value) is 0.238661
The prediction of the model is sufficient. Use for estimation of the
95% confidence limits the Standard Normal Deviate
```

```
No correction for variances required!


Estimation of conc mg/m3
Response  = 50.000000  percent
minutes   = 60.000000

Estimated conc mg/m3  50.000000 percent = 8.919e+004
Deviate Corresponding to Confidence Level of Interest = 1.960000
Lower limit conc mg/m3  50.000000 percent  = 8.304e+004
Upper limit conc mg/m3  50.000000 percent  = 9.594e+004


Probability of correct model (p-value) is 0.238661
The prediction of the model is sufficient. Use for estimation of the
95% confidence limits the Standard Normal Deviate

No correction for variances required!


Estimation of response
conc mg/m3      = 500.000000
minutes   = 15.000000

Response   = 3.52e-016 percent
Deviate Corresponding to Confidence Level of Interest = 1.960000
LL-response   = 4.55e-020 percent
UL-response   = 2.73e-012 percent


Probability of correct model (p-value) is 0.238661
The prediction of the model is sufficient. Use for estimation of the
95% confidence limits the Standard Normal Deviate

No correction for variances required!


Estimation of ratio between regression coefficients
Ratio between regression coefficients
conc mg/m3 and minutes

Deviate Corresponding to Confidence Level of Interest = 1.960000

Ratio      =    -0.088527

Confidence limits
  -0.239546     0.062492
```

Appendix D: Third Example Input File (Example3.(d)) and Associated Output File

Input File:

```
3
 84
conc mg/m3
minutes
body weight
exposed
responded
77    60    200   6     0
96    60    200   17    0
96    120   200   12    0
96    240   200   12    0
125   120   200   9     0
144   60    200   31    3
144   120   200   12    1
144   240   200   12    7
163   60    200   12    6
163   240   200   10    5
192   30    200   5     0
192   60    200   5     3
192   120   200   8     8
192   150   200   11    9
192   180   200   10    10
192   240   200   29    29
288   30    200   10    2
288   60    200   13    10
288   120   200   12    10
288   240   200   4     4
383   5     200   12    6
383   10    200   12    8
383   20    200   5     5
383   30    200   4     4
460   20    200   4     4
479   5     200   4     2
479   10    200   4     2
479   20    200   4     4
77    60    20    13    0
86    180   20    10    0
96    60    20    5     0
96    120   20    5     0
96    240   20    5     0
144   60    20    6     1
144   120   20    6     2
144   240   20    6     5
192   30    20    10    2
192   60    20    10    8
192   120   20    14    13
192   240   20    10    10
240   5     20    6     0
240   30    20    6     4
240   60    20    6     6
240   240   20    6     6
383   5     20    6     4
```

```
383   10    20    6     6
383   20    20    6     6
77    60    3000  6     0
96    60    3000  6     1
96    120   3000  6     1
144   60    3000  4     1
144   120   3000  4     3
144   240   3000  4     2
192   30    3000  2     1
192   60    3000  2     2
192   120   3000  4     3
288   30    3000  4     3
288   120   3000  3     3
383   5     3000  2     2
96    60    2500  4     0
144   60    2500  8     1
144   120   2500  6     0
144   240   2500  8     2
192   30    2500  3     1
192   60    2500  4     0
192   120   2500  4     2
192   240   2500  4     3
288   5     2500  2     0
288   60    2500  6     1
288   240   2500  4     3
383   5     2500  2     0
383   10    2500  2     1
383   20    2500  4     2
96    60    10000 1     0
96    120   10000 2     0
96    240   10000 2     0
144   60    10000 2     0
144   120   10000 2     0
144   240   10000 3     1
192   30    10000 2     0
192   120   10000 3     1
192   240   10000 2     2
288   60    10000 3     2
383   20    10000 2     2

 modeling
 2 1 1 1 3
 2
 1 2
 0
 1 84
 dose
 1 1.664 10 1 240
 response
 1 1.664 500 15
 ratio
 1 1.664 2 1 2
 graph/response
 1 1.96 1 0 1000 2 20 3 200
 end
```

Output File:

```
 ================================================================
```

```
       Ten Berge Model. (Version: 1.0; Date: 12/26/2006)
       Input Data File: example3.(d)
       Gnuplot Plotting File:  example3.plt
                                        Mon Apr 02 14:37:42 2007
 ===============================================================================

 Dose-Response Analysis

 Method of Maximum Likelihood according to:
 D.J. Finney, 1977. Probit Analysis. Cambridge University Press.
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

   Model: P(v1, v2, ...) = Link(B0 + B1*v1 + B2*v2 + ...)

   Link is either Logit or Probit
   v1, v2, ... are the variables (transformations of the input parameters)


   Number of input parameters = 3
   Total number of observations = 84
   Total number of records with missing values = 0


     conc mg/m3        minutes        exposed      responded

          77.00          60.00            6.             0.
          96.00          60.00           17.             0.
          96.00         120.00           12.             0.
          96.00         240.00           12.             0.
         125.00         120.00            9.             0.

         144.00          60.00           31.             3.
         144.00         120.00           12.             1.
         144.00         240.00           12.             7.
         163.00          60.00           12.             6.
         163.00         240.00           10.             5.

         192.00          30.00            5.             0.
         192.00          60.00            5.             3.
         192.00         120.00            8.             8.
         192.00         150.00           11.             9.
         192.00         180.00           10.            10.

         192.00         240.00           29.            29.
         288.00          30.00           10.             2.
         288.00          60.00           13.            10.
         288.00         120.00           12.            10.
         288.00         240.00            4.             4.

         383.00           5.00           12.             6.
         383.00          10.00           12.             8.
         383.00          20.00            5.             5.
         383.00          30.00            4.             4.
         460.00          20.00            4.             4.

         479.00           5.00            4.             2.
         479.00          10.00            4.             2.
         479.00          20.00            4.             4.
          77.00          60.00           13.             0.
          86.00         180.00           10.             0.
```

96.00	60.00	5.	0.
96.00	120.00	5.	0.
96.00	240.00	5.	0.
144.00	60.00	6.	1.
144.00	120.00	6.	2.
144.00	240.00	6.	5.
192.00	30.00	10.	2.
192.00	60.00	10.	8.
192.00	120.00	14.	13.
192.00	240.00	10.	10.
240.00	5.00	6.	0.
240.00	30.00	6.	4.
240.00	60.00	6.	6.
240.00	240.00	6.	6.
383.00	5.00	6.	4.
383.00	10.00	6.	6.
383.00	20.00	6.	6.
77.00	60.00	6.	0.
96.00	60.00	6.	1.
96.00	120.00	6.	1.
144.00	60.00	4.	1.
144.00	120.00	4.	3.
144.00	240.00	4.	2.
192.00	30.00	2.	1.
192.00	60.00	2.	2.
192.00	120.00	4.	3.
288.00	30.00	4.	3.
288.00	120.00	3.	3.
383.00	5.00	2.	2.
96.00	60.00	4.	0.
144.00	60.00	8.	1.
144.00	120.00	6.	0.
144.00	240.00	8.	2.
192.00	30.00	3.	1.
192.00	60.00	4.	0.
192.00	120.00	4.	2.
192.00	240.00	4.	3.
288.00	5.00	2.	0.
288.00	60.00	6.	1.
288.00	240.00	4.	3.
383.00	5.00	2.	0.
383.00	10.00	2.	1.
383.00	20.00	4.	2.
96.00	60.00	1.	0.
96.00	120.00	2.	0.
96.00	240.00	2.	0.
144.00	60.00	2.	0.
144.00	120.00	2.	0.
144.00	240.00	3.	1.
192.00	30.00	2.	0.
192.00	120.00	3.	1.

```
      192.00         240.00             2.             2.
      288.00          60.00             3.             2.
      383.00          20.00             2.             2.

Selection of observations from number  1 through  84

Transformation of input parameters
conc mg/m3          is transformed logaritmically!
minutes             is transformed logaritmically!
body weight         is not transformed at all!

Logit link used without background response correction!

Variable  1  =  Transformed conc mg/m3
Variable  2  =  Transformed minutes


Chi-Square          =  162.35
Degrees of Freedom  =  81

B0 = -3.559e+001        Student t  for B0 = -8.49
B1 = 5.525e+000         Student t  for B1 = 8.51
B2 = 1.524e+000         Student t  for B2 = 7.03

  variance  B00 = 1.756e+001
covariance  B01 = -2.694e+000
covariance  B02 = -7.794e-001
  variance  B11 = 4.218e-001
covariance  B12 = 1.100e-001
  variance  B22 = 4.697e-002


Probability of correct model(p-value) is 0.000000
The prediction of the model is not sufficient. Use for estimation of the
95% confidence limits Student t with  81 degrees of freedom

Correction for variances Chi-Squares/Degrees of Freedom = 2.004


Estimation of conc mg/m3
Response  = 10.000000  percent
minutes   = 240.000000

Estimated conc mg/m3  10.000000 percent = 9.294e+001
Deviate Corresponding to Confidence Level of Interest = 1.664000
Lower limit conc mg/m3  10.000000 percent  = 8.158e+001
Upper limit conc mg/m3  10.000000 percent  = 1.023e+002


Probability of correct model(p-value) is 0.000000
The prediction of the model is not sufficient. Use for estimation of the
95% confidence limits Student t with  81 degrees of freedom

Correction for variances Chi-Squares/Degrees of Freedom = 2.004


Estimation of response
conc mg/m3      = 500.000000
minutes   = 15.000000

Response   = 9.47e+001 percent
```

```
Deviate Corresponding to Confidence Level of Interest = 1.664000
LL-response   = 8.96e+001 percent
UL-response   = 9.73e+001 percent


Probability of correct model(p-value) is 0.000000
The prediction of the model is not sufficient. Use for estimation of the
95% confidence limits Student t with  81 degrees of freedom

Correction for variances Chi-Squares/Degrees of Freedom = 2.004


Estimation of ratio between regression coefficients
Ratio between regression coefficients
conc mg/m3 and minutes

Deviate Corresponding to Confidence Level of Interest = 1.664000

Ratio      =    3.624755

Confidence limits
  3.088489     4.161022
```

ATTACHMENT A. TEN BERGE MODEL SOURCE CODE

```
//
*************************************************************************
*
//    Converted to C from VisualBasic program of Dosresp (Dose-Response
Analysis)
//    software (W.F. ten Berge, Nov. 2001) without the visual components
//
//    by Qun He (12/26/2006)
// ***************************************************************

// *********************************************************************
//               General information of the model
// *********************************************************************

char Version_no[] = "Ten Berge Model. (Version: 1.0; Date: 12/26/2006)";
char Model_info[] = "Dose-Response Analysis\n\n Method of Maximum Likelihood
according to: \
                              \n D.J. Finney, 1977. Probit Analysis. Cambridge
University Press.";
char Formula[] = "Model: P(v1, v2, ...) = Link(B0 + B1*v1 + B2*v2 + ...)\n \
                  \n   Link is either Logit or Probit \
                        \n   v1, v2, ... are the variables (transformations
of the input parameters)\n";


// ****************************************
//      Header files
// ****************************************
#include <windows.h>    // Win32 Header File
#include <windowsx.h>   // Win32 Header File
#include <commctrl.h>   // Win32 Header File
#include <mmsystem.h>   // Win32 Header File
#include <shellapi.h>   // Win32 Header File
#include <shlobj.h>     // Win32 Header File
#include <richedit.h>   // Win32 Header File
#include <wchar.h>      // Win32 Header File
#include <objbase.h>    // Win32 Header File
#include <ocidl.h>      // Win32 Header File
#include <winuser.h>    // Win32 Header File
#include <olectl.h>     // Win32 Header File
#include <conio.h>
#include <direct.h>
#include <ctype.h>
#include <io.h>
#include <math.h>
#include <stdio.h>
#include <string.h>
#include <stddef.h>
#include <stdlib.h>
#include <setjmp.h>
#include <time.h>
#include <stdarg.h>
#include <process.h>

#include <benchmark.h>
#include <ERRORPRT.h>
```

```
#include <allo_memo.h>
#include <matrix_agb.h>
#include <specialfun.h>
#include <computation.h>
#include <in_outfun.h>

// ************************************************
//                System Variables
// ************************************************

char    BCX_STR [1024*1024];
jmp_buf GosubStack[32];
int     GosubNdx;
#define DEBUG 1;

// ************************************************
//            User Global Variables
// ************************************************

static int     I;
static int     J;
static int     SW;
static FILE*  FP2;
static FILE* logfile;
static FILE* response;

#define NUMOFVAR 14;           /* See variable Vrx[][] */
#define LENOFNAME 20;          /* See variable Vrx[][] */

int NZ;                        /* No. of variables */
int NW;                        /* No. of observations */
int NY;                        /* No. of selected single transformed variable
for analysis */
int NC;                        /* No. of pairs of product of transformed
variables */
int NX;                        /* Total No. (selected single variable +
selected pair of variables) */
int KBZ;                       /* (1=without; 2=with) background response
correction */
int KPZ;                       /* (1=Probit; 2=Logit) model */
int KZ, KZY, HQ, MQ, NXT;
int NB;                        /* Flag of error in transform variables */
int MV;                        /* Index of requested variable for evaluation
*/
int NS, MX, DF;
int N1;                        /* First sequence number of trials for analysis
*/
int N2;                        /* Last sequence number of trials for analysis
*/
int TW;
int NXC;                       /* No. of variables selected for product
analysis */
double CH;                     /* Chi-Squares value */
double CX;                     /* Correction for variances (Chi-
Squares/Degrees of Freedom) */
double ZM, P;
double PW;                     /* Requested response in percentage (set to 50
when user specify <= 0) */
double Q, SQ, Y, Z;
double XZ;
double TX;                     /* Value of Student t set by user */
```

```
double CZ, DC, DW;
double AW, BW, CW, Cy, RA, RB;
int K0[10];                    /* List of selected single transformed variable
for analysis */
int K1[5];                     /* List of the left variables for the product
analysis */
int K2[5];                     /* List of the right variables for the product
analysis */
int WF[12];
int NT[10];                    /* List of variables' type of transformation
(logrithmic, reciprocal, none) */
double XU[10];                 /* List of transformed values for each variable
*/
double XW[10], PA[10];
double AX[10], BX[10], FX[10];
double ZA[10][10], BV[10][10], ZC[10][10], ZB[10];
int TR;                        /* Number of variables chosen for ratio
regression coefficient evaluation */
int NRC1;                      /* First variable chosen for ration regression
coefficient evaluation */
int NRC2;                      /* Second variable chosen for ration regression
coefficient evaluation */
int FlEr;                      /* Error flag */
char Fina[128];                /* Input file name */
int *N;                        /* No. of subjects array */
int *LR;                       /* No. of responders array */
double **X;                    /* Input data matrix */
char Vrx[14][40];              /* Input data column names */
char conc_col[20];
char expos_time_col[20];
char subj_prop_col[20];
char num_expos_col[20];
char num_respd_col[20];
char comment_line[128];
int num_missing_value;         /* No. of records with missing values */
int stdchisqr;                 /* Flag indicates Chisquare() calculated */
int chisqr_skip = 1;           /* Flag the print out of Chisquare()
calculattion */
time_t  ltime;

// ************************************************
//       Define input and output files's name
// ************************************************

char fout[128];                     /* output temp file */
char fout2[128];
char plotfilename[128];       /* file to send to GnuPlot */
char infilestem[128];       /* input file name stem */
char respgraphfile[128];     /* graphics file name */

// ************************************************
//               Standard Macros
// ************************************************

#define VAL(a)(double)atof(a)


// ************************************************
//               Standard Prototypes
// ************************************************
```

```
char*   BCX_TmpStr(size_t);
char*   str (double);
char*   join (int, ... );
double  Abs (double);
double  Exp (double);

// ************************************************
//               User's Prototypes
// ************************************************

void    RetrDat (void);
void    SuPPDat (void);
void    Filcomb3 (void);
void    nrm21 (int);
void    nrm22 (void);
void    nrm23 (void);
void    nrm24 (void);
void    nrm25 (int);
void    nrm26 (void);
void    nrm28 (void);
void    nrm29 (void);
void    pqa (void);
void    pqb (void);
void    matrinv (double A[10][10], double B[10][10]);
void    matrix (double A[10][10], double B[10], double C[10]);
void    CalcML (void);
void    CalcDoseResponse(void);
void    CalcResponseDose(void);
void    CalcResponseGraph(void);
void    Graphic2(double Dx,  double *Px1, double *Px2, double *Px3, double
SV[]);
void    CalcRatio(void);
void    nrm41 (int);
void    nrm42 (void);
void    nrm43 (void);
void    nrm44 (void);
void    Chisquare (void);
void    Warn1 (double QZ);
void    Warn2 (double QZ);
void    WriDat (void);
void    CalcErr (void);
void    check_args(int argc, char *argv[]);
void    OPEN_FILES(int argc, char *argv[]);
void    read_modeling_data(void);
int     READ_OBSDATA (int r, int c, double **m);
void    output_read_data(void);
void    CLOSE_FILES(void);
void    calML_H1(void);
void    NRM2BGR(void);
void    Get_File_Stem(char *argv, char *infilestem);
void    Derive_File_Name(char *stem, char *newfile, const char *ext);

int     check_data_sectors(const char *sector);


// ************************************************
//                  Main Program
// ************************************************

int main(int argc, char *argv[])
{
```

```
  char sector[20];
  int compare;
  time(&ltime);

  check_args(argc, argv);
  Get_Names(argv[1], fout, fout2, plotfilename);    /* get input and output
file names */
  Get_File_Stem(argv[1], infilestem);               /* get input file name
stemp */
  Derive_File_Name(infilestem, respgraphfile, "Asc");/* derive graphics file
name from infile stem */
  OPEN_FILES(argc, argv);                           /* open output files */
  read_modeling_data();                                    /* read input
file */

  Output_Header(Version_no, argv[1], plotfilename, ctime(&ltime), Model_info);
  output_read_data();                               /* output some input data
*/

  CalcML();
  WriDat();

  fscanf(fp_in, "%s", sector);
  compare = check_data_sectors(sector);

  while (compare != 0) {
        if (compare == 1)
              CalcDoseResponse();
        else if (compare == 2)
              CalcResponseDose();
        else if (compare == 3)
              CalcRatio();
        else if (compare == 4)
              CalcResponseGraph();

        fscanf(fp_in, "%s", sector);
        compare = check_data_sectors(sector);
  }

  FREE_IVECTOR (N,1,NW);
  FREE_IVECTOR (LR,1,NW);
  FREE_DMATRIX(X, 1, NW, 1, NZ+2);
  CLOSE_FILES();                                    /* close all opened files
*/

  return 0;
}                                                   /* End of main program */


// ************************************************
//                  Run Time Functions
// ************************************************

char *BCX_TmpStr (size_t Bites)
{
  static int   StrCnt;
  static char *StrFunc[2048];
  StrCnt=(StrCnt + 1) & 2047;
  if(StrFunc[StrCnt]) free (StrFunc[StrCnt]);
  return StrFunc[StrCnt]=(char*)calloc(Bites+128,1);
}
```

```
char *str (double d)
{
  register char *strtmp = BCX_TmpStr(16);
  sprintf(strtmp,"% .15G",d);
  return strtmp;
}

char * join(int n, ...)
{
  register int i = n, tmplen = 0;
  register char *s_;
  register char *strtmp = 0;
  va_list marker;
  va_start(marker, n); // Initialize variable arguments
  while(i-- > 0)
  {
    s_ = va_arg(marker, char *);
    tmplen += strlen(s_);
  }
  strtmp = BCX_TmpStr(tmplen);
  va_end(marker); // Reset variable arguments
  i = n;
  va_start(marker, n); // Initialize variable arguments
  while(i-- > 0)
  {
    s_ = va_arg(marker, char *);
    strtmp = strcat(strtmp, s_);
  }
  va_end(marker); // Reset variable arguments
  return strtmp;
}


double Exp (double arg)
{
  return pow(2.718281828459045,arg);
}


double Abs (double a)
{
  if(a<0) return -a;
  return  a;
}

// ***********************************
//       User Functions
// ***********************************

void check_args(int argc, char *argv[]) {
  if(argc == 2)
        show_version(argv[1], Version_no);

  if(argc < 2) {
      fprintf(stderr, "ERROR:  Requires two arguments\nUsage:  %s
<file.(d)>\n", argv[0]);
      fprintf (stderr, "   or:  %s -v for version number.\n", argv[0]);
      exit (1);
  }
```

```
}

int check_data_sectors(const char *sector)
{
      int compare = strcmp("end", sector);

      if (compare == 0)
            return 0;

      compare = strcmp("dose", sector);

      if (compare == 0)
            return 1;

      compare = strcmp("response", sector);

      if (compare == 0)
            return 2;

      compare = strcmp("ratio", sector);

      if (compare == 0)
            return 3;

      compare = strcmp("graph/response", sector);

      if (compare == 0)
            return 4;
}

// ************************************************
//  OPEN_FILES--used to open input and output files.
// ************************************************

void OPEN_FILES (int argc, char *argv[]) {
  logfile = fopen("log.txt", "a");  /* open log file */
  fp_in = fopen(argv[1], "r");      /* open input file */

  // open output files
  fp_out = fopen(fout, "w");      /* overwrite output */
  fp_out2 = fopen(fout2, "w");          /* always overwrite plotting file */

  // check to make sure files are open, if not, print error message and exit
  if (fp_in == NULL || fp_out == NULL || fp_out2 == NULL) {
      fprintf(logfile, "Error in opening input and output files.");
      ERRORPRT ("Error in opening input and output files.");
  }

  response = fopen(respgraphfile, "w");  /* open response graph file */
}

// ************************************************
//  CLOSE_FILES--used to close input and output files.
// ************************************************

void CLOSE_FILES (void) {
  if (fclose(fp_in) != 0 || fclose(fp_out) != 0 || fclose(fp_out2) != 0) {
        fprintf(logfile, "Error in closing opened files.");
        fclose(response);
        fclose(logfile);
        ERRORPRT ("Error in closing opened files.");
```

```
  }

  fclose(response);
  fclose(logfile);
}


void read_modeling_data () {
  char junk[255];
  fscanf(fp_in, "%d", &NZ);                          /* begin reading input from
input .(d) file */
  fscanf(fp_in, "%d", &NW);
  fgets(junk, 128, fp_in);                           /* to get the junk out of
the previous line */

  int i;
  Vrx[0][0] = 0;
  for (i = 1; i < 14; i++) {
        if (i <= NZ + 2) {
              fgets(Vrx[i], 40, fp_in);
              Vrx[i][strlen(Vrx[i]) - 1] = 0;         /* strip carriage return
*/
        } else {
              Vrx[i][0] = 0;
        }
  }

  #ifdef DEBUG
  fprintf(logfile, "\n############################ %s\n", ctime(&ltime));
  fprintf(logfile, "Reading data from input .(d) file...\n\n");
  fprintf(logfile, "number of variables: %d \n", NZ);
  fprintf(logfile, "number of observations: %d \n", NW);
  fprintf(logfile, "concentration column: %s \n", Vrx[1]);
  fprintf(logfile, "exposure time: %s \n", Vrx[2]);
  fprintf(logfile, "subject property: %s \n", Vrx[3]);
  fprintf(logfile, "number of exposed: %s \n", Vrx[4]);
  fprintf(logfile, "number of reponded: %s \n", Vrx[5]);
  #endif

  N = IVECTOR(1, NW);
  LR = IVECTOR(1, NW);
  X = DMATRIX (1, NW, 1, NZ);          /* init data variables */
  num_missing_value = READ_OBSDATA(NW, NZ, X);

  #ifdef DEBUG
  fprintf(logfile, "\n\nNumber of missing value = %d\n", num_missing_value);
  fprintf(logfile, "Reading input data file finished.\n");
  #endif

  fscanf(fp_in, "%s", comment_line);
  fscanf(fp_in, "%d%d", &KPZ, &KBZ);

  for (i = 1; i <= NZ; i++) {
        fscanf(fp_in, "%d", &NT[i]);
  }

  fscanf(fp_in, "%d", &NY);

  for(i = 1; i <= NY; i++) {
        fscanf(fp_in, "%d", &K0[i]);
  }
```

```
  fscanf(fp_in, "%d", &NC);

  for(i = 1; i <= NC; i++) {
        fscanf(fp_in, "%d%d", &K1[i], &K2[i]);
  }

  fscanf(fp_in, "%d%d", &N1, &N2);                /* first & last sequence
number of trials for analysis */

  #ifdef DEBUG
  fprintf(logfile, "\n\nUser selections\n");
  fprintf(logfile, "%s\n", comment_line);
  fprintf(logfile, "KPZ=%d\n", KPZ);
  fprintf(logfile, "KBZ=%d\n", KBZ);
  fprintf(logfile, "NT[1]=%d\n", NT[1]);
  fprintf(logfile, "NT[2]=%d\n", NT[2]);
  fprintf(logfile, "NT[3]=%d\n", NT[3]);
  fprintf(logfile, "NY=%d\n", NY);
  fprintf(logfile, "K0[1]=%d\n", K0[1]);
  fprintf(logfile, "K0[2]=%d\n", K0[2]);
  fprintf(logfile, "K0[3]=%d\n", K0[3]);
  fprintf(logfile, "NC=%d\n", NC);
  fprintf(logfile, "K1[1]=%d\n", K1[1]);
  fprintf(logfile, "K2[1]=%d\n", K2[1]);
  fprintf(logfile, "N1=%d\n", N1);
  fprintf(logfile, "N2=%d\n", N2);
  #endif
}

void output_read_data(void) {
  fprintf(fp_out, "\n   %s", Formula);
  fprintf(fp_out, "\n\n   Number of input parameters = %d", NZ);
  fprintf(fp_out, "\n   Total number of observations = %d", NW);
  fprintf(fp_out, "\n   Total number of records with missing values = %d\n\n",
num_missing_value);

  fprintf(fp_out2,"\n\n Model calculations done.");
}

void nrm21 (int JM)
{
  NB = 0;
  int IV;
  for(IV = 1; IV <= NZ; IV++) {
        if (NT[IV] == 1) {
              if (X[JM][IV] == 0) {
                    NB = 1;
                    return;
              } else {
                    XU[IV] = log(X[JM][IV]);
              }
        }

        if (NT[IV] == 2) {
              if (X[JM][IV] == 0) {
                    NB = 1;
                    return;
              } else {
                    XU[IV] = 1 / X[JM][IV];
              }
```

```
        }

        if (NT[IV] == 3) {
              XU[IV] = X[JM][IV];
        }

#ifdef DEBUG
fprintf(logfile, "X[%d][%d]=%f XU[%d]=%f\n", JM, IV, X[JM][IV], IV, XU[IV]);
#endif
  }//end for(IV = 1; IV <= NZ; ++IV)
}


void nrm22 (void)
{
  int IW, NQ;

  if (NY == 0) {
      return;
  }

  for (IW = 1; IW <= NY; IW++) {
      NQ = K0[IW];
      XW[IW] = XU[NQ];
  }
}


void nrm23 (void)
{
  int IX, H1, H2;

  if (NC == 0) {
        return;
  }

  for (IX = 1; IX <= NC; IX++) {
        H1 = K1[IX];
      H2 = K2[IX];
      XW[NY+IX] = XU[H1] * XU[H2];
  }
}


void nrm24 (void)
{
  int I, J;
  for(I = MX; I <= NXT; I++) {
        for(J = MX; J <= NXT; J++) {
              ZA[I][J] = 0;
        }

        ZB[I] = 0;
  }

  CH = 0;
  NS = 0;
  CX = 0;
}

void nrm25 (int JM)
```

```
{
  int H, K;

  if (KBZ == 1) {
        RA = N[JM] / (P * Q);
        RB = (LR[JM] + 0.0) / N[JM] - P;    /* we don't realy want a int / int
result */

#ifdef DEBUG
        fprintf(logfile, "RA=%f RB=%f N[%d]=%d LR[%d]=%d P=%f Q=%f\n", RA, RB,
JM, N[JM], JM, LR[JM], P, Q);
#endif
  } else if(KBZ == 2) {
        RA = N[JM] / (PW * Q * SQ);
        RB = (LR[JM] + 0.0) / N[JM] - PW;
  }

  CH = CH + RA * pow(RB,2);
  #ifdef DEBUG
  fprintf(logfile, "CH=%f\n", CH);
  #endif
  NS += 1;

  for(H = MX; H <= NXT; H++) {
        for(K = MX; K <= NXT; K++) {
              //ZA[H][K] = ZA[H][K] + RA * PA[H] * PA[K];
              ZA[K][H] = ZA[K][H] + RA * PA[H] * PA[K];
        }

        ZB[H] = ZB[H] + RA * RB * PA[H];
  }
}


void nrm26 (void)
{
  int I, J;

  for (I = MX; I <= NXT; I++) {
        for (J = MX; J <= NXT; J++) {
              ZC[J][I] = ZA[J][I];
        }
    }
}


void nrm28 (void)
{
  int I;
  ZM = 0;

  for(I = MX; I <= NXT; I++) {
      BX[I] = BX[I] + FX[I];
      ZM = ZM + Abs(FX[I] / BX[I]);
        //fprintf(fp_out,"\n   BX[%d]=%f",(int)I, BX[I]);
  }

  CX = 1;
  DF = NS - NXT + MX - 1;
}
```

```
void nrm29 (void)
{
//print out
}


void pqa (void)
{
  double S;

  if (KPZ == 1) {
        S = 1 / (1 + 0.2316419 * Abs(Y));
      Z = Exp(-(pow(Y,2)) / 2) / sqrt(6.283185307);
      P = Z * (S * 0.31938153 - pow(S,2) * 0.356563782 + pow(S,3) *
1.781477937 - pow(S,4) * 1.821255978 + pow(S,5) * 1.330274429);

        if (Y > 0) {
              P = 1 - P;
        }
  } else if (KPZ == 2) {
      P = Exp(Y) / (Exp(Y) + 1);
      Z = P * (1 - P);
  }
}


void pqb (void)
{
  double QT, TQ, AQ, BQ;
  QT = PW / 100;

  if (KPZ == 1) {
        if (QT > 0.5) {
              QT = 1 - QT;
        }

      TQ = sqrt(log(1 / pow(QT,2)));
      AQ = 2.515517 + 0.802853 * TQ + 0.010328 * pow(TQ,2);
      BQ = 1 + 1.432788 * TQ + 0.189269 * pow(TQ,2) + 0.001308 * pow(TQ,3);
      Y = TQ - AQ / BQ;

        if (PW < 50) {
          Y = -Y;
      }

  } else if (KPZ == 2) {
        Y = log(QT / (1 - QT));
  }
}


void matrinv (double A[10][10], double BZ[10][10])
{
  int HA, HB, HS, HX, HY, I, J;
  double G;

  #ifdef DEBUG
  fprintf(logfile, "\n\nmatrinv function:\n");
  #endif

  for(I = MX; I <= NXT; I++) {
```

```
        for(J = MX; J <= NXT; J++) {
          BZ[J][I] = 0;
              if(I == J) {
              BZ[J][I] = 1;
              }
        }
  }

  HS = 1;
  HA = MX;
  HB = NXT - 1;

  for(HY = 1; HY <= 2; HY++) {
        #ifdef DEBUG
        fprintf(logfile, "HS=%d\n", HS);
        fprintf(logfile, "HY=%d\n", HY);
        #endif

        for(HX = HA; HS >= 0 ? HX <= HB : HX >= HB; HX += HS) {
              #ifdef DEBUG
              fprintf(logfile, "HX=%d\n", HX);
              #endif

              for(I = HX+HS; HS >= 0 ? I <= HB+HS : I >= HB+HS; I += HS) {
                    #ifdef DEBUG
                    fprintf(logfile, "I=%d\n", I);
                    fprintf(logfile, "A[%d][%d]=%f\n", HX, HX, A[HX][HX]);
                    #endif

                    if(A[HX][HX] != 0) {
                   //G = -A[I][HX] / A[HX][HX];
                          G = -A[HX][I] / A[HX][HX];
                          for(J = MX; J <= NXT; J++) {
                      //A[I][J] = A[I][J] + G * A[HX][J];
                      //BZ[I][J] = BZ[I][J] + G * BZ[HX][J];
                                A[J][I] = A[J][I] + G * A[J][HX];
                      BZ[J][I] = BZ[J][I] + G * BZ[J][HX];
                          }
                    }
              }
        }

        HS = -1;
      HA = NXT;
      HB = MX + 1;
  }

  for(I = MX; I <= NXT; I++) {
      G = A[I][I];

        for(J = MX; J <= NXT; J++) {
          //A[I][J] = A[I][J] / G;
          //BZ[I][J] = BZ[I][J] / G;
              A[J][I] = A[J][I] / G;
          BZ[J][I] = BZ[J][I] / G;
        }
  }
}

void matrix (double A[10][10], double B[10], double C[10])
{
```

```
  int K, L, HM, I, J;
  double D, E, U, UH;

  for (I = MX; I <= NXT; I++) {
        C[I] = 0;
  }

  for (I = MX; I <= NXT - 1; I++) {
        UH = A[I][I];
      HM = I;

        for (K = I+1; K <= NXT; K++) {
              //if (Abs(UH) < Abs(A[K][I])) {
              if (Abs(UH) < Abs(A[I][K])) {
              HM = K;
              }
        }

        if (HM != I) {
              for (K = MX; K <= NXT; K++) {
              //U = A[HM][K];
              //A[HM][K] = A[I][K];
              //A[I][K] = U;
                    U = A[K][HM];
              A[K][HM] = A[K][I];
              A[K][I] = U;
              }

          U = B[HM];
          B[HM] = B[I];
          B[I] = U;
        }

        for (K = I+1; K <= NXT; K++) {
              if (Abs(A[I][I]) > 0) {
              //D = A[K][I] / A[I][I];
                    D = A[I][K] / A[I][I];
              }

              for (L = MX; L <= NXT; L++) {
              //A[K][L] = A[K][L] - D * A[I][L];
                    A[L][K] = A[L][K] - D * A[L][I];
              }

              B[K] = B[K] - D * B[I];
        }
  }

  for (I = NXT; I >= MX; I--) {
      E = 0;

        for (K = MX; K <= NXT; K++) {
          //E = E + C[K] * A[I][K];
              E = E + C[K] * A[K][I];
        }

        if (Abs(A[I][I]) > 0) {
          C[I] = (B[I] - E) / A[I][I];
        }
  }
}
```

```
void CalcML (void)
{
  int I, J, TRX;
  TRX = 0;

  if (N1 < 1 || N2 > NW || N2 < N1){
    return;
  }

  NX = NY + NC;

  if(NX == 0){
    return;
  }

  MX = 0;
  XW[0] = 1;

  if(KBZ == 1){
        NXT = NX;
  }

  if(KBZ == 2){
        NXT = NX + 1;
  }

  #ifdef DEBUG
  fprintf(logfile, "\n\nKBZ=%d MX=%d NXT=%d\n", KBZ, MX, NXT);
  #endif

  for(I = 0; I <= NX; I++){
        BX[I] = 0;
  }

  if(KPZ == 1){
        BX[0] = 5;
  }

  BX[NX + 1] = 0.01;

  for (I = 0; I < 10; I++) {
      #ifdef DEBUG
        fprintf(logfile, "Initital BX[%d]=%f\n", I, BX[I]);
      #endif
  }

  do {
        //fprintf(fp_out, "\nCalML called %d times. ZM=%f", TRX, ZM);
        nrm24();
        SQ = 1 - BX[NX + 1];

        for(J = N1; J <= N2; J++) {
              if(KBZ == 1) {
                    nrm21(J);
                    calML_H1();
                    Q = 1 - P;

                    for(I = 0; I <= NX; I++) {
                          PA[I] = Z * XW[I];
                    }
```

```
            } else if(KBZ == 2) {//end if(KBZ == 1)
                  nrm21(J);
                  if(NB == 1) {
                        P = 0;
                        Z = 0;
                  } else {
                        calML_H1();
                  }
                  NRM2BGR();
            }//end if(KBZ == 2)

            nrm25(J);
      }//end for(J = N1; J <= N2; ++J)

      nrm26();
      matrix(ZA, ZB, FX);
      nrm28();
      TRX += 1;
      //fprintf(fp_out, "\nAt loops end - CalML called %d times. ZM=%f",
TRX, ZM);
  } while (TRX <= 100 && ZM >= 0.0001); //end while(TRX <= 20 && ZM >= .0001)

  matrinv(ZC, BV);
  Chisquare();
  nrm29();

  #ifdef DEBUG
  fprintf(logfile, "\n\nCalcML() finished.\n");
  #endif

  return;
}

void calML_H1(void) {
  nrm22();
  nrm23();
  Y = -5;

  if(KPZ == 2) {
        Y = 0;
  }

  for(I = 0; I <= NX; I++) {
        Y = Y + BX[I] * XW[I];
  }

  pqa();
}

void NRM2BGR(void) {
  Q = 1 - P;
  PW = BX[NX + 1] + P * SQ;

  for(I = 0; I <= NX; I++) {
        PA[I] = Z * XW[I] * SQ;
  }

  PA[NX + 1] = Q;
}
```

```
void CalcDoseResponse(void)
{
      int i, j, IQ, JQ;
      int Plus = 0;
      int student;                        /*Student t indicator*/

      fscanf(fp_in, "%d%lf%lf%d", &student, &TX, &PW, &MV);

      for (i = 1; i <= NY; i++) {
            if (K0[i] == MV)
                  Plus = 1;
      }

      for (i = 1; i <= NC; i++) {
            if (K1[i] == MV || K2[i] == MV)
                  Plus = 1;
      }

      if (Plus == 0)
            return;

      for (i = 1; i <= NZ; i++)
            XU[i] = 0;                     /* initialize XU[3] */


      for (i = 1; i <= NY; i++) {
            for (j = 1; j <= NZ; j++) {
                  if (K0[i] == j) {
                        XU[j] = j;
                  }
            }
      }

      for (i = 1; i <= NC; i++) {
            for (j = 1; j <= NZ; j++) {
                  if (K1[i] == j || K2[i] == j) {
                        XU[j] = j;
                  }
            }
      }

      for (i = 1; i <= NZ; i++) {
            if (i != MV && XU[i] != 0)
                  fscanf(fp_in, "%lf", &XU[i]);
      }

      if (PW <= 0)
            PW = 50;

      if (student == 1) {
            Chisquare();
      } else {
            TX = 0;
      }

      fprintf(fp_out, "\n   Estimation of %s", Vrx[MV]);
      fprintf(fp_out, "\n   Response\t = %lf  percent", PW);

      pqb();

    if (KPZ == 1)
```

```
            Y = Y + 5;

      AX[0] = BX[0] - Y;

      for (i = 1; i <= NX; i++)
            AX[i] = BX[i];

      XU[MV] = 1;

      for (i = 1; i <= NZ; i++) {
            if (i != MV && XU[i] != 0) {
                  fprintf(fp_out, "\n   %s\t = %lf", Vrx[i], XU[i]);
                  nrm41(i);
            }
      }

      nrm22();
      nrm23();

      XW[0] = 1;
      AW = 0;
      BW = 0;
      CW = 0;
      XZ = 0;
      Cy = 0;

      for (i = 0; i <= NX; i++) {
            IQ = 0;

            if (i > 0 && i < NY + 1)
                  IQ = K0[i];

            MQ = i;
            nrm42();

            if (IQ == MV && HQ == 1) {
                  Cy += AX[i];
                  goto nrm4c;
            }

            if (HQ == 2) {
                  Cy += AX[i] * XW[i];
                  goto nrm4c;
            }

            XZ = XZ - AX[i] * XW[i];

nrm4c:
            for (j = 0; j <= NX; j++) {
                  JQ = 0;

                  if (j > 0 && j < NY + 1)
                        JQ = K0[j];

                  if (HQ == 2)
                        goto nrm4d;

                  MQ = j;
                  nrm42();

nrm4d:
```

```
                  CZ = (AX[i] * AX[j] - pow(TX, 2) * BV[j][i] * CX) * XW[i] *
XW[j];

                  if (HQ == 1) {
                        if (IQ != MV && JQ != MV)
                              CW += CZ;

                        if (IQ == MV && JQ != MV)
                              BW += CZ;

                        if (JQ == MV && IQ != MV)
                              BW += CZ;

                        if (IQ == MV && JQ == MV)
                              AW += CZ;
                  }

                  if (HQ == 2) {
                        if (IQ != MV && JQ != MV && i != j)
                              BW += CZ;
                        else
                              AW += CZ;
                  }
            }
      }

      XZ = XZ / Cy;
      fprintf(fp_out, "\n\n   Estimated %s  %lf percent = ", Vrx[MV], PW);
      nrm43();

      if (student == 1)
            fprintf(fp_out, "   Deviate Corresponding to Confidence Level of
Interest = %f\n", TX);

      if (DF == 0) {
            char Msg1[] = "number of degrees of freedom = ";
            char Msg2[] = "Confidence limits cannot be calculated!";
            fprintf(stderr,"\n%s%d\n%s\n", Msg1, DF, Msg2);
            fprintf(fp_out,"\n   %s%d\n%s\n", Msg1, DF, Msg2);

            return;
      }

      DC = pow(BW, 2) - 4 * AW * CW;

      if (DC <= 0) {
            char Msg1[] = "95% confidence limits cannot be calculated!";
            char Msg2[] = "Try a smaller Student t or standard normal";
        char Msg3[] = "deviate with smaller confidence probability!";
            fprintf(stderr,"\n%s\n%s\n%s\n", Msg1, Msg2, Msg3);
            fprintf(fp_out,"\n   %s\n%s\n%s\n", Msg1, Msg2, Msg3);

            return;
      }

      DW = sqrt(DC);
      XZ = (-BW - DW) / (2 * AW);
      fprintf(fp_out,"   Lower limit %s  %lf percent  = ", Vrx[MV], PW);
      nrm43();

      XZ = (-BW + DW) / (2 * AW);
```

```
      fprintf(fp_out,"   Upper limit %s  %lf percent  = ", Vrx[MV], PW);
      nrm43();
}

void CalcResponseDose(void)
{
      int I, J;
      double VY, SY, Y1;
      int student;                        /*Student t indicator*/

      if (NX == 0)
            return;

      fscanf(fp_in, "%d%lf", &student, &TX);

      for (I = 1; I <= NZ; I++)
            XU[I] = 0;                      /* initialize XU[3] */

      for (I = 1; I <= NY; I++) {
            for (J = 1; J <= NZ; J++) {
                  if (K0[I] == J) {
                        XU[J] = J;
                  }
            }
      }

      for (I = 1; I <= NC; I++) {
            for (J = 1; J <= NZ; J++) {
                  if (K1[I] == J || K2[I] == J) {
                        XU[J] = J;
                  }
            }
      }

      for (I = 1; I <= NZ; I++) {
            if (XU[I] != 0)
                  fscanf(fp_in, "%lf", &XU[I]);
      }

      if (student == 1) {
            Chisquare();
      } else {
            TX = 0;
      }

      fprintf(fp_out, "\n   Estimation of response");

      for (I = 1; I <= NZ; I++) {
            if (XU[I] != 0)
                  fprintf(fp_out, "\n   %s\t = %lf", Vrx[I], XU[I]);

            nrm41(I);
      }

      nrm22();
      nrm23();

      XW[0] = 1;
      VY = 0;
      Y = 0;
```

```
      if (KPZ == 1)
            Y = -5;

      for (I = 0; I <= NX; I++) {
            for (J = 0; J <= NX; J++)
                  VY += XW[I] * XW[J] * BV[J][I] * CX;
            Y += BX[I] * XW[I];
      }

      SY = sqrt(VY);
      Y1 = Y;

      pqa();
      fprintf(fp_out, "\n\n   Response   = ");
      nrm44();

      if (student == 1)
            fprintf(fp_out, "   Deviate Corresponding to Confidence Level of
Interest = %f\n", TX);

      if (DF == 0) {
            fprintf(fp_out, "   Degrees of freedom = 0");
            return;
      }

      Y = Y1 - TX * SY;

      pqa();
      fprintf(fp_out, "   LL-response   = ");
      nrm44();

      Y = Y1 + TX * SY;

      pqa();
      fprintf(fp_out, "   UL-response   = ");
      nrm44();

      return;
}

void CalcResponseGraph(void)
{
      double Wdx, Dx, DX1, DX2;
      double Px1, Px2, Px3;
      double SV[NZ];
      int student;                        /*Student t indicator*/
      int index;

      fscanf(fp_in, "%d%lf%d%lf%lf", &student, &TX, &MV, &DX1, &DX2);

      if (student == 0)
            TX = 0;

      for (I = 1; I <= NZ; I++)
            SV[I] = 0;                     /* initialize XU[3] */

      for (I = 1; I <= NY; I++) {
            for (J = 1; J <= NZ; J++) {
                  if (K0[I] == J) {
                        SV[J] = J;
                  }
```

```
            }
      }

      for (I = 1; I <= NC; I++) {
            for (J = 1; J <= NZ; J++) {
                  if (K1[I] == J || K2[I] == J) {
                        SV[J] = J;
                  }
            }
      }

      for (I = 1; I <= NZ; I++) {
            if (SV[I] != 0 && I != MV) {
                  fscanf(fp_in, "%d%lf", &index, &SV[I]);
            }
      }

      Wdx = DX2 - DX1;

      if (Wdx <= 0)
            return;

      if (DX1 = 0)
            DX1 = Wdx / 100;

      Graphic2(DX1, &Px1, &Px2, &Px3, SV);

      for (Dx = DX1; Dx <= DX2; Dx += Wdx/100) {
            Graphic2(Dx, &Px1, &Px2, &Px3, SV);

            fprintf(response, "   %f", Dx);
            fprintf(response, "   %f", Px1);
            fprintf(response, "   %f", Px2);
            fprintf(response, "   %f\n", Px3);
      }
}

void Graphic2(double Dx,  double *Px1, double *Px2, double *Px3, double SV[])
{
      int I, J;
      double SY, VY, Y1;

      for (I = 1; I <= NZ; I++) {
            if (MV == I)
                  XU[I] = Dx;
            else
                  XU[I] = SV[I];

            nrm41(I);
      }

      nrm22();
      nrm23();

      XW[0] = 1;
      VY = 0;
      Y = 0;

      if (KPZ == 1)
            Y = -5;
```

```
	for (I = 0; I <= NX; I++) {
		for (J = 0; J <= NX; J++ )
			VY += XW[I] * XW[J] * BV[J][I] * CX;

		Y += BX[I] * XW[I];
	}

	SY = sqrt(VY);
	Y1 = Y;


	pqa();

	*Px1 = P;

	if (DF == 0)
		return;

	Y = Y1 - TX * SY;

	pqa();

	*Px2 = P;

	Y = Y1 + TX * SY;

	pqa();

	*Px3 = P;
}

void CalcRatio(void)
{
	int H, I, J;
	double RL, RV;
	int student;				/* Student t indicator */
	char Temp[14][40];			/* Temporary column names array */

	NX = NY + NC;
	fscanf(fp_in, "%d%lf%d", &student, &TX, &TR);

	if (student == 0)
		TX = 0;

	if (NX < 2 || TR != 2)
		return;

	fscanf(fp_in, "%d%d", &NRC1, &NRC2);

	for (I = 1; I <= NY; I++) {
		strcpy(Temp[I], Vrx[K0[I]]);
	}

	for (I = NY + 1; I <= NX; I++) {
		strncat(Vrx[K1[I-NY]], " ; ", 3);
		int leng = strlen(Vrx[K2[I-NY]]);
		strncat(Vrx[K1[I-NY]], Vrx[K2[I-NY]], leng);
		strcpy(Temp[I], Vrx[K1[I-NY]]);
	}

	if (student == 1) {
```

```
            Chisquare();
      } else {
            TX = 0;
      }

      fprintf(fp_out, "\n   Estimation of ratio between regression
coefficients");
      fprintf(fp_out, "\n   Ratio between regression coefficients\n   %s and
%s", Temp[NRC1], Temp[NRC2]);

      I = NRC1;
      J = NRC2;

      if (I > NZ || J > NZ)
            return;

      RL = BX[I] / BX[J];
      RV = BV[I][I] / pow(BX[I], 2) + BV[J][J] / pow(BX[J], 2) - 2 * BV[J][I]
/ BX[I] / BX[J];
      RV = RV * CX * pow(RL, 2);

      if (student == 1)
            fprintf(fp_out, "\n\n   Deviate Corresponding to Confidence Level
of Interest = %f", TX);

      fprintf(fp_out, "\n\n   Ratio      =    %f", RL);
      fprintf(fp_out, "\n\n   Confidence limits");
      fprintf(fp_out, "\n     %f     %f", (RL - TX * sqrt(RV)), (RL + TX *
sqrt(RV)));

      return;
}

void nrm41 (int IM)
{
      if (NT[IM] == 1) {
            if (XU[IM] > 0)
                  XU[IM] = log(XU[IM]);

            return;
      }

      if (NT[IM] == 2)
            if(XU[IM]>0)
                  XU[IM]=1/XU[IM];

      return;
}


void nrm42 (void)
{
  HQ = 1;
  if (MQ > NY) {
        if (K1[MQ-NY] == MV || K2[MQ-NY] == MV)
              HQ = 2;
  }
}


void nrm43 (void)
```

```
{
      if (NT[MV] == 1) {
            fprintf(fp_out,"%9.3e\n", Exp(XZ));
            return;
      }

      if (NT[MV] == 2) {
            fprintf(fp_out,"%9.3e\n", 1/XZ);
            return;
      }

      if (NT[MV] == 3)
            fprintf(fp_out,"%9.3e\n", XZ);

      return;
}


void nrm44 (void)
{
  fprintf(fp_out, "%8.2e percent\n", P*100);
}


void Chisquare (void)
{
  double XA, CC, CP, CQ, QP, QZ, S, ZL, ZI;
  int I;

  XA = sqrt(CH);

  if(DF % 2 == 0)
      goto CHI1;

  Y = 0;
  if (setjmp(GosubStack[GosubNdx++])==0) goto CHI9;

  if(DF == 1)
      goto CHI2;

  CQ = -log(XA);

  for(I = 1; I <= DF - 2; I += 2) {
      CQ = CQ + 2 * log(XA) - log(I);
      CP = CQ + ZL;
      CC = Exp(CP);
      Y += CC;
  }

CHI2:
  QZ=2*(Z*QP+Y);
  if (setjmp(GosubStack[GosubNdx++])==0) goto CHI8;
#ifdef DEBUG
  fprintf(logfile, "\njumped back to CHI2, QZ=%f QP=%f Y=%f\n\n", QZ, QP, Y);
#endif
  return;

CHI1:
  if (setjmp(GosubStack[GosubNdx++])==0) goto CHI9;
  Y=Z;
  if(DF==2)
```

```
    {
      goto CHI3;
    }
  CQ=0;
  for(I=2; I<=DF-2; I+=2)
    {
      CQ=CQ+2*log(XA)-log(I);
      CP=CQ+ZL;
      CC=Exp(CP);
      Y+=CC;
    }

CHI3:
  QZ = ZI * Y;
  if (setjmp(GosubStack[GosubNdx++])==0) goto CHI8;
#ifdef DEBUG
  fprintf(logfile, "\njumped back to CHI3, QZ=%f ZI=%f Y=%f\n\n", QZ, ZI, Y);
#endif
  return;

CHI8:
#ifdef DEBUG
  fprintf(logfile, "\njumped to CHI8, QZ=%f\n\n", QZ);
#endif
  if(QZ>.05 && chisqr_skip == 0)
      Warn1(QZ);

  if(QZ<.05 && chisqr_skip == 0)
      Warn2(QZ);

  chisqr_skip = 0;
  longjmp(GosubStack[--GosubNdx],1);


CHI9:
  S = 1 / (1 + .2316419 * Abs(XA));
  ZI = sqrt(8 * atan(1));
  ZL = -(pow(XA,2)) / 2 - log(ZI);
  Z = Exp(ZL);
  QP=.31938153*S-.356563782*pow(S,2)+1.781477937*pow(S,3)-
1.821255978*pow(S,4)+1.330274429*pow(S,5);
#ifdef DEBUG
  fprintf(logfile, "\njumped to CHI9, S=%f ZI=%f ZL=%f Z=%f QP=%f", S, ZI, ZL,
Z, QP);
#endif
  longjmp(GosubStack[--GosubNdx],1);
}



void Warn1 (double QZ)
{
  CX = 1;

  fprintf(fp_out, "\n\n   Probability of correct model (p-value) is %f\n",
QZ);
  fprintf(fp_out,"%s\n","   The prediction of the model is sufficient. Use for
estimation of the");
  fprintf(fp_out,"%s\n\n","   95% confidence limits the Standard Normal
Deviate");
  fprintf(fp_out,"%s\n\n","   No correction for variances required!");
```

```
}


void Warn2 (double QZ)
{
  CX=CH/DF;

  fprintf(fp_out, "\n\n   Probability of correct model(p-value) is %f\n", QZ);
  fprintf(fp_out,"%s\n","   The prediction of the model is not sufficient. Use
for estimation of the");
  fprintf(fp_out,"%s% d%s\n\n","   95% confidence limits Student t with
",(int)DF," degrees of freedom");
  fprintf(fp_out,"%s%3.3f\n\n","   Correction for variances Chi-
Squares/Degrees of Freedom = ",CX);
}


void WriDat (void) {
  int I, J, TM, TX;

  NX = NY + NC;

  if( NX == 0) {
      return;
  }

  for(I = 1; I <= NY; I++) {
    WF[I] = K0[I];
  }

  for(I = 1; I <= NC; I++) {
    WF[NY + I] = K1[I];
    WF[NY + NC + I] = K2[I];
  }

  TW = NY + 2 * NC;
  //fprintf(fp_out, "\nInitial TW=%d\n", TW);

  for(I = TW; I >= 2; I--) {
        for(J = 1; J <= I; J++) {
              if(WF[I] < WF[J]) {
            TM = WF[I];
            WF[I] = WF[J];
            WF[J] = TM;
          }
      }
  }

  for (I = 1; I <= TW; I++) {
   #ifdef DEBUG
   fprintf(logfile, "WF[%d]=%d ", I, WF[I]);
   #endif
  }

  #ifdef DEBUG
  fprintf(logfile, "\n\n");
  #endif

  I = 1;

  while(I < TW) {
```

```
    I += 1;

      if(WF[I] == WF[I - 1]) {
        TW -= 1;
            for(J = I; J <= TW; J++) {
            WF[J] = WF[J + 1];
                  #ifdef DEBUG
            fprintf(logfile, "WF[%d]=%d ", J, WF[J]);
            #endif
        }

            #ifdef DEBUG
        fprintf(logfile, "\nTW=%d\n", TW);
        #endif
     }
  }

  if(FlEr == TRUE) {
    return;
  }

time( &ltime );


fprintf(fp_out, "\n");

for(I = 1; I <= TW; I++){
    fprintf(fp_out, "%15s", Vrx[WF[I]]);
}

fprintf(fp_out,"%15s%15s\n\n", Vrx[NZ+1], Vrx[NZ+2]);

// Print out the raw data
TX = 0;

for(I = N1; I <= N2; I++) {

    for(J = 1; J <= TW; J++) {
            fprintf(fp_out,"%15.2f",X[I][WF[J]]);
    }

    fprintf(fp_out,"%14d.",N[I]);
    fprintf(fp_out,"%14d.\n",LR[I]);
    TX += 1;

    if(TX % 5 == 0)
        fprintf(fp_out,"\n");
}
// End of printing the raw data

fprintf(fp_out,"\n   %s% d%s% d\n","Selection of observations from number
",(int)N1," through ",(int)N2);
fprintf(fp_out,"\n   %s\n","Transformation of input parameters");

for(I=1; I<=NZ; I+=1)
  {
    fprintf(fp_out,"   %-15s ", Vrx[I]);
    while(1)
    {
    if(NT[I]==1)
      {
```

```
        fprintf(fp_out,"   %s\n"," is transformed logaritmically!");
        break;
      }
    if(NT[I]==2)
      {
        fprintf(fp_out,"   %s\n"," is transformed reciprocally!");
        break;
      }
    if(NT[I]==3)
      {
        fprintf(fp_out,"   %s\n"," is not transformed at all!");
      }
    break;
    }
}
fprintf(fp_out,"\n");
while(1)
{
if(KPZ==1)
  {
    fprintf(fp_out,"   %s","Probit link used ");
    break;
  }
if(KPZ==2)
  {
    fprintf(fp_out,"   %s","Logit link used ");
  }
break;
}
while(1)
{
if(KBZ==1)
  {
    fprintf(fp_out,"%s\n","without background response correction!");
    break;
  }
if(KBZ==2)
  {
    fprintf(fp_out,"%s\n","with background response correction!");
  }
break;
}
fprintf(fp_out,"\n");
for(I=1; I<=NY; I+=1)
  {
        fprintf(fp_out,"   %s% d%s%s%s\n","Variable ",(int)I,"  =
",((NT[I]==3) ? "" : "Transformed "),Vrx[K0[I]]);
  }
if(NC>0)
  {
    for(I=1; I<=NC; I+=1)
      {
        fprintf(fp_out,"   %s% d%s%s%s%s%s%s\n","Variable ",(int)NY+I,"  =
Product of ",((NT[I]==3) ? "" : "transformed "), Vrx[K1[I]]," and
",((NT[I]==3) ? "" : "transformed "),Vrx[K2[I]]);
      }
  }

fprintf(fp_out,"\n\n   Chi-Square          =  %-6.2f", CH);
fprintf(fp_out,"\n   Degrees of Freedom  =  %-5d\n\n", DF);
```

```
for(I=MX; I<=NXT; I+=1)
  {
    fprintf(fp_out,"   %s%d%s","B",(int)I," = ");
    fprintf(fp_out,"%9.3e\t",BX[I]);
    if(DF>0)
      {
        fprintf(fp_out,"   %s%d%s","Student t  for B",(int)I," = ");
        fprintf(fp_out,"%2.2f\n",BX[I]/sqrt(BV[I][I]*CX));
      }
    else
      {
        fprintf(fp_out,"\n");
      }
  }
if(DF==0)
  {
    return;
  }
fprintf(fp_out,"\n");
for(I=MX; I<=NXT; I+=1)
  {
    for(J=I; J<=NXT; J+=1)
      {
        while(1)
        {
        if(I==J)
          {
            fprintf(fp_out,"   %s%d%d%s","  variance  B",(int)I,(int)J," = ");
            break;
          }
        if(I!=J)
          {
            fprintf(fp_out,"   %s%d%d%s","covariance  B",(int)I,(int)J," = ");
          }
        break;
        }
      fprintf(fp_out,"%9.3e\n",BV[I][J]*CX);
    }
}

return;

FAULT1:;
CalcErr();
}


void CalcErr (void)
{
  time( &ltime );
  static char Msg[2048];
  memset(&Msg,0,sizeof(Msg));
  NY=0;
  NC=0;
  NX=0;
  if(FP2)
   {
     fflush(FP2);
     fclose(FP2);
   }
  FlEr=TRUE;
```

```
  sprintf(BCX_STR,"%s","The combination of data\n");
  strcpy(Msg,BCX_STR);
  sprintf(Msg,"%s%s",Msg,"and design for mathematical\n");
  sprintf(Msg,"%s%s",Msg,"analysis produced an error\n");
  sprintf(Msg,"%s%s",Msg,"in the calculating procedures!\n");
  sprintf(Msg,"%s%s",Msg,"Please, revise your design\n");
  sprintf(Msg,"%s%s",Msg,"for mathematical analysis!");
  MessageBox (GetActiveWindow(),Msg,"",0);

  if((FP2=fopen("DoseResp.Log","a"))==0){
      fprintf(stderr,"Can't open file %s\n","DoseResp.Log");
      exit(1);
  }

  fprintf(FP2,"\n");
  fprintf(FP2,"%s%s\n","Filename = ",Fina);
  fprintf(FP2,"%s\n",ctime(&ltime));
  fprintf(FP2,"%s\n","An error occurred in the calculation!");
  fprintf(FP2,"\n");
  if(FP2)
   {
     fflush(FP2);
     fclose(FP2);
   }
}

// ***********************************************************
//    READ_OBSDATA--used to read data in a matrix(row, col).
// ***********************************************************

int READ_OBSDATA (int r, int c, double **matrix)
{
  int Nmiss;                  /*number of records with missing values */
  int i, j, n, m;       /*count and iteration control variables */
  double dvalue;              /*temp variable */
  int ivalue;

  Nmiss = 0;
  for (i = 1; i <= r; i++) {
      n = i - Nmiss;
      m = 0;

      for (j = 1; j <= c + 2; j++) {
              if (j <= c)
                    fscanf(fp_in, "%lf", &dvalue);
              else
                    fscanf(fp_in, "%d", &ivalue);

            if ((j <=c && dvalue != MISSING) || (j > c && ivalue != MISSING))
{
                  if(j == NZ + 1) {
                        N[n] = ivalue;
                  #ifdef DEBUG
                        fprintf(logfile, "\nN[%d] = %d", n, N[n]);
            #endif
                  }

                  if(j == NZ + 2) {
                        LR[n] = ivalue;
                  #ifdef DEBUG
                        fprintf(logfile, "\nLR[%d] = %d", n, LR[n]);
```

```
            #endif
                  }

                  if (j <= c) {
                        matrix[n][j] = dvalue;
                  #ifdef DEBUG
                        fprintf(logfile, "\nData matrix[%d,%d] = %lf", n, j,
matrix[n][j]);
            #endif
                  }
               }
            else
                  m++;
         }

      if (m != 0)
            Nmiss++;
  }

  return Nmiss;
}


void Get_File_Stem(char *argv, char *filestem)
{
      int i = 0;

      for (; i <= 127; i++) { /* TODO: change to dynamically calculate the
length of argv */
            filestem[i] = argv[i];

            if (argv[i] == '.')
                  break;
      }

      infilestem[i] = 0; /* ends the file stem properly*/
}

void Derive_File_Name(char *stem, char *newfile, const char *ext)
{
      int len = strlen(stem);
      strcpy(newfile, stem);
      newfile[len] = '.';
      strncat(newfile, ext, strlen(ext));
      newfile[len + strlen(ext) + 1] = 0; /* to end the new file name properly
*/
}
```