

The Challenges of Automated Spectral Analysis

Reflections on eighteen years of development by Jason Tranter, Managing Director, Mobius Institute

For the past eighteen years I have been writing vibration analysis software - first my own Australian company ARGO, then for DLI Engineering (now AzimaDLI) in the United States, and now I am at it again for my own company, Mobius Institute. The goal has always been to create software that can analyze spectra and tell the user what may be wrong with the machine. It has been an interesting journey, and this paper will share some of my experiences.

The early days...

In the early days, starting in around 1985 the goal was primarily to store the spectra and provide graphical tools so that the analyst could quickly access the data and determine for him or herself what might be wrong with the machine. But as the data collection devices (originally spectrum analyzers) got smaller, faster, and more portable, the amount of data to analyze grew. Pretty soon the data collector could store hundreds of spectra - lots of data.

Portable data collector in action



Portable data collector in action

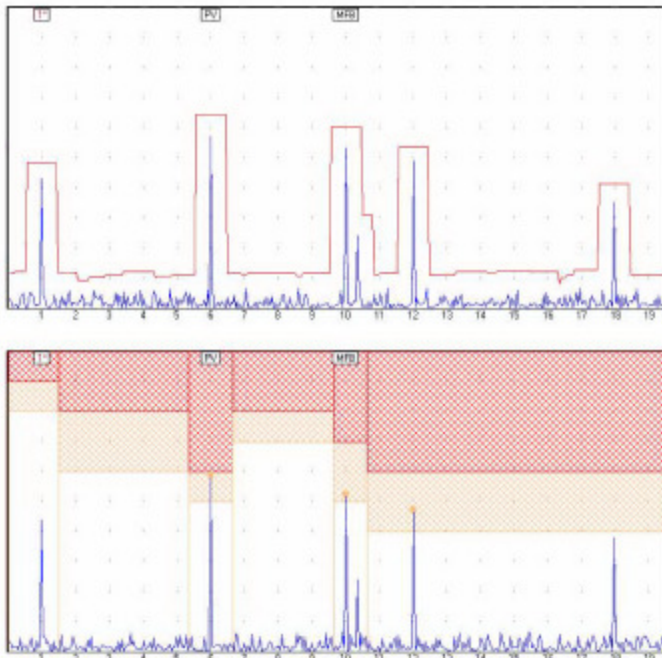
Some people chose to collect overall data and only collect spectra when “really” necessary. Some people chose to limit the amount of data they collected - to save collection time, but also to save analysis time. But pressure on the analyst grew, and the data collectors and measurement techniques improved, and as a result the analyst was faced with a mountain of data.

Seeing this, I originally decided to create the ultimate “screening” tool - software that would scan all of the new measurements and allow the user to focus their attention on the data that needed it the most. I started with techniques that may be considered a little old fashioned - creating mask and band alarms, then comparing the spectra to those limits. Rather than just reporting on which passed and failed, my goal was to try, in as few words as possible, describe to the user why and how the spectrum failed. I tried to relate what I could see in the data to what I knew about the machine. It was my first attempt at being able to warn users that the machine may have an imbalance problem, for example, before they had personally looked at the data.

One of the biggest challenges was the task of setting alarm limits. How do you know how

much vibration is OK? And who has time to set up all of the alarms anyway? With so many machines being tested, multiplied by the number of test points and axes, there were a very large number of alarm limits to set up - particularly if you were using band alarms.

So I tried to automate the process.



I allowed the user to specify upper and lower limits, and “deltas” of change allowed (for example, they could collect the “reference” or “baseline” spectrum and then tell the software that the levels could only change by a certain amount from that baseline).

It was not all that successful. At the time I had my supporters, and it did save people a lot of time, but it was not as accurate as I would have liked.

Off to the US of A

In late 1990 I sold the rights to my software to DLI Engineering in Seattle and joined them on Bainbridge Island in the Puget Sound. DLI had been working on an expert system for a couple of years, with funding from the US Navy. I thought this sounded pretty cool. So I joined them in that development - I was in charge of product development at DLI.

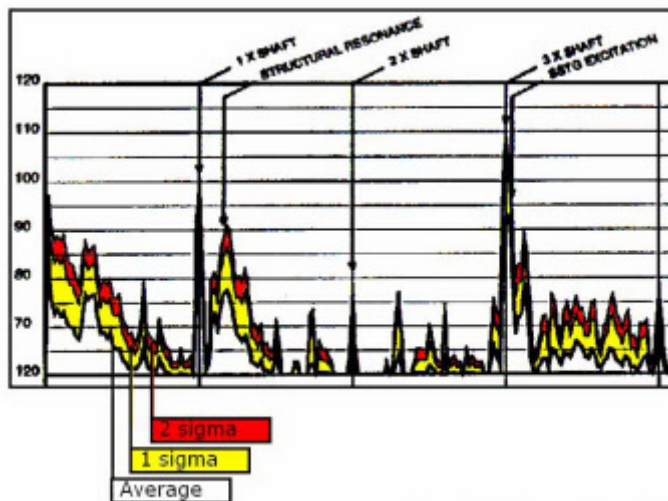
It was quite an interesting approach that they had taken. They began by carefully analyzing how “human analysts” analyzed spectra. They looked at how the analyst would focus their attention on specific features - depending upon what they knew about the machine. And they looked at how the analyst assessed whether a feature in the data existed - had it changed, was it relevant.

They also looked at the best way to set alarms, or thresholds. They observed that the analysts would look at previous measurements in order to try and determine what the vibration levels

were “normally” like. What amount of variation was “normal”. But how do you do that for a large number of spectra?

So we decided to employ “statistics”. After first “normalizing” the spectra (to reduce the impact of speed variation), previous measurements were averaged together, keeping track of the amount of variation witnessed - at every position (tested), in every axis (they always collected data in three axes) on every machine.

When new data was collected, it could be normalized and compared to this statistical dataset to determine just how much it had varied, and where it varied. By tolerating a certain amount of variation (related to how much the data “normally” varied), they could greatly reduce the false alarms. So, once we had a set of data that did “abnormally” vary in amplitude, we could then compare the pattern of variation to the “expected” patterns for machines of that type. It meant that we had to build a set of “rules” for lots of different types of machines.



Spectrum with average, 1 sigma and 2 sigma (standard deviation) overlaid

It also meant that we had to give the user a convenient way of describing the machine to the software - and this is where the fun began. We needed to know the key details required in order to perform the analysis - what “type” was it (motor, pump, etc.), what was the running speed, was there a gearbox, belt drive, etc., and where possible, we endeavored to get the number of rotating elements, the number of gears, etc. All of the information that the human analyst needed.

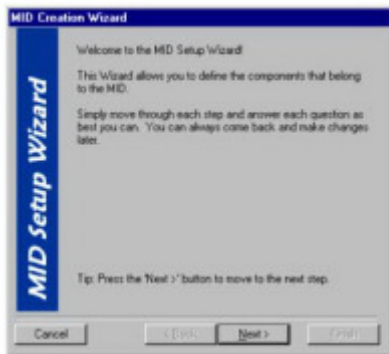
They also looked at the desired format for the results. Most “expert systems” give results in terms of probabilities - there is an 82% chance of imbalance, 92% chance of misalignment, and so on. It was not considered that this type of information was actually very useful. What people wanted was a clearer statement of what was wrong with the machine, and a statement of severity: “This machine has severe bearing wear”.

Sample report format

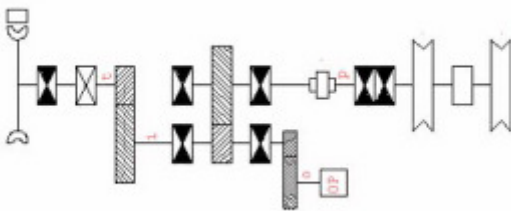
Anyway, to cut a real long story short, it worked quite well. Lots of people around the world use the software to analyze their data. Some use it to simply screen their data before performing their own manual analysis. Some took heed of the reported recommendations. It all depends on the experience level of the person running the program, and the time spent setting it up in the first place.

And here is the challenge. Once upon a time, in the early eighties, the people performing the analysis were engineers. They were given time to perform the analysis, and they were able to study the machine and data closely. But as time marched on, the engineers were assigned to other tasks, and less well trained people took over the role. And they were given more, and more, and more tasks to perform. So after ten years, the typical customer was someone who often did not have the time to nurture and care for every spectrum. They had ten times the amount of data, and ten times the number of tasks to perform.

So, this was one of the issues that drove me first towards trying to create automated systems - because while they were not perfect, nor was the user of the system - and later towards creating computer based training systems - with the right kind of education and support structure, the chance of success would be far greater.



The "MID" creation wizard



"VTAG" schematic diagram

After six years at DLI, it was time to move on (at least, that is what the US Immigration Department told me). After spending a couple more years of software development and consulting back here in Australia, it was time to move on to the second phase - training.

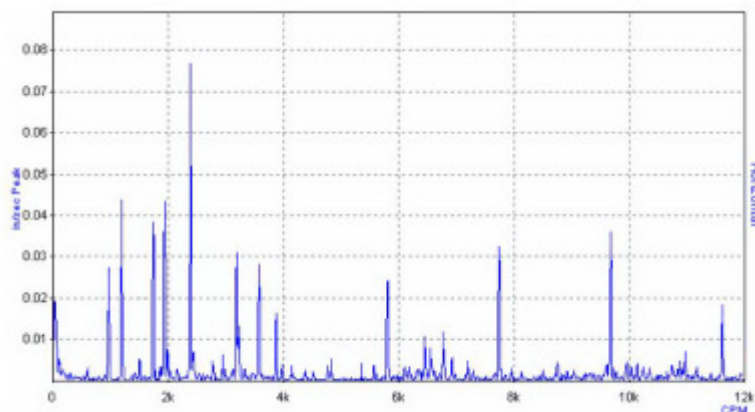
The answer is training

Over the previous years I had naturally been involved in training the users of the systems I had developed, and giving technical papers. I tried my best to pass on both the background knowledge that every analyst needs, and the steps required to operate the systems. But I could see some users floundering - first in the classroom (and not all users would even opt to take the training), and then back in their plant. If the customer did not really understand what was going on, it was difficult for them to be successful. So I decided to try and change all that!

A typical classroom (sleep well!)

I created the iLearnInteractive series of training products. Now, exactly what I did, and why, and how many we have sold, is probably information that is not appropriate to discuss in a technical paper - so I won't. However, something very interesting happened.

In the early days the training system was just that - training only. Sure, it was very practical, filled with "live" cases studies, and lots of tables and great information, but the intent was to prepare them for the real world of analysis, and then set them free. But then I received e-mails from customers thanking me for helping them solve machines problems. You see, instead of just taking the training and moving on, they were accessing the material almost every day. And when they saw something unusual in their data, they would look through the training system to find the section that described what it was and how to deal with it. Wow - it had become a reference system!

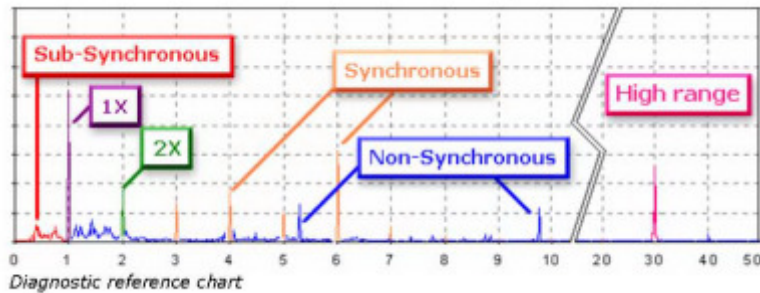


A typical, challenging spectrum

So then came version 2.0 of the system. It had a real reference system, where the users of the software could easily search the material and basically diagnose machine faults. All they had to do was find the pattern in the data and describe it to the Reference Center - it would then tell them what the probable causes were.

Diagnostic reference chart

This was a great help - but it got me thinking. The e-mails I was getting now told me that people often found it hard to find the patterns in the first place: “I know there are harmonics and sidebands in there, but it is too hard to isolate them”. Once they found the pattern they could then manually use the Reference Center. There had to be a better way.



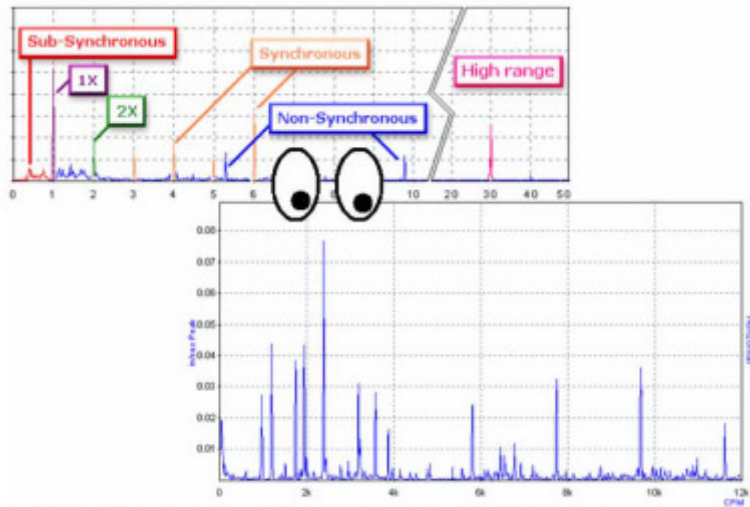
The other issue was that even the automated diagnostic systems, like the one I was involved with at DLI, would give you its opinion on what was wrong with the machine, but if you had any questions, you basically had to go back to square one and manually analyze the data for yourself.



Nobody said it would be easy

So, I started to wonder if there was any way I could, in effect, get the Reference Center to automatically analyze the data for the user - to assist them in the pattern identification process, and then interpret the pattern for them. But then the challenges began.

Could the Reference Center look at the data on behalf of the user?



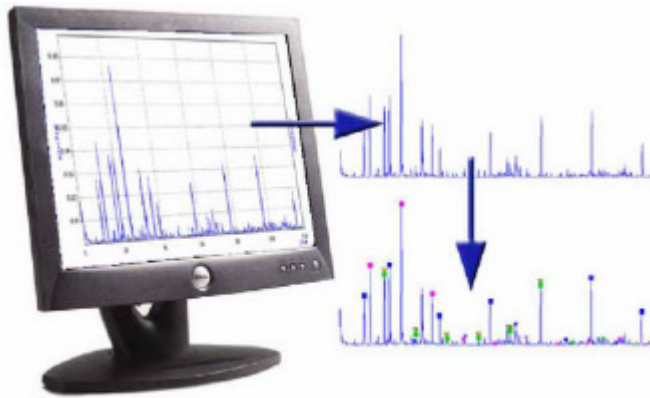
Could the Reference Center look at the data on behalf of the user?

Extract the data

The first challenge was that I did not want to write software that read the database of the condition monitoring systems directly. I have been beating my head against that wall for too many years - no one will tell you how their data is stored, and it is a huge programming task even if they did. But look, I can see the data right there on the screen - typically on high resolution monitors too.

So, I decided to write software that would actually read the spectrum off the screen - grab the pixels and build the spectrum in memory - all I need to know was where the spectrum was on the screen, and what colors were used for the border and graph.

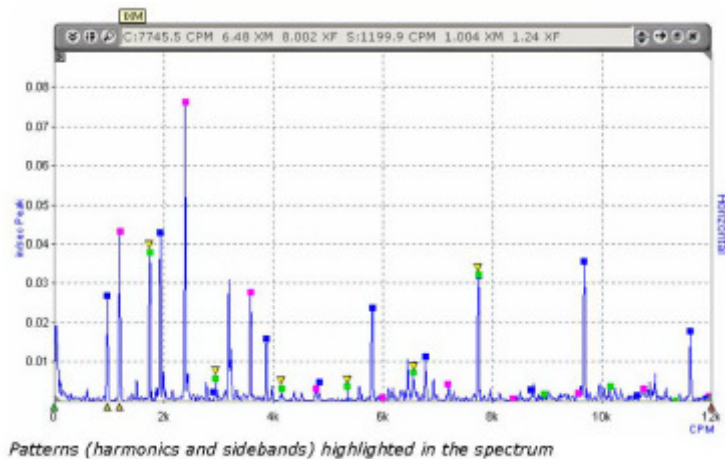
Find the pattern



Find the pattern

As if that was not challenging enough, the next step was to look at this spectrum and find the patterns. I first tried some of the techniques used at DLI - cepstrum analysis - but that did not give me the results I wanted. Instead, to cut another long story short, I developed an algorithm to find the patterns in the data without knowing very much about the data at all. It basically works like this:

1. First I start with a list of all the peaks found in the data.
2. Then I group them according to their size - although the small peaks can be very important, and can form all or part of the pattern, the larger peaks allow me to prioritize the pattern.
3. Next I look at the separation between all of the peaks. I basically iterate through all of the peaks looking for patterns. I make the assumption that any pair could be a part of a pattern. For example, if I was to start with the first and last peak in the spectrum, I would first check to see if there were 20 peaks evenly spaced between them, and then 19, and then 18. Each time I would “expect” to find peaks at certain frequencies - so I would check my peak list to see if they existed. If I find enough matching peaks I would determine that I had found a “pattern” and set it aside.
4. In a perfect world, all of the patterns would be perfectly clear. They would not mix with other patterns. And peaks from other sources would not coincidentally appear in inconvenient locations - perhaps to make something look like a pattern that actually was not a pattern at all. Because of this, and the fact that I was often dealing with limited resolution (either because the user only collected low resolution data, or because I had insufficient pixels to work with) I needed a way of determining whether a peak really belonged to a pattern or not. For example, if I looked for a peak at a given pixel position and it was there, then I assumed it was part of the pattern. But what if the peak was just one pixel off? Is that close enough? The location of a peak, either in the original raw data or in the data taken from the screen, could be “just off” for a large number of valid reasons. So, I allow the user to set the “sensitivity” of the algorithm, but otherwise I simply dealt with it as best I could.
5. Once I had my list of patterns, I had to check that I had not simply found the same pattern over more than once.
6. I also allow the user to set a few “preferences”. Did they want patterns to be reported if they only have three peaks? What if the patterns contained only low amplitude peaks. What if a pattern contained three good matches, but one imperfect peak match?



It was certainly a huge challenge to make it work. And I expect that I will be fine tuning the algorithm for some time.

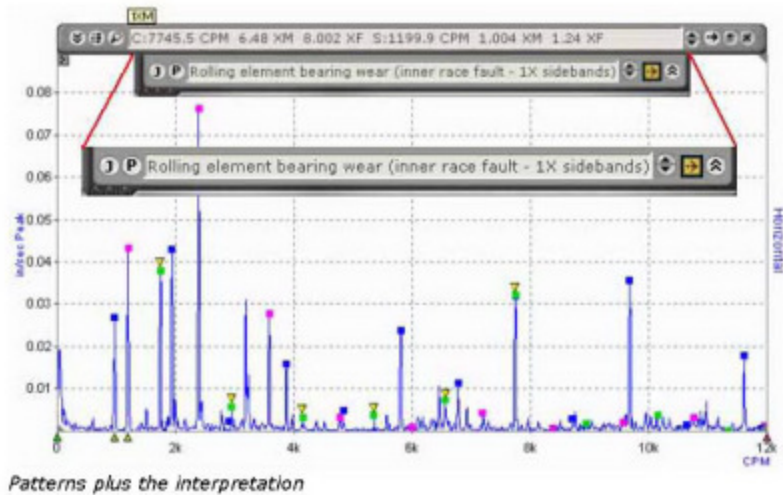
As a small aside, one of the challenges came when I came to look at the graphs from various software vendors. Some used borders all around the graph, others did not. Some started the graph at the left hand border, others start it just inside the border - so there were a couple of challenges to deal with.

Interpret the pattern

Now that the pattern was revealed, the next step was to interpret what the patterns tell us about the machine. But to do that properly we really need to know a little about the machine. We really need to know the running speed, and it would be helpful to know the Fmax of the graph.

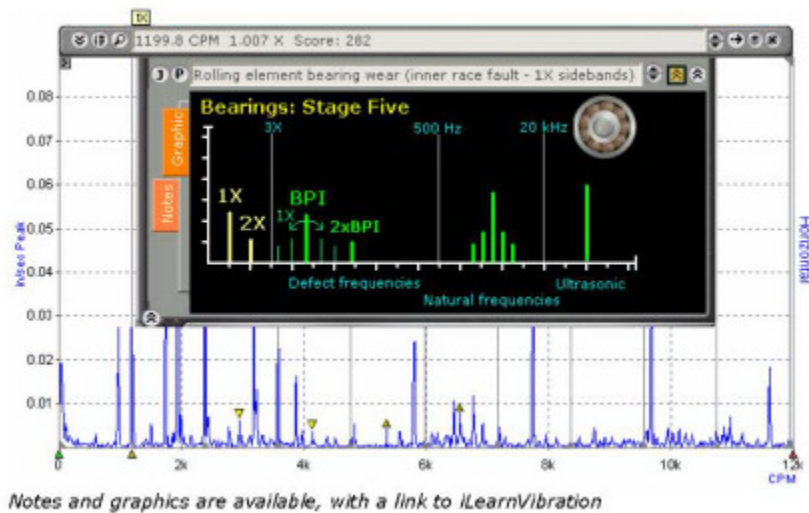
I could give the user a way to tell me the running speed, just by pointing to the peak on the graph, but the Fmax turned out to be a bit of a challenge because not all graphs have the Fmin and Fmax exactly at the start and end of the graph. So, I developed a way for them to handle this graphically - I actually look at the pixels for the tick mark.

So, now that I know the delta of the peaks, I could actually interpret the patterns. A strong series of 1X peaks might be looseness. A set of 1X sidebands might be a bearing fault.



Patterns plus the interpretation

It was important that I explain to users why these faults could be looseness or a bearing fault - but that is easy with the resources of the Reference Center and iLearnVibration.



The user could then see the patterns, and see my interpretation, and decide for him or herself what to do next. This might be enough to enable a complete diagnosis (or a decision to move to the next graph). Or it might require further analysis.

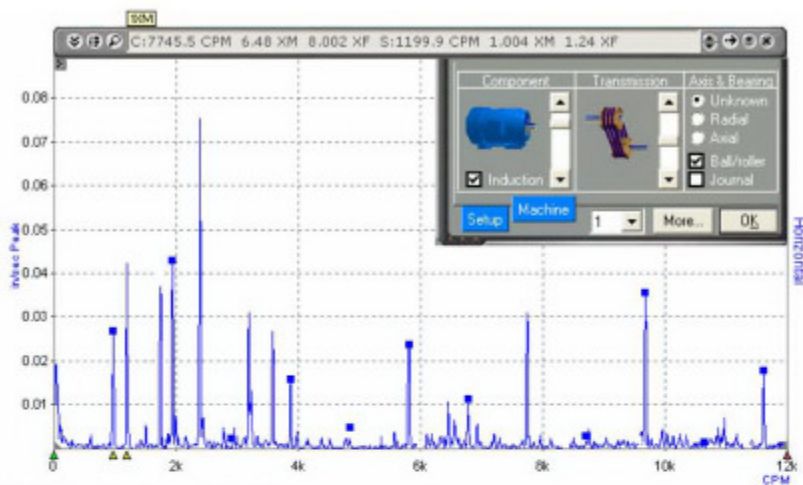
One important point is that if you don't know whether the data came from the axial, horizontal or vertical direction, there can be a larger number of possibilities for a given

pattern. And if you knew what the machine was, then you could also rule out some of the possible interpretations.

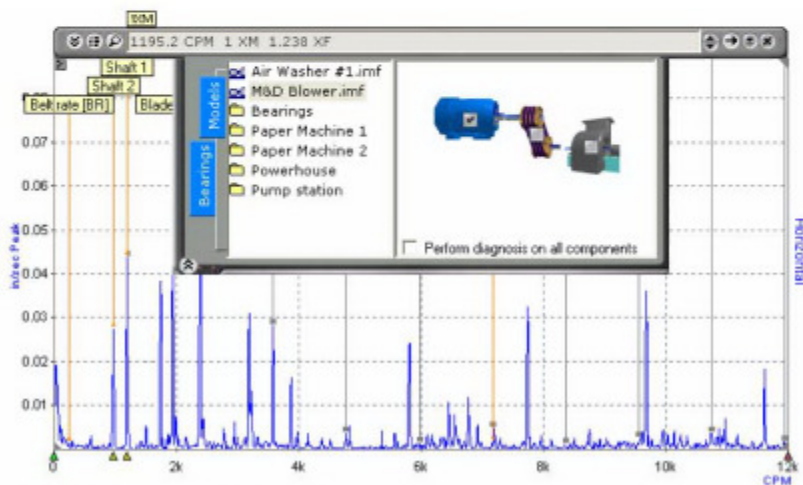
When the system was first released, it really only looked for the patterns and tried to give basic interpretations (and provided explanations). But I immediately received two strong messages.

First, it was important to comment on features in the data that might not be a part of a pattern - for example, imbalance and misalignment may only involve one or two peaks. So this meant looking at the peaks and seeing what significant peaks were not part of a pattern, and then determining their significance.

Second, the users wanted to be able to tell the software more about the machine so that I could rule out (or in) some of the interpretations. So, by allowing them to quickly select the component type(s) and measurement axis, and by allowing them to optionally build a model and use the computed forcing frequencies, the system was able to go from some very general comments to a shorter list of specific interpretations.



Quickly select from a list of common components



Selecting the user-created model

Final comment

We should reflect upon what my goals were when I began this project. I was not trying to create an “expert system”, better than the one we created at DLI, that would solve every machine vibration problem. Instead my goal was to create a tool that would give the analyst a “helping hand”. If the analyst was very experienced, and was able to find patterns quickly, and knew all of the diagnostic rules, then he or she would not need my software.

On the other hand, if a person needed help finding the patterns (those harmonics and sidebands can be hard to find sometimes), and/or struggled to remember all of the diagnostic rules, (or just wanted to save some time and have a cross reference), then iLearnInterpreter would help.

It has been a huge challenge, and the challenge is not completely over yet. But then, what would life be without a few challenges?

Visit the web site below to find out where this technology is going in the future

www.ilearninteractive.com

Jason@ilearninteractive.com