NISTIR 7023

Missing Data Methods and Toolbox Users Guide

Curtis Parks
Gerard N. Stenbakken
Alexei Nikolaev
U. S. DEPARTMENT OF COMMERCE
Technology Administration
National Institute of Standards
and Technology
Electricity Division
Convergent Information Systems Division
Gaithersburg, MD 20899

NIST
National Institute of Standards and Technology
Technology Administration
U.S. Department of Commerce

NISTIR 7023

Missing Data Methods and Toolbox Users Guide

Curtis Parks
Gerard N. Stenbakken
Alexei Nikolaev
U. S. DEPARTMENT OF COMMERCE
Technology Administration
National Institute of Standards
and Technology
Electricity Division
Convergent Information Systems Division
Gaithersburg, MD 20899

July 2, 2003

Missing Data Methods and Toolbox Users Guide

Curtis Parks, and
Gerard N. Stenbakken
Electricity Division

and

Alexei Nikolaev
Convergent Information Systems Division

National Institute of Standards and Technology
Gaithersburg, MD 20899

Abstract

Measurement data are collected to characterize a production run of devices that can then be analyzed to identify a reduced set of measurement points. Such measurement arrays can contain missing values. Missing data analysis methods are described, together with a graphical toolbox for executing the methods. This Users Guide describes the mathematical techniques and the utility of the toolbox in identifying an optimal approach in using the incomplete information in processes which expect complete information. The resultant measurements are shown to give an improved model of the device.

Keywords: data analysis, device models, missing data

1. Introduction

Devices may be tested at a number of test points. For many analog instruments a full calibration often requires measurements at a large number of test points. This testing may be combined with a variety of test conditions. For example, an electrical instrument may be tested at various ranges, voltages, and frequencies. To be able to reduce the number of measurements that characterize the instrument, test data arrays are used to build models, termed empirical models[1], of instrument performance. The HELP (High-Dimensional Empirical Linear Prediction) Toolbox[2] was developed for the analysis of the models and to identify reduced sets of test points for characterizing the instrument or device. Measurement errors result in imperfect models. Thus, the model accuracy improves when more data is used in the model.

The information used to build models of devices may have missing information. This may be due to spurious data being discarded, information not taken, or the number of test points expanded. Normally the partial data is discarded from the data used to build the model, leaving only full and reliable measurement sets. However, this practice cannot take advantage of the partial data that was in the model. The question of how to use the incomplete information to build a model of a device type and the use of such a model built from this data is examined in this study.

Several methods for using the incomplete data set are proposed and predicted results from the best method are shown in simulations. Methods to predict the uncertainty of these methods are being developed.

Following the Introduction is a discussion on methods for estimating missing data that also identifies the variables used in the remainder of the paper. Next is a section describing the Missing Data Toolbox as a guide to its use. Included in the Toolbox are array preparation aids and automated plotting of the errors returned from the execution of a selected method. Then follows a section on the underlying model and the mathematical basis for each of the methods. The results from an example set of measurements is included, showing the differences in errors obtained from each of the methods.

2. Methods for Estimating Missing Data

In general, the missing data may be at any location. In fact, it may be possible to develop models for the case where information on all test points of interest is not available for any of the devices. However, to simplify the problem, the methods considered in this paper look at the problem when the missing data is a block of test points missing for a subset of the devices. Thus, consider that the information for a full characterization of a device consists of m test points and that this full characterization is available for $p1$ devices. The full set of information is an m by $p1$ matrix called b. Also available is data from $p2$ devices, but for each of these devices the same $m1$ data points are missing. Without loss of generality we consider that the missing data are the first $m1$ test points for each device. Let c designate the full set of information for these devices, including the missing data. Let ct be the submatrix of c that represents the $m1$ by $p2$ set of missing data, and cb be the $m\text{-}m1$ by $p2$ set of information that is available for these $p2$ devices. Let bt be the submatrix of b corresponding to the $m1$ test points of data missing from the c matrix and bb be the submatrix corresponding to cb.

3. The Missing Data Toolbox

A graphical user interface (GUI) has been developed and code written to function as a toolbox based on methods 1 through 6 described in Section 4. The GUI provides a means of exercising the methods for generating models using partial data in large collections of measurements. The interface allows the data set to be separated into modeling and validation sets, and models of various sizes to be built from the modeling set. The models are evaluated by calculating the prediction errors when they are used on the validation set. The interface allows the errors to be displayed as a graph comparing the error as a function of selected model sizes. This toolbox is expected to augment the High-Dimensional Empirical Linear Prediction (HELP) toolbox[2].

Initially, a Web page was developed to document and review the toolbox requirements and variables. Some of the variables used in the methods, however, are different than those in the HELP Toolbox. Thus the Web page included as Attachment 1 may prove to be useful in a future alignment of the variables with those found in the HELP toolbox.

Based on an initial concept sketch, the GUI shown as Attachment 2 was developed. As the Toolbox may aid in the preparation of measurements for later use with the HELP Toolbox, the Missing Data Toolbox GUI was patterned to be familiar to users of the HELP Toolbox.

3.1 General Description

The Missing Data Toolbox has been constructed to provide user control of the methods developed and described in Section 4.4. The Toolbox was developed under MATLAB®[†] version 6.1, and has been shown to run under 6.0. Each method is a separate MATLAB execution routine, and all may be executed independently of the Toolbox. This modularity allows the methods to be revised, or for the addition of new methods with minimal changes to the Missing Data Toolbox. This section describes the Missing Data Toolbox that evolved from the initial concept GUI and describes the sequence of steps for using the Toolbox.

Two files (see Figure 1), each containing arrays of measurement data, are used as inputs by the toolbox. The array containing missing data points (or missing rows) is renamed "c." The related array file (with no missing data) is renamed "raw." The pre-processing is the detection of missing data in the partial data array and sorting the array so the missing data rows occupy the highest-numbered rows of that array. A Toolbox menu item may be optionally used to pre-process the array containing the missing data. The file "raw" will be split into columns for a validation array called *vm* and columns for a training array called *b*.

All loading of data and method execution is controlled by the Missing Data GUI menu selections. These menu items control (in order, left to right) the Data Sets, the Initialization, and the Execution. The menu items are ordered in the sequence of expected usage. That is, the Data Sets menu items are to be executed before the Initialization menu items, then lastly the Execution menu items. Each of these has sub-menus, or *items* that are described below. The figures referenced refer to figures in the next section, a tutorial on the use of the Toolbox.

The first Input menu (see Figure 2) item allows the user to identify two files: the file containing the raw full data matrix (*b*; see Figure 4) file, and the raw partial data matrix (*cb*; see Figure 5) file. A dialog box(see Figure 3) asks to select MATLAB format (.mat) or ASCII for the input. The dimensionality of the input is checked, and the size of the *cb* matrix stored is displayed in the Data Set window of the GUI.

[†] In order to describe the procedures discussed in this paper, commercial products are identified. In no case does such identification imply recommendation or endorsement by the National Institute of Standards and Technology or that the materials or equipment specified are necessarily the best available for the purpose.

The second menu provides initialization of the data. The first sub-menu (see Figure 6) controls the optional ordering of missing data rows in *cb* array. When selected, the arrays *b* and *cb* are checked for further processing. *cb* is examined for the existence of null elements or "nan" (not a number) strings. If there are no missing data elements in the array, and the array contains fewer rows than the *b* array, the missing data array requires no further processing. When *cb* is processed, those rows containing missing data elements (vector *missdatarow*) are moved to the highest-numbered rows of the *cb* array. The same-numbered rows of the *b* array are also moved to the highest-numbered rows. The missing data rows of *cb* are set to zero, and the count of those rows is displayed in the m1 window of the GUI. The array c is created from the array *cb*.

The next sub-menu controls the division of *b* into a validation array and a training array. A subfigure (see Figure 7) requests the number of the beginning column, the increment between selection columns, and the number of the end column, using the form "[1:3:150]" for the respective selections. The first value is the number of the first column to be selected. The second is the increment to be applied to select successive columns. The third is the number of the final column to be selected. Columns of the raw full input array (m by n) are split into a validation matrix (SVAL; m by q) and the remainder becomes the training matrix (STRAIN; m by p). The following figure indicates the dimensions of the three matrices and the labels of the dimensions.

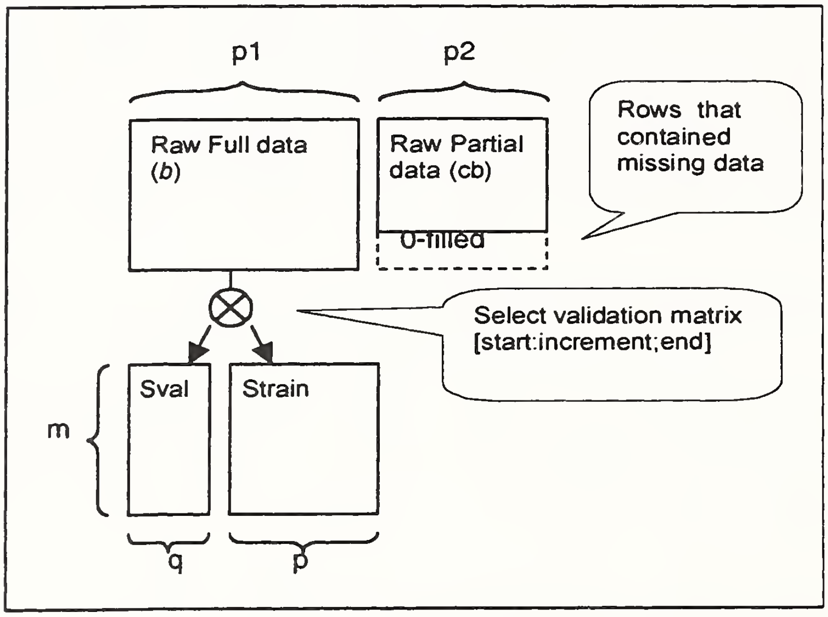

p1
p2
Raw Full data (*b*)
Raw Partial data (cb)
0-filled
Rows that contained missing data
Select validation matrix [start:increment;end]
Sval
Strain
m
q
p

Figure 1. Initialization of Arrays

The remaining Initialization menu item brings up a dialog box (see Figure 8) for entering the number of model vectors. A range is specified by entering a minimum, maximum, and step size.

The Execute menu item (see Figure 9) runs the selected method for each of the estimated model vectors. The RMS errors from the validation set for test points with (*Y1*) and without (*Y2*) missing data are returned and plotted (see Figure 10) in the GUI's axis. The numerical RMS error values are printed (see Figure 11) in the MATLAB command window.

Executing Method 4 brings up an additional dialog box for entering a value for K0; the number of singular vectors to be used in the initial estimation of missing data points.

Executing Methods 5 and 6 will run iterations of Methods 4 and 3, respectively, using the setting for K0. Either Method 5 or 6 will bring up a dialog box (see Figure 12) for entering the number of iterations and tolerance to control actions of those methods.

The results of execution of any method are plotted on the GUI. The axis of the plot is labeled with the number of the method used. Two lines indicate the error values returned, RM error for points with missing data and the RMS errors for points with no missing data, and a legend indicates which plot line represents which value. The error values are also printed in MATLAB's command line window. Successive methods may be executed without re-entering or re-initializing the input arrays.

3.2 Missing Data Toolbox Tutorial

In the following example, demonstration arrays have been created using random number routines to generate a 100 by 150 array for the full data (*b*) array, and a 100 by 50 array for the partial data (*cb*) array. The formulas used to create these arrays are described in the Simulation section below.

To start the MissingData Toolbox from the MATLAB Command Window, type

```
>> missingdata
```

The Missing Data Toolbox window (GUI) will appear. Along the top of the window you will see the menu headings [Input], [Initialization], and [Execute]. Sub-menus of the Input menu item are shown in Figure 2.

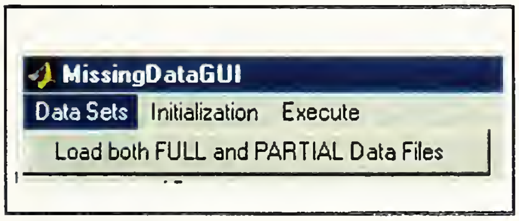


Figure 2. Data Sets menu item

The first input will be the full raw data array. Selecting the sub-menu item under the Data Sets item will bring up a dialog box for selecting the file type to be input as shown in Figure 3. Normally, input will be a MATLAB-native matrix.

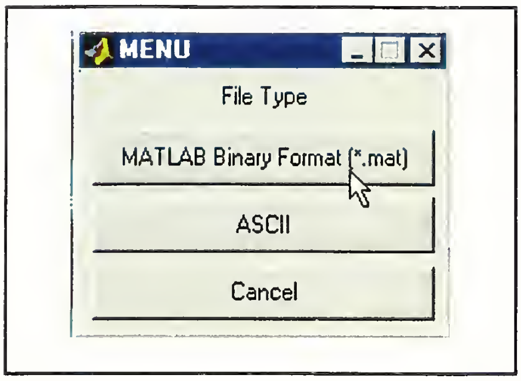


Figure 3. File Type selection dialog box

On selecting the file type, two successive input selection browser windows appear. The first of these expects selection of the full raw data array (*b*) as shown in Figure 4.

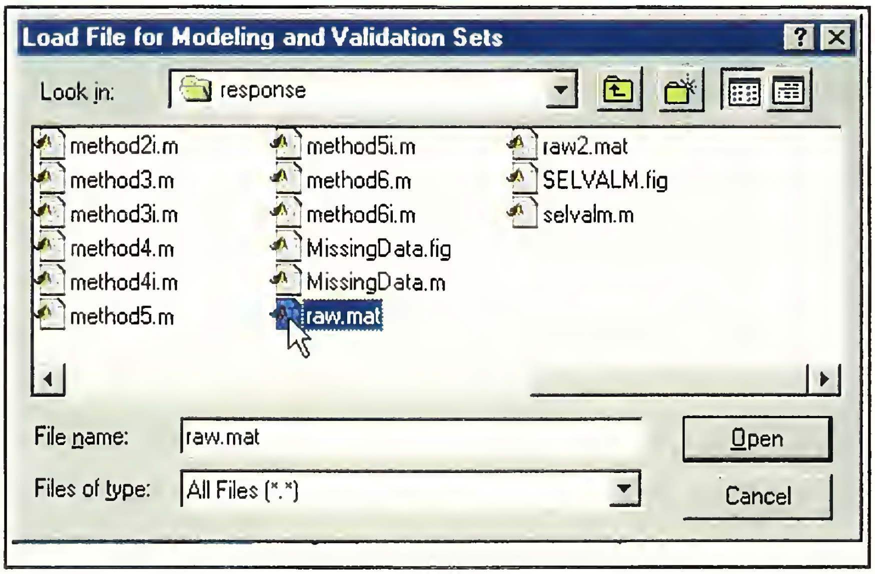


Figure 4. Loading Raw-Full array dialog box

The input selection browser returns, and in the same manner as before, the selection of the partial array containing missing data (*cb*) is selected as shown in Figure 5.

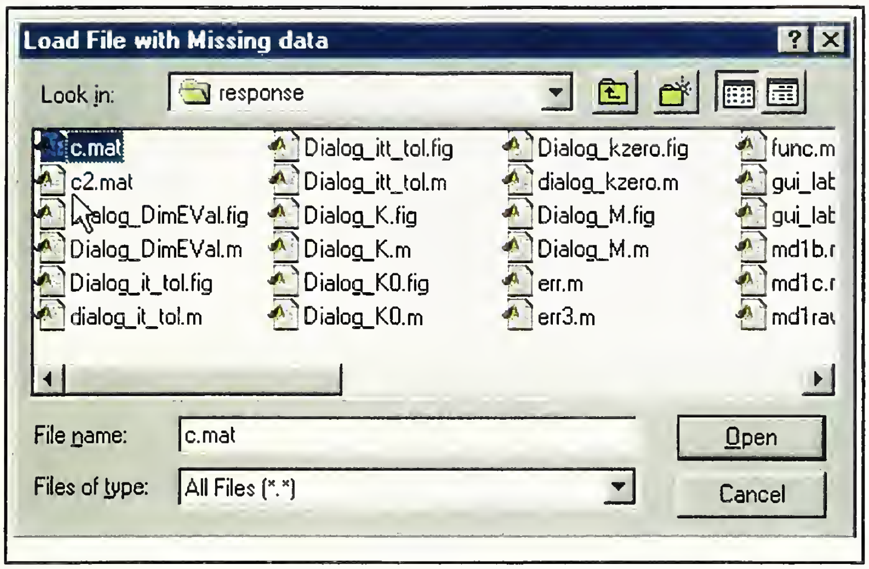
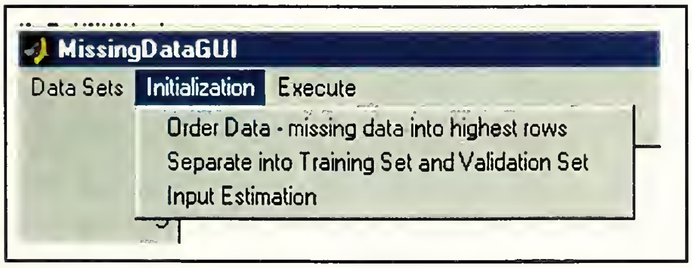


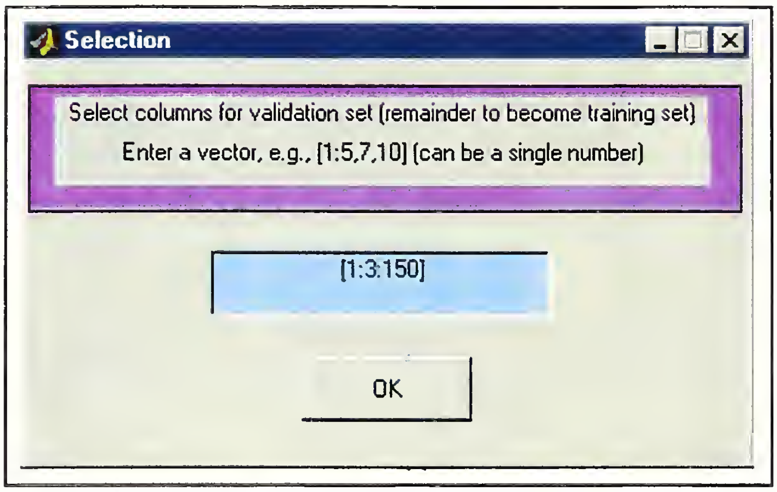
Figure 5. Enter Raw-Partial array dialog box

Following the second file selection, the size of the full data array will be displayed in the Toolbox window. The Initialization menu item is selected next. The first sub-menu item is optional. If the missing data has not been manually moved into the highest-numbered rows of the partial-data array, selecting this sub-menu item will search that array. Each occurrence of a missing data element results in its row being moved into the highest-numbered row of the array. The same-numbered row of the full data array also is moved into the highest-numbered row of its array. On reaching the end of the partial data array, the number of rows found to contain missing data will be displayed in the m1 area of the toolbox. The vector *missdatarow* contains the new ordering of the rows and can be used to interpret the results or to return the arrays to their original order after the missing data elements have been estimated.

The second menu item controls the preparation of the partial data array, the separation of the full array into validation and training arrays, and the control of the parameter K as shown in Figure 6.


MissingDataGUI
Data Sets
Initialization
Execute
Order Data - missing data into highest rows
Separate into Training Set and Validation Set
Input Estimation
Figure 6. Initialization menu item

The second sub-menu item under the Initialization menu controls the division of the full array into two separate arrays needed by the evaluation methods. Either a single value is entered, or a vector is entered as shown in Figure 7. Clicking the OK button results in dividing the full array into a Validation array (SVAL) and a training array (STRAIN). The size of each of these arrays will be displayed in the Toolbox.



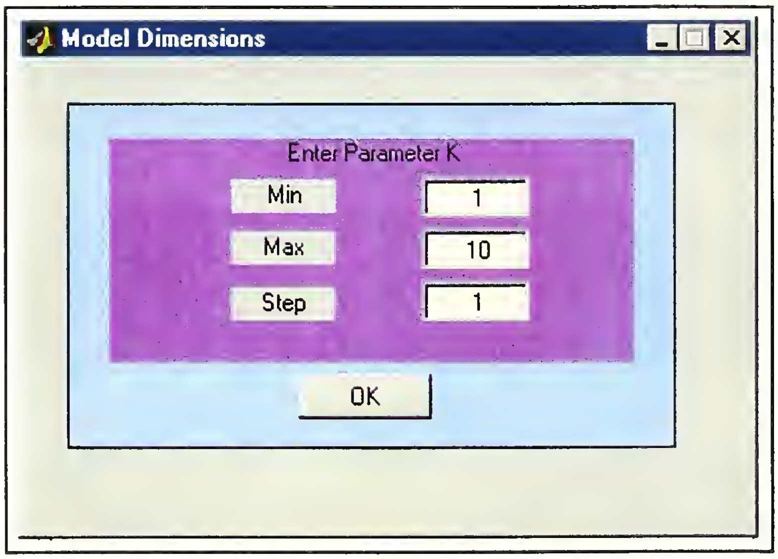
Select columns for validation set (remainder to become training set)

Enter a vector, e.g., [1:5,7,10] (can be a single number)

[1:3:150]

Figure 7. Selection criteria for Validation array dialog box

Selecting the third sub-menu item under the Initialization menu item brings up a dialog box for controlling the number of model vectors, *k*, to be used by the methods for execution as shown in Figure 8.



Enter Parameter K

Min	1
Max	10
Step	1

Figure 8. Parameter K dialog box

The Toolbox is now prepared for executing any or all of the 6 methods. The third menu item is used to execute one of the 6 methods as shown in Figure 9.

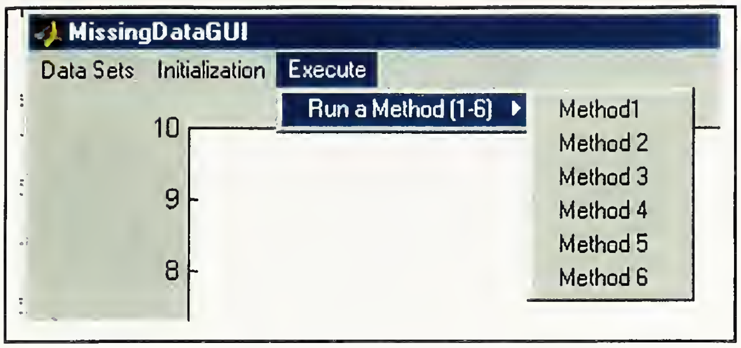
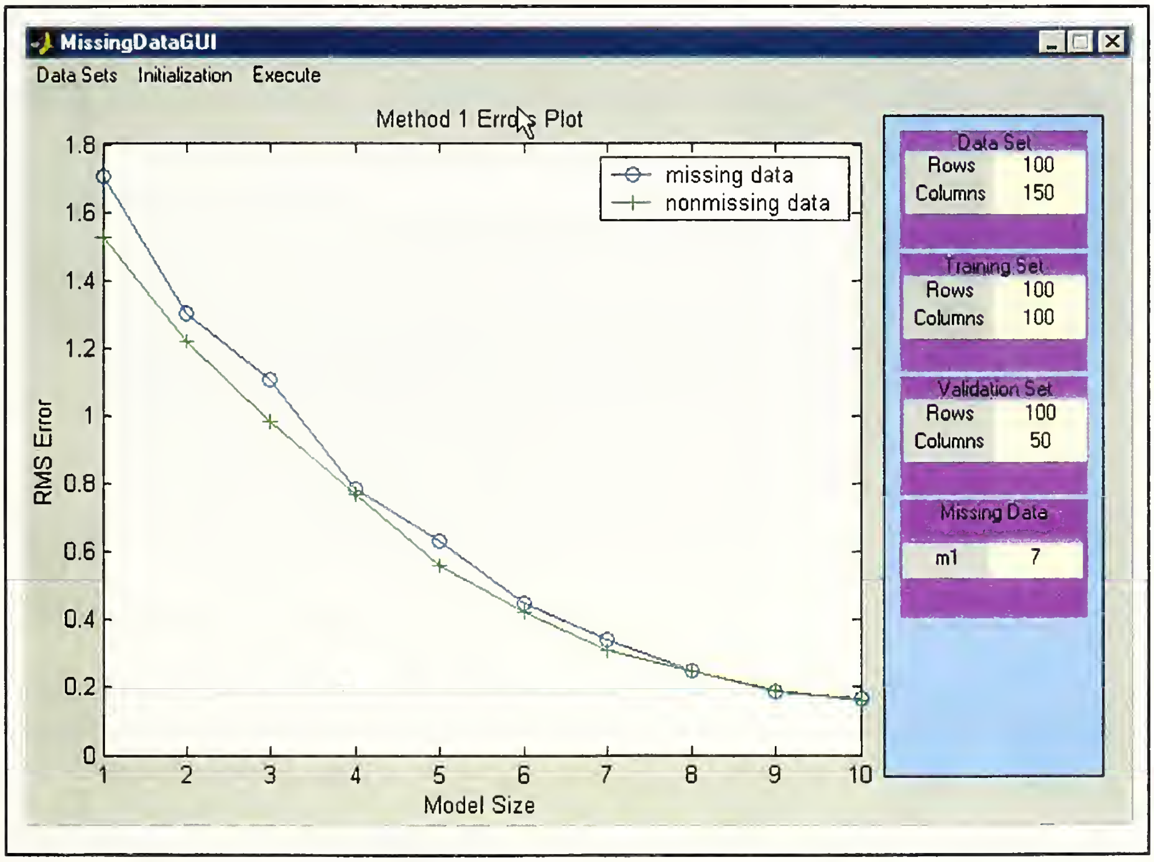


Figure 9. Execute menu item

To run Method 1, for example, select the Execute menu item, then Run a Method (1 – 6), then over to Method 1. Method 1 is executed and the error values are returned for each model size value selected earlier. The returned error results will be plotted as shown in Figure 10.



Data Set	
Rows	100
Columns	150

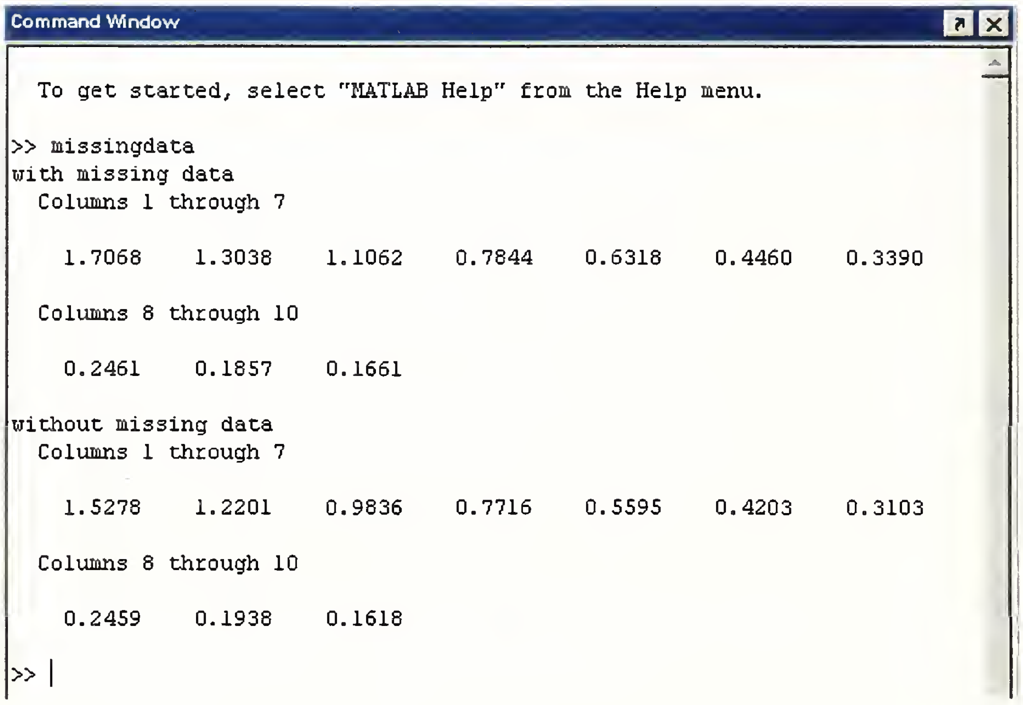
Training Set	
Rows	100
Columns	100

Validation Set	
Rows	100
Columns	50

Missing Data	
m1	7

Figure 10. Method 1 example results plot

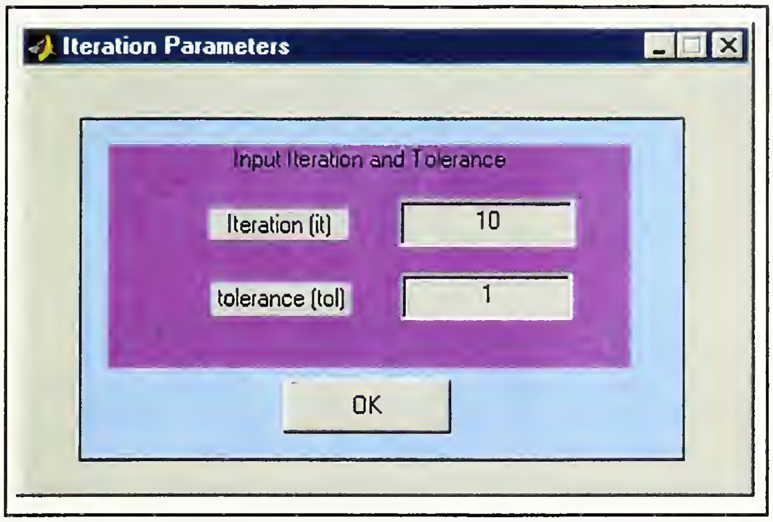
The returned error results are also printed in the MATLAB command window as shown in Figure 11.



```
  To get started, select "MATLAB Help" from the Help menu.

>> missingdata
with missing data
  Columns 1 through 7

    1.7068    1.3038    1.1062    0.7844    0.6318    0.4460    0.3390

  Columns 8 through 10

    0.2461    0.1857    0.1661

without missing data
  Columns 1 through 7

    1.5278    1.2201    0.9836    0.7716    0.5595    0.4203    0.3103

  Columns 8 through 10

    0.2459    0.1938    0.1618

>> |
```

Figure 11. Values of Error returned

Returning to the Execute menu item, other methods may also be run. Each will plot and print the returned error results. When method 5 or 6 is selected, a dialog box appears asking for the values for the number of iterations to be run and the tolerance as shown in Figure 12.


Iteration Parameters

Input Iteration and Tolerance

Iteration (it): 10

tolerance (tol): 1

Figure 12. Iteration and Tolerance dialog box

The graphs of errors returned from the methods executed provide a quick means of evaluating the method(s) appropriate for a given set of data. The numeric error values may be copied from the MATLAB Command window for documenting the analysis of missing data.

4. Simulation

The data on these devices is assumed to come from an underlying linear[3] model. This model has *n* components, where the true value of *n* is unknown. The data available on the devices is corrupted by noise *en*, which is random, normally distributed, and independent for each test point. The standard deviation of this noise is *e*. Let *at* be the true model, which is *m* by *n*, and *amp* is the *n*-dimensional vector of the average amplitudes of the *n* components for the set of devices being modeled. Let *randn(m,n)* designate an *m* by *n* matrix whose values are random, normally distributed with standard deviation one. For purposes of this study, candidate true models are generated by

$$at = randn(m,n).*(ones(m,1)*amp)$$

where .* represents an element by element multiplication, * represents a matrix multiplication, and *ones(m,1)* is an *m* by 1 matrix of ones. The corresponding *b* and *c* matrices are given by

$$b = at*randn(n,p1) + e*randn(m,p1),$$

and

$$c = at*randn(n,p2) + e*randn(m,p2).$$

A validation matrix represents data taken at all m test points on q devices. The true value of the validation devices is given by

$$vt = at * randn(n,q),$$

and the measured validation data is given by

$$vm = vt + e * randn(m,q).$$

4.1 Ignore partial data

For reference purposes, method 0 considers the case where there is no missing data, i.e., when *ct* is known. Let *k* be the estimate for number of model vectors *n*. Then the estimated model *a*0*k* is given by

$$[u,s,v] = svd([c\ \ b]),$$

$$a0k = u(:,1:k).$$

where $[c\ \ b]$ is the *m* by *p*1+*p*2 matrix from appending *c* to *b*, *svd* is the singular value decomposition function that calculates the usual vectors *u*, *s*, and *v*, and *u*(:,1:*k*) is the first *k* columns of *u*.

4.2 Validation errors

The validation errors are calculated three ways for the points with and without missing data. In the first case, the errors are calculated by using the model estimate and the true validation data *vt* to predict the missing values, and taking the difference of this prediction with the true validation data as

$$vh1 = aik * (aik'*aik) \backslash aik'*vt$$
$$v1diff = vt - vh1,$$
$$v1difft = rms(v1diff(1:m1,:)), \quad \text{and}$$
$$v1diffb = rms(v1diff(m1+1:m,:)).$$

Where “'” denotes the transpose of a matrix.

In the second case, the errors are calculated by using the model estimate to predict the measured validation data *vm* and taking the difference of this prediction with the true validation data as

$$vh2 = aik * (aik'*aik) \backslash aik'*vm$$
$$v2diff = vt - vh2,$$
$$v2difft = rms(v2diff(1:m1,:)), \quad \text{and}$$

$v2diffb = rms(v2diff(m1+1:m,:))$.

In the third case, the errors are calculated by using the model estimate to predict the measured validation data *vm* and taking the difference of this prediction with the measured validation data as

$vh3 = aik*(aik'*aik) \backslash aik'*vm$
$v3diff = vm - vh3$,
$v3difft = rms(v3diff(1:m1,:))$, and
$v3diffb = rms(v3diff(m1+1:m,:))$.

4.3 Ignore partial data

The first method considered is to ignore the partial information. That is, only use the full data set information in *b* (See Section 2). Thus, the estimated model is given by

$[u,s,v] = svd(b)$,

$a1k = u(:,1:k)$.

4.4 Missing data estimation methods

Six methods were evaluated for using the partial information in *cb*. The first of these, method 2, uses *b* and *cb* to estimate the model as follows:

$[u,s,v] = svd([cb\ \ bb])$,
$abk = u(:,1:k)$,
$a2k = b*((bb'*bb) \backslash bb')*abk$.

Method 3 uses *b* and *cb* to first estimate *ct* to give an estimate of the full *c*. It then estimates the model from *b* and the estimated *c* as follows:

$ct = bt*(bb'*bb) \backslash bb'*cb,$
$[u,s,v] = svd([[ct;cb]\ \ b]),$
$a3k = u(:,1:k),$

where [*ct*;*cb*] is the *m* by *p*2 matrix obtained by placing *ct* above *cb*.

Method 4 uses the *k*0 principle components of *b* to estimate *ct*, then to estimate the model from *b* and the estimated *c* as follows:

$[u,s,v] = svd(b),$

$ak0t = u(1:m1,1:k0),$

$ak0b = u(m1+1:m,1:k0),$

$ct = ak0t * ((ak0b'*ak0b) \backslash ak0b') * cb,$

$[u,s,v] = svd([[ct;cb]\ \ b]),$

$a4k = u(:,1:k),$

where *ak0t* is the first *m*1 rows of the first *k*0 principle components of *b* and *ak0b* are the lower *m-m1* rows.

Method 5 iterates on method 4 as follows:

```
$[u,s,v] = svd(b),$
for $i = 1:it$
    $ak0t = u(1:m1,1:k0),$
    $ak0b = u(m1+1:m,1:k0),$
    $cti = ak0t * ((ak0b'*ak0b) \backslash ak0b') * cb,$
    $[u,s,v] = svd([[cti;cb]\ \ b]),$
end
$a5k = u(:,1:k),$
```

where *it* is the number of iterations (chosen by trial and error).

Method 6 uses method 3 for the initial estimate of *ct*, than iterates as in method 5 as follows:

```
$ct = bt * (bb'*bb) \backslash bb'*cb,$
$[u,s,v] = svd([[ct;cb]\ \ b]),$
for $i = 1:it$
    $ak0t = u(1:m1,1:k0),$
    $ak0b = u(m1+1:m,1:k0),$
    $cti = ak0t * ((ak0b'*ak0b) \backslash ak0b') * cb,$
    $[u,s,v] = svd([[cti;cb]\ \ b]),$
end
$a6k = u(:,1:k).$
```

5. Example of Results

The following results depict a case where the full number of test points m is 100, and the number of missing data points m1 is 4. The number of devices with full data p1 is 100, the number with partial data p2 is 100, and the number of validation devices q is 50. The true and estimated number of model vector n, and nh (as well as k and k0) are all 40. The error amplitude e is 0.01. The model component amplitudes amp are provided for by the equation

$$amp = 1./(1+((1:n)/(n/8)).{}^{\wedge}3).$$

For $n = 40$ this gives amplitudes from 0.9912 to 0.0019 as follows

0.9921	0.9398	0.8224	0.6614	0.5000	0.3666	0.2671
0.1962	0.1464	0.1111	0.0859	0.0675	0.0538	0.0436
0.0357	0.0296	0.0248	0.0210	0.0179	0.0154	0.0133
0.0116	0.0102	0.0090	0.0079	0.0071	0.0063	0.0057
0.0051	0.0046	0.0042	0.0038	0.0035	0.0032	0.0029
0.0027	0.0025	0.0023	0.0021	0.0019 .		

Using the models to predict the true validation data *vt* and taking the difference between the prediction and the true validation data gives the average as

method0	method1	method2	method3	method4	method5	method6	
0.0039	0.0060	0.0143	0.0058	0.0061	0.0060	0.0059	(missing)
0.0040	0.0064	0.0066	0.0052	0.0041	0.0041	0.0050	(nonmissing).

The standard deviation of these results are

method0	method1	method2	method3	method4	method5	method6	
0.0002	0.0004	0.0020	0.0009	0.0004	0.0004	0.0012	(missing)
0.0001	0.0004	0.0005	0.0001	0.0001	0.0001	0.0002	(nonmissing).

Using the models to predict the measured validation data *vm* and taking the difference between the prediction and the true validation data gives the average as

method0	method1	method2	method3	method4	method5	method6	
0.0073	0.0086	0.0165	0.0110	0.0086	0.0087	0.0107	(missing)
0.0075	0.0090	0.0091	0.0081	0.0076	0.0076	0.0080	(nonmissing).

The standard deviation of these results are

method0	method1	method2	method3	method4	method5	method6	
0.0006	0.0006	0.0021	0.0006	0.0005	0.0005	0.0010	(missing)
0.0001	0.0002	0.0003	0.0001	0.0001	0.0001	0.0001	(nonmissing).

Using the models to predict the measured validation data *vm* and taking the difference between the prediction and the measure validation data gives the average as

method0	method1	method2	method3	method4	method5	method6

0.0089	0.0099	0.0156	0.0066	0.0100	0.0098	0.0077	(missing)
0.0088	0.0100	0.0103	0.0094	0.0088	0.0088	0.0093	(nonmissing).

The standard deviation of these results are

method0	method1	method2	method3	method4	method5	method6	
0.0005	0.0005	0.0018	0.0008	0.0005	0.0006	0.0010	(missing)
0.0001	0.0002	0.0004	0.0002	0.0001	0.0001	0.0002	(nonmissing).

6. Conclusions

The Toolbox described has been used to evaluate improvements in measured instrument models[4], indicating that some of the methods produce models that are close to those cases where no data are missing.

7. References

[1] A.D Koffman, T.M. Souders, G.N. Stenbakken, T.E. Lipe, and J.R. Kinard, "Emperical Linear Prediction Applied to a NIST Calibration Service," Proceedings of the 1996 National Conference of Standards Laboratories (NCSL) Workshop and Symposium, August 25-29, Monterey, CA, pages 207-212.

[2] A. Koffman, T. Souders, G. Stenbakken, H. Engler, "High-Dimensional Empirical Linear Prediction (HELP) Toolbox Users Guide 2.2," NIST TN 1428, May 1999.

[3] G.N. Stenbakken and T.M Souders, "Developing Linear Error Models for Analog Devices," IEEE Transactions on Instrument Measurement, Volume 43 Number 2 (April 1994), pages 157-163.

[4] G.N. Stenbakken, H. Liu, and G. Hwang, "Empirical Modeling Methods using Partial Data," IEEE Instrumentation and Measurement Technology Conference, 21-23 May 2002, Anchorage, AK, pages 875-879.

Attachment 1: The Missing Data Web Page

The requirements for the Missing Data Toolbox, interfaced to the method and error calculation routines, were evaluated by other project members by means of a project Web page as shown below.

Missing Data

- a new Toolbox interface definition

Toolbox Functional Description

A series of measurements can be recorded where there are gaps (missing data) in the final matrix. The HELP Toolbox expects the matrix to be complete (no missing data).
For cases where the measurement matrix is incomplete, the Missing Data Toolbox becomes a means for analyzing the accuracy impact of the missing data, and for developing suitable data inclusions such that the HELP Toolbox may be utilized.

Step

1. Ask for dimension of EVAL*
2. Separate *B* into STRAIN and SVAL*
3. Display dimensions of each set
4. Ask for value estimation method
5. Call appropriate method with parameters (VEParam, MData, STRAIN, SVAL) (returned is QV and Estimated matrix: QualV, ETRAN, EVAL)
6. Display quality of estimation (all done so far, i.e., QualM)
7. Ask if Finished or another method to be tried
8. If repeat go to step 4
   Otherwise set
   - ATRAIN to ETRAIN,
   - AVAL to EVAL
   - Quality to last QualV

*These items use the same method as the HELP Toolbox.

Parameters

B

- Raw full data matrix

cb

- Raw partial data matrix

MData
- Missing data indicator (same size as *cb*)

STRAIN
- Selected training matrix from *b*

SVAL
- *B* - STRAIN to be used validation matrix

ETRAIN
- Estimated training matrix

EVAL
- Estimated validation matrix

VEMeth
- Value estimation method; select from 1 - 6

QualV
- Quality vector

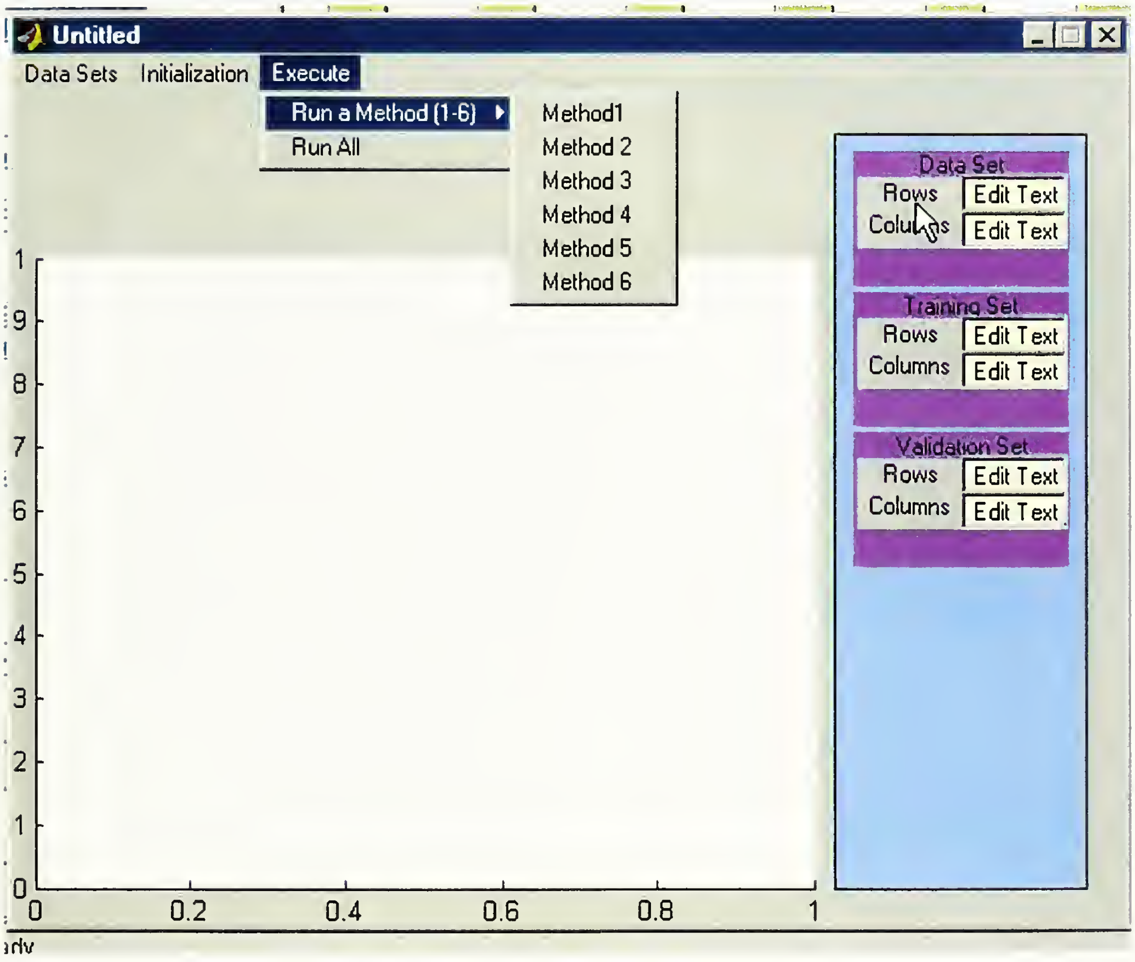
DT>VEParam
- Value estimation parameter; may consist of k or k&k0 or k, it, tol (unique to each VEMeth)

Routines

Value Estimation Method
VEM11-n
- input: (VEParam,MData,SModel,SValid)
- output: (QualV,EModel,EValid)

Attachment 2: Missing Data Graphical User Interface

A prototype graphical user interface was originally developed to evaluate the appearance and utility expected of the Toolbox. Review of this prototype confirmed that an interface that is similar to the HELP Toolbox interface would be advantageous for the Missing Data Toolbox.


Untitled
Data Sets
Initialization
Execute
Run a Method (1-6)
Run All
Method1
Method 2
Method 3
Method 4
Method 5
Method 6
Data Set
Rows
Columns
Edit Text
Edit Text
Training Set
Rows
Columns
Edit Text
Edit Text
Validation Set
Rows
Columns
Edit Text
Edit Text
0
0.2
0.4
0.6
0.8
1