A Computer Program to Obtain Time-Correlated Gust Loads for Nonlinear Aircraft Using the Matched-Filter-Based Method

February 1994
Final Report

This document is available to the public through the National Technical Information Service, Springfield, Virginia 22161.
NOTICE

This document is disseminated under the sponsorship of the U. S. Department of Transportation in the interest of information exchange. The United States Government assumes no liability for the contents or use thereof.

The United States Government does not endorse products or manufacturers. Trade or manufacturers' names appear herein solely because they are considered essential to the objective of this report.
A COMPUTER PROGRAM TO OBTAIN TIME-CORRELATED GUST LOADS FOR NONLINEAR AIRCRAFT USING THE MATCHED-FILTER-BASED METHOD

Robert C. Scott, Anthony S. Pototsky and Boyd Perry, III

Aeroelasticity Branch
NASA Langley Research Center
Mail Stop 340
Hampton, VA 23681

This report was produced as a part of the aircraft Catastrophic Failure Prevention Research Program ID# 066-110, and under FAA-NASA Interagency Agreement No. DTFA03-89-A-00019. The work was performed at NASA Langley Research Center, Hampton, VA 23681. The FAA Technical Center COTR was Thomas DeFiore.

NASA Langley Research Center has, for several years, conducted research in the area of time-correlated gust loads for linear and nonlinear aircraft. The results of this work led NASA to recommend that the Matched-Filter-Based One-Dimensional Search Method be used for gust load analyses of nonlinear aircraft. This manual describes this method, describes a FORTRAN code which performs this method, and presents example calculations for a sample nonlinear aircraft model. The name of the code is MFBDIS (Matched-Filter-Based One-Dimensional Search). The program source code, the example aircraft equations of motion, a sample input file, and a sample program output are all listed in the appendices.
TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Executive Summary</th>
<th>v</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>1</td>
</tr>
<tr>
<td>Background</td>
<td>1</td>
</tr>
<tr>
<td>Review of the MFB Method For Linear Systems</td>
<td>2</td>
</tr>
<tr>
<td>MFB One-Dimensional Search Description</td>
<td>3</td>
</tr>
<tr>
<td>Selection of Gust Intensities</td>
<td>4</td>
</tr>
<tr>
<td>Description of MFB1DS</td>
<td>5</td>
</tr>
<tr>
<td>Required Files</td>
<td>6</td>
</tr>
<tr>
<td>MFB1DS.F Subroutines</td>
<td>6</td>
</tr>
<tr>
<td>MODEL.F</td>
<td>6</td>
</tr>
<tr>
<td>Description of Impulse Input</td>
<td>7</td>
</tr>
<tr>
<td>Description of Output</td>
<td>7</td>
</tr>
<tr>
<td>Numerical Example</td>
<td>8</td>
</tr>
<tr>
<td>Examination of Results</td>
<td>10</td>
</tr>
<tr>
<td>Concluding Remarks</td>
<td>11</td>
</tr>
<tr>
<td>References</td>
<td>12</td>
</tr>
<tr>
<td>Appendix A - MFB1DS.F source code listing</td>
<td></td>
</tr>
<tr>
<td>Appendix B - MFB1DS.INC source code and description of parameters</td>
<td></td>
</tr>
<tr>
<td>Appendix C - MODEL.F (ARW-2) source code listing</td>
<td></td>
</tr>
<tr>
<td>Appendix D - MODEL.INP file for the ARW-2 model</td>
<td></td>
</tr>
<tr>
<td>Appendix E - A sample listing of the MFB1DS program output ...</td>
<td></td>
</tr>
</tbody>
</table>
### LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>MFB linear method signal flow diagram.</td>
<td>13</td>
</tr>
<tr>
<td>2</td>
<td>Nonlinear MFB signal flow diagram for the one-dimensional search.</td>
<td>14</td>
</tr>
<tr>
<td>3</td>
<td>MFB1DS solution procedure.</td>
<td>15</td>
</tr>
<tr>
<td>4</td>
<td>Nonlinear block diagram for the ARW-2 drone aircraft.</td>
<td>16</td>
</tr>
<tr>
<td>5</td>
<td>Example calculations illustrating the effects of k variation on maximum WRBM, $\sigma_g = 1,530$ in/sec.</td>
<td>17</td>
</tr>
<tr>
<td>6</td>
<td>Maximum WRBM versus k using a refined range of k values, $\sigma_g = 1,530$ in/sec.</td>
<td>18</td>
</tr>
<tr>
<td>7</td>
<td>Maximum WRBM versus k for 3 gust intensities.</td>
<td>19</td>
</tr>
</tbody>
</table>
EXECUTIVE SUMMARY

This report was sponsored under the Federal Aviation Administration's Aircraft Catastrophic Failure Prevention Research Program, Mission Need Statement No. 066-110. This report documents the NASA recommended Matched-Filter-Based One-Dimensional Search Method for the gust load analysis of nonlinear aircraft. The FAA Technical Program Monitor was Mr. Terry Barnes, ANM-105N, National Resource Specialist, Flight Loads and Aeroelasticity. FAA COTR was Mr. Thomas DeFiore, ACD-220, Principal Investigator, Flight Loads.
INTRODUCTION

This manual and the accompanying code were developed as partial fulfillment of a NASA agreement with the FAA. The agreement had three main tasks. First, two NASA developed gust analysis methods were to be brought to the same level of maturity. These analysis methods were the Matched-Filter-Based (MFB) method and the Stochastic-Simulation-Based (SSB) method. Upon completion of the development work, the second task was to compare the methods and make a recommendation selecting which approach was best suited for nonlinear analyses. This work was completed and the results given in a presentation at the Gust Specialists Meeting in LaJolla, California on April 22, 1993. At the meeting, it was recommended by NASA that the MFB one-dimensional search was the method of choice for time-correlated gust loads analysis of aircraft with nonlinear systems. The third and final task was to develop a transportable computer program and accompanying documentation for using the recommended method.

This manual describes a computer code called "MFB1DS" which performs the Matched-Filter-Based one-dimensional search. The MFB one-dimensional search is a deterministic method for obtaining maximized and time-correlated design loads for aircraft with nonlinearities. This method can, however, be applied to both linear and nonlinear aircraft. The paper summarizes the method, discusses the selection of gust intensity for the method, describes the FORTRAN code MFB1DS that performs the calculations, and presents numerical results for an example aircraft.

BACKGROUND

With the advent of aircraft that contain large numbers of nonlinearities in their flight control systems and/or gust load alleviation systems, existing methods for certifying aircraft for gust loads may not be adequate. For several years NASA Langley Research Center has conducted research in the area of time correlated gust loads and has published a number of papers on the subject (refs. 1-6). The initial research was restricted to mathematically linear systems (refs. 1-3). Recently, however, the focus of the research has been on defining methods that will compute design gust loads for an airplane with a nonlinear control system (refs. 4-6). To date, two such methods have been defined: one is based on matched filter theory; the other is based on stochastic simulation.

The Matched-Filter-Based (MFB) method was developed first and was reported on in reference 4. The MFB method employs optimization to solve for its answers and this method comes in two varieties: the first uses a one-dimensional search procedure and the second a multi-dimensional search procedure. The first is significantly faster to run and, based on experience with a number of nonlinear models gives design loads only slightly lower in magnitude than the second.
The Stochastic-Simulation-Based (SSB) method has evolved over the past two years. References 5 and 6 describe the method. In reference 6 a comparison of the MFB and SSB methods was made. The results predicted by the two methods are strikingly similar and demonstrate that the key quantities from the MFB method (viz. critical gust profile, maximized load, and time-correlated load) are realizable in a stochastic analysis.

Another significant finding in reference 6 was the relative computational costs of performing analyses using the MFB and SSB methods. For linear models, the MFB method is much more efficient than the SSB method. For nonlinear models, the MFB multi-dimensional search is much more expensive than the SSB method, while the MFB one-dimensional search requires less time that the SSB method.

Since the MFB multi-dimensional search method as now implemented is prohibitively expensive, the options for methods that can practically be applied to nonlinear systems are the MFB one-dimensional search and the SSB methods.

Based on the results in reference 6, the one-dimensional search is able to predict the maximized loads for nonlinear systems. In addition, it requires less computer resources than the SSB method. These factors show the utility of using the MFB one-dimensional search as a means of obtaining time-correlated gust loads for aircraft with nonlinear control systems.

**REVIEW OF THE MFB METHOD FOR LINEAR SYSTEMS**

The purpose of this section of the manual is to review the basic matched filter concepts. A detailed theoretical development of the MFB linear method can be found in reference 2. The signal flow diagram in figure 1 outlines the implementation and illustrates the intermediate and final products of the method.

Transfer-function representations of atmospheric turbulence and airplane loads are combined in series and represent the "known dynamics" boxes in the figure. A transfer-function representation of the von Karman spectrum is chosen for the gust filter. Load \( y \) is the load to be maximized. Loads \( z_1 \) through \( z_n \) are the loads to be time correlated with load \( y \). There are three major steps in the process:

- **Step i**  The application of an impulse function of unit strength to the combined linear system, producing the impulse response of load \( y \), \( h(t) \).

- **Step ii**  The normalization of this impulse response by the square root of its energy and then reversing it in time.

- **Step iii**  The application of this normalized reversed signal to the combined linear system, producing time histories of load \( y \) and time histories of loads \( z_1 \) through \( z_n \).

For simplicity of discussion these three steps will be referred to as the "MFB linear method."
The square root of the energy is defined by

\[ \sqrt{\text{energy}} = \sqrt{\int_0^{t_0} h^2(t) \, dt} \]  

(1)

where \( h(t) \) is the impulse response of load \( y \). For a stable system the numerical value of the quantity on the right side of equation (1) approaches a constant value as \( t_0 \) is increased.

The selection of \( t_0 \) is based on the time required for the load impulse responses to damp out to near zero. Figure 1 shows that the impulse response of load \( y \) has essentially damped out at time \( t_0 \). Thus, \( t_0 \) is large enough for the response shown in figure 1. Too large a \( t_0 \) value will, however, unduly increase the amount of computations required. The analyst must choose a value that provides accurate results while minimizing the use of computer resources.

Within the time history of load \( y \) in step iii, the maximum value is \( y_{\text{max}} \). For linear systems, the theory guarantees that no other signal similarly normalized will produce a value of \( y \) larger than \( y_{\text{max}} \). This guarantee is a fundamental result of the MFB linear method.

**MFB ONE-DIMENSIONAL SEARCH DESCRIPTION**

The goal of Matched Filter Theory as applied to nonlinear systems is the same as that for linear systems: to find the maximized response time history, the maximum value of the response within that time history, and the time-correlated response time histories. Because the systems are not linear, the superposition principle of the MFB linear method no longer holds and the solutions for maximized loads cannot conveniently be obtained directly. The only practical means of finding the excitation waveform that maximizes \( y_{\text{max}} \) is a search procedure. The search is conducted systematically, subject to the constraint that the excitation waveform have a "unit" energy.

Because superposition no longer holds, the magnitude and character of the responses are not necessarily proportional to the magnitude of the input. For the remainder of this paper, two input magnitudes are important: \( k \), the strength of the initial impulse; and \( \sigma_g \), the design value of the gust intensity. (For the MFB linear method, the magnitude of both of these quantities was unity.) For nonlinear systems and a specific \( \sigma_g \), the shape of the excitation waveform is a function of \( k \), and, consequently, the quantity \( y_{\text{max}} \) is also a function of this parameter.

The one-dimensional search procedure performs a systematic variation of the quantity \( k \) to find the shape of an excitation waveform that maximizes \( y_{\text{max}} \). Figure 2 contains a signal flow diagram for this search procedure. Figure 2 is very similar to figure 1, but contains some subtle yet important differences that are indicated by the shaded boxes and by quotation marks. In figure 2 the initial impulse has a non-unity strength; the aircraft loads portion of the known dynamics box contains nonlinearities; and the shape of the excitation waveform and the value of \( y_{\text{max}} \) are functions of the initial impulse strength. In addition, the "matched" excitation waveform and the "matched" load are shown in quotes because, for nonlinear systems, there is no guarantee that \( y_{\text{max}} \) is a global maximum.
The application of the one-dimensional search procedure is as follows:

**Step 1** Select a specific design value for $\sigma_g$.

**Step 2** Select a set of values for $k$.

**Step 3** For each value of $k$, perform steps i through iii of the MFB linear method to obtain a set of "$y_{\text{max}}$" values and also the corresponding "matched" excitation waveforms.

**Step 4** From step 3 above, find the maximum value of $y_{\text{max}}$ and its corresponding "matched" excitation waveform.

As seen in the first pass (upper half) of figure 2, the variation of the impulse strength ($k$) affects the MFB one-dimensional search analysis by changing the shape of the excitation waveform. For sufficiently low impulse strengths, the shape of the excitation waveform for nonlinear models will be invariant with $k$. While in this invariant region, the excitation waveforms will be similar to those that are obtained for linear models. For larger intensities the system nonlinearities are engaged and will cause the impulse responses and corresponding excitation waveforms to change shape. Consequently, they will differ from those obtained by linear systems.

As seen in the second pass (lower half) of figure 2, the gust intensity affects the MFB one-dimensional search analysis by scaling the excitation waveform prior to being applied to the nonlinear model. Consequently, a low gust intensity should result in the nonlinear model behaving linearly. As gust intensity is increased beyond some threshold the nonlinear model response will begin to deviate from that of its linear counterpart.

The effects of varying impulse strength and gust intensity will be discussed further in the numerical results section of the paper.

**SELECTION OF GUST INTENSITIES**

The purpose of this section of the paper is to present the reasoning behind the selection of the values of $\sigma_g$. Reference 6 describes the selection of gust intensities for the MFB and SSB methods.

The following equation, from reference 7, expresses the "design value" of quantity $y$ as defined in the design envelope criterion

$$y_{\text{design}} = \bar{A}_y U_\sigma$$  \hspace{1cm} (2)

where the quantity $\bar{A}_y$ is the RMS value of quantity $y$ per unit RMS gust intensity, obtained from a conventional random process analysis of the airplane and $U_\sigma$ is specified in the criterion. From reference 8 the quantity $U_\sigma$ in equation (2) is shown to be the product of the gust RMS value and the design ratio of peak value of load to RMS value of load, and, therefore the quantity $y_{\text{design}}$ is interpreted as a peak value.
Reference 4 shows that, as a consequence of the normalization of the excitation waveform by the square root of its own energy and the use of unity gust intensity, the quantity $y_{\text{max}}$ from the MFB linear method is equal to the quantity $\overline{A}_y$ from a conventional random process analysis, or

$$y_{\text{max}}(\sigma_g = 1) = \overline{A}_y$$  \hspace{1cm} (3)

In equation (3) $y_{\text{max}}$ is interpreted as an RMS value, not a peak value. Substituting equation (3) into equation (2), $y_{\text{design}}$ is now

$$y_{\text{design}} = y_{\text{max}}(\sigma_g = 1)U_{\sigma}$$  \hspace{1cm} (4)

If, in performing the MFB linear method, $U_{\sigma}$ is used for the gust intensity then the quantity $y_{\text{max}}$ is equal to

$$y_{\text{max}}(\sigma_g = U_{\sigma}) = \overline{A}_yU_{\sigma}$$  \hspace{1cm} (5)

The right hand sides of equations (2) and (5) are seen to be equal, therefore

$$y_{\text{design}} = y_{\text{max}}(\sigma_g = U_{\sigma})$$  \hspace{1cm} (6)

Two options for the value of $\sigma_g$ have been offered: $\sigma_g = 1$, for which $y_{\text{design}}$ is defined by equation (4); and $\sigma_g = U_{\sigma}$, for which $y_{\text{design}}$ is defined by equation (6). When analyzing a linear system the choice of $\sigma_g$ is irrelevant because the same value of $y_{\text{design}}$ will be obtained in either case. However, when nonlinearities are introduced into aircraft control systems, loads are not simply proportional to gust intensity. Consequently, $\sigma_g$ should be set to $U_{\sigma}$ in the MFB nonlinear calculations, or

$$\sigma_{g,\text{MFB}} = U_{\sigma}$$  \hspace{1cm} (7)

and the resulting "$y_{\text{max}}$" values from the method should be interpreted as $y_{\text{design}}$.

**DESCRIPTION OF MFBIDS**

MFBIDS is a FORTRAN 77 program which performs the Matched-Filter-Based one-dimensional search. The MFBIDS solution procedure follows what was outlined in figure 2. The code must be run once for each combination of gust intensity and output quantity for which a maximized value is desired.

Figure 3 shows the solution procedure used by MFBIDS. The program first reads the input data then generates the impulse responses for each k value by calling the simulation subroutine. The simulation subroutine uses a public domain ordinary differential equation solver to generate output time histories using a user defined subroutine containing the aircraft equations of motion. Next, the excitation waveforms are generated by reversing the impulse responses in time and normalizing them by the square root of their respective energies. Then, the simulation subroutine is again called for each of the normalized excitation waveforms to obtain the "maximized" load response time histories. Finally, the pertinent output quantities are written to computer files.
This section of the manual describes the main parts of MFB1DS.

Required Files

Six files are required to run MFB1DS.

- **MFB1DS.F** Main program, see Appendix A
- **MFB1DS.INC** File containing the common blocks used by MFB1DS.F, see Appendix B
- **MODEL.F** User supplied file containing the subroutine EQSMOT, the aircraft equations of motion, see Appendix C
- **MODEL.INP** User supplied file containing the input quantities required to run the code, see Appendix D
- **LSODE.F** Public domain ordinary differential equation solver
- **INTUTILS.F** Public domain subroutines used by LSODE.F

The two public domain files LSODE.F and INTUTILS.F are well documented in their respective source codes and will not be discussed in detail in this manual. These files were selected because of the fact that they are in the public domain. The user is encouraged to substitute more efficient ordinary differential equation solvers if available.

**MFB1DS.F Subroutines**

- **Subroutine DATAIN** Reads the input parameters from the input file
- **Subroutine SIMULATE** Computes output time histories using LSODE.F and MODEL.F
- **Subroutine SAVMATFORM** Saves output quantities to a MATRIX readable file
- **Subroutine MATSAV** Used by Subroutine SAVMATFORM to save data in MATRIXX format
- **Subroutine SAVASCIIFORM** Saves output quantities in a more easily readable ascii file

**MODEL.F**

MODEL.F is a user supplied file containing the subroutine EQSMOT, the equations for the gust filter in series with the aircraft equations of motion. This subroutine uses variables x (vector of states) and u (excitation) to obtain xd (derivative of x) and y (vector of outputs).
The following lines of code must be present at the top of the file MODEL.F:

```fortran
subroutine EQSMOT(neq,t,x,xd)
  parameter(maxout=20)
  double precision x(*), xd(*), y(maxout)
  double precision t, tstart, tend, u0, u1
  common /eqsmotcom/y,tstart,tend,u0,u1
  c
  u=(t-tstart)*(u1-u0)/(tend-tstart)+u0
end subroutine
```

In spite of the fact that the user specifies a desired time step, the ordinary differential equation solver LSODE calls the EQSMOT subroutine at values of time \( t \) between time steps. Consequently, the equation on the last line of the above code is included to provide a linear interpolation for values of \( u \) between time steps.

A linear system can be used to demonstrate how \( x \), \( x_d \), \( y \) and \( u \) are related. For a linear system only, the equations of motion can be written in state space form:

\[
\begin{align*}
\{x_d\} &= [A]\{x\} + [B]u \\
\{y\} &= [C]\{x\} + [D]u
\end{align*}
\]  

(8)

While equation (8) is linear, the relationship between \( x \), \( u \) and \( x_d \) is, in general, a nonlinear one.

**Description Of Impulse Input**

Since the impulse generating procedure is one of the key components of the program and there are several methods that can be used to generate impulse functions, a few words describing the impulse function procedure implemented in the program are warranted. The procedure chosen to generate the impulse function in this code is straightforward and can be found in main program listing in Appendix A. In general, the impulse input function, as seen by the differential equation solver, is a ramp up from zero at the first time step to a value of \( k/(2*deltat) \) at the second time step, a constant value of \( k(2*deltat) \) between time steps two and three, and a ramp down to zero between time steps three and four.

**Description Of Output**

Key information is written to standard out (unit 6) as shown in Appendix E. This information allows the user to monitor the progress of the program. Two additional output files can be created which contain time history and correlated output information. Both of these output files are ascii files: one is in a MATRIXX readable form and the other is in a more easily read form. The variables contained in the MATRIXX readable file are discussed below. The quantities found in the other output file are self explanatory and will not be discussed here.

The following scalar quantities are written to the MATRIXX file:

- \( \text{sigmag} \) — Gust intensity.
- \( \text{noutmx} \) — The output quantity to be maximized.
**deltat**  The time step.

**tmaximp**  The length of the impulse responses.

The following arrays can also written to the MATRIXX file:

**allkvals**  This vector has length nkvals, and stores all the k values used in the analysis.

**maxout**  The maximum output values for each k value are stored in each column. Thus, this array will be an nout by nkvals array.

**impres#**  An array of this form is created for each output quantity. For instance, for output 1 an array named "impres1" will be created and for output 5 an array named "impres5" will be created. Each column of these arrays are impulse response time histories for one of the k values. Thus, these are ntsteps by nkvalues arrays.

**wavef#**  An array of this form is created for the output to be maximized. For instance, if output 6 is to be maximized, then an array named "wavef6" will be created. Each column of this array is an excitation waveform matched to output # for a k value. Thus, this is an ntsteps by nkvalues array.

**exresp#**  An array of this form is created for each output quantity. For instance, for output 1 an array named "exrespl" will be created. Each column of this array is an excitation waveform response for a k value. Thus, these are (2*ntsteps+1) by nkvalues arrays.

**NUMERICAL EXAMPLE**

This section describes the step by step process of obtaining numerical results at a gust intensity of 1,530 in/sec.

**Step 1 - Create a subroutine containing the gust filter in series with the aircraft equations of motion.**

The nonlinear simulation model of the ARW-2 drone aircraft equipped with a nonlinear control system was used in this example. This model was connected in series with a transfer-function representation of atmospheric turbulence. The transfer function used in this example was (ref. 8)

\[
\frac{w_g}{\eta} = \sigma_g \sqrt{\frac{L}{\pi \nu \left[ 1 + 2.618(L/V)s \right] \left[ 1 + 0.1298(L/V)s \right]}}
\]

which approximates the square root of the von Karmon power spectral density function. The quantity \( \sigma_g \) is the intensity of the gust or standard deviation -- which, assuming zero mean, is also equal to the root-mean-square, or RMS, value -- of gust velocity.

Figure 4 shows a block diagram of the ARW-2 simulation model and includes the aeroelastic plant, a gust load alleviation (GLA) control law, and nonlinear control elements. The aeroelastic plant is a linear, s-plane aeroelastic half-model consisting of
two longitudinal rigid-body modes and three symmetric flexible modes. Unsteady aerodynamics were obtained using the doublet lattice method (ref. 9). The model also includes the dynamics of the control surface actuators. The two-input/two-output GLA control law was obtained using a Linear Quadratic Gaussian design approach with the intent of reducing wing root bending moment (ref. 10). The nonlinear elements impose deflection limits of $\pm 1^\circ$ on the elevators and $0^\circ$ to $+1^\circ$ on the ailerons to simulate spoilers. The model contains 32 states, and the analysis conditions are at a Mach number of 0.86 and an altitude of 24,000 feet.

The resulting combined system has a single input (white noise) and many outputs including the gust velocity ($y(14)$). The output quantities for this model are as follows:

- $y(1)-y(3)$ = No physical interpretation
- $y(4)$ = Tip acceleration
- $y(5)$ = Fuselage acceleration
- $y(6)$ = Wing root bending moment (WRBM)
- $y(7)$ = Wing root shear (WRS)
- $y(8)$ = Wing outboard bending moment (WOBBM)
- $y(9)$ = Wing outboard torsion moment (WOBTM)
- $y(10)$ = Elevator deflection
- $y(11)$ = Elevator rate
- $y(12)$ = Aileron deflection
- $y(13)$ = Aileron rate
- $y(14)$ = Gust velocity

Appendix C contains the FORTRAN version of the analytical model. This model was created using MATRIXX SYSTEM BUILD (ref. 11) and converted to FORTRAN using the HYPERCODE (ref. 12) package. As shown in Appendix C, the appropriate lines of code were added to the model created by the HYPERCODE package to make it compatible with the differential equation solving scheme used in MFB1DS.

**Step 2 - Create an input file for the model.**

The length of the state vector and number of model output quantities are contained in the input file MODEL.INP and must be made compatible with the analytical model created in step 1. Other input quantities like the gust intensity, identifying the output quantity to be maximized, and the range of $k$ values must also be selected. (Trial and error will be required to find the range of $k$ values where the maximum load value is obtained.) In addition to these parameters, the length of time for the impulse responses and the time step must be chosen. The length of the impulse responses should be only long enough to allow the output quantity to damp out to a relatively small value as shown in the example in figure 4 (a), while the time step should be chosen in accordance with the model response characteristics.

The input file used in the ARW-2 analysis is shown in Appendix D.

**Step 3 - Modify the parameters.**

Appendix B contains a listing of MFB1DS.INC and a description of the parameters and variables contained there. The parameters found in the file MFB1DS.INC and MODEL.F must be made large enough to accommodate the analytical model (step 1) and the input file (step 2). Computer storage can be minimized by making parameters "maxstates" and "maxout" equal to the minimum values required for a given analytical model.
Step 4 - Compile the code.

The following statement compiles the code on a SUN workstation.

```
fc77 MFBIDS.F MODEL.F LSODE.F INTUTILS.F
```

Step 5 - Execute the code.

Appendix D shows the initial input file used in the analysis of the ARW-2 model. The initial value of gust intensity was 1.530 in/sec (1.5×85 ft/sec). This initial value of gust intensity was chosen so that the nonlinearities would be invoked in this example. Nine k values were used ranging from 10 to 15,000. The output quantity to be maximized was y(6), wing root bending moment, WRBM. The standard output from this run is shown in Appendix E.

The following statement will run the code on a UNIX system.

```
a.out < MODEL.INP
```

Step 6 - Examine output.

Of the nine k values used in the analysis, all the intermediate results for three of the nine k values are shown in figure 5. Beginning with plot (a), the impulse responses for the three k values are shown. Plot (b) gives the corresponding excitation waveforms obtained from these impulse responses. Plot (c) shows the critical gust profiles and plot (d) shows the maximized load responses created by the gust input. Plot (e) depicts maximum WRBM as a function of initial impulse strength: the circles are actual numerical results; the solid line is a faired line. For this particular example, a value of k of about 2.410 creates an excitation waveform and critical gust profile that yields the largest maximized value of WRBM. 296,994 in-lbs.

Step 7 - Vary parameters.

The user may wish to refine the results shown in figure 5. With this in mind, a new range of k values was selected and the case was rerun. Figure 6 shows the results of using the new set of k values distributed between 400 and 6,000 producing a much smoother curve. The maximum load value obtained using the refined k distribution was 296,804 in-lbs at a k value of approximately 2,173. For this particular airplane and set of flight conditions a larger load value was not obtained with the new range of k values.

EXAMINATION OF RESULTS

This section has been included in the manual to provide the user with insight into how the results from MFBIDS should be interpreted. The results presented in the previous section will be shown along with results obtained using two additional gust intensities. To obtain results at different gust intensities, the sigma value of the input file shown in Appendix D was changed and MFBIDS was run. In addition, answers were obtained for each of the three gust intensities using a linearized version of the model. For comparison these linear answers will be plotted with the results from the nonlinear model. Note that when using a linear model, only one k value needs to be used because
the answer will not be a function of k. Consequently, the maximum load value plotted versus k is a horizontal line for a linear model.

The results for gust intensity values of 1,020 in/sec, 1,530 in/sec and 2,040 in/sec are shown in figure 7. Before proceeding, it is important to point out that results are problem dependent, and while these results are typical of all the aircraft examined by NASA, different aircraft with different nonlinearities may exhibit different trends. With this in mind, two important points will be made concerning the general trends exhibited in figure 7. First, regardless of the gust intensity value, the maximum load is constant for small values of k. Consequently, when searching for the maximum load, the range of interest for the k values can be limited on the low end. Second, the maximum load value decreases toward some relatively small value as k is made very large. Thus, the range of interest for k values can also be limited on the high end. What happens in between these two extremes depends on the specific gust intensity.

In figure 7 the maximum load values are constant for k less that 400. In plot 7(a) a significantly larger load value was not found when k was increased beyond 400, while for plots 7(b) and 7(c) a larger value was found. This indicates that the character of the results (i.e., the shape of the maximum load versus k plot) and the specific k value that produces the maximum load value are functions of gust intensity. But, the range of k values where the maximum will be found is generally limited, and in this case the range is roughly between 400 and 10,000. It should also be noted that the difference between answers from linear and nonlinear models is also a function of the gust intensity. Larger gust intensity values generally result in larger differences between answers from linear and nonlinear models. Here, this difference is 2% for the lowest gust intensity and 18% for the largest gust intensity.

CONCLUDING REMARKS

This manual has reviewed the theory behind the Matched-Filter-Based one-dimensional search procedure. The code that performs this procedure has been discussed and example numerical results were presented and interpreted. The code, MFBIDS, is available in a self contained form. It has all the required equation solvers and files necessary to run the example problem. The user is, however, encouraged to modify the existing code by inserting more efficient routines if available.
REFERENCES


Figure 1. MFB linear method signal flow diagram.
Figure 2. Nonlinear MFB signal flow diagram for the one-dimensional search.
Input
Model Parameters:
nstates, nout
Case Parameters:
sigma, kmin, kmax, nkvals
noutmx

Impulse responses at each k value
Multiply unit impulse by k value
Call subroutine SIMULATE for each k value

Create excitation waveforms
Compute energy for impulse responses
Reverse impulse responses in time and divide by SQRT(energy)

Obtain maximized output quantities
Call subroutine SIMULATE for each excitation waveform

Simulation
Uses LSODE and a user defined subroutine containing A/C equations of motion

Write output to file
Write to easily read ascii file and/or a MATRIXx readable file

Figure 3. MFB1DS solution procedure.
Figure 4. Nonlinear block diagram for the ARW-2 drone aircraft.
(a) Impulse responses.

(b) Excitation waveforms (WRBM time histories that have been normalized, reversed and multiplied by $\sigma_g$).

(c) Critical gust profiles.

(d) Wing root bending moment time histories.

(e) Maximum wing root bending moment.

Figure 5.- Example calculations illustrating the effects of $k$ variation on maximum WRBM. $\sigma_g = 1.530$ in/sec.
Figure 6: Maximum WRBM versus k using a refined range of k values. \( \sigma_g = 1.530 \) in/sec.
Maximum Wing Root Bending Moment, in-lb

(a) $\sigma_g = 1.020$ in/sec (85 ft/sec)

Maximum Wing Root Bending Moment, in-lb

(b) $\sigma_g = 1.530$ in/sec (1.5X85 ft/sec)

Maximum Wing Root Bending Moment, in-lb

(c) $\sigma_g = 2.040$ in/sec (2X85 ft/sec)

Figure 7.- Maximum WRBM versus k for 3 gust intensities.
Appendix A - MFB1DS.F source code listing

program MFB1DS

include 'MFB1DS.INC'
character*80 ctitle
character*80 ccase
common /case/ccase
common /title/ctitle
dimension yth(maxout,2*maxtsteps)

c **read input parameters
call DATAIN
write(6,*), ctitle
case

c **calculate number of time steps for impulse response
ntsteps=int(tmaximp/deltat + 0.001) + 1
write(6,**)
write(6,**) ' length of simulation for impulse responses = ', tmaximp
cntsteps
write(6,**) ' number of time steps for impulse responses = ', ntsteps
write(6,**)
write(6,**) ' length of simulation for excitation responses = ', 2*tmaximp
cntsteps
write(6,**) ' number of time steps for excitation responses = ', 2*ntsteps-1
write(6,**)
cntsteps
write(6,**) ' gust intensity = ', sigma
write(6,**)

c **create array of zeros for impulse input
do 100 i=1,(2*ntsteps-1)
100 aimpt(i)=0.0

c **calculate k values
do 200 k=1,nkvals
if (nkvals.eq. 1) then
   allkvals(1)=akmin
else
   allkvals(k)=10.0**( (k-1)*(log10(akmax)-log10(akmin))/
      float(nkvals-1)+log10(akmin) )
endif

200 continue

c **obtain impulse response for each k value
write(6,*') 'calculating impulse responses for each k value'
do 300 k=1,nkvals
   ak=allkvals(k)
c **assign impulse strength k to array aimpt
   aimpt(2)=ak/deltatl2
   aimpt(3)=ak/deltatl2
c **call subroutine SIMULATE to obtain impulse response
call SIMULATE(yth,aimpt,ntsteps)
do 300 iout=1,nout
do 300 j=1,ntsteps
300 aimpt_res(j,iout,k)=yth(ioutj)
c **calculate normalized excitation waveformns
write(6,*') 'calculating normalized excitation waveforms'
do 400 k=1,nkvals
   i=noutmx
   energy=0.0
   energy=aimpt_res(1,i,k)**2 + aimpt_res(ntsteps,i,k)**2
   do 401 j=2,ntsteps-1
501 energy = energy+2*(aimpt_res(j,i,k))*2
   energy = sqrt( tmaximp*energy*.5/float(ntsteps)/3.14159 )
   write(6,*') ' ', k, ' k = ', allkvals(k),
   ' sqrt(energy) of output', noutmx, ' = ', energy
   do 402 j=ntsteps,1,-1
502 wave(ntsteps-j+1,k)=sigma*aimpt_res(j,i,k)/energy
400 continue

c **obtain maximized responses
   Note that here the length of the simulation is 2*tmaximp and the
c number of time steps for maximized responses is 2*ntsteps-1
write(6,*') 'calculating maximized responses'
do 500 k=1,nkvals
   do 501 j=1,ntsteps
501 aimpt(j)=wave(j,k)
do 502 j=(ntsteps+1), (2*ntsteps-1)
502 aimpt(j)=0.0
   call SIMULATE(yth,aimpt, (2*ntsteps-1) )
do 503 j=1,(2*ntsteps-1)
do 503 i=1,nout
503 aexc_res(j,i,k)=yth(i,j)

A-2
write(6,*) ' ', k, ' k = ', allkvals(k),
+ ' maximum value of output',
+ noutmx, ' = ', aexc_res(ntsteps,noutmx,k)

500 continue

c **save data
write(6,*) 'saving data'
if (iouttype.eq.1.or.iouttype.eq.3) call SAVASCIIFORM
if (iouttype.eq.2.or.iouttype.eq.3) call SAVMATFORM

stop
end

C ------------------------------------------------------------------------------------
C SIMULATE: subroutine to obtain a time history for the a/c model I
C ------------------------------------------------------------------------------------

subroutine SIMULATE(yth,u,nsteps)
include 'MFB1DS.INC'
dimension yth(maxout,2*maxtsteps), u(2*maxtsteps)
double precision y(maxout)
double precision atol, rtol, xstate(maxstates)
double precision rwork(22*10*maxstates+(2*1+1)*maxstates)
double precision tstart,tend, u0,u1
integer neq,itol,itask,istate,iopt,lrw,liw,mf
integer iwork(20+maxstates)
common /eqsmotcom/y,tstart,tend,u0,u1

C **for guidance in selecting sizes for arrays rwork and iwork
C see source code LSODE.F

external EQSMOT

C **Setup input for LSODE
C see LSODE.F for explanation of these parameters
C
neq=nstates
tstart=starting time
tend=ending time
itol=1
rtol=1.0E-06
atol=1.0E-10
itask=1
istate=1
iopt=0
rwork=real work space
lrw=22*10*maxstates+(2*1+1)*maxstates
iwork=imaginary work space
liw=20+maxstates
jaceqs
mf=23

do 1 i=1,maxstates
1 xstate(i)=0.0
do 11 i=1,maxout
11    yth(i,1)=0.0

tstart=0.0
tend=0
    do 2 i=2,nsteps
    tend=tend+deltat
    u0=u(i-1)
    u1=u(i)
    call LSODE(EQSMOT,neq,xstate,tstart,tend,itol,rtol,atol,
       +     itask,istate,iopt,rwork,lrw,iwork,liw,jaceqs,mf)
    if (istate.ne.2) write(6,*) 'tstart = ', tstart,
       +    'tend = ', tend
    tstart=tend
    do 3 j=1,nout
3    yth(j,i)=y(j)
    continue
2    return
end

subroutine DATAIN
include 'MFB1DS.INC'
character*80 ctitle
character*80 ccase
character*15 cdata, cmatrixx
common /case/ccase
common /title/ctitle
common /fnames/cdata, cmatrixx
character cdummy*80

read(unit=5,fmt='(a80)') cdummy
read(unit=5,fmt='(a80)') ctitle

read(unit=5,fmt='(a80)') cdummy
read(unit=5,fmt='(a80)') ccase

read(unit=5,fmt='(a80)') cdummy
read(unit=5,fmt='(a80)') ccase

read(unit=5,fmt='(a80)') cdummy
read(unit=5,fmt='(a80)') ccase

read(unit=5,fmt='(a80)') cdummy
read(unit=5,fmt='(a80)') cdummy
read(unit=5,fmt='*') tmaximp, deltat
**read sigma, akmin, akmax, nkvals**
read(unit=5,fmt='(a80)') cdummy
read(unit=5,fmt='*') sigma, akmin, akmax, nkvals

**read iouttype, impres, iexres, iwave**
read(unit=5,fmt='(a80)') cdummy
read(unit=5,fmt='*') iouttype, impres, iexres, iwave

**read data file name**
read(unit=5,fmt='(a80)') cdta
read(unit=5,fmt='(a12)') cdata

**read matrixx file name**
read(unit=5,fmt='(a80)') cdummy
read(unit=5,fmt='(a12)') cmatrixx

close(unit=5)
return
eend

---

**SAVMATFORM: subroutine to write output to a MATRIXx readable file**

**SAVMATFORM**
include 'MFBIDS.INC'
dimension rtemp(2*maxtsteps,maxstates)
character*10 varname, cnout
character*15 cdata, cmatrixx
common /fnames/cdata, cmatrixx

open(unit=1, file=cmatrixx)
 rewind(1)

**write gust intensity**
rtemp(1,1)=sigma
 call MATSAV(1,'sigmag', 2*maxtsteps, 1, + 1, 0, rtemp, rtemp, '(1p2e24.15)')

**write k values for each case**
do 1 i=1,nkvals
  rtemp(i,1) = allkvals(i)
  call MATSAV(1,'kvals', 2*maxtsteps, nkvals, + 1, 0, rtemp, rtemp, '(1p2e24.15)')
1

**write which output quantity was "maximized"**
rtemp(1,1)=noutmx
 call MATSAV(1,'noutmx', 2*maxtsteps, 1, + 1, 0, rtemp, rtemp, '(1p2e24.15)')

**write time step**
rtemp(1,1)=deltat
 call MATSAV(1,'deltat', 2*maxtsteps, 1, + 1, 0, rtemp, rtemp, '(1p2e24.15)')
**write tmaximp

c	temp(1,1)=tmaximp
call MATSAV(1,'tmaximp', 2*maxtsteps, 1,
+ 1, 0, rtemp, rtemp, '(1p2e24.15)')

**write maximum output values

do 3 i=1,nout
do 3 k=1,nkvals

c	temp(k,i)=aexc_res(ntsteps,i,k)
call MATSAV(1,'maxout', 2*maxtsteps, nkvals,
+ nout, 0, rtemp, rtemp, '(1p2e24.15)')

**write impulse responses

if (impres.eq.1) then

do 4 i=1,nout

open(unit=50, status='SCRATCH')
write(unit=50,fmt= '(i3)') i
rewind(50)
ndigits=1
if (i.gt.9) ndigits=2
if (i.gt.99) ndigits=3
cnout=' 
read(unit=50,fmt=('a3')) cnout(1:ndigits)
close(50)

varname='impres'//cnout
do 5 k=1,nkvals
do 5 nt=1,ntsteps

c	temp(nt,k)=aimp_res(nt,i,k)
call MATSAV(1, varname, 2*maxtsteps, ntsteps,
+ nkvals, 0, rtemp, rtemp, '(1p2e24.15)')
4 continue
endif

**write excitation responses

if (iexcres.eq.1) then

do 6 i=1,nout

open(unit=50, status='SCRATCH')
write(unit=50,fmt= '(i3)') i
rewind(50)
ndigits=1
if (i.gt.9) ndigits=2
if (i.gt.99) ndigits=3
cnout=' 
read(unit=50,fmt=('a3')) cnout(1:ndigits)
close(50)
varname='exresp'//cnout

do 7 k=1,nkvals
do 7 nt=1,(2*ntsteps-1)

c	temp(nt,k)=aexc_res(nt,i,k)
call MATSAV(1, varname, 2*maxtsteps, (2*ntsteps-1),
    nkvals, 0, rtemp, rtemp, '(1p2e24.15)')
6    continue
endif

**write excitation waveforms**
if (iwave.eq.1) then
    open(unit=50, status='SCRATCH')
    write(unit=50,fmt='(i3)') noutmx
    rewind(50)
    ndigits=1
    if (noutmx.gt.9) ndigits=2
    if (noutmx.gt.99) ndigits=3
    cnout='wave'
    read(unit=50,fmt='(a3)') cnout(1:ndigits)
    close(50)
    varname='wave'/cnout
    do 9 k=1,nkvals
    do 9 nt=1,ntsteps
    rtemp(nt,k)=wave(nt*k)
    call MATSAV(1, varname, 2*maxtsteps, ntsteps,
        nkvals, 0, rtemp, rtemp, '(1p2e24.15)')
    9    continue
endif
    close(1)
    return
end

SAVASCIIIFORM: subroutine to write output to an scii file

subroutine SAVASCIIIFORM
include 'MFB1DS.INC'
character*80 ctitle
character*80 cease
character*15 cdata, cmatrixx
*    common /case/ccase
    common /title/ctitle
    common /fnames/cdata, cmatrixx
    open(unit=9, file=cdata)
    rewind(9)
    **write general information
    write(9,'(a80)') ctitle
    write(9,'(a80)') cease
    write(9,*)' sigma = ', sigma
    write(9,*)' matched output quantity = ', noutmx
    write(9,*)' tmaximp = ', tmaximp
    write(9,*)' deltat = ', deltat
**write maximum output values for each case**

```fortran
write(9,'')
write(9,'**************************')
write(9,'MAXIMIZED AND TIME CORRELATED MAX OUTPUT QUANTITIES')
write(9,'**************************')
do i=1,nout
    write(9,*)' output quantity = ', i
    write(9,*)' k value maximum output value'
    do k=1,nkvals
        write(9,*) allkvals(k),
        + aexc_res(ntsteps,i,k)
    continue
enddo
```

**write impulse responses**

```fortran
if (impres.eq.1) then
    do k=1,nkvals
        write(9,'')
        write(9,'**************************')
        write(9,'IMPULSE RESPONSES')
        write(9,'**************************')
        write(9,*) 'kvalue = ', allkvals(k),
        + ' maximized output quantity = ', noutmx
        do iout=1,nout
            write(9,*) 'output quantity = ', iout
            do j=1,ntsteps
                write(9,*) aimp_res(j,iout,k)
            continue
        endif
    continue
endif
```

**write excitation waveforms**

```fortran
if (iwave.eq.1) then
    do k=1,nkvals
        write(9,'')
        write(9,'**************************')
        write(9,'EXCITATION WAVEFORMS')
        write(9,'**************************')
        write(9,*) 'kvalue = ', allkvals(k),
        + ' output quantity = ', noutmx
        do j=1,ntsteps
            write(9,*) wave(j,k)
        continue
    endif
```

**write excitation responses**

```fortran
if (iexcres.eq.1) then
    do k=1,nkvals
        write(9,'')
        write(9,'**************************')
        write(9,'EXCITATION WAVEFORM RESPONSES')
        write(9,'**************************')
        write(9,*) 'kvalue = ', allkvals(k),
```
'maximized output quantity = ', noutmx
  do 4 iout=1,nout
    write(9,*) 'output quantity = ', iout
  do 4 j=1,2*ntsteps-1
    write(9,*) aexc_res(j,iout,k)
  continue
  endif
  close(9)
  return
end

C------------------------------------------------------------------------

subroutine MATSAV ( lunit, name, nr, m, n, img, xreal, ximag, formt)
C------------------------------------------------------------------------

MATSAV writes a matrix to a file in a format suitable for the matrixx load operation.

param. type on input- on output-
------------------------------------------------------------------------

lunit integer fortran logical unit number. unchanged.

name character*(*) name of the matrix. one alphanumeric characters.
  (maximum length 10) unchanged.

nr integer row-dimension in the defining dimension or type statement in the calling program. nr must be greater than or equal to m.

m integer number of rows of the matrix unchanged.

n integer number of columns of the matrix.

img integer if img = 0, the imaginary part (ximag) is assumed to be zero and is not saved.

xreal double precision real part of the matrix to be saved.

ximag double imaginary part of the matrix unchanged.
precision to be saved.

formt character*(*) string containing the fortran format to be used for writing the elements of the matrix.

--------------------------------------------------------------------

example: the following fortran program generates an elementary matrix in x and writes it to fortran unit 1. assume that unit 1 has been preallocated as file (data set) test.

dimension x(20,3), dummy
   do 200 j=1,3
      do 100 i=1,10
         x(i,j)=0.0d0
      100 continue
      x(j,j)=1.0d0
   200 continue
   call MATSAV( 1, 'amatrix', 20, 10, 3, 0, $ x, dummy, '(lp2e24.15)' )
   stop
   end

after this program runs, invoke matrixx and type:

<> load 'test'

this will put x on the stack as stack-variable-name amatrix.

--------------------------------------------------------------------

integer lunit, m, n, nr, img
character*(*) name, formt
dimension xreal(nr,1), ximag(nr,1)
character nam* 10, form*20

write header record.

write(lunit,'(a10,3i5,a20)') nam,m,n,img,form
write real-part of the matrix.
write(lunit,form) ((xreal(i,j),i=1,m),j=1,n)
write imaginary-part if nonzero.
if(img.ne.0) write(lunit,form) ((ximag(i,j),i=1,m),j=1,n)
return
end
Appendix B - MFB1DS.INC source code and description of parameters

```
parameter(maxtsteps=5001, maxstates=40, maxout=20, maxkvals=20)
common /modelinfo/nstates, nout
common /siminfo/tmaximp, deltat, ntsteps, nsubintvl
common /case_info/noutmx
common /excitation_info/sigma, akmin, akmax, nkvals,
   + allkvals(maxkvals)
common /t_hist/aimp_res(maxtsteps,maxout,maxkvals),
   + aexc_res(2*maxtsteps,maxout,maxkvals),
   + wave(maxtsteps,maxkvals),
   + aimpt(2*maxtsteps)
common /output/iouttype, impres, iexcres, iwave
```

The following is a description of the parameters:

- **maxtsteps**: This parameter is the maximum number of time steps for the impulse responses. \((\text{maxtsteps} \geq \text{tmaximp}/\text{deltat} + 1)\)

- **maxstates**: This parameter is the maximum number of states in the equations of motion. \((\text{maxstates} \geq \text{nstates})\)

- **maxout**: This the maximum number of output quantities required by the aircraft model. \((\text{maxout} \geq \text{nout})\)

- **maxkvals**: This the maximum number of kvalues which can be run.

To change these parameters edit MFB1DS.INC. In addition, maxout is also found in MODEL.F and must be equal that the value in MFB1DS.INC. Computer storage can be minimized by making maxstates and maxout equal to the minimum values required for a given analytical model.
Appendix C - MODELF (ARW-2) source code listing

```fortran
subroutine EQSMOT(neq,t,x,xd)
parameter(maxout=20)
double precision x(*), xd(*), y(maxout)
double precision t,tstart,tend, u0, u1
common /eqsmotcomly,tstart,tend,u0,u1

u=(t-tstart)*(u1-u0)/(tend-tstart)+u0

C----------------------------------
state-space system
2
C-- nlarw2.controller.1)
C - ss c--
y(1) = -5.244970395787696D-4*x(1) - 2.74700469140994399D-4*x(2) +
     + 1.42946149773359D-3*x(3) + 5.56258182659891597D-6*x(4)
y(2) = -3.2870447856514102D-3*x(1) - 6.743878267402364D-5*x(2) +
     + 9.87258135859253064D-3*x(3) +
     + 9.9055693932225104D-6*x(4)
C----------------------------------
C - u bounded limit
C-- nlarw2.aileron limiter.2)
y(3) = min( 0.0D0, max( -0.017449999999999999D0, y(2) ) )
y(3) = min( 0.017449999999999999D0, max(
y + -0.017449999999999999D0, y(2) ) )
C----------------------------------
state-space system
C - nlarw2.arw2 mod.3)
y(4) = -2846.473761074609D0*x(5) - 2699.576720018260D0*x(6) +
     + 6978.294110534196D0*x(7) + 2.1435029315500892D0*x(8) -
     + 22.3289763900712601D0*x(9) + 3.399062943714242D0*x(10) -
     + 9.13144768834393994D0*x(11) + 5.05173942454337099D0*x(12) +
     + 75.7538109829206405D0*x(13) -
     + 0.958821750950846804D0*x(14) -
     + 0.86024628746771303D0*x(15) - 78.958529074215004D0*x(16) -
     + 43.9221734285092698D0*x(17) +
     + 8.35298937508213202D0*x(18) - 0.958821750950846804D0*x(19) -
     - 0.86024628746771303D0*x(20) -
y(4) = y(4) - 78.958529074215004D0*x(21) +
     + 43.9221734285092698D0*x(22) + 8.35298937508213202D0*x(23) -
     - 1954.93624505099399D0*x(24) -
     - 2.43416702624021802D0*x(25) - 2597.51251185656298D0*x(26) -
     + 37578.7624927093302D0*x(27) -
     + 4.17208401208873203D0*x(28) + 3.0779209320968897D9*x(29) -
     - 41.063443973176799D0*x(35) +
     + 3.57316720948225297D-3*x(36)
y(5) = -42.166683238785903D0*x(5) + 16.36587051303695D0*x(6) +
     + 453.12958109553202D0*x(7) + 351.460438213513001D0*x(8) -
     - 0.599754637061799697D0*x(9) -
     - 4.63233083144964902D-2*x(10) - 1.520116146415901D-2*x(11) -
     - 6.86370728184497701D-2*x(12) +
     + 0.483698505531560102D0*x(13) -
     + 0.96916615596266099D0*x(14) +
     + 0.51999486945893497D0*x(15) + 1.26302828846864701D0*x(16) +
     - 1.74000709591828001D0*x(17) -
     + 1.189239579268164D-2*x(18) - 0.969166155996266099D0*x(19)
```

C-1
\[ y(5) = y(5) + 1.26302828846864701D0 \cdot x(21) - 1.189239579268164D-2 \cdot x(23) + 900.378460475793005D0 \cdot x(24) + 0.64379949067852503D0 \cdot x(25) + 1238.1470008882901D0 \cdot x(26) + 9.35961201178221493D-2 \cdot x(28) + 3.816088230193029497D0 \cdot x(35) + 2.32409992594539401D-5 \cdot x(36) \]

\[ y(6) = 6384.7719999999698D0 \cdot x(6) - 26281.1300000009001D0 \cdot x(7) - 44252.8600000000997D0 \cdot x(8) \]

\[ y(7) = 81.7875899999999092D0 \cdot x(6) - 1061.036D0 \cdot x(7) + 2228.21D0 \cdot x(8) \]

\[ y(8) = -231.7430000000000000D0 \cdot x(6) - 5791.6359999999894D0 \cdot x(7) - 398.1630718496296D0 \cdot x(8) \]

\[ y(9) = 28.7296599999999702D0 \cdot x(6) + 534.1022000000001204D0 \cdot x(7) + 6162.5810982906405D0 \cdot x(8) \]

\[ y(10) = x(24) \]

\[ y(11) = x(25) \]

\[ y(12) = x(27) \]

\[ y(13) = x(28) \]

\[ y(14) = x(35) \]

\[ y(15) = -2846.473761074609D0 \cdot x(5) - 2699.57672001826D0 \cdot x(6) + 6978.29411601534196D0 \cdot x(7) + 2.143502931500892D0 \cdot x(8) - 22.3289763900712609D0 \cdot x(9) + 3.399062943714224D0 \cdot x(10) + 9.131447688343999D0 \cdot x(11) + 5.0517394245437099D0 \cdot x(12) + 75.758109829206405D0 \cdot x(13) + 95882175950846804D0 \cdot x(14) + 860246287496771303D0 \cdot x(15) - 78.958529074215004D0 \cdot x(16) + 43.9221734285092698D0 \cdot x(17) + 8.35298937508213202D0 \cdot x(18) - 0.958821750950846804D0 \cdot x(19) + 860246287496771303D0 \cdot x(20) \]

\[ y(15) = y(15) - 78.958529074215004D0 \cdot x(21) + 43.9221734285092698D0 \cdot x(22) + 8.35298937508213202D0 \cdot x(23) - 1954.9362450509939D0 \cdot x(24) + 2.4341670262401802D0 \cdot x(25) - 2597.51251185656298D0 \cdot x(26) + 37578.7624927093302D0 \cdot x(27) + 4.172084041208873203D0 \cdot x(28) + 3.0779209320068897D9 \cdot x(29) + 41.0633443973176799D0 \cdot x(30) + 3.57316720948225297D-3 \cdot x(36) \]

\[ y(16) = -42.1666883238785903D0 \cdot x(5) + 16.36587051303695D0 \cdot x(6) + 453.129586109553202D0 \cdot x(7) + 351.460438213513001D0 \cdot x(8) - 0.599754637061799697D0 \cdot x(9) + 4.63233083144964902D-2 \cdot x(10) + 3.701161646415901D-2 \cdot x(11) + 6.86370728184497701D-2 \cdot x(12) + 0.483698505531560102D0 \cdot x(13) - 0.96916615599626609D0 \cdot x(14) + 0.519949869945893497D0 \cdot x(15) + 1.26302828846864701D0 \cdot x(16) - 1.74000709591828001D0 \cdot x(17) + 1.89239579268164D-2 \cdot x(18) - 0.96916615599626609D0 \cdot x(19) + 0.519949869945893497D0 \cdot x(20) \]

\[ y(16) = y(16) + 1.26302828846864701D0 \cdot x(21) - 1.74000709591828001D0 \cdot x(22) - 1.189239579268164D-2 \cdot x(23) + 900.378460475793005D0 \cdot x(24) + 900.378460475793005D0 \cdot x(25) + 0.64379949067852503D0 \cdot x(26) + 291.107672370571898D0 \cdot x(27) - 9.35961201178221493D-2 \cdot x(28) + 3.816088230193029497D0 \cdot x(35) + 2.32409992594539401D-5 \cdot x(36) \]
\[ + 0.643799490067852503D0 \times x(25) + 1238.14470008882901D0 \times x(26) + 291.107672370571898D0 \times x(27) - 9.35961201478221493D-2 \times x(28) + 0.652116307184986296D0 \times x(35) + 2.32409992594539401D-5 \times x(36) \]

\[ c \text{----------------------------------} \text{1 - u bounded limit} \]

\[ c \text{-- (nlarw2.evel limiter.4)} \]
\[ y(17) = \min((0.01744999999999999D0, \max( -0.01744999999999999D0, y(1))) \]

\[ c \text{---------------------------------- state-space system} \]

\[ c \text{-- (nlarw2.controller.1)} \]
\[ c \text{--ss ab --} \]
\[ x(1) = -27.7971207546241899D0 \times x(1) - 25.4156325927014D0 \times x(2) + 0.966874297031381502D0 \times x(3) - 8.71552238769890408D-2 \times x(4) - 9.51654087303377394D-3 \times y(4) + 1.68032371405914D-2 \times y(5) \]
\[ x(2) = -6.8760262263934302D0 \times x(1) - 7.36711056293705599D0 \times x(2) - 8.27135760534558596D-3 \times x(3) + 0.976139521572193303D0 \times x(4) + 2.63798849248624001D-3 \times y(4) + 1.501039866773363D-2 \times y(5) \]
\[ x(3) = -52.3804248827116199D0 \times x(1) - 35.899410486724791D0 \times x(2) - 0.290086695207493903D0 \times x(3) - 0.218148213265887D0 \times x(4) - 7.09094969390322604D-3 \times y(4) - 9.45879838013578680D-3 \times y(5) \]
\[ x(4) = -4067.666992244+397999D0 \times x(1) - 5071.3341736882870D0 \times x(2) - 5.043376280250499D0 \times x(3) - 17.025381968938801D0 \times x(4) - 1.15740514598378101D-2 \times y(4) + 1.73620839739127D-2 \times y(5) \]

\[ c \text{---------------------------------- state-space system} \]

\[ c \text{-- (nlarw2.arw2 moD.3)} \]
\[ c \text{--ss ab --} \]
\[ x(5) = x(10) \]
\[ x(6) = x(11) \]
\[ x(7) = x(12) \]
\[ x(8) = x(13) \]
\[ x(9) = -125.245923867737901D0 \times x(5) + 45.0335041319353904D0 \times x(6) - 117.872130364775799D0 \times x(7) + 3906.63061647083703D0 \times x(8) - 1.26414087776252901D0 \times x(9) - 0.1273297457868012D0 \times x(10) - 0.2001582942733044D0 \times x(11) + 0.1463652541766454D0 \times x(12) + 1.15969827063154399D0 \times x(13) - 0.969152580431533295D0 \times x(14) + 8.277525390783397D-4 \times x(15) + 6.043889655949902D0 \times x(16) + 6.85614516592095201D-3 \times x(17) + 5.01807022310764295D-4 \times x(18) - 0.969152580431533295D0 \times x(19) + 8.277525390783397D-4 \times x(20) \]
\[ x(9) = x(9) + 6.043889655949902D0 \times x(21) + 6.85614516592095201D-3 \times x(22) + 5.01807022310764295D-4 \times x(23) + 1223.06162596226201D0 \times x(24) + 0.899244109822731702D0 \times x(25) + 1763.75969191059301D0 \times x(26) + 698.161055553561D0 \times x(27) - 0.12642793532889D0 \times x(28) + 6.71666950223660506D7 \times x(29) - 1.867833656108772D0 \times x(35) + 7.65562322571364598D-5 \times x(36) \]

C-3
\begin{align*}
+ & - 4.84533115551499498D-4 \epsilon \\
\text{xd}(10) &= - 31.797364346982102D0 \epsilon x(5) - 16.6021798016908D0 \epsilon x(6) \\
+ & - 135.2985761432064D0 \epsilon x(7) + 2147.94963514691301D0 \epsilon x(8) \\
+ & - 1.2486797397536309D0 \epsilon x(9) - 0.2655459480335534D0 \epsilon x(10) \\
+ & + 0.15325792353896599D0 \epsilon x(11) - 1.050858874417987D-2 \epsilon x(12) \\
+ & + 1.2782272593746799D0 \epsilon x(13) - 1.27530495555126799D-3 \epsilon x(14) \\
+ & - 5.7912513541229769D0 \epsilon x(15) + 2.918827846093392D-3 \epsilon x(16) - 1.20106974970340501D-3 \epsilon x(17) \\
+ & + 3.667313377607812D-3 \epsilon x(18) - 1.2753049555126799D-3 \epsilon x(19) - 5.7912513541229769D0 \epsilon x(20) \\
+ & + 2.918827846093392D-3 \epsilon x(21) \\
\text{xd}(10) &= \text{xd}(10) - 1.20106974970340501D-3 \epsilon x(22) + 3.667313377607812D-3 \epsilon x(23) + 2467.34662588543097D0 \epsilon x(24) \\
+ & + 3.51373580753656301D0 \epsilon x(25) + 6539.00920066869003D0 \epsilon x(26) + 779.8473335337912D0 \epsilon x(27) - 0.1006533599710573D0 \epsilon x(28) + 5.69431331591210403D7 \epsilon x(29) \\
- & - 5.308857313745769910D0 \epsilon x(35) + 6.99578120780643495D-5 \epsilon x(36) - 4.427709625193978D-4 \epsilon \\
\text{xd}(11) &= - 4035.453860038237D0 \epsilon x(5) - 5039.8049881928803D0 \epsilon x(6) - 10315.3494347062669D0 \epsilon x(7) + 2.42234482371904D5 \epsilon x(8) - 27.9029350865744909D0 \epsilon x(9) - 5.00483961999364602D0 \epsilon x(10) \\
- & - 16.919934275288030D0 \epsilon x(11) + 1.21745740818824499D0 \epsilon x(12) + 75.014420843864629D0 \epsilon x(13) + 8.243002362753327083D0 \epsilon x(14) + 1.93965721381322799D-3 \epsilon x(15) - 152.1134291940743D0 \epsilon x(16) \\
+ & + 5.80386541425226303D-2 \epsilon x(17) + 5.2523764977082796D-2 \epsilon x(18) + 8.243002362753327083D-3 \epsilon x(19) + 1.93965721381322799D-3 \epsilon x(20) \\
\text{xd}(11) &= \text{xd}(11) - 152.1134291940743D0 \epsilon x(21) - 5.80386541425226303D-2 \epsilon x(22) + 5.2523764977082796D-2 \epsilon x(23) - 3224.495306375349D0 \epsilon x(24) \\
- & - 3.66154656177599203D0 \epsilon x(25) - 3889.28797354870801D0 \epsilon x(26) + 70121.2766765817105D0 \epsilon x(27) - 16.2309588268850602D0 \epsilon x(28) + 8.24312132121215198D9 \epsilon x(29) - 58.8147948818334498D0 \epsilon x(35) + 6.03774882724283902D-3 \epsilon x(36) - 3.82136001724231499D-2 \epsilon \\
\text{xd}(12) &= 3871.75335549381299D0 \epsilon x(5) + 5135.837069757428D0 \epsilon x(6) - 34825.3167988588102D0 \epsilon x(7) - 1.03428353574103699D5 \epsilon x(8) \\
+ & + 26.29546899718999098D0 \epsilon x(9) + 1.250989170291227D0 \epsilon x(10) + 2.630979503694448D0 \epsilon x(11) - 18.3240096889994701D0 \epsilon x(12) \\
+ & + 9.72678577610167805D0 \epsilon x(13) + 4.20112078815981804D-3 \epsilon x(14) + 1.35305651065170701D-3 \epsilon x(15) + 4.06376012479173199D-3 \epsilon x(16) - 140.368755107959299D0 \epsilon x(17) \\
+ & + 0.141684134431355902D0 \epsilon x(18) + 4.20112078815981804D-3 \epsilon x(19) + 1.35305651065170701D-3 \epsilon x(20) \\
\text{xd}(12) &= \text{xd}(12) + 4.06376012479173199D-3 \epsilon x(21) - 140.368755107959299D0 \epsilon x(22) + 0.141684134431355902D0 \epsilon x(23) \\
+ & + 2449.1224831598669D0 \epsilon x(24) + 2.2052998184531003D0 \epsilon x(25) + 1948.39110573395001D0 \epsilon x(26) \\
+ & + 20503.2354698112499D0 \epsilon x(27) -
\[
\begin{align*}
8.71772134128349308D0*x(28) + 3.55976045603601098D9*x(29) + & 
56.3946108862526199D0*x(35) - \\
3.6337807673383630D2-3*x(36) + 2.29986124515086901D-2 & 
\times 10 - 0.555117409039766599D0*x(11) + \\
0.16710154453369159D0*x(12) - 29.8328022920818499D0*x(13) + \\
- 3.81290566534986402D-3 & 
\times 10 - 0.1002548480818097D0*x(16) + \\
+ 9.994206021040908D-2 & 
\times 10 - 6.49269676792184203D0*x(18) + \\
- 3.81290566534986402D-3 & 
\times 10 - 0.578150706319158894D-4* & 
\times 10 - 0.1002548480818097D0*x(21) + \\
+ 9.994206021040908D-2 & 
\times 10 - 6.49269676792184203D0*x(23) + \\
+ 1019.796982139836D0*x(24) + 0.97690500705910916D0*x(25) + \\
+ 2093.94171462465602D0*x(26) + \\
5232.109080639069D0*x(27) - 1.296059064581286D0*x(28) + \\
+ 1.28570190987768D8*x(29) - 7.11996318155942698D0*x(35) + \\
+ 6.1637295081372805D-4 & 
\times 36 - 3.90109462353402403D-3 & 
\times 10 - 0.1002548480818097D0*x(11) + \\
- 98.356099999999694D0*x(12) + 1044.980000000003D0*x(13) + \\
- 84.2542655304641794D0*x(14) + \\
+ 146.225000000001399D0*x(25) + 158.84699999999799D0*x(28) + \\
+ 8.527042966739183977D-2 & 
\times 35 + \\
+ 4.0655937683870898D-2 & 
\times 36 - 0.2573160612904246D0*u & 
\times 10 - 0.2573160612904246D0*u - \\
+ 5.66919999999999D0*x(10) + 10.00299999999989D0*x(11) + \\
- 29.1312000000000040D0*x(12) - \\
+ 92.856099999999694D0*x(13) - 84.2542655304641794D0*x(15) + \\
- 111.353000000000099D0*x(25) + \\
+ 17.204399999999602D0*x(28) + 4.65810662194168197D-2 & 
\times 35 + \\
+ 2.2093043608741999D0-2 & 
\times 36 - 0.140565217473887401D0*u & 
\times 10 - 0.140565217473887401D0*u - \\
+ 6.3020900000000295D0*x(10) - 5.168499999999949D0*x(11) + \\
+ 2.076999999999801D0*x(12) + \\
+ 205.77099999999699D0*x(13) - 84.2542655304641794D0*x(16) + \\
+ 19.06519999999999D0*x(25) + \\
- 59.31069999999998D0*x(28) + 3.427498257283779D-4 & 
\times 35 + \\
+ 1.634190845563998D-4 & 
\times 36 - 1.0342980003521519D-3 & 
\times 10 - 0.744481999999993D0*x(9) + \\
+ 6.8934700000000798D0*x(10) + 8.56765000000001509D0*x(11) + \\
+ 2.0852D0*x(12) - 145.5889999999899D0*x(13) - \\
+ 84.2542655304641794D0*x(17) - 15.72730000000001D0*x(25) - \\
+ 32.450399999999504D0*x(28) + 3.19475760574534596D-2 & 
\times 35 + \\
+ 1.52322841035782D-2 & 
\times 36 - 9.64065089263157499D-2 & 
\times 2*u & 
\times 10 - 13.323899999999801D0*x(9) - \\
+ 18.165200000000302D0*x(10) - 118.12700000000001D0*x(11) + \\
+ 134.61499999999799D0*x(12) - \\
+ 949.75400000000801D0*x(13) - 84.2542655304641794D0*x(18) \end{align*}
\]
\[
\begin{align*}
+ & + 3.1790500000000000398D0\times(x(25)) - \\
+ & 397.355999999999803D0\times(x(28)) + 0.2788932280042591D0\times(x(35)) - \\
+ & + 0.1329730100156308D0\times(x(36)) - \\
+ & 0.846101329212856797D0\times(u) - \\
+ & + 82.22080000000000503D0\times(x(9)) + \\
+ & 38.1446000000000804D0\times(x(10)) - 9.0163099999997607D0\times(x(11)) + \\
+ & + 31.3315999999999799D0\times(x(12)) + \\
+ & + 656.121999999999403D0\times(x(13)) - 168.6879863655722D0\times(x(19)) + \\
+ & + 431.7630000000000801D0\times(x(25)) + 123.543000000000101D0\times(x(28)) + \\
+ & + 0.783519507974698798D0\times(x(35)) + \\
+ & + 0.132973010015630801D0\times(x(36)) - \\
+ & - 0.841601329212856797D0\times(x(9)) - \\
+ & + 29.27620000000000202D0\times(x(10)) - 12.371699999997997D0\times(x(11)) + \\
+ & + 20.67390000000000002D0\times(x(12)) + \\
+ & + 247.386999999999698D0\times(x(13)) - 168.6879863655722D0\times(x(20)) + \\
+ & + 264.54900000000000002D0\times(x(25)) + 26.5588999999999902D0\times(x(28)) + \\
+ & + 1.00638907302692501D-2\times(x(35)) + \\
+ & + 4.79834470147791304D-3\times(x(36)) - 3.036927026251846D-2\times(u) - \\
+ & + 22.5940000000000000201D0\times(x(10)) - \\
+ & - 5.37708000000000697D0\times(x(9)) + \\
+ & + 8.38994999999999891D0\times(x(11)) - 18.90300000000000002D0\times(x(12)) + \\
+ & + 142.49300000000000002D0\times(x(13)) - 168.6879863655722D0\times(x(21)) + \\
+ & - 22.19690000000000002D0\times(x(25)) + \\
+ & + 168.935999999999702D0\times(x(28)) - 0.223846017964695498D0\times(x(35)) + \\
+ & + 0.10672714788302501D0\times(x(36)) - \\
+ & + 0.675488277740527096D0\times(u) - \\
+ & + 4.776639999999998601D0\times(x(10)) - 12.28980000000000002D0\times(x(11)) + \\
+ & + 13.8938999999999999D0\times(x(12)) - 266.941999999999098D0\times(x(13)) + \\
+ & - 168.6879863655722D0\times(x(22)) + 19.1595999999995D0\times(x(25)) - \\
+ & + 43.06590000000000596D0\times(x(28)) - 0.231364769970956501D0\times(x(35)) + \\
+ & - 0.11032000000000009681901D0\times(x(36)) + \\
+ & + 0.698177217618241003D0\times(u) - \\
+ & + 150.0799999999998999D0\times(x(9)) - \\
+ & + 14.4340999999999999D0\times(x(10)) + 184.73899999999599D0\times(x(11)) - \\
+ & - 262.15300000000000198D0\times(x(12)) + \\
+ & + 2774.57000000000698D0\times(x(13)) - 168.6879863655722D0\times(x(23)) + \\
+ & + 28.31230000000000502D0\times(x(25)) + 935.65599999998997D0\times(x(28)) + \\
+ & + 0.3023539299912410402D0\times(x(35)) + \\
+ & + 0.144158796714765301D0\times(x(36)) - 0.91239744756180406D0\times(u) - \\
+ & + x(25) - \\
+ & 3.94784179999999702D5\times(x(24)) - \\
+ & + 1256.63700000000019D0\times(x(25)) + 7.8956835999999404D6\times(x(26)) - \\
+ & + 20.0D0\times(x(26)) + y(17) - \\
+ & + x(28) - \\
+ & + 3.348D5\times(x(27)) - 818.400000000001498D0\times(x(28)) + \\
+ & + 7.201548D+11\times(x(29)) - \\
+ & + x(30) - \\
+ & + 2.151D6\times(x(29)) - 540.09999999998502D0\times(x(30)) + \\
+ & + 2.191D5\times(x(31)) - \\
+ & + x(32) - \\
+ & + 2.191D5\times(x(31)) - 185.300000000002D0\times(x(32)) + \\
+ & + 3.742D6\times(x(33)) - \\
+ & + x(34) - \\
+ & + 3.742D6\times(x(33)) - 1446.5D0\times(x(34)) + y(3)
\end{align*}
\]

C-6
\[
xd(35) = -0.3399999999999999D0 \cdot x(35) - \\
+ 0.162107999999999901D0 \cdot x(36) + 1.02600000000000299D0 \cdot u \\
xd(36) = -0.359999999999999397D0 \cdot x(36) + u
\]

C-----------------------------------------------------
C
.
return
end
Appendix D - MODEL.INP file for the ARW-2 model

This is an example of the input file, MODEL.INP, that must be created by the user. The odd numbered lines are dummy variables used to name the fields and the even numbered lines contain the actual data. This is the input file used to generate the results shown in figure 5.

model
arw2 flexible nonlinear model, mach .86, altitude 24k ft
case
maximize output #6
nstates nout noutmx
36 17 6
tmaximp deltat
10.0 0.005
sigma akmin akmax nkvals
1530.0 10.0 15000.0 9
iouttype impres iexcre iwave
3 1 1 1
standard filename (iouttype=1)
arw1530.out
matrixx filename (iouttype=2)
arw1530.mat

Where these input quantities are defined:

model A character variable describing the model used in the analysis
case A character variable describing the case
nstates Number of states in the model
nout Number of output quantities used in the model
tmaximp The length of the simulation used in the impulse responses
deltat The time step used in the simulations
noutmx The number of the output quantity to be maximized
sigma The gust intensity used in the analysis, units of velocity
akmin The minimum k value to be used in the analysis
akmax The maximum k value to be used in the analysis
nkvals The total number of k values to be used in the analysis

ioouttype Specifies the format for the output
1, Normal ascii file
2, MATRIXx readable ascii file
3, Both 1 and 2
impres If 1, write the impulse responses to the output file(s)
iexcre If 1, write the excitation waveform responses to the output file(s)
iwave If 1, write the excitation waveform(s) to the output file(s)
Appendix E - A sample listing of the MFB1DS program output using the input file shown in appendix D and the model listed in Appendix C

arw2 flexible nonlinear model, Mach .86, Altitude 24k ft
maximize output #6

length of simulation for impulse responses = 10,000000
number of time steps for impulse responses = 2001

length of simulation for excitation responses = 20,00000
number of time steps for excitation responses = 4001

gust intensity = 1530.00

calculating impulse responses for each k value

calculating normalized excitation waveforms

1 k = 10.000000 sqrt(energy) of output 6 = 568.177
2 k = 24.9466 sqrt(energy) of output 6 = 1417.29
3 k = 62.2333 sqrt(energy) of output 6 = 3536.37
4 k = 155.251 sqrt(energy) of output 6 = 8820.35
5 k = 387.298 sqrt(energy) of output 6 = 22003.6
6 k = 966.177 sqrt(energy) of output 6 = 56134.6
7 k = 2410.28 sqrt(energy) of output 6 = 162952.
8 k = 6012.84 sqrt(energy) of output 6 = 509979.
9 k = 15000.0 sqrt(energy) of output 6 = 1.49411e+06

calculating maximized responses

1 k = 10.000000 maximum value of output 6 = 287000.
2 k = 24.9466 maximum value of output 6 = 286965.
3 k = 62.2333 maximum value of output 6 = 286988.
4 k = 155.251 maximum value of output 6 = 286997.
5 k = 387.298 maximum value of output 6 = 287025.
6 k = 966.177 maximum value of output 6 = 289885.
7 k = 2410.28 maximum value of output 6 = 296994.
8 k = 6012.84 maximum value of output 6 = 279944.
9 k = 15000.0 maximum value of output 6 = 249730.

saying data