Manual Práctico: Clasificación y Diagnóstico Automático por Vibración para Turbogeneradores

Normas aplicadas: ISO 13373-3:2015 (Guía de diagnóstico por vibración) y ISO 10816 (Partes 1 y 2)
Idioma: Español
Contenido: Explicaciones desde cero, fórmulas y ejemplos de implementación en Python (NumPy/SciPy)

📥 Descargar PDF

---

1. Introducción

Se presenta un manual práctico para implementar un sistema automático de etiquetado y diagnóstico de turbogeneradores, usando únicamente las normas provistas: **ISO 10816** (partes 1 y 2) para clasificación por severidad y **ISO 13373-3:2015** para diagnóstico. El documento está dirigido a estudiantes de Computación sin formación en Ingeniería Mecánica; por tanto, todos los conceptos técnicos se explican desde el inicio y se muestran implementaciones prácticas en Python.

2. Qué hace cada norma (resumen breve y claro)

📋 ISO 10816 (partes 1 y 2)

Define cómo evaluar la severidad de vibración mediante el valor r.m.s. de la velocidad de vibración medido en partes no giratorias (alojamientos de cojinetes, carcasas, base). Proporciona las zonas A–D y rangos numéricos para ciertas máquinas, que se usan aquí como etiquetas automáticas.

📋 ISO 13373-3:2015

Proporciona una guía estructurada de diagnóstico por vibración (diagramas de flujo, tablas de proceso y tablas de fallas). No define límites numéricos para severidad; en cambio indica qué inspecciones y qué características de las señales se deben analizar para identificar la causa probable de una vibración anómala.

3. Etiquetas de salida (mapa desde Zonas ISO)

Se debe usar la siguiente correspondencia para las etiquetas finales del sistema:

Zona ISO	Etiqueta Final
Zona A	Excelente
Zona B	Bueno
Zona C	A revisar (diagnóstico requerido)
Zona D	Acción inmediata (diagnóstico urgente y acción recomendada)

3.1 Determinación de Zonas ISO 10816 - Valores específicos y rangos

Las zonas ISO 10816 se basan en valores RMS de velocidad de vibración medidos en partes no giratorias de las máquinas. Los valores específicos dependen del tipo de máquina, su tamaño y velocidad de operación.

Zonas de Evaluación según ISO 10816

Zona	Descripción	Estado de la Máquina	Acción Recomendada
Zona A	Vibración de máquinas nuevas o recién reparadas	Excelente	Ninguna acción requerida
Zona B	Máquinas en buen estado, operación sin restricciones	Bueno	Operación normal continuada
Zona C	Máquinas tolerables pero requieren atención	A revisar	Diagnóstico requerido, planificar mantenimiento
Zona D	Máquinas con vibración severa, operación peligrosa	Acción inmediata	Parada inmediata y reparación urgente

Rangos específicos para Turbogeneradores (ISO 10816-2)

Aplicable a: Turbinas de vapor y generadores mayores de 50 MW con velocidades rotacionales de 1500 a 3600 RPM

Potencia de la Máquina	Zona A (mm/s RMS)	Zona B (mm/s RMS)	Zona C (mm/s RMS)	Zona D (mm/s RMS)
Turbogeneradores > 50 MW Fundación Rígida	0 - 2.3	2.3 - 4.5	4.5 - 7.1	> 7.1
Turbogeneradores > 50 MW Fundación Flexible	0 - 3.5	3.5 - 7.1	7.1 - 11.0	> 11.0
Máquinas Grandes 15-75 MW	0 - 2.8	2.8 - 5.6	5.6 - 8.9	> 8.9

Implementación práctica en Python

```python
def classify_iso_zone(rms_velocity,
    machine_type='turbogenerator_rigid', power_mw=100):
    """
    Clasifica la zona ISO según el valor RMS de velocidad

    Args:
    rms_velocity: Valor RMS en mm/s
    machine_type: Tipo de máquina ('turbogenerator_rigid',
'turbogenerator_flexible')
    power_mw: Potencia en MW

    Returns:
    tuple: (zona, etiqueta, descripcion)
    """

    # Definir límites según ISO 10816-2
    if machine_type == 'turbogenerator_rigid' and power_mw > 50:
        limits = [2.3, 4.5, 7.1]
    elif machine_type == 'turbogenerator_flexible' and power_mw > 50:
        limits = [3.5, 7.1, 11.0]
    elif 15 <= power_mw <= 75:
        limits = [2.8, 5.6, 8.9]
    else:
        raise ValueError("Tipo de máquina o potencia no soportada")

    if rms_velocity <= limits[0]:
        return ('A', 'Excelente', 'Máquina en estado excelente')
    elif rms_velocity <= limits[1]:
        return ('B', 'Bueno', 'Operación sin restricciones')
    elif rms_velocity <= limits[2]:
        return ('C', 'A revisar', 'Diagnóstico requerido')
    else:
        return ('D', 'Acción inmediata', 'Parada y reparación urgente')

# Ejemplo de uso
zona, etiqueta, descripcion = classify_iso_zone(5.2,
'turbogenerator_rigid', 100)
print(f"Zona {zona}: {etiqueta} - {descripcion}")
```

3.2 Tablas de Fallas y Tablas de Procesos según ISO 13373-3

ISO 13373-3:2015 proporciona un enfoque estructurado para el diagnóstico de fallas mediante tablas de procesos y tablas de fallas que guían sistemáticamente el análisis de vibraciones.

¿Qué son las Tablas de Procesos?

Las tablas de procesos son diagramas de flujo estructurados que definen la secuencia lógica de pasos para el diagnóstico. Están diseñadas para ser utilizadas por profesionales de vibración, ingenieros y técnicos proporcionando un enfoque práctico y estructurado para el diagnóstico de fallas.

Estructura de las Tablas de Procesos:

- **Entrada:** Síntoma inicial o alarma de vibración
- **Proceso:** Serie de pruebas y análisis secuenciales
- **Decisiones:** Puntos de bifurcación basados en resultados
- **Salida:** Diagnóstico probable o siguiente paso

Parámetros utilizados en las Tablas de Procesos

Parámetro	Descripción	Unidad	Uso en Diagnóstico
Amplitud 1X	Amplitud a frecuencia de rotación	mm/s, g	Identificar desbalanceo, excentricidad
Amplitud 2X	Amplitud al segundo armónico	mm/s, g	Detectar desalineación, problemas eléctricos
Relación 2X/1X	Ratio entre segundo y primer armónico	Adimensional	Diferenciar tipos de falla
Fase relativa	Diferencia de fase entre puntos	Grados	Identificar modo de vibración
Espectro envolvente	FFT de la señal demodulada	mm/s, g	Diagnóstico de rodamientos

Parámetro	Descripción	Unidad	Uso en Diagnóstico
Kurtosis	Medida de impulsos en la señal	Adimensional	Detectar impactos tempranos

¿Qué son las Tablas de Fallas?

Las tablas de fallas contienen características específicas de cada tipo de defecto, incluyendo patrones de amplitud y fase en diferentes frecuencias. Sirven como base de conocimiento para comparar los síntomas observados.

Estructura típica de una Tabla de Fallas

Tipo de Falla	Frecuencias Características	Direcciones Afectadas	Patrón de Fase	Otros Indicadores
Desbalanceo	1X dominante	Radial (H y V)	Fase estable entre cojinetes	Aumenta con velocidad²
Desalineación	1X, 2X, 3X	Radial y Axial	Diferencia de fase 180°	2X predominante en axial
Holgura Mecánica	1X, 2X, 3X, múltiples	Todas las direcciones	Fase variable	Aumenta con carga
Falla Rodamientos	BPFO, BPFI, BSF, FTF	Según ubicación del defecto	No aplicable	Mejor en envolvente
Problema Engranajes	GMF ± nX	Línea de engrane	Depende del tipo	Sidebands modulados
Resonancia	Frecuencia natural	Según modo	90° respecto excitación	Pico muy estrecho

Metodología de aplicación de las Tablas

1. **Recopilación de síntomas:** Identificar el síntoma inicial (alarma, ruido, vibración excesiva)
2. **Selección de tabla de proceso:** Elegir la tabla apropiada según el tipo de máquina y síntoma
3. **Mediciones dirigidas:** Realizar las mediciones específicas indicadas en la tabla
4. **Evaluación de criterios:** Comparar resultados con los umbrales definidos
5. **Consulta tabla de fallas:** Buscar coincidencias de patrones en la base de conocimiento
6. **Diagnóstico diferencial:** Descartar fallas con características no coincidentes
7. **Confirmación:** Realizar pruebas adicionales si es necesario

Implementación práctica en Python

```python
class FaultDiagnosisTable:
    def __init__(self):
        # Tabla de fallas basada en ISO 13373-3
        self.fault_patterns = {
            'unbalance': {
                'frequencies': ['1X'],
                'directions': ['radial_h', 'radial_v'],
                'phase_stable': True,
                'amplitude_ratio': {'1X_dominant': True},
                'load_dependency': 'speed_squared'
            },
            'misalignment': {
                'frequencies': ['1X', '2X', '3X'],
                'directions': ['radial_h', 'radial_v', 'axial'],
                'phase_stable': False,
```

```
                'amplitude_ratio': {'2X_1X_ratio': 0.5},
                'axial_component': 'high'
            },
            'looseness': {
                'frequencies': ['1X', '2X', '3X', 'multiple'],
                'directions': ['all'],
                'phase_stable': False,
                'amplitude_ratio': {'harmonics_present': True},
                'load_dependency': 'load_dependent'
            }
        }

    def evaluate_symptoms(self, measurements):
        """
        Evalúa síntomas contra tabla de fallas

        Args:
        measurements: dict con mediciones (amplitudes, fases, ratios)

        Returns:
        list: Lista de fallas probables con scores
        """
        probable_faults = []

        for fault_name, pattern in self.fault_patterns.items():
            score = self.calculate_match_score(measurements, pattern)
            if score > 0.3: # Umbral mínimo
                probable_faults.append({
                    'fault': fault_name,
                    'score': score,
                    'confidence': self.get_confidence_level(score)
                })

        return sorted(probable_faults, key=lambda x: x['score'],
reverse=True)

    def calculate_match_score(self, measurements, pattern):
        """Calcula score de coincidencia con patrón de falla"""
        score = 0.0
        total_criteria = 0

        # Evaluar frecuencias características
        if '1X' in pattern['frequencies'] and measurements.get('A_1X',
0) > 0:
            score += 0.3
            total_criteria += 1
```

```
        # Evaluar ratios de amplitud
        if 'amplitude_ratio' in pattern:
            if '2X_1X_ratio' in pattern['amplitude_ratio']:
                expected_ratio = pattern['amplitude_ratio']
['2X_1X_ratio']
                actual_ratio = measurements.get('A_2X', 0) /
max(measurements.get('A_1X', 1), 0.01)
                if actual_ratio >= expected_ratio:
                    score += 0.4
                    total_criteria += 1

        # Evaluar componente axial (para desalineación)
        if pattern.get('axial_component') == 'high':
            if measurements.get('A_axial', 0) >
measurements.get('A_radial', 0) * 0.5:
                score += 0.3
                total_criteria += 1

        return score / total_criteria if total_criteria > 0 else 0.0

    def get_confidence_level(self, score):
        """Asigna nivel de confianza basado en score"""
        if score >= 0.8:
            return 'Alto'
        elif score >= 0.6:
            return 'Medio'
        else:
            return 'Bajo'

# Ejemplo de uso
diagnosis = FaultDiagnosisTable()
measurements = {
    'A_1X': 3.2,
    'A_2X': 1.8,
    'A_axial': 2.1,
    'A_radial': 3.0
}

results = diagnosis.evaluate_symptoms(measurements)
for result in results:
    print( f" {result['fault']} : {result['score']:.2f} (
{result['confidence']} confianza)" )
```

Información extraída de las Tablas de Proceso

Las tablas de proceso proporcionan:

- **Secuencia lógica:** Orden específico de pruebas para maximizar eficiencia diagnóstica
- **Umbrales de decisión:** Valores críticos para bifurcaciones en el proceso
- **Mediciones requeridas:** Qué parámetros medir en cada paso
- **Configuraciones de equipos:** Settings específicos de instrumentos
- **Criterios de parada:** Cuándo se considera suficiente información
- **Acciones recomendadas:** Próximos pasos según diagnóstico

Ventajas del enfoque estructurado ISO 13373-3

Aspecto	Beneficio	Aplicación Práctica
Sistematización	Proceso repetible y consistente	Resultados comparables entre técnicos
Eficiencia	Minimiza mediciones innecesarias	Reduce tiempo de diagnóstico
Trazabilidad	Documenta proceso de decisión	Facilita auditorías y revisiones
Capacitación	Estructura clara para entrenamiento	Acelera curva de aprendizaje
Automatización	Facilita implementación en software	Permite diagnóstico automático

4. Datos que sería ideal tener

Se debe pedir explícitamente lo siguiente, de estar disponible (cada ítem es necesario para distintos pasos del diagnóstico):

- **Señales de vibración en tiempo:** aceleración (m/s²) o velocidad (mm/s) en 3 direcciones por punto de medición (dos radiales y una axial).
- **Velocidad de rotación (rpm)** sincronizada con las vibraciones (timestamp común).
- **Valores r.m.s.** ya calculados por el sistema de adquisición si existieran (pero se debe poder calcularlos localmente).
- **Parámetros operativos** con timestamps: carga (MW o %), temperatura, presión, eventos (arranque/parada).
- **Metadatos:** posición exacta del sensor (ej. cojinete 1 - extremo A, radial horizontal), sensibilidad del sensor y unidad.

5. Conceptos básicos (explicaciones sencillas)

5.1 ¿Qué es RMS (valor eficaz) y por qué se usa?

El valor r.m.s. (root mean square) de una señal x(t) discretizada x_i, i=1..N se define como:

$$RMS = sqrt((1/N) * sum(x_i^2))$$

Si la señal de vibración está en aceleración a(t) y se desea la RMS de la velocidad v(t) (mm/s), se debe convertir (integrar) la aceleración para obtener velocidad antes de calcular la RMS. La RMS de velocidad se usa en ISO 10816 porque está correlacionada con la energía de vibración y con daños mecánicos.

5.2 ¿Qué es la Transformada Rápida de Fourier (FFT) y para qué se usa?

La FFT convierte la señal del dominio del tiempo x(t) al dominio de la frecuencia X(f). Se debe interpretar el espectro de magnitudes para identificar picos en frecuencias concretas que indican fenómenos físicos (ej.: 1X = frecuencia de rotación, armónicos 2X/3X, frecuencias de engranajes o de rodamientos). En términos discretos, la DFT es:

```
X_k = sum(x_n * exp(-j*2*π*k*n/N)), k=0..N-1
```

En la práctica se usa la implementación eficiente FFT. La frecuencia asociada al bin k es f_k = k·f_s / N, donde f_s es la frecuencia de muestreo (Hz). Se debe escoger f_s y N para obtener la resolución en frecuencia necesaria.

5.3 Definiciones de términos comunes (explicadas)

Término	Definición
1X	Frecuencia de rotación del rotor. Si el equipo gira a RPM, la frecuencia de rotación es f_rotor = RPM/60 (Hz).
2X, 3X	Armónicos del 1X: 2X = 2·f_rotor, 3X = 3·f_rotor. Su presencia y relación con 1X indican desalineamiento u otros fenómenos.
Sidebands	Picos situados a ambos lados de una frecuencia dominante (por ejemplo alrededor de la frecuencia de engrane) que indican modulación en amplitud, típicamente por fallo en engranajes o variaciones de carga.
Envelope / Demodulación	Técnica que extrae la envolvente de una señal filtrada en una banda de resonancia para revelar trenes de impulsos (útil para fallo de rodamientos). Se obtiene aplicando un filtro band-pass y luego la transformada de Hilbert para obtener la envolvente.
Órbita	Traza de la posición del eje en el plano radial (x vs y) obtenida con sensores proximidad en puntos opuestos; útil para análisis dinámico del eje y vibraciones rotativas.

6. Fórmulas clave y cómo se implementan en Python

6.1 RMS (velocidad) - Fórmula y Python

La expresión matemática para RMS (discreta) ya se mostró. En Python (si se tiene un vector vel de velocidad en mm/s):

```python
import numpy as np

def rms(x):
    return np.sqrt(np.mean(np.square(x)))

# ejemplo
# vel = array de velocidades (mm/s)
rms_vel = rms(vel)
```

6.2 De aceleración a velocidad (integración) - explicación y Python

Si el sensor entrega aceleración a[n] (m/s²) y se desea velocidad v[n] (m/s), se realiza una integración numérica. Se recomienda eliminar tendencia y aplicar un filtro high-pass para evitar drift. Una aproximación sencilla:

En notación discreta: v[k] ≈ sum(a[i]·Δt) para i=1..k, donde Δt = 1/f_s

```python
import numpy as np
from scipy import signal

# a: señal de aceleración (m/s^2), fs: frecuencia de muestreo (Hz)
# eliminar tendencia
a_detrend = signal.detrend(a)

# integración por trapecios
dt = 1.0 / fs
v = np.cumsum(a_detrend) * dt # aproximación simple

# opcional: filtrar para eliminar baja frecuencia que causa drift
b, a_f = signal.butter(2, 1.0/(fs/2), 'high') # highpass cut-off 1 Hz
aprox
v = signal.filtfilt(b, a_f, v)
```

6.3 FFT: fórmula y Python

La DFT se dio arriba. En Python, con SciPy/NumPy se procede así:

```python
import numpy as np
from scipy import fftpack

# x: señal en tiempo, fs: frecuencia de muestreo, N = len(x)
X = np.fft.rfft(x) # transformada (solo media banda si x real)
freqs = np.fft.rfftfreq(len(x), 1/fs) # eje de frecuencias en Hz
amp = np.abs(X) / len(x) # amplitud (normalizada)
phase = np.angle(X) # fase
```

6.4 Envelope (demodulación) para rodamientos - explicación y Python

Procedimiento resumido: filtrar la señal de aceleración en una banda de resonancia donde el impacto del defecto excite la estructura, obtener la envolvente (Hilbert) y luego aplicar FFT a la envolvente para ver las frecuencias de fallo (BPFO, BPFI...).

```python
from scipy.signal import butter, filtfilt, hilbert

# x: aceleración, fs: sampling rate
def bandpass(x, fs, low, high, order=4):
    b, a = butter(order, [low/(fs/2), high/(fs/2)], btype='band')
    return filtfilt(b, a, x)

# ejemplo
x_filt = bandpass(x, fs, 2000, 8000) # banda de resonancia (ejemplo)
analytic = hilbert(x_filt)
envelope = np.abs(analytic)

# FFT de la envolvente
EnvX = np.fft.rfft(envelope)
freqs_env = np.fft.rfftfreq(len(envelope), 1/fs)
amp_env = np.abs(EnvX) / len(envelope)
```

7. Frecuencias características de rodamientos y cómo calcularlas

Se deben calcular las frecuencias teóricas de fallo del rodamiento a partir de la geometría del rodamiento y la velocidad de rotación. **Denotaciones:**

Variable	Descripción
f_r	frecuencia de rotación del eje (Hz) = RPM/60
N	número de elementos rodantes
d	diámetro del rodamiento (bola o rodillo)
D	diámetro del círculo primitivo (pitch diameter)
φ	ángulo de contacto (radianes)

Fórmulas típicas:

$$BPFO = (N/2) * f_r * (1 - (d/D)*\cos(\varphi))$$
$$BPFI = (N/2) * f_r * (1 + (d/D)*\cos(\varphi))$$
$$FTF = (f_r/2) * (1 - (d/D)*\cos(\varphi))$$
$$BSF = (D/(2d)) * f_r * (1 - ((d/D)*\cos(\varphi))^2)$$

En Python se debe implementar estas fórmulas y superponer las frecuencias calculadas sobre el espectro para comprobar si aparecen picos en esas frecuencias o en sus armónicos.

```python
def bearing_freqs(rpm, N, d, D, phi_deg):
    fr = rpm / 60.0
    phi = np.deg2rad(phi_deg)

    bpfo = (N/2.0)*fr*(1 - (d/D)*np.cos(phi))
    bpfi = (N/2.0)*fr*(1 + (d/D)*np.cos(phi))
    ftf = 0.5*fr*(1 - (d/D)*np.cos(phi))
    bsf = (D/(2.0*d))*fr*(1 - ((d/D)*np.cos(phi))**2)

    return {'BPFO': bpfo, 'BPFI': bpfi, 'FTF': ftf, 'BSF': bsf}
```

Nota: las fórmulas son estándar en análisis de vibraciones y se deben usar con las dimensiones del rodamiento. Si no se conoce la geometría exacta, se pueden estimar relaciones aproximadas y comprobar la presencia de picos en el espectro.

8. Reglas prácticas (cómo decidir la falla probable) — lógica para programar

Las reglas que se muestran a continuación son heurísticas estructuradas según ISO 13373-3 y práctica industrial. Se debe implementar un sistema que calcule estas métricas y aplique reglas lógicas con scores.

Tipo de Falla	Características	Regla Práctica
Desbalanceo (Unbalance)	Pico dominante en 1X (f_rotor) en las direcciones radiales; fase del 1X coherente entre pedestales.	Si A_1X es el pico dominante y A_1X/energía_total > 0.3–0.5 y la fase entre pedestales es estable → marcar 'Desbalanceo probable'.
Desalineación (Misalignment)	Picos en 1X y 2X (y a veces 3X), axial con contribución significativa; fases entre pedestales muestran desajustes.	A_2X/A_1X mayor que un umbral (ej. 0.2–0.5) y componente axial elevada → 'Desalineación probable'.
Holgura / Piezas sueltas (Looseness)	Múltiples armónicos (1X,2X,3X...), espectro con 'smearing' y picos cambiantes con carga; señales intermitentes.	Presencia de varios armónicos de 1X con energía repartida → 'Holgura probable'.

Tipo de Falla	Características	Regla Práctica
Fallo en rodamientos (Rolling element)	Tren de impulsos; en espectro de la señal cruda puede haber picos a frecuencias características (BPFO/BPFI/BSF/FTF), pero muchas veces se detecta mejor en la FFT de la envolvente (envelope FFT).	Si la FFT de la envolvente muestra picos en las frecuencias calculadas para el rodamiento → 'Fallo de rodamiento probable'.
Fallo en engranajes (Gear)	Picos en la frecuencia de engrane (gear mesh = número de dientes * f_rotor) y sidebands alrededor de esa frecuencia; sidebands regulares indican desgaste de dientes.	Detectar gear mesh y sus sidebands → 'Fallo de engranaje probable'.
Resonancia estructural	Picos muy estrechos y de alta amplitud en una frecuencia natural de la estructura; se acentúan en un rango de velocidad.	Si un pico aumenta dramáticamente en un rango de velocidades (run-up) → 'Resonancia probable'.

9. Procedimiento completo (pipeline que debe ejecutar el programa)

1. **Ingesta de datos:** leer tabla con timestamps, señal de vibración (o RMS precomputado), RPM y parámetros operativos.

2. **Ventaneo y selección de ventanas:** identificar ventanas en las que la RPM y la carga son estables (steady-state) usando umbral de variación. Se debe trabajar sobre ventanas con suficiente longitud para la resolución deseada: resolución $\Delta f = fs / N$.

3. **Cálculo RMS:** calcular RMS de velocidad en cada ventana y asignar Zona ISO (A–D) según los umbrales apropiados para la velocidad nominal.

4. **Decisión inicial:** Si Zona A o B → etiqueta final (Excelente / Bueno). Si Zona C o D → pasar al diagnóstico detallado.

5. **Análisis espectral:** calcular FFT (magnitud y fase), identificar A_1X, A_2X, ratios (A_1X/energía_total, A_2X/A_1X), buscar gear mesh y sidebands.

6. **Análisis de envelope** (si se sospecha rodamiento): band-pass alrededor de resonancia, Hilbert, FFT de la envolvente y búsqueda de BPFO/BPFI/BSF/FTF.

7. **Análisis de fase entre pedestales:** calcular fase relativa del 1X para diferenciar desbalanceo de otros problemas.

8. **Aplicar reglas heurísticas:** con scoring para producir lista ordenada de fallos probables y un 'confidence score'.

9. **Salida:** etiqueta final (Excelente/Bueno/A revisar/Acción inmediata), fallback: si Zona D, marcar 'Acción inmediata' y listar fallos probables con métricas que justifican la decisión.

10. Ejemplos de flujo (casos paso a paso)

Ejemplo 1: Caso sencillo — salida 'Bueno'

Entrada (tabla): columnas: timestamp, rpm, vel_rms_mm_s (medida radial principal), temp, carga.

timestamp	rpm	vel_rms_mm_s	temp_C	carga_%
2025-09-01T10:00:00	1800	4.0	75	60

timestamp	rpm	vel_rms_mm_s	temp_C	carga_%
2025-09-01T10:01:00	1800	4.2	75	60

Proceso:

- Se calcula RMS (ya dado). Se compara con límites ISO para 1800 RPM: 2.8–5.3 mm/s → Zona B.
- Zona B → Etiqueta final 'Bueno'. No se requiere diagnóstico detallado.

Ejemplo 2: Caso con alarma — salida 'Acción inmediata: Desalineamiento probable'

Entrada (tabla simplificada):

timestamp	rpm	vel_rms_mm_s	temp_C	carga_%
2025-09-01T11:00:00	1800	9.2	80	85
2025-09-01T11:01:00	1800	9.5	80	85

Proceso (pasos que el programa ejecuta):

- RMS ~9.3 mm/s para 1800 RPM → supera 8.5 mm/s → Zona D → Etiqueta provisional 'Acción inmediata'.
- Se selecciona ventana steady-state y se calcula FFT: aparece pico significativo en 1X y pico también en 2X, con componente axial marcado.
- Se calcula razón A_2X/A_1X = 0.45 (> umbral heurístico 0.2) y la axial muestra amplitud alta → regla heurística indica 'Desalineamiento probable'.
- Se coteja con datos operativos: la amplitud aumenta con carga → confirma patrón típico de desalineamiento.

- **Salida final:** 'Acción inmediata: Desalineamiento probable'. Se incluye evidencia: RMS, A_1X, A_2X, ratio, fase y registro de rpm/carga.

11. Implementaciones y bibliotecas Python recomendadas

Biblioteca	Uso	Ejemplo de Importación
NumPy	Operaciones numéricas y FFT básica	`import numpy as np`
SciPy	Filtros, Hilbert y utilidades de señal	`from scipy import signal`
pyFFTW	FFT acelerada (opcional) para grandes volúmenes	`import pyfftw`
Pandas	Manejo de tablas con timestamps	`import pandas as pd`
Matplotlib / Plotly	Visualización de espectros y tendencias	`import matplotlib.pyplot as plt`

Importante: Se deben documentar y conservar las ventanas usadas para cada cálculo (audit trail) y almacenar métricas calculadas para entrenamiento de ML.

12. Entregables (formato sugerido)

Archivos de salida recomendados:

- **CSV o base de datos** con: timestamp_inicio, timestamp_fin, rpm_mediana, RMS_vel_mm_s, ZonaISO, EtiquetaFinal, fallo_probable, score, métricas (A_1X, A_2X, ratios, fase_1X), notas.
- **Informes PDF** con evidencia de ventanas en Zona C/D (espectros y envolventes).

- **Dashboard interactivo** para visualizar tendencias y alertas.

13. Referencias (documentos consultados para esta versión)

- **ISO 10816-2:2009** — Mechanical vibration — Evaluation of machine vibration by measurements on non-rotating parts (valores de bandas A–D).
- **ISO 13373-3:2015** — Condition monitoring and diagnostics of machines — Part 3: Guidelines for vibration diagnosis.
- Materiales técnicos de **SKF** y literatura técnica sobre envelope analysis y frecuencias de fallo de rodamientos.
- Documentación de **NumPy/SciPy** para FFT, filtros y transformada de Hilbert.

14. Manejo de datos con timestamps en milisegundos y vibración

En algunos casos, los datos provistos consisten únicamente en timestamps con resolución de milisegundos y valores de vibración.

Cuando se tienen muchas entradas de este tipo:

- Se debe verificar la frecuencia de muestreo efectiva calculando la diferencia promedio entre timestamps consecutivos.
- Con esa frecuencia de muestreo se pueden aplicar las mismas técnicas de FFT, RMS y envelope que se explicaron antes.
- Si existen gaps o irregularidades en los timestamps, se recomienda interpolar o re-muestrear la señal para trabajar con un paso uniforme.

15. Variables críticas que podrían faltar

En la práctica, puede suceder que los datos entregados no incluyan todas las variables mínimas.

Ejemplo: la columna de RPM no necesariamente está disponible

En ese caso:

- Sin RPM no se puede calcular directamente la frecuencia 1X ni los armónicos.
- Una alternativa es usar técnicas de estimación de velocidad de giro a partir de picos dominantes en la FFT.
- Sin carga/temperatura/otros parámetros operativos, se pierde la capacidad de correlacionar vibración con condiciones de operación.

Por tanto, se debe exigir al menos: vibración en tiempo, timestamps confiables y RPM. Todo lo demás enriquece el diagnóstico.

🔧 Manual Completo para Diagnóstico Automático de Turbogeneradores

Versión: 1.0 | **Fecha:** Septiembre 2025

Basado en normas ISO 10816 e ISO 13373-3:2015

📥 Descargar Manual Completo (PDF)

Este manual contiene toda la información necesaria para implementar un sistema automático de clasificación y diagnóstico por vibración.