CR-XXXX

Wouldn't it be nice to use all of your data points: An introduction to NICE

Steven Read

14 Feb 2023

ICEAA Workshop Paper and Presentation May 2023
San Antonio, Texas

TECOLOTE RESEARCH, INC.
420 S. Fairview Ave., Suite 201
Goleta, CA 93117-3626
(805) 571-6366

TECOLOTE RESEARCH, INC.
5383 Hollister Ave., Suite 100
Santa Barbara, CA 93111-2304
Phone

TABLE OF CONTENTS

SECTION **PAGE**

TABLE OF FIGURES

TABLE OF TABLES

ABSTRACT

A key challenge for data collection directives is predicting which data elements are both easy to collect and valuable to future analysts. In practice, real-world data collection is messy and missing data values that create non-homogeneous datasets are commonplace. Data imputation fill in missing data values when fitting estimating relationships, including Cost Estimating Relationships (CERs). Non-linear Iterative Constrained Estimator (NICE) is an approach to both data imputation and CER fitting that considers the whole model and uses the Work Breakdown Structure (WBS) relationships to recalibrate systems of equations for a better fit. This paper explains what NICE is, how it works, and begins to explore what problems this method can solve.

WHAT IS THE PROBLEM?

The purpose of fitting a CER is to provide a means of estimating the cost of a system using at least one other variable. An example is estimating the cost of a vehicle system using vehicle power. For this paper, we use a CER of the form:

$$\text{Vehicle\$} = a * \text{Payload\$} + b * \text{Power}$$

Where Vehicle$ is the cost of the new vehicle, Payload$ is used as a surrogate for payload size and ride requirements, and Power is the power of the vehicle. With this goal in mind, the first step is forming the WBS outline that is the basis of our model. This WBS must include the detailed levels that we want to estimate, namely Vehicle$, and include all of the summation relationships to cover the overall cost of the whole system. The model outline should also include any technical descriptors that are being estimated as part of the CER analysis. **Table 1** details the WBS for this example.

Table 1 – Example WBS

WBS
Ownership$
Acquisition$
Vehicle$
Payload$
Operation$
Power
Endurance

The overall goal of this WBS is to build a model sensitive to the included technical descriptors, (Power and Endurance) produce results at the level of detail specified, (Vehicle$) and define a summary structure to account for the overall system cost.

With the WBS defined, the next step is to collect data from analogous systems and map those values into the chosen WBS. Extra care should be taken in this step to ensure that the analogous data points collected truly are comparable to the new vehicle and that the values collected for each WBS element align with the definition for that element. In this example, vehicle costs that

also include operating costs should not be included for Vehicle$ because the intention of that WBS element is to only include costs associated to the vehicle itself and not the operation of that vehicle.

Unfortunately, missing data often presents a challenge. This fabricated example includes seven analogous vehicle systems, as seen in **Table 2** below. However, only five of the systems contain a complete set of desired information. Two vehicle systems are missing key information regarding vehicle costs – the main element of the CER. The missing data presents a difficult situation without a clear path forward, but with several options to consider:

1) *Keep the CER at the lowest level elements and continue estimating from the bottom up in the WBS.* This requires that the two vehicles missing costs for Vehicle$ be excluded from the analysis. There are two main drawbacks to this approach. First, if the data points with missing values represent parts of the underlying vehicle population not represented by the remaining vehicles in the dataset, removing these two vehicles can introduce a bias in the resulting CER. Second, removing these data points reduces the overall degrees of freedom available to fit the CER.
2) *Change to a top down estimating method.* This requires modifying the CER to switch to a higher level WBS element which does contain values for all systems. In this case the CER would shift from estimating Vehicle$ to Ownership$. This reduces the fidelity of the resulting CER since the new relationship describes how the overall cost of ownership is influenced by the Power requirement and the Payload cost. Additional work is also required to determine an allocation scheme to give values to the lower level WBS elements based on the estimated top-level value.

Table 2 – Example Collected Data

WBS	VEHICLE219	VEHICLE220	VEHICLE223	VEHICLE224	VEHICLE225	VEHICLE227	VEHICLE233
Ownership$	16,281	1,176	4,614	711	520	4,768	11,600
Acquisition$	15,643	1,040		661	460		5,800
Vehicle$	11,543	820		439	356		3,600
Payload$	4,100	220	181	222	104	844	2,200
Operation$	638	136		50	60		5,800
Power	13,001	1,200	940	1,501	900	5,200	15,000
Endurance	121	36	72	25	96	48	240

3) *Use data imputation.* This is a process where the blank values are filled in systematically to create a homogeneous dataset with values for every element in the table. A recent presentation, "Dealing with Missing Data – The Art and Science of Imputation" by Roye, Hilton, and Smart, introduced the ICEAA audience to several different data imputation methods. The presentation concluded that the two strongest methods were Multiple Imputation by Chained Equations (MICE) and Expectation Maximization (EM).

NICE is a framework for data imputation that includes two separate methods used to fill in missing data point values: a single pass method and an iterative method.

The strength of the NICE approach is that all 7 data points can be used to fit the CER – even if their data is incomplete. An additional benefit of NICE is that it maintains the additive relationships of the existing WBS, something that is not accounted for in either the MICE or EM imputation methods.

Real World Situations

In the above example, the data is consistently incomplete, so the same strategy works for both Vehicle 223 and Vehicle 227. In the real world, data is often inconsistently incomplete. **Figure 1** demonstrates several different ways that data can be incomplete and provide even more of a hurdle to performing a CER analysis.

Figure 1 – Real World Incompleteness

WBS	Vehicle219	Vehicle220	Vehicle223	Vehicle224	Vehicle225	Vehicle227	Vehicle233
Ownership$	16,281	1,176	4,614	711	520		11,600
Acquisition$	15,643	1,040		661	460		5,800
Vehicle$	11,543	820		439	356		3,600
Payload$	4,100	220	181	222	104	844	2,200
Operations$	638	136		50	60		5,800
Power	13,001	1,200	940	1,501	900	5,200	15,000
Endurance	121	36	72	25	96	48	240

WBS	Vehicle219	Vehicle220	Vehicle223	Vehicle224	Vehicle225	Vehicle227	Vehicle233
Ownership$	16,281	1,176	4,614	711	520		11,600
Acquisition$	15,643	1,040		661	460		5,800
Vehicle$	11,543			439	356		3,600
Payload$	4,100		181	222	104	844	2,200
Operations$	638			50	60		5,800
Power	13,001	1,200	940	1,501		5,200	15,000
Endurance	121	36	72	25	96	48	240

1. Missing total system costs
2. Only summary data available
3. Missing technical descriptor

In Real-World Situations, Many Observations Are Inconsistently Incomplete

Background

Analysts at Tecolote began exploring the research and development of NICE in the 1980s (see References [2], [5], and [10]). NICE was developed for the computer systems of that era (see [4] and [6]). Tecolote analysts utilized NICE for a number of cost estimates and CER analyses covering aerospace, electronics, and software domains (see [3], [8], [9], [11], [12], and [13]). These original software efforts were programmed in Fortran, which required programmer staff to support the projects.

In 1990 Tecolote performed a study on how to migrate NICE to a PC environment in order to be more accessible to analysts (see [14]). The resulting suggestions included adding a graphical user interface to allow users to edit inputs to NICE, updates to the CER inputs that did not require a programmer to implement them, and improvements to some of the internal heuristics. The recommendations from the study were not implemented at the time.

However, Tecolote recently revisited NICE and implemented some of the previous recommendations. This project produced a prototype R Shiny application of NICE which takes Excel based inputs and can calculate in both the single pass and iterative modes using a Newton's-method-based non-linear minimization algorithm.

PURPOSE

This paper presents an introduction to NICE and highlights several key discussion topics. The main goal of this paper is to introduce the NICE algorithm and outline the information required to apply NICE to a problem set. The secondary goal of this paper is identifying additional problem sets to validate the process and ensure that the method is accessible to analysts.

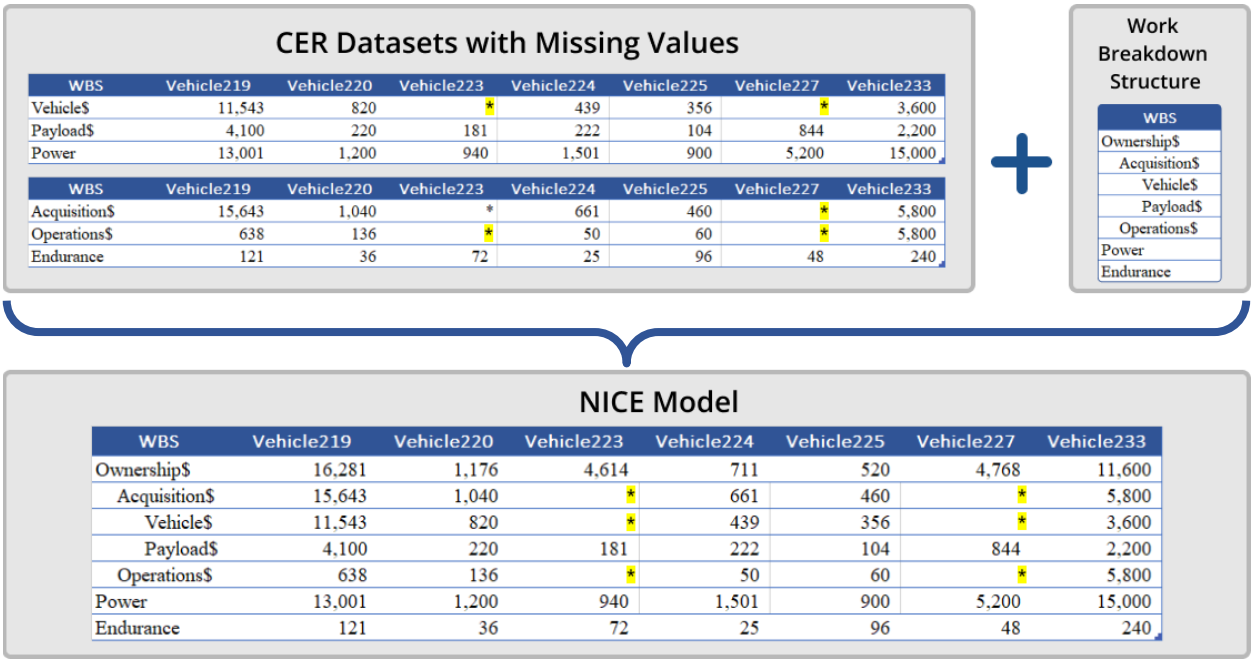
Tecolote seeks to collaborate with analysts who have experience with CER analysis or who work with data that fits into the NICE problem domain. The goal for this collaboration would be to run experiments and do testing on the applicability of NICE along with any potential tuning to improve performance on real datasets. With the resulting NICE imputed data and CER coefficients we can then evaluate the overall fit and potentially compare to existing methods of data imputation.

WHAT IS NICE?

The NICE approach incorporates data outside of a traditional CER analysis and allows an analyst to study multiple CERs simultaneously. In traditional CER analysis, the data is selected and filtered to a clean subset and hypotheses are tested one at a time. With NICE we attempt to unlock the potential value in missing data.

NICE incorporates additional WBS data beyond the traditional scope of CER inputs and allows for the inclusion of a WBS hierarchy within the cost data values through the use of constraint matrices. Many CER studies analyze specific lower level cost elements of an overall WBS. When a data point is missing the cost value for that element, filtering leaves the blank value as "unknown". Including a full WBS as part of the NICE inputs can change that "unknown" to "sums with one sibling to a specific parent value". In this way the imputed data can be verified to sum to valid higher level WBS totals. **Figure 2** demonstrates how different CER datasets can combine with a WBS structure to build the NICE model inputs.

Figure 2 – Building a NICE Model



With the inclusion of multiple lower level cost elements in the NICE inputs comes the ability to estimate multiple CERs simultaneously for those elements. Further, NICE is also capable of fitting any linear or non-linear format of CER, with no requirement that a CER match the format of any other CER.

DEFINING THE INPUTS

NICE works with two main sets of inputs, model inputs and equation inputs. The following sections build on the initial dataset to help better understand these input types.

MODEL INPUTS

Model inputs are the initial values for the cost elements and other variables, along with the constraint matrices. Constraint matrices are used to define hard constraints for specific values as well as the additive relationships between different model elements which define how the WBS structure sums.

Table 3 provides a sample set of initial values for the dataset from **Table 2**. The initial data inputs to NICE will follow the format of the original table with any missing values replaced with numeric values. These can be potentially invalid values which do not conform to the additive constraints of the WBS. NICE will then calculate a set of valid initial values which do conform to the additive constraints of the WBS before starting the imputation process. On this table, each row is labeled in NICE as x_1 through x_7. Each of the columns after the WBS column are labeled d_1 through d_7.

Table 3 – NICE Initial Data Example

LABEL	WBS	VEHICLE219 D_1	VEHICLE220 D_2	VEHICLE223 D_3	VEHICLE224 D_4	VEHICLE225 D_5	VEHICLE227 D_6	VEHICLE233 D_7
x_1	Ownership$	16,281	1,176	4,614	711	520	4,768	11,600
x_2	Acquisition$	15,643	1,040	0	661	460	0	5,800
x_3	Vehicle$	11,543	820	0	439	356	0	3,600
x_4	Payload$	4,100	220	181	222	104	844	2,200
x_5	Operation$	638	136	0	50	60	0	5,800
x_6	Power	13,001	1,200	940	1,501	900	5,200	15,000
x_7	Endurance	121	36	72	25	96	48	240

For each of the data points d_i there is a corresponding constraint matrix, C_i, as demonstrated in **Table 4** – NICE Constraint Matrix Example for Vehicle 223 (d_3). Each row of the constraint matrix defines a hard constraint on the NICE algorithm for how to modify the imputed values. There are two main kinds of constraints defined in a NICE constraint matrix.

The first type of constraint is a value override constraint. For each x_i that has a defined value there is a row in the constraint matrix that has a one in the x_i column, zero in the other columns, and the value override in the b column. So, the first row in **Table 4** defines the equation:

$$x_1 = 4614$$

This value matches the values row 1 in Table 3 and tells NICE that this variable can only ever take one value. In **Table 4**, you can see that fourth, fifth, and sixth rows also define value override constraints.

The second type of constraint is an additivity constraint. For x_i rows that participate in the WBS hierarchy, the constraint matrix can define the additive relationships between different x_i values based on how the WBS sums. This can be seen on the second row in Table 4, that row defines the additive relationship between x_1, x_2, and x_5, which can be written as:

$$-x_2 + x_3 + x_5 = 0$$

Which we can substitute the WBS names to write as:

$$\text{Acquisition\$} + \text{Operation\$} - \text{Ownership\$} = 0$$

Which can be rewritten in a more familiar form as:

$$\text{Ownership\$} = \text{Acquisition\$} + \text{Operation\$}$$

This row is thus telling NICE that the two level 2 WBS elements sum to equal the level 1 WBS element. The third row in Table 4 also defines an additive constraint on the two level 3 sibling rows which add up to the level 2 parent total.

Table 4 – NICE Constraint Matrix Example for Vehicle 223 (d_3)

Constraint Type	X_1	X_2	X_3	X_4	X_5	X_6	X_7	B
Value	1	0	0	0	0	0	0	4614
Additive	-1	1	0	0	1	0	0	0
Additive	0	-1	1	1	0	0	0	0
Value	0	0	0	1	0	0	0	181
Value	0	0	0	0	0	1	0	940
Value	0	0	0	0	0	0	1	72

Although the WBS column in **Table 3** is formatted with indentation for readability, NICE does not actually do anything with the indentation in that column. The only way to define the additive relationships between model elements is through rows in the constraint matrix. There is no restriction on the number of rows in a constraint matrix, and there is no requirement that the number of rows should match between the rows in the model table and the rows in the constraint matrix. Specific x_i values can participate in multiple rows to both define value override constraints as well as additive constraints. Each of the constraint matrices are labeled C_1 through C_7 where C_1 is the constraint matrix for data point d_1 and so on. **Table 4** is thus showing C_3.

Equation Inputs

Equation inputs are the list of equations relating different model elements to each other and the initial coefficient values for those equations. Along with the model data, which defines hard constraints on the NICE algorithm, there is a set of equation data which defines relationships between model elements that NICE then uses to calculate the objective function used for non-linear minimization. Going with the current example would be the following two equations:

1. Vehicle$ = $coef_1$ * Payload$ + $coef_2$ * Power
2. Operation$ = $coef_3$ * Acquisition$ + $coef_4$ * Endurance

Here each of the equations are labeled e_1 and e_2, with coefficients labeled $coef_1$ through $coef_4$. The number of equations and the number of coefficients are unlimited in NICE and do not depend on the number of data points or the number of model elements. The second piece of the equation inputs are the initial coefficient values as show in Table 5. The coefficient vector contains a single column which lists out each $coef_i$ on its own row.

Table 5 – NICE Coefficient Example

Coefficients
0.9
0.3
0.2
4

These coefficient values can be the result of running a standard regression on the existing data points, or they can come from previous CER studies on analogous systems.

DEFINING THE PROCESS

NICE is a framework for data imputation that includes two operating modes, an Estimation mode and an Iteration mode. Both modes will impute the missing data values.

ESTIMATION MODE

In Estimation mode NICE steps through each data point, using the constraint matrix for that data point, and converts the potentially invalid starting values to values which fit within the constraints. Using this valid initial point, NICE finds the basis vectors for the null space of the constraint matrix and converts the constrained search problem into an unconstrained search problem. Then, using the input equations and initial coefficients, NICE runs a Newton's-method-style non-linear minimization method to find the best imputed values to fit those equation inputs.

Let's look back at our small example. Since data points d_1, d_2, d_4, d_5, and d_7 are all fully defined, those data points are skipped in Estimation mode. This means that only data points d_3 and d_6 are changed in Estimation mode. Further, for these data points, both will have the same structure for the constraint matrix, with only the B column values changing for each data point. Since the unconstrained minimization method uses a non-linear search algorithm, this will cause the NICE results for this mode to be sensitive to the initial data. In some cases, changes to the initial values can have big changes in the results. In other cases, large changes in the initial data produce no change in the final result. This sensitivity to initial data is a good reminder to explore the space of initial values and check how well each potential result fares in terms of fitting with the equation data.

ITERATION MODE

Iteration mode has multiple steps. First it performs an Estimation mode step, keeping the equation inputs fixed and then imputing values for the model data to form a homogeneous data set. The next step runs the same non-linear minimization process but keeps the new model data fixed while modifying the coefficient values to find the best fitting coefficients for those model values. These two steps combine to form one iteration. At the end of each iteration, the current iteration objective function values are compared to the previous iteration objective function values to check for convergence. Iteration mode is complete when both sets of data have converged.

For the small example, Iteration mode starts with the Estimation mode as the first step in a single iteration. From the example model data, only d_3 and d_6 are modified as part of the estimation step. The result of this step is a homogeneous set of data where all of the values have been filled in. The next step takes the resulting model data for each d_1 through d_7 as being fixed. Then it steps through each equation, e_1 and e_2, and perform a Newton's type of minimization on the objective function for that equation by varying the coefficient values being used. This step utilizes the values for each d_1 through d_7 even though some data points don't change during the estimation step. These two steps combine to form one iteration, and at the end of each iteration, the results for the objective functions are evaluated against the results from the previous iteration. Once those values have converged within a certain threshold, the iteration process stops and the final set of model data and equation coefficients are the final results for Iteration mode.

FIGURE 3 – NICE Iteration Mode Flowchart

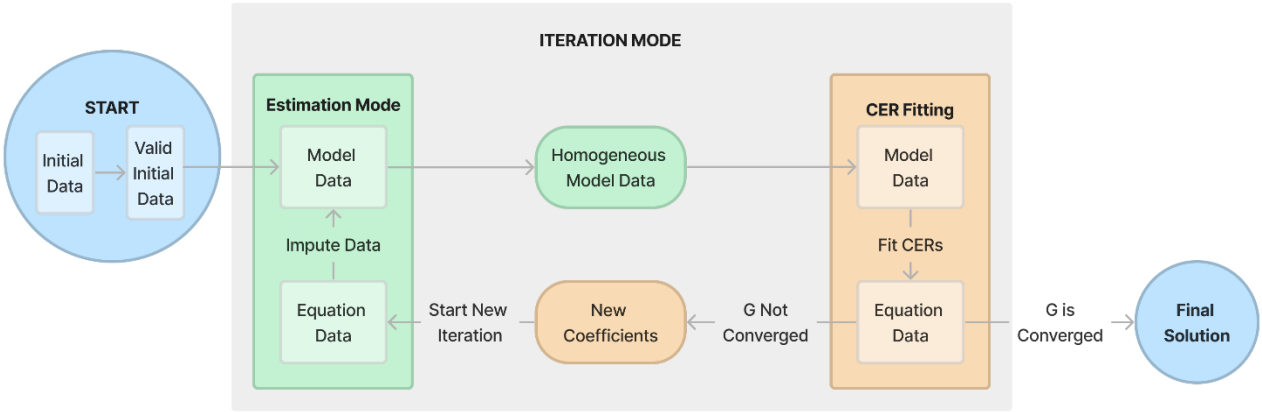


FIGURE 3 shows how the NICE algorithm proceeds in Iteration mode. Here G is the value of the overall objective function from fitting the whole system of equations.

In this mode, as with the previous one, the final result is sensitive to the initial inputs for both the model data and the equation coefficients. For this reason, it is important to explore the space of initial values for both kinds of inputs and evaluate the results produced by those inputs. Another situation that can arise during the iteration process are values that bounce between two specific results so that they never converge to a solution. In this case it is again important to test different initial values for model data and equation coefficients to try and find initial data that does converge to a final solution.

OPPORTUNITIES AND LIMITATIONS

OPPORTUNITIES

NICE provides several opportunities for cost researchers.

1. PRESERVING DATA

The biggest opportunity for any data imputation method is the ability to utilize all of the existing data available in an analysis. The challenge of small datasets is a real hurdle in many different areas. NICE takes this a step further and allows for the incorporation of more information about the whole system within which a CER is functioning to be included as part of the analysis process. This can move the missing data from being 'unknown' to fitting within an existing WBS structure and utilizing the additivity constraints on the WBS to guide the search for imputed values. Additionally, including CERs for multiple WBS elements allows for the incorporation of even more knowledge about how the system described by the WBS works and hopefully improves the overall results for fitting all of the CERs.

2. TESTING NON-LINEAR HYPOTHESES

NICE allows for defining equations that fit any linear or non-linear form. Other data imputation methods may make assumptions about fitting only linear CERs. NICE does not make this assumption and allows for multiple equations of different forms to be defined as part of the analysis.

3. INCORPORATING ADVANCED METHODOLOGIES

NICE provides a lot of flexibility in the available framework. Previous sections describe the use of Newton's type of minimization method to find both imputed model data as well as to fit equation coefficients. There is no proscription within the NICE framework against using other methods though. There are other available options, such as the downhill simplex method. There is also flexibility available in terms of the objective function. Many options exist beyond the sum of squared errors that is most familiar in CER analysis. Further, given that NICE can include any number of equations to fit, extra functions may be included to penalize certain model values from exceeding certain maximum or minimum bounds. This option would allow for penalizing negative cost values. These equations do not provide hard constraints in the same way that values are defined in the constraint matrix, but they do push the search for model data in the direction provided.

4. UTILIZING NEW TECHNOLOGIES

Recent advancements in both mathematical and computational fields make this a good time to revisit the initial NICE research. We now have the benefit of new algorithms and tools. For example, the R programming language makes available all of the matrix manipulation functions used in NICE along with implementations of several different non-linear minimization methods. This really lowers the barrier to entry for utilizing NICE in a CER analysis because much of the heavy lifting is covered by existing tools. Advances in computer hardware during the last 30 years also mean that utilizing NICE even in Iteration mode does not tax current level hardware appreciably.

LIMITATIONS

Nice does have its limits. When considering potential datasets, it helps to consider these limitations.

1. USING IMPUTED DATA IN EACH ITERATION

One of the main limitations of NICE as a method for CER fitting is that estimated model values are utilized within each iteration to do equation fitting. Given that this process is sensitive to starting values in both steps of an iteration, this can lead to a chaotic search path through the problem space before NICE lands on a final value for the solution.

2. NON-LINEAR MINIMIZATION PITFALLS

NICE is limited when utilizing non-linear minimization methods. This affects both operating modes and is related to how most non-linear minimization methods exhibit the butterfly effect,

where small changes to input values can have large changes on the final solution. In addition to input sensitivity, non-linear minimization methods have the possibility to never converge to a final solution. These limitations can be overcome by taking careful consideration during an analysis to explore the available problem space thoroughly.

3. MODEL AND EQUATION COUPLING

Another limitation is the overall binding between the model and equation data that creates a tight coupling between the inputs and the outputs. Adding a new data point requires both a new set of model data for the new data point, along with re-running the entire NICE process over again with these new inputs. This requirement along with input sensitivity could lead to differences in the final results that may not be found with a more traditional CER fitting process.

4. INTER-EQUATION COUPLING

Further coupling exists amongst the different equations that are defined as inputs to NICE. Since NICE is fitting all of the equations simultaneously, this limits what can be said about a single equation in isolation of the others. It may be possible that NICE sacrificed some of the objective function value in one equation to gain a larger improvement with another equation instead. Thus, discussion of NICE results should take care to address how the overall system of equations fared as well as how specific equations within the system of equations fared.

5. UNCONSTRAINED MINIMIZATION PITFALLS

Portions of NICE use unconstrained minimization methods which can result in non-sensical negative values for costs. As a heuristic method, it is important for an analyst to take care with situations where the results may not make sense. For the situation of NICE finding negative costs, there are workarounds for this limitation that can be implemented by defining penalizing equations to limit the presence of negative values. It may also be worthwhile to explore the ability of current minimization code to incorporate non-negative constraints or allow for bounds definition on input values.

TOPICS FOR FURTHER STUDY

With the mathematics defined and code developed, the next phased of this project focuses on application. The following present potential areas for further study.

MULTIVARIATE CAIV

Multivariate Cost As Input Variable (CAIV) analysis starts from a model with cost WBS elements and input variables very similar to the model inputs used with NICE. These elements can also be related to each other with equations that define the cost of some WBS elements in relation to the values of some of the input variables or even between two different cost elements. This again fits with the equation inputs used with NICE. A multivariate CAIV analysis then modifies the values of one or more input variables to recalculate one or more cost elements in the WBS with a goal of reaching a desired cost total at a specific WBS level. The input values may also have a desired overall input total as a constraint as well. In this particular case then, the NICE Estimation Mode applies almost directly to this problem. You could imagine that there is really only one NICE data point as the estimating model from CAIV and the equation inputs are the parametric relationships between the input variables and the cost WBS elements.

This is somewhere where collaboration would be very helpful. Multivariate CAIV could apply to a myriad of situations, and having useful concrete examples would really help in working out how useful NICE can be for such problems. One such example would be a model which incorporates a complex buy quantity structure involving several different variant versions of a ground vehicle where there are relationships between the different variants, as well as overall cost targets for the total purchase price. This situation can be too complex to handle with a single input variable driving the cost and so using NICE opens up an opportunity to perform this kind of analysis where it may not have been possible before.

FURTHER NICE STUDY

Tecolote's early research (see [14]) suggested improvements to include a graphical user interface used to edit inputs, along with suggestions about how to improve certain heuristic element within the NICE process. The new NICE research has already produced a prototype version of NICE in the R programming language which uses inputs defined in Excel. R was selected because it is capable of doing much of the heavy lifting mathematically, like doing matrix manipulations and non-linear Newtons' method minimization. Further updates to improve upon the existing NICE heuristic process involve incorporating bounds on model data values or equation coefficients, incorporating automatic initial data exploration methods like simulated annealing, and inclusion of more options for the non-linear minimization method. Each of these issues is a worthy element to study and potentially opens up more opportunities to utilize NICE, or improve the overall results of NICE.

Along with opportunities to grow the NICE algorithm, there are opportunities to explore the potential limitations encountered back in 1990 that may have caused this long pause in the development of NICE and whether they are still applicable today. In the intervening years, it has become much more commonplace for analysts to be acquainted with using a programming language like R. There are no longer any requirements to code the objective function in Fortran

that might have been a hurdle before. Also, there are a many more options available for non-linear minimization problems. Being limited to just a Newton's method search may fail in specific cases or it may be that too much trial and error was required to find initial data points that provided a good equation fit. These kinds of hurdles may have been just enough to slow down the pace of NICE development to sidetrack the overall progress. Exploring the opportunities for growth may go hand in hand with addressing previous limitations on the algorithm.

SUGGESTIONS

Tecolote is seeking suggestions on other uses for NICE and how to apply this algorithm to other problem domains or types of analysis. This paper is the first step in what is hopefully a much larger journey of the development of NICE.

REFERENCES

1. Roye, K., Hilton, D., Smart, C., "Dealing with Missing Data – The Art and Science of Imputation," ICEAA 2021 Online Workshop, May 2021
2. Dwyer, T., "The NICE Model," Tecolote Publication Library, June 1980
3. Dwyer, T., Frederic, B., Nordsieck, R., "Oracle Data Book," Tecolote Publication Library, Aug 1980
4. Dwyer, T., "NICE Guide to Software Usage (For NWC UNICAV 1110)," Tecolote Publication Library, March 1981
5. Dwyer, T., "Applications of NICE," Tecolote Publication Library, March 1981
6. Dwyer, T., "Programmer's Guide to NICE for HP-1000," Tecolote Publication Library, July 1981
7. Vokolek, A., "NICE – A User's Perspective," Tecolote Publication Library, February 1984
8. Dwyer, T., Horak, J., "RDTEII Tactical Missile RDT&E Cost Model," September 1981
9. Vokolek, A., "Missile Guidance Cost Estimating Models," Tecolote Publication Library, September 1984
10. Dumas, E., Hu, S., "Mathematics of NICE," Tecolote Publication Library, September 1984
11. Frederic, B., Vokolek A., "Updated Cost Estimate Relationship for ABM Radars," Tecolote Publication Library, June 1985
12. Horak, J., Hu, S., Vokolek, A., "RDTEIII Tactical Missile RDT&E Cost Model," Tecolote Publication Library, February 1986
13. Caindec, E., Hu, S., Jaffurs, C., Sullivan, K., "The Black Box Estimators (BBEST) Electronics Cost Models," Tecolote Publication Library, November 1987
14. Hu, S., Jaffurs, C., McGahan, J., Nordesieck, R., Sturgeon, C., "A Review of Options for Development of a PC-based Version of the NICE Program," Tecolote Publication Library, October 1990