DEVELOPMENT OF A ROTORDYNAMIC SIGNAL PROCESSING MATLAB INTERFACE
AND A TWO-DISK ROTOR MODEL

A Thesis
presented to
the Faculty of California Polytechnic State University,
San Luis Obispo

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in Mechanical Engineering

by
David Lewis Baker
December 2017
COMMITTEE MEMBERSHIP

TITLE: Development of a Rotodynamic Signal Processing MATLAB Interface and a Two-Disk Rotor Model

AUTHOR: David Lewis Baker

DATE SUBMITTED: December 2017

COMMITTEE CHAIR: Xi Wu, Ph.D.
Professor of Mechanical Engineering

COMMITTEE MEMBER: Hemanth Porumamilla, Ph.D.
Associate Professor of Mechanical Engineering

COMMITTEE MEMBER: Eltahry Elghandour, Ph.D.
Associate Professor of Mechanical Engineering
ABSTRACT

Development of a Rotordynamic Signal Processing MATLAB Interface and a Two-Disk Rotor Model

David Lewis Baker

Using MATLAB and a National Instruments data acquisition card, a signal processing program meant to monitor the behavior of rotordynamic systems in real-time was developed and tested. By using traditional analysis methods in this field of engineering, commonly desired data representations such as bode, polar, orbit, full spectrum plots were able to be produced to a very high accuracy. Additional capabilities offered by this application are slow roll compensation, synchronous and sub-synchronous filtering, and true three dimensional plotting. The verification of this program was done by comparing the results to the ones acquired with Bently Nevada’s “Automated Diagnostics for Rotating Equipment” (ADRE) system. In addition to a data acquisition program, theoretical models of the two-disk rotor were created to estimate the unknown physical parameters of the system. By simulating the rotor with and without gyroscopic effects included, estimates for the stiffness, damping, eccentricity, initial phase, and initial skew values present in the system were determined.

Key words: real-time data acquisition, two-disk rotor, gyroscopic effects, full spectrum plots
ACKNOWLEDGMENTS

I would like to acknowledge the Donald E. Bently Center for Engineering Innovation at California Polytechnic State University San Luis Obispo for support of this work.

I would like to thank the members of my thesis committee, whose guidance throughout my time at Cal Poly has made this paper possible.

I would like to thank my parents for their endless support in all of my academic endeavors.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIST OF TABLES</td>
<td>viii</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>ix</td>
</tr>
<tr>
<td>1. INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Background</td>
<td>1</td>
</tr>
<tr>
<td>2. BENTLY NEVADA’S ADRE SYSTEM AND ROTOR KIT</td>
<td>3</td>
</tr>
<tr>
<td>2.1 Introduction to Bently Nevada’s Products</td>
<td>3</td>
</tr>
<tr>
<td>2.2 Rotor Kit Description</td>
<td>3</td>
</tr>
<tr>
<td>2.3 ADRE Application Description</td>
<td>8</td>
</tr>
<tr>
<td>3. MATLAB SIGNAL PROCESSING PROGRAM</td>
<td>17</td>
</tr>
<tr>
<td>3.1 Goals of the project</td>
<td>17</td>
</tr>
<tr>
<td>3.2 Explanation and capabilities of DAQ card</td>
<td>18</td>
</tr>
<tr>
<td>3.3 Explanation of MATLAB code</td>
<td>19</td>
</tr>
<tr>
<td>3.4 Explanation of MATLAB app</td>
<td>31</td>
</tr>
<tr>
<td>4. COMPARISON BETWEEN ADRE SOFTWARE AND MATLAB SIGNAL PROCESSING PROGRAM</td>
<td>34</td>
</tr>
<tr>
<td>4.1 Software and hardware comparison</td>
<td>34</td>
</tr>
<tr>
<td>4.2 Experimental Results Comparison</td>
<td>35</td>
</tr>
</tbody>
</table>
5. DEVELOPMENT OF THEORETICAL MODEL FOR TWO DISK SYSTEM AND

EXPERIMENTAL COMPARISON.............................................................................. 45

5.1 Overview of theoretical models used .......................................................... 45

5.2 Two-Disk Theoretical Model without Gyroscopic Forces .......................... 45

5.2.1 Flexibility Matrix Derivation: Flexible Shaft........................................... 46

5.2.2 Flexibility Matrix Derivation: Flexible Bearings ....................................... 49

5.3 Two-Disk Theoretical Model with Gyroscopic Forces ............................... 54

5.4 Identifying system parameters from experimental data ............................. 56

6. DISCUSSION AND CONCLUSIONS .......................................................... 70

REFERENCES .................................................................................................... 72

APPENDICES

A....................................................................................................................... 74

B....................................................................................................................... 84
## LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Physical geometry and material descriptions of single disk rotor system.</td>
<td>36</td>
</tr>
<tr>
<td>2. Physical geometry and material descriptions of two-disk rotor system.</td>
<td>57</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Bently Nevada model RK 4 rotor kit.</td>
<td>4</td>
</tr>
<tr>
<td>2. Close-up of mounting locations on a Bently Nevada rotor kit.</td>
<td>4</td>
</tr>
<tr>
<td>3. XY probe mounting block on Bently Nevada rotor kit, showing proper transducer installation.</td>
<td>5</td>
</tr>
<tr>
<td>4. Bently Nevada rotor kit speed control box.</td>
<td>6</td>
</tr>
<tr>
<td>5. Back of Bently Nevada Proximitor Assembly.</td>
<td>7</td>
</tr>
<tr>
<td>6. Plotting options for Bently Nevada ADRE for Windows software.</td>
<td>8</td>
</tr>
<tr>
<td>7. Polar plot configuration window in Bently Nevada ADRE for Windows software.</td>
<td>9</td>
</tr>
<tr>
<td>8. Configuration menu on Bently Nevada ADRE for Windows software.</td>
<td>10</td>
</tr>
<tr>
<td>9. Channel configuration in Bently Nevada ADRE for Windows software.</td>
<td>11</td>
</tr>
<tr>
<td>10. Transducer configuration window on Bently Nevada ADRE for Windows software.</td>
<td>12</td>
</tr>
<tr>
<td>11. Keyphasor configuration window in Bently Nevada ADRE for Windows software.</td>
<td>13</td>
</tr>
<tr>
<td>12. Trigger event configuration window in Bently Nevada ADRE for Windows software.</td>
<td>14</td>
</tr>
<tr>
<td>13. Sample mode configuration window in Bently Nevada ADRE for Windows software.</td>
<td>15</td>
</tr>
<tr>
<td>14. Spectrum sample configuration window in Bently Nevada ADRE for Windows software.</td>
<td>16</td>
</tr>
<tr>
<td>15. National Instruments cDAQ-9174 module installed with two NI 9215 cards.</td>
<td>18</td>
</tr>
<tr>
<td>16. Speed calculation model in Simulink for rotor kit.</td>
<td>21</td>
</tr>
<tr>
<td>17. Collapsed class definition of TestObject in MATLAB.</td>
<td>23</td>
</tr>
<tr>
<td>18. Screenshot of “startDAQ” function in MATLAB code.</td>
<td>25</td>
</tr>
<tr>
<td>19. DC offset and concatenation process of “plotData” function.</td>
<td>26</td>
</tr>
</tbody>
</table>
20. Speed and amplitude calculations in MATLAB code. ................................................................. 27

21. Rotor kit MATLAB application designed using “App Designer” tool. ................................. 32

22. Set-up of Bently Nevada rotor kit for single-disk experiment to compare ADRE for Windows and MATLAB DAQ systems. ......................................................................................................................... 35

23. Amplitude and phase lag results from the ADRE for Windows application and MATLAB program recording the horizontal transducer for a single-disk rotor kit ramp-up experiment.................................................................................................................................................... 37

24. Amplitude and phase lag results with slow roll compensation from the ADRE for Windows application and MATLAB program recording the horizontal transducer for a single-disk rotor kit ramp-up experiment. ................................................................................................................................. 39

25. Polar plot from the ADRE for Windows application displaying amplitude with respect to phase of the vertical transducer for a single-disk rotor kit ramp-up experiment. ...................... 40

26. Polar plot from the MATLAB application displaying amplitude with respect to phase of the horizontal transducer for a single-disk rotor kit ramp-up experiment. ......................... 41

27. Full spectrum cascade plot from the ADRE for Windows application displaying amplitude with respect to the frequency of vibration and running speed for a single-disk rotor kit ramp-up experiment................................................................................................................................. 43

28. Full spectrum cascade plot from the MATLAB application displaying amplitude with respect to the frequency of vibration and running speed for a single-disk rotor kit ramp-up experiment................................................................................................................................. 44

29. Two-disk schematic diagram for flexible shaft stiffness derivation........................................ 47

30. Two-disk schematic diagram for flexible bearing stiffness derivation.................................... 49
31. MATLAB input file used to construct the finite element model of a two-disk rotor with gyroscopic effects and internal damping ........................................ 55

32. Experimental set up of two-disk, Bently Nevada rotor kit .................................. 56

33. Experimental horizontal and vertical amplitude response plots from two-disk ramp up experiment .................................................................................. 58

34. 3D plot illustrating the effect that varying $K_X/K_Y$ ratios have on the responses of the horizontal and vertical displacements .................................................. 59

35. Horizontal displacement results from a two-disk ramp-up experiment and two-disk theoretical model without gyroscopic forces ........................................ 61

36. Phase results from a two-disk ramp-up experiment and two-disk theoretical model without gyroscopic forces ................................................................. 62

37. Experimental 3D cascade plot generated from the data of disk 2 on the two-disk rotor ...... 63

38. Theoretical 3D cascade plot generated from model without gyroscopic effects ............... 64

39. Horizontal displacement results from a two-disk ramp-up experiment and two-disk theoretical model with gyroscopic forces .................................................. 66

40. First and second mode shapes for a beam between two fixed points (Turbomachinery Balancing Basics, 2008) ................................................................. 67

41. MATLAB generated experimental 3D orbit plot ..................................................... 68

42. Theoretical 3D orbit plot without gyroscopic effects .................................................. 69
1. INTRODUCTION

1.1 Background

The study of rotordynamics is a major focus in the field of engineering because of its prevalence in modern technology. Companies that use and develop any sort of rotating machinery such as engines, turbines, and compressors rely on extremely accurate monitoring systems and data processing software for their products to function properly. As a result, measurement systems made for this type of equipment are in high demand and come at a very large price. Data acquisition products such as Bently Nevada’s “Automated Diagnostics for Rotating Equipment” (ADRE) system and Agilis’ c360 Vibration Intelligence Software are examples of the industry’s standards when it comes to this subject. With knowledge of basic rotordynamic analysis and methods of obtaining commonly desired data, the computations performed by these software packages is well known (Bently, 2002; Goldman & Muszynska, 1999). After researching the possibility of developing a user friendly signal processing application, a low-cost alternative to commercially available data acquisition systems was proposed.

Rotational speed, phase angle, and displacements of a rotordynamic system are the main pieces of data that are needed to analyze the behavior of rotating machinery. This data can be displayed through several types of plots in order to convey different types of information. The ADRE application has the ability to produce amplitude, phase, orbit, and full spectrum plots all in real-time. This allows the operator to be fully aware of any problems that arise in a system. In addition to real-time monitoring, the ability to work with data after a test has been completed allows for more in depth analysis and diagnosing of rotordynamic phenomena. The ADRE
application includes this post-processing functionality, but it will be shown that this can be improved upon. The ADRE system’s capabilities will be the focus of this paper as a benchmark for this alternative data acquisition system.

Instances of research based data acquisition methods in rotordynamics have been found, but these do not discuss their application and functionality for producing real-time results. Wu (Wu, Naugle, & Meagher, 2016) discusses an analysis methodology on an overhung rotor system through MATLAB, and the techniques mentioned are used in this paper. Although the results shown there are proven to be very accurate, the analysis is not done in real time which adds several more challenges. Other universities have attempted real-time applications (McNicoll & Wu), but capabilities for slow roll compensation, filtering, and full spectrum computations are not mentioned.

In a two-disk rotor system, there are several physical parameters that have a large effect on the behavior during dynamic tests. Specifications such as shaft and disk geometry are easily attainable and can be used for rough estimations of critical speeds and amplitudes. Other parameters in the system however, are very difficult to measure directly but are needed for more accurate modeling and simulations. This warrants the use of theoretical models to estimate these values. In order to demonstrate the application and usefulness of this developed signal processing program, it was used in combination with theoretical models to determine the unknown parameters of a two-disk rotor. Bisoi (Bisoi, Samantaray, & Bhattacharrya, 2017) and Karthikeyan (Karthikeyan, Bisoi, Samantaray, & Bhattacharyya, 2013) discuss modeling similar systems with gyroscopic effects included, but like many other publications on this topic the comparison with experimental results is limited or non-existent.
2. BENTLY NEVADA’S ADRE SYSTEM AND ROTOR KIT

2.1 Introduction to Bently Nevada’s Products

In 1993, ADRE for Windows Software (Automated Diagnostics for Rotation Equipment) and the 208 DAIU (Data Acquisition Interface Unit) system was introduced by Bently Nevada, advertising the ability to capture and process up to sixteen channels of machinery data simultaneously. This product set a new standard for rotor dynamic analysis which resulted in its dominance of this market and very few comparable systems from competitors. This software along with the rotor kit systems that they produced, allowed for relatively low cost experimentation due to the scaled down size of the apparatus.

2.2 Rotor Kit Description

The physical rotor kit system, also developed by Bently Nevada, allows different rotor dynamic experiments to be run in a laboratory setting with great modularity. By changing the characteristics of the system, different vibration phenomena typically found in large rotating machinery can be duplicated on this smaller scale model. The main components of the RK 4 model are labeled in figure 1. The experiments and analyses described in this paper were performed using the provided ball bearings, rotor mass wheels, and steel shaft. For other types of analysis, the rotor kit is also able to accommodate a fluid film bearing, shaft rubbing or hitting conditions, and perturbation options.
As seen in figure 1, the large amount of available mounting locations along the base are what make the kit so versatile. It is along these flat surfaces, shown in figure 2, where the user can change the location of the disks, bearings, transducers, and safety covers. The bearing blocks can be moved closer and farther away from each other to decrease or increase the effective length of the shaft. This is typically done to adjust the natural frequency to appropriate level in order to capture the most useful data for each experiment during a ramp up procedure.

The transducers, which are also referred to as XY probes, are what transmit the displacement data from the rotor kit to an oscilloscope or monitoring software. There are no specific
instructions for the exact location of the XY probe mounting blocks, but fastening them as close as possible to the mass wheels will proved the most accurate data if trying to analyze that portion of the shaft. There are ten available locations for the probes on each mounting block, with five different surfaces in 45° increments, as shown in figure 3. Installing the probes orthogonal to one another in a desired orientation will result in data corresponding to an “X” and “Y” direction based on the user’s coordinate system. It is also important to note that mounting the probes in the same plane perpendicular to the shaft can result in electrical coupling, so this should be avoided. One example of a proper X and Y probe installation is shown in figure 3.

![Figure 3](image)

Figure 3. XY probe mounting block on Bently Nevada rotor kit, showing proper transducer installation.

There are also two transducers near the motor of the rotor kit which are labeled as “speed probe” and “Keyphasor® probe” in figure 1. The Keyphasor probe, like the X and Y probes, outputs data to a monitoring system, such as the ADRE DAIU 208. This probe focuses on a portion of the shaft that has a gap which results in a spike in the Keyphasor signal once per turn during rotation. The location of this spike in the signal serves as a reference for phase lag calculations and can also be used to get an approximation of the rotor’s rotational velocity. The speed probe focuses on several gaps on another area of the shaft, and this signal is used by the control box
that is provided with the rotor kit. It uses the same method as the Keyphasor, however a larger
number of spikes per rotation allows for a more accurate reading of rotational velocity. The
display screen on the control box displays the calculated speed of the rotor as it is running. The
speed control box, shown in figure 4, determines the behavior of the motor and has options to set
the maximum speed and ramp rate of the rotor during experiments.

Figure 4. Bently Nevada rotor kit speed control box.

The probes installed on the rotor kit are connected to the Bently Nevada Proximitior Assembly
which provides power for the signals, and allows splitting for the signals to be directed to
separate pieces of equipment. In figure 5, the Keyphasor and XY probe lines are plugged into the
desired channels on the back of the Proximitior Assembly. The banana cables shown are
providing power to the assembly, and come from an available port in the back of the Motor
Speed Control box.
For the set up in the Cal Poly, SLO vibrations lab, the probe signals going into the Proximitor Assembly are split and transmitted to two separate systems. One set of channels is connected to two Bently Nevada TK 21 Oscilloscope Interfaces which then output to two dual channel oscilloscopes. The other set of channels go directly from the Proximitor Assembly to the ADRE DAIU 208 system. The purpose of monitoring the signals on the 2 channel oscilloscopes is to provide a 2-D representation of the real time orbit shape and magnitude of the rotor kit shaft. This feedback is important to display certain phenomena that occurs during high speed rotation and also acts as a gauge for the user in case the amplitude of vibration gets too high. The two TK21 Oscilloscope Interfaces are to offset any bias in the signal leading to the Oscilloscopes. The horizontal and vertical knobs can be adjusted independently so the orbit displayed on the oscilloscope starts in the center of the screen before high speed rotation. The Keyphasor signal is split so it is able to be plugged into the back of these two oscilloscope interfaces however does not get affected when adjusting the offset. The Keyphasor lines are then connected to each of the oscilloscopes and act as a reference for the refresh rate of the display and location of the Keyphasor dot.
2.3 ADRE Application Description

The second set of channels from the Proximitior Assembly runs to the Bently Nevada 208 DAIU system. This paired with the ADRE software on a Windows computer allows real-time data to be presented to the user while capturing signals from rotating machinery. The ADRE software in the Cal Poly, SLO vibrations laboratory lets students and professors alike set up a data acquisition session for the Bently Nevada rotor kits. The main functions of this application are to determine how the DAIU collects, processes, and displays data for each separate experiment run. Bode, polar, and spectrum plots are the typical graphs selected for classes at Cal Poly, however the program has additional options that can be used to display and output data. Under the “Plot…” tab in the ADRE application, all of the output options can be viewed and configured by the user, shown in figure 6.

![Plot Control Window](image)

Figure 6. Plotting options for Bently Nevada ADRE for Windows software.

Next to each of the plot selection boxes in this plot control window there is a “configure” button which, when clicked, brings up a new window that controls the display of that type of graph. The
polar plot configuration window is shown in figure 7, which is very similar to the windows corresponding to the other types of graphs. The main portion of the configuration window is made up by the channel selection, which tells the software which data will be graphed and if compensation or filtering will be used. Along the right side of the configuration window, labeling, scaling, and formatting options can also be selected.

Figure 7. Polar plot configuration window in Bently Nevada ADRE for Windows software.

An option in the “plot control” window (Figure 6) that should be noted is the “Real Time” option. Configuring this option allows the user to choose up to six different plots to be displayed in real-time while the rotor kit experiment is being run. Although any of the plotting options are available after the DAIU has completed taking data, providing the user with graphs as the machinery is operating can be extremely important. In addition, with this real-time capability, students observing the rotor kit in motion can more easily correlate the physical behavior of the system to the trends in the plots.
Before any of this plotting occurs, the data acquisition session must be set up correctly and the transducer signals must be paired to the correct channels in the software. Clicking the “configure…” tab at the top of the ADRE application brings up a configuration window which has these options, shown in figure 8.

![Configuration menu on Bently Nevada ADRE for Windows software.](image)

In the “Reference” section of this window, the button “Plant/Job” opens a dialog where you can select a previously saved configuration. Below this reference section are the options for channel, transducer, and Keyphasor configuration. It is within each of these selections where the type and orientation of the probes can be specified. Figure 9 shows the window that opens when the “Channel…” button is selected. Along the top portion of this window there are five boxes that have options for the different characteristics of each signal. Under “Channel Names” the user can select the best fitting description of the channel, as this is only used as a reference for people viewing the data and has no effect on the functionality of the program. The “Machine” category is also another reference for description purposes, and for experiments described here only “Rotor Kit” will be used. The “Angle” section allows the operator to select the angle that the transducer for each channel is oriented at with an outboard view from the motor as the reference
plane. In our case, 0 degrees are where the vertical transducers are installed while the horizontal ones are 90 degrees to right, which corresponds to the “90 R” selection shown in the figure. The “Rot.” section describes the direction of rotation again with respect to an outboard view from the motor. All of the experiments described in this report will be spinning in a counter-clockwise rotation. The “Keyphasor” options tell the system which reference signal the channel will base certain calculations off of if there are multiple. Correct selection of this option is extremely important for getting correct values of phase lag, which depend on the relationship between the X and Y probes and the Keyphasor.

Figure 9. Channel configuration in Bently Nevada ADRE for Windows software.

“Transducer Configuration” is also a very important part of the software that allows you to designate the type of probe used and the settings associated with it. Figure 10 shows the “Transducer Configuration” window which has a similar layout and functionality of the “Channel Configuration” window described previously. Under “Transducer Type”, different types of sensors are available to be selected described by their series number and diameter. The
The scaling factor that the application uses for the sensors can also be chosen here. For the 7200 5mm probes that are installed on the rotor kits at Cal Poly, 200 mV/mil is the scaling factor used. Finally, the F/S range and bandwidth of the transducers are what determine the accuracy of the signal. 20 F/S with 12 bandwidth is the typical selection.

![Transducer Configuration Window](image)

Figure 10. Transducer configuration window on Bently Nevada ADRE for Windows software.

The final option in the “Channel/Transducer” section of the configuration window (figure 8) is the Keyphasor configuration. Although the settings in this window are very similar to those of the transducer configuration window, there are some important differences that should be understood. The hysteresis option determines what amount of lag the system should accommodate for based on the delay from signals. The “Man/Auto” category is short for “manual” or “automatic” and determines how the threshold value is calculated when running the experiment. Because the signal from the Keyphasor is a series of spikes over a somewhat constant value, the threshold is how the software can pick out the difference between a peak triggered by the Keyphasor and a smaller peak due to unwanted noise. Unless there are problems
with the ADRE software picking up on these Keyphasor spikes, it is advised to keep the “auto threshold” option selected. The option next to the threshold is “Maximum RPM”, which can be set to any desired value and specifies the range of rotational speeds that want to be focused on. The “Keyphasor Configuration” window can be seen in figure 11.

![Keyphasor Configuration Window](image)

Figure 11. Keyphasor configuration window in Bently Nevada ADRE for Windows software.

In the “Data Acquisition” section of the configuration window (figure 8), there are six different categories that affect how the DAIU collects and processes the data from the machinery it is connected to. The trigger event is what commands the unit to start collecting data from the connected equipment. This initiation event can be based off of a certain speed, amplitude, or phase that is measured from the operating machinery. The data acquisition can also be started with a predefined date and time, or an external contact from a separate electrical circuit. These options are shown in the “Trigger Source” section of figure 12. Also shown in this window is the “Vectors per Trigger” option, which allows the user to set a limit to how many data points can be recorded for each data set. For whatever the trigger source setting is selected to be, the
under/over setpoint option tells the software to start capturing data just before or after that value is reached.

![Trigger Event Configuration Window](image)

Figure 12. Trigger event configuration window in Bently Nevada ADRE for Windows software.

Configuring the sampling mode can also be controlled through the application, and determines how often the DAIU pushes data through the software to be processed and plotted. Figure 13 shows this window and the option of the mode being controlled by a specified change in time and/or a change in the rotational speed. These options are named “Delta Time” and “Delta RPM” in the sample mode section. Based on what the user specifies, the DAIU will collect a data sample whenever either a certain amount of time has passed or the speed of the shaft has increased or decreased by the RPM calculated from the selected Keyphasor source.
The “Nx Vector” option is a simple dialog box that allows the input of a specific filtering coefficient proportional to the operating speed of the rotor kit. This is in addition to the 1x and 2x vectors that the Bently Nevada software already includes by default. More explanation on the theory and reasoning behind these filters is explained in section 3.3.

The final option to cover in the configuration window (figure 8) is the “Spectrum Sample” configuration. These settings affect the span and the resolution of the spectrum and cascade plots. The frequency, which is typically plotted along the horizontal axis of these graphs, will range from zero to the number specified the “frequency span” section of the window (figure 14). Or, in the case of a full spectrum, this span will be doubled since the frequency ranges from the negative frequency value to the positive frequency value. The “Spectral Lines” category determines the resolution of those graphs, with 400 lines being the highest resolution and 50 lines being the lowest. For example, a half spectrum graph plotted from channels 1 or 2 with the settings shown in figure 14, a span of 200 Hz and 400 spectral lines, the resolution along the horizontal axis will be 0.5 Hz.
Figure 14. Spectrum sample configuration window in Bently Nevada ADRE for Windows software.
3. MATLAB SIGNAL PROCESSING PROGRAM

3.1 Goals of the project

The Bently Nevada ADRE application and DAIU has served its purpose extremely well in the vibrations lab at Cal Poly, however as more research has focused around these results and the rotor kit, more capabilities have been desired. With a better understanding of the theory behind the rotor dynamics and the calculations that the ADRE software performs, the feasibility of an alternative program paired with a commercially available off-the-shelf data acquisition card was discussed. The goals of this alternative option to the Bently Nevada DAIU and ADRE application was to be less expensive, contain as many if not more capabilities than ADRE, and be intuitive to students and professors using the rotor kit equipment.

Prior to this project, several theoretical models and calculations pertaining to the subject of rotor dynamics have been done in the programming language MATLAB both at Cal Poly and other universities around the world. Because this language is already widely used in this area of engineering and fully capable to take on the computations that the ADRE system performs, it was decided to move forward on the project using this software. Other benefits that come with using MATLAB are its ability to be used simultaneously with Simulink and be paired easily with certain data acquisition devices. With some research done on the different brands of compatible DAQ cards commonly available, the National Instruments cDAQ-9174 module installed with NI 9215 cards was chosen.
Figure 15. National Instruments cDAQ-9174 module installed with two NI 9215 cards.

The National Instruments module and data acquisition cards described are shown in figure 15. To capture the signals from the rotor kit, the BNC cables from each transducer were connected to the ports available on the cards. The entire module is then connected via USB to the desired computer to transmit the data.

3.2 Explanation and capabilities of DAQ card

The National Instruments cDAQ-9174 is a housing that is capable of being installed with four separate National Instruments cards. The housing is powered by any regular wall outlet and connects to any computer through a USB port. The USB connection and modular capability for expanding the number of channels were the main reasons why it was selected. Having this “plug-and-play” capability with the USB connection minimizes the time needed for setup (National Instruments, 2013). The specific National Instruments card that was chosen to be installed with this module was the NI 9215 model. Each card has four separate channels with BNC connection capability, and can sample each channel to a frequency up to 100 kS/s (National Instruments,
2016). Since the transducers installed on the rotor kit used BNC connections, this was a very easy card to integrate into the system. The analog input range for this model +/- 10V which is acceptable for the variation in voltage that comes from the transducers installed on the kit. The full data sheet for the NI 9215 card can be found in Appendix B.

3.3 Explanation of MATLAB code

As stated earlier, one of the main goals of this MATLAB program is to have the same capabilities as the ADRE software, and a large part of that is the graphing ability of the Bently Nevada application. The plotting options available in the ADRE application consist of polar, bode, orbit, half/full spectrum, cascade, and waterfall graphs. With these requirements understood, the different types of variables eventually needed to be obtained from the data could be determined. The polar plot requires a phase and amplitude value in order to illustrate important values at certain operating speeds of the rotor kit. Although only the phase and amplitude are required for the plot, including the rpm in this graph allows it to be used for single plane balancing experiments which are very common in the vibration lab where these kits are located. The bode plot, which is a combination of two graphs, requires the same three variables. Both plotted in Cartesian coordinates with operating speed along the horizontal axes, one of the graphs in the bode plot has phase along the vertical axis while the other has the shaft’s amplitude. The orbit plot is real-time display of a horizontal and vertical traducer signal, which is currently what the two-channel oscilloscopes in the lab are set up to do. The orbit represents displacement and does not require any complex calculations on the raw data; however a refresh rate and display of the Keyphasor signal is required. The remaining three plots do not require any additional variables than those that have been discussed except the waterfall plot, which has elapsed time along one of its axis. Therefore, the values needed from the data in order to get the
desired results are rotational velocity of the shaft, displacement, phase lag, horizontal and vertical and amplitude, and elapsed time throughout the experiment.

The first technique attempted to obtaining these values discussed above was using Simulink paired with the NI-DAQ device. This was both a test for the functionality of the data acquisition and a good way to see how the data was fed into Simulink and how it behaved with commonly used blocks. Understanding how the data is organized when collected by the card is extremely important and depends on the settings when starting the DAQ session. In Simulink, there is a NI-DAQ block available when the device is plugged in and the correct drivers are downloaded. When this block is opened, the input sample rate and block size settings are what determine the size of the data. The sample rate is the frequency of the speed that the data is recorded at in samples per second. The block size is how many data values are taken before that “block” of data is fed to the rest of the Simulink model. Because of this, the data coming from the NI-DAQ block is not a single array of values, but a matrix where each column contains the number of data values specified in this block size setting. The amount of columns depends on the duration of the data acquisition session.

The rotational speed of the rotor kit is an extremely important value that other calculations depend on and was the first variable focused on calculating with the Simulink model. Different methods of calculating the speed based off of a Keyphasor signal were researched, however all suggestions found in online forums and on the MathWorks website referred to more complex blocks than what was thought necessary. Because the Keyphasor signal only consists of a repeating spike over a relatively constant value, a simple logic diagram was implemented for the speed calculation. This method, shown in figure 16, is able to calculate the rotational speed of the shaft based off of the Keyphasor signal with an “if” statement and the “detect positive rise”
block in Simulink. The data from the card comes in at port 1, in the matrix orientation that was described previously in this section. The “DC Blocker” block is set at “subtract mean” which subtracts any constant offset and sets the center of the Keyphasor signal at zero. The “unbuffer” block then converts the data from a matrix into a single array, which allows values from different blocks in the data to be fed continuously through the “if” statement rather than one set at a time. Now unbuffered, the data goes through a “detect positive rise” block which sets a value to either “true” (1) or “false” (0). The data value is set at 1 if the previous value was less than zero and the current one is greater or equal to zero. Otherwise, all other data points are set to zero. The “if” statement then feeds only the “true” data points to the subsystem block. This subsystem then calculates the period of time between each of these “true” values which is theoretically the same as the period of time between two Keyphasor spikes (i.e. one rotation of the shaft). This data is then buffered back into the original size of data and then averaged and converted to obtain a speed in rotations per minute.

![Diagram](image)

Figure 16. Speed calculation model in Simulink for rotor kit.

This model was first tested with the rotor running at a slow roll, or around 300 rpm, and proved to be very accurate when compared with the readout on the motor control box. At higher running speeds however, the measurements from Simulink would intermittently spike from the actual rpm. This was most likely due to the increased amplitude of vibration at higher speeds resulting
in extra peaks in the Keyphasor signal that did not necessarily come from the once-per-turn reference. A temporary fix for this was subtracting an offset off the Keyphasor signal so the positive rise detector would only see the larger Keyphasor peaks while the unwanted vibration was kept below that threshold. A more robust fix for this would be to subtract a certain value off the signal based on a percentage of the calculated amplitude, but this was not implemented.

Although this Simulink model could continue to be developed, the focus was shifted to MATLAB once interfacing with the NI-DAQ was successful and better understood. Simulink has all of the capabilities that MATLAB offers for the requirements of this project, but because of the desire to run through a separate graphic user interface, MATLAB was chosen.

The general process to obtain all the desired results for the experiment contains three main steps. The first is collecting the data from the connected probes on the rotor kit to the data acquisition card. Next, there has to be some method of streaming that data at a certain rate and size to a MATLAB script. The final step is for that script to be able to process that data automatically and output the desired results, which includes graphs and data files. Each of these steps would contain several commands and functions to run, and rather than keeping them in separate files an object-oriented programming approach was chosen to keep the program better organized. This makes improvements to the program much simpler and documentation of the code more understandable. In addition, the “App Designer” tool which helps build a user interface for a program runs on object-oriented code, so interfacing with this would also be easier. It is important to note that this application, which will be described later in more detail, is not an extra step in this process, but a simplified display of the adjustable variables in the program.

To understand the functionality of this MATLAB program, it’s necessary to comprehend the purpose of a class definition which allows the creation of objects in MATLAB. Objects in
MATLAB are structures that can contain properties, methods, events, and enumerations. “Properties” are available spaces in the object where values can be stored, viewed, and managed by the object itself or other functions in MATLAB. “Methods” are just collection of one or more functions in the object which have all the capabilities of standalone functions in MATLAB. “Events”, defined by MathWorks, “are notices broadcast when some change or action occurs involving an object”. This is a critical feature that is used in this program because, as explained later, it allows the streaming of data from the rotor kit to MATLAB at a desired rate. “Enumerations” represent a fixed set of named values, however this object feature is not utilized in this code. The entire class definition for this code is shown collapsed in figure 17.

```
classdef TestObject < handle
    properties (SetObservable)
    methods
        function measureGap(obj)
        function sendData(obj,~,event)
        function startDAQ(obj)
        function plotData(obj,~,event) % (AffectedObject, Source, EventName)

    end
end
```

Figure 17. Collapsed class definition of TestObject in MATLAB.

The first step of the process is collecting data from the transducer lines with the connected National Instruments card. In the class definition code, this is done in the “startDAQ” function under the methods category which is shown in figure 18. First, a DAQ session is created with the “daq.createSession” command that allows parameters and channels to be set by the user. Settings on the card able to be adjusted in this function are the sampling rate and duration of the session. The sampling rate, also explained in the Simulink section, controls how often the DAQ card records a data value. In this function there is also a listener added (lis1) which trigger when the “Data Available” event fires. This event is active by default in MATLAB and control when a set
of data can be processed. On line 76 of this code, the “NotifyWhenDataAvailableExceeds” value determines how big that data set is, and is analogous to the “block size” setting described previously in the Simulink section, which can also be referred to as a “time window”. Using a window method is a common approach to analyzing time-based data and is necessary for the real-time analysis capabilities of the program. The size of the window or block can also be seen as a percentage of the entire data that is being analyzed at a time. Therefore, a larger window can result in less accurate calculations if averaging is occurring during each of the windows. Alternatively, smaller windows can result in more accuracy, however constricting a window to a small size might not provide enough data in calculations like phase lag, where at least two full oscillations in the signal need to occur. With simple math, the amount of windows per second can be calculated from the sample rate and window size, or “obj.notify”, values. The default values for this code have the sampling rate set at 20,000 sample per second and window size set at 10000 samples per window. Dividing the rate by the window size results in the number of windows per second, which in this case is 2. The amount of channels added to the session depends on how many probe signals are connected to the NI device. Finally, starting the acquisition of data in the background with the “startBackground” command allows separate commands in MATLAB to be called while the session is still running.
In the “addlistener” command of the “startDAQ” function, the third argument determines what function is called when the event is triggered. In this code, it is set to the “plotData” function in the class definition and is where the variables are calculated and graphs are produced. Because this function is called every time a window is available, it acts as a loop, so proper organization and concatenation of the values is necessary for the different calculations. Using this window method is a very common approach to analyzing time-based data and is necessary for the real-time analysis capabilities of the program. For each time the “plotData” function is called by the listener, the data from the event has any constant offset removed in line 99 of figure 19. This sets the mean value of the oscillating signal coming in to zero, the same effect that the “DC Blocker” block did in the Simulink model. In line 103, all of the data from the event is added on to the end of a persistent variable, which holds all of the data up to the current time window. The columns of this matrix, tempData, in order are the timestamps and then the consecutive channels that are active for the session.
Calculating speed from the Keyphasor signal in the code is done in a different manor than the Simulink model. Fortunately, there is a function available called “pulseperiod” that does almost exactly what was created through the “if” statement described earlier. This function detects the period of time between two pulses, however in the Keyphasor signal there is both a negative and positive spike for each rotation. This initially caused problems since the period between either two positive pulses or two negative pulses for each rotation was desired. The fix for this was removing the negative spikes from the signal with an “if” statement that is shown on line 135 in figure 20. With the proper period calculated, the values for the window are average and added into a “speed” matrix where it is also converted to rotations per minute. Also in this section of the code, the X and Y amplitudes of the signal are calculated by subtracting the maximum and minimum values in the current window. Now in peak to peak format, the root of the sum of the squares of these values are put into a “Tamp” variable. This variable along with the calculated speed allows one half of the bode plot to be created.
Figure 20. Speed and amplitude calculations in MATLAB code.

The second portion of the bode plot is a graph of the phase lag versus the running speed of the shaft. This is very important to have when analyzing dynamic behavior because it shows when the vibrations become in or out of phase with the source. Combined with the amplitude data, the phase plot allows verification of natural frequencies when undergoing a ramp up cycle. The phase lag value is calculated by analyzing the changing relationship between a reference signal and a signal from another point on the shaft. In this case, the reference signal is the Keyphasor pulse which is compared to one of the signals from the XY probes. When the shaft rotates at a slow roll, the peak amplitude of a point on the shaft is a relatively constant phase measurement from the Keyphasor spike. As the speed of the shaft increases however, the distance in this peaks change which can be seen in their displacement output. The way that the MATLAB code calculates the specific phase lag occurring starts with the “findpeaks” function. This function can output both local peak amplitudes and locations in a specified signal with settings adjusting the amplitude threshold and the minimum distance between peaks. With successful tuning of the
function, this code finds all of the peaks and locations associated with the Keyphasor and XY probe signals. With this information, the period of the Keyphasor signal is found for each window along with the probe’s peak location within that period. The initial location of the XY peak in the window set equal as a reference, and as the location changes with respect to the Keyphasor period, the variation in phase is calculated.

With the phase and amplitude values from the signal calculated, the polar plot can also be created. The polar plot displays the amplitude of vibration with respect to the phase lag in polar coordinates and extremely helpful for disk balancing experiments and quickly visualizing where heavy spots along the disks are. An additional option that the ADRE system is capable of performing for this graph is calculating the “slow roll compensation” value. When the rotor is spinning at slower speeds, the data that is recorded does not necessarily have meaningful value when the dynamic behavior at higher speeds is the main focus. Subtracting the phase and amplitude at this lower speed removes the offset caused by the slow roll vector and is typically called “slow roll compensation”. The effect of the compensation can easily be understood when viewed in a polar plot. As seen in figure 26 when compared with figure 25, the compensated polar plot has its starting value shifted to the origin of the coordinate system. Not only does this remove the dynamic effects of slow roll, it allows the theoretical calculation of heavy spots on the shaft much easier. To calculate this slow roll compensation value, the user can choose which data point to select as their slow roll vector. This is usually either the first or one of the first data points during a ramp up experiment. Once the desired slow roll point is determined, the compensation isn’t as simple as subtracting every value in the polar plot by the amplitude and phase of the slow roll point. This is because the slow roll compensation is a Cartesian coordinate transformation while the plot is in polar coordinates. Once the Cartesian coordinate of the slow
roll vector is determined by calculating the angle and amplitude off of the horizontal and vertical axes, each of the values in the data sample can be compensated.

The orbit plot is a two dimensional representation of the displacement occurring on a portion of the shaft during rotation. From a computational standpoint this is a much easier graph to obtain since it is simply the signals from the X and Y probes plotted against one another. The only modifications to the signals that have to occur are the DC offset that was explained earlier and shown in figure 19, as well as the scaling value to account for the voltage to displacement conversion. For the 7200 5mm model of transducer that is installed on the rotor kit used for these experiments, the scaling value is 200 mV/mil. Although not much post-processing of the data is need in this section of the code, the refresh rate of the orbit plot does have to be controlled if an accurate representation of each orbit and Keyphasor pulse is desired. Since each window or block of data can contain data from multiple rotations of the shaft, the orbit plot has to be refreshed or redrawn after every full rotation in order to avoid multiple orbits being plotted over one another. This action is done by using the peak location data obtained from the phase calculation section of the code. By only plotting the data between each peak within the Keyphasor signal, a single orbit of the shaft is plotted at a time. The location of the Keyphasor dot and gap on the plot is also very important because it symbolizes the direction of rotation. In the Bently Nevada software, the direction of rotation is determined by going from “blank to bright” on the orbit plot. This same rule is used in this code, and is done by leaving out a small portion of the data at the end of the orbit while plotting.

The spectrum, cascade, and waterfall plotting options all include information about the vibratory response of the system in the frequency domain. The half/full spectrum plot, and simplest of the three, displays the amplitude of vibration with respect to the frequency of vibration at a specified
running speed. A third axis is added to the spectrum plot in order to create the cascade and waterfall plots. For the cascade, the third axis is running speed of the system while the waterfall plot has the elapsed time along the third axis. Typically the cascade plot is most used, as it relates certain behavior to a running speed rather than a point in time which might not have as much significance. In order to have the amplitude of vibration plotted against the frequency of vibration, Fast Fourier transformation (FFT) is used. Before the data undergoes the transformation, a variable is created with the horizontal and vertical amplitude data set respectively as the real and complex components of the value. This value is input into a Hanning window function which allows it to be properly introduced as an argument to the “FFT” function in MATLAB. These steps rearrange the amplitude data so it corresponds with a frequency range that is calculated from the sampling rate and window size of the session.

By default, all of these plotting options described display the entirety of the data collected from a data acquisition session, however only focusing on a certain portion of the data is commonly desired. In dynamic analysis, filtering the data can help narrow down the causes of the behavior observed in a system. For rotating machinery, synchronous and non-synchronous filtering can magnify or neglect the response of the system that is associated with the running speed of the motor. This filtering tool was implemented into the MATLAB code with the capability for the user to choose which speeds to filter out. Using the notation that is most familiar in the industry and used by the ADRE system, the filter can be selected by choosing any real number for the “nX” filtering value. For example, using a “1X” filter would result in data only relating to one times the operating speed being shown. With more knowledge of a specific system’s behavior, this value could be tuned by the operator to show critical information needed to identify different phenomena. The filter in the code is built by using the filter designing tool in MATLAB. For this
an elliptical filter is used which requires an order, attenuation, and bandpass to be specified. By tuning these settings over several trial runs, the filter is effective at removing unwanted data and is comparable to the capabilities of the ADRE system. Although this technique was successful, further research into this filtering topic is suggested since there is substantial theory behind their applications.

3.4 Explanation of MATLAB app

A MATLAB graphic user interface application was developed to work with this code in order to allow students to run the program without having to understand all of the programming behind it. The goal of this application is for users to be able to transition easily from using the familiar ADRE software to the MATLAB alternative. The MATLAB code can still be run on its own, the application designed simply presents the main adjustable parameters in an intuitive way. The application was built using the “App Designer” tool in MATLAB, which drastically cuts down on the time and effort spent on formatting and editing the physical layout of the application. The tool also presents the callback editing ability in a way that is very easy to pick up without any prior experience. The only downside to using “App Designer” with the rotor kit MATLAB program is that the application is a separate file than the rotor kit code itself. This is not a huge issue but means that if the application needs to be run on different computers, both files need to be transferred. Ideally this program would be able to be condensed in standalone executable file, and this is certainly something that should be pursued in future revisions. The current design of the rotor kit MATLAB application is shown in figure 21.
Starting at the top of the application, the “Reset” button clears the workspace of MATLAB and changes any of the adjustable variable in the code back to their default values. This should always be the first thing selected when setting up a new DAQ session to avoid any errors resulting from currents variables and data from previous runs or scripts. Below this are the options to set up the channels that are connected to the NI-DAQ device(s). Currently the number of channels allows any number from 1 to 8 to be selected, since this is the range available with the current two-card setup in the vibrations lab. With four cards installed in one of the NI-DAQ
devices, the maximum number here could be raised to 16 channels. Based on the number of channels selected, the user can designated which transducers installed on the rotor kit correspond to the channels in the program. This is done by selecting the number from the drop down boxes. Next, there are three editable numeric boxes that determine the size and resolution of the data being processed. The sampling duration requires a positive number to be input and sets how long in seconds the DAQ card will record data after starting. The sampling rate and window size correspond to the same variables that were explained in the “startDAQ function” section of the report. The editable “speed limit” field tells the program the upper limit that the plots with a speed variable should go up to, and does not affect the computations that are performed. Underneath the “speed limit” box are five different plotting options available to be selected, which control the plots that are displayed in real-time during the experiment. At this point, the “Run” button can be selected which starts the DAQ session, and if the user wishes to stop the session before the time duration is reached, the “Stop DAQ” button can be clicked. In order to ensure accurate displacement data, it is advised to measure the gap of the transducers before running the experiment. While the motor is not running, the gap at the horizontal and vertical probes on the shaft should be about 23 mils, which corresponds to a -5.75 ±.5 V reading (Bently Nevada, 2002). This measurement can also be performed through the application. Selecting the “Measure Gap” button at the bottom of the window starts a separate DAQ session and function from the class definition code. It displays in real-time the voltage reading from transducers so they can be adjusted while feedback is provided. Once an acceptable gap is set, the “Stop Measuring” option will end this separate session.
4. COMPARISON BETWEEN ADRE SOFTWARE AND MATLAB SIGNAL PROCESSING PROGRAM

4.1 Software and hardware comparison

The capabilities of the ADRE software and MATLAB program are very similar but the differences between them lead to pros and cons of using one over the other. As a commercial product developed by Bently Nevada, the ADRE software paired with the DAIU has shown through extensive use why it has been one of leading data acquisition systems in this industry. Alternatively, the student-created MATLAB program features modularity, which is the main factor contributing to its possible preference over the Bently Nevada counterpart. A more in depth comparison of the differences and similarities between these products will be discussed in this section.

The biggest difference physically and functionally between these two systems might very well be the hardware associated with each. Bently Nevada’s product includes the DAIU, a standalone processing unit that has dedicated resources to computing and displaying the results that are requested through the ADRE application. With the DAIU operating independently of the computer running the ADRE application, the processing time and associated lag during experiments is minimal and limited only by the capabilities of the hardware inside the unit. The student-created program runs through the MATLAB application on a typical computer system unit. The drawbacks with this method are the limitations and fluctuation of processing power that the program receives. Oftentimes the computer with the MATLAB application installed is also used for other purposes, which can result in increased time for computations. Another factor that
can lead to this lag is the actual specifications of the computer, which can vary drastically from one unit to another.

**4.2 Experimental Results Comparison**

In order to compare the two different DAQ systems against each other most effectively, they were both connected to the same channels while a single-disk rotor kit experiment was performed. This was done on a Bently Nevada RK4 model rotor kit in the vibrations laboratory at Cal Poly, SLO. For this experiment, only two transducers, one vertical and one horizontal, were installed and connected to each of the DAQ devices. The weighted disk and bearings were installed in the same configuration and with the same dimensions as shown in figure 22 and table 1.

![Figure 22. Set-up of Bently Nevada rotor kit for single-disk experiment to compare ADRE for Windows and MATLAB DAQ systems.](image)
Table 1. Physical geometry and material descriptions of single disk rotor system.

<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L_1$ (Bearing 1 to Disk)</td>
<td>22.0</td>
<td>cm</td>
</tr>
<tr>
<td>$L_2$ (Disk to Bearing 2)</td>
<td>23.0</td>
<td>cm</td>
</tr>
<tr>
<td>$L$ (Total Shaft Length)</td>
<td>45.0</td>
<td>cm</td>
</tr>
<tr>
<td>Shaft Diameter</td>
<td>1</td>
<td>cm</td>
</tr>
<tr>
<td>Disk Diameter</td>
<td>7</td>
<td>cm</td>
</tr>
<tr>
<td>Disk Thickness</td>
<td>2.5</td>
<td>cm</td>
</tr>
<tr>
<td>Shaft Material</td>
<td>Stainless Steel</td>
<td>N/A</td>
</tr>
<tr>
<td>Disk Material</td>
<td>Stainless Steel</td>
<td>N/A</td>
</tr>
</tbody>
</table>

The motor control box for the experiment was set to a maximum speed of 3000 RPM and a ramp rate of 3500 RPM/min. Both systems were set to produce bode, polar, and cascade plots in real time based off of the horizontal and vertical transducers. For the MATLAB application, a sampling rate of 20000 samples per second and 5000 samples per window were selected. For the sampling mode on the ADRE application, the “delta RPM” option was selected with 30 RPM increasing or decreasing input as the controlling parameter. After a speed of 3000 RPM was reached by the system, the separate sessions on the two programs were stopped. A comparison of the results between ADRE and MATLAB from the horizontal channel are focused on starting with the uncompensated bode plots in figure 23.
Figure 23. Amplitude and phase lag results from the ADRE for Windows application and MATLAB program recording the horizontal transducer for a single-disk rotor kit ramp-up experiment.

The uncompensated amplitude and phase results from each of the systems are very well correlated, with the biggest differences being apparent in the calculated phase lag at slower speeds. The “noise” that is present in the MATLAB phase values at speeds typically below 1000 rpm results from the smaller amplitudes at these speeds. Because the phase is calculated from
peaks in the signal’s waveform, smaller amplitudes lead to increased error when identifying the peaks associated with the rotordynamic behavior. Fortunately the specific value of the phase lag, especially at slower speeds, is rarely of high importance. The trend shown by the phase, specifically at the points of natural frequencies is what is usually desired for these types of tests. Even through the slower speeds, the trend of the phase from both systems can still be identified and, when compared with the behavior in the amplitude graph, the location of the natural frequency can be confirmed with confidence. From the 180 phase shift and spike in the amplitude, the first natural frequency of this system is at approximately 2093 rpm. Slow roll compensation is commonly desired in rotordynamic tests in order to filter out data that is not attributed to rotordynamic behavior. This slow roll behavior comes from a combination of mechanical and electrical runout in the system. In order to achieve this compensation, a slow roll vector first must be selected and then subtracted from all of the other values collected in the data set for each experiment. As long as a range of slow-roll speeds is known for the system, Maalouf (Maalouf, 2007) states that “any convenient speed within this range can be used for collection of slow-roll data.” By default, since all experiments will be started from slow roll, the MATLAB system selects the values at the first recorded speed (or one the user specifies) as the components of the slow roll vector. The ADRE system selects the slow roll vector in a similar way. By default one of the first data points is selected, but the application also allows the user to select a different rpm for the slow roll vector. Figure 24 shows the bode plots from the ADRE and MATLAB systems with slow roll compensation included.
Figure 24. Amplitude and phase lag results with slow roll compensation from the ADRE for Windows application and MATLAB program recording the horizontal transducer for a single-disk rotor kit ramp-up experiment.

Once again, there is fluctuation present in the trend of the phase lag at speeds below 1000 rpm, but now magnified because of the introduction of slow roll compensation. The effect of the compensation is also apparent in the amplitude graph, where the calculated amplitudes from both programs now start at close to 0 mil pp. The effects and comparison of the slow roll compensation is also illustrated in the polar plots of figure 26.
The next graphical representation of the data to be compared between the ADRE and MATLAB system is the polar plot. Although this plot uses the same data as a bode plot (amplitude and phase), the different coordinate system allows some differences in the data to be seen more easily. Figure 25 shows the uncompensated polar plots from the ADRE and MATLAB system. The red portion of each of the plots denotes the low rpm side of the graph. This information is important when determining the heavy side of an unbalanced system such as this one.

Figure 25. Polar plot from the ADRE for Windows application displaying amplitude with respect to phase of the vertical transducer for a single-disk rotor kit ramp-up experiment.

One difference that stands out in these plots that was not able to be seen as easily when comparing the bode plots, is the resolution of the two systems at higher speeds. In figures 23 and 24, the differences between the amplitude and phase values from ADRE and MATLAB seemed
to stay relatively constant through the entire experiment. However, from the polar plots in figures 25 and 26 the difference in resolution is apparent, especially at the larger amplitudes. This is a result of the sampling method chosen for each of the systems. The “delta rpm” method selected in ADRE has benefits associated with it in some cases but one of the drawbacks is shown here. When a large change in the signal happens over a small range of speeds, such as the natural frequency, some accuracy is lost. Since the constant sampling rate of the MATLAB system was not set to depend on speed, the resolution remains the same.

Figure 26. Polar plot from the MATLAB application displaying amplitude with respect to phase of the horizontal transducer for a single-disk rotor kit ramp-up experiment.

From the slow roll compensated polar plots in figure 26 above, the effect of removing this slow roll vector can be seen when compared with the uncompensated alternative. Subtracting this
vector from the entire data set translates the graph so that the starting point is at the origin of the coordinate system.

The full spectrum cascade plots are also compared here to show the accuracy of the calculations the MATLAB program has to the results of the ADRE software. Figure 27 shows the cascade plot from the Bently Nevada system, which has both the second and third axes in the vertical direction and frequency along the horizontal axis.
Figure 27. Full spectrum cascade plot from the ADRE for Windows application displaying amplitude with respect to the frequency of vibration and running speed for a single-disk rotor kit ramp-up experiment.

From the cascade plot produced by the ADRE system, the peaks in the frequency spectrum occur at around 35Hz for the forward vibration components and -35 Hz for the reverse components. The axis along the left side of the plot shows that the amplitude spikes at around 2000 rpm, which agrees with the results in the bode and polar plots. Figure 28 shows how the MATLAB application graphs the same spectrum data in cascade form.
Figure 28. Full spectrum cascade plot from the MATLAB application displaying amplitude with respect to the frequency of vibration and running speed for a single-disk rotor kit ramp-up experiment.

Although the results are very similar to those from the ADRE system, an immediate difference that is apparent between the two cascade plots is the true third dimension of the MATLAB plot. This is due to all three variables being plotted on separate axes unlike the ADRE system which combines two variables in the vertical direction. A benefit of having this third axis in MATLAB is the ability for the user to rotate this graph in any orientation to view the results from all directions.
5. DEVELOPMENT OF THEORETICAL MODEL FOR TWO DISK SYSTEM AND
EXPERIMENTAL COMPARISON

5.1 Overview of theoretical models used

With single disk systems having their weight located in the center of the rotor, such as the one described in the section 4, gyroscopic forces do not play as much of a role in the behavior of the system because of the limited out-of-plane motion the disk experiences. With systems that have disks not centered along the span of the shaft or multiple disks installed, gyroscopic forces can have a much larger effect on the outcome of critical speeds and amplitudes during dynamic tests. In this section, two different theoretical models for a two-disk Jeffcott rotor will be described: one that does not include gyroscopic effects and one that does. The results of these two models will be then compared against each other as well as the MATLAB experimental results obtained from a two-disk ramp-up experiment.

5.2 Two-Disk Theoretical Model without Gyroscopic Forces

The two-disk system is modeled with a system of ordinary differential equations that correspond to the dynamic behavior at each disk location. In order to account for possible anisotropy in the bearings, the vertical and horizontal components of movement are included. This results in four separate equations. The equations of motion (1) used to model this system are determined using conservation of momentum and dynamic analysis. These equations include forcing functions on the right hand side of (1) that depend on the acceleration of the shaft so that the theoretical model can account for start-up and run-down situations (Giancarlo, 2005; Muszynska, 2005).
\[
\begin{aligned}
&M_1\ddot{x}_1 + d_{x11}\dot{x}_1 + d_{x12}\dot{x}_2 + k_{x11}x_1 + k_{x12}x_2 = M_1a_1\omega^2\cos(\omega t + \varphi_1) - M_1a_1\alpha\sin(\omega t + \varphi_1) \\
&M_2\ddot{x}_2 + d_{x21}\dot{x}_1 + d_{x22}\dot{x}_2 + k_{x21}x_1 + k_{x22}x_2 = M_2a_2\omega^2\cos(\omega t + \varphi_2) - M_2a_2\alpha\sin(\omega t + \varphi_2) \\
&M_1\ddot{y}_1 + d_{y11}\dot{y}_1 + d_{y12}\dot{y}_2 + k_{y11}y_1 + k_{y12}y_2 = M_1a_1\omega^2\sin(\omega t + \varphi_1) + M_1a_1\alpha\cos(\omega t + \varphi_1) \\
&M_2\ddot{y}_2 + d_{y21}\dot{y}_1 + d_{y22}\dot{y}_2 + k_{y21}y_1 + k_{y22}y_2 = M_2a_2\omega^2\sin(\omega t + \varphi_2) + M_2a_2\alpha\cos(\omega t + \varphi_2)
\end{aligned}
\]

(1)

In order to determine the stiffness matrix used for the theoretical model of the two-disk Jeffcott rotor model, expressions for deflection along the shaft must be found. These expressions determine the behavior of the shaft when subjected to different forces. Initially, the flexibility of the shaft and bearings are considered separately then combined using the superposition method.

The flexible influence coefficients method in the general form of (2) is used to derive the flexibility matrix \([C]\) which can be applied in both the \(xz\) and \(yz\) plane, respectively. Since the shaft is considered to be isotropic, the matrix is identical for the \(xz\) and \(yz\) plane (Kramer, 2012; Dimarogonas & Haddad, 1992). Parts of these derivations include the consideration of two different loading conditions, which are designated by “Case 1” and “Case 2” in each section.

\[
\begin{bmatrix}
\text{Linear Displacement of Disk 1} \\
\text{Linear Displacement of Disk 2}
\end{bmatrix} = [C]\begin{bmatrix}
\text{Force Applied at Disk 1} \\
\text{Force Applied at Disk 2}
\end{bmatrix}
\]

(2)

5.2.1 Flexibility Matrix Derivation: Flexible Shaft

One cause of the deflection of the weighted disks comes from the flexibility of the shaft and can have a large effect on the behavior of the system. In order to include this effect in the theoretical model, shape functions that approximate the displacement based on an applied force are used. The two cases apply the point load at each of the weighted disk locations separately.
Figure 29. Two-disk schematic diagram for flexible shaft stiffness derivation.

Case 1

\[ a = L_1, \ b = L_2 + L_3 \]

\[ F_1 = 1, \ F_2 = 0 \]

The sixth condition in table E-9 of “Useful Tables” in *Shigley’s Mechanical Engineering Design* (Budynas & Nisbett, 2011) gives the following equation for the deflection of a beam with an intermediate load between two simple supports.

\[ y_{AB} = \frac{-Fb}{6EIL} \left( x^2 + b^2 - l^2 \right) \]

\[ y_1 = \frac{-1(L_2 + L_3)(L_1)}{6EIL} \left( L_1^2 + (L_2 + L_3)^2 - l^2 \right) \]

\[ y_1 = \frac{L_1^2(L_2 + L_3)^2}{3EIL} = C_{11} \]

Similarly, *Shigley’s Mechanical Engineering Design* provides an expression for deflection along the shaft for a point where the second disk is located.
\[ y_{BC} = \frac{-Fa(L - x)}{6EI} (x^2 + a^2 - 2Lx) \]

\[ y_2 = \frac{-1(L_1)(L_3)}{6EI} \left( (L_1 + L_2)^2 + (L_1)^2 - 2L(L_1 + L_2) \right) \]

\[ y_2 = \frac{L_1L_3(L^2 - L_1^2 - L_3^2)}{6EI} = C_{12} \]

**Case 2**

\[ a = L_1 + L_2, b = L_3 \]

\[ F_1 = 0, F_2 = 1 \]

The shape function in "Case 1" used to find \( C_{11} \) is used again to find \( C_{22} \) due to the change in the loading condition.

\[ y_{AB} = \frac{-Fbx}{6EI} (x^2 + b^2 - L^2) \]

\[ y_2 = \frac{-1(L_3)(L_1 + L_2)}{6EI} \left( (L_1 + L_2)^2 + L_3^2 - L^2 \right) \]

\[ y_2 = \frac{L_3^2(L_1 + L_2)^2}{3EI} = C_{22} \]

And with the Maxwell-Betti Law of Reciprocal Deflections, an expression for \( C_{21} \) is found.

\[ C_{12} = C_{21} \]

Assembling each of the flexible influence coefficients in matrix form results in the flexibility matrix for the shaft.
Another cause of deflection in the system is the movement of the bearings that support the shaft. The flexibility of these bearings is derived below by modeling each of them as simple linear springs. The two cases, like the previous derivation, apply the point load at each of the weighted disk locations separately.

\[ C_s = \begin{bmatrix} \frac{L_1^2(L_2 + L_3)^2}{3EI} & \frac{L_1L_3(L_2^2 - L_1^2 - L_3^2)}{6EI} \\ \frac{L_1L_3(L_2^2 - L_1^2 - L_3^2)}{6EI} & \frac{L_3^2(L_1 + L_2)^2}{3EI} \end{bmatrix} \]

5.2.2 *Flexibility Matrix Derivation: Flexible Bearings*

Figure 30. Two-disk schematic diagram for flexible bearing stiffness derivation.

**Case 1**

\[ F_1 = 1, F_2 = 0 \]

Using properties of similar triangles, the following expression is found.
\[
\frac{y_1 - y_A}{y_B - y_A} = \frac{L_1}{L}
\]

Rearranging the terms to solve for \(y_1\) results in equation (1).

\[
y_1 = \frac{L_1}{L} (y_B - y_A) + y_A
\]

\[
y_1 = \frac{L - L_1}{L} y_A + \frac{L_1}{L} y_B
\] (3)

\[
y_2 - y_A \quad y_2 = \frac{L_1 + L_2}{L} y_B
\]

Similarly, an expression for \(y_2\) is shown in equation (2).

\[
y_2 = \frac{L_1 + L_2}{L} (y_B - y_A) + y_A
\]

\[
y_2 = \frac{L_3}{L} y_A + \frac{L_1 + L_2}{L} y_B
\] (4)

By summing moments around point A, an expression for the resulting force at B is found.

\[
\sum M_A = -F_B(L) + F_1(L_1) = 0
\]

\[
F_B = \frac{F_1 L_1}{L}
\] (5)

Summing moments around point B results in an expression for the force at A, shown in equation (6).

\[
\sum M_B = F_A(L) - F_1(L_2 + L_3) = 0
\]
Applying Hooke’s law at points A and B allows $y_A$ and $y_B$ to be found in terms of force, stiffness, and respective locations.

$$y_B = \frac{F_B}{k_B} = \frac{F_1 L_1}{k_B L}$$

$$y_A = \frac{F_A}{k_A} = \frac{F_1 (L_2 + L_3)}{k_A L}$$

Substituting equations (7) and (8) into equations (3) and (4) provides the flexibility coefficients in terms of the desired variables.

$$y_1 = \left( \frac{L - L_1}{L} \right)^2 \frac{F_1}{k_A} + \frac{L_1^2 F_1}{L^2 k_B}$$

$$y_1 = \frac{1}{L^2} \left( \frac{(L - L_1)^2}{k_A} + \frac{L_1^2}{k_B} \right) = C_{11}$$

$$y_2 = \frac{L_3 (L - L_1)}{L^2 k_A} F_1 + \frac{L_1 (L_1 + L_2)}{L^2 k_B} F_1$$

$$y_2 = \frac{1}{L^2} \left( \frac{L_3 (L - L_1)}{k_A} + \frac{L_1 (L_1 + L_2)}{k_B} \right) = C_{21}$$

**Case 2**

$$F_1 = 0, F_2 = 1$$

By summing moments around point A, an expression for the resulting force at B is found.

$$\sum M_A = -F_B (L) + F_2 (L_1 + L_2) = 0$$
\[ F_B = \frac{F_2(L_1 + L_2)}{L} \]  \hspace{1cm} (9)

Summing moments around point B results in an expression for the force at A, shown in equation (10).

\[ \sum M_B = F_A(L) - F_2(L_3) = 0 \]

\[ F_A = \frac{F_2L_3}{L} \]  \hspace{1cm} (10)

Applying Hooke’s law at points A and B allows \( y_A \) and \( y_B \) to be found in terms of force, stiffness, and respective locations.

\[ y_A = \frac{F_A}{k_A} = \frac{F_2L_3}{k_A L} \]  \hspace{1cm} (11)

\[ y_B = \frac{F_B}{k_B} = \frac{F_2(L_1 + L_2)}{k_A Lk_B} \]  \hspace{1cm} (12)

Substituting equations (11) and (12) into equation (4) provides the flexibility coefficient in terms of the desired variables.

\[ (11) + (12) \rightarrow (4) \]

\[ C_{22} = y_2 = \frac{L_3^2 F_2}{L^2 k_A} + \frac{(L_1 + L_2)^2 F_2}{L^2 k_B} \]

\[ y_2 = \frac{1}{L^2} \left( \frac{L_3^2}{k_A} + \frac{(L_1 + L_2)^2}{k_B} \right) = C_{22} \]

And with the Maxwell-Betti Law of Reciprocal Deflections, an expression for \( C_{21} \) is found.

\[ C_{12} = C_{21} \]
Assembling each of the flexible influence coefficients in matrix form results in flexibility matrix for the bearings.

\[
C_b = \frac{1}{L^2} \begin{bmatrix}
\frac{(L - L_1)^2}{k_A} + \frac{L_1^2}{k_B} & \frac{L_3(L - L_1)}{k_A} + \frac{L_1(L_1 + L_2)}{k_B} \\
\frac{L_3(L - L_1)}{k_A} + \frac{L_1(L_1 + L_2)}{k_B} & \frac{L_3^2}{k_A} + \frac{(L_1 + L_2)^2}{k_B}
\end{bmatrix}
\]

To obtain the total flexibility matrix of the system, the matrices from the shaft and bearing flexibility coefficients are added together.

\[
C_{tot} = C_s + C_b
\]

For the total stiffness matrix, which is used in the equations of motion denoted by the term \(K\), the following relationship is used.

\[
K_{tot} = C_{tot}^{-1}
\]

Because of the size of the resulting stiffness matrix, its symbolic representation is omitted in this section.

The above matrices can be applied in the \(xz\) and \(yz\) plane, respectively, which leaves only the damping terms, eccentricity values, and initial phase values as the unknown parameters for each equation in (1). These are determined by referencing approximate starting values from other similar theoretical models (Wu, Naugle, & Meagher, 2016) and then by tuning each to best match the experimental results. For this model, the damping values in the equations of motion will be proportionally related to the bearing stiffness values and tuned to best match the experimental results. This assumption that the damping matrix is proportional to the stiffness matrix is to facilitate the process of obtaining an estimated value. This is a valid simplification if the
damping is extremely light in the system (Vance, 1988). The values used in both theoretical models will be described later when compared with the experimental results.

To obtain the desired displacement values from this system of equations, it is solved with the use of ode45 in MATLAB. With time values included as an input to the solver, the shaft’s rotational speeds and the disks’ phase values are computed using the same method as the real-time signal processing code described in section 3. With all of the required values obtained, the theoretical bode, polar, and cascade plots are able to be constructed.

5.3 Two-Disk Theoretical Model with Gyroscopic Forces

The theoretical model that does include gyroscopic effects also takes into consideration the presence of viscous forms of internal material damping. This is done by using the finite element method and Rayleigh beam theory to obtain the shape functions for analysis. This approach was inspired by a similar model that was used in the paper “Vibration control and stability analysis of rotor-shaft system with electromagnetic exciters” (Das, Nighil, Dutt, & Irretier, 2008). A more in depth explanation of the development and application of this model can be referenced there, but a brief description behind the formulation will be included here. The finite element model includes the bearings, disks, and shaft of the rotor. Each element in the discretized shaft has two nodes, with four degrees of freedom per node. Both disks are considered rigid elements with concentrated inertia properties and their respective mass unbalance values. Each disk is located at a specified node in the finite element mesh and can also have an initial skew angle specified. The bearings are also located at specified nodes, and are represented by their effective stiffness and damping values.
This finite element model is organized in MATLAB as class definition that uses an input file to
determine the geometry and parameters of the system. The input file used for the two-disk rotor
model focused on in this paper is shown in figure 31. Once again, like the first theoretical model
described, there are certain input parameters that need to be tuned independently by comparing
the results against the experimental data. This includes the mass unbalances, initial skew angles
(if any), and the bearing stiffness and damping values. The internal damping coefficient, \( \nu \), is set
to .0002, a value suggested and used by the reference that this finite element model was based
off of (Das, Nighil, Dutt, & Irretier, 2008).

```
%$$$$$$$$$$[Material #, length, diameter]$$$$$$$$$$$$$$$$$elesm = [ 1, .140, 0.01;
  1, .145, 0.01;
  1, .155, 0.01];

%$$$$$$$$$$[Material #, length, diameter, unbalance,   skew, node #]
disks = [ 1, 0.025, 0.075, 6.0e-5, 0.010, 2;
  1, 0.025, 0.075, 2.0e-5, -.018, 3];

%$$$$$$$$$$[     Type, node #]$$$$$$$$$$$$$$$$$bears = [ 1, 1;
  2, 4];

%$$$$$$$$$$[     ID #,   F,   rho,   nUV]$$$$$$$$$$$$mate = [ 1, 209e9, 7700, 0.0002];

%$$$$$$$$$$[     ID #,   Kx,   Ky,   Cx,   Cy]$$$$$$$$$beartypes = [ 1, 1.26e5, 1.2e5, 94, 90;
  2, 1.25e5, 1.2e5, 94, 90];

DOF = 4;
```

Figure 31. MATLAB input file used to construct the finite element model of a two-disk rotor
with gyroscopic effects and internal damping.
5.4 Identifying system parameters from experimental data

As mentioned in the descriptions of the theoretical models, there are certain theoretical parameters that need to be determined through independent tuning so that their results best match the experimental data. The two-disk rotor system used for the collection of experimental data described in this section is shown in figure 32.

Figure 32. Experimental set up of two-disk, Bently Nevada rotor kit.

The rotor kit used is the Bently Nevada RK4 model which is briefly described in section 2 and annotated in more detail in figure 1. The shaft material is stainless steel, and has a diameter of 1 cm. The two disks installed on the shaft are also stainless steel, have a diameter of 7 cm, a thickness of 2.5 cm, and were not installed with any additional weights. The distances annotated from left to right in figure 32 correspond to the lengths $L_1$, $L_2$, and $L_3$, respectively, in the stiffness matrices present in the theoretical models. An overview of these parameters is given in Table 1 below.
Table 2. Physical geometry and material descriptions of two-disk rotor system.

<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L_1$ (Bearing 1 to Disk 1)</td>
<td>14.0</td>
<td>cm</td>
</tr>
<tr>
<td>$L_2$ (Disk 1 to Disk 2)</td>
<td>14.5</td>
<td>cm</td>
</tr>
<tr>
<td>$L_3$ (Disk 2 to Bearing 2)</td>
<td>15.5</td>
<td>cm</td>
</tr>
<tr>
<td>$L$ (Total Shaft Length)</td>
<td>44.0</td>
<td>cm</td>
</tr>
<tr>
<td>Shaft Diameter</td>
<td>1</td>
<td>cm</td>
</tr>
<tr>
<td>Disk Diameter</td>
<td>7</td>
<td>cm</td>
</tr>
<tr>
<td>Disk Thickness</td>
<td>2.5</td>
<td>cm</td>
</tr>
<tr>
<td>Shaft Material</td>
<td>Stainless Steel</td>
<td>N/A</td>
</tr>
<tr>
<td>Disk Material</td>
<td>Stainless Steel</td>
<td>N/A</td>
</tr>
</tbody>
</table>

For the two-disk ramp-up experiment, the motor speed control box was set to a constant acceleration of 3500 rpm/min and a maximum speed of 7000 rpm. The MATLAB data acquisition program was set to record the data at a sampling rate of 20,000 samples/sec and a window size of 5000 samples. Five channels were connected to the National Instruments data card module: the horizontal and vertical transducers for each disk and one transducer focusing on the Keyphasor. Bode, polar, and cascade plots were all selected to be produced in real time for each channel.

Since there is an ability to represent any possible anisotropy present in the bearings in the theoretical models, comparing the horizontal and vertical data from the experiment would give insight into the correct $K_X/K_Y$ ratio of the stiffness values. Figure 33 shows the experimental amplitude responses for both disks during the ramp-up test.
Based on the amplitude responses shown in figure 33, the two peaks occurring within the speed range of 0 to 7000 rpm vary in amplitude between the horizontal and vertical channels, but not in location. This suggests that the amount of anisotropy in the stiffness of the system is very small, since a different stiffness value for each plane would affect the rpm location of each peak. The difference in amplitudes between the horizontal and vertical channel is important to note, and is effected by another parameter described later in this section. Figure 34 shows the effect that varying $K_x/K_y$ ratios would have on the results of the horizontal and vertical responses. This plot was produced using the first theoretical model described in the previous section. It can be
seen that within the ratio span of about 0.88 to 1.13, differences in the amplitude responses between the horizontal and vertical directions are very small. Since the experimental horizontal and vertical amplitude responses are approximately the same in terms of peak location, we can assume that the $K_x/K_y$ ratio falls within this range very close to one. A more accurate ratio will be determined later when the cascaded full spectrum plots are described.

Figure 34. 3D plot illustrating the effect that varying $K_x/K_y$ ratios have on the responses of the horizontal and vertical displacements.

Because the anisotropy ratio of the bearing stiffness in this two-disk system is assumed to be very close to one, results concerning only the horizontal channels will be focused on for amplitude and phase to limit repetitiveness. From the experimental results seen in figure 33, an important characteristic to note is the difference in peak amplitudes between the two disks. This can best be seen in the second or higher speed, natural frequency peak in the graphs. In regards to the theoretical parameters of the system, the eccentricities of each disk would be a contributor to the different amplitude levels. For the first theoretical model described (without gyroscopic effects), the eccentricities used for disk 1 and disk 2 to best match the experimental results were
6.5 \times 10^{-5} \, m \text{ and } 2 \times 10^{-5} \, m, \text{ respectively. These do not necessarily represent the actual eccentricities in the disks, and could also be a result of other irregularities in the shaft and bearings. The initial bearing stiffness value estimate was taken from another study done on a similar system, which determined it to be 120,000 \, N/m (Wu, Naugle, & Meagher, 2016). The final stiffness values used are discussed in the 3D full spectrum portion of this section. The damping values affected both the peaks’ amplitudes and shapes, and with tuning were determined to be 44 \, Ns/m for } d_x \text{ and } 42 \, Ns/m for } d_y. \text{ Figures 35 and 36 show the horizontal amplitude and phase comparisons between the experimental results and results obtained from the theoretical model without gyroscopic forces.}
Figure 35. Horizontal displacement results from a two-disk ramp-up experiment and two-disk theoretical model without gyroscopic forces.
Figure 36. Phase results from a two-disk ramp-up experiment and two-disk theoretical model without gyroscopic forces.

If the peak locations in the amplitude responses of the theoretical model are tuned properly, the phase shifts will also occur at the correct shaft speed. The offset and trend of the phase values however, directly depend on the initial phase angles that are input into the model. In the theoretical model’s system of equations, these are designated as $\varphi_1$ and $\varphi_2$. With the same method of tuning and comparing the theoretical and experimental results, the correct initial phase values of $\varphi_1$ and $\varphi_2$ were determined to be $205^\circ$ and $305^\circ$, respectively.

The cascaded full spectrum plots use amplitude, phase, and shaft rotation data just like the previous plots, but also shows the frequency vibrations which provides more information on the
actual parameters of the system. Using the same methods described in “A Full Spectrum Analysis Methodology Applied to an Anisotropic Overhung Rotor” (Wu, Naugle, & Meagher, 2016), experimental and theoretical 3D full spectrum plots were generated through MATLAB. Figure 37 shows the 3D cascading full spectrum plot produced from the MATLAB obtained experimental results of disk 2.

![3D cascade plot](image)

Figure 37. Experimental 3D cascade plot generated from the data of disk 2 on the two-disk rotor.

An important characteristic to note in this 3D full spectrum plot is the presence of smaller peaks in the negative frequency domain. Earlier when the $K_x/K_Y$ ratio was introduced, an approximation that was drawn from observing from figure 34 was that this ratio was very close to the value of one for this system. However, when this ratio is set equal to one for the theoretical model, in other words $K_x = K_Y$, there is no response in the negative frequency domain. Knowing
this, the $K_X/K_Y$ ratio was tuned in the theoretical model within the range of approximately 0.88 to 1.13, as determined earlier. The most accurate $K_X/K_Y$ ratio for this system was found to be 1.05, with $K_X = 126,000 \, N/m$ and $K_Y = 126,000 \, N/m$. Figure 38 shows the 3D cascaded full spectrum plot generated from the theoretical results of disk 2.

![3D Cascaded Full Spectrum Plot](image)

**Figure 38.** Theoretical 3D cascade plot generated from model without gyroscopic effects.

Using the parameters estimated through the theoretical model that did not take gyroscopic effects into consideration, the model with gyroscopic effects was compared with the experimental data. After tuning the second model slightly, new values were found to better match the experimental data. With the internal damping coefficient set equal to .0002, as stated earlier in the description of the theoretical model, the respective $d_x$ and $d_y$ damping values appropriate for this model
were determined to be $94 \, Ns/m$ and $90 \, Ns/m$. The eccentricity values changed only slightly for disk 1 when compared with the first theoretical model. The new values were $6 \times 10^{-5} \, m$ and $2 \times 10^{-5} \, m$, respectively, for disk 1 and disk 2. Parameters unique to this model, because of the included gyroscopic forces, are the initial skew angles for each of the disks on the shaft. This initial skew angle would be a combination of an imperfect alignment to the axial direction of the shaft and the slight angle that is caused by the weight of the two disks sitting between the two fixed bearings. With this information, the initial skew angles would most likely be very small (within one to two degrees) and opposite in their signs. The initial skew angles for disk 1 and disk 2 were determined to be $0.010$ and $-0.018 \, \text{radians}$, respectively. This FEM model does not include the computation of phase angles at different positions along the discretized shaft, so only the amplitude responses will be compared here. The results from the theoretical model including gyroscopic effects can be seen compared against the experimental results in figure 39.
Figure 39 shows that the theoretical model that includes gyroscopic forces matches the trends of both graphs very well up until the second peak for disk 1. The large difference in amplitudes between the second peaks of disk 1 and disk 2 could not be repeated with this theoretical model. A possible explanation for this is the location of the sensors that were installed along the span of the shaft for the two-disk experiment. Figure 32 shows how much farther away the set of orthogonal transducers are from the location of disk 1 when compared to disk 2. This could result in an inaccurate representation of the second peak because of the shape of the second mode of the shaft during rotation. Figure 40 illustrates the first and second mode shapes associated with rotational natural frequencies of a shaft. The second mode could have a larger difference in
displacement magnitude over a specified shaft distance compared to the first mode. If the first set of orthogonal sensors were located near one of the antinodes in the second mode, a small change in axial location could contribute to the difference in amplitudes between the theoretical and experimental results. A possible solution to investigating this behavior further would be to discretize the shaft into more elements and observe the behavior at a node closer to the sensor location. Since the model uses Rayleigh elements however, only slender elements produce accurate results. Introducing more elements would require Timoshenko beams that account for shear effects in addition to rotary inertia (Friswell, 2010).

Figure 40. First and second mode shapes for a beam between two fixed points (Turbomachinery Balancing Basics, 2008).

With data obtained from the experiment and first theoretical model, the orbits in each of the systems are able to be plotted at any speed within the ramp-up range. To give a clearer understanding of how the shaft is behaving throughout the speed range, 3D orbit plots can be generated with rotational velocity along the x axis, and horizontal and vertical amplitudes in the yz plane. Figures 41 and 42 show the experimental and theoretical 3D orbit plots resulting from the two-disk system.
Figure 41. MATLAB generated experimental 3D orbit plot.
Figure 42. Theoretical 3D orbit plot without gyroscopic effects.
6. DISCUSSION AND CONCLUSIONS

The MATLAB signal processing program and user interface application developed includes many of the same input options that Bently Nevada’s ADRE software does to control the data acquisition of rotating machinery. Using only a commercially available off-the-shelf data acquisition card that interfaces directly with MATLAB, the cost of resources needed is dramatically reduced. The MATLAB based program uses the same calculation methods applied in “A Full Spectrum Analysis Methodology Applied to an Anisotropic Overhung Rotor” (Wu, Naugle, & Meagher, 2016). The real-time methods introduced in this paper proved to be an efficient way of streaming and saving data from rotordynamic tests. Experimental amplitude, phase, and speed information that is generated in real-time with the MATLAB program has its accuracy verified when compared with the results produced with the ADRE system. The bode, polar, full spectrum, and orbit plots all agree with those displayed in the ADRE application within a small percent error and contain additional post-processing capabilities as a result of being based in the MATLAB environment. A downside that comes with using this MATLAB based application is the computational resources that are available while in use. Since the allocation of processing power varies depending on the computer installed with the program, the time lag between the physical test and data being displayed is not a constant value. Because the ADRE system is a standalone unit, it can outperform the MATLAB program in display speed at higher rotational velocities and when three dimensional plots are requested.

Having the different plots available from the experiment allowed for the unknown physical parameters to be determined through the theoretical models. With two different types of theoretical models for the two-disk rotor, the amount of influence that each parameter had on the
behavior could be gauged. The theoretical model that did not include gyroscopic effects solved the two-disk’s system of ordinary differential equations with ode45 in MATLAB. This model was used to estimate the initial phase angle of each disk, and the effective stiffness and damping values for each bearing. The initial phase angles, $\varphi_1$ and $\varphi_2$, were determined by observing the changes in the phase lag plot and aligning the theoretical trend with the one produced from the experiment. The stiffness values were first estimated by matching the first two natural frequencies of the system. The anisotropy ratio, $K_X/K_Y$, was then finalized to a more accurate degree by using the characteristics of the cascaded full spectrum plots. This ratio, estimated at $K_X/K_Y = 1.05$, showed that there was limited anisotropy in the bearings of the two-disk rotor system. The finite element model discretized the rotating shaft into 3 Rayleigh beam elements, and produced displacement data for different locations while taking the effects of gyroscopic forces and viscous internal damping into account. Using the suggested internal damping constant of $0.002 \, \text{s}$ (Das, Nighil, Dutt, & Irretier, 2008), the initial skew angles of each disk were able to be tuned until the theoretical model best matched the experimental results. Each theoretical model provides insight into some of the physical parameters of a rotordynamic system that are not easily determined through direct measurement.
REFERENCES


72


classdef TestObject < handle
    properties (SetObservable)
        xfilt
        yfilt
        Xtemp
        Ytemp
        Xtemp2
        Ytemp2
        Xamp
        Yamp
        Xamp2
        Yamp2
        xchop
        ychop
        xchop2
        ychop2
        Tamp
        Tamp2
        Samp
        Samp2
        SPhase
        SPhase2
        X
        Y
        X2
        Y2
        ref
        Time
        speed
        Phase

        pcoord

        options = struct('signals',0,'orbit',0,'bode',0,'polar',0,'cascade',0);
        filtering = 0;
        n=1; %filter synchronous multiplier: 1X, 2X, 3X, ... nX the running speed
        nchannels = 3;
        refch = 0;
        Xch = 1;
        Ych = 2;
        Xch2 = 3;
        Ych2 = 4;
sprobe = 5; %transducer shaft constant [mil/volts]
DeviceID = 'cDAQ1Mod1'
s %DAQ session
g %Gap session
xgap=0;
ygap=0;
speedlimit = 4000; %Set rpm limit for bode plot graph
srate = 20000; % Samples/second
duration = 10; % Duration of session in seconds
notify = 10000; % Window size
flag
end
methods
function measureGap(obj)
disp('Measuring Gap')
daqreset % Reset to prevent bugs
obj.g = daq.createSession(['ni']);
obj.g.Rate = 2;
obj.g.DurationInSeconds = 300;
lis1 = addlistener(obj.g, 'DataAvailable', @obj.sendData); % See
https://www.mathworks.com/help/daq/ref/dataavailable.html
obj.g.NotifyWhenDataAvailableExceeds = 1; % Controls when
DataAvailable event fires
for i = 0:(obj.nchannels-1)
addAnalogInputChannel(obj.g,obj.DeviceID, i, 'Voltage');
end
obj.g.startBackground() % Starts data acquisition in background
obj.g.wait() % Waits until data acquisition is complete
delete(lis1); % Deletes listeners
release(obj.g)
end
function sendData(obj,~,event)
obj.xgap = event.Data(1,obj.Xch+1);
obj.ygap = event.Data(1,obj.Ych+1);
end
function startDAQ(obj)
disp('Starting DAQ')
daqreset % Reset to prevent bugs
obj.s = daq.createSession(['ni']);
obj.s.Rate = obj.srate;
obj.s.DurationInSeconds = obj.duration;
lis1 = addlistener(obj.s, 'DataAvailable', @obj.plotData); % See
https://www.mathworks.com/help/daq/ref/dataavailable.html
obj.s.NotifyWhenDataAvailableExceeds = obj.notify; % Controls when DataAvailable event fires

for i = 0:(obj.nchannels-1)
  if i<=3
    addAnalogInputChannel(obj.s, obj.DeviceID, i, 'Voltage');
  else
    addAnalogInputChannel(obj.s, 'cDAQ1Mod2', i-4, 'Voltage');
  end
end
tic
obj.s.startBackground() % Starts data acquisition in background
obj.s.wait() % Waits until data acquisition is complete
toc
delete(lis1); % Deletes listeners
end

function plotData(obj,~,event) % (AffectedObject, Source, EventName)
  persistent tempData ychop xchop tempTime SRYcon_X SRXcon_X
  SRXcon_X2 SRYcon_X2 xf yf;
  % Data Organizing
  if (isempty(tempData))
    tempData = [];
    tempTime = [];
    obj.xchop = [];
    obj.ychop = [];
  end
  filtData = zeros(obj.notify, obj.nchannels); % Preallocating
  filtData
  for i=1:obj.nchannels
    filtData(:,i) = event.Data(:,i) - mean(event.Data(:,i));
    % Subtracts dc offset
  end
  obj.Xtemp = filtData(:,obj.Xch+1); % Temporary data for current window
  obj.Ytemp = filtData(:,obj.Ych+1); % Temporary data for current window
  obj.Xtemp2 = filtData(:,obj.Xch2+1); % Temporary data for current window
  obj.Ytemp2 = filtData(:,obj.Ych2+1); % Temporary data for current window
  tempData = [tempTime tempData; event.TimeStamps filtData];
  obj.xchop = [ obj.xchop, filtData(:,obj.Xch+1)]; % Data chopped into separate columns based on window
  obj.ychop = [ obj.ychop, filtData(:,obj.Ych+1)]; % Data chopped into separate columns based on window
  obj.xchop2 = [ obj.xchop, filtData(:,obj.Xch+1)]; % Data chopped into separate columns based on window
  obj.ychop2 = [ obj.ychop, filtData(:,obj.Ych+1)]; % Data chopped into separate columns based on window
  data = tempData;
  obj.Time = data(:,1); % Stores data into object properties
  obj.ref = data(:,obj.refch+2); % Concatenated data
  obj.X = data(:,obj.Xch+2); % Concatenated data
  obj.Y = data(:,obj.Ych+2); % Concatenated data
  obj.X2 = data(:,obj.Xch2+2); % Concatenated data
  obj.Y2 = data(:,obj.Ych2+2); % Concatenated data
  obj.flag = floor(length(obj.Y)/obj.notify); % Index for window

76
%% Signal Plotting
h = figure(1);
h.Position = [1114 150 700 800]; % [left bottom width height]

if obj.options.signals==1
    subplot(4,2,1) %Channel 0 data
    plot(data(:,1),data(:,2))
    hold on
    plot(data(:,1),data(:,3)) %Channel 1 data
    hold on
    plot(data(:,1),data(:,4)) %Channel 2 data
    hold on
    plot(data(:,1),data(:,5)) %Channel 3 data
    hold on
    plot(data(:,1),data(:,6)) %Channel 4 data
    hold off
    title('Channel Signals')
    legend('0','1','2')
    xlabel('Time (sec)')
    xlim([0 obj.duration])
end

%% Speed & Amp Calc
refchop = filtData(:,obj.refch+1);
for i = 1:length(refchop) %Removes negative spikes
    if refchop(i)<0
        refchop(i)=0;
    end
end
pp = abs(pulseperiod(refchop,obj.srate,'Tolerance',7));
dt = mean(pp);
if isempty(obj.speed)
    obj.speed = [] ; %REVISIT Preallocate
    obj.Xamp = [] ;
    obj.Yamp = [] ;
    obj.Xamp2 = [] ;
    obj.Yamp2 = [] ;
    obj.Tamp = [] ;
    obj.Tamp2 = [] ;
end
xval = (max(obj.Xtemp)-min(obj.Xtemp)).*obj.sprobe; %X amplitude
yval = (max(obj.Ytemp)-min(obj.Ytemp)).*obj.sprobe; %Y amplitude
xval2 = (max(obj.Xtemp2)-min(obj.Xtemp2)).*obj.sprobe; %X amplitude
yval2 = (max(obj.Ytemp2)-min(obj.Ytemp2)).*obj.sprobe; %Y amplitude

obj.Xamp = [obj.Xamp; xval ]; %Calculates and stores amplitude
obj.Yamp = [obj.Yamp; yval ]; %Calculates and stores amplitude
obj.Xamp2 = [obj.Xamp2; xval2 ]; %Calculates and stores amplitude
obj.Yamp2 = [obj.Yamp2; yval2 ]; %Calculates and stores amplitude
obj.Tamp = [obj.Tamp; sqrt((xval.^2)+yval.^2)]; %Total amplitude
obj.Tamp2 = [obj.Tamp2; sqrt((xval2.^2)+yval2.^2)];  \%Total amplitude

obj.speed = [obj.speed; (1./dt).*60];  \%Calculates and stores rpm

if obj.options.bode == 1
    subplot(5,2,[3,4])
    plot(obj.speed,obj.Tamp)  \% plot of amplitude vs speed
title('Bode Plot')
xlabel('RPM')
ylabel('Amplitude (milPP)')
xlim([0 obj.speedlimit])

    subplot(5,2,[7,8])
    plot(obj.speed,obj.Tamp2)  \% plot of amplitude vs speed
title('Bode Plot 2')
xlabel('RPM')
ylabel('Amplitude (milPP)')
xlim([0 obj.speedlimit])
end

\% Filter Data
if obj.filtering == 1
    Fpass1 = obj.speed(obj.flag)*obj.n/60-5;  \%Upper frequency of bandpass filter
    Fpass2 = obj.speed(obj.flag)*obj.n/60+5;  \%Lower frequency of bandpass filter

    if obj.flag == 1
        h  = fdesign.bandpass('N,Fp1,Fp2,Ast1,Ap,Ast2', 10, Fpass1, Fpass2, 50, .1, 50, obj.srate);
        Hd(obj.flag) = design(h, 'ellip');
        Hd(obj.flag).persistentmemory = true;
        obj.xfilt(:,obj.flag) = filter(Hd(obj.flag),obj.Xtemp);
        xf = Hd(obj.flag).states;
        reset(Hd(obj.flag));
        obj.yfilt(:,obj.flag) = filter(Hd(obj.flag),obj.Ytemp);
        yf = Hd(obj.flag).states;
    else
        h  = fdesign.bandpass('N,Fp1,Fp2,Ast1,Ap,Ast2', 10, Fpass1, Fpass2, 50, .1, 50, obj.srate);
        Hd(obj.flag) = design(h, 'ellip');
        Hd(obj.flag).persistentmemory = true;
        Hd(obj.flag).states = xf;
        obj.xfilt(:,obj.flag) = filter(Hd(obj.flag),obj.Xtemp);
        xf = Hd(obj.flag).states;
        yf = Hd(obj.flag).states;
        Hd(obj.flag).states = yf;
        obj.yfilt(:,obj.flag) = filter(Hd(obj.flag),obj.Xtemp);
        yf = Hd(obj.flag).states;
    end
end

\% Phase Calc
distance = .5*obj.srate/100;
threshold = .5*(max(refchop) - mean(refchop));
function [~, locsref] = findpeaks(refchop, 'MinPeakHeight', threshold, 'MinPeakDistance', distance);
    [~, locsX] = findpeaks(filtData(:,2), 'MinPeakProminence', .1, 'MinPeakDistance', distance);
    [~, locsY] = findpeaks(filtData(:,3), 'MinPeakProminence', .1, 'MinPeakDistance', distance);
    [~, locsX2] = findpeaks(filtData(:,4), 'MinPeakProminence', .1, 'MinPeakDistance', distance);
    [~, locsY2] = findpeaks(filtData(:,5), 'MinPeakProminence', .1, 'MinPeakDistance', distance);
    if length(locsref) < 2 || length(locsX) < 2 || length(locsY) < 2
        obj.Phase.PhaseX(obj.flag) = NaN;
        obj.Phase.PhaseY(obj.flag) = NaN;
    else
        if length(locsX) ~= length(locsY)
            if length(locsX) < length(locsY)
                if abs(locsX(1) - locsY(1)) < abs(locsX(end) - locsY(end))
                    locsY(end) = [];
                else
                    locsY(1) = [];
                end
            else
                if abs(locsX(1) - locsY(1)) < abs(locsX(end) - locsY(end))
                    locsX(end) = [];
                else
                    locsX(1) = [];
                end
            end
        end
        if locsref(1) > locsX(1) || locsref(1) > locsY(1)
            locsX(1) = [];
            locsY(1) = [];
        end
        if length(locsref) > length(locsX)
            locsref(length(locsX):end) = [];
        end
        phx = zeros(length(locsref)-1,1);
        phy = zeros(length(locsref)-1,1);
        for j = 1:length(locsref)-1
            phx(j) = mod((locsX(j) - locsref(j))/(locsref(j+1) - locsref(j))*2*pi,2*pi);
            phy(j) = mod((locsY(j) - locsref(j))/(locsref(j+1) - locsref(j))*2*pi,2*pi);
        end
        obj.Phase.PhaseX(obj.flag) = 180/pi*mod(mean(phx(:)),2*pi); % REVISIT flag indexing
        obj.Phase.PhaseY(obj.flag) = 180/pi*mod(mean(phy(:)),2*pi); % REVISIT flag indexing
    end
    %%%%%%%%%%%%%%%%% 2nd phase calc %%%%%%%
    if length(locsref) < 2 || length(locsX2) < 2 || length(locsY2) < 2
        obj.Phase.PhaseX2(obj.flag) = NaN;
        obj.Phase.PhaseY2(obj.flag) = NaN;
    else
if length(locsX2) ~= length(locsY2)
    if length(locsX2) < length(locsY2)
        if abs(locsX2(1) - locsY2(1)) < abs(locsX2(end) - locsY2(end))
            locsY2(end) = [];
        else
            locsY2(1) = [];
        end
    else
        if abs(locsX2(1) - locsY2(1)) < abs(locsX2(end) - locsY2(end))
            locsX2(end) = [];
        else
            locsX2(1) = [];
        end
    end
else
    if abs(locsX2(1) - locsY2(1)) < abs(locsX2(end) - locsY2(end))
        locsX2(end) = [];
    else
        locsX2(1) = [];
    end
end
if locsref(1) > locsX2(1) || locsref(1) > locsY2(1)
    locsX2(1) = [];
    locsY2(1) = [];
end
if length(locsref) > length(locsX2)
    locsref(length(locsX2):end) = [];
end
phx2 = zeros(length(locsref)-1,1);
phy2 = zeros(length(locsref)-1,1);
for j = 1:length(locsref)-1
    phx2(j) = mod((locsX2(j) - locsref(j))/(locsref(j+1) - locsref(j))*2*pi,2*pi);
    phy2(j) = mod((locsY2(j) - locsref(j))/(locsref(j+1) - locsref(j))*2*pi,2*pi);
end
obj.Phase.PhaseX2(obj.flag) = 180/pi*mod(mean(phx2(:)),2*pi);
% REVISIT flag indexing
obj.Phase.PhaseY2(obj.flag) = 180/pi*mod(mean(phy2(:)),2*pi);
% REVISIT flag indexing

end

if obj.options.bode==1
    subplot(5,2,[5,6])
    plot(obj.speed,obj.Phase.PhaseX)
    h(gca;
    h.YDir = 'reverse';
    ylim([0 360])
    xlabel('RPM')
    ylabel('Phase Lag (degrees)')
    xlim([0 obj.speedlimit])
    subplot(5,2,[9,10])
    plot(obj.speed,obj.Phase.PhaseX2)
    h(gca;
    h.YDir = 'reverse';
    ylim([0 360])
    xlabel('RPM')
    ylabel('Phase Lag (degrees)')
    xlim([0 obj.speedlimit])
end

%% Slow Roll Comp
if isempty(SRXcon_X)
if isnan(obj.Phase.PhaseX(obj.flag)) || isnan(obj.Xamp(obj.flag))
    obj.Samp(obj.flag) = NaN;
    obj.SPhase(obj.flag) = NaN;
else
    SRXcon_X = obj.Xamp(obj.flag)*cosd(obj.Phase.PhaseX(obj.flag));
    SRYcon_X = obj.Xamp(obj.flag)*sind(obj.Phase.PhaseX(obj.flag));
    Xvec_X = obj.Xamp(obj.flag)*cosd(obj.Phase.PhaseX(obj.flag));
    Yvec_X = obj.Xamp(obj.flag)*sind(obj.Phase.PhaseX(obj.flag));
    Xvec_comp = Xvec_X - SRXcon_X;
    Yvec_comp = Yvec_X - SRYcon_X;
    obj.Samp(obj.flag) = sqrt(Xvec_comp^2+Yvec_comp^2);
    obj.SPhase(obj.flag) = atan2(Yvec_comp,Xvec_comp);
end
else
    Xvec_X = obj.Xamp(obj.flag)*cosd(obj.Phase.PhaseX(obj.flag));
    Yvec_X = obj.Xamp(obj.flag)*sind(obj.Phase.PhaseX(obj.flag));
    Xvec_comp = Xvec_X - SRXcon_X;
    Yvec_comp = Yvec_X - SRYcon_X;
    obj.Samp(obj.flag) = sqrt(Xvec_comp^2+Yvec_comp^2);
    obj.SPhase(obj.flag) = atan2(Yvec_comp,Xvec_comp);
end

%%%%%%% 2nd slow roll comp%%%%%%%
if isempty(SRXcon_X2)
if isnan(obj.Phase.PhaseX2(obj.flag)) || isnan(obj.Xamp2(obj.flag))
    obj.Samp2(obj.flag) = NaN;
    obj.SPhase2(obj.flag) = NaN;
else
    SRXcon_X2 = obj.Xamp2(obj.flag)*cosd(obj.Phase.PhaseX2(obj.flag));
    SRYcon_X2 = obj.Xamp2(obj.flag)*sind(obj.Phase.PhaseX2(obj.flag));
    Xvec_X2 = obj.Xamp2(obj.flag)*cosd(obj.Phase.PhaseX2(obj.flag));
    Yvec_X2 = obj.Xamp2(obj.flag)*sind(obj.Phase.PhaseX2(obj.flag));
    Xvec_comp2 = Xvec_X2 - SRXcon_X2;
    Yvec_comp2 = Yvec_X2 - SRYcon_X2;
    obj.Samp2(obj.flag) = sqrt(Xvec_comp2^2+Yvec_comp2^2);
    obj.SPhase2(obj.flag) = atan2(Yvec_comp2,Xvec_comp2);
end
else
    Xvec_X2 = obj.Xamp2(obj.flag)*cosd(obj.Phase.PhaseX2(obj.flag));
    Yvec_X2 = obj.Xamp2(obj.flag)*sind(obj.Phase.PhaseX2(obj.flag));
    Xvec_comp2 = Xvec_X2 - SRXcon_X2;
    Yvec_comp2 = Yvec_X2 - SRYcon_X2;
    obj.Samp2(obj.flag) = sqrt(Xvec_comp2^2+Yvec_comp2^2);
    obj.SPhase2(obj.flag) = atan2(Yvec_comp2,Xvec_comp2);
end

%%% Polar Plotting
obj.pcoord = NaN(obj.srate*obj.duration/obj.notify,8); %Preallocation for polar coordinates
for i = 1:length(obj.Xamp)
    obj.pcoord(i,1) = obj.Phase.PhaseX(i)*(pi/180);
    obj.pcoord(i,2) = obj.Xamp(i);
    obj.pcoord(i,3) = obj.SPhase(i);
    obj.pcoord(i,4) = obj.Samp(i);
    obj.pcoord(i,5) = obj.Phase.PhaseX2(i)*(pi/180);
    obj.pcoord(i,6) = obj.Xamp2(i);
    obj.pcoord(i,7) = obj.SPhase2(i);
obj.pcoord(i,8) = obj.Samp2(i);
end

if obj.options.polar ==1
 subplot(5,2,1)
polarplot(obj.pcoord(:,1),obj.pcoord(:,2),'r',obj.pcoord(:,3),obj.pcoord(:,4),'
'b') %Plot Tamp WRT PhaseX
 subplot(5,2,2)
polarplot(obj.pcoord(:,5),obj.pcoord(:,6),'r',obj.pcoord(:,7),obj.pcoord(:,8),'
'b') %Plot Tamp WRT PhaseX
end

%% Orbit Plot
 if obj.options.orbit ==1
 subplot(4,2,2)
 for i=1:(length(locsref)-1)
   gap = fix((1/10)*(locsref(i+1)-locsref(i)));
   plot(filtData(locsref(i):(locsref(i+1)-
   gap),2),filtData(locsref(i):(locsref(i+1)-
gap),3),filtData(locsref(i),2),filtData(locsref(i),3),'r', 'markersize',40)
   title('Orbit Plot')
   axis square
   drawnow limitrate
 end

%% Cascade Plotting
 df = obj.srate/obj.notify;
 f = df*(0:obj.notify-1);
 Q = ceil((obj.notify+1)/2);
 fQ = df*(Q-1);
 freq = f-fQ;

 Zwin = hanning(obj.notify, 'Periodic');
 Z = obj.xchop + 1i*obj.ychop;
 Z(:,obj.flag) = Zwin.*Z(:,obj.flag);
 Zf = 2*abs(fft(Z)/obj.notify);
 Zf = fftshift(Zf',2); %Shifts graph to center on the x axis

if obj.filtering ==1
 Z2 = obj.xfilt + 1i*obj.yfilt;
 Z2(:,obj.flag) = Zwin.*Z2(:,obj.flag);
 Z2f = 2*abs(fft(Z2)/obj.notify);
 Z2f = fftshift(Z2f',2); %Shifts graph to center on the x axis
end

if obj.options.cascade ==1
 subplot(4,2,8)
 waterfall(freq, obj.speed, Zf)
 xlabel('Frequency of Vibration (Hz)');
 ylabel('Running obj.speed (RPM)');
 zlabel('Amplitude of Vibration (mils)');
 axis([-100 100 -inf inf 0 inf])
end

if obj.filtering ==1
 figure(6)
waterfall(freq, obj.speed, Z2f)
xlabel('Frequency of Vibration (Hz) - Filtered');
ylabel('Running obj.speed (RPM)');
zlabel('Amplitude of Vibration (mils)');
axis([-100 100 -inf inf 0 inf])
end
end

%% Peakfinder troubleshooting
% figure(obj.flag)
% plot(filtData(:,1))
% hold on
% plot(locsref,filtData(locsref,1),'o')
% plot(filtData(:,2))
% plot(filtData(:,3))
% plot(locsX,filtData(locsX,2),'o')
% plot(locsY,filtData(locsY,3),'o')
end
end
end
APPENDIX B

NI 9215 Datasheet
The NI 9215 is an analog input module for use with NI CompactDAQ and CompactRIO systems. The NI 9215 includes four simultaneously sampled analog input channels and successive approximation register (SAR) 16-bit analog-to-digital converters (ADCs). The NI 9215 contains NIST-traceable calibration, a channel-to-earth ground double isolation barrier for safety and noise immunity, and high common-mode voltage range.

- BNC, screw-terminal, or spring-terminal connectivity
- 250 Vrms, CAT II, channel-to-earth isolation (screw terminal); 60 VDC, CAT I, channel-to-earth isolation (BNC)
## C SERIES ANALOG INPUT MODULE COMPARISON

<table>
<thead>
<tr>
<th>Product Name</th>
<th>Signal Levels</th>
<th>Channels</th>
<th>Sample Rate</th>
<th>Simultaneous</th>
<th>Resolution</th>
<th>Connectivity</th>
</tr>
</thead>
<tbody>
<tr>
<td>NI 9201</td>
<td>±10 V</td>
<td>8 Single-Ended</td>
<td>500 kS/s</td>
<td>No</td>
<td>12-Bit</td>
<td>Screw-Terminal, Spring-Terminal, DSUB</td>
</tr>
<tr>
<td>NI 9205</td>
<td>±200 mV, ±1 V, ±5 V, ±10 V</td>
<td>32 Single-Ended, 16 differential</td>
<td>250 kS/s</td>
<td>No</td>
<td>16-Bit</td>
<td>Spring-Terminal, DSUB</td>
</tr>
<tr>
<td>NI 9206</td>
<td>±200 mV, ±1 V, ±5 V, ±10 V</td>
<td>32 Single-Ended, 16 differential</td>
<td>250 kS/s</td>
<td>No</td>
<td>16-Bit</td>
<td>Spring-Terminal, DSUB</td>
</tr>
<tr>
<td>NI 9207</td>
<td>±10 V</td>
<td>8 Differential</td>
<td>500 kS/s</td>
<td>No</td>
<td>24-Bit</td>
<td>DSUB</td>
</tr>
<tr>
<td>NI 9209</td>
<td>±10 V</td>
<td>20 Single-Ended, 16 Differential</td>
<td>500 kS/s</td>
<td>No</td>
<td>24-Bit</td>
<td>DSUB</td>
</tr>
<tr>
<td>NI 9215</td>
<td>±10 V</td>
<td>4 Differential</td>
<td>100 kS/s/ch</td>
<td>Yes</td>
<td>16-Bit</td>
<td>Screw-Terminal, Spring-Terminal, BNC</td>
</tr>
<tr>
<td>NI 9220</td>
<td>±10 V</td>
<td>16 Differential</td>
<td>100 kS/s/ch</td>
<td>Yes</td>
<td>16-Bit</td>
<td>Spring-Terminal, DSUB</td>
</tr>
<tr>
<td>NI 9221</td>
<td>±60 V</td>
<td>8 Single-Ended</td>
<td>800 kS/s</td>
<td>No</td>
<td>12-Bit</td>
<td>Screw-Terminal, Spring-Terminal, DSUB</td>
</tr>
<tr>
<td>NI 9222</td>
<td>±10 V</td>
<td>4 Differential</td>
<td>500 kS/s/ch</td>
<td>Yes</td>
<td>16-Bit</td>
<td>Screw-Terminal, BNC</td>
</tr>
<tr>
<td>NI 9223</td>
<td>±10 V</td>
<td>4 Differential</td>
<td>1 MS/s/ch</td>
<td>Yes</td>
<td>16-Bit</td>
<td>Screw-Terminal, BNC</td>
</tr>
</tbody>
</table>

## NI C Series Overview

NI provides more than 100 C Series modules for measurement, control, and communication applications. C Series modules can connect to any sensor or bus and allow for high-accuracy measurements that meet the demands of advanced data acquisition and control applications.

- Measurement-specific signal conditioning that connects to an array of sensors and signals
- Isolation options such as bank-to-bank, channel-to-channel, and channel-to-earth ground
- –40 °C to 70 °C temperature range to meet a variety of application and environmental needs
- Hot-swappable
The majority of C Series modules are supported in both CompactRIO and CompactDAQ platforms and you can move modules from one platform to the other with no modification.

**CompactRIO**

CompactRIO combines an open-embedded architecture with small size, extreme ruggedness, and C Series modules in a platform powered by the NI LabVIEW reconfigurable I/O (RIO) architecture. Each system contains an FPGA for custom timing, triggering, and processing with a wide array of available modular I/O to meet any embedded application requirement.

**CompactDAQ**

CompactDAQ is a portable, rugged data acquisition platform that integrates connectivity, data acquisition, and signal conditioning into modular I/O for directly interfacing to any sensor or signal. Using CompactDAQ with LabVIEW, you can easily customize how you acquire, analyze, visualize, and manage your measurement data.

**Software**

**LabVIEW Professional Development System for Windows**

- Use advanced software tools for large project development
- Generate code automatically using DAQ Assistant and Instrument I/O Assistant
- Use advanced measurement analysis and digital signal processing
- Take advantage of open connectivity with DLLs, ActiveX, and .NET objects
- Build DLLs, executables, and MSI installers
NI LabVIEW FPGA Module

- Design FPGA applications for NI RIO hardware
- Program with the same graphical environment used for desktop and real-time applications
- Execute control algorithms with loop rates up to 300 MHz
- Implement custom timing and triggering logic, digital protocols, and DSP algorithms
- Incorporate existing HDL code and third-party IP including Xilinx IP generator functions
- Purchase as part of the LabVIEW Embedded Control and Monitoring Suite

NI LabVIEW Real-Time Module

- Design deterministic real-time applications with LabVIEW graphical programming
- Download to dedicated NI or third-party hardware for reliable execution and a wide selection of I/O
- Take advantage of built-in PID control, signal processing, and analysis functions
- Automatically take advantage of multicore CPUs or set processor affinity manually
- Take advantage of real-time OS, development and debugging support, and board support
- Purchase individually or as part of a LabVIEW suite
NI 9215 Input Circuitry

- Input signals on each channel are buffered, conditioned, and then sampled by an ADC.
- Each AI channel provides an independent track-and-hold amplifier, enabling you to sample all channels simultaneously.

NI 9215 Specifications

The following specifications are typical for the range -40 °C to 70 °C unless otherwise noted.

⚠️ Caution Do not operate the NI 9215 in a manner not specified in this document. Product misuse can result in a hazard. You can compromise the safety protection built into the product if the product is damaged in any way. If the product is damaged, return it to NI for repair.

Input Characteristics

<table>
<thead>
<tr>
<th>Characteristics</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels</td>
<td>4 analog input channels</td>
</tr>
<tr>
<td>ADC resolution</td>
<td>16 bits</td>
</tr>
<tr>
<td>Type of ADC</td>
<td>Successive approximation register (SAR)</td>
</tr>
<tr>
<td>Input range</td>
<td>±10.0 V</td>
</tr>
</tbody>
</table>
Input Voltage Ranges

<table>
<thead>
<tr>
<th>Measurement Voltage, AI+ to AI-</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum (V)</td>
<td>±10.2</td>
</tr>
<tr>
<td>Typical (V)</td>
<td>±10.4</td>
</tr>
<tr>
<td>Maximum (V)</td>
<td>±10.6</td>
</tr>
</tbody>
</table>

Maxum Voltage (Signal + Common Mode)

- NI 9215 with screw terminal: Each channel must remain within ±10.2 V of common.
- NI 9215 with spring terminal: Each channel must remain within ±10.2 V of common.
- NI 9215 with BNC: All inputs must remain within 10.2 V of the average AI inputs.

Overvoltage protection: ±30 V

Conversion time:
- Channel 0 only: 4.4 μs
- Channels 0 and 1: 6 μs
- Channels 0, 1, and 2: 8 μs
- Channels 0, 1, 2, and 3: 10 μs

<table>
<thead>
<tr>
<th>Table 1. Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Measurement Conditions</strong></td>
</tr>
<tr>
<td>Calibrated</td>
</tr>
<tr>
<td>Typical (23 °C ±5 °C)</td>
</tr>
<tr>
<td>Uncalibrated</td>
</tr>
<tr>
<td>Typical (23 °C ±5 °C)</td>
</tr>
</tbody>
</table>

Stability

- Gain drift: 10 ppm/°C
- Offset drift: 60 μV/°C

1 The minimum measurement voltage range is the largest voltage the NI 9215 is guaranteed to accurately measure.
2 Range equals ±10.4 V.
3 Uncalibrated accuracy refers to the accuracy achieved when acquiring in raw or unscaled modes where the calibration constants stored in the module are not applied to the data.
### Specifications

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMRR ($f_{in} = 60$ Hz)</td>
<td>73 dB min</td>
</tr>
<tr>
<td>Input bandwidth (-3 dB)</td>
<td>420 kHz minimum</td>
</tr>
<tr>
<td>Input impedance</td>
<td></td>
</tr>
<tr>
<td>Resistance</td>
<td></td>
</tr>
<tr>
<td>NI 9215 with screw terminal (AI-to-COM)</td>
<td>1 GΩ</td>
</tr>
<tr>
<td>NI 9215 with spring terminal (AI-to-COM)</td>
<td>1 GΩ</td>
</tr>
<tr>
<td>NI 9215 with BNC (Between any two AI terminals)</td>
<td>200 kΩ</td>
</tr>
<tr>
<td>Input bias current</td>
<td>10 nA</td>
</tr>
<tr>
<td>Input noise</td>
<td></td>
</tr>
<tr>
<td>RMS</td>
<td>1.2 LSB$_{rms}$</td>
</tr>
<tr>
<td>Peak-to-peak</td>
<td>7 LSB</td>
</tr>
<tr>
<td>Crosstalk</td>
<td>-80 dB</td>
</tr>
<tr>
<td>Setting time (to 2 LSBs)</td>
<td></td>
</tr>
<tr>
<td>NI 9215 with screw terminal</td>
<td></td>
</tr>
<tr>
<td>10 V step</td>
<td>10 µs</td>
</tr>
<tr>
<td>20 V step</td>
<td>15 µs</td>
</tr>
<tr>
<td>NI 9215 with spring terminal</td>
<td></td>
</tr>
<tr>
<td>10 V step</td>
<td>10 µs</td>
</tr>
<tr>
<td>20 V step</td>
<td>15 µs</td>
</tr>
<tr>
<td>NI 9215 with BNC</td>
<td></td>
</tr>
<tr>
<td>10 V step</td>
<td>25 µs</td>
</tr>
<tr>
<td>20 V step</td>
<td>35 µs</td>
</tr>
<tr>
<td>No missing codes</td>
<td>15 bits guaranteed</td>
</tr>
<tr>
<td>DNL</td>
<td>-1.9 to 2 LSB</td>
</tr>
<tr>
<td>INL</td>
<td>±6 LSB maximum</td>
</tr>
<tr>
<td>MTBF</td>
<td>1,167,174 hours at 25 °C; Bellcore Issue 6, Method 1, Case 3, Limited Part Stress Method</td>
</tr>
</tbody>
</table>
Power Requirements

<table>
<thead>
<tr>
<th>Mode</th>
<th>Active mode</th>
<th>Sleep mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power consumption from chassis (full-scale input, 100 kS/s)</td>
<td>560 mW maximum</td>
<td>25 μW maximum</td>
</tr>
</tbody>
</table>

Thermal dissipation (at 70 °C)

<table>
<thead>
<tr>
<th>Mode</th>
<th>Active mode</th>
<th>Sleep mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power 560 mW maximum</td>
<td>25 μW maximum</td>
<td></td>
</tr>
</tbody>
</table>

Physical Characteristics

If you need to clean the module, wipe it with a dry towel.

💡 Tip: For two-dimensional drawings and three-dimensional models of the C Series module and connectors, visit ni.com/dimensions and search by module number.

Screw-terminal wiring

<table>
<thead>
<tr>
<th>Description</th>
<th>Diameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gauge</td>
<td>0.2 mm² to 2.5 mm² (26 AWG to 14 AWG) copper conductor wire</td>
</tr>
<tr>
<td>Wire strip length</td>
<td>13 mm (0.51 in.) of insulation stripped from the end</td>
</tr>
<tr>
<td>Temperature rating</td>
<td>90 °C minimum</td>
</tr>
<tr>
<td>Torque for screw terminals</td>
<td>0.5 N·m to 0.6 N·m (4.4 lb·in. to 5.3 lb·in.)</td>
</tr>
<tr>
<td>Wires per screw terminal</td>
<td>One wire per screw terminal; two wires per screw terminal using a 2-wire ferrule</td>
</tr>
<tr>
<td>Ferrules</td>
<td>0.25 mm² to 2.5 mm²</td>
</tr>
</tbody>
</table>

Spring-terminal wiring

<table>
<thead>
<tr>
<th>Description</th>
<th>Diameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gauge</td>
<td>0.2 mm² to 2.5 mm² (30 AWG to 12 AWG) copper conductor wire</td>
</tr>
<tr>
<td>Wire strip length</td>
<td>10 mm (0.39 in.) of insulation stripped from the end</td>
</tr>
<tr>
<td>Temperature rating</td>
<td>90 °C minimum</td>
</tr>
<tr>
<td>Wires per spring terminal</td>
<td>One wire per spring terminal; two wires per spring terminal using a 2-wire ferrule</td>
</tr>
<tr>
<td>Ferrules</td>
<td>0.25 mm² to 2.5 mm²</td>
</tr>
</tbody>
</table>
Connector securement

<table>
<thead>
<tr>
<th>Securement type</th>
<th>Screw flanges provided</th>
</tr>
</thead>
<tbody>
<tr>
<td>Torque for screw flanges</td>
<td>0.2 N·m (180 lb·in)</td>
</tr>
</tbody>
</table>

Weight

<table>
<thead>
<tr>
<th>Type</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>NI 9215 with screw terminal</td>
<td>150 g (5.3 oz)</td>
</tr>
<tr>
<td>NI 9215 with spring terminal</td>
<td>138 g (4.9 oz)</td>
</tr>
<tr>
<td>NI 9215 with BNC</td>
<td>173 g (6.1 oz)</td>
</tr>
</tbody>
</table>

**NI 9215 with Screw Terminal and NI 9215 with Spring Terminal Safety Voltages**

Connect only voltages that are within the following limits.

<table>
<thead>
<tr>
<th>Channel-to-COM</th>
<th>±30 V maximum</th>
</tr>
</thead>
</table>

Isolation

<table>
<thead>
<tr>
<th>Channel-to-channel</th>
<th>None</th>
</tr>
</thead>
<tbody>
<tr>
<td>Channel-to-earth ground</td>
<td></td>
</tr>
<tr>
<td>Continuous</td>
<td>250 Vrms, Measurement Category II</td>
</tr>
<tr>
<td>Withstand</td>
<td>2,300 Vrms, verified by a 5 s dielectric withstand test</td>
</tr>
</tbody>
</table>

Measurement Category II is for measurements performed on circuits directly connected to the electrical distribution system. This category refers to local-level electrical distribution, such as that provided by a standard wall outlet, for example, 115 V for U.S. or 230 V for Europe.

⚠️ **Caution** Do not connect the NI 9215 to signals or use for measurements within Measurement Categories III or IV.

**NI 9215 with BNC Safety Voltages**

Connect only voltages that are within the following limits.

<table>
<thead>
<tr>
<th>AI+ to AI-</th>
<th>±30 V maximum</th>
</tr>
</thead>
</table>

Isolation

<table>
<thead>
<tr>
<th>Channel-to-channel</th>
<th>None</th>
</tr>
</thead>
<tbody>
<tr>
<td>Channel-to-earth ground</td>
<td></td>
</tr>
<tr>
<td>Continuous</td>
<td>60 VDC, Measurement Category I</td>
</tr>
<tr>
<td>Withstand</td>
<td>1,500 Vrms, verified by a 5 s dielectric withstand test</td>
</tr>
</tbody>
</table>

Measurement Category I is for measurements performed on circuits not directly connected to the electrical distribution system referred to as *MAINS* voltage. MAINS is a hazardous live...
electrical supply system that powers equipment. This category is for measurements of voltages from specially protected secondary circuits. Such voltage measurements include signal levels, special equipment, limited-energy parts of equipment, circuits powered by regulated low-voltage sources, and electronics.

\textbf{Note} Measurement Categories CAT I and CAT 0 are equivalent. These test and measurement circuits are not intended for direct connection to the MAINS building installations of Measurement Categories CAT II, CAT III, or CAT IV.

\textbf{Caution} Do not connect the NI 9215 to signals or use for measurements within Measurement Categories II, III, or IV.

\section*{Hazardous Locations}

\begin{tabular}{|l|l|}
\hline
U.S. (UL) & Class I, Division 2, Groups A, B, C, D, T4; Class I, Zone 2, AEx nA IIC T4 \\
\hline
Canada (C-UL) & Class I, Division 2, Groups A, B, C, D, T4; Class I, Zone 2, Ex nA IIC T4 \\
\hline
Europe (ATEX) and International (IECEx) & Ex nA IIC T4 Gc \\
\hline
\end{tabular}

\section*{Safety and Hazardous Locations Standards}

This product is designed to meet the requirements of the following electrical equipment safety standards for measurement, control, and laboratory use:

- IEC 61010-1, EN 61010-1
- UL 61010-1, CSA 61010-1
- EN 60079-0:2012, EN 60079-15:2010
- IEC 60079-0: Ed 6, IEC 60079-15; Ed 4
- UL 60079-0; Ed 5, UL 60079-15; Ed 3

\textbf{Note} For UL and other safety certifications, refer to the product label or the Online Product Certification section.

\section*{Electromagnetic Compatibility}

This product meets the requirements of the following EMC standards for electrical equipment for measurement, control, and laboratory use:

- EN 61326-1 (IEC 61326-1): Class A emissions, Industrial immunity
- EN 55011 (CISPR 11): Group 1, Class A emissions
- AS/NZS CISPR 11: Group 1, Class A emissions
• FCC 47 CFR Part 15B: Class A emissions
• ICES-001: Class A emissions

Note For EMC declarations and certifications, and additional information, refer to the Online Product Certification section.

CE Compliance
This product meets the essential requirements of applicable European Directives, as follows:
• 2014/35/EU; Low-Voltage Directive (safety)
• 2014/30/EU; Electromagnetic Compatibility Directive (EMC)
• 94/9/EC; Potentially Explosive Atmospheres (ATEX)

Online Product Certification
Refer to the product Declaration of Conformity (DoC) for additional regulatory compliance information. To obtain product certifications and the DoC for this product, visit ni.com/certification, search by model number or product line, and click the appropriate link in the Certification column.

Shock and Vibration
To meet these specifications, you must panel mount the system.

Operating vibration
- Random (IEC 60068-2-64) 5 g_{rms}, 10 Hz to 500 Hz
- Sinusoidal (IEC 60068-2-6) 5 g, 10 Hz to 500 Hz
- Operating shock (IEC 60068-2-27) 30 g, 11 ms half sine; 50 g, 3 ms half sine; 18 shocks at 6 orientations

Environmental
Refer to the manual for the chassis you are using for more information about meeting these specifications.

- Operating temperature (IEC 60068-2-1, IEC 60068-2-2) -40 °C to 70 °C
- Storage temperature (IEC 60068-2-1, IEC 60068-2-2) -40 °C to 85 °C
- Ingress protection IP40
- Operating humidity (IEC 60068-2-78) 10% RH to 90% RH, noncondensing
- Storage humidity (IEC 60068-2-78) 5% RH to 95% RH, noncondensing
- Pollution Degree 2
- Maximum altitude 2,000 m
Indoor use only.

**Environmental Management**
NI is committed to designing and manufacturing products in an environmentally responsible manner. NI recognizes that eliminating certain hazardous substances from our products is beneficial to the environment and to NI customers.

For additional environmental information, refer to the *Minimize Our Environmental Impact* web page at [ni.com/environment](http://ni.com/environment). This page contains the environmental regulations and directives with which NI complies, as well as other environmental information not included in this document.

**Waste Electrical and Electronic Equipment (WEEE)**

EU Customers  At the end of the product life cycle, all NI products must be disposed of according to local laws and regulations. For more information about how to recycle NI products in your region, visit [ni.com/environment/weee](http://ni.com/environment/weee).

**电子信息产品污染控制管理办法（中国 RoHS）**

中国客户  National Instruments 符合中国电子信息产品中限制使用某些有害物质指令(RoHS)。关于 National Instruments 中国 RoHS 合规性信息，请登录 [ni.com/environment/rohs_china](http://ni.com/environment/rohs_china)。 (For information about China RoHS compliance, go to [ni.com/environment/rohs_china](http://ni.com/environment/rohs_china).

**Calibration**

You can obtain the calibration certificate and information about calibration services for the NI 9215 at [ni.com/calibration](http://ni.com/calibration).

<table>
<thead>
<tr>
<th>Calibration interval</th>
<th>1 year</th>
</tr>
</thead>
</table>

Refer to the **NI Trademarks and Logo Guidelines** at ni.com/trademarks for information on NI trademarks. Other product and company names mentioned herein are trademarks or trade names of their respective companies. For patents covering NI products/technology, refer to the appropriate location: Help > Patents in your software, the patents.txt file on your media, or the National Instruments Patent Notice at ni.com/patents. You can find information about end-user license agreements (EULAs) and third-party legal notices in the readme file for your NI product. Refer to the Export Compliance Information at ni.com/legal/export-compliance for the NI global trade compliance policy and how to obtain relevant HTS codes, ECCNs, and other import/export data. NI MAKES NO EXPRESS OR IMPLIED WARRANTIES AS TO THE ACCURACY OF THE INFORMATION CONTAINED HEREIN AND SHALL NOT BE LIABLE FOR ANY ERRORS. U.S. Government Customers: The data contained in this manual was developed at private expense and is subject to the applicable limited rights and restricted data rights as set forth in FAR 52.227-14, DFAR 252.227-7014, and DFAR 252.227-7015.

© 2016 National Instruments. All rights reserved.