

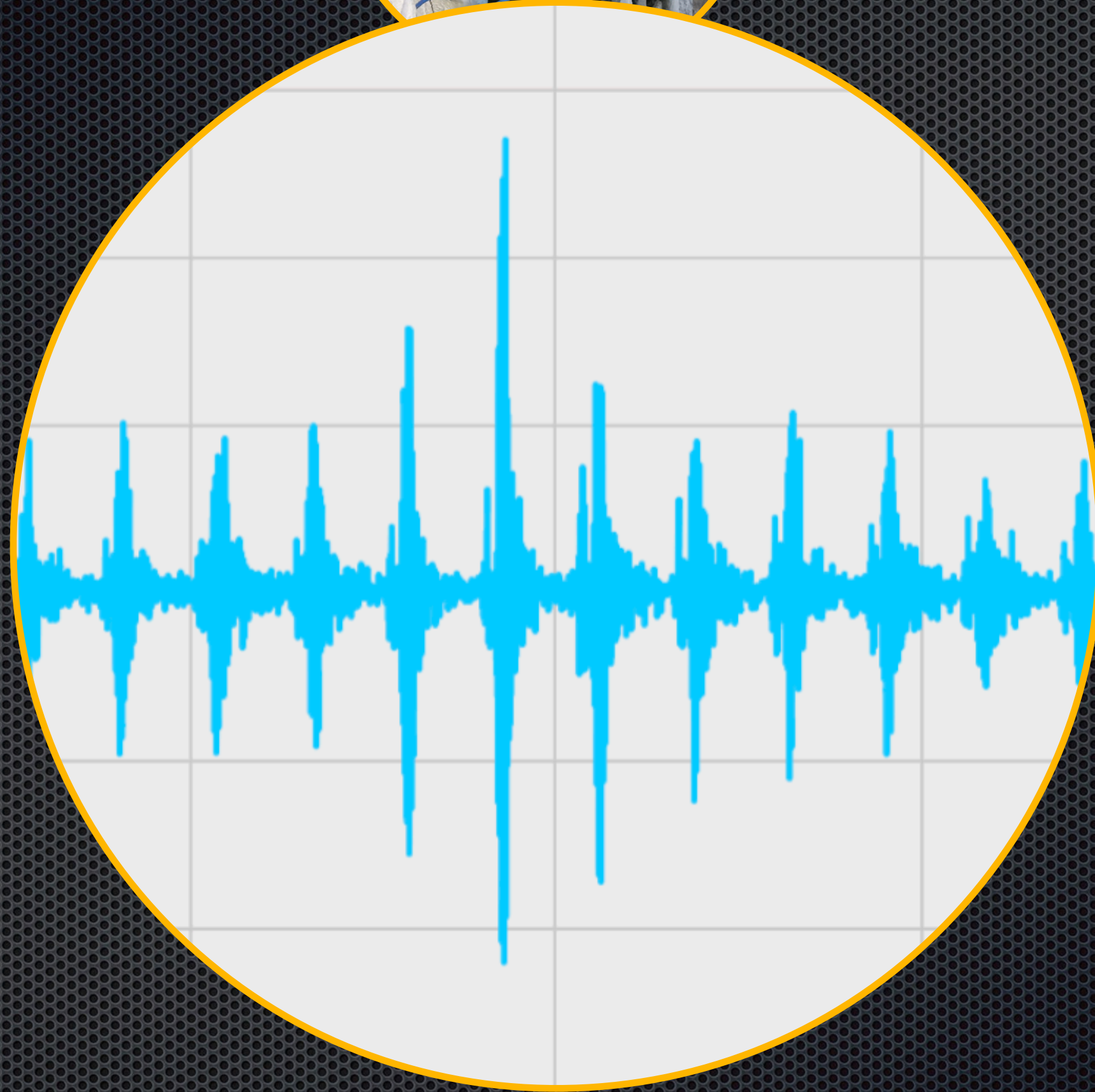
CBM Journey

From SIGNAL to INFORMATION

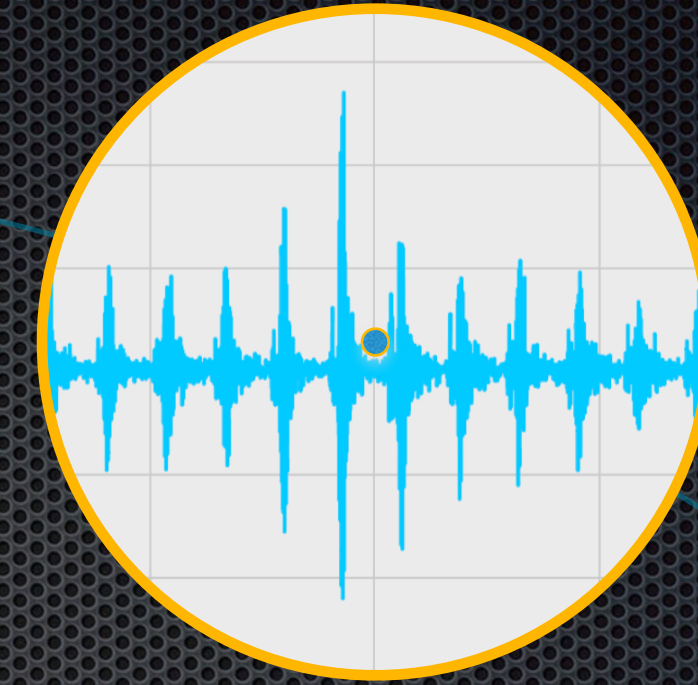
Explaining the Process of **Transforming** the **Machine Voice** to **Information** for the use of Maintenance and Production teams.



The CBM Journey starts from the **CM Point** on the machine

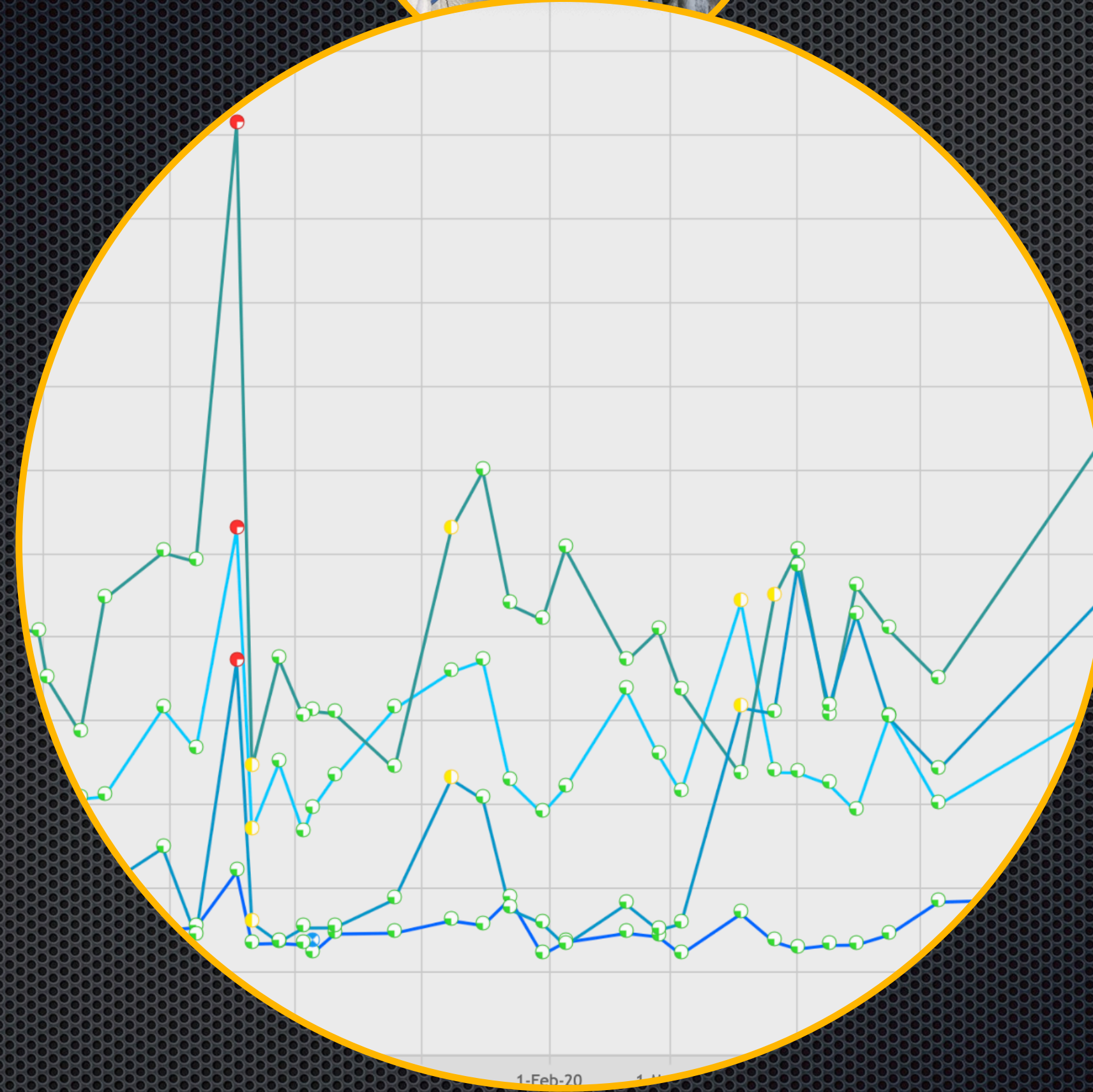
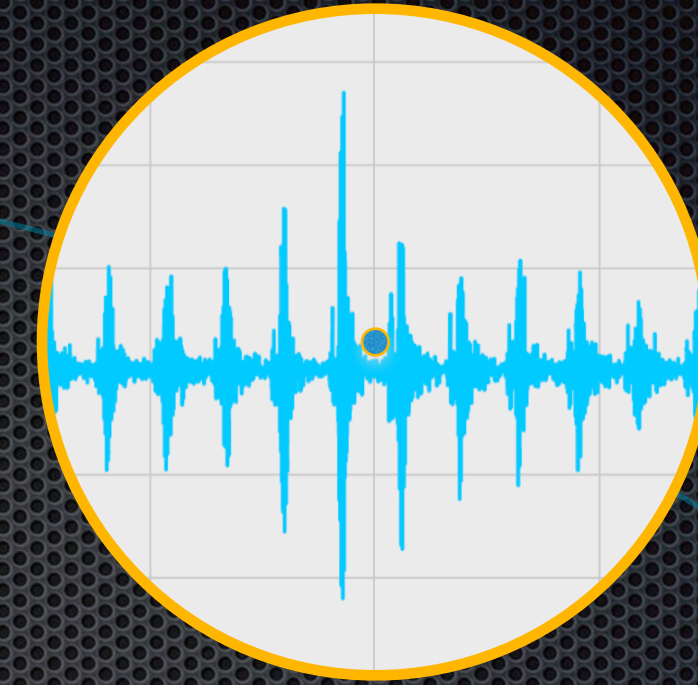


The **Accelerometer** sends time **Signal** to logger



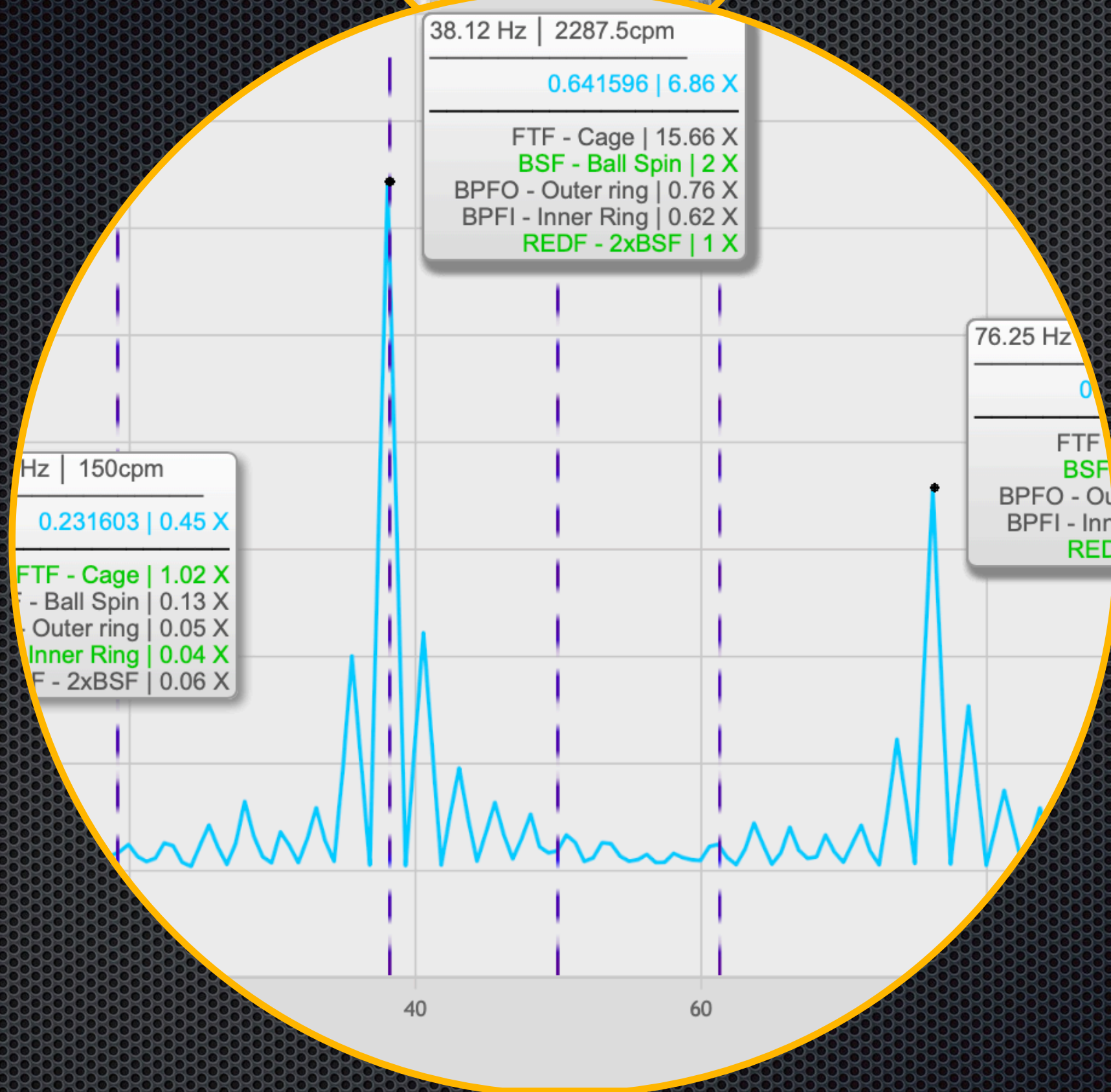
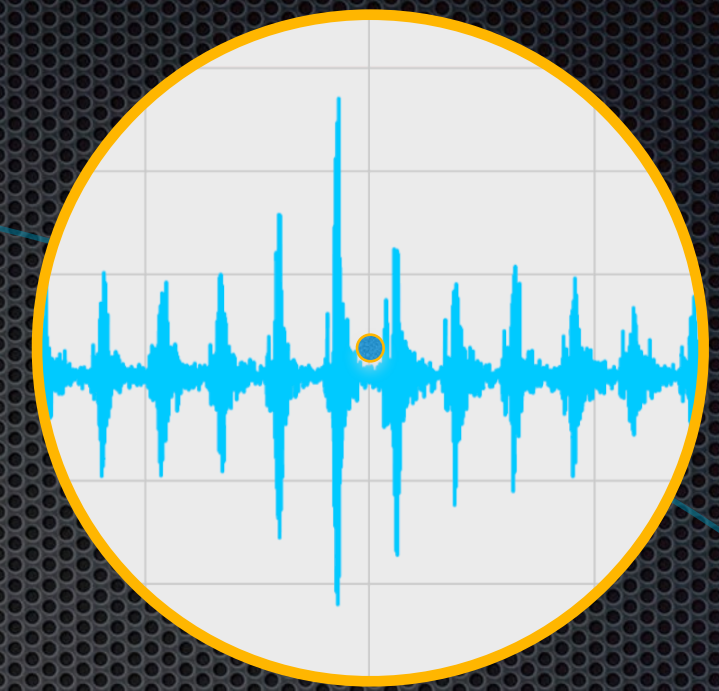
```
... offset from the start of the ...  
... -- fractional overlap (between 0 and 1)  
...  
... returns:  
... (maxfreq, nlines, offset, overlap) -- tuple of suggested  
... parameters for Fourier transforms.""""  
if maxfreq > self.twf.freq/2.0:  
    newmaxfreq = self.twf.freq/2.0  
    logging.warning('Requested maximum frequency {0} too high, setting  
                    'to {1} (half the sampling frequency)'.format(maxfreq,  
                                                                    newmaxfreq))  
    maxfreq = newmaxfreq  
length = int((nlines)*self.twf.freq/maxfreq)  
if length > self.twf.samples:  
    length = self.twf.samples  
    newnlines = int(length*maxfreq/self.twf.freq)  
    logging.warning('Requested number of lines {0} too large, setting  
                    'to {1} (maximum possible)'.format(nlines, newnlines))  
    nlines = newnlines  
if offset < 0 or offset+length > self.twf.samples:  
    logging.warning('Requested offset {0} too large, setting to {1} '  
                    '(maximum possible)'.format(offset, self.twf.samples))  
    offset = self.twf.samples - length  
if overlap is not None:  
    # FFT averaging functions handle gracefully nooverlap=0  
    if overlap < 0.0 or overlap >= 1.0:  
        logging.warning('Requested overlap {0} should be between 0 and  
                        '1, setting to 0 (no averaging)'.format(overlap))  
        overlap = 0.0  
    elif 2*length-int(overlap*length) > self.twf.samples:  
        nooverlap = 2*length - self.twf.samples  
        if nooverlap >= self.twf.samples:  
            newoverlap = 0.0  
            logging.warning('Requested overlap {0} too small, setting to  
                            '0 (no averaging)'.format(overlap))  
        else:  
            newoverlap = 1.0*nooverlap/length  
            assert(2*length-int(newoverlap*length) <= self.twf.samples)  
            logging.warning('Requested overlap {0} too small, setting to  
                            '{1}'.format(overlap, newoverlap))  
            overlap = newoverlap
```

The **CM platform** processes the **Signal**

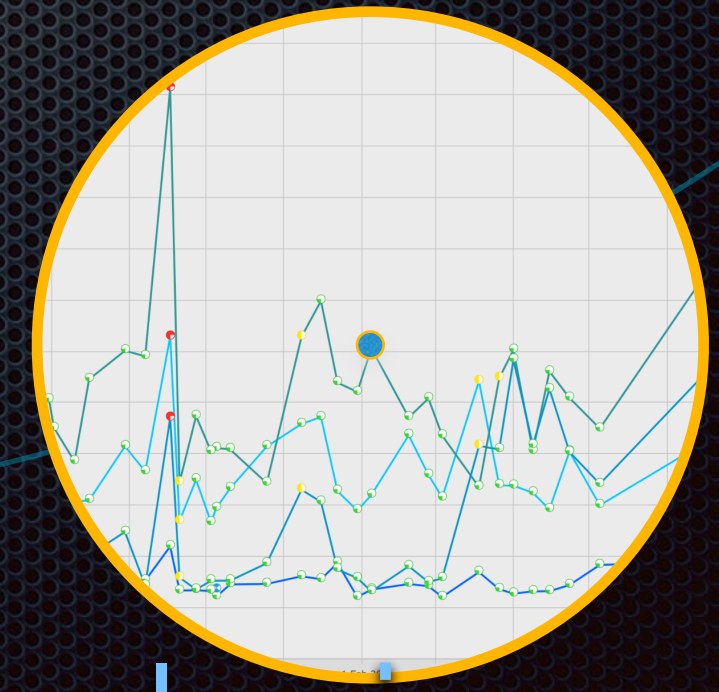


```
def __init__(self, maxfreq, nlines, offset, overlap):
    """
    Parameters for Fourier transforms.
    """
    if maxfreq > self.twf.freq/2.0:
        newmaxfreq = self.twf.freq/2.0
        logging.warning("Requested maximum frequency (0) too high,
            setting to (1) (half the sampling frequency).".format(
                maxfreq, newmaxfreq))
        maxfreq = newmaxfreq
    length = int((nlines)*self.twf.freq/maxfreq)
    if length > self.twf.samples:
        length = self.twf.samples
        newnlines = int(length*maxfreq/self.twf.freq)
        logging.warning("Requested number of lines (0) too large, setting
            to (1) (maximum possible).".format(nlines, newnlines))
        nlines = newnlines
    if offset < 0 or offset+length > self.twf.samples:
        logging.warning("Requested offset (0) too large, setting to (1)
            (maximum possible).".format(offset, self.twf.samples))
    offset = self.twf.samples - length
    # FFT averaging functions handle gracefully nlines=0
    if overlap < 0.0 or overlap >= 1.0:
        logging.warning("Requested overlap (0) should be between 0 and
            1, setting to 0 (no averaging).".format(
                overlap, 0.0))
        overlap = 0.0
    elif 2*length-int(overlap*length) > self.twf.samples:
        newoverlap = 2*length - self.twf.samples
        if newoverlap >= self.twf.samples:
            newoverlap = 0.0
            logging.warning("Requested overlap (0) too small,
                setting to 0 (no averaging).".format(
                    overlap, 0.0))
        else:
            newoverlap = 1.0*newoverlap/length
            assert(2*length-int(newoverlap*length) <= self.twf.samples)
            logging.warning("Requested overlap (0) too large, setting to
                (1) (maximum possible).".format(
                    overlap, 1.0))
            newoverlap = 1.0
```

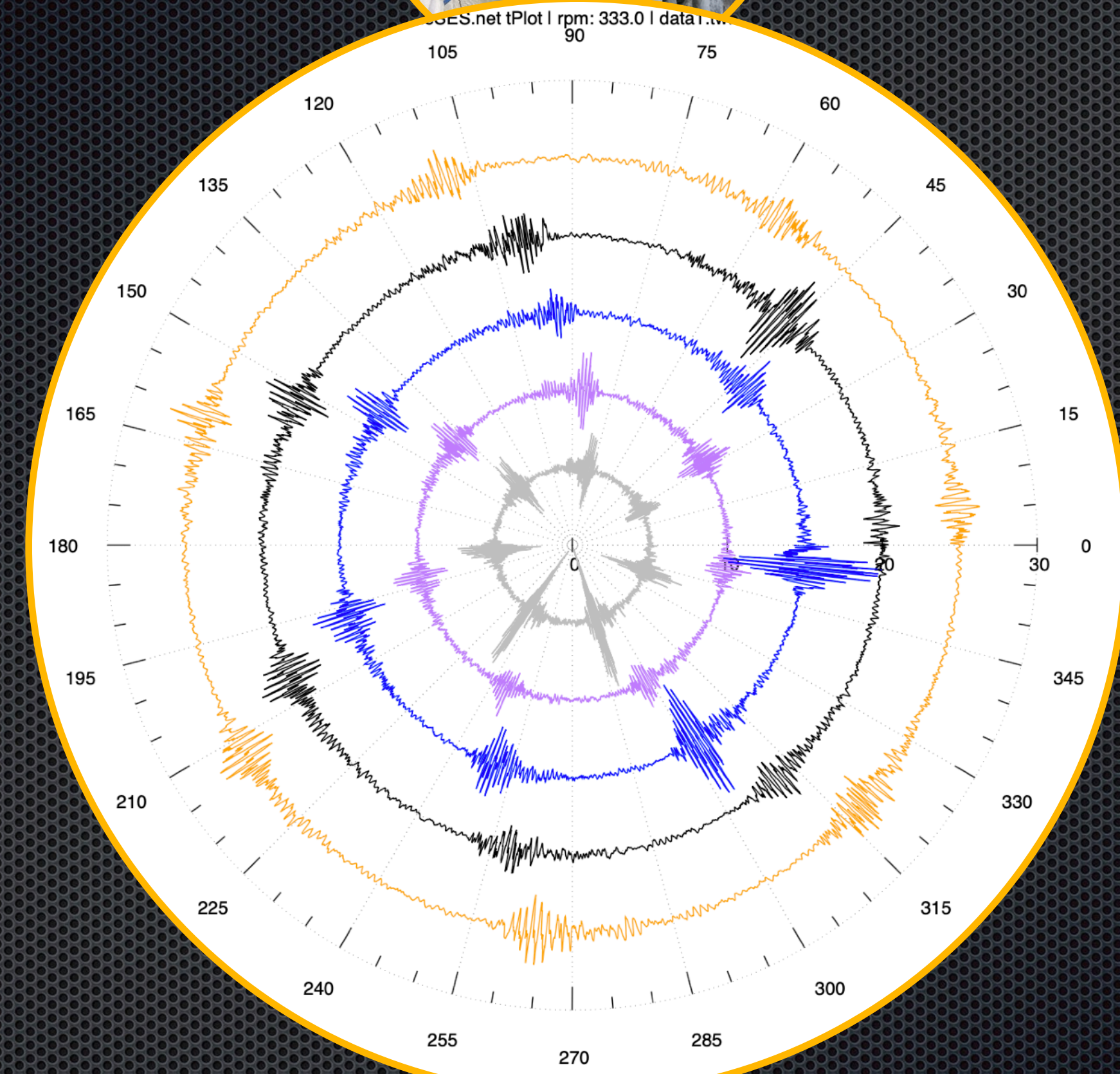
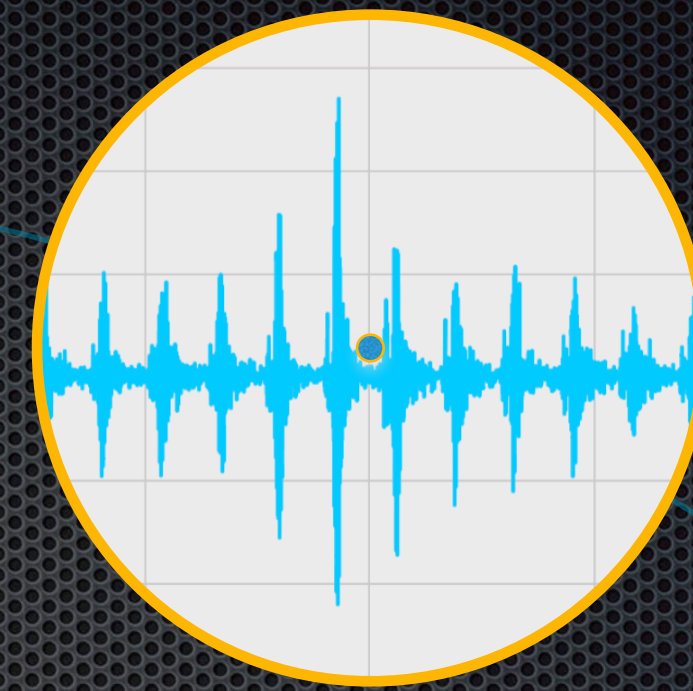
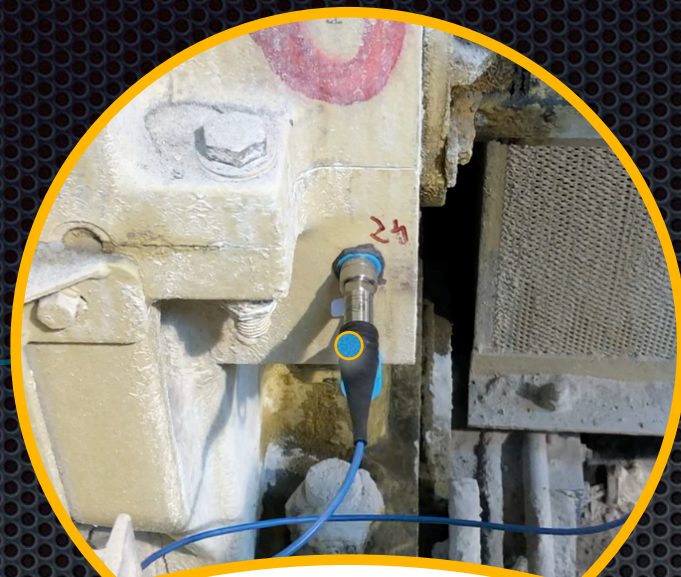
CM Specialist reviews Trends of the computed Parameters



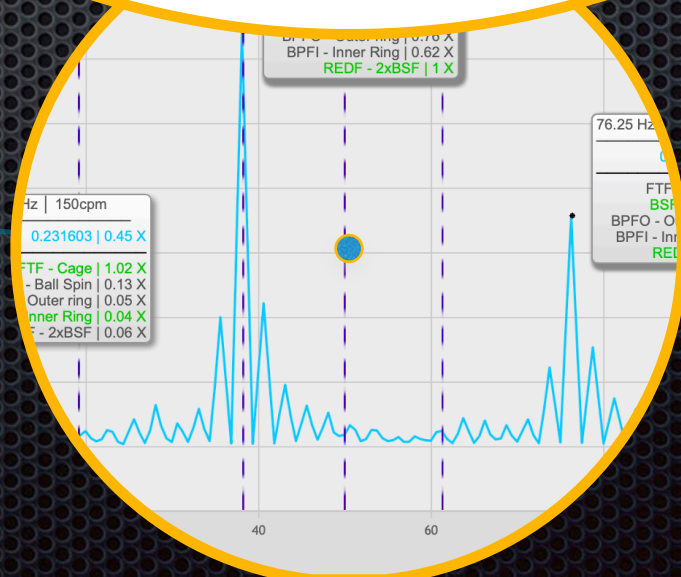
```
def __init__(self, maxfreq, nlines, offset, overlap):
    """
    Parameters for Fourier transforms.
    """
    if maxfreq > self.twf.freq/2.0:
        newmaxfreq = self.twf.freq/2.0
        logging.warning("Requested maximum frequency (0) too high, setting to (1) (half the sampling frequency).")
        maxfreq = newmaxfreq
    length = int((nlines)*self.twf.freq/maxfreq)
    length = self.twf.samples
    if length > self.twf.samples:
        newlines = int(length*maxfreq/self.twf.freq)
        logging.warning("Requested number of lines (0) too large, setting to (1) (maximum possible).")
        nlines = newlines
    if offset < 0 or offset+length > self.twf.samples:
        logging.warning("Requested offset (0) too large, setting to (1) (maximum possible).")
        offset = self.twf.samples - length
    if overlap is not None:
        # FFT averaging functions handle gracefully noverlap=0
        if overlap < 0.0 or overlap >= 1.0:
            logging.warning("Requested overlap (0) should be between 0 and 1, setting to 0 (no averaging).")
            overlap = 0.0
        elif 2*length-int(overlap*length) > self.twf.samples:
            noverlap = 2*length - self.twf.samples
            if noverlap >= self.twf.samples:
                logging.warning("Requested overlap (0) too small (0 no averaging).")
            else:
                noverlap = 1.0*overlap*length
                assert(2*length-int(noverlap*length) > self.twf.samples)
                logging.warning("Requested overlap (0) too large, setting to (1) (maximum possible).")
                noverlap = 1.0
```



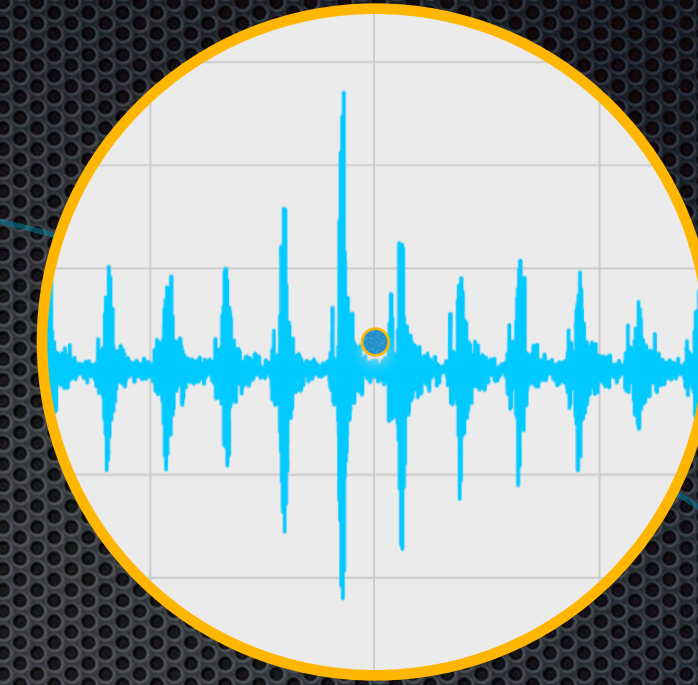
Analyzing Velocity, Acceleration and Enveloping spectra



```
def __init__(self, maxfreq, nlines, offset, overlap):
    """
    Parameters for Fourier transforms.
    """
    if maxfreq > self.twf.freq/2.0:
        newmaxfreq = self.twf.freq/2.0
        logging.warning("Requested maximum frequency (0) too high,
            setting to (1) (half the sampling frequency).".format(
                maxfreq, newmaxfreq))
        maxfreq = newmaxfreq
    length = int((nlines)*self.twf.freq/maxfreq)
    if length > self.twf.samples:
        length = self.twf.samples
    newlines = int(length*maxfreq/self.twf.freq)
    logging.warning("Requested number of lines (0) too large, setting
        to (1) (maximum possible)".format(nlines, newlines))
    nlines = newlines
    if offset < 0 or offset+length > self.twf.samples:
        logging.warning("Requested offset (0) too large, setting to (1)
            (maximum possible)".format(offset, self.twf.samples))
    offset = self.twf.samples - length
    # FFT averaging functions handle gracefully noverlap=0
    if overlap < 0.0 or overlap >= 1.0:
        logging.warning("Requested overlap (0) should be between 0 and
            1, setting to 0 (no averaging)".format(
                overlap, 0.0))
        overlap = 0.0
    elif 2*length-int(overlap*length) > self.twf.samples:
        noverlap = 2*length - self.twf.samples
        if noverlap >= self.twf.samples:
            noverlap = 0.0
            logging.warning("Requested overlap (0) too small
                (no averaging)".format(
                    noverlap, 0.0))
        else:
            noverlap = 1.0*noverlap/length
            assert(2*length-int(noverlap*length)
                >= self.twf.samples)
            logging.warning("Requested overlap (0) too small
                (no averaging)".format(
                    noverlap, 0.0))
```



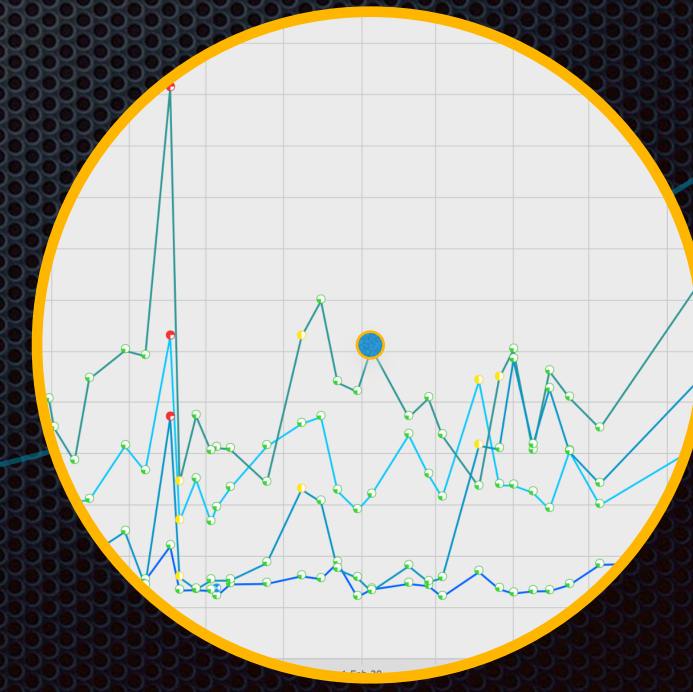
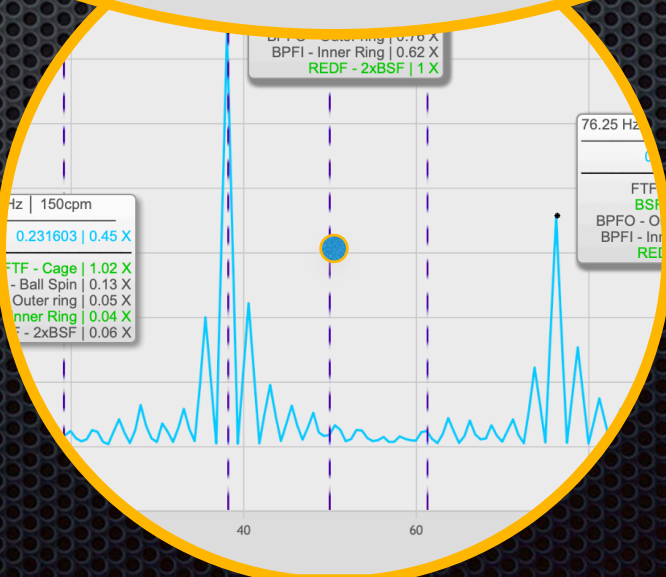
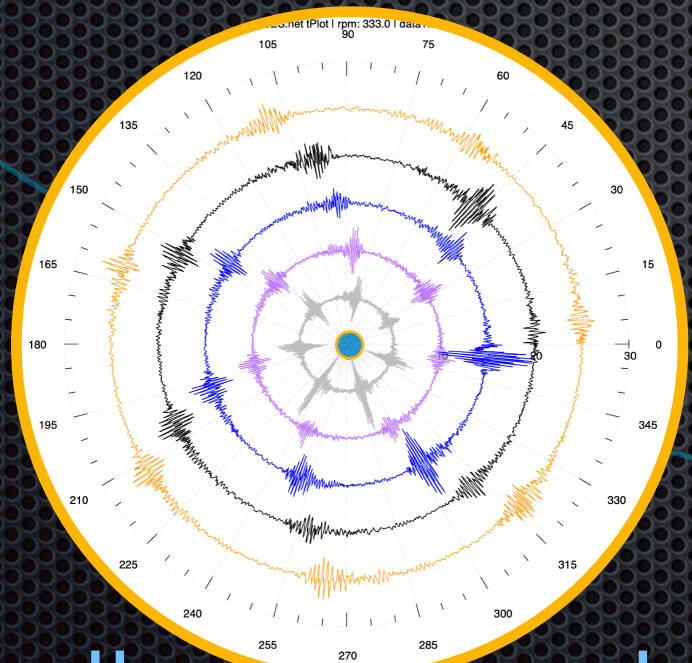
Using advance tools



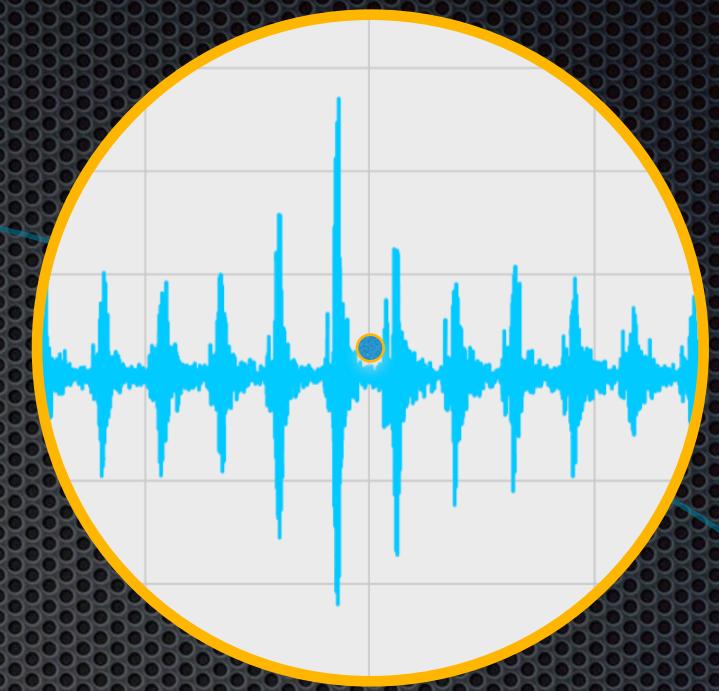
	Fault Notes	Re
Med	Raised Velocity trend. Be	Plan for
Med	Raised 2X and 5X.	Visual Ins
Med	Raised Velocity trend.	Plan for repla
Med	Raised Velocity overall va	Overhaul mac
Low	Raised 1X.	
Low	One point is not collected	
Low	Raising Velocity trend.	
Low	Only two points are collec	Collect
	Raised 3X in Velocity.	Vir

```
def __init__(self, start, stop, fractional_overlap):
    self.start = start
    self.stop = stop
    self.fractional_overlap = fractional_overlap

    (maxfreq, nlines, offset, overlap) = tuple of suggested
    parameters for Fourier transforms.
    if maxfreq > self.twf.freq/2.0:
        newmaxfreq = self.twf.freq/2.0
        logging.warning("Requested maximum frequency (0) too high,
            setting to (1) (half the sampling frequency).".format(
                maxfreq = newmaxfreq))
    length = int((nlines)*self.twf.freq/maxfreq)
    if length > self.twf.samples:
        length = self.twf.samples
    newlines = int(length*maxfreq/self.twf.freq)
    logging.warning("Requested number of lines (0) too large, setting
        to (1) (maximum possible).".format(nlines, newlines))
    nlines = newlines
    if offset < 0 or offset+length > self.twf.samples:
        logging.warning("Requested offset (0) too large, setting to (1)
            (maximum possible).".format(offset, self.twf.samples))
    offset = self.twf.samples - length
    if overlap is not None:
        # FFT averaging functions handle gracefully nverlap=0
        if overlap < 0.0 or overlap >= 1.0:
            logging.warning("Requested overlap (0) should be between 0 and
                1, setting to 0 (no averaging).".format(
                    overlap = 0.0))
            nverlap = 2*length - self.twf.samples
        if nverlap >= self.twf.samples:
            nverlap = 0.0
            logging.warning("Requested overlap (0) too small,
                setting to 0 (no averaging).".format(
                    nverlap = 0.0))
        else:
            nverlap = 1.0*nverlap/length
            assert(2*length-int(nverlap*length)
                >= self.twf.samples)
            logging.warning("Requested overlap (0) too small,
                setting to 0 (no averaging).".format(
                    nverlap = 0.0))
```



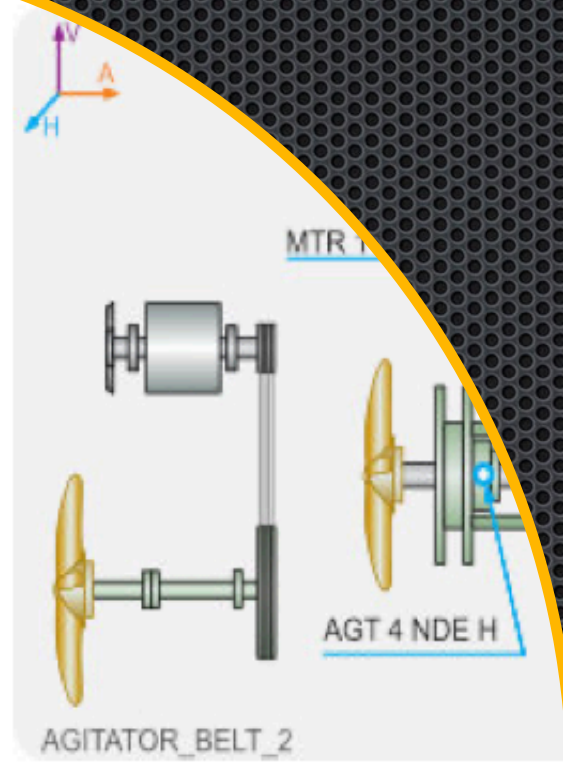
Document the discovered faults,
and give recommendations



	Fault Notes	Recommended Action
High	Raised Velocity trend. Bearing defect frequencies in SE spectrum.	Plan for replacement
Med	Raised 2X and 5X.	Visual Inspection
Med	Raised Velocity trend.	Plan for replacement
Med	Raised Velocity overall value.	Overhaul machine
Low	Raised 1X.	
Low	One point is not collected.	
Low	Raising Velocity trend.	
Low	Only two points are collected.	Collect more data
Low	Raised 3X in Velocity.	Visual Inspection

Asset	Status	Analysed by
AGT 4 NDE BRG 1	ATTENTION	CBM Dreamer

Component	CM Status	CMMS ID
EL MOTOR	ATTENTION	3000206331
AGT 4 NDE BRG 1	ACCEPTABLE	3000567644
AGT 4 NDE BRG 2	ATTENTION	3000206332



```

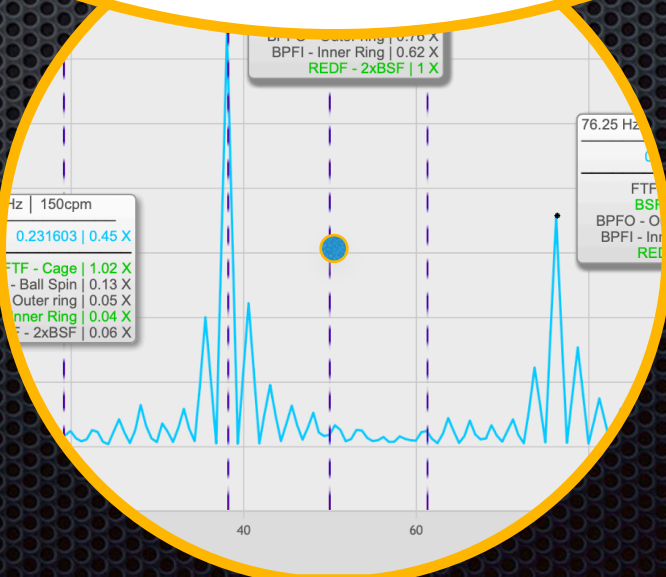
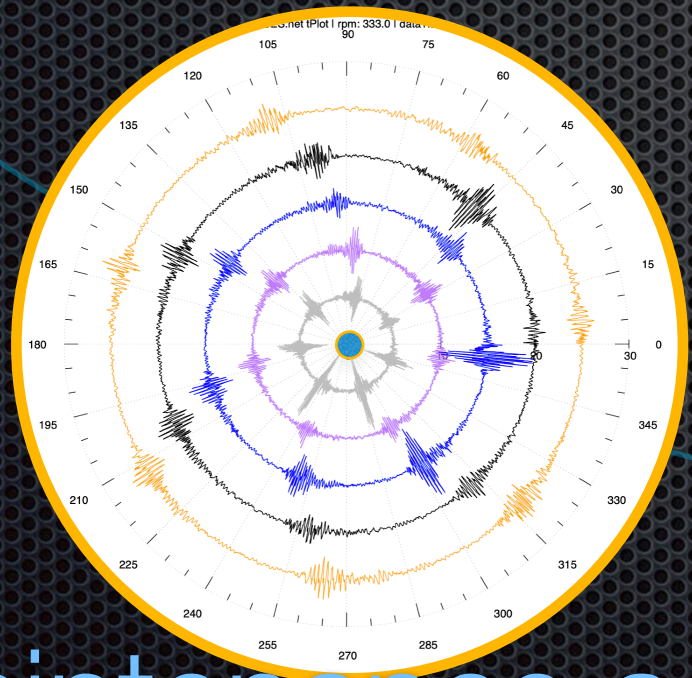
def __init__(self, maxfreq, nlines, offset, overlap):
    """
    Parameters for Fourier transforms.
    """
    self.maxfreq = maxfreq
    self.nlines = nlines
    self.offset = offset
    self.overlap = overlap

    # FFT averaging functions handle gracefully nlines=0
    if self.nlines < 0 or self.nlines > self.samples:
        self.nlines = min(self.nlines, self.samples)

    # Requested maximum frequency (0) too high, setting to (1) (half the sampling frequency).
    if self.maxfreq > self.twf.freq/2.0:
        self.maxfreq = self.twf.freq/2.0
        logging.warning("Requested maximum frequency (0) too high, setting to (1) (half the sampling frequency).")

    self.length = int((self.nlines)*self.twf.freq/self.maxfreq)
    if self.length > self.twf.samples:
        self.length = self.twf.samples
        logging.warning("Requested number of lines (0) too large, setting to (1) (maximum possible).")
    self.newlines = int(self.length*self.maxfreq/self.twf.freq)
    if self.newlines < 1:
        self.newlines = 1
        logging.warning("Requested number of lines (0) too large, setting to (1) (maximum possible).")
    self.offset = self.twf.samples - self.length
    if self.offset < 0 or self.offset > self.twf.samples:
        self.offset = 0
        logging.warning("Requested offset (0) too large, setting to (1) (maximum possible).")
    self.offset = self.twf.samples - self.length
    if self.offset < 0 or self.offset > self.twf.samples:
        self.offset = 0
        logging.warning("Requested offset (0) too large, setting to (1) (maximum possible).")

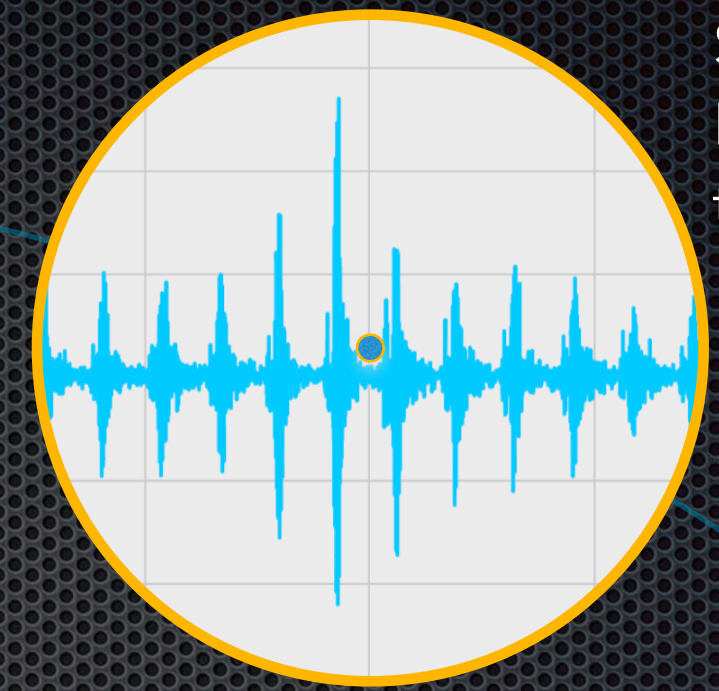
    # FFT averaging functions handle gracefully nlines=0
    if self.overlap < 0.0 or self.overlap > 1.0:
        self.overlap = 0.0
        logging.warning("Requested overlap (0) should be between 0 and 1, setting to 0 (no averaging).")
    self.overlap = 0.0
    elif 2*length-int(overlap*length) > self.twf.samples:
        self.overlap = 2*length - self.twf.samples
        logging.warning("Requested overlap (0) too small, setting to (1) (maximum possible).")
    if self.overlap >= self.twf.samples:
        self.overlap = 0.0
        logging.warning("Requested overlap (0) too small, setting to 0 (no averaging).")
    else:
        self.overlap = 1.0*overlap/length
        assert(2*length-int(newoverlap*length) >= self.twf.samples)
        logging.warning("Requested overlap (0) too small, setting to (1) (maximum possible).")
    self.newoverlap = int(self.length*self.overlap)
    
```



Planing the maintenance activities, based on the recommendations



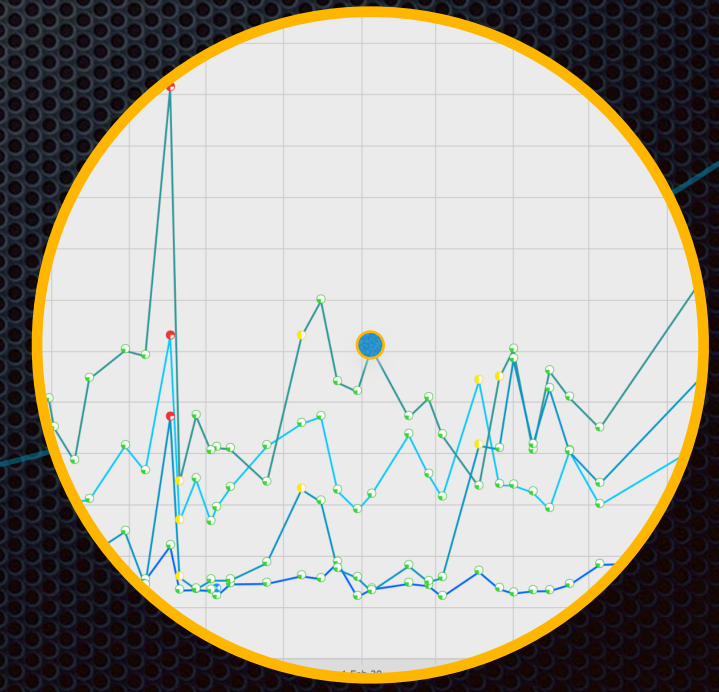
Data collection



Signal Digitization and transfer

```
def __init__(self, maxfreq, nlines, offset, overlap):
    """
    Parameters for Fourier transforms.
    """
    self.maxfreq = maxfreq
    self.nlines = nlines
    self.offset = offset
    self.overlap = overlap
    # FFT averaging functions handle gracefully nlines=0
    if self.nlines < 0 or self.nlines >= 1:
        logging.warning("Requested number of lines (0) too large, setting to (1) (maximum possible).")
        self.nlines = 1
    if self.offset < 0 or self.offset >= self.nlines:
        logging.warning("Requested offset (0) too large, setting to (1) (maximum possible).")
        self.offset = 1
    if self.overlap < 0 or self.overlap >= 1:
        logging.warning("Requested overlap (0) should be between 0 and 1, setting to 0 (no averaging).")
        self.overlap = 0
    self.length = int((self.nlines * self.maxfreq) / self.offset)
    self.newlines = int(self.length * self.offset / self.maxfreq)
    self.nlines = self.newlines
    # FFT averaging functions handle gracefully nlines=0
    if self.nlines < 0 or self.nlines >= 1:
        logging.warning("Requested number of lines (0) too large, setting to (1) (maximum possible).")
        self.nlines = 1
    if self.offset < 0 or self.offset >= self.nlines:
        logging.warning("Requested offset (0) too large, setting to (1) (maximum possible).")
        self.offset = 1
    if self.overlap < 0 or self.overlap >= 1:
        logging.warning("Requested overlap (0) should be between 0 and 1, setting to 0 (no averaging).")
        self.overlap = 0
    self.length = int((self.nlines * self.maxfreq) / self.offset)
    self.newlines = int(self.length * self.offset / self.maxfreq)
    self.nlines = self.newlines
```

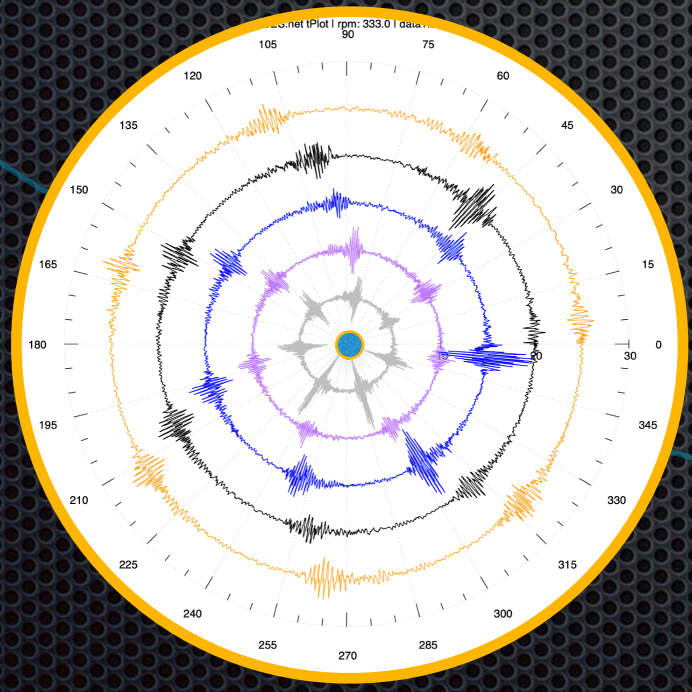
Signal processing



Trends analyses



Spectra analyses



Using customized advance tools

Severity	Fault Notes	Recommendations
Med	Raised Velocity trend. Bearing defect frequencies in SE spectrum.	Plan for replacement
Med	Raised 2X and 5X.	Visual Inspection
Med	Raised Velocity trend.	Plan for replacement
Med	Raised Velocity overall value	Overhaul machine
Low	Raised 1X.	
Low	One point is not collected	
Low	Raising Velocity trend.	
Low	Only two points are collected	Collect more data
Low	Raised 3X in Velocity.	Visual Inspection

Documenting faults and recommendations

CM Status	CMMS ID
ATTENTION	3000206331
ACCEPTABLE	3000567644
ATTENTION	3000206332

Component	Fault	Sev	Recommended Action
EL MOTOR	UNKNOWN - no changes - Raised Velocity trend. Bearing defect frequencies in SE spectrum.	Med	Plan for replacement
AGT 4 NDE BRG 1	Broken component - Raised 2X and 5X.	Med	Visual Inspection

Using recommendations to Act on Time

Site Maintenance: Act based on DATA

Site Organization: It has to be Easy.

Technology: Simplicity & Cost

Understand the data
Know how machines work

Trained & Certified

CM Data Analytics Team

THOUGHTS

- The CBM Journey starts with data collection.
- The technology should simplify instead of complicating the process. The technology only transforms the signal to data.
- The data analysis is a complicated process, and it is the core of transformation from data to information.
- If the Maintenance team is not going to use the CBM Information, don't waste your money for CM technology.