OpenSD 2023 – Proceedings

Ljubljana, 26. – 27. June 2023

Edited by:
Janko Slavič
Martin Česnik

OpenSD 2023

Ljubljana, 26. – 27. June 2023
CONFERENCE PROCEEDINGS

Edited by:

Janko Slavič
Martin Česnik

Reviewers:

Gregor Čepon
Martin Česnik
Domen Gorjup
Janko Slavič
Ivan Tomac
Klemen Zaletelj

Publisher:

University of Ljubljana, Faculty of mechanical engineering

Aškerčeva 6, 1000 Ljubljana

June, 2023

Pricing:

Proceedings are freely available on the conference homepage.

Contents

Closed Shell Sound Radiation Estimate From Camera Measurements

Sofia Baldini[1] Gianluca Guernieri[1] Domen Gorjup[2] Paolo Gardonio[1] Janko Slavič[2] Roberto Rinaldo[1]

[1] *Università degli Studi di Udine - DPIA, Via delle Scienze 206, 33100, Udine,* (*IT*)
[2] *University of Ljubljana - Faculty of Mechanical Engineering, Askerceva 6 1000 Ljubljana* (*SI*)

Abstract

This paper presents a study on the estimate of the sound radiation field generated by the flexural vibration of a thin-walled cylinder using optical measurements. The cylinder is excited by a time-harmonic point force generated by a shaker located on its interior. The flexural deflection shape response of the cylinder is then reconstructed using frequency-domain triangulation from multi-view video acquisitions taken with a single high-resolution and high-speed camera. The camera is kept in a fixed position with the optical axis pointing to the middle of the cylinder axis. The cylinder is mounted via a turning-joint to a seismic basement such that it can be rotated around its axis to acquire the multiple views with the single camera. The sound field radiated by the cylinder is then reconstructed from a discretised boundary integral formulation. The paper provides insights on both the background theory and the MatLab codes developed for the discretised boundary integration.

Keywords: videogrammetry; flexural vibration measurement; sound radiation measurement

1 Introduction

Normally, the measurement of sound radiation by machinery is carried out in specially dedicated reverberant or anechoic rooms [1-6]. These are rather expensive facilities, which, apart from big industries, are normally available at specially dedicated research and academic institutions that provide measurement services to third parties. Hence, beside the costs for the measurement infrastructure, which, alongside the room construction, involve the acquisition, the conditioning and post processing equipment, the costs for the transport and installation of the machine to be tested are quite relevant too. In some cases where it is difficult or too expensive to move the machinery, sound radiation measurements are taken directly *in situ* where the equipment is installed, although this solution is rather delicate since it should be arranged in such a way as the effects of reflections from walls or partitions, as well as the flanking noise generated by other machineries or plants, are minimised.

This paper investigates a novel measurement approach recently proposed in Refs. [7,8] and verified experimentally in Refs. [9,10], where the sound radiation in free field is reconstructed using the Kirchhoff-Helmholtz integral equation [11] from maps of the machinery flexural vibration acquired with video camera acquisitions. Thus, overall, the sound radiation field is derived from optical measurements of the machinery vibrations and thus it is not influenced by the acoustics of the room where the measurement is taken as well as it is not affected by flanking acoustic sources or background noise. This is quite a relevant advantage since it is expected this methodology can be suitably implemented for *in situ* measurements without the need of moving the equipment into special infrastructures. Moreover, the camera acquisitions provide full field vibration measurements such that the sound radiation can be estimated with great accuracy [10]. In this study a stereo acquisition

approach is employed, where either the camera or the machine are rotated around the vertical axis of the machine itself to generate the indeed the stereo acquisition [12,13].

This paper presents the proposed measurement methodology and some initial measurement results of the work currently in progress, which, nevertheless refer to reference optical acquisitions taken with a laser vibrometer. The paper is structured in three sections. Section 2 presents the measurement setup and the image acquisition and post-processing necessary to reconstruct the transverse displacements and sound radiation generated by the flexural vibrations of the cylinder. Then, Section 3 presents the reference results based on laser vibration measurements.

2 Cylinder and measurement setup

2.1 Thin-walled baffled cylinder radiating shell

Figure 1a shows three sketches of the thin-walled baffled cylinder considered in this study. The thin walled cylinder is made of steel and has thickness $h = 1$ mm, radius $R = 149$ mm and height H = 296 mm. As depicted in Figures 1b, 1c the radial displacements generated by the cylinder flexural vibrations have been measured with a high frame rate camera and a laser vibrometer at a regular grid of points. Then the sound radiation has been derived on a vertical and horizontal plane starting from the Kirchhoff – Helmholtz integral formulation [11].

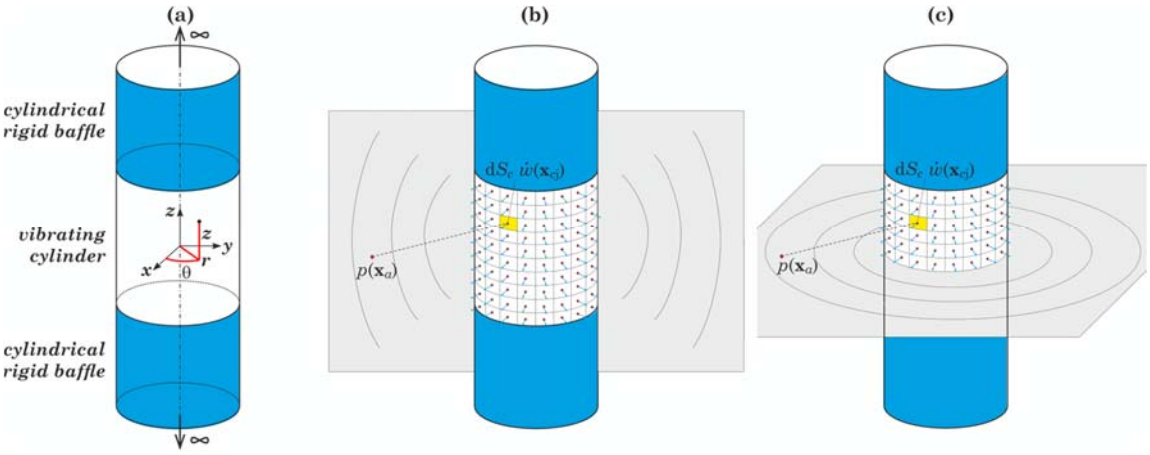


Fig. 1: (a) Baffled cylinder with (b) mesh of radiating elements. Sound radiation in (c) vertical plane and (d) horizontal plane.

As shown in Figure 2a, the cylinder has been assembled on two flanges such that it can be considered fixed along the two circular rims. The bottom flange is joined to a base flange via a turning joint such that the cylinder was rotated during the measurement campaign with the optical camera and with the laser vibrometer. Figure 2b shows the speckle pattern glued on the cylinder surface for the measurement with the optical camera shown in Figure 2c. Also, Figure 2c shows the grid of circular markers located at the centres of the mesh or radiating elements, which was glued on the cylinder surface for the measurement with the laser vibrometer shown in Figure 2d.

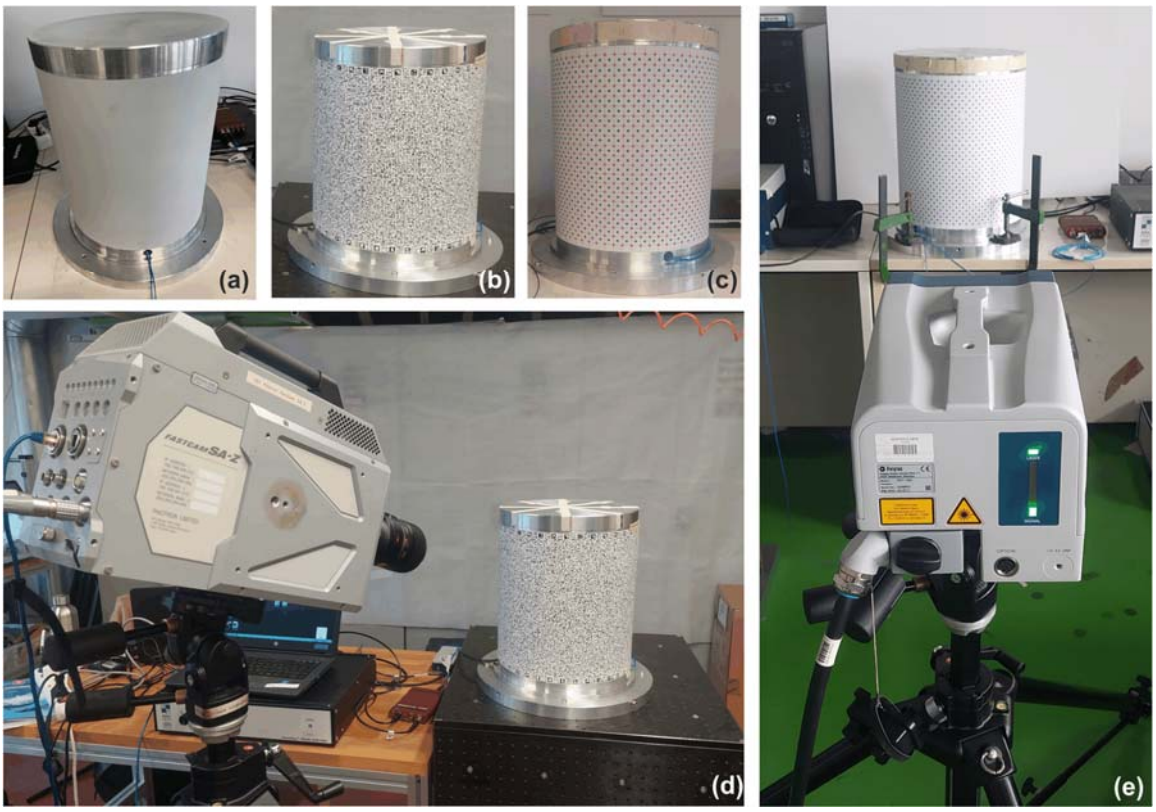


Fig. 2: (a) Flanged cylinder, (b) cylinder with the speckled surface, (c) cylinder with the circular markers, (d) optical camera measurement, (e) laser vibrometer measurement.

2.2 Camera acquisitions and image processing to reconstruct the markers displacements

The implementation of the camera measurement was organised in three phases.

- First, the calibration of the camera's intrinsic parameters was performed using images of a standard ChArUco calibration pattern and the OpenCV image processing library.
- Second, the acquisition of the images was performed using a single Photron FASTCAM SA-Z high-speed camera at the rate of 20.000 frames-per-second. The cylinder was rotated around its vertical axis by approximately 30 degrees between measurements, to achieve 12 different views of the specimen. A structured array of ArUco markers was applied to the surface of the cylinder (Figure 2b) to facilitate extrinsic calibration in post-processing.
- Third, the post processing of the recordings is perfomed to reconstruct the actual radial displacements at the centre positions of the mesh of radiating pistons from the image recordings of the speckled pattern.

The stationary flexural response of the cylinder excited by a stationary white noise point force is reconstructed using multi-view triangulation of image-based displacement measurements in the frequency domain, producing full-field 3D displacement spectra at the selected points on the specimen surface [12]. The method is based on the brightness-constancy constraint of optical flow [15] and enables full-field 3D displacement identification for high-frequency stationary vibration measurement of linear structures using a single, moving high-speed camera. Expanding on the Spectral Optical Flow (SOFI) method [14] (not utilized in this research), the frequency-domain triangulation approach also enables image-based measurement of 3D high-frequency vibrations using

a single still-frame camera [13]. With this methodology, the frequency of the measurement is not dictated by the image-acquisition rate of the camera, but rather by harmonically controlling the light source. Hence, the frequency range of the measurement can be very wide, and a still-frame camera can be used for the image acquisition.

The proposed spatial deflection shape measurement method has two stages. First, the 2D deflection shape spectral components are measured in multiple camera views (e.g., by rotating the cylinder around its axis) using the Lucas-Kanade algorithm of Digital Image Correlation [16]. In this step, the deflection shape images at different observed response frequencies are constructed using a single high-speed monochrome camera.

In the second stage, the multiview deflection shape images are passed on to the frequency-domain triangulation algorithm [12], producing full-field spatial ODSs of the cylinder at selected frequencies, as is illustrated in Fig. 3.

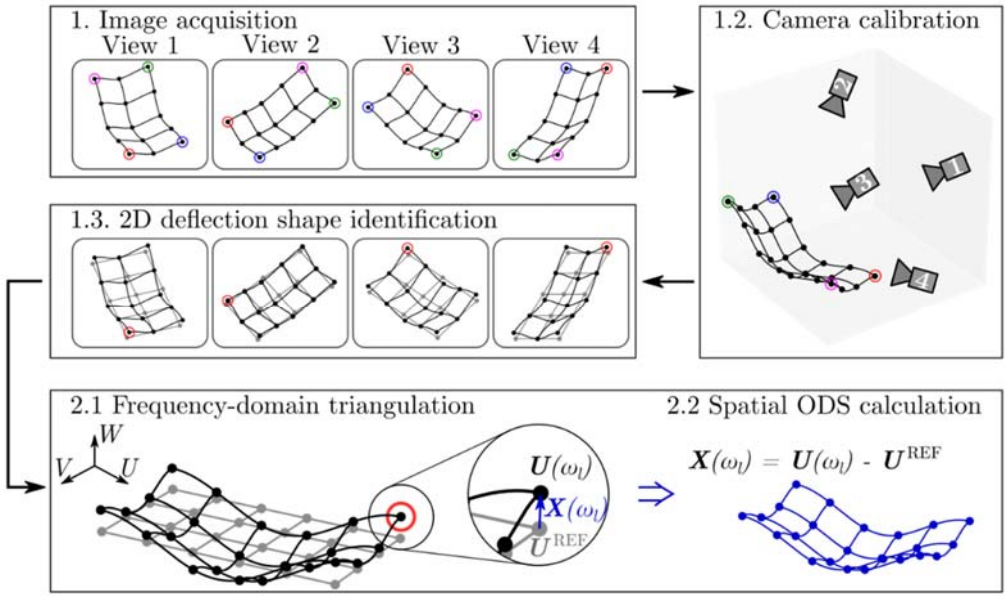


Fig. 3: The image-based 3D displacement measurement procedure using frequency-domain triangulation and the single-camera multiview approach.

2.3 Reconstruction of the flexural vibration field

Once the positions of the grid of markers were reconstructed in cylindrical coordinates, the flexural displacements $w(R,\theta_j,z_j,t_k)$ at the marker positions (R,θ_j,z_j) were taken directly from the radial components such that, for the time-harmonic excitation with circular frequency ω

$$w(R,\theta_j,z_j,t_k) \cong \mathrm{Re}\{\dot{w}(R,\theta_j,z_j,\omega)e^{j\omega t_k}\}\,. \tag{1}$$

where t_k is the kth time sample and $\dot{w}(R,\theta_j,z_j,\omega) = w_0(R,\theta_j,z_j,\omega)e^{j\varphi_0(R,\theta_j,z_j,\omega)}$ is the frequency-dependent complex amplitude of the velocity at the centre position of the radiating elements (R,θ_j,z_j). Here, $w_0(R,\theta_j,z_j,\omega)$ and $\varphi_0(R,\theta_j,z_j,\omega)$ are the amplitude and phase of the displacement at each marker point (for synchronous vibrations $\varphi_0(R,\theta_j,z_j,\omega)$ is bound to be either $\varphi_0(\omega)$ or $\varphi_0(\omega)+\pi$). Since the camera operated at a frame rate $1/T$, the amplitude and phase terms, $w_0(R,\theta_j,z_j,\omega)$ and $\varphi_0(R,\theta_j,z_j,\omega)$, were derived from the sequence of r-coordinates $r_j(t_k)$ of the jth marker centre position assuming $k = 0,\dots,N-1$. For instance, one can choose the data length N

so that the sequence contains an integral number of periods of the sinusoid. In such a case, the coefficients of the Discrete Fourier Transform (DFT) of the data, computed for example via the Fast Fourier Transform (FFT) algorithm, give exactly the required modulus and phase [17].

In this study, the overall flexural vibration was first assessed with respect to the time-averaged total flexural kinetic energy of the thin walled cylinder, which is given by the following expression:

$$\overline{K} = \lim_{T\to\infty} \frac{1}{T} \int_0^T \int_{S_c} \frac{1}{2} \rho_c \, h \dot{w}(R,\theta,z,t)^2 dS_c \, . \tag{2}$$

For time harmonic vibrations, this equation becomes

$$\overline{K}(\omega) = \frac{\rho_c h}{4} \int_{S_c} |\dot{w}(R,\theta,z,\omega)|^2 \approx \frac{M_c}{4N_e} \sum_{j=1}^{N_e} \left|\dot{w}\left(R,\theta_j,z_j,\omega\right)\right|^2 \tag{3}$$

where M_c is the mass of the cylinder and N_e is the number of radiating elements.

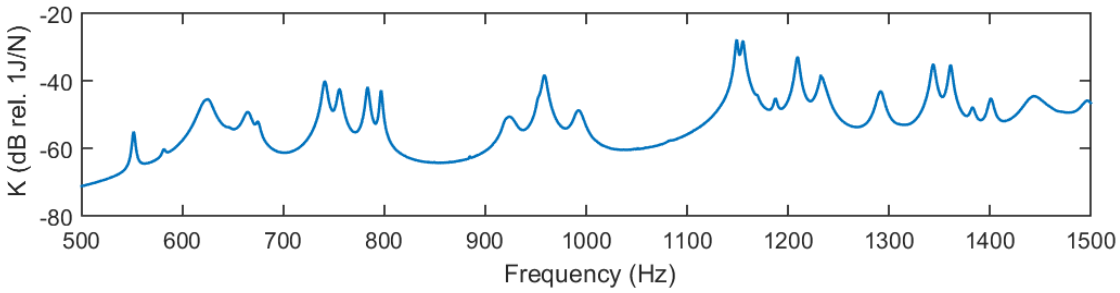


Fig. 4: Spectrum of the total flexural kinetic energy from laser vibrometer measurements.

Figure 4 shows the spectrum of the total flexural kinetic energy derived with Eq. (3) with the complex velocities per unit force exerted by the shaker, i.e. the mobility Frequency Response Functions (FRFs), measured at the centre of the radiating elements with the laser vibrometer. The plot shows that in the 500 – 1500 Hz frequency of interest the flexural response of the cylinder is characterised by distinct resonance peaks. This suggests that at the resonance frequencies the flexural deflection shape of the cylinder is controlled by the mode shape of the resonant mode. Nevertheless, since the cylinder is excited by a point force source, it is expected that, even at resonance frequencies, the deflection shapes are slightly distorted by the point excitation, which also generates the typical nearfield effect [11,18].

2.4 Reconstruction of the sound radiation field

The acoustic pressure $p(\mathbf{x}_a, t)$ in the radiation field has been reconstructed considering the Kirchooff – Helmholtz formula for the sound radiation in free field by a flexible closed shell [11]:

$$p(\mathbf{x}_a, t) = \mathrm{Re}\left\{ \frac{1}{c(\mathbf{x}_a)} \int_{S_c} \left(p(\mathbf{x}_c, t) \frac{\partial g(|\mathbf{x}_a - \mathbf{x}_c|)}{\partial n} + j\omega\rho_0 g(|\mathbf{x}_a - \mathbf{x}_c|) \dot{w}(\mathbf{x}_c, t) \right) dS_c \right\}, \tag{4}$$

which refers to the acoustic pressure $p(\mathbf{x}_c, t)$ and the radial velocity $\dot{w}(\mathbf{x}_c, t)$ on the surface of the cylinder. Here $\mathbf{x}_a = (r, \theta, z)$ identifies the position in the sound radiation field in cylindrical coordinates whereas $\mathbf{x}_c = (R, \theta, z)$ gives the position in cylindrical coordinates on the surface of the cylinder. Also

$$g(|\mathbf{x}_a - \mathbf{x}_c|) = \frac{e^{jk|\mathbf{x}_a - \mathbf{x}_c|}}{4\pi|\mathbf{x}_a - \mathbf{x}_c|} \, . \tag{5}$$

is the free-space 1st-kind Green's function. As shown in Refs. [19], this integral equation can be reworked into an explicit formula that gives the sound radiation with respect to the radial vibration of the cylinder $\dot{w}(\mathbf{x}_c, t)$ only:

$$p(r,\theta,z,t) = \mathrm{Re}\left\{\frac{j\omega\rho_0}{4\pi^2}\int_0^{2\pi}\int_0^{L}\dot{w}(R,\theta',z',\omega)e^{j\omega t}\sum_{n=-\infty}^{+\infty} cos\big(n(\theta-\theta')\big)I_n(z-z')\,d\theta' dz'\right\}, \quad (6)$$

where

$$I_n(z-z') = \int_{-\infty}^{+\infty}\frac{cos\big(k_z(z-z')\big)}{\sqrt{k^2-k_z^2}R}\frac{H_n^{(1)}\left(\sqrt{k^2-k_z^2}r\right)}{H_n^{(1)'}\left(\sqrt{k^2-k_z^2}R\right)}dk_z\,. \quad (7)$$

In these two equations, ρ_0 is the density of air, $k=\omega/c$ is the acoustic wavenumber and c is the speed of sound in air. Also, k_z is the projection of the acoustic wavenumber into the longitudinal direction of the cylinder. Finally, $H_n^{(1)}$ $H_n^{(1)'}$ are the 1st-kind Hankel function and its derivative respectively.

The surface integral in Eq. (4) can calculated numerically considering the mesh of rectangular radiators with a Reimann quadrature such that

$$p(r,\theta,z,t) \approx \mathrm{Re}\left\{\frac{j\omega\rho_0 A_c}{2\pi^2}\sum_{j=1}^{N_e}\dot{w}\big(R,\theta_j,z_j,\omega\big)e^{j\omega t}\sum_{n=-\infty}^{+\infty} cos\left(n\big(\theta-\theta_j\big)\right)I_n\big(z-z_j\big)\right\}. \quad (8)$$

In this equation, A_c represents the area of the surface element at position θ_j, z_j, and N_e is the number of surface elements. After a few mathematical steps, this equation is further simplified into the following expression

$$p(r,\theta,z,t) \approx \mathrm{Re}\left\{\frac{j\omega\rho_0 A_c}{2\pi^2}\sum_{j=1}^{N_e}\dot{w}\big(R,\theta_j,z_j,\omega\big)e^{j\omega t}\sum_{n=0}^{N}\varepsilon_n cos\left(n\big(\theta-\theta_j\big)\right)\hat{I}_n\big(z-z_j\big)\right\}, \quad (9)$$

where $\varepsilon_n = 1$ for $n=0$, $\varepsilon_n = 2$ for $n>0$, and the infinite summation is approximated into the sum of the first $N+1$ terms $\hat{I}_n$ which contribute to the real part

$$\hat{I}_n\big(z-z_j\big) = \int_0^{k}\frac{cos\big(k_z(z-z_j)\big)}{\sqrt{k^2-k_z^2}R}\frac{H_n^{(1)}\left(\sqrt{k^2-k_z^2}r\right)}{H_n^{(1)'}\left(\sqrt{k^2-k_z^2}R\right)}dk_z\,. \quad (10)$$

This integral was also approximated into a Reimann sum, such that

$$\hat{I}_n\big(z-z_j\big) = \sum_{m=0}^{M}\frac{cos(m\Delta k_z(z-z_j))}{\sqrt{k^2-(m\Delta k_z)^2}R}\frac{H_n^{(1)}\left(\sqrt{k^2-(m\Delta k_z)^2}r\right)}{H_n^{(1)'}\left(\sqrt{k^2-(m\Delta k_z)^2}R\right)}\Delta k_z\,. \quad (11)$$

where $M = k/\Delta k_z$. Appendix A reports the MatLab code that has been developed and used in this study to generate the acoustic maps depicted in Figure 5.

3 Measurement results

Figure 5 shows the initial results of this research project, which have been generated from optical measurements taken with the laser vibrometer using the formulation presented in Sections 2.3 and 2.4. These results will be used as a benchmark to assess the methodology proposed in this paper for the estimate of the cylinder flexural vibration and sound ration starting from optical measurements taken with the single camera approach described in Section 2.2 and shown in Figure 2d.

The maps of the flexural deflection shapes taken at three resonance frequencies are controlled by specific flexural modes shapes of the cylinder. For instance, the first deflection shape at 622 Hz is characterised by the mode shape with circumferential number 5 and axial number 1. Also, the second deflection shape at 741 Hz is characterised by the mode shape with circumferential number 4 and axial number 1. Finally, the third deflection shape at 1344 Hz is characterised by the mode shape with circumferential number 10 and axial number 2. The maps are slightly distorted by the point force excitation, which somewhat twists the mode shapes and, also, generates the typical nearfield vibration in the vicinity of the excitation point [18].

For the frequency range considered in this paper, the maps of the sound radiation fields depicted in Figure 5, show the actual acoustic near fields in a horizontal midplane and a vertical plane, which result from the interference of the sound radiation generated by the whole surface of the cylinder. For

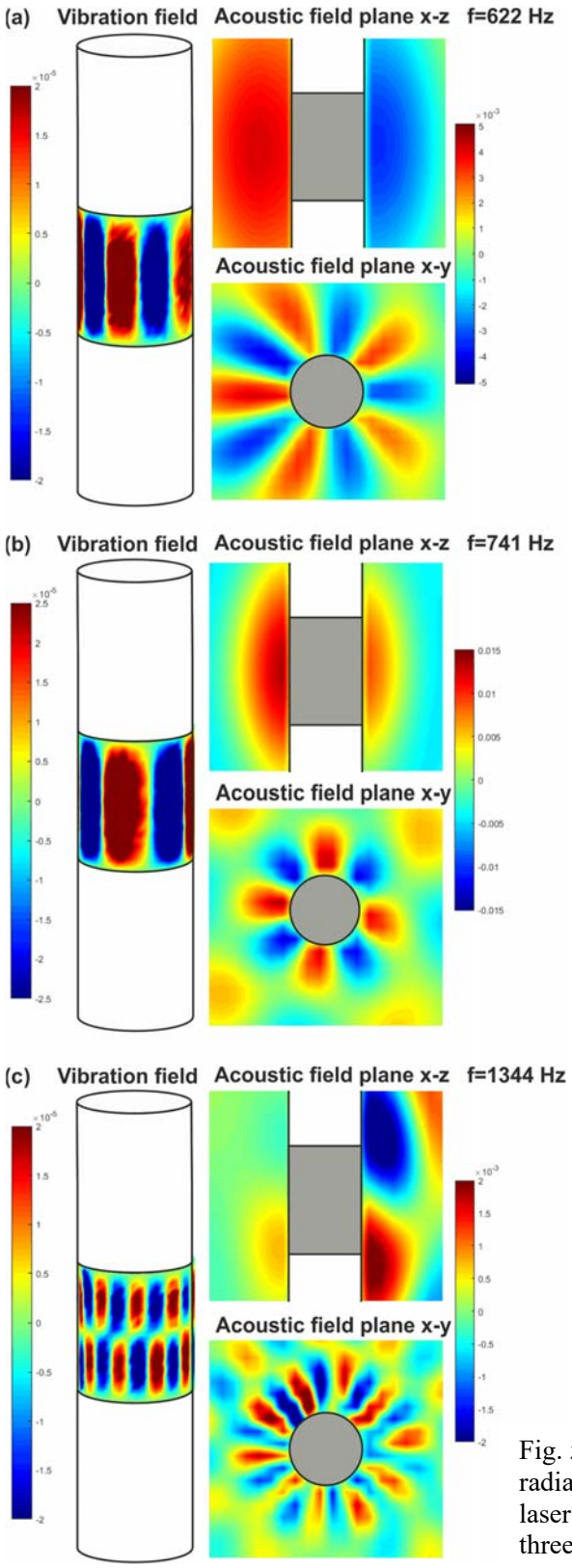


Fig. 5: Vibration and sound radiation fields reconstructed from laser vibrometer measurements at three resonance frequencies.

the baffled cylinder, the far-field is bound to be characterised by cylindrical waves whose amplitude progressively fades in axial direction. At 622 Hz and 741 Hz the horizontal maps are thus characterised respectively by 10 and 8 out of phase acoustic lobes, whereas the vertical maps are characterised by a single acoustic lobe. Instead, at 1344 Hz, the horizontal map, which was taken at 3/4 of the cylinder height, is characterised by 10 alternating acoustic lobes and the vertical map is characterised by two out of phase lobes. The acoustic near fields replicates the flexural vibration patterns at the given frequencies, with the typical radial pattern that forms a lobe-time acoustic field.

These initial results provide a benchmark for the future work where the vibration field will be derived from the camera measurements and the frequency triangulation approach described in Section 2.2.

4 Conclusions

This paper has presented a new methodology for the estimate of the flexural response and sound radiation of a closed shell structure using a single high-speed video camera. The paper has briefly described the methodologies proposed to reconstruct from optical measurements taken with a single camera the flexural vibration of the shell and then to derived the sound radiation into free field. Then it has reported initial results for the flexural vibration and sound radiation of a baffled cylinder derived from optical measurements taken with a laser vibrometer. Future work will prove the proposed methodology starting from measurements taken with a fast optical camera.

Appendix A. MatLab code for the derivation of the sound radiation field

This appendix reports the MatLab codes used to derive the sound radiation field from the flexural vibration of the cylinder measured with optical methods as discussed in Section 2.

```
function p=acoust_press(r,theta,z, a, sourcethetaz, sourcev, f0, c0, rho0, S, M, tol, Madd)

% Computes the contrbution of all source elements
% to the target at r,theta,z. We are usually interested in
% the real part of the result.
%
% r,theta,z, coordinates of the target point
% a cylinder radius
% sourcethetaz Ne x 2 matrix with the theta, z coordinates of the source
%   elements on the cylinder
% sourcev Ne x 1 vector with the velocities of the source points in sourcethetaz
% f0 frequency
% c0 speed of sound
% rho0 density of medium
% S area element
% M number of ponts between 0 and k (default M=180)
% tol tolerance to stop integration (default tol=1e-6)
% Madd additional terms for kz beyond k - imaginary argument (default Madd=0)

if nargin==10
    tol=1e-6;
    Madd=0;
    M=180;
end

Ne=size(sourcethetaz,1);
p=0;
```

```
for i=1:Ne
    thetas=sourcethetaz(i,1);
    zz=sourcethetaz(i,2);
    v=sourcev(i);
    p=p+zij(r,theta,z, a, thetas, zz, f0, c0, rho0, S, v, M, tol, Madd);
end


function p=zij(r,theta,z, a, thetas, zz, f0, c0, rho0, S, v, M, tol, Madd)

% Computes the contribution of a single source element
% at a, thetas, zz to the target at r,theta,z
% r,theta,z, coordinates of the target point
% a, thetas, zz coordinates of source point
% a cylinder radius
% f0 frequency
% c0 speed of sound
% rho0 density of medium
% S area element
% v velocity of source element
% M number of ponts between 0 and k
% tol tolerance to stop integration (default tol=1e-6)
% Madd additional terms for kz beyond k - imaginary argument (default Madd=0)

if nargin==11
    tol=1e-6;
    Madd=0;
    M=180;
end

omega=2*pi*f0;
k=omega/c0;
delta=k/M;

p=1i*rho0*omega*S*v*delta/(2*pi^2);

n=0;
addi=0;
for m=0:M+Madd-1
    kz=m*delta;
    arg1=sqrt((k^2-kz^2)*a);
    arg2=sqrt((k^2-kz^2)*r);
    if ((arg1~=0) & (arg2~=0) )
        addi=addi+(cos(kz*(z-zz))*besselh(n,1,arg2))/...
            (n*besselh(n, 1, arg1) - arg1*besselh(n + 1, 1, arg1));
    end
end

add=addi;
en=2;
while (abs(imag(addi))>tol)
    n=n+1;
    addi=0;
    for m=0:M+Madd-1
        kz=m*delta;
        arg1=sqrt((k^2-kz^2)*a);
        arg2=sqrt((k^2-kz^2)*r);
        if ((arg1~=0) & (arg2~=0) )
            addi=addi+(cos(kz*(z-zz))*besselh(n,1,arg2))/...
                (n*besselh(n,1,arg1) - arg1*besselh(n+1,1,arg1));
        end
    end
    add=add+en*cos(n*(theta-thetas))*addi;
end

p=p*add;
```

References

[1] R. F. Barron, Industrial Noise Control and Acoustics, Marcel Dekker Inc., New York, 2003.

[2] D. A. Bies, C. H. Hansen, Engineering Noise Control Theory and Practice, Spon Press, London, 2009.

[3] F. Fahy, Measurement of audio-frequency sound in air, in: Fundamentals of Sound and Vibration, CRC Press, 2015.

[4] Acoustics - Determination of sound power levels of noise sources – Guidelines for the use of basic standards, ISO 3740:2019 (2019).

[5] E. Petersen, An Overview of Standards for Sound Power Determination, Application Note, Bruel & Kjear Instruments, BO 0416 - 12, 1995.

[6] Acoustics - Measurement of sound insulation in buildings and of building elements using sound intensity - Part 1: Laboratory measurements; - Part 2: Field measurements; Part 3: Laboratory measurements at low frequencies, ISO 15186-1:2000 (2000).

[7] P. Gardonio, R. Rinaldo, R. Del Sal, L. Dal Bo, E. Turco, A. Fusiello, Multi-view videogrammetry for the estimate of plate sound radiation, in: Proceedings of the 14th Intl Conference on Vibration Measurements by Laser and Noncontact Techniques, 2021.

[8] P. Gardonio, R. Rinaldo, L. Dal Bo, R. Del Sal, E. Turco, A. Fusiello, Freefield sound radiation measurement with multiple synchronous cameras, Measurement 188 (2022) 110605. doi:https://doi.org/10.1016/j.measurement.2021.110605.

[9] P. Gardonio, G. Guernieri, R. Rinaldo, A. Fusiello, E. Turco, Plate sound radiation estimate from vibration measurements with multiple cameras, in: Proceedings of INTER NOISE 2022, Glasgow (UK), 2022.

[10] P. Gardonio, G. Guernieri, E. Turco, L. Dal Bo, R. Rinaldo, A. Fusiello, Reconstruction of the sound radiation field from flexural vibration measurements with multiple cameras. Mechanical Systems and Signal Processing 195 (2023) 110289

[11] F. Fahy, P. Gardonio, Sound and Structural Vibration. Radiation, Transmission and Response, 2nd Ed., Academic Press, London, 2007.

[12] D. Gorjup, J. Slavič, M. Boltežar, Frequency domain triangulation for full-field 3D operating-deflection-shape identification, Mech. Syst. Signal Pr. 133 106287 (2019) 143–152.

[13] D. Gorjup, J. Slavič, A. Babnik, M. Boltežar, Still-camera Multiview Spectral Optical Flow Imaging for 3D operating-deflection-shape identification, Mech. Syst. Signal Pr. 152 107456.

[14] J. Javh, J. Slavič, M. Boltežar, Measuring full-field displacement spectral components using photographs taken with a DSLR camera via an analogue fourier integral, Mech. Syst. Signal Process. 100 (2018) 17–27

[15] B.K. Horn, B.G. Schunck, Determining optical flow, Artif. Intell. 17 (1) (1981) 185–203, https://doi.org/10.1016/0004-3702(81)90024-2

[16] B.D. Lucas, T. Kanade, An iterative image registration technique with an application to stereo vision, International Joint Conference on Artificial Intelligence (1981)

[17] A.V. Oppenheim, R.W. Schafer, Discrete-Time Signal Processing, third ed., Prentice Hall Press, USA, 2009.

[18] L. Cremer, M. Heckl, Structural Vibrations and Sound Radiation At Audio Frequencies, Springer-Verlag, Berlin, 1988.

[19] Y. Sun, T. Yang, Y. Chen Sound radiation modes of cylindrical surfaces and their application to vibro-acoustics analysis of cylindrical shells Journal of Sound and Vibration 424 (2018) 64-77

Operational modal analysis of a footbrige using open source Python library pyOMA

Uroš Bohinc *

Slovenian National Building and Civil Engineering institute

Abstract

To validate the numerical model of a pedestrian and cycling bridge an experimental campaign was performed. The acquired data was processed according to the principles of operational modal analysis (OMA) using open source python package pyOMA.

Keywords: Operational modal analysis, Bridge structure, Open Source, Structural Dynamics, Python

1 Introduction

This paper focuses on an experimental study of a contemporary steel footbridge structure over a highway in Slovenia. The primary aim of this study was to identify the modal parameters of the structure. However, traditional experimental modal analysis techniques that use modal hammers or shakers were not feasible. Instead, operational modal analysis (OMA) was employed, which relies on ambient vibration sources such as wind and traffic [1]. OMA can identify the dynamics of large civil engineering structures by measuring their dynamic response under operational conditions. To collect the necessary data, a comprehensive measurement campaign was conducted using various configurations of one- and triaxial accelerometers placed throughout the steel structure. After the measurements were taken, Python-based scripts were used for additional processing, and the open-source Python module, pyOMA [2], was used to perform the operational modal analysis.

2 Bridge

From a structural perspective the bridge can be categorized as a through arch bridge. The two arches possess a span of 50.50m and are constructed using circular hollow cross-sections. The transversal beams are used to connect the two arches, while the combination of diagonal and transversal beams ensures stability and wind bracing. The ends of the arches are assumed to be entirely fixed to their foundations. To suspend the deck from the arches, hangers have been employed. The bridge deck comprises two longitudinal girders with a span of 59.10 m. At both ends of the longitudinal girders, rocker bearings are used to facilitate horizontal movement in the longitudinal direction of the bridge.

*Corresponding author, Email address: uros.bohinc@zag.si

The transversal girders are positioned between the two longitudinal girders. The composite steel-concrete slab is constructed using trapezoidal steel sheeting and a reinforced concrete slab with shear connectors. All steel components are constructed using structural steel S235, except for the arches and hangers, which are made of steel S275 and S355, respectively. The concrete used is classified as C25/30.

Fig. 1: Lateral and side view of the pedestrian and cycling bridge with data acquisition station during experimental campaign.

3 Experimental campaign

The fundamental principle behind operational modal analysis is to obtain synchronized acceleration measurements from multiple locations on the structure. Due to limitations in sensor availability and connectivity, it is typically not feasible to instrument all measurement points simultaneously, and thus, several test runs with different configurations are necessary. To facilitate result comparison across individual test runs and to perform the final synthesis of the outcomes, one or more reference measurement points that remain unchanged throughout the testing process are chosen.

3.1 Data acquisition

The measurement campaign took place over two days. The acquisition of data was performed using specialized software called Dewesoft X, with two 8-channel measurement amplifiers, Dewesoft Sirius ACC. These amplifiers were used to obtain signals from 8 uniaxial accelerometers, Dytran 3192A, and three high-sensitivity accelerometers, Dytran 3191A. A reference measuring location was established on the northern lateral girder, at the mid-span from the top, using high-sensitivity accelerometers mounted perpendicular to a custom-made steel platform, as shown in Fig. 3. Another reference point was instrumented with a high-sensitivity three-axial MEMS accelerometer and placed at the top of the arch beam directly above the first reference measuring point. The 8 uniaxial accelerometers were divided into 4 roving groups, with one accelerometer mounted in vertical and the other horizontally perpendicular to the lateral axis of the bridge. These accelerometers were mounted on the lateral girder using magnetic adapters, as shown in Fig. 3. Additionally, 12 triaxial MEMS accelerometers, Dewesoft MonoDAQ, were mounted on the arch at fixed positions, as depicted in Fig. 3. These accelerometers were connected in series using the EtherCAT protocol and synchronized with the Dewesoft Sirius measurement amplifiers. Locations of measurement nodes are shown in 2.

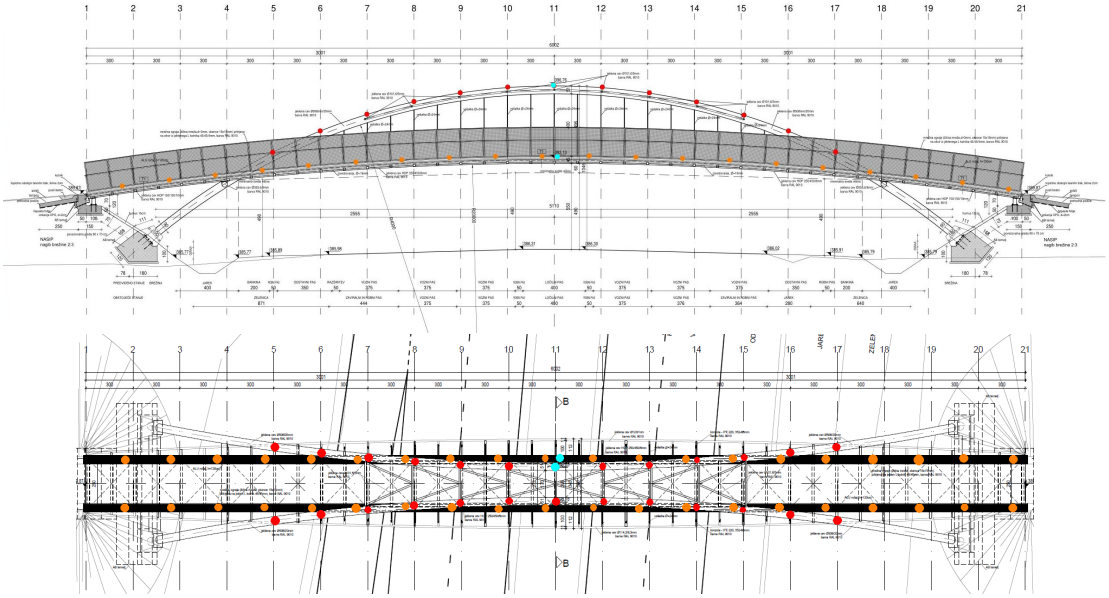


Fig. 2: Measurements nodes, lateral and top view. Orange - roving nodes, red - fixed nodes, cyan - reference nodes.

The data acquisition for each test setup lasted for 30 minutes as the computed fundamental frequency was estimated around 2 Hz. The acqusition rate was 2000 samples per second.

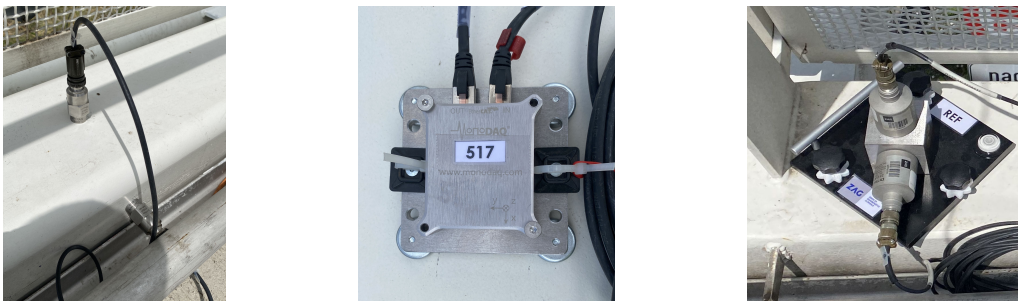


Fig. 3: Roving measurement node - two uniaxial accelerometers on a longitudinal girder (left), three axial MEMS accelerometer (center), reference measurement node with high sensitivity uniaxial accelerometers mounted on a custom made platform (right).

Before the data was further analyzed the acquired sygnals were preprocessed in order to remove the outliers, decimate, detrend and filter the data in the desired frequency range. All the preprocessing steps were implemented through python scripts using conventional python module numpy. Intermediate results were saved in the HDF5 format.

Frequency Domain Decomposition algorithm which is implemented in functions `FDDsvd` and `FDDmodEx` of Python module pyOMA was used to perform the basic modal identification. For each test run (12 in total with different configurations of roving measurement nodes), first the FRF matrix was computed followed by the SVD to help with the identification of possible modal frequencies. Modal parameters were extracted using `FDDmodEx` function.

4 Results

Inspection of SVD plot of individual test runs for the horizontal and vertical direction gives the first insight of the modal frequencies of basic vertical modal shapes of the bridge. After hand picking the frequencies the `FDDmodEx` function is called to exctract the corresponding modal shapes. Since individual test runs consisted of 4 roving measurement nodes only, complete representation of the modal shape needs to be assembled from the results of individual test runs. Assembly is possible through the reference measurement nodes which do not change during the test runs. The modal shapes presented in 4 refer only to the vertical direction. Each identified modal shape is presented only for the measurement nodes located on the deck (depicted with orange in 2) with black line corresponding to the measurement nodes on northern longitudinal girder and red line to the southern longitudinal girder. Even though the FDD algorithm has limited range of applicability the results correspond well to the results of the numerical model of the structure.

To obtain more comprehensive results, further analysis that incorporates measurement nodes located on the main steel arches of the structures will be performed. This will allow us to obtain data from different points of the structures, providing a more complete understanding of their behavior.

Further analysis will be performed using also fixed measurement nodes located on the main steel arches of the structure. Additionally, more sophisticated algorithms, such as Stochastic Sub-space Iteration, will be employed, which is also a part of the pyOMA module.

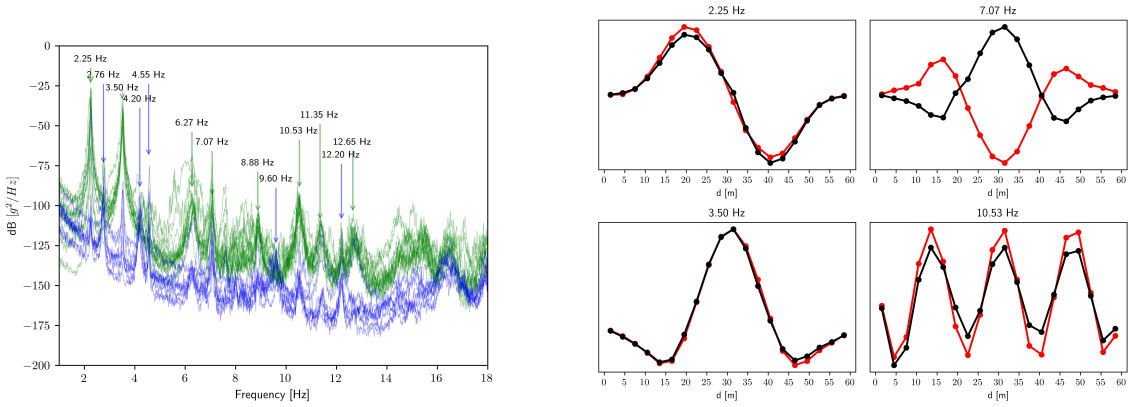


Fig. 4: Left - SVD plot of individual test runs with indicated peak frequencies. Green lines refer to the vertical direction, blue ones refer to the horizontal direction. Right - selected mode shapes taking into account only vertical accelerometers located on the deck (black - northern longitudinal girder and red - southern longitudinal girder).

References

[1] C. Rainieri and G. Fabbrocino, "Operational modal analysis of civil engineering structures an introduction and guide for applications."

[2] D. P. Pasca, A. Aloisio, M. M. Rosso, and S. Sotiropoulos, "PyOMA and PyOMA_GUI: A Python module and software for Operational Modal Analysis," *SoftwareX*, vol. 20, 12 2022.

Investigation of nonlinear dynamics using open-source tools

Elvio Bonisoli Luca Dimauro Simone Venturini* Salvatore Paolo Cavallaro

Politecnico di Torino, Department of Mechanical and Aerospace Engineering
Corso Duca degli Abruzzi, 24 – 10129 Torino, Italy

Abstract

This paper investigates the behaviour of a Euler-Bernoulli cantilever beam characterised by nonlinearities, highlighting how the use of open-source programming languages is suitable for studying system dynamics. The investigation of nonlinear dynamics in engineering systems is an important research area that has attracted significant attention in recent years. The Euler-Bernoulli beam is a widely used model in this field due to its simplicity and effectiveness in describing the behaviour of slender structures subjected to dynamic loads. An experimental campaign has been carried out to analyse the corresponding non-smooth nonlinearity due to a non-holonomic contact. Displacements measurements in time domain are performed by mean of laser sensors that have the advantage to acquire data about the system dynamics without affecting the structural properties of the beam. Time-frequency analysis has emerged as a powerful tool for studying nonlinear systems, allowing for the identification of time-varying features. In this study, we employ open-source time-frequency analysis tools to investigate the nonlinear dynamics of the beam. Therefore, two linearised models have been developed and tuned to describe the two configurations related to the nonlinear dynamics of the beam, since this nonlinear phenomenon can be considered as the superposition of the linearised behaviour corresponding to simplified models. The open-source nature of the tools used in this study also makes them readily accessible to researchers and engineers in several fields, facilitating further investigations and advancements in nonlinear dynamics research.

Keywords: Euler-Bernoulli Beam, Non-holonomic Contact, Nonlinear Dynamics, Open-Source.

1 Introduction

In structural dynamics, linearisation is a common practice in approximating system behaviour. From a physical perspective, the behaviour is well approximated if deformations are small enough to not exceed the material limit of linear proportionality [1,2]. The nodal nonlinear displacement have been monitored by Keyence Laser spot sensors [3,4], which demonstrated the methodology effectiveness. Experimental tests are carried out and the frequency contents are examined to spot the nonlinear characteristics with respect to the linearised configurations of the structure. Moreover, numerical simulations with LUPOS [5] are performed to fully interpret the obtained results. Finally, experimental test frequency contents are post-processed with the use of different time–frequency analysis approaches with Python open-source programming language and Matlab: the Fast Fourier Transform (FFT) converting signals to frequency

* Corresponding author, Email address: simone.venturini@polito.it

domains as the summation of harmonics and the Continuous Wavelet Transform (CWT) [6,7] allowing the description of different occurring time scale phenomena.

2 Experimental setup

An experimental campaign is performed to characterise the nonlinear dynamics of a Euler-Bernoulli cantilever beam. One end of the aluminium cantilever beam shown in Fig. 1 (left) is clamped by two steel parts, locked by Bosch profiles, trying to reproduce an ideally rigid constraint. Two steel disks are mounted at the beam free end. Finally, a non-holonomic constraint introduces nonlinearity acting like a pin during the contact between itself and the beam at each oscillation of the latter.

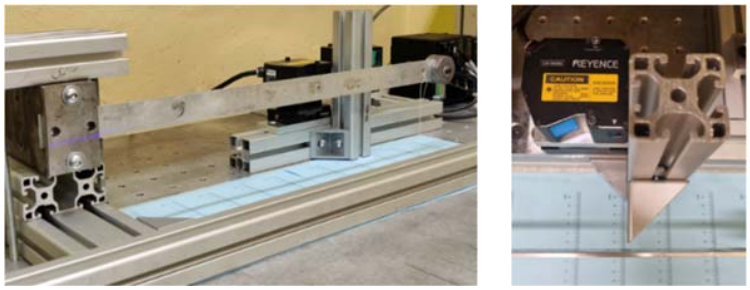


Fig. 1: Euler-Bernoulli cantilever beam (left) and non-holonomic constraint (right).

Displacements over the entire length of the beam are acquired by means of optical sensors produced by Keyence. The results used for the following analysis refer to the data acquired at the point near the non-holonomic constraint, depicted in Fig. 1 (right), to highlight the dynamics nonlinear effects. The structure is excited by non-null initial conditions reached by pulling a wire tied to the beam.

3 Time-frequency analysis

The acquired time response of the performed test on the Euler-Bernoulli cantilever beam is post-processed with the use of both FFT and CWT to investigate the frequency content of the dynamics caused by the introduction of the nonlinearity. For the following analyses both Python and Matlab software are exploited and compared to confirm the suitability of open-source programming languages for this kind of case study.

3.1 Numerical model

The described configuration of the Euler-Bernoulli beam with the non-holonomic constraint can be seen as two alternating linearised systems of a clamped – free beam and a clamped – pinned – free beam. Hence, the two models are developed and then studied with modal analysis with the software LUPOS to identify the main frequencies characterising both systems and to understand how the realised configuration behaves. The numerical model results are shown in Tab. 1.

Tab. 1: Numerical model frequencies.

Mode	Clamped – Free [Hz]	Clamped –Pinned - Free [Hz]	Description
1	2.973	13.88	1st bending
2	38.75	85.18	2nd bending

3.2 Fast Fourier Transform analysis

The time–frequency analysis using FFT consists of a series of data in frequency domain on windowed time histories with high overlapping percentages. The aim of such a type of analysis is to identify the main frequencies that characterise the system dynamics and to investigate how they change over time. The results obtained with the open-source Python code, by using the function `specgram` from `Matplotlib` library, and the Matlab analysis performed with `spectrogram` command are shown in Fig. 2.

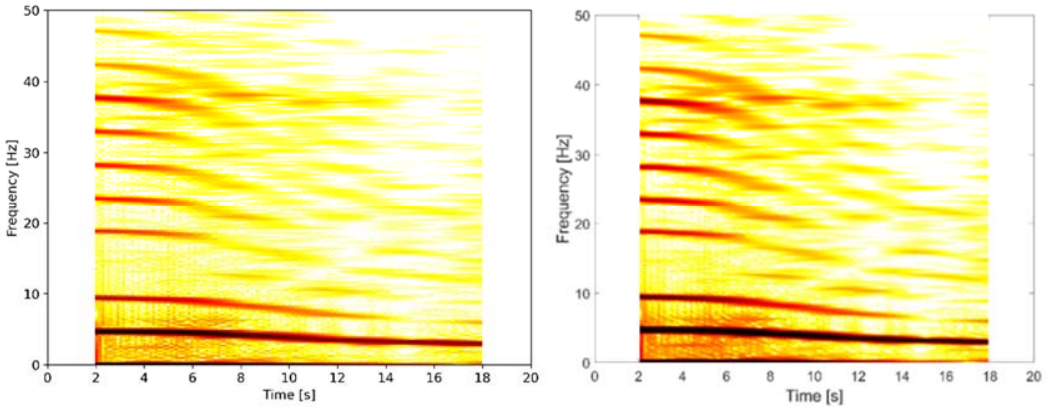


Fig. 2: Python FFT spectrogram (left) and Matlab FFT spectrogram (right).

The two spectrograms of Fig. 2 match each other showing how both programming languages let to the same results. The FFT time-frequency analysis clearly shows the superharmonics of the fundamental natural frequency listed in Tab. 2. The 1st natural frequency, that decreases from a value of ~5 Hz to ~3 Hz, is an average of the fundamental frequencies of the clamped – free and clamped – pinned – free configurations. It is weighted with respect to the contact time between the constraint and the beam (more details are given in [4]). It is possible to distinguish a higher amplitude of the harmonic at ~37 Hz that, agreeing with Tab. 1, corresponds to the 2nd resonance frequency of the clamped – free linearised system.

Tab. 2: Main frequencies.

Frequency [Hz]	Description
4.7	Mixed frequency
9.4	2nd superharmonics of the mixed frequency
18.8	4th superharmonics of the mixed frequency
37	Clamped – free 2nd mode frequency

The 3rd superhamonics at 14.1 Hz is not visible since the non-holonomic contact stops the beam that cannot move in that direction. Consequently, this non-symmetrical displacement is not describable with a polynomial expansion mainly characterised by the term of 3rd order.

3.3 Continuous Wavelet Transform analysis

An alternative approach to time–frequency analysis with FFT is the adoption of CWT analysis that allows to study fast varying phenomena, as demonstrated from the results shown in Fig. 3. This

method consists in using an analysing function called “wavelet” to obtain a set of projections of the measured displacement time response in time and frequency scale. Even if the default “Morse” wavelet used by Matlab is not developed in the Python library `Pywavelets`, exploited for the specific sketch, the results obtained with the latter by mean of the “Complex Morlet” wavelet are totally viable and comparable to the Matlab `cwt` command results. The analysis with Python is developed using the open-source code in [8].

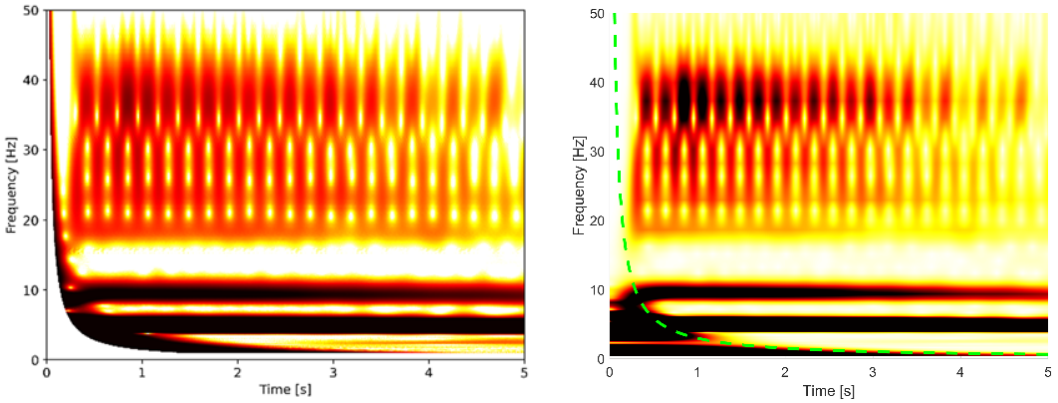


Fig. 3: Python CWT spectrogram (left) and Matlab CWT spectrogram (right).

The main harmonics and its superharmonics, already discussed in § 3.2, are still visible, but in addition to the outcomes of the FFT analysis, in the first part of the time history it is possible to notice a sort of bouncing frequency content. This is due to the switch between the excitation of the clamped – free system 2nd mode at 37 Hz thanks to the beam - constraint contact and the behaviour of the behaviour of the mixed system described in § 3.1.

References

[1] D. Wagg, S. Neild, "Nonlinear Vibration with Control: For Flexible and Adaptive Structures," 2nd ed.; Springer: Dordrecht, The Nederland, pp. 1–461, 2015.

[2] K. Worden, G.R. Tomlinson, "Nonlinearity in Structural Dynamics: Detection, Identification and Modelling", 1st ed.; Institute of Physics Publishing: Bodmin; pp. 1–659, UK, 2001, doi: 10.1201/9780429138331.

[3] E. Bonisoli, S. Venturini, S.P. Cavallaro, "Nonlinear characterisation of a rotor on passive magnetic bearings" *Int. J. Mech. Control*, vol. 23, pp. 121–128, 2022, ISSN 1590-8844.

[4] E. Bonisoli, L. Dimauro, S. Venturini, S.P. Cavallaro, “Experimental Detection of Nonlinear Dynamics Using a Laser Profilometer”, *Applied Sciences*, 2023, doi: 10.3390/app13053295

[5] E. Bonisoli, L. Dimauro, S. Venturini, “Lupos: Open-source scientific computing in structural dynamics”. In Proceedings of the 41st IMAC, 2022.

[6] J. Slavič, I. Simonovski, M. Boltežar, “Damping identification using a continuous wavelet transform: Application to real data”. *J. Sound Vib*, vol. 262, pp. 291–307, 2003 doi: 10.1016/S0022-460X(02)01032-5.

[7] J.B. Tary, R.H. Herrera, M. van der Baan, “Analysis of time-varying signals using continuous wavelet and synchrosqueezed transforms”, *Philos. Trans. R. Soc.* vol 376, 2018.

[8] PyWavelets: Wavelet Transforms in Python. Available online: http://www.neurotec.uni-bremen.de/drupal/node/46 (accessed on 8 April 2023).

Accurate finite elements for 2D and 3D structural analysis

Elvio Bonisoli Carlo Rosso Fabio Bruzzone
Simone Venturini* Salvatore Paolo Cavallaro

Politecnico di Torino, Department of Mechanical and Aerospace Engineering
Corso Duca degli Abruzzi, 24 – 10129 Torino, Italy

Abstract

In this paper, the development of a first order quadrilateral finite element with flexural and membranal behaviour and a first order hexahedral solid element are addressed. The finite element method is a popular technique used for the design and analysis of structural systems. However, literature is overcrowded with formulations which do not account for the numerical implementation aspects, which limits their applicability. Moreover, the availability of accurate and reliable finite element models is limited and expensive, especially for complex structures.
The developed first order quadrilateral finite element with flexural and membranal behaviours is focused on these limitations by incorporating these features in its open formulation. This element provides accurate results for the structural dynamics of structures, including plate and shell structures. It also avoids spurious modes, making it a versatile tool for 3D structural dynamics.
The isoparametric hexahedral element employs a selective underintregration numerical scheme to avoid shear and volumetric locking also avoiding spurious zero strain energy modes with a small computing effort by using an enhanced assumed strain field. This allows an efficient and accurate modelling of most solid structures.
The development of these elements is crucial for the open-source community as it offers a new and valuable tool for the design and analysis of complex structural systems. The availability of these elements in open-source software will also make it more accessible to researchers, allowing for the exploration of new ideas and innovation in the field of structural engineering.

Keywords: Finite element method, Structural dynamics, Kirchoff plate, Hexahedral element

1 Introduction

The Finite Element Method (FEM) is a well-known and consolidated technique in the engineering design and validation processes. The methodology, born in 1940s, got its strongest impetus with NASA open-source programs collected then in NASTRAN [1]. Beyond the preliminary simple working principles analytically exposed [2], [3], [4], [5], FEM industrial codes often present closed source modifications to obtain accurate results which are difficult to recognise without extended manual explanation. Under this scenario, the authors proposed LUPOS (Lumped Parameters Open-Source FEM code) [6] and PoliFEMo, codes developed in Matlab, and compatible with GNU Octave

* Corresponding author, Email address: simone.venturini@polito.it

[7], with the aim of providing a complete 3D FEM code suitable for analysing elements formulations, assembly approaches, reduction techniques in a complete parametric scheme.
In this paper, a first order quadrilateral finite element with flexural and membranal behaviour and a first order hexahedral solid element are developed.
The developed first order quadrilateral finite element with flexural and membranal behaviours is developed in open formulation with industrial accuracy in mind. This element provides accurate results for the structural dynamics of structures, including plate and shell structures. It also avoids spurious modes, making it a versatile tool for 3D structural dynamics such as Numerical Modal Analysis (NMA).
The isoparametric hexahedral element exploits a selective underintregration numerical scheme to avoid shear and volumetric locking also avoiding spurious zero strain energy modes with a small computing effort by using an enhanced assumed strain field. This allows an efficient and accurate modelling of most solid structures.
In section 2 the 4-node shell formulation is developed both in membrane and bending behaviour.
In section 3 the 8-node hexahedral formulation is developed.
In section 4 a case study is presented, and the results are compared with MSC.Nastran.
Finally, conclusions are expressed.

2 2D shell element

The chosen element is an isoparametric in-plane 4-node shell element, also known as "CQUAD4", whose global reference frame (X,Y,Z) and the local one (ξ,η) in which the *i*-th node coordinates assume the values of ± 1 are visible in Fig. 1.

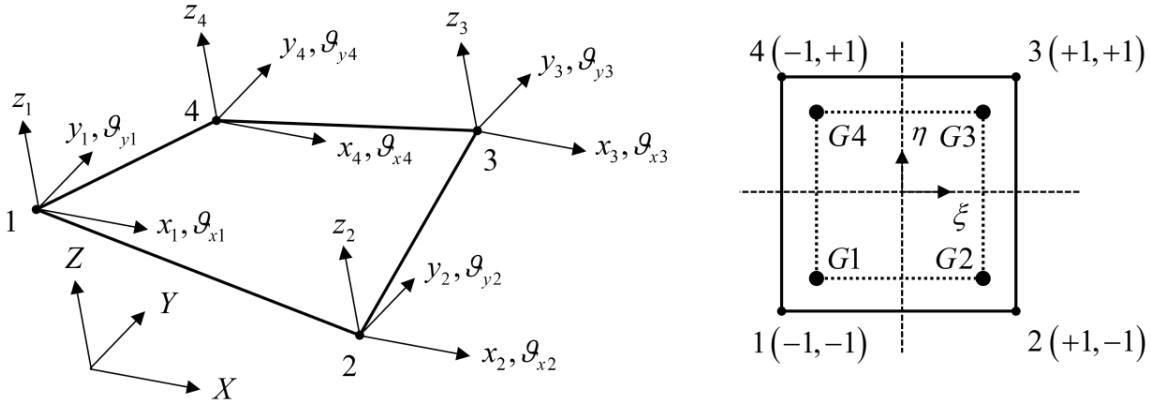


Fig. 1: Global (X,Y,Z) and local (ξ,η) reference frames.

The developed shell element has 5 degrees of freedom (DOFs) per each *i*-th node: $x_i, y_i, \mathrm{z}_i, \vartheta_{xi}, \vartheta_{yi}$.
The isoparametric element is a square shell in the $\xi\eta$ plane centred in $O\,(0,0)$ and with side length equal to 2. Gauss integration points $G1, \ldots, G4$, that have unit weight and coordinates $\pm\sqrt{3}/3$, are in a 2 × 2 scheme for full integration. The element is, by simplification, homogeneous isotropic with constant thickness t in all directions.

2.1 Membranal behaviour

As described in [1], the equation of the bilinear co-ordinate transformation is used for transforming the arbitrarily quadrilateral element from the (x,y) coordinate system into a square element in the (ξ,η) coordinate system of Fig. 1. The membranal displacement field $\mathbf{u}_m(\xi,\eta)=\{x,y\}^T$ in (2.1) is described by bilinear shape functions related to nodal displacement (2.2) and incompatible mode nodeless displacements (2.3) introduced by Wilson et al. [8] and Taylor et al. [9].

$$\mathbf{u}_m=\sum_{i=1}^{4}N_{i\,m}(\xi,\eta)\,\mathbf{u}_{i\,m}+N_{5\,m}(\xi,\eta)\,\bar{\mathbf{u}}_{5\,m}+N_{6\,m}(\xi,\eta)\,\bar{\mathbf{u}}_{6\,m} \tag{2.1}$$

$$N_{i\,m}(\xi,\eta)=\frac{(1+\xi_i\,\xi)(1+\eta_i\,\eta)}{4} \tag{2.2}$$

$$N_{5\,m}(\xi,\eta)=\frac{1-\xi^2}{2}\,,\;N_{6\,m}(\xi,\eta)=\frac{1-\eta^2}{2} \tag{2.3}$$

The shape functions in (2.2) are stored in $\mathbf{N}_m$ matrix.

$$\mathbf{N}_m=\begin{bmatrix}N_{1\,m} & 0 & N_{2\,m} & 0 & N_{3\,m} & 0 & N_{4\,m} & 0\\ 0 & N_{1\,m} & 0 & N_{2\,m} & 0 & N_{3\,m} & 0 & N_{4\,m}\end{bmatrix}_{(2\times 8)} \tag{2.4}$$

Then, it is possible to calculate the Jacobian matrix $\mathbf{J}$ by using derivatives of (2.1).

$$\mathbf{J}=\begin{bmatrix}\dfrac{\partial x}{\partial \xi} & \dfrac{\partial y}{\partial \xi}\\ \dfrac{\partial x}{\partial \eta} & \dfrac{\partial y}{\partial \eta}\end{bmatrix}_{(2\times 2)} \tag{2.5}$$

The dimensional co-ordinate partial derivatives of shape functions are calculated by chain rule and collected in $\nabla\mathbf{N}'$ matrix. The procedure requires the calculation of non-dimensional co-ordinate partial derivatives of shape functions $\nabla\mathbf{N}$ matrix and the inversion of $\mathbf{J}$.

$$\nabla\mathbf{N}'=\mathbf{J}^{-1}\,\nabla\mathbf{N}=\mathbf{J}^{-1}\begin{bmatrix}\dfrac{\partial N_{1\,m}}{\partial \xi} & \cdots & \dfrac{\partial N_{4\,m}}{\partial \xi}\\ \dfrac{\partial N_{1\,m}}{\partial \eta} & \cdots & \dfrac{\partial N_{4\,m}}{\partial \eta}\end{bmatrix}_{(2\times 4)}=\begin{bmatrix}\dfrac{\partial N_{1\,m}}{\partial x} & \cdots & \dfrac{\partial N_{4\,m}}{\partial x}\\ \dfrac{\partial N_{1\,m}}{\partial y} & \cdots & \dfrac{\partial N_{4\,m}}{\partial y}\end{bmatrix}_{(2\times 4)} \tag{2.6}$$

Hence, $\mathbf{B}_{i\,m}$ strain-displacement matrix is obtained for each shape function (2.2) from the terms of matrix of (2.6) as described by [2]. Similarly, auxiliary strain-displacement matrices are composed for Wilson incompatible modes in (2.7). It is important to notice that $\mathbf{B}_{i\,m}$ matrix is composed by reorganising $\nabla\mathbf{N}'$ terms in which is embedded the inverse of $\mathbf{J}$, while $\mathbf{B}_{5\,m}$ and $\mathbf{B}_{6\,m}$ are developed by highlighting the dependence of open form inverse of $\mathbf{J}$. The full procedure is described in [8], [9].

The results are collected in a $\bar{\mathbf{B}}_m$ matrix which defines strain relationship even for nodeless DOFs (2.8) allowing to cure shear and dilatation locking.

$$\mathbf{B}_{i\,m}=\begin{bmatrix}\dfrac{\partial N_{i\,m}}{\partial x} & 0\\ 0 & \dfrac{\partial N_{i\,m}}{\partial y}\\ \dfrac{\partial N_{i\,m}}{\partial y} & \dfrac{\partial N_{i\,m}}{\partial x}\end{bmatrix}_{(3\times 2)},\ \mathbf{B}_{5\,m}=\frac{1}{|\mathbf{J}|}\begin{bmatrix}-\xi\dfrac{\partial y}{\partial \eta} & 0\\ 0 & \xi\dfrac{\partial x}{\partial \eta}\\ \xi\dfrac{\partial x}{\partial \eta} & -\xi\dfrac{\partial y}{\partial \eta}\end{bmatrix},\ \mathbf{B}_{6\,m}=\frac{1}{|\mathbf{J}|}\begin{bmatrix}\eta\dfrac{\partial x}{\partial \xi} & 0\\ 0 & -\eta\dfrac{\partial x}{\partial \xi}\\ -\eta\dfrac{\partial x}{\partial \xi} & \eta\dfrac{\partial x}{\partial \xi}\end{bmatrix} \tag{2.7}$$

$$\bar{\mathbf{B}}_m=\begin{bmatrix}\mathbf{B}_{1\,m} & \cdots & \mathbf{B}_{4\,m} & \mathbf{B}_{5\,m} & \mathbf{B}_{6\,m}\end{bmatrix}_{(3\times 12)} \tag{2.8}$$

Therefore, the expanded stiffness matrix $\bar{\mathbf{K}}_m$ is defined as follows in (2.9)

$$\bar{\mathbf{K}}_m=\int_{-1}^{1}\int_{-1}^{1}|\mathbf{J}|\bar{\mathbf{B}}_m^T\,\mathbf{D}_m\,\bar{\mathbf{B}}_m\,d\xi d\eta \tag{2.9}$$

in which the material elasticity matrix $\mathbf{D}_m$ (3 × 3) is defined for isotropic material subject to an in-plane tensile status, i.e., $\sigma_z=0$ [3], [4]. The membrane stiffness matrix building procedure is completed by using static condensation [10]

$$\mathbf{K}_m=\bar{\mathbf{K}}_{m,MM}-\bar{\mathbf{K}}_{m,SM}^T\,\bar{\mathbf{K}}_{m,SS}^{-1}\,\bar{\mathbf{K}}_{m,SM} \tag{2.10}$$

where M identifies master DOFs of node generalised displacements, while S identifies slave DOFs of nodeless displacements. Similarly to (2.9), mass matrix $\mathbf{M}_m$ is defined as follows:

$$\mathbf{M}_m=\rho\int_{-1}^{1}\int_{-1}^{1}|\mathbf{J}|\mathbf{N}_m^T\,\mathbf{N}_m\,d\xi d\eta \tag{2.11}$$

The integration is performed by approximated Gauss integration with a 2 × 2 scheme shown in Fig. 1.

2.2 Bending behaviour

Assuming the rectangular in-plane element of Fig. 1, non-dimensional coordinates are defined with respect to its mapping. Therefore, the displacement field $\mathbf{u}_b(\xi,\eta)=\{z,\vartheta_x,\vartheta_y\}^T$ is defined with a partial polynomial of quartic degree. The polynomial terms are defined by Kirchoff formulation [5]:

$$\begin{aligned}z(\xi,\eta)&=a_1+a_2\xi+a_3\eta+a_4\xi\eta+a_5\xi^2+a_6\eta^2+\\&\quad+a_7\xi^2\eta+a_8\xi\eta^2+a_9\xi^3+a_{10}\eta^3+a_{11}\xi^3\eta+a_{12}\xi\eta^3\\ \vartheta_x(\xi,\eta)&=\frac{\partial z}{\partial y}=\frac{1}{2}\frac{\partial z}{\partial \eta}\\ \vartheta_y(\xi,\eta)&=-\frac{\partial z}{\partial x}=-\frac{1}{2}\frac{\partial z}{\partial \xi}\end{aligned} \tag{2.12}$$

where $a_1,...,a_{12}$ are 12 unknown coefficients. The coefficients are calculated as function of the element generalised nodal displacement vector $\mathbf{q}$ only i.e., an ordered collection of generalised displacements in (2.12). Therefore, the functional matrix $\mathbf{\Phi}(\xi,\eta)$ (3 × 12) is defined by collecting the non-dimensional coordinate operators of each generalised displacement field [3]. Starting from

(2.12), it is possible to express the correspondence matrix $\mathbf{A}$ (12 × 12) with respect to non-dimensional co-ordinates and to obtain shape function matrix $\mathbf{N}(\xi,\eta)$ which determines the displacement field $\mathbf{u}_b(\xi,\eta)$ for a given $\mathbf{q}$ set of nodal generalised displacements.

$$\mathbf{u}_b(\xi,\eta) = \mathbf{\Phi}(\xi,\eta)\,\mathbf{A}^{-1}\mathbf{q} = \mathbf{N}(\xi,\eta)\,\mathbf{q} \tag{2.13}$$

Therefore, the shape functions defined in (2.13) are polynomials of 4 variables that are functions of non-dimensional co-ordinates ξ and η. The methodology describing the bending properties of the shell element involves a way to express the shape functions that makes hard to evaluate $\mathbf{M}_b$ and $\mathbf{K}_b$ matrices in open form, due to the lack of a proper Jacobian matrix definition and application of chain rule. Here, Ming procedure is adopted [11] to build bending strain-displacement matrix $\mathbf{B}_{ib}$.

$$\mathbf{B}_{ib} = -\begin{bmatrix} \dfrac{\partial^2 N_{ib}}{\partial x^2} & \dfrac{\partial^2 N_{ixb}}{\partial x^2} & \dfrac{\partial^2 N_{iyb}}{\partial x^2} \\ \dfrac{\partial^2 N_{ib}}{\partial y^2} & \dfrac{\partial^2 N_{ixb}}{\partial y^2} & \dfrac{\partial^2 N_{iyb}}{\partial y^2} \\ 2\dfrac{\partial^2 N_{ib}}{\partial x \partial y} & 2\dfrac{\partial^2 N_{ixb}}{\partial x\,\partial y} & 2\dfrac{\partial^2 N_{iyb}}{\partial x\,\partial y} \end{bmatrix}_{(3\times 3)} \tag{2.14}$$

The results are collected in a $\mathbf{B}_b$ matrix which defines strain relationship for bending behaviour.

$$\mathbf{B}_b = \begin{bmatrix} \mathbf{B}_{1b} & \cdots & \mathbf{B}_{4b} \end{bmatrix}_{(3\times 12)} \tag{2.15}$$

Therefore, the bending stiffness matrix $\mathbf{K}_b$ is defined as follows:

$$\mathbf{K}_b = \int_{-1}^{1}\int_{-1}^{1} |\mathbf{J}|\,\mathbf{B}_b^T\,\mathbf{D}_b\,\mathbf{B}_b\,d\xi d\eta \tag{2.16}$$

Where material properties are contained in elasticity matrix $\mathbf{D}_b$ (3 × 3), expressed as [5], [11] for in-plane tensile status. Instead, mass matrix $\mathbf{M}_m$ is defined as follows by using bilinear membrane shape functions for translational generalised displacements only.

$$\mathbf{M}_b = \rho\int_{-1}^{1}\int_{-1}^{1} |\mathbf{J}|\,\mathbf{N}_b^T\,\mathbf{N}_b\,d\xi d\eta \tag{2.17}$$

$$\mathbf{N}_b = \begin{bmatrix} N_{1m} & 0 & 0 & N_{2m} & 0 & 0 & N_{3m} & 0 & 0 & N_{4m} & 0 & 0 \end{bmatrix}_{(1\times 12)} \tag{2.18}$$

As for membrane behaviour, Gauss 2 × 2 integration scheme is adopted [1].

3 3D hexahedral element

The element chosen is an isoparametric 8-node brick element, also known as “HEXA8”, whose global reference frame (X,Y,Z) and the local one (ξ,η,ζ) in which the *i*-th node coordinates assume the values of ± 1 are visible in Fig. 2 [1], [12].

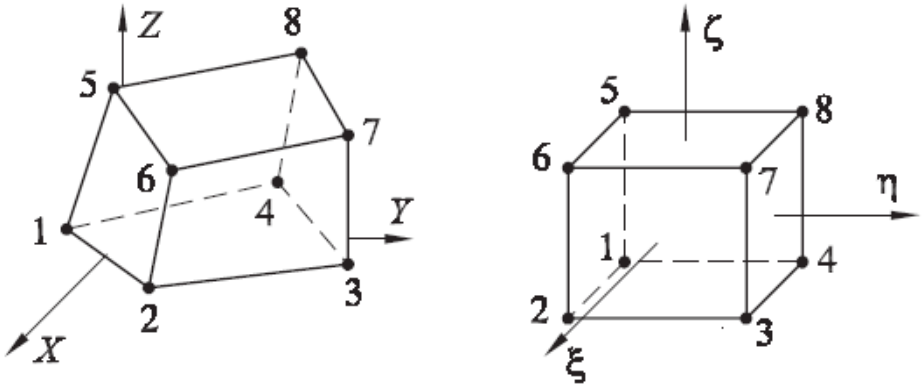


Fig. 2: Global (X,Y,Z) and local (ξ,η,ζ) reference frames [1].

For a generic element, the displacement field in the local reference frame $\mathbf{u}(\xi,\eta,\zeta)$ is obtained from the *i*-th node displacement field in the local reference frame $\mathbf{u}_i(\xi,\eta,\zeta)=\{\xi_i,\eta_i,\zeta_i\}^T$ as:

$$\mathbf{u}=\sum_{i=1}^{8} N_i(\xi,\eta,\zeta)\,\mathbf{u}_i \tag{3.1}$$

where the *i*-th shape function $N_i(\xi,\eta,\zeta)$ is defined as:

$$N_i(\xi,\eta,\zeta)=\frac{1}{8}(1+\xi_i\,\xi)(1+\eta_i\,\eta)(1+\zeta_i\,\zeta) \tag{3.2}$$

The stiffness matrix $\mathbf{K}_e$ of each element can be obtained by integrating over its volume the energy of deformation as:

$$\mathbf{K}_e=\int_V \mathbf{B}^T\,\mathbf{D}\,\mathbf{B}\,dV \tag{3.3}$$

where $\mathbf{B}$ is the strain-displacement relationship matrix composed by the derivatives of the shape functions in space and $\mathbf{D}$ is the matrix defining the elastic properties of the material while V is the volume of the element. In the natural coordinates frame the infinitesimal volume for integration can hence be expressed as $dV=|\mathbf{J}|\,d\xi d\eta d\zeta$ and therefore the integral becomes:

$$\mathbf{K}_e=\int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1}|\mathbf{J}|\mathbf{B}^T\mathbf{D}\,\mathbf{B}\,d\xi d\eta d\zeta \tag{3.4}$$

where $\mathbf{J}$ is the Jacobian of the coordinate transformation:

$$\mathbf{J} = \begin{bmatrix} \dfrac{\partial x}{\partial \xi} & \dfrac{\partial y}{\partial \xi} & \dfrac{\partial z}{\partial \xi} \\ \dfrac{\partial x}{\partial \eta} & \dfrac{\partial y}{\partial \eta} & \dfrac{\partial z}{\partial \eta} \\ \dfrac{\partial x}{\partial \zeta} & \dfrac{\partial y}{\partial \zeta} & \dfrac{\partial z}{\partial \zeta} \end{bmatrix}_{(3\times 3)} \tag{3.5}$$

To evaluate the integral in (3.4), a numerical integration scheme is employed and a 2 × 2 × 2 Gauss scheme is used in which each integration point has unit weight and coordinates $\pm\sqrt{3}/3$ [13]. Without modifications, this formulation suffers from shear and volumetric locking. To overcome this problem selective underintegration is often employed. The material tensor $\mathbf{D}$ is separated in the normal $\mathbf{D}_{11}$ and shear $\mathbf{D}_{22}$ strain components as well as the strain-displacement matrix $\mathbf{B}$ in matrices $\mathbf{B}_1$ and $\mathbf{B}_2$. Then the integral for the stiffness matrix is also split as:

$$\mathbf{K}_e = \int_V \mathbf{B}^T \mathbf{D}\, \mathbf{B}\, dV = \int_V \mathbf{B}_1^T \mathbf{D}_{11}\, \mathbf{B}_1\, dV + \int_V \mathbf{B}_2^T \mathbf{D}_{22}\, \mathbf{B}_2\, dV \tag{3.6}$$

The stiffness portion due to the normal strains is integrated using the same 2 × 2 × 2 Gauss scheme presented earlier, while the shear terms are integrated using only one point in the centre of the element with a weight of 8. The obtained matrix is then less stiff and does not suffer from shear or volumetric locking. However, another problem is introduced since spurious zero strain energy modes arise. Those are elastic deformation modes in which the deformation strain energy is zero and hence could be assimilated to rigid body modes, but those are not rigid and hence are unphysical and must be eliminated to obtain the correct results. For the 8-node brick there are 3 of these modes which are visible in Fig. 3.

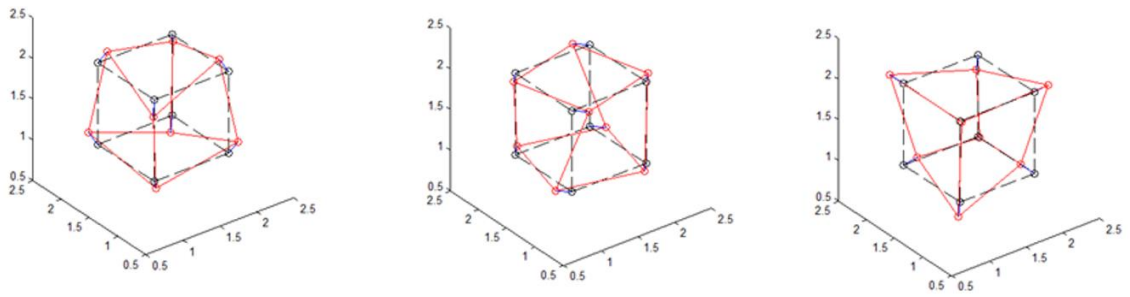


Fig. 3: Spurious zero strain energy modes for the 8-node brick.

To avoid these spurious modes several techniques have been studied in literature and most of them are computationally intensive. The solution implemented in the current model is the fastest one and is based on the work of MacNeal [1] in which instead of using only the central node for underintegration one virtual node is used for each face. The values are obtained averaging the values obtained in the neighbouring Gauss points, weighted by their Jacobian for the elements to pass the patch test. Even with these precautions the formulation still suffers from volumetric locking. A solution can be obtained by introducing an Enhanced Assumed Strain (EAS) field which extends the $\mathbf{B}_1$ matrix as $\mathbf{B}_1' = \begin{bmatrix} \mathbf{B}_1 & \mathbf{B}_\alpha \end{bmatrix}$ where the additional EAS field is defined as

$$\mathbf{B}_{\alpha} = \begin{bmatrix} \xi & 0 & 0 & \xi\eta & 0 & \xi\zeta \\ 0 & \eta & 0 & \xi\eta & \eta\zeta & 0 \\ 0 & 0 & \zeta & 0 & \eta\zeta & \xi\zeta \end{bmatrix}_{(3\times 6)} \tag{3.7}$$

in which again the factor $1/|\mathbf{J}|$ is needed to satisfy the patch test [1]. The combination of this EAS approach with the selective reduced integration solves all locking problems and does not introduce spurious modes.

However, due to the introduction of the additional EAS field the modified stiffness matrix $\tilde{\mathbf{K}}_e$ has 6 more additional DOFs which need to be eliminated since they do not correspond to any physical DOF. Including the selectively underintegrated matrix $\bar{\mathbf{B}}'_2$ and the EAS, matrix $\bar{\mathbf{B}}$ is partitioned as follows

$$\bar{\mathbf{B}} = \begin{bmatrix} \bar{\mathbf{B}}_1 & \bar{\mathbf{B}}_{\alpha} \\ \bar{\mathbf{B}}'_2 & \mathbf{0} \end{bmatrix} \tag{3.8}$$

Therefore, $\tilde{\mathbf{K}}_e$ is partitioned as

$$\tilde{\mathbf{K}}_e = \begin{bmatrix} \mathbf{K}_{iso,iso} & \mathbf{K}_{iso,\alpha} \\ \mathbf{K}_{\alpha,iso} & \mathbf{K}_{\alpha,\alpha} \end{bmatrix} \tag{3.9}$$

where $\mathbf{K}_{iso,iso}$ is the isoparametric part of the matrix, $\mathbf{K}_{\alpha,\alpha}$ the part due to EAS field and $\mathbf{K}_{iso,\alpha} = \mathbf{K}^T_{\alpha,iso}$ the connecting portions. The matrix $\mathbf{K}_e$ is obtained by static condensation [10] as

$$\mathbf{K}_e = \mathbf{K}_{iso,iso} - \mathbf{K}^T_{\alpha,iso}\, \mathbf{K}^{-1}_{\alpha,\alpha}\, \mathbf{K}_{\alpha,iso} \tag{3.10}$$

For a dynamic analysis also, the inertial properties are needed, which are expressed by the mass matrix

$$\mathbf{M}_e = \rho \int_V \mathbf{N}^T\, \mathbf{N}\, dV \tag{3.11}$$

where $\mathbf{N}$ is a matrix containing all shape functions

$$\mathbf{N} = \begin{bmatrix} N_1 \cdots N_i \cdots N_8 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & N_1 \cdots N_i \cdots N_8 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & N_1 \cdots N_i \cdots N_8 \end{bmatrix}_{(3\times 24)} \tag{3.12}$$

The volume integral can be again expressed as

$$\mathbf{M}_e = \rho \int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} |\mathbf{J}|\, \mathbf{N}^T\mathbf{N}\, d\xi d\eta d\zeta \tag{3.13}$$

Which is again numerically evaluated on a 2 × 2 × 2 Gauss grid.

4 Case study

A case study is developed to validate the accuracy of the elements in numerical modal analysis. An aluminium (Young modulus of elasticity 71 GPa, density 2700 kg/m^3, Poisson ratio 0.3) hollow beam of 600 mm length, width 60 mm, height 30 mm and 3 mm thickness is meshed with the developed 2D and 3D elements with the same amount of longitudinal sections and compared in natural

frequencies in free-free conditions. Fig. 4 shows the overall mesh structure for the two models, while Fig. 5 shows the details of the two cross sections, having different number of nodes, hence total DOFs. Indeed, hexahedral model has 72 DOFs per section, while shell model only 40 DOFs per section.

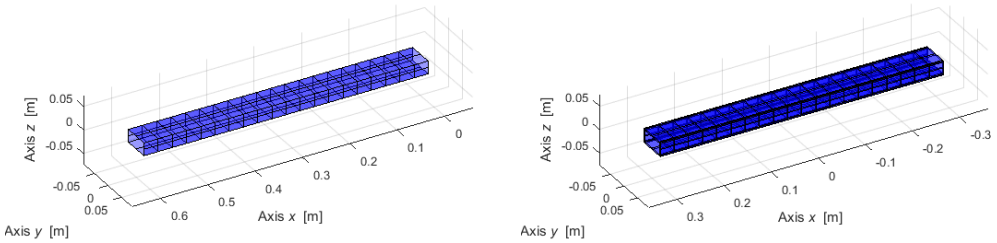


Fig. 4: Shell and brick hollow beam mesh.

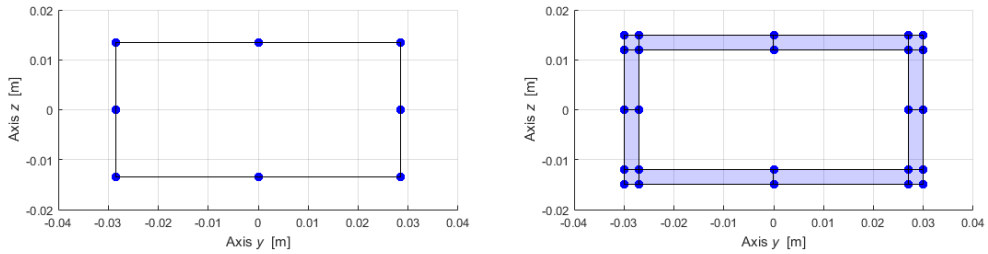


Fig. 5: Shell and brick model cross sections.

In Tab. 1 the comparison between MSC.Nastran and LUPOS formulations is performed. The hexahedron model is stiffer than the shell formulation as can be seen from the slight difference mainly in torsional behaviour. The difference is also present in LUPOS formulations, which are almost identical to MSC.Nastran.

Tab. 1: Numerical modal analysis comparison.

Mode	**MSC.Nastran [Hz]**		**LUPOS [Hz]**		**Description**
	CQUAD4	**HEXA**	**Sh4**	**Hex**	
7	595.49	593.22	595.2	593.2	1st xz bending
8	1027.3	1018.6	1027	1019	1st xy bending
9	1564.2	1543.9	1555	1544	2nd xz bending
10	1792.7	1900.0	1794	1900	1st x torsion
11	2438.2	2773.1	2424	2773	2nd x torsion
12	2674.2	2589.3	2674	2589	2nd xy bending
13	2816.7	2786.6	2730	2787	3rd xz bending
14	2848.7	3159.1	2817	3159	3rd x torsion
15	2903.7	3492.7	2928	3493	1st z shear
16	3294.5	3797.6	3304	3798	1st y shear
17	3718.4	3815.6	3691	3816	4th x torsion

5 Conclusions

The developed elements are significantly similar in accuracy to industrial standard FEM code as MSC.Nastran. The 2D and 3D proposed elements are described in the critical passages of mass and stiffness matrices construction procedure, helping the open-source community obtaining compliant results to efficient industrial codes. The resumed methodology offers a valuable tool for the design and analysis of complex structural systems.
Currently, these elements are present in the open source projects LUPOS and PoliFEMo. The current implementation is fully compatible with GNU Octave programming language. GNU Octave and Python are both powerful open-source programming languages that are used for scientific computing and data analysis. While Python has gained significant popularity in recent years, GNU Octave remains a highly effective tool for numerical computations, especially in the field of engineering. One of the key advantages of GNU Octave is its built-in support for matrix operations and linear algebra, which are often essential in engineering applications. In future, translation to open-source language such Python will also make it more accessible to researchers.

Acknowledgments

This project is developed under the National Recovery and Resilience Plan (NRRP), Mission 4 Component 2 Investment 1.3 - Call for tender No. 1561 of 11.10.2022 of Ministero dell'Università e della Ricerca (MUR); funded by the European Union – NextGenerationEU, project code PE0000021 "Network 4 Energy Sustainable Transition – NEST".

References

[1] R.H. MacNeal, *Finite Elements: Their Design and Performance*. New York: Marcel Dekker, 1993.
[2] T.J.R. Hughes, *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis*. Englewood Cliffs, New Jersey Prentice-Hall, 1987, ISBN: 9780133170252.
[3] G. Belingardi, *Il metodo degli elementi finite nella progettazione meccanica*. Levrotto & Bella, Torino, 1995.
[4] A. Gugliotta, *Elementi finiti.* Torino: Otto Editore, 2002.
[5] G. Genta, *Vibration dynamics and control*. Springer, New York, 2009, ISBN: 9780387795799.
[6] E. Bonisoli, L. Dimauro, S. Venturini, "Lupos: Open-source scientific computing in structural dynamics," In *Proceedings of the41st IMAC, A Conference and Exposition on Structural Dynamics,* Austin, TX, USA, 13–16 February 2023.
[7] J.W. Eaton, "GNU Octave" https://octave.org/.
[8] E.L. Wilson, R.L. Taylor, W.P. Doherty, J. Ghaboussi, "Incompatible displacement models," *Numerical and Computer Methods in Structural Mechanics*, 1973, pp. 43-57, doi: 10.1016/B978-0-12-253250-4.50008-7.
[9] R.L. Taylor, P.J. Beresford, E.L. Wilson, "A non-conforming element for stress analysis," *International Journal for Numerical Methods in Engineering*, vol. 10, no.6, pp. 1211-1219, 1976, doi: 10.1002/nme.1620100602.
[10] R.J. Guyan, "Reduction of stiffness and mass matrices," *AIAA Journal*, vol. 3, n. 2, pp. 3800, 1965, doi: 10.2514/3.2874.

[11] P.G. Ming, L.S. Fa, "A new element used in the non-orthogonal boundary plate bending theory—an arbitrarily quadrilateral element," *International Journal for Numerical Methods in Engineering*, vol. 24, no. 6, pp. 1031-1042, 1987, doi: 10.1002/nme.1620240602.

[12] R.D. Cook, *Concepts and applications of Finite Element Analysis*. New York: Jhon Wiley & Sons, 2001.

[13] A.R. de Sousa, J.R. Natal, V.R. Fontes, C.J. de Sà, "A new volumetric and shear locking-free 3D enhanced strain element," *Engineering Computations*, vol. 20, n. 7, pp. 896-925, 2003.

FLife - Vibration fatigue by spectral methods

Martin Česnik Aleš Zorman Janko Slavič * Miha Boltežar

University of Ljubljana, Faculty of Mechanical Engineering

Abstract

FLife is an open-source Python package for efficient and reliable estimation of structure's service life for known vibration load. Primarily, the package is designed to apply existing spectral methods to known stress-load power spectral density (PSD) in the fatigue zone. With given set of material fatigue parameters in Basquin's equation, FLife provides an accurate estimation of vibration fatigue life. Load profiles can be either in form of Numpy array for closed-form analysis or by using a simple and intuitive GUI. The evaluation of all 20 spectral methods, available in FLife, have been thoroughly evaluated in review paper [1]. Secondly, FLife also handles load profiles, defined in terms of time series, where it relies on the FatPack package.

To process data, FLife uses the open-source Numpy and Scipy libraries, ensuring reliability and repeatability through well-tested and optimized source code for numerical processing. The code is structured using a clear and readable functional programming approach to allow easy modification and extension. FLife includes publicly available examples of usage. The source code is openly available through FLife's GitHub repository with a permissive MIT license and is open to the community issue reporting as well as feature and pull-requests.

Keywords: Vibration fatigue, spectral methods, load distribution, frequency cycle-counting

1 Statement of Need

The FLife package development began from a need to standardize as well as simplify the vibration fatigue calculation and to establish an exact and transparent benchmark tool for existing and future spectral methods. Since the FLife code has been thoroughly investigated for its exactness it offers an analyst a reliable and accurate fatigue life estimation. The open-source development model also has the potential of improving the quality and reliability of the developed solutions, either through methodical code testing and documentation or through global community input.

2 Supported spectral methods

For general theoretical background on vibration fatigue (structural dynamics, uniaxial/multiaxial fatigue, non-Gaussianity, non-stationarity, etc), the reader is encouraged to see Slavič et al. [2].

*Corresponding author, Email address: janko.slavic@fs.uni-lj.si

The main functionality of FLife is provided by `FLife.SpectralData` object. Within it, a range of 20 spectral methods are implemented, which can be organized into 4 subgroups, see Fig. 1:

- Narrowband correction factor; methods are based on narrowband approximation, accounting for broadband process with correction factor,
- RFC PDF approximation; methods are based on approximation of Rainflow Probability Density Function,
- Combined fatigue damage - cycle damage combination; methods are based on splitting of PSD of broadband process into N narrowband approximations and accounting the formation of distinct categories of cycles,
- Combined fatigue damage - narrowband damage combination; methods are based on splitting of PSD of broadband process into N narrowband approximations and summing narrowband damages by suitable damage combination rule.

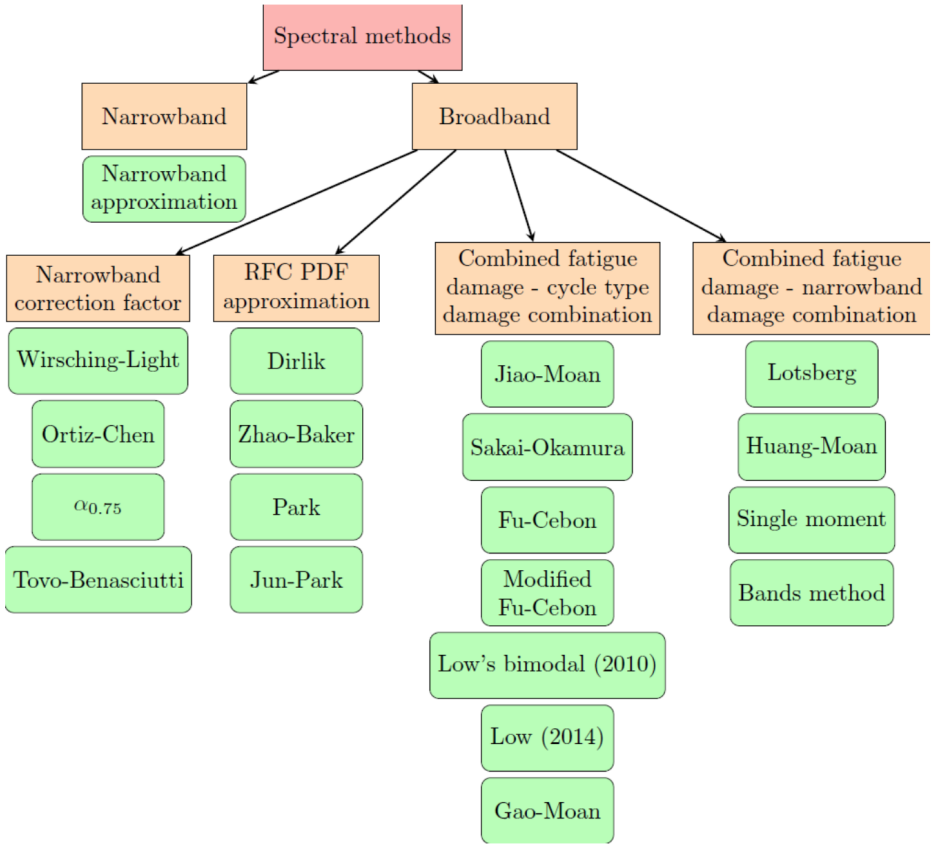


Fig. 1: Spectral methods available in `FLife`.

`SpectralData` object contains data, required for fatigue-life estimation: power spectral density (PSD), spectral moments, spectral band estimators and others parameters. `SpectralData` is instantiated with input parameter:

- input = 'GUI' - PSD is provided by user via GUI (graphically and tabulary), see Fig. 2
- input = (PSD, freq) - tuple of PSD and frequency vector is provided.
- input = (x, dt) - tuple of time history and sampling period is provided.

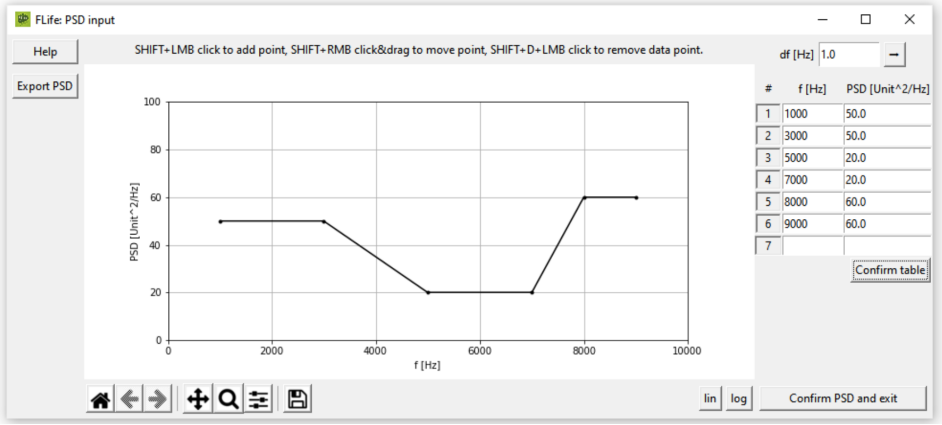


Fig. 2: PSD definition using GUI.

3 FLife usage example

Code Listing 1 shows a short example of using FLife to obtain expected vibration fatigue life with three different counting methods. More detailed examples are available in the FLife source code repository [3].

```
import FLife
import numpy as np

dt = 1e-4
x = np.random.normal(scale=100, size=10000)
C = 1.8e+22  # S-N curve intercept [MPa**k]
k = 7.3 # S-N curve inverse slope [/]

# Spectral data
sd = FLife.SpectralData(input=(x, dt))

# Rainflow reference fatigue life
rf = FLife.Rainflow(sd)

# Spectral methods
dirlik = FLife.Dirlik(sd)
tb = FLife.TovoBenasciutti(sd)
print(f'          Rainflow: {rf.get_life(C = C, k=k):4.0f} s')
print(f'            Dirlik: {dirlik.get_life(C = C, k=k):4.0f} s')
print(f'Tovo Benasciutti 2: {tb.get_life(C = C, k=k, method="method 2"):4.0f} s')
```

Listing 1: Example `FLife` usage.

4 Conclusion

The FLife package has proved to be an applicable tool that has enabled us to perform vibration fatigue life calculation with great reliability and simple usage. It offers options for implementation of new spectral methods and can be also used as a tool in addressing complex vibration fatigue phenomenon (multiaxial loads, non-stationary loads, complex materials and manufacturing technologies). We believe that the open-source approach to its development can benefit the scientific community, as well as improve the quality of the package itself through community input.

References

[1] A. Zorman, J. Slavic, and M. Boltezar, "Vibration fatigue by spectral methods — a review with open-source support," *Mechanical Systems and Signal Processing*, vol. 190, p. 110149, May 2023.

[2] J. Slavič, M. Mršnik, M. Česnik, J. Javh, and M. Boltežar, *Vibration Fatigue by Spectral Methods*. Elsevier, 2020.

[3] LADISK, "The FLife source code repository." `https://github.com/ladisk/FLife`, Dec 2022.

py-Fatigue: Efficiently process, store and analyse load data for fatigue assessments

Pietro D'Antuono Wout Weijtjens*

Offshore wind infrastructure-Lab (OWI-lab), Vrije Universiteit Brussel, Pleinlaan 2, 1050 Brussels, Belgium

Abstract

py-Fatigue is an open-source Python package that performs cycle counting along with stress-life or crack propagation calculations. It is designed to help engineers and researchers quantify the fatigue life of structures with a finite fatigue life. It resolves several practical challenges in the collection and handling of continuous strain measurements on operational structures, e.g., wind turbines, for extended periods of time up to several years. Among its features it includes tools to export and store cyclecount data, to import stored cyclecount data and efficiently combine results and calculate fatigue rates over the entire operational life.

The package uses well-established fatigue analysis methods, such as the ASTM rainflow counting algorithm and a dedicated algorithm to accurately retrieve the effect of low-frequency fatigue cycles when combining multiple cycle-count matrices (cycle_count module). The damage module implements the most common linear and nonlinear stress-life damage accumulation rules and the Paris law to model the damage initiation and propagation effects of cyclic loading on materials. The mean_stress module also includes the most common mean stress effect correction models. Concerning crack-propagation, the geometry module implements the most common geometrical correction factors. Numba (just-in-time compiler) is the core dependency of the cycle-counting and damage estimation implementation; in this way it is possible to approach the speed of C or FORTRAN. Concerning visualization and interpretation of the analysis, Py-Fatigue provides several tools depending on the Pandas, Matplotlib and Plotly packages. (https://github.com/OWI-Lab/py_fatigue).

Keywords: Fatigue, Structural Health Monitoring, Open Source, Load measurements, Python

1 Introduction

1.1 Motivation

The py-Fatigue toolbox was developed as part of the research activities of the Offshore Wind Infrastructure lab (OWI-lab). At OWI-lab one of our main research topics has been the development of a methodology to monitor the structural health (SHM) of offshore wind turbines (OWTs). As OWTs are often fatigue driven structures it is essential to collect the fatigue load histories of these structures. Therefor we have instrumented dozens of OWTs with sensors such as strain gauges and

* Corresponding author, Email address: wout.weijtjens@vub.be

fiber optical strain gauges. Measurements are often collected continuously over the course of several years to obtain a detailed look at the fatigue life of each instrumented asset.
To achieve this goal, we've built up a codebase to process continuous streams of measurement data, store this data in Postgres databases and perform fatigue calculations, alongside other SHM tools such as Operational Modal Analysis.

1.2 Stepping away from the monolith and separating concerns

In 2017 the existing codebase was transitioned from MATLAB to Python. The resulting **DYNA***wind* package enveloped many features, ranging from sensor-configuration management, database interaction, signal processing tools, visualization of both timeseries and long-term statistics, operational modal analysis and fatigue assessment tools. Such a monolithic Python package was convenient at first, with many application-specific features baked deep into the code. However, it soon became difficult to maintain and distribute. In particular, the requirement to share the codebase as a whole became a limiting factor when collaborating with external researchers or even students. As early as 2018 we started splitting up the functionality into a family of smaller, more dedicated packages based on functionality, e.g., signal processing, database interactions and fatigue assessments (*dw_fatigue*). While this remediated the need to always share the whole codebase, the packages were still strongly intertwined, with dependencies between them. For our application this posed no problem, the packages functioned as cogs to a machine that was always assembled in a similar way. However, more advanced packages such as *dw_fatigue* had private dependencies to almost all other packages in the **DYNA***wind* family. To share *dw_fatigue* thus still required sharing and maintaining the entire **DYNA***wind* family. Despite the best intentions, and despite the separated packages, the codebase still behaved as a monolith.
In 2021 we realized that if the highest quality of code is to be achieved, key functionality needed to be split from the family. We adopted a strategy of 'separation of concerns' in which small, dedicated packages, with a strong CI/CD pipeline, were to be prioritized. A key prior requirement was that there would be no private dependencies to facilitate widespread distribution. Unit testing would guarantee both the correct functioning of the code and a stable interface with the larger **DYNA***wind* family. py_fatigue is the first package of OWI-lab that achieves this goal.

2 py-Fatigue

py_fatigue serves to process long term measurements of bending moments and strain histories into estimates of fatigue damage. While the toolbox envelops many relevant features, we would like to use this abstract to highlight a one key element, the py_fatigue.CycleCount class.

The py_fatigue.CycleCount class incorporates the main information about a processed signal for fatigue analysis. CycleCounts can be instantiated from stress time series. Provided the cycle-counting parameters, time series are cycle-counted on instantiation through an implemented ASTM rainflow [1] analysis routine based on the Numba [2] package. Numba can produce performances comparable with C or FORTRAN languages, hence overcoming slow computation times, arguably the biggest limitation of Python.

CycleCount also embeds an implementation of the __add__ magic method that is based on a brilliant idea of Amzallag *et al.* [3], later also applied to the wind energy sector by Marsh *et al.* [4]. The idea solves the discrepancy that arises when comparing a rainflow matrix obtained from

concatenating multiple stress time series with the bare sum of multiple rainflow matrices coming from cycle-counting the time series separately, since the concatenated time series embeds some low-frequency fatigue cycles that cannot be seen when cycle-counting the signals separately. Amzallag *et al.* [3] found that storing the sequence of half-cycles (or residuals, i.e., the open hysteresis loops) alongside the cycle-count matrix solves this discrepancy, as concatenating and cycle-counting the residuals sequence correctly recovers those low-frequency cycles that would otherwise be lost. The following pseudocode explains the approach.

Summing multiple CycleCount instances and recovering the low-frequency fatigue dynamics

```
import numpy
from py_fatigue import CycleCount

# ts1, ts2: 1Darray  # time series 1 and 2
ts_conc = numpy.append(ts1, ts2)  # concatenated time series
cc_sum_ref = CycleCount.from_timeseries(ts_conc)  # reference

cc1 = CycleCount.from_timeseries(ts1)
cc2 = CycleCount.from_timeseries(ts2)

cc_sum_lf = (cc1 + cc2).solve_lffd()  # Low-frequency recovery
cc_sum_ref == cc_sum_lf  # returns True
```

Once instantiated, a CycleCount can be exported a as a Python dictionary and saved to JSON file. The file can be re-imported as CycleCount for later use. Avoiding the rainflow counting operation at every instantiation guarantees improved performances.

Comparing CycleCounts from time series and from rainflow

```
cc_rf = cc1.as_dict(**kwargs)# Export as a dictionary

cc_from_rf = CycleCount.from_rainflow(cc_rf)
cc1 == cc_from_rf  # Returns True
```

The CycleCount class also implements multiple mean stress correction methods as well as visualisation methods. CycleCount instances can be undergo mean used to perform stress-life or crack growth analyses, provided the proper material properties. Currently, py-Fatigue implements the SNCurve and ParisCurve classes that respectively allow the usage of the py_fatigue.damage.stress_life and py_fatigue.damage.crack_growth modules. These modules implement fatigue metrics such as the Palmgren-Miner rule [5], [6], some non-linear damage accumulation rules (e.g., [7], [8]) and the Paris' law which can be applied to multiple geometries, such as infinite plates or hollow cylinders.

3 Use case: properly accounting for long term cycles in wind turbines

In our recent publication [9] the functionality to resolve long term cycles is used to process several years of monitoring data of an OWT. Figure 1 shows the residuals sequence of data collected from an OWT over the course of several days. It shows how the long term signal comprises large and slow varying cycles that are not considered when processing data in 10 minute blocks.

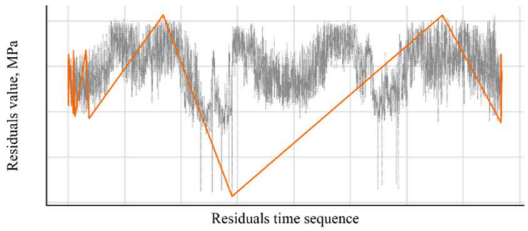


Figure 1: Residuals sequence – grey – over several days of an offshore wind turbine and residuals of the cycle-counted residuals sequence – orange –

Using the tools in py_fatigue we were able to calculate that over the course of a three year period not including these long term cycles can lead up to underestimating the accumulated fatigue life by a factor of 2 for a Wöhler slope m=5.

References

[1] ASTM International, *Standard Practices for Cycle Counting in Fatigue Analysis*, vol. 85. 2017. Accessed: Dec. 27, 2021. [Online]. Available: https://www.techstreet.com/standards/astm-e1049-85-2017?product_id=1983342

[2] S. K. Lam, A. Pitrou, and S. Seibert, 'Numba: a LLVM-based Python JIT compiler', in *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, in LLVM '15. New York, NY, USA: Association for Computing Machinery, Nov. 2015, pp. 1–6. doi: 10.1145/2833157.2833162.

[3] C. Amzallag, J. P. Gerey, J. L. Robert, and J. Bahuaud, 'Standardization of the rainflow counting method for fatigue analysis', *International Journal of Fatigue*, vol. 16, no. 4, pp. 287–293, Jun. 1994, doi: 10.1016/0142-1123(94)90343-3.

[4] G. Marsh *et al.*, 'Review and application of Rainflow residue processing techniques for accurate fatigue damage estimation', *International Journal of Fatigue*, vol. 82, pp. 757–765, Jan. 2016, doi: 10.1016/j.ijfatigue.2015.10.007.

[5] A. Palmgren, 'Die lebensdauer von kugellargern', *Zeitshrift des Vereines Duetsher Ingenieure*, vol. 68, no. 4, p. 339, 1924.

[6] M. A. Miner, 'Cumulative damage in fatigue journal of applied mechanics 12 (1945) no. 3, pp', *A159-A164*, 1945.

[7] D. Pavlou, 'A deterministic algorithm for nonlinear, fatigue-based structural health monitoring', *Computer-Aided Civil and Infrastructure Engineering*, p. mice.12783, Oct. 2021, doi: 10.1111/mice.12783.

[8] Hectors Kris and De Waele Wim, 'Cumulative Damage and Life Prediction Models for High-Cycle Fatigue of Metals: A Review', p. 32, Jan. 2021.

[9] N. Sadeghi, K. Robbelein, P. D'Antuono, N. Noppe, W. Weijtjens, and C. Devriendt, 'Fatigue damage calculation of offshore wind turbines' long-term data considering the low-frequency fatigue dynamics', *J. Phys.: Conf. Ser.*, vol. 2265, no. 3, p. 032063, May 2022, doi: 10.1088/1742-6596/2265/3/032063.

Strategies and tactics for a sustainable open source project

Gunnstein T. Frøseth Anders Rönnquist

Norwegian University of Science and Technology, Department of Structural Engineering, Rich. Birkelands vei 1A, 7491 Trondheim, Norway

Abstract

Open source software (OSS) is used everywhere in today's society, and has been essential to the great technological development and increase in living standards experienced from the end of the previous century. OSS is used in many parts of society, from home electronics such as routers and TV's, to navigation systems in cars or airplanes. Despite the many success stories found in OSS, many OSS projects fail and does not survive long term. There can be several reasons as to why an OSS project fails, from lack of user base, to lack of resources for development and maintenance. Large and complex OSS can be managed and maintained by several large organization with hundreds or even thousands of contributors with ample resources, but more common, and especially for smaller and more specialized software, is that a few people or even a single individual maintains and develop the software. This paper presents strategies and tactics implemented in a small and specialized software, named fatpack, for ensuring that the software survives long term.

Keywords: Open Source Software, Sustainable open source project, Software management, Fatigue Analysis, Python

1 Introduction

Material fatigue is a failure mode that can cause catastrophic collapse of structures and machinery and must be considered in both in design of new structures and assessment of existing structures. There are several approaches to perform fatigue analysis, but the stress-life approach is commonly adopted in design and assessment of structures in the elastic domain and where both crack initiation and crack growth must be included, i.e. the majority of use cases in mechanical and civil engineering structures. Fatigue analysis is therefore a common activity in the life of most mechanical, structural and civil engineers.

Python is arguably the most popular programming language among engineers due to being free and open source, easy to use and extend, portable and available on virtually any operating system and having a large and active community of scientists and engineers that provides support for all aspects of programming. All these features, but in particular the fact that it is easy to use and extend, enables users in any discipline to implement and share the latest domain algorithms and methods and further help expand the applicability of the programming language.

fatpack is an open source package for fatigue analysis in Python [1]. The package aims to provide the most important functionality for fatigue design and assessment in machinery and structures and thus serving the engineering community with a free, reliable and useful toolbox for every day work.

Currently, the package has implementation of stress-based damage accumulation methodology. This includes rainflow cycle counting for conversion of time series to stress ranges, mean and compressive stress range correction, such as Goodman, Gerber, Smith-Watson-Topper and Walker stress correction methods, fatigue endurance with SN-curves and linear damage accumulation by Miner's summation.

fatpack aims to be a sustainable open source project to ensure that all the work that has gone into the package so far is preserved and ensure that the python users and engineering community has a free and reliable alternative for fatigue analysis. This manuscript gives reflections from the maintainers on how to achieve a sustainable open source project.

2 Strategies and tactics for a Sustainable open source project

Research has shown that the most important factor for open source software projects survive long term [2], is to ensure that sufficient developer resources are available at all times. To achieve this, it is essential to attract and retain users of the project, which in turn ensures a steady supply of potential developer resources to the project. fatpack adopts two strategies to ensure that suffient developer resources are available:

A) minimize need for developer resources
B) maximize number of users

These strategies are implemented by the following practices in package development and maintenance:

Good documentation:
Good documentation means that each module, function and class have a concise and descriptive documentation. This means liberal use of examples within the code and in package repository to help users in application of the functionality. The package preferably implements algorithms and methods that are well described in the literature and good documentation provides references the literature. Good documentation ensures that users need less support (by developers) in use and allows users to adopt and use the package more easily. Good documentation therefore supports both strategy A and B.

Complete test suite:
Software testing validates the package functionality by verifying the output of code against known input/output test cases. A complete test suite means that every statement in the package is covered by a test case. This ensures that the software works as intended and avoids any bugs being introduced in maintenance and extension of the package. This eases maintenance and thus reduces the need for developer resources. Avoiding bugs and having a validated software also ensures a better user experience and confidence. Complete test suite therefore supports both strategy A and B.

Few and mature dependencies:
A dependency is a software code or package that is used in the project package. The benefits of dependencies are that code can be reused and in turn allows faster development. The initial developer resource usage is therefore reduced by software dependencies. However, the drawback is that the project becomes dependent on the health and development of the dependency. Any bug or (restrictive) design choice introduced in a dependency is propagated to the current project. Dependencies will therefore deteriorate user experience by increasing the number of bugs and restrictions, and increases

the need for developer resources in maintenance to fix bugs and provide support for users. Only using tested, well managed and maintained packages in project development, i.e. only using mature dependencies, supports both strategy A and B.

Liberal licensing:
A liberal license allows free use of the software under some limitations and conditions and is very attractive for any user. It supports further software development and commercial use, ultimately increasing the user base. Liberal licensing supports strategy B.

Responsive and supportive community:
A supportive community gives timely and inclusive answers to issues and pull requests from users. Creating an inclusive environment encourages more package users to become contributors to the software project. This requires more from the developers/maintainers of the package, i.e. more developer resources (against strategy A), but with the other practices mentioned above in place, the number of issues should be relatively low. Hopefully, the increased developer resources is outweighed by the recruitment of new resources from the increased user base. A responsive and supportive community therefore supports strategy B.

Finally, it should be noted that promotion of the software is a very effective for increasing the user base. The fatpack project experienced a large increase in user base after the package was suggested in common python and engineering forums, e.g. StackOverflow and Research Gate. Active promotion should be used more in the future to support strategy B.

3 Conclusion

This manuscript has presented the fatpack package for fatigue analysis in python. The aim of the package is to serve python users and the engineering community with a package for fatigue analysis today and in the future. To achieve this, fatpack must be a sustainable open source project. The authors have provided the reflections on strategies and practices for a sustainable open source project.

References

[1] Frøseth; Gunnstein T. et. al. “fatpack –Fatigue Analysis in Python” Zenodo, 2022. DOI: 10.5281/zenodo.7246500

[2] Chengalur-Smith, Indushobha; Sidorova, Anna; and Daniel, Sherae L. "Sustainability of Free/Libre Open Source Projects: A Longitudinal Study," Journal of the Association for Information Systems, 11(11), 2010. DOI: 10.17705/1jais.00244

PyIDI: Open-source image-based vibration measurement in Python

Domen Gorjup * Janko Slavič Miha Boltežar

University of Ljubljana, Faculty of Mechanical Engineering

Abstract

PyIDI is an open-source Python package designed to simplify the workflow of detecting displacements in digital images, with a focus on measuring vibrations. It implements several image-based displacement identification algorithms, focusing on methods that prioritize subpixel accuracy and numerical efficiency when analyzing large amounts of high-speed camera data, namely the Simplified Optical Flow method and the Lucas-Kanade image correlation algorithm. PyIDI is designed as a programming library that can be easily integrated into Python image processing workflows. Its data model is built around the Numpy ndarray data type, which is used to represent the input images as well as the resulting displacement data. However, the Photron MRAW image data type is supported out of the box, with the PyMRAW open-source package providing efficient memory mapping for working with large data sets. The code is structured using a modular, object-oriented programming approach, so it can be easily extended with custom displacement identification algorithms. In addition to the user-friendly programming interface, a simple graphical interface based on the Napari image viewer is included. PyIDI includes unit tests and automatically generated, publicly available documentation. The source code is openly available through pyIDI's GitHub repository with a permissive MIT license and is open to the community issue reporting as well as feature and pull-requests.

Keywords: Image-based, optical flow, DIC, optical methods, vibration measurement

1 Statement of Need

Vibration measurement based on images, captured by a high-speed camera, has become a valid alternative to the more established measurement methods in recent years. Image-based methods enable non-contacting response measurement of the device under test with a high spatial resolution, opening numerous new possibilities in the fields of operational and experimental modal analysis. The methods are not without limitations, however. High computational complexity and a multitude of possible sources of error, resulting in a relatively low signal to noise ratio of the resulting displacement data, are often deterring factors when considering image-based vibration measurement techniques. The main aim of the pyIDI package[1], an open-source Python toolkit for image-based displacement identification in the field of vibration testing, is therefore to accelerate and standardize the processing of images in vibration measurements to ensure the repeatability and validity of research. The resulting software package has enabled us to unify the image-processing conventions inside our research group,

*Corresponding author, Email address: domen.gorjup@fs.uni-lj.si

simplify the development and validation of new algorithms and remove some of the uncertainty when integrating the image-based displacement measurement process into broader workflows. The open-source development model also has the potential of improving the quality, efficiency and scope of the implemented methods, either through methodical code testing and documentation or through global community input.

2 PyIDI code structure - modularity and extensibility

From the beginning of the development process, a code structure that promotes clarity of operation and enables easy extensibility of the implemented methods was at the center of the project. The PyIDI framework code structure is illustrated in 1.

2.1 The `pyIDI` class

At the top level of the PyIDI framework is the **`pyIDI`** class, meant to establish a unified interface to the various image displacement identification methods, implemented as a subclasses of the `IDIMethod` class. The `pyIDI` class is responsible for the handling of image data (videos), the configuration of the selected displacement identification method and handling of the computed results. Among other functionality, it implements the following methods:

`__init__` (the class constructor) loads the image data, passed as an argument during object construction. The currently supported types of image data to be loaded are `np.ndarray`, `np.memmap` (a memory-efficient way of loading binary data from secondary - HDD - computer memory) [2] or Photron MRAW high-speed data, using the pyMRAW package [3].

`set_method` : select and configure the `IDIMethod` image displacement identification method to be used for analysis.

`set_points` : select the image-coordinates of the points-of-interest for which the displacements will be calculated.

`get_displacements` : an interface to the selected `IDIMethod`'s displacement computation method. Computes and returns the displacements, using the selected configuration.

2.2 The `IDIMehtod` Class

The second essential part of the PyIDI infrastructure is the `IDIMethod` class, serving as the basis for image-based displacement method implementation.

To facilitate the highest level of extensibility and modularity of the displacement computation framework, the `IDIMethod` class itself does not implement a specific motion identification algorithm, but rather provides a template for all subsequent displacement identification methods. In order to assure compatibility with the `pyIDI` class (2.1), from which the displacement computation will later be called (see Figure 1), the `IDIMethod` class implements the following methods, which are re-implemented when subclassing `IDIMethod`:

`__init__` (the class constructor) links the calling `pyIDI` class and performs initial configuration of the method with arguments, passed during construction.

`configure` : (re)configure the image displacement identification method with currently selected user settings.

`calculate_displacements` : perform the actual displacement computation, using the currently active configuration. For compatibility with `pyIDI`, the resulting displacements should be saved in the 'displacements' attribute of the calling `pyIDI` instance.

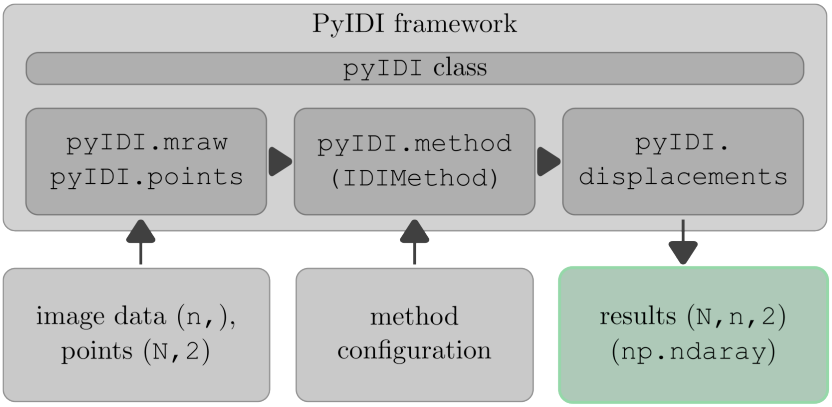


Fig. 1: PyIDI programming framework schematic.

3 PyIDI usage examples

The PyIDI code repository [4] includes examples and showcases to easily get you started. Extensive programming API documentation is also available [5] for users that want to learn more about the inner workings of PyIDI.

4 Conclusions and call for community input

As with every piece of computer software, there is room for improvement in the current code base. The PyIDI package was primarily developed out of need for a unified image processing framework inside our research group. Consequently, the currently supported features are those that we use in our daily workflow, and are extended based on our current research focus.

That is where the open-source community is invited to join in! We believe that the PyIDI package, which is published openly, under the permissive MIT license [6] can serve the scientific community greatly by increasing the clarity and repeatability of published research, as well as lowering the effort required to introduce researchers, working in the field of structural dynamics, to image processing methods.

Building a programming library, useful to a wide scientific community, is not feasible without open extensive multi-institutional community input. The easiest way to contribute to PyIDI is through the GitHub's repository's "Issue" and "Pull Request" features [4]. As the individual potential contributor's interests, areas of expertise and time constraints are unique, there are multiple potential areas and scopes of open-source contributions.

The following list of potential community contributions to PyIDI is sorted by increasing level of involvement and expected effort requirements:

1. Additional unit-tests;
2. Standardization of `IDIMethod` unit testing;
3. Standardization of `IDIMethod` documentation;
4. Standardization of the multiprocessing workflow;
5. Implement additional motion identification methods (e.g. DIC with arbitrary geometrical transform shape functions, phase-based motion identification etc.).

As is hopefully evident from this publication, open-source development process can have multiple advantages. However, these can be fully realized only with strong foundations, based on clear code, extensive testing, sufficient documentation and above all, an involved community.

References

[1] K. Zaletelj, D. Gorjup, and J. Slavič, "ladisk/pyidi: Release of the version v0.23," Sep. 2020. [Online]. Available: https://doi.org/10.5281/zenodo.4017153

[2] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep 2020. [Online]. Available: https://doi.org/10.1038/s41586-020-2649-2

[3] J. Javh, J. Slavič, and D. Gorjup, "pyMRAW: Photron MRAW File Reader for Python," May 2023. [Online]. Available: https://doi.org/10.5281/zenodo.7928283

[4] LADISK, "The PyIDI source code repository," https://github.com/ladisk/pyidi, May 2023.

[5] K. Zaletelj, D. Gorjup, and J. Slavič, "The PyIDI documentation," https://pyidi.readthedocs.io/en/latest/index.html, May 2023.

[6] Open Source Initiative, "The MIT licence," https://opensource.org/license/mit/, Feb 2023.

PyExSi: Excitation signals for structural dynamics and vibration fatigue

Domen Gorjup * Janko Slavič Aleš Zorman Miha Boltežar

University of Ljubljana, Faculty of Mechanical Engineering

Abstract

PyExSi is an open-source Python package designed to easily generate excitation signals used in structural dynamics applications such as experimental modal analysis and vibration fatigue testing. It supports several commonly used signal types, including pulse (half-sine) excitation signals, random and pseudo-random, burst random, and sine sweep signals with easily configurable frequency content. To process data, pyExSi uses the open-source Numpy and Scipy libraries, ensuring reliability and repeatability through well-tested and optimized source code for numerical processing. The data model is based on the Numpy ndarray data type, which is used to define the required signal parameters (e.g. PSD of the random signal) as well as the resulting excitation signals. One of the main features of pyExSi is the random signal generation method, which can generate both Gaussian and non-Gaussian, non-stationary random signals. The code is structured using a clear and readable functional programming approach to allow easy modification and extension. The functional programming approach is also reflected in pyExSI's user-friendly programming interface, which allows seamless integration into existing Numpy-based signal processing workflows. PyExSi includes unit tests and publicly available examples of usage. The source code is openly available through pyIDI's GitHub repository with a permissive MIT license and is open to the community issue reporting as well as feature and pull-requests.

Keywords: Excitation, signal generation, vibration measurement, vibration fatigue, non-stationary

1 Statement of Need

The PyExSi package development began from a need to standardize as well as simplify the generation of excitation signals for vibration testing applications inside our research group. The open-source publication [1] of the package has enabled us to expedite the planning of experiments testing procedures, promote cooperation as well as increase the repeatability of our research. The open-source development model also has the potential of improving the quality and reliability of the developed solutions, either through methodical code testing and documentation or through global community input.

2 Supported excitation types

The PyExSi package supports the generation of multiple excitation signal types. Examples of some of the available signal types are shown in Figure 1. See the source code [1] for more information.

*Corresponding author, Email address: domen.gorjup@fs.uni-lj.si

- Impulse (half-sine, square, triangular...) (`pyExSi.impulse`);
- Sine sweep (`pyExSi.sine_sweep`);
- Normal random (`pyExSi.normal_random`);
- Uniform random (`pyExSi.uniform_random`);
- Pseudo-random (`pyExSi.pseudo_random`);
- Burst random (with controllable burst ratio and random distribution) (`pyExSi.burst_random`);
- Stationary Gaussian random (from PSD) (`pyExSi.random_gaussian`);
- Stationary non-Gaussian random (`stationary_nongaussian_signal`);
- Non-stationary non-Gaussian random (`nonstationary_signal`);

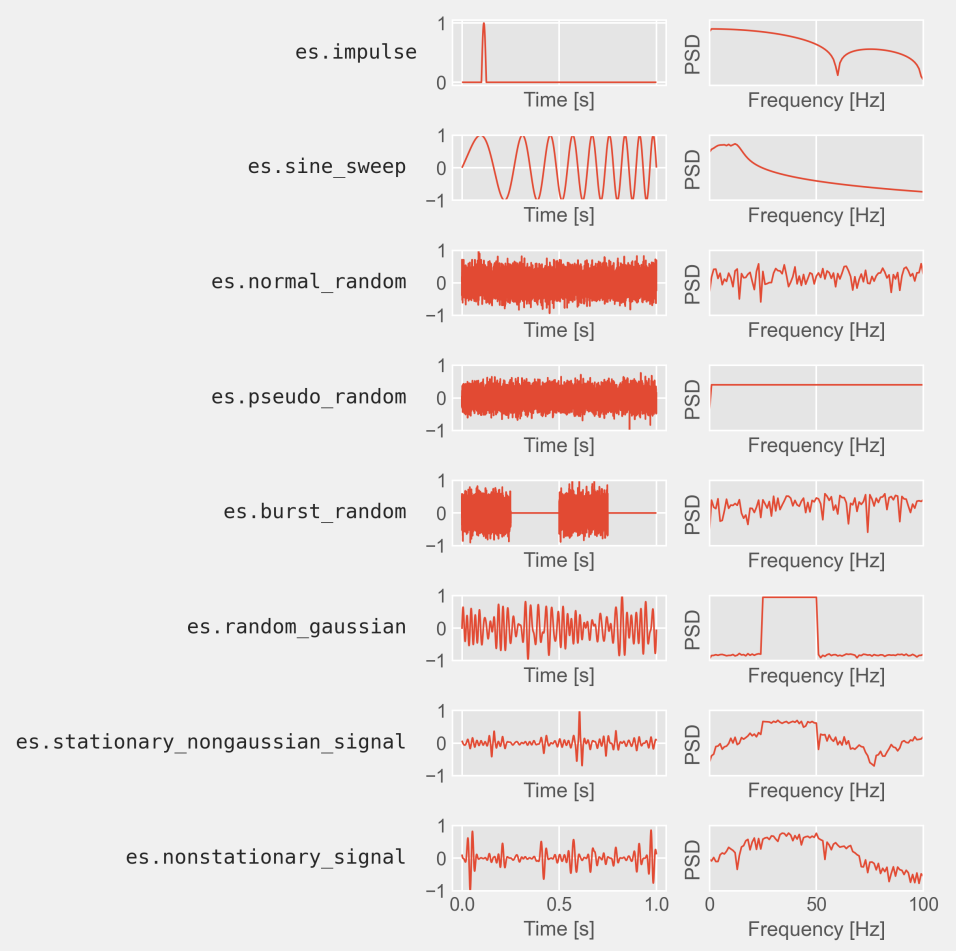


Fig. 1: Examples of excitation signal types available in `pyExSi`.

3 PyExSi usage example

PyExSi is built using the functional programming paradigm, with the Numpy [2] `ndarray` array data type as its main output (and input) data type. It is designed to easily fit into existing Python data processing workflows.

Code Listing 1 shows a short example of using PyExSi to generate two different types of random excitation signals. More detailed examples are available in the PyExSi source code repository [3].

```
import pyExSi as es
import numpy as np

N = 2**16 # number of data points of time signal
fs = 1024 # sampling frequency [Hz]
t = np.arange(0, N) / fs

# Define frequency vector and one-sided flat-shaped PSD
M = N//2 + 1
freq = np.arange(0, M, 1) * fs / N
freq_lower = 50 # PSD lower frequency limit  [Hz]
freq_upper = 100 # PSD upper frequency limit [Hz]
PSD = es.get_psd(freq, freq_lower, freq_upper) # one-sided flat-shaped PSD

# Gaussian stationary signal
gausian_signal = es.random_gaussian(N, PSD, fs)

# Non-gaussian non-stationary signal, with kurtosis k_u=10
PSD_modulating = es.get_psd(freq, freq_lower=1, freq_upper=10)
# Define array of parameters delta_m and p
delta_m_list = np.arange(0.1, 2.1, 0.5)
p_list = np.arange(0.1, 2.1, 0.5)

# Get signal
nongaussian_nonstationary_signal = es.nonstationary_signal(
    N, PSD, fs, k_u=5,
    modulating_signal=('PSD', PSD_modulating),
    param1_list=p_list,
    param2_list=delta_m_list)
```

Listing 1: Example `pyExSi` usage.

4 Conclusion

The PyExSi package has proved to be a useful tool that has enabled us to expedite the planning of experiments and simplify the test procedures in our work group. We believe that the open-source approach to its development can benefit the scientific community in a similar way, as well as improve the quality of the package itself through community input.

References

[1] LADISK, "The pyExSi source code repository." `https://github.com/ladisk/pyExSi`, May 2023.

[2] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, pp. 357–362, Sep 2020.

[3] LADISK, "The pyExSi usage examples." `https://github.com/ladisk/pyExSi/tree/main/example`, May 2023.

Digital twin of washing machine tub-drum oscillations

Gianluca Guernieri[1] Paolo Gardonio[1] Marius Stücheli[2]

[1] *Università degli Studi di Udine - DPIA, Via delle Scienze 206, 33100, Udine, (IT)*
[2] *Electrolux Professional AB - Franzéngatan 6, SE-112 51 Stockholm, (SE)*

Abstract

This paper presents the development of a digital twin model, which can be used to detect online the cross-plane displacements and the centrifugal force of the drum-suspension assembly of a spinning washing machine. The work is structured in three parts. The first part presents background measurements of the cross-plane displacements during a test spinning cycle (black box model), which are indispensable to identify off-line the principal features of the dynamic response of the drum-suspension assembly. The second part discusses the derivation of the lumped parameter model for the plane dynamics of the drum-suspension assembly (white box model), which has been built starting from the black box information. Finally, the third part describes the digital twin model (grey box model), which encompasses an augmented Kalman filter such that the displacements of the drum assembly and the centrifugal force of the load are reconstructed online starting from acceleration measurements of the drum assembly vibrations. The paper is focused on the hardware and software development of the digital twin model, which is based on a dSPACE platform and Simulink-MatLab software.

Keywords: digital twin; augmented Kalman filter; washing machine vibrations; black box model; white box model; grey box model.

1 Introduction

Compared to domestic appliances, professional washing machines are specifically designed for heavier garment loads, higher unbalance and faster washing cycles. Furthermore, the environmental impact of laundry activities has become subject of increasing awareness and attention. Washing machine manufacturers strive to reduce the consumption of chemicals, water and heating energy in the washing cycle. Another optimization objective is the reduction of the residual moisture content in the garments after extraction, to minimize the energy consumption in the subsequent drying process, resulting in higher rotational speeds during extraction. For these reasons, commercial washers will face larger oscillations of the drum assembly elements, mainly when the drum angular speed hits the resonance frequencies of the fundamental natural modes of the drum-suspension assembly, and larger force transmission to the floor, particularly at the high angular speeds during the extraction phase [1]–[3]. To ensure the correct functioning of the machine and avoid failures due to wide oscillations and high forces transmitted to the ground, better washing machine models are equipped with accelerometer sensors on the drum assembly.

The aim of this paper is to investigate the possibility of increasing the use of these sensors in such a way as to bring forward improved designs and improved operation cycles of the washing machine. To this end, a new methodology is emerging in the industry the so-called digital twin [4]–[6] which can be seen as a digital duplicate of a physical system that can be used to better understand the system and, as a consequence, to optimize the design process, to improve the resulting design, as well as to optimize the operation of a system. The principal characteristics of a Digital Twin can thus be synthesized into the following points [7]:

- The Digital Twin forms a high-fidelity simulation model of a system, which, among others, encompasses the design information and constraints, the physical models, the measured data (including online measurements), and the experts' knowledge.
- The Digital Twin grows and evolves along with the real system throughout its life cycle in such a way as to integrate the whole knowledge gathered and developed during the design, production, and operation phases of the system.
- The Digital Twin is not a mere simulation tool, but it can be used to monitor and optimize the real system.

A digital twin can be used to perform several tasks. In this study, the interest is in predicting the low-frequency vibrations of the drum-suspension assembly of a professional washing machine and the centrifugal force produced by the garments. Conrad and Soedel [8], showed that the low-frequencies dynamic response of a washing machine can be satisfactorily modeled with rudimentary lumped parameter models encompassing few degrees of freedom. However, the effective dynamic response of the drum-suspension assembly strongly depends on the excitation force components exerted by the garments in the drum, which are characterized by high variability. In this paper, a time-domain identification is employed based on the Kalman filter [9]–[11] optimal estimator [12]. The standard Kalman filter requires knowledge of the excitation force, which is not available. To overcome this problem, an augmented Kalman filter has been developed [10], which, recently, has been the subject of several applications and studies for vibration problems [13–17].

The paper is structured into two sections. In Section 2 the fundamental features of the digital twin are studied: first, the black box of the structure is presented, with a brief description of the measurement setup and the analysis of the measures taken. Then, the lumped parameter model is introduced, which is limited to the plane oscillations of the drum-suspension assembly. At the end of this Section, the grey box model is described and the formulation of the augmented Kalmen filter is reported. Section 3 presents the experimental results on the implementation of the proposed augmented Kalman filter to reconstruct both the displacements of the drum-suspension assembly and the force exerted by the garments load in the tub. Finally, Section 4 provides the principal conclusion of this study. There is also an appendix that recalls the code used to implement the augmented Kalman filter.

2 Digital twin

To better understand what a digital twin is, an introduction to different ways to model a physical system is required. The "***black-box***" models are derived entirely from measured data, with no knowledge of physics required. On the other side, a model can be built with physics-based reasoning, then the object of interest is called a "***white-box***" model and the solution is performed by solving differential equations, assuming the parameters are known. In between these two extremes, there are the "***grey-box***" models, a combination of both physics-based and data-based modelling. This hybrid model is exactly the format required for a digital twin, which is a virtual duplicate of a physical system built from a fusion of models and data.

Figure 1 summarizes the three ways to model the washing machine. On the right side, the physical system is shown, while on the left side, the plane lumped parameter model is displayed. These two models, black and white boxes, merge together to form the grey box model in the middle, which takes, from the black box, the measured data and, from the white box, the parameter estimated by physics-based reasoning. The three models are now presented in detail.

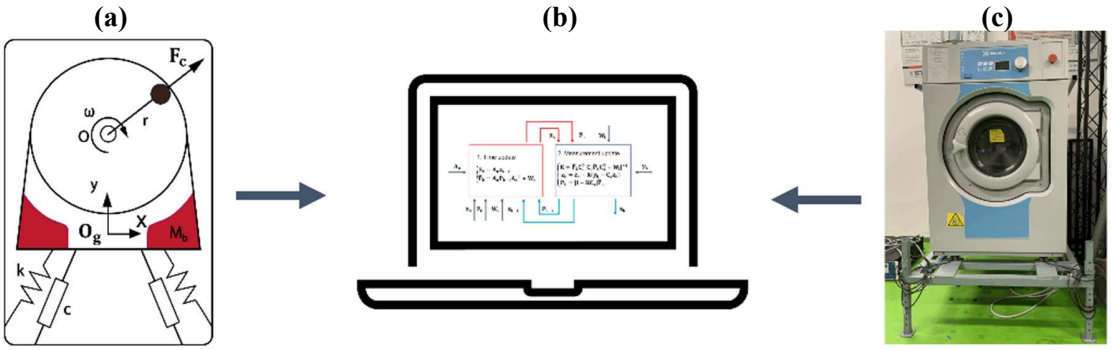


Fig. 1: The ***white-box*** (a), ***grey-box*** (b) and ***black-box*** (c) models of a washing machine

2.1 Black box: measured drum vibration

The right side of Figure 1 shows the washing machine used in this study, which was fixed on a rigid framework structure equipped with four force load cells located under the feet of the machine to measure the force transmitted to the ground. The tub was equipped with two accelerometers: the first on the top of the cylinder to measure the acceleration in the vertical direction and the second on the left-hand side of the cylinder to measure the acceleration in the transverse direction. To mimic the load exerted by the clothes inside the drum, small bags of sand were fixed on the drum inner surface.

The time-histories of the vertical and horizontal displacements were recorded when, as shown in Figure 2a, the drum rotation was set to undergo a constant acceleration ramp until the spinning speed was reached, then a short interval at extraction speed and, finally, a constant deceleration slope from the maximum angular speed to the rest condition.

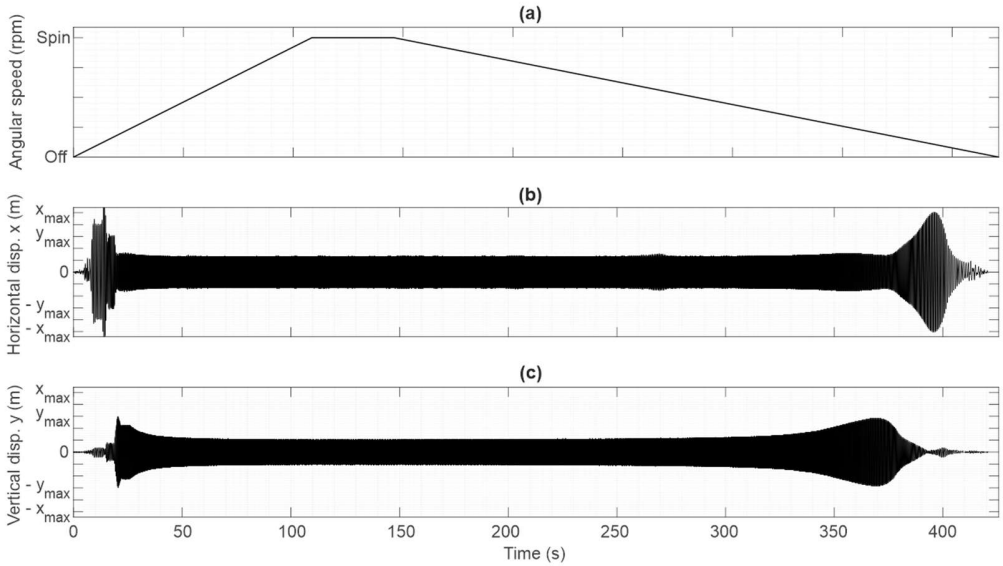


Fig. 2: Time histories of the drum angular speed (a), the horizontal displacements (b), and the vertical displacements (c).

Figures 2b and 2c show that the drum oscillations tend to rise as the angular speed builds up from zero and is then significantly amplified when the angular speed passes by the resonance frequencies

of the fundamental natural modes of the drum-suspension system. At higher frequencies, the oscillations level to constant values, which are dictated by the specific unbalance. The same displacements effect is noticed during the constant deceleration. Therefore, these graphs show that the resonance peak linked to the horizontal natural oscillations occurs at lower frequencies than the resonance peak due to the vertical natural oscillations.

2.2 White box: lumped parameter model

The measurement results presented above have shown that the low-frequency oscillations of the drum-suspension assembly in the horizontal and vertical directions are controlled by a horizontal-like oscillation and a vertical-like oscillation natural modes respectively. Accordingly, the vibrations in the two directions can be assumed uncoupled. The equations of motion for the plane oscillations x,y are thus given by the following second-order differential equations:

$$\begin{cases} m\ddot{x} + c_{\mathrm{x}}\dot{x} + k_{\mathrm{x}}x = F_{\mathrm{cx}} \\ m\ddot{y} + c_{\mathrm{y}}\dot{y} + k_{\mathrm{y}}y = F_{\mathrm{cy}}, \end{cases} \tag{1}$$

where $m = m_{\mathrm{td}} + m_{\mathrm{l}}$, is the mass of the tub-drum assembly (m_{td}) and garments load (m_{l}) and c_{i}, k_{i}, are the equivalent damping and stiffness components in horizontal and vertical direction of the four suspensions. Finally, $F_{\mathrm{c,x}}$ and $F_{\mathrm{c,y}}$ are the component of the centrifugal force exerted by the garments in horizontal and vertical directions respectively.

2.3 Grey box: augmented Kalman filter reconstruction of drum displacements and centrifugal force

The Kalman filter is based on a state-space formulation of the equation of motion, which can be derived straightforwardly by defining the state vector

$$\mathbf{z} = \begin{bmatrix} x \\ \dot{x} \\ y \\ \dot{y} \end{bmatrix}, \tag{2}$$

such that the state equations and the output equation that describe the dynamics of the system in horizontal and vertical direction can be casted in the following standard matrix formulation:

$$\dot{\mathbf{z}} = \mathbf{A}\mathbf{z} + \mathbf{B}\mathbf{F}_{\mathbf{c}} + \mathbf{w}_{\mathbf{x}}\,, \tag{3}$$

$$\ddot{\mathbf{y}} = \mathbf{C}\mathbf{z} + \mathbf{D}\mathbf{F}_{\mathbf{c}} + \mathbf{w}_{\mathbf{y}}\,. \tag{4}$$

where $\ddot{\mathbf{y}} = \begin{pmatrix} \ddot{x} \\ \ddot{y} \end{pmatrix}$ is the vector of the accelerations in horizontal and vertical directions, and the state, input, output and feedthrough matrix are given respectively by:

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\frac{k_{\mathrm{x}}}{m} & -\frac{c_{\mathrm{x}}}{m} & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -\frac{k_{\mathrm{y}}}{m} & -\frac{c_{\mathrm{y}}}{m} \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 0 & 0 \\ \frac{1}{m} & 0 \\ 0 & 0 \\ 0 & \frac{1}{m} \end{bmatrix}, \quad \mathbf{C} = \begin{bmatrix} -\frac{k_{\mathrm{x}}}{m} & -\frac{c_{\mathrm{x}}}{m} & 0 & 0 \\ 0 & 0 & -\frac{k_{\mathrm{y}}}{m} & -\frac{c_{\mathrm{y}}}{m} \end{bmatrix}, \quad \mathbf{D} = \begin{bmatrix} \frac{1}{m} & 0 \\ 0 & \frac{1}{m} \end{bmatrix}. \tag{5-8}$$

The state matrix equation includes the vector $\mathbf{w_x}$ with the process noise variables. Alternatively, the output equation encompasses the measurement noise $\mathbf{w_y}$. The Kalman filter formulation assumes that these are white noise components independent of each other [9–12]. However, to implement the algorithm, the time-history of the centrifugal force excitation is required. In principle, this parameter can be reconstructed from knowledge of the load fit in the drum and the angular speed of the drum. But the variation of load due to the water filling and emptying during the different machine operations, and the sequence of impulses generated by the garments rolling over and falling from the top of the drum rim during low-speed washing operations, make this data difficult to evaluate. An alternative formulation where the excitation force is actually part of the state variables is required. This approach is known as the augmented Kalman filter [13–17], which is described below in detail for the problem at hand.

The augmented Kalman observer employed in this study to simulate the plane vibration of the washing machine is based on a discrete-time extended version of the state space formulation presented above, where the state vector is augmented with the force excitation produced by the load in the drum $F_{c,y}$, such that

$$\mathbf{z_k^a} = \begin{bmatrix} x_k \\ \dot{x}_k \\ y_k \\ \dot{y}_k \\ F_{cx,k} \\ F_{cy,k} \end{bmatrix}. \tag{9}$$

The state space and output equations are thus rewritten in the following form

$$\mathbf{z_{k+1}^a} = \mathbf{A}_a \mathbf{z_k^a} + \mathbf{w_x} \ , \tag{10}$$

$$\ddot{\mathbf{y}}_\mathbf{k} = \mathbf{C}_a \mathbf{z_k^a} + \mathbf{w_y} \ , \tag{11}$$

where

$$\mathbf{A}_a = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{O} & \mathbf{I} \end{bmatrix}, \qquad \mathbf{C}_a = [\mathbf{C} \quad \mathbf{D}] \tag{12,13}$$

The aim of the observer is to reconstruct the state vector in Eq. (9) using the state space model of the system given in Eqs. (10) and (11) and the measured output variables, that are the accelerations in horizontal and vertical directions.

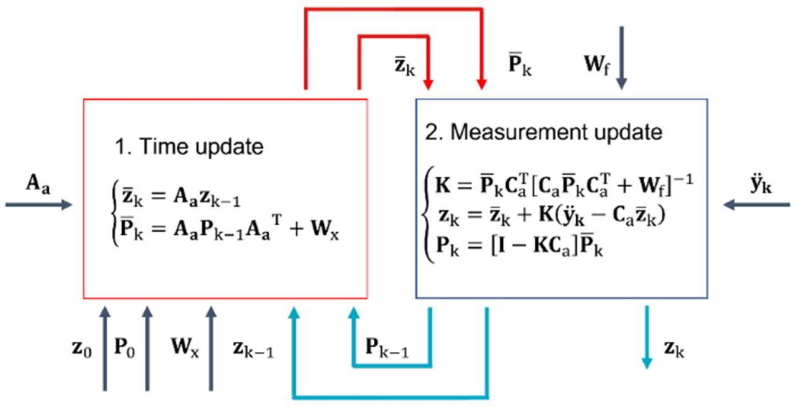


Fig. 3: Block diagram for the augmented Kalman filter algorithm.

The discrete-time implementation of the Kalman filter algorithm to reconstruct the states of the system, i.e. the state vector $\mathbf{z}_k$, can be conveniently described using the block diagram shown in Figure 3. The algorithm is characterized by a loop formed by two blocks. To start with, the so called time-update equations (or predictor equations) project forward in time the current state variables, $\mathbf{z}_{k-1}$, and the error covariance estimates $\mathbf{P}_{k-1}$ to give a priori estimates $\bar{\mathbf{z}}_k$ and $\bar{\mathbf{P}}_k$ for the next time step. Then, the so-called measurement update equations (or corrector equations) generate improved a posteriori estimates $\mathbf{z}_k$ and $\mathbf{P}_k$ through the measured output variables $\ddot{\mathbf{y}}_k$.

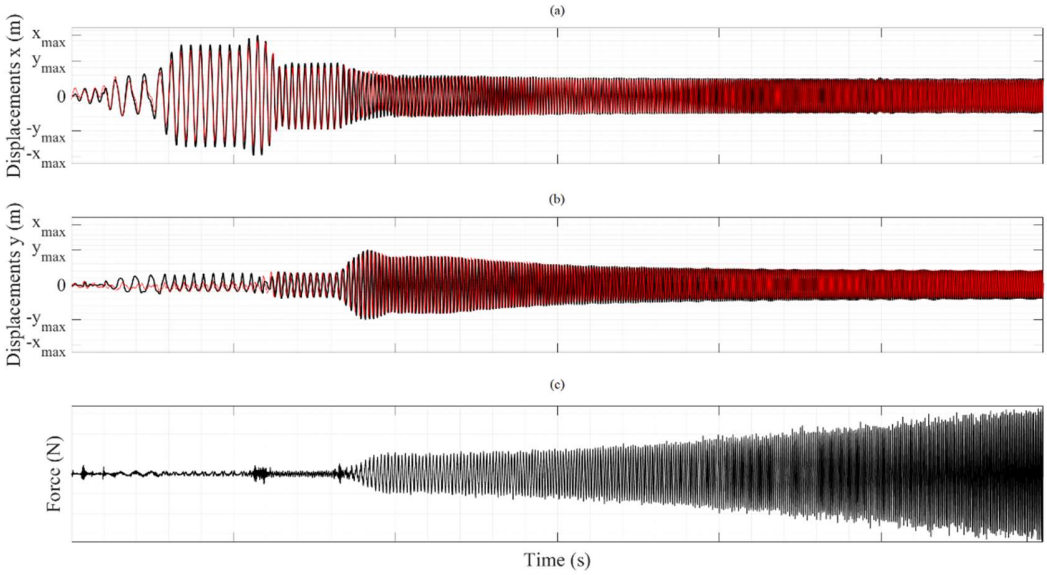


Fig. 4: Measured with cameras (red lines) and reconstructed from Kalman filter (black lines): horizontal (a) and vertical (b) displacements and centrifugal force (c) of the tub when the angular speed of the drum increases from zero to the maximum spinning velocity with constant acceleration.

Figure 4 shows the measured (with a camera [18]) and reconstructed the drum-suspension assembly horizontal (a), vertical (b) displacements and centrifugal force (c) when the angular speed of the tub is raised from zero to the maximum spinning velocity with constant acceleration. The first two graphs show the magnification effects when the angular speed passes by the resonance frequencies of the drum-suspension assembly fundamental modes characterised by horizontal and vertical oscillations respectively. Then, at higher angular speeds, the amplitude of the oscillations levels to a constant value dictated by the specific unbalance [19]. The displacements of the drum oscillations reconstructed with the augmented Kalman filter show a good agreement with those measured by cameras. The displacements in both horizontal and vertical directions are not affected by drift effects, which are typically encountered when the displacements are retrieved directly from measurements of accelerations with a double numerical integration of the measured signal with respect to time.

The bottom plot (c) shows the reconstructed centrifugal force, which, apart the initial phase at low angular velocities of the drum, it is characterised by a harmonic evolution with time and an amplitude amplification proportional to the square of the drum angular speed. The force predicted at low angular speeds of the drum does not replicate the evolution expected when the rotation of the drum undergoes a constant acceleration. This is probably due to the fact that at low angular speeds

the power train does not generate a constant moment excitation and thus a centrifugal force variation with the square of angular velocity.

3 Conclusions

This paper has presented a digital twin model for the online detection of the cross-plane displacements and the centrifugal force of the drum assembly of a spinning washing machine. More specifically, based on a lumped parameter (white-box) model and experimental measurements taken on a (black box) model washing machine, a digital twin (grey-box) model has been developed using an augmented Kalman filter. The study has proven experimentally that the grey-box model can be suitably used to derive the oscillations of the drum assembly in horizontal and vertical directions with no drift effects and, more importantly, can be effectively used to reconstruct the centrifugal force exerted by the wash load.

Appendix A. MatLab code for augmented Kalman filter

This appendix reports the MATLAB code used to implement an augmented Kalman filter for the evaluation of the centrifugal force produced by the garments and the horizontal and vertical displacements of the drum assembly, using only signals from two accelerometers.

```
% Augmented Kalman filter for the evaluation of centrifugal force and plane
% displacements produced by garments during a washing cycle
%
% Input:
%
% Aa: augmented state matrix containing washing machine parameters
% Ca: augmented output matrix
% Q: process noise covariance matrix
% R: measurement noise covariance matrix
% input: accelerations from accelerometers
%
% Output:
% x: augmented state vector containing displacements, velocities and centrifugal
% force

for i=2:length(t)
    % Time update
    x_g(:,i)=Aa*x(:,i-1);
    P_g(:,:,i)=Aa*P(:,:,i-1)*Aa'+Q;
    % Measurement update
    K(:,i)=P_g(:,:,i)*Ca'*(Ca*P_g(:,:,i)*Ca'+R)^-1;
    x(:,i)=x_g(:,i)+K(:,i)*(input(2,i)-Ca*x_g(:,i));
    P(:,:,i)=(eye(3)-K(:,i)*Ca)*P_g(:,:,i);
end
```

References

[1] S. Bae, J. M. Lee, Y. J. Kang, J. S. Kang, and J. R. Yun, “Dynamic analysis of an automatic washing machine with a hydraulic balancer,” *J Sound Vib*, vol. 257, no. 1, pp. 3–18, 2002, doi: https://doi.org/10.1006/jsvi.2001.4162.

[2] T. Nygårds, “Washing Machine Design Optimization Based on Dynamics Modeling,” 2011.
[3] I. Johansson and P. Somasundaran, *Handbook for Cleaning/Decont. of Surfaces*. 2007.
[4] S. Haag and R. Anderl, “Digital twin – Proof of concept,” *Manuf Lett*, vol. 15, pp. 64–66, Jan. 2018, doi: 10.1016/J.MFGLET.2018.02.006.
[5] D. J. Wagg, K. Worden, R. J. Barthorpe, and P. Gardner, “Digital Twins: State-of-the-Art and Future Directions for Modeling and Simulation in Engineering Dynamics Applications,” *ASCE-ASME J Risk and Uncert in Engrg Sys Part B Mech Engrg*, vol. 6, no. 3, May 2020, doi: 10.1115/1.4046739.
[6] D. Jones, C. Snider, A. Nassehi, J. Yon, and B. Hicks, “Characterising the Digital Twin: A systematic literature review,” *CIRP J Manuf Sci Technol*, vol. 29, pp. 36–52, May 2020, doi: 10.1016/J.CIRPJ.2020.02.002.
[7] S. Boschert and R. Rosen, “Digital Twin—The Simulation Aspect,” in *Mechatronic Futures: Challenges and Solutions for Mechatronic Systems and their Designers*, P. Hehenberger and D. Bradley, Eds. Cham: Springer International Publishing, 2016, pp. 59–74. doi: 10.1007/978-3-319-32156-1_5.
[8] D. C. Conrad and W. Soedel, “On the problem of oscillatory walk of automatic washing machines,” *J Sound Vib*, vol. 188, pp. 301–314, 1995.
[9] R. E. Kalman, “A New Approach to Linear Filtering and Prediction Problems,” *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, Mar. 1960, doi: 10.1115/1.3662552.
[10] R. E. Kálmán and R. S. Bucy, “New Results in Linear Filtering and Prediction Theory,” *Journal of Basic Engineering*, vol. 83, pp. 95–108, 1961.
[11] G. Welch and G. Bishop, “Welch & Bishop , An Introduction to the Kalman Filter 2 1 The Discrete Kalman Filter In 1960,” 1994.
[12] D. Simon, *Optimal state estimation: Kalman, H infinity, and nonlinear approaches*. John Wiley & Sons, 2006.
[13] E. Lourens, E. Reynders, G. De Roeck, G. Degrande, and G. Lombaert, “An augmented Kalman filter for force identification in structural dynamics,” *Mech Syst Signal Process*, vol. 27, pp. 446–460, 2012, doi: https://doi.org/10.1016/j.ymssp.2011.09.025.
[14] R. Cumbo, T. Tamarozzi, K. Janssens, and W. Desmet, “Kalman-based load identification and full-field estimation analysis on industrial test case,” *Mech Syst Signal Process*, vol. 117, pp. 771–785, 2019, doi: https://doi.org/10.1016/j.ymssp.2018.08.045.
[15] R. Cumbo, L. Mazzanti, T. Tamarozzi, P. Jiranek, W. Desmet, and F. Naets, “Advanced optimal sensor placement for Kalman-based multiple-input estimation,” *Mech Syst Signal Process*, vol. 160, p. 107830, 2021, doi: https://doi.org/10.1016/j.ymssp.2021.107830.
[16] E. Risaliti, T. Tamarozzi, M. Vermaut, B. Cornelis, and W. Desmet, “Multibody model based estimation of multiple loads and strain field on a vehicle suspension system,” *Mech Syst Signal Process*, vol. 123, pp. 1–25, May 2019, doi: 10.1016/j.ymssp.2018.12.024.
[17] F. Naets, J. Cuadrado, and W. Desmet, “Stable force identification in structural dynamics using Kalman filtering and dummy-measurements,” *Mech Syst Signal Process*, vol. 50–51, pp. 235–248, 2015, doi: https://doi.org/10.1016/j.ymssp.2014.05.042.
[18] G. Guernieri, P. Gardonio, M. Stücheli, A. Fusiello, “Washing Machine Drum Oscillations Envelope From Camera Measurements,” *Proceedings of OpenSD2023*, Ljubljana, SLO, 26-28 June 2023, 2023.
[19] L. Dal Bo, P. Gardonio, N. Battistella, E. Turco “Non-linear Isolator for Vibration and Force Transmission Control of Unbalanced Rotating Machines,” *Journal of Vibration Engineering and Technologies*, 2022, doi: https://doi.org/10.1007/s42417-022-00668-9

Washing Machine Drum Oscillations Envelope From Camera Measurements

Gianluca Guernieri[1] Paolo Gardonio[1] Marius Stücheli[2] Andrea Fusiello[1]

[1] *Università degli Studi di Udine - DPIA, Via delle Scienze 206, 33100, Udine, (IT)*
[2] *Electrolux Professional AB - Franzéngatan 6, SE-112 51 Stockholm, (SE)*

Abstract

This paper presents a case study on the reconstruction from camera measurements of the envelope enclosing the space filled in by the cross-plane oscillations of the drum assembly of a professional washing machine during spinning. More specifically, the 2D space occupied by the front *trim-panel*, where the *loading-door* is hinged, is reconstructed from camera measurements of the cross-plane displacements of four markers bonded on the panel itself. To start with, the cross-plane displacements of these points are estimated via triangulation of the sequence of images acquired by two cameras positioned in front of the washing machine with the optical axis pointing to the centre of the drum. Solid-body geometry is then employed to reconstruct a sequence of *rim-frames* that depict the outer border of the *trim-panel* during the acquisition sequence. Finally, the envelope generated by the plane displacements and rotations of the panel is obtained from the overlap of the sequence of *rim-frames*. The paper illustrates the camera measurements and image processing tasks implemented using the open-source Computer Vision Toolkit (*https://github.com/fusiello/Computer_Vision_Toolkit*). More specifically, it describes the markers detection and point tracking from the video recordings, then it shows the reconstruction of the markers 3D positions with triangulation and, finally, it shows the construction of the sequence of *rim-frames* used to generate the envelope enclosing plane oscillations of the *trim-panel*.

Keywords: videogrammetry; photogrammetry, triangulation; washing machine vibrations; rigid-body vibration envelope

1 Introduction

The design of both professional and domestic appliances is guided by two principal requirements: on the one hand the minimisation of production and operation costs of the devices and on the other hand the development of high-quality products, which guarantee superior performance and high comfort. This is indeed the case of washing machines, whose designs are constantly upgraded towards higher load capacities, which, nevertheless, require less housing space, involve less raw materials and implement fast washing cycles with low consumption of water, detergents and energy (both mechanical and thermal) [1]. Of particular interest is also a high rotational speed during extraction (even with significant unbalances), as this reflects in low residual moisture in the textiles and therefore in energy and time consumed in the subsequent drying process. These are rather challenging objectives, which entail quite advanced design studies. To this end, a new design approach is emerging in the industry where the classical work on paper and CAD software followed by laboratory tests on prototypes is progressively replaced by work on digital models of the system, which are commonly known as digital twins [2]–[6]. These are indeed digital replicas of the system, which, incorporate progressively refined mathematical models of the system and numerical platforms for the simulation of their responses under realistic operation conditions, which are acquired from data and measurements taken on running products. The availability of reference measurements is instrumental to have accurate and reliable input data for the digital twins.

This study is focused on a key feature of washing machines, that is the oscillations of the drum-suspension assembly during the washing cycles. The concurrent requirement for smaller housing layouts of the washing machines, with, nevertheless, higher garment loads, higher extraction speeds and higher load unbalance effects, poses quite demanding design challenges, particularly for the limitation of the mechanical vibrations of the drum-suspension assembly, which may lead to severe operation problems such as the generation of high structure borne noise transmission to the floor, the typical oscillatory walk phenomenon, and failure of components due to mechanical fatigue [7]–[13]. The dynamic response of the drum-suspension assembly is affected by quite a few phenomena and indeed requires both advanced models as well as detailed data on the real oscillatory motion of the drum assembly with respect to the outer housing. As an example, this paper proposes a non-invasive measurement approach based on optical cameras to reconstruct the surface area covered during spinning by the *trim-panel* where the *loading-door* is hinged. The objective is to provide indications on what should be the geometry of the opening in the *front-panel* of the *washing machine housing* to ensure there are no contacts with the oscillating *trim-panel*.

Single-camera or stereo-camera setups can be effectively used to reconstruct the oscillating motion of machines and structures [14]. In general, 2D or 3D Point Tracking (2DPT and 3DPT) approaches were developed to reconstruct via triangulation the motion of specific points marked with tiny patches bonded at precise points of the system at hand [14]–[16]. Alternatively, Digital Image Correlation (DIC) techniques have been developed too [14], [17]–[19], which reconstruct the displacements of the vibrating object by correlating the patterns of its surface in consecutive frames acquired by the cameras. More recently, advanced techniques based on multi-view (i.e. more than 2) techniques have also been proposed [20]–[22]. In this study, the 3DPT approach proposed in Ref. [23] is employed to detect the cross-plane oscillations of a professional washing machine *trim-panel*.

The paper is structured in three sections. Section 2 presents the measurement setup and the image acquisition and post-processing necessary to reconstruct the displacement of four target points highlighted by small circular patches bonded on the *trim-panel*. Then, Section 3 describes how the outer envelope generated by the plane displacements and rotations of the *trim-panel* is reconstructed from these measurements. Finally, Section 4 summarises the principal conclusions of the study. There is also an Appendix that recalls the code used to derive the envelope starting from image acquisitions, which is based on specific functions provided in Ref. [24].

2 Experimental rig and cameras measurement

2.1 Washing machine and cameras setup

Figure 1 shows the professional washing machine (a) and the cameras measurement setup (b) used in this study. As can be noticed in Figure 1a, the machine is characterised by a hollowed front panel, which is fixed to the machine housing. The tub is equipped with a *trim-panel* where the *loading-door* is hinged. To identify and reconstruct the displacements of the *trim-panel*, four small circular markers are glued onto the *trim-panel* itself, in such a way their geometrical centre of mass coincide with the centre of the *loading-door*. Two types of measurements were performed using *GoPro Hero 10* cameras, having spatial resolution of 2704×1520 *px* and temporal resolution of 240 *fps*. A centre camera was used separately to implement 2DPT measurements. Alternatively, the other two cameras were employed to perform 3DPT measurements. The centre camera was located in front of the washing machine in such a way as its optical axis coincides with the principal axis of rotation of the drum. The pair of cameras were also located in front of the panel, but their optical axes were tilted symmetrically with respect to the drum principal axis in such a way they converge symmetrically to the centre of the loading door with an opening angle of about 120 *deg* and an elevation angle of about

30 *deg*. Both the single and pair of cameras were located at about 0.7 m away from the centre of the loading door. To avoid flickering and to guarantee good quality measurements, the front panel was illuminated with a continuous and intense light source. The tests were taken with a fixed load formed by a 0.3 kg bag of sand stuck to the rim of the tub, which was not filled with water. The measurements were taken during a test cycle where the angular speed of the rim was gradually accelerated up to the spinning velocity. For brevity only the 3DPT results are reported in this paper.

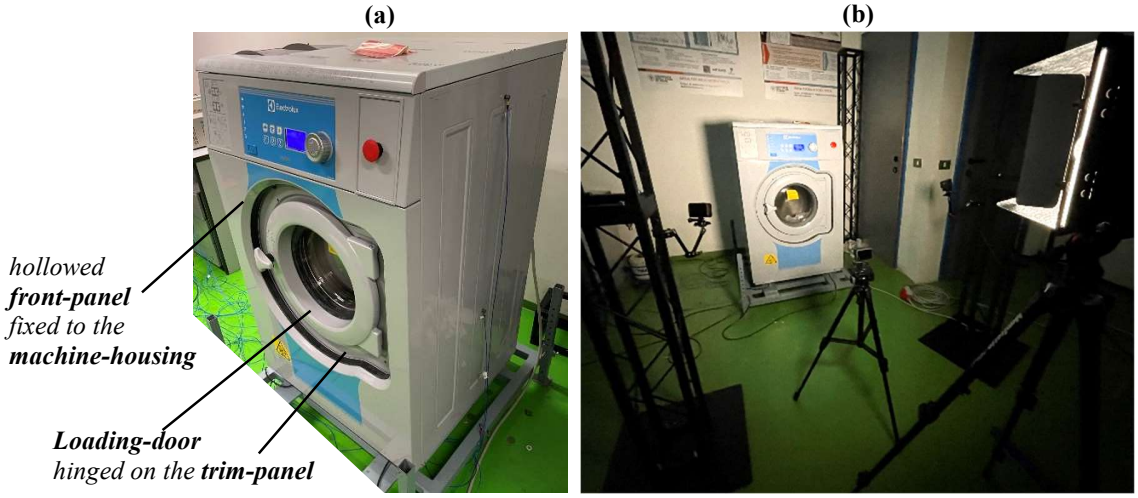


Fig. 1: Washing machine (a) and cameras setup (b).

2.2 Cameras acquisitions and image processing to reconstruct the markers positions

The implementation of the cameras measurement was organised in three phases: first the calibration of the cameras, second the acquisition of the videos and third the post processing of the recordings in such a way as to identify the markers positions in each image and then to reconstruct via triangulation the 3D coordinates of the centre positions of the markers. Overall, the following sequence of tasks were implemented as outlined in [25],[26] and therein references.

1. **Cameras calibration** – Beforehand, each camera was calibrated using the so-called Sturm-Maybank-Zhang method [27], [28], which included the compensation of the intrinsic radial distortion generated by the optics of the cameras.
2. **Video acquisitions** – Synchronized videos were then taken with the angular speed of the drum gradually accelerated from 0 to the maximum spinning velocity.
3. **Frames extraction** – Subsequently, two coherent sequences of frames were extracted from the videos recorded by the cameras (see Figure 2a).
4. **Perspective rectification** – Moreover a 2D transformation H_i, called *homography*, was applied to each image to remove perspective effects (see Figure 2b).
5. **Tentative markers detection** – At this point, a coarse identification of the markers centre coordinates was implemented on the rectified frames with a classical template matching technique, also known as cross-correlation technique.
6. **Markers validation** – Here, to avoid errors may arise from fictitious marker-like spots in the images, the Hungarian method [29] was adopted to check the markers tentative coordinates against their reference coordinates.

7. **Refined markers detection** – Hence, a sub-pixel refinement of the markers centre positions was implemented by fitting a parabola to the point of maximum correlation and to its two-neighbour points along the image axes (see Figure 2c).
8. **Markers image coordinates** – Finally, the centre coordinates of the markers were brought back to the original image space by applying the inverse *homography* 2D transformation H_i. The output of this stage was thus the 2D image coordinates in pixels of the centres of the four markers bonded onto the front panel of the washing machine.
9. **Markers physical coordinates** – Finally, the actual physical coordinates of the markers centre positions were reconstructed from triangulation using the so-called *Bundle Adjustment* (BA) procedure [30].

In general, for a given frame instant t_k, a given camera *i* and a given marker *j*, the physical 3D mm-coordinates of the marker $\mathbf{M}_j(t_k) = [X_j(t_k) \quad Y_j(t_k) \quad Z_j(t_k)]^T$ are linked to its image 2D pixel-coordinates on the *i*-th camera $\mathbf{m}_j^i(t_k) = [u_j(t_k) \quad v_j(t_k)]^T$ by the perspective projection equation:

$$\mathbf{m}_j^i(t_k) = f\left(\mathbf{M}_j(t_k), \mathbf{g}_j\right), \tag{1}$$

where the vector $\mathbf{g}_j$ contains the extrinsic parameters of the *i*-th camera. The BA simultaneously recovers the exterior orientation of the two cameras and the markers centre coordinates by solving a non-linear least squares problem, which turns into minimising the cost function

$$\sum_{i,j} \left\| \mathbf{m}_j^i(t_k) - f\left(\mathbf{M}_j(t_k), \mathbf{g}_j\right) \right\|_2^2, \tag{2}$$

with reference to both $\mathbf{M}_j(t_k)$ and $\mathbf{g}_j$. The BA computes only one set of unknowns when the other is given, and thus it will be referred as a partial BA. The minimisation of this cost function is carried out iteratively with the Levenberg–Marquardt algorithm. To initialise the exterior orientation of the cameras, six separate resections were implemented with the Direct Linear Transform algorithm (DLT) [30,31] using both the image coordinates of the markers detected in a rest frame $\mathbf{m}_j^i(t_0)$ acquired with no motion of the drum-assembly and the nominal rest coordinates of the markers $\mathbf{M}_j(t_k)$. The actual computation of the markers 3D centre coordinates $\mathbf{M}_j(t_k)$ at each instant t_k was implemented in the final BA, which comprises all the frames of all the cameras, with the exterior orientation of the cameras too considered among the unknowns. The initialisation exploited the nominal rest coordinates of the centres of the markers and the exterior orientation derived in the previous step.

The camera calibration, image processing and triangulation post-processing tasks described above were all implemented in MatLab using routines from Ref. [24].

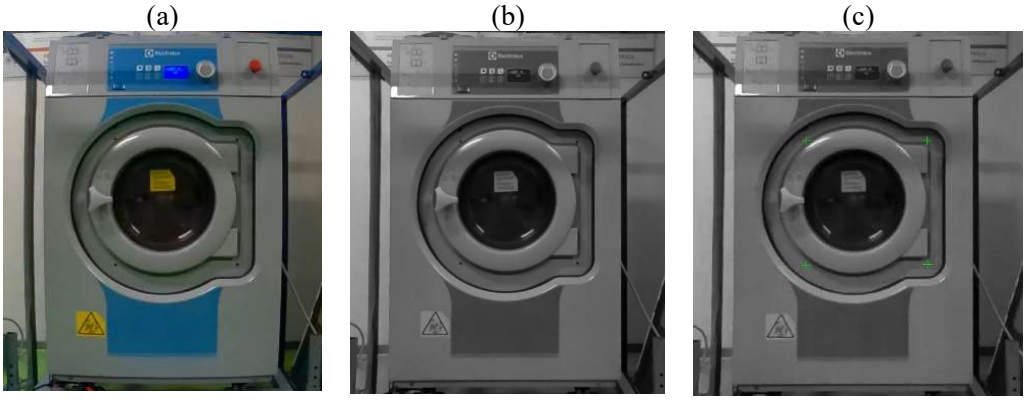


Fig. 2: Marker detection process: a) distorted frame, b) undistorted frame, c) marker detected (green crosses).

3 Envelope reconstruction

The reconstruction of the envelope-surface filled by the cross-plane linear and angular displacements of the *trim-panel* is now discussed. To this end two tasks were implemented. First, the cross-plane displacements of the four markers bonded on the trim-panel were reconstructed from the time-histories of their coordinates, which, as discussed above, were derived from triangulation of the images acquired by the cameras. Then, the linear (i.e. horizontal and vertical) and angular (about the spinning axis) displacements of the *trim-panel* were reconstructed with respect to the centre of motion of the panel itself. Second, solid-body geometry was employed to generate from the linear and angular displacements of the *trim-panel* a sequence of *rim-frames* that depict the border of the panel at at each instant t_k of the measurement.

3.1 Displacements of the centre of motion of the trim-panel

Figure 3 shows the displacements in horizontal and vertical direction of the trim-panel reconstructed from the cross-plane displacements of the four markers bonded on the trim-panel. The latter were retrieved directly from the horizontal and vertical coordinates of the markers generated by the triangulation of the images acquired with the cameras. The graphs show that the oscillations of the *trim-panel*, and thus of the whole drum assembly, tend to rise as the drum angular speed builds up from zero. In fact, they are significantly amplified when the angular speed passes by the resonance frequencies of the fundamental natural modes of the drum-suspension assembly, which are characterised by large vertical oscillations and large horizontal oscillations respectively. For this reason, they are normally specified as "horizontal-mode" and "vertical-mode", although both modes have a small vibration component is the other cross direction. At rather large angular speeds, the oscillations level to a constant value dictated by the specific unbalance [32]. Comparing the time-histories in Plots (b) and (c), it is noticed that the resonance amplification effect occurs before, that is at lower angular speeds or frequencies, for the horizontal displacement than the vertical displacement. Therefore, it is expected that the low-frequencies oscillations of the drum-mounts assembly are characterised by a lower-frequency "horizontal-mode" and a higher-frequency "vertical-mode".

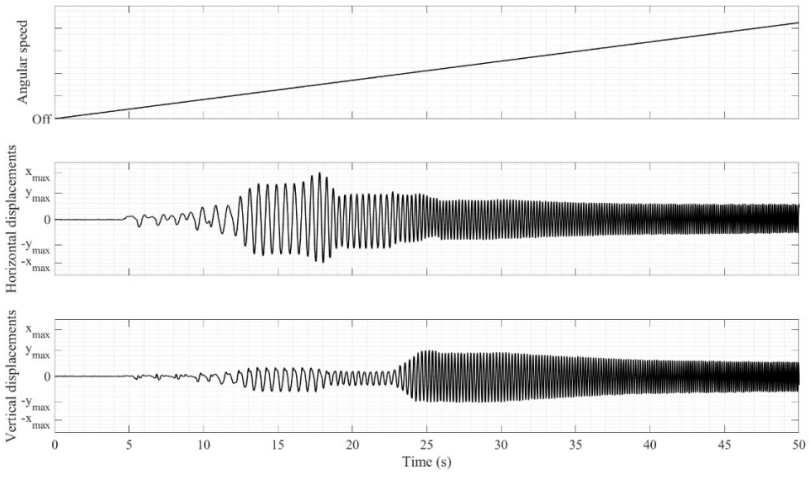


Fig. 3: Trim-panel oscillations: a) angular speed of the drum, b) horizontal displacement, c) vertical displacement.

3.2 Envelope reconstruction of the trim-panel rim

Once the linear and angular plane displacements of the trim-panel with respect to its centre of motion were reconstructed from the camera measurements, the actual area occupied by the trim panel at each time sample t_k was derived straightforwardly from simple considerations on the linear and angular rigid-body motion of the trim-panel border with respect to its centre of motion. The geometry of the border was acquired from CAD design drawings. For simplicity a polyline loop was built with quite a dense number of points. Hence, as depicted in Figure 4, a sequence of *rim-frames* was assembled considering the area enclosed by the polyline loop that define the trim-panel border at each instant t_k. This figure reports a subset of 24 frames only, which, nevertheless, highlight the typical motion of the trim-panel, that is of the drum assembly, which is characterised by a circular trajectory of the panel centre of motion accompanied by small plane rotations of the panel (i.e. drum assembly).

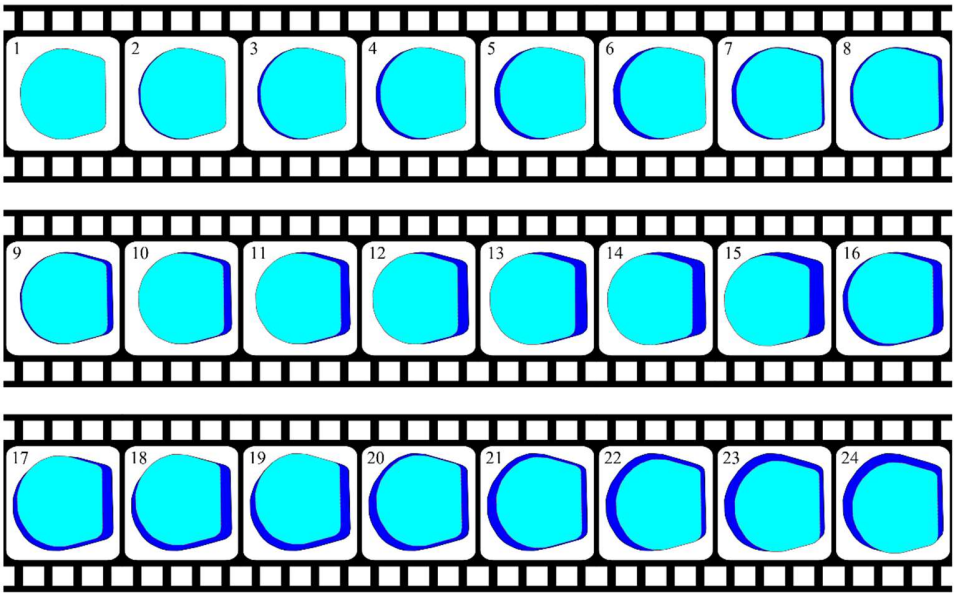


Fig. 4: Sequence of 24 rim-frames showing the positions of the trim-panel during spinning (light cyan areas) and the whole area occupied by trim panel after the spinning has been completed.

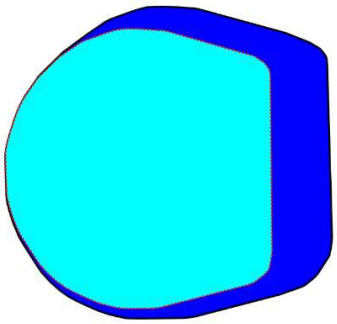


Fig. 5: Trim-panel envelope throughout the entire spinning phase

At this point the overall envelope space occupied by the oscillations of the trim-panel during spinning was derived straightforwardly form the superposition of the areas defined by the whole sequence of rim-frames. The overall result is the envelope surface depicted in blue in Figure 5. To better highlight the motion of the trim panel, the results presented in Figures 4 and 5 were slightly magnified.

As anticipated above the image processing, triangulation and envelope reconstruction were carried out using MatLab functions from the open-source Computer Vision Toolkit (*https://github.com/fusiello/Computer_Vision_Toolkit*). Also, the code developed for the generation of the envelope is recalled in the Appendix.

4 Conclusions

This paper has reported a case study on the reconstruction from camera measurements of the envelope enclosing the space filled in by the cross-plane oscillations of the front trim-panel of a professional washing machine. The study has shown that the plane rigid body motion of the panel defined by horizontal and vertical displacement and cross plane rotations, can be suitably reconstructed from triangulation of video recordings taken with two cameras positioned in front of the panel. Based on simple solid-geometry considerations, a sequence of rim-frames depicting the area occupied by the trim panel at each time-sample can then be straightforwardly assembled. Then the overall envelope area occupied by the motion of the trim-panel during spinning can be generated by superimposition of the rim-frames.

Although this paper has discussed quite a simple case study, in general the proposed approach can be conveniently employed to derive the motion-envelope of the whole drum assembly with the machine in its operation configuration. Thus, there is no need of having direct access to the interior of the machine housing.

Appendix A. MatLab codes for the reconstruction of the drum oscillations envelope

This appendix reports the MatLab codes used to define the drum oscillations envelope when the washing machine is running a certain cycle. The centroid translations and drum rotations are easily calculated once all the markers positions are known. The marker detection routine can be found in Computer Vision Toolbox [reference]

```
% Define the drum oscillations envelope during a cycle, knowing the position of 4
% markers detected using the Computer Vision Toolbox.
%
% Input:
%
% drum_points: the matrix containing the x-y coordinates of the drum borders
% translation: matrix containing the x-y translations of the markers centroid
% rotation: vector containing the rotations [rad] of the drum around the marker's
% centroid
%
% Output:
% env_points: matrix containing the x-y coordinates of the envelope borders
%
% Initialisation step :
%
```

```
% The drum border points are transformed into a polygon by polyshape

drum=polyshape(drum_points (1,:), drum_points (2,:));

% Initialization of the envelope border points: at the beginning, they coincide with
% the drum border points
envelope=drum.Vertices;

% Envelope border points transformed into a polygon
envpol=polyshape(envelope(:,1),envelope(:,2));

% Envelope formation for loop: at each frame of the video, the corresponding
% translation and rotation are imposed on the drum polygon. The new drum border
% points are calculated.
for i=1:length(translation(1,:))
    drum=translate(drum,translation(1,i),translation(2,i));
    drum=rotate(drum,180/pi*rotation(i));
    drum_points=drum.Vertices;

% It is now checked if the new positions of the drum border points are inside or
% outside the polygon of the previous envelope. If they are outside, the envelope
% need to be enlarged and the outside points will be added to the envelope
% border points
    in=inpolygon(drum_points(:,1),drum_points(:,2),envelope(:,1),envelope(:,2));
    envelope=[envelope;drum_points(~in,:)];

% To avoid bad definitions of the envelope polygon, the envelope border points
% (matrix "envelope") must be rearranged because of the added new points.
% The proposed solution is to sort the points by the angle between the joining
% line among the centroid and the point considered and the horizontal lined passing
% through the centroid.
    point_angle=NaN(length(envelope(:,1)),1);
    for j=1:length(envelope(:,1))
        point_angle(j)=atan2(envelope(j,2),envelope(j,1));
    end
    [point_angleord,iangle]=sort(point_angle);
    xenvdis=envelope(:,1)';
    yenvdis=envelope(:,2)';
    envelope=[xenvdis(iangle)' yenvdis(iangle)'];
end

% The final step is to force the polygon to be convex.
envpol=polyshape(envelope(:,1),envelope(:,2));
envpol=convhull(envpol);
envelope=envpol.Vertices;  % final x-y envelope border points
```

References

[1] I. Johansson and P. Somasundaran, *Handbook for Cleaning/Decontamination of Surfaces*. 2007.

[2] S. Haag and R. Anderl, "Digital twin – Proof of concept," *Manuf Lett*, vol. 15, pp. 64–66, Jan. 2018, doi: 10.1016/J.MFGLET.2018.02.006.

[3] D. Jones, C. Snider, A. Nassehi, J. Yon, and B. Hicks, “Characterising the Digital Twin: A systematic literature review,” *CIRP J Manuf Sci Technol*, vol. 29, pp. 36–52, May 2020, doi: 10.1016/J.CIRPJ.2020.02.002.

[4] K. Worden, E. J. Cross, R. J. Barthorpe, D. J. Wagg, and P. Gardner, “On Digital Twins, Mirrors, and Virtualizations: Frameworks for Model Verification and Validation,” *ASCE-ASME J Risk and Uncert in Engrg Sys Part B Mech Engrg*, vol. 6, no. 3, May 2020, doi: 10.1115/1.4046740.

[5] D. J. Wagg, K. Worden, R. J. Barthorpe, and P. Gardner, “Digital Twins: State-of-the-Art and Future Directions for Modeling and Simulation in Engineering Dynamics Applications,” *ASCE-ASME J Risk and Uncert in Engrg Sys Part B Mech Engrg*, vol. 6, no. 3, May 2020, doi: 10.1115/1.4046739.

[6] J. Autiosalo, J. Vepsäläinen, R. Viitala, and K. Tammi, “A Feature-Based Framework for Structuring Industrial Digital Twins,” *IEEE Access*, vol. 8, pp. 1193–1208, 2020.

[7] D. C. Conrad and W. Soedel, “ON THE PROBLEM OF OSCILLATORY WALK OF AUTOMATIC WASHING MACHINES,” *J Sound Vib*, vol. 188, pp. 301–314, 1995.

[8] O. S. Türkay, B. Kiray, A. K. Tugcu, and İ. T. Sümer, “Formulation and implementation of parametric optimisation of a washing machine suspension system,” *Mech Syst Signal Process*, vol. 9, no. 4, pp. 359–377, 1995, doi: https://doi.org/10.1006/mssp.1995.0029.

[9] M. J. Chrzan and J. D. Carlson, “MR fluid sponge devices and their use in vibration control of washing machines,” in *Proc.SPIE*, Jul. 2001, vol. 4331, pp. 370–378. doi: 10.1117/12.432719.

[10] C. Spelta, F. Previdi, S. M. Savaresi, G. Fraternale, and N. Gaudiano, “Control of magnetorheological dampers for vibration reduction in a washing machine,” *Mechatronics*, vol. 19, pp. 410–421, 2009.

[11] H.-T. Lim, W.-B. Jeong, and K.-J. Kim, “Dynamic modeling and analysis of drum-type washing machine,” *International Journal of Precision Engineering and Manufacturing*, vol. 11, no. 3, pp. 407–417, 2010, doi: 10.1007/s12541-010-0047-7.

[12] Q. H. Nguyen, S. B. Choi, and J. K. Woo, “Optimal design of magnetorheological fluid-based dampers for front-loaded washing machines,” *Proc Inst Mech Eng C J Mech Eng Sci*, vol. 228, no. 2, pp. 294–306, Apr. 2013, doi: 10.1177/0954406213485908.

[13] J. Buśkiewicz and G. Pittner, “Reduction in vibration of a washing machine by means of a disengaging damper,” *Mechatronics*, vol. 33, pp. 121–135, 2016.

[14] J. Baqersad, P. Poozesh, C. Niezrecki, and P. Avitabile, “Photogrammetry and optical methods in structural dynamics – A review,” *Mech Syst Signal Process*, vol. 86, pp. 17–34, 2017, doi: https://doi.org/10.1016/j.ymssp.2016.02.011.

[15] T. Ryall and C. Fraser, “Determination of Structural Modes of Vibration Using Digital Photogrammetry,” *AIAA Journal of Aircraft*, vol. 39, pp. 114–119, Jan. 2002, doi: 10.2514/2.2903.

[16] S. W. Park, H. S. Park, J.-H. Kim, and H. Adeli, “3D displacement measurement model for health monitoring of structures using a motion capture system,” *Measurement*, vol. 59, pp. 352–362, Jan. 2015, doi: 10.1016/j.measurement.2014.09.063.

[17] M. N. Helfrick, C. Niezrecki, P. Avitabile, and T. Schmidt, “3D digital image correlation methods for full-field vibration measurement,” *Mech Syst Signal Process*, vol. 25, no. 3, pp. 917–927, Apr. 2011, doi: 10.1016/j.ymssp.2010.08.013.

[18] T. J. Beberniss and D. A. Ehrhardt, “High-speed 3D digital image correlation vibration measurement: Recent advancements and noted limitations,” *Mech Syst Signal Process*, vol. 86, pp. 35–48, 2017, doi: https://doi.org/10.1016/j.ymssp.2016.04.014.

[19] P. L. Reu, D. P. Rohe, and L. D. Jacobs, “Comparison of DIC and LDV for practical vibration and modal measurements,” *Mech Syst Signal Process*, vol. 86, pp. 2–16, 2017, doi: https://doi.org/10.1016/j.ymssp.2016.02.006.

[20] D. Gorjup, J. Slavič, and M. Boltežar, “Frequency domain triangulation for full-field 3D operating-deflection-shape identification,” *Mech Syst Signal Process*, vol. 133, Nov. 2019, doi: 10.1016/j.ymssp.2019.106287.

[21] D. Gorjup, J. Slavič, A. Babnik, and M. Boltežar, “Still-camera multiview Spectral Optical Flow Imaging for 3D operating-deflection-shape identification,” *Mech Syst Signal Process*, vol. 152, May 2021, doi: 10.1016/j.ymssp.2020.107456.

[22] S. Barone, P. Neri, A. Paoli, and A. V. Razionale, “Low-frame-rate single camera system for 3D full-field high-frequency vibration measurements,” *Mech Syst Signal Process*, vol. 123, pp. 143–152, May 2019, doi: 10.1016/j.ymssp.2019.01.016.

[23] R. Del Sal *et al.*, “Structural vibration measurement with multiple synchronous cameras,” *Mech Syst Signal Process*, vol. 157, p. 107742, Aug. 2021, doi: 10.1016/j.ymssp.2021.107742.

[24] A. Fusiello, “Computer Vision Toolbox for Matlab,” *https://github.com/fusiello/Computer_Vision_Toolkit*, 2022.

[25] P. Gardonio, R. Rinaldo, L. Dal Bo, R. Del Sal, E. Turco, and A. Fusiello, “Free-field sound radiation measurement with multiple synchronous cameras,” *Measurement*, vol. 188, p. 110605, 2022, doi: https://doi.org/10.1016/j.measurement.2021.110605.

[26] P. Gardonio, G. Guernieri, E. Turco, L. Dal Bo, R. Rinaldo, A. Fusiello. “Reconstruction of the sound radiation field from flexural vibrations measurements with multiple cameras,” *Mechanical Systems and Signal Processing*, vol. 195, p. 110289, 2023 doi: 10.1016/j.ymssp.2023.110289.

[27] P. F. Sturm and S. J. Maybank, “On plane-based camera calibration: A general algorithm, singularities, applications,” in *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*, 1999, vol. 1, pp. 432-437 Vol. 1. doi: 10.1109/CVPR.1999.786974.

[28] Z. Zhang, “A flexible new technique for camera calibration,” *IEEE Trans Pattern Anal Mach Intell*, vol. 22, no. 11, pp. 1330–1334, 2000, doi: 10.1109/34.888718.

[29] H. W. Kuhn, “The Hungarian method for the assignment problem,” *Naval Research Logistics Quarterly*, vol. 2, no. 1–2, pp. 83–97, Mar. 1955, doi: https://doi.org/10.1002/nav.3800020109.

[30] R. Hartley, A. Zisserman, Multiple View Geometry in Computer Vision, Cambridge University Press, 2004.

[31] I. Sutherland, Three-dimensional data input by tablet, Proc. IEEE 62 (4) (1974) 453–461, http://dx.doi.org/10.1109/PROC.1974.9449

[32] L. Dal Bo, P. Gardonio, N. Battistella, E. Turco “Non-linear Isolator for Vibration and Force Transmission Control of Unbalanced Rotating Machines,” *Journal of Vibration Engineering and Technologies*, 2022, doi: https://doi.org/10.1007/s42417-022-00668-9

Nonlinear vibration analysis with NLVIB

Patrick Hippold * Johann Groß Malte Krack

University of Stuttgart

Abstract

NLVIB is a free MATLAB tool for nonlinear vibration analysis. Periodic solutions of nonlinear mechanical systems can be computed in frequency domain by applying Harmonic Balance. This allows for a computationally efficient treatment of nonlinear problems. Nonlinear forces are evaluated with the Alternating Frequency-Time scheme which works for smooth and non-smooth force laws. As an alternative to Harmonic Balance, solutions can be obtained in time domain with a Newmark integrator and the shooting method. The latter natively provides information about the stability of the identified orbits. By applying numerical path continuation, solution branches can be continued under the variation of a parameter. To this end, a predictor-corrector scheme with a Newton type solver is used. To accelerate the convergence, the numerical conditioning of the problem is optimized through a scaling approach. Continuation with respect to the excitation frequency results in a frequency response analysis. If the continuation parameter is the response amplitude, a nonlinear modal analysis is conducted which yields the nonlinear modal frequency and damping ratio. Mechanical systems with multiple degrees-of-freedom that combine different types of nonlinearities can be treated with NLVIB. This includes generic local nonlinearities (e. g., unilateral springs and friction elements) as well as distributed polynomial stiffness nonlinearities. Multiple examples demonstrate the possible analyses on a variety of mechanical systems, covering a range from a Duffing oscillator to a Finite Element model of a beam with different nonlinear attachments.

Keywords: nonlinear vibrations, harmonic balance, numerical continuation, frequency response, nonlinear modal analysis

1 Introduction

Vibration-induced stress can cause failure of mechanical structures. Hence, precise tools for vibration prediction are needed during the development of novel structures. Especially for lightweight structures in modern aerospace applications, deformations can easily excess the range where linear behavior of the structure is a valid assumption. Thus, analysis methods that account for the nonlinearity are needed. A standard method to solve the nonlinear differential equations describing the vibration problem is numerical integration. Starting from a given set of initial values, the time evolution of the dependent variables is in this case determined successively from one time level to the next (numerical forward integration) [1]. However, for many engineering applications, only a periodic steady state is of interest. Reaching this periodic limit state with time integration from arbitrary initial conditions requires the simulation of a transient. If the transient decays only slowly, as in the case of lightly

*Corresponding author, Email address: patrick.hippold@ila.uni-stuttgart.de

damped vibrations, long simulation times are required to determine the periodic limit state in this way. The time discretization inherently introduces approximation errors including numerical instability, numerical damping, and nonphysical energy redistribution within the system. Moreover, non-smooth nonlinear terms in the differential equations can lead to jump discontinuities of the solution [1].

Furthermore, many applications require knowledge not only of a single periodic state but on how the system's response changes under the variation of a parameter. A typical example is the computation of a frequency response curve (FRC) which shows the response amplitude versus the excitation frequency for a system under harmonic external forcing. It is well known that FRCs of nonlinear systems may exhibit complex features such as co-existing periodic responses for a single excitation frequency. Therefore, purely solving the problem for different frequencies is not sufficient to obtain a complete picture.

2 Methodology

Different methods have been developed in the past to address the challenges named above. The Harmonic Balance (HB) method in conjunction with numerical path continuation is one way to analyze a nonlinear mechanical system. Both components are implemented in the MATLAB tool NLVIB and are briefly described in the following.

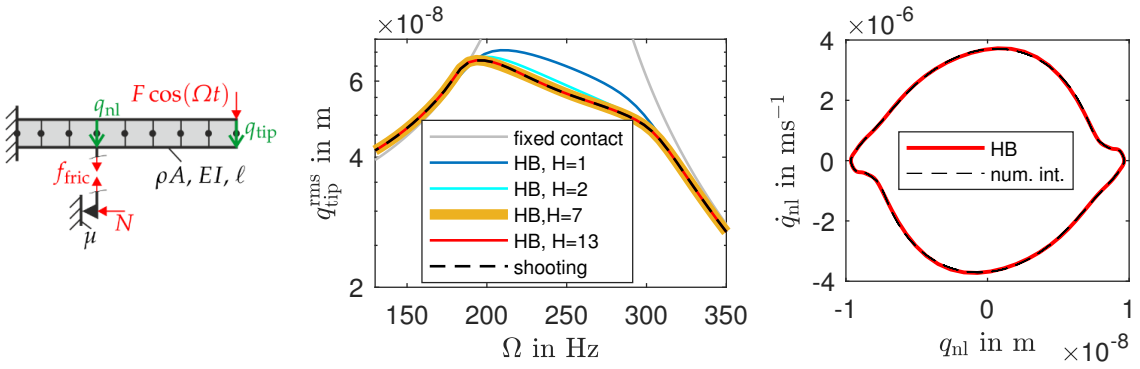
2.1 Harmonic balance

The idea of HB is summarized in [1] as the following: A regular periodic solution of an ordinary or differential-algebraic equation system can be represented by a Fourier series, i. e., a combination of sinusoids. In many cases, a reasonably accurate approximation is already achieved when only a small number of sinusoids is considered. Thus, a natural approach seems to seek an approximate solution in the form of a truncated Fourier series. Substituting this ansatz into the differential equation system generally leads to a residual term. As the approximate solution is periodic, so is the residual term. Hence it can also be expanded in a Fourier series. HB now requires that the Fourier coefficients of the residual term vanish, up to the truncation order of the ansatz. This yields an algebraic equation system with respect to the unknown Fourier coefficients of the approximation.

A major challenge when applying HB is the determination of the Fourier coefficients of the nonlinear forces. In NLVIB, the Alternating Frequency-Time scheme is used which can handle generic nonlinearities [2, 3]. Samples of the displacements and velocities are taken at equidistant time instants within one period of oscillation (or more, in case of hysteretic nonlinearities). Evaluation of the nonlinear force law at these time instants gives samples of the nonlinear force. Finally, the discrete Fourier transform yields (in general an approximation of) the Fourier coefficients of the nonlinear forces [1].

2.2 Numerical path continuation

As mentioned above, common analysis tasks require the computation of multiple points on a solution branch. To this end, NLVIB features numerical path continuation to offer two analysis types. First, FRCs can be computed by assuming a harmonic external forcing and performing continuation with the excitation frequency as continuation parameter. Second, a nonlinear modal analysis using the periodic motion definition of nonlinear modes in its extended form for dissipative systems [4] can be conducted by performing continuation with (the logartithm of) an amplitude quantity as continuation parameter.



(a) **(b)** **(c)**

Fig. 1: Mechanical system used in the example (a) and results of the numerical vibration analysis: frequency response curve (b) and phase portrait for the attachment point of the nonlinearity at maximum tip amplitude (c); length $l = 2\,\mathrm{m}$, linear mass density $\rho A = 235\,\mathrm{kg\,m^{-1}}$, flexural rigidity $EI = 4.6 \times 10^6\,\mathrm{N\,m^2}$, limit force $\mu N = 1.5\,\mathrm{N}$, excitation force $F = 0.2\,\mathrm{N}$, and regularization parameter $\varepsilon = 6.2 \times 10^{-7}$.

The continuation algorithm is a predictor-corrector scheme [5] with tangent or secant predictors and a Newton-type solver as corrector (MATLAB's `fsolve`). Different parametrization methods, e. g., *arclength* and *local*, are implemented. The step size is automatically adjusted based on the number of iterations required in the corrector step. Moreover, scaling is applied to accelerate the convergence by improving the conditioning of the numerical problem.

3 Nonlinear mechanical systems and application example

NLVIB is designed to handle mechanical equations-of-motion with a finite number of generalized coordinates. Common types of nonlinearities are pre-defined. Most of them are *local*, i. e., each element depends on a single input displacement (and velocity) and delivers a single force to the system [1]. Examples are unilateral springs and friction elements. An alternative is a distributed polynomial stiffness nonlinearity. Moreover, adding own types of nonlinearities is possible.

One example that comes with the tool is the analysis of a Finite Element model of an Euler–Bernoulli beam with clamped-free boundary conditions and a dry friction element as depicted in Fig. 1a [1]. The model consists of eight beam elements. Dry friction is here modeled using the Coulomb law with a tanh-regularization of the signum-function:

$$f_{\mathrm{fric}}(q_{\mathrm{nl}}, \dot{q}_{\mathrm{nl}}) = \mu N \tanh \frac{\dot{q}_{\mathrm{nl}}}{\varepsilon} \tag{1}$$

wherein q_{nl} and $\dot{q}_{\mathrm{nl}}$ are lateral displacement and velocity at the node where the nonlinear element is attached, respectively. Harmonic forcing is applied at the beam's tip, the corresponding displacement is called q_{tip}.

The amplitude-frequency curves are illustrated in Fig. 1b. The near-resonant vibration response is considerably reduced, in a way typical for friction damping. It can be seen that the tip displacement level does not significantly change once the harmonic truncation order H is sufficiently large. In this

case, the amplitude-frequency curves for harmonic truncation orders larger or equal to $H = 7$ cannot be distinguished "by eye" in the plot and agree well with the results of the shooting method [1].

The good agreement between numerical integration and HB with $H = 13$ can also be confirmed from the phase projection into the q_{nl}-$\dot{q}_{\mathrm{nl}}$-plane in Fig. 1c. Note that the phase projection clearly deviates from an ellipse, which indicates the presence of higher harmonics due to nonlinear forces [1]. While this single solution by time integration took about 21.4 s to compute, the computation of the whole FRC with HB only took 4.6 s on the same hardware. This highlights the advantage of HB over time integration when periodic solutions are sought.

4 Discussion and Conclusions

HB as implemented in NLVIB is only useful for periodic oscillations. However, extensions of HB to quasi-periodic oscillations exist (see, e. g., [6]). A drawback of HB for non-smooth nonlinearities is the occurence of the Gibbs phenomenon due to harmonic base functions [1]. Active topics of research include the generic assessment of branching behavior and the stability assessment for solutions found through HB. To address the latter, shooting methods provide an alternative to HB because the stability of the solution is automatically determined during the procedure. A shooting method with an unconditionally stable Newmark integrator is also part of NLVIB.

As a conclusion, HB can be beneficial for different problems in nonlinear vibration analysis. If the goal is to identify periodic solutions, it can be computationally much more efficient than time integration. The MATLAB tool NLVIB includes HB, the shooting method as an alternative, and numerical path continuation perform nonlinear frequency response analyses or nonlinear modal analyses. The tool is designed as a reasonable compromise between simplicity and a broad portfolio of capabilities. The code for NLVIB, examples, documentation, and licensing information are available via `www.ila.uni-stuttgart.de/nlvib`.

References

[1] M. Krack and J. Gross, *Harmonic Balance for Nonlinear Vibration Problems*. Cham: Springer International Publishing, 2019.

[2] T. M. Cameron and J. H. Griffin, "An alternating frequency/time domain method for calculating the steady-state response of nonlinear dynamic systems," *Journal of Applied Mechanics*, vol. 56, no. 1, pp. 149–154, 1989.

[3] A. Cardona, A. Lerusse, and M. Géradin, "Fast fourier nonlinear vibration analysis," *Computational Mechanics*, vol. 22, no. 2, pp. 128–142, 1998.

[4] M. Krack, "Nonlinear modal analysis of nonconservative systems: Extension of the periodic motion concept," *Computers & Structures*, vol. 154, pp. 59–71, 2015.

[5] R. Seydel, *Practical bifurcation and stability analysis: From equilibrium to chaos*, vol. 5 of *Interdisciplinary applied mathematics*. New York and Berlin: Springer, 2. ed. ed., 1994.

[6] F. Schilder, W. Vogt, S. Schreiber, and H. M. Osinga, "Fourier methods for quasi-periodic oscillations," *International Journal for Numerical Methods in Engineering*, vol. 67, no. 5, pp. 629–671, 2006.

A visual line tracking technique overcoming various noisy backgrounds

Tengjiao Jiang* Gunnstein Thomas Frøseth Anders Rønnquist

Norwegian University of Science and Technology, Department of Structural Engineering, Rich. Birkelands vei 1A, 7491 Trondheim, Norway

Abstract

An image-processing technique is proposed to solve the challenge of tracking linear objects in front of noisy backgrounds. The primary architecture includes coarse search and subpixel detection. The coarse search aims for a quick location of linear objects, and subpixel detection is responsible for accurate position. Railway catenary systems are used as an engineering application to measure the displacement response during train passage. The accuracy, robustness and applicability are demonstrated by comparing the laser displacement meter in field tests. The results show that the technique can accurately identify and track wire motion in various noisy backgrounds. It offers the attractive advantages of remote, non-contact and non-marker measurement, easy installation and application without track access. The open-source code can be downloaded from http://doi.org/10.5281/zenodo.3685219.
Keywords: line tracking; photogrammetry; image processing; displacement measurement, open source.

1 Introduction

Pantograph-catenary system is one of the critical components of electrified railways, which is responsible for ensuring a stable and continuous power supply for trains [1]. The catenary system is a wire system, and its interaction with the pantograph will result in its uplift when a train pass. Any excessive uplift will increase the risk for either mechanical contact between components or loss of contact between the contact wire and the pantograph [2]. The latter is more common and will not only interrupt the train power supply but also generate arcs that will accelerate the wear of the contact wire and the collector strip [3]. The uplift measurement of existing contact wires is an important method to survey and assess the operational state of the system.

The traditional uplift measuring equipment usually requires installing markers on the catenary wires or fixing the measuring devices on either the support structure or temporary poles mounted for the measurement. These methods demand track access and manpower for installation and measurement. Thus, a portable vision-based tracking system with a novel line tracking method [4] is proposed and applied to enable low-cost, remote, non-contact and non-target uplift measurements of catenary systems. The camera system can be installed far from the railway track to perform measurements without any contact, i.e., no track access is required. The proposed line-tracking algorithm has addressed the challenge of tracking the catenary wires in front of a noisy background.

* Corresponding author, Email address: tj.jiang@outlook.com

Field tests at Oppdal railway station were carried out to test the accuracy, robustness and applicability of the tracking system.

2 A line tracking method

Generally, it is expected that there can be plants, other infrastructure objects or other disturbing things on either side of the railway line. Thus, backgrounds behind catenary systems are usually nonuniform or noisy, which makes it challenging to identify and track a moving wire object. Some image processing techniques (e.g., digital image correlation) were applied to track the catenary wires without markers in front of noisy backgrounds, but they cannot work very well due to the loss of correlation caused by the noisy backgrounds.
A novel line tracking technique [4] has been proposed to track wires without markers in front of noisy backgrounds. The open-source code has been published in [5]. The algorithm consists of coarse search and subpixel detection, as shown in Fig. 1. The purpose of the coarse search method is to quickly search the whole image and identify the approximate locations of the catenary wires from the noisy background. The subpixel detection is to accurately detect the subpixel location of these catenary wires by grayscale bi-cubic interpolation method.

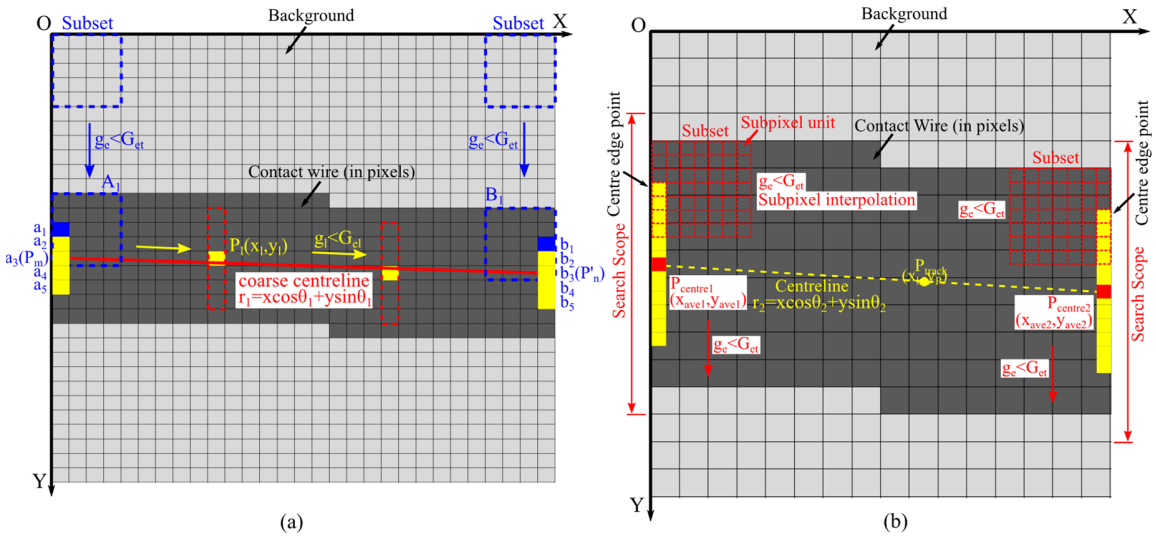


Fig. 1. A novel line tracking algorithm based on coarse search and subpixel detection. (a) coarse search method; (b) subpixel detection method.

3 Field tests

Field uplift measurement of the catenary wires at Oppdal railway station in Norway was carried out to test the performance of the proposed tracking system. A vision-based tracking system is developed to measure catenary wire displacement, consisting of a camera, a fixed-focal optical lens, a trigger, a laptop computer and a laser range finder, as shown in Fig. 2 (b). In the field test, the camera system was mounted at a secure distance from the railway track, and measurements were carried out without contact with the catenary system. A laser displacement meter was mounted on the support structure above the contact wire, and a reflective plate was installed on the contact wire to measure the uplift in an alternative way. A schematic overview of the uplift measurement is shown in Fig. 2 (a). The

catenary system was fixedly excited by a hit of a hammer, and the two measurement systems simultaneously obtained the uplift data at the same sampling frequency of 200 Hz.

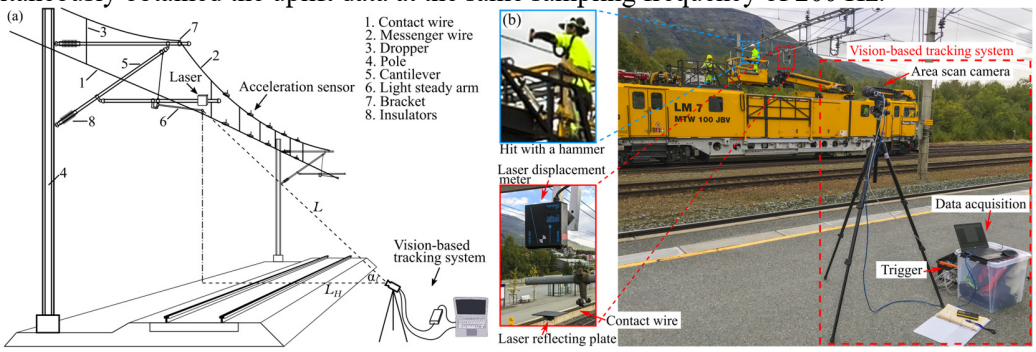


Fig. 2. Uplift measurement of the catenary wires by the vision-based tracking system and laser displacement meter.

4 Results

The robustness and applicability of the system were tested against the background of swaying trees to see if the catenary wires can be tracked without interruption. Fig. 3 (b) shows the identification result of the catenary wires by using the line tracking algorithm, and the centre red points are the selected tracking points. The accuracy of the tracking system is compared with the data of the laser displacement meter. The accuracy of the laser is approximately 0.05 mm for the measuring range 10 to 30 cm and is considered to be ground truth for this application. For the sake of clarity, Fig. 3 (c) shows the first 20 seconds of the displacement comparisons between the vision-based tracking system and the laser displacement meter. The displacement data of the proposed tracking system is very close to that of the laser displacement meter. After statistical analysis, the accuracy of the proposed vision-based tracking system is ± 0.6 mm at 95% confidence. The results demonstrated that the proposed vision-based tracking system, with the novel line tracking method, can successfully identify line objects from noisy backgrounds and accurately measure the displacement response.

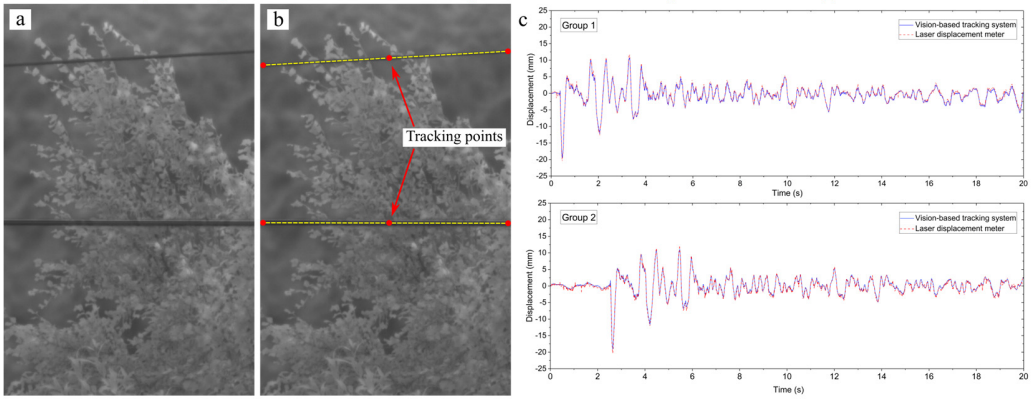


Fig. 3. (a) Original image of the catenary wires; (b) The identification results; (c) Displacement comparison between the vision-based tracking system and laser displacement meter.

5 Conclusion

A novel line-tracking technique is proposed to address the challenge of tracking linear objects without markers in front of noisy backgrounds. A portable measuring system is developed for a low-cost, remote, non-contact and non-marker uplift measurements of a railway catenary system.
The accuracy, robustness and applicability of the system are demonstrated through the field tests at Oppdal railway station in Norway. The statistical results show that the system can achieve an accuracy of ± 0.6 mm at 95% confidence with a measuring distance of 10 m.

6 Acknowledgements

The research presented in this article was financed by the Norwegian Railway Directorate.

References

[1] T. Jiang, “Development and Application of a Vision-Based System for Structural Monitoring of Railway Catenary System,” PhD Thesis, Norwegian University of Science and Technology, Trondheim, 2021. doi: https://hdl.handle.net/11250/2979682.

[2] T. Jiang, A. Rønnquist, Y. Song, G. T. Frøseth, and P. Nåvik, “A detailed investigation of uplift and damping of a railway catenary span in traffic using a vision-based line-tracking system,” *J Sound Vib*, vol. 527, 2022, doi: 10.1016/j.jsv.2022.116875.

[3] T. Jiang, G. T. Frøseth, P. Nåvik, and A. Rønnquist, “Assessment of pantograph-catenary interaction in a railway overlap section via a novel optical-based method,” *Mech Mach Theory*, vol. 177, p. 105045, Nov. 2022, doi: 10.1016/J.MECHMACHTHEORY.2022.105045.

[4] T. Jiang, G. T. Frøseth, A. Rønnquist, and E. Fagerholt, “A robust line-tracking photogrammetry method for uplift measurements of railway catenary systems in noisy backgrounds,” *Mech Syst Signal Process*, vol. 144, p. 106888, 2020.

[5] T. Jiang, G. T. Frøseth, A. Rønnquist, and E. Fagerholt, “A vision-based line-tracking technique,” Jun. 2020, doi: 10.5281/ZENODO.3779886.

System Equivalent Model Mixing with pyFBS: Accurately Estimating Dynamic Properties at Unmeasurable Locations

Miha Kodrič Tomaž Bregar Gregor Čepon * Miha Boltežar

Faculty of mechanical engineering, University of Ljubljana
Gorenje, Predevelopment - Numerical simulations and Acoustics

Abstract

In the field of modal analysis, the evaluation of dynamic properties is often limited by the number of measurable degrees of freedom. This limitation can be due to several factors, such as limited resources or physical accessibility. However, there are cases where NVH engineers require data from locations that cannot be directly measured for purposes such as structural optimization or substructuring processes. To address this challenge, a new method called System Equivalent Model Mixing (SEMM) has been developed. In this model-based approach, two equivalent models are used: one (typically numerical) with high spatial density but approximate properties, and one (typically experimental) with low spatial density but real dynamic properties. The SEMM method effectively combines these two models into a hybrid model that expands the real dynamic properties to the unmeasurable dense set of degrees of freedom of the first model. This technique has been implemented in an open-source Python package called pyFBS, making it easily accessible to NVH engineers. In this paper, we present a new approach for identifying inconsistent measurements in the frequency domain using SEMM. Traditional tools for identifying measurement consistency, such as FRAC, MAC, and CoMAC, only allow comparison with other data sets, typically from numerical models, which may not accurately reflect system behaviour. In contrast, the proposed approach uses the numerical model solely for SEMM expansion. The results show that by removing inconsistent frequency response functions (FRFs) from the experimental substructure model, the accuracy of the coupling process can be significantly improved. This highlights the effectiveness of the SEMM approach in improving the overall reliability and accuracy of the coupling process.

Keywords: System equivalent model mixing, pyFBS, Open Source, Structural Dynamics, Python

1 Introduction

Providing a consistent experimental response model is a crucial step for accurate dynamic analysis. The quality of the experimental model can be verified using comparative criteria like FRAC, MAC, CoMAC, etc. They are not able to identify the relationship between the measurements inside the experimental data set. These tools only allow a comparison with other data sets, usually obtained by a numerical model that might not reflect the behavior of the actual system. Recently, a new, general approach called the Data Consistency Assessment Function (DCAF) [1] was proposed that is able to evaluate the consistency of a set of measurements. The method is formulated in the modal domain using the SEREP expansion process and the MAC criterion. This paper presents a new approach to the identification of inconsistent

*Corresponding author, Email address: gregor.cepon@fs.uni-lj.si

measurements. The basic idea is similar to the DCAF method, however, here the entire formulation is developed within the frequency domain using System Equivalent Model Mixing (SEMM) method [2] and coherence criteria. To present the capability of the introduced method, a coupling of two simple beam-like structures is presented. It will be shown that by removing the inconsistent FRFs from the experimental substructure model, it is possible to improve the accuracy of the coupling process.

2 System Equivalent Model Mixing - SEMM

System Equivalent Model Mixing (SEMM) was first introduced by Klaassen et al. [2]. The method forms a hybrid structural dynamic model by mixing the numerical and experimental FRFs. The method is based on the parent model (Fig. 1a), which provides the extensive DoF set and is usually of numerical nature. The dynamic properties are introduced by the overlay model (Fig. 1b), which is generally obtained by experiment. To form the final hybrid model (Fig. 1d) the dynamic properties of the parent model are eliminated with the removed model (Fig. 1c).

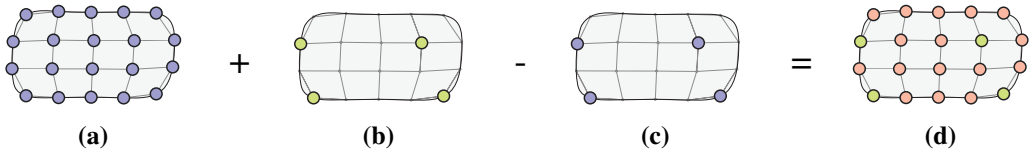


Fig. 1: Equivalent models for SEMM method; a) Parent model $\mathbf{Y}^{\text{par}}$, b) Overlay model $\mathbf{Y}^{\text{ov}}$, c) Removed model $\mathbf{Y}^{\text{rem}}$, d) Hybrid model $\mathbf{Y}^{\text{SEMM}}$.

The compatibility and equilibrium conditions between equivalent models are imposed using the Lagrange Multiplier Frequency Based Substructuring (LM FBS) framework [3], resulting in following form of basic formulation of SEMM method:

$$\mathbf{Y}^{\text{SEMM}} = \left[\mathbf{Y}\right]^{\text{par}} - \begin{bmatrix}\mathbf{Y}_{\text{ib}}\\ \mathbf{Y}_{\text{bb}}\end{bmatrix}^{\text{par}} \left(\mathbf{Y}^{\text{rem}}\right)^{-1} \left(\mathbf{Y}^{\text{rem}} - \mathbf{Y}^{\text{ov}}\right) \left(\mathbf{Y}^{\text{rem}}\right)^{-1} \begin{bmatrix}\mathbf{Y}_{\text{bi}} & \mathbf{Y}_{\text{bb}}\end{bmatrix}^{\text{par}}. \tag{1}$$

The basic SEMM method also has some extensions [2] that increase its robustness. The ability to remove spurious peaks, which are a consequence of the conflicting dynamics between the overlay ($\mathbf{Y}^{\text{ov}}$) and the removed numerical ($\mathbf{Y}^{\text{par}}$), is essential to improve the method's applicability [2]. If the removed interface is extended to all the internal DoFs, then the removed model is same as parent model. The final version of the fully extend SEMM method in a single-line notation can be written as:

$$\mathbf{Y}^{\text{SEMM}} = \mathbf{Y}^{\text{par}} - \mathbf{Y}^{\text{par}} \left(\begin{bmatrix}\mathbf{Y}_{\text{bi}} & \mathbf{Y}_{\text{bb}}\end{bmatrix}^{\text{rem}}\right)^{+} \left(\mathbf{Y}^{\text{rem}}_{\text{bb}} - \mathbf{Y}^{\text{ov}}\right) \left(\begin{bmatrix}\mathbf{Y}_{\text{ib}}\\ \mathbf{Y}_{\text{bb}}\end{bmatrix}^{\text{rem}}\right)^{+} \mathbf{Y}^{\text{par}}. \tag{2}$$

In pyFBS library [4], the SEMM expansion is performed with function `pyFBS.expansion.SEMM()`, where you can control `SEMM_type` by setting it to `basic`, `fully-extend` or `fully-extend-svd`.

3 Identification of inconsistent measurements in the frequency domain

The identification algorithm is depicted in Figure 2. Its operation requires an experimental and numerical model of analyzed structure. The latter represents only a tool for extending the DoFs within the

SEMM method. In the second step, an individual measurement is extracted from the experimental model and reconstructed using the SEMM method based on the remaining measurements in the experimental model. When the procedure is repeated for all DoFs of the experimental model, the reconstructed measurements are compared with real measurements via the coherence criterion [5]. DoFs, where the value of average coherence does not reach a coherence limit, are identified as inconsistent measurements.

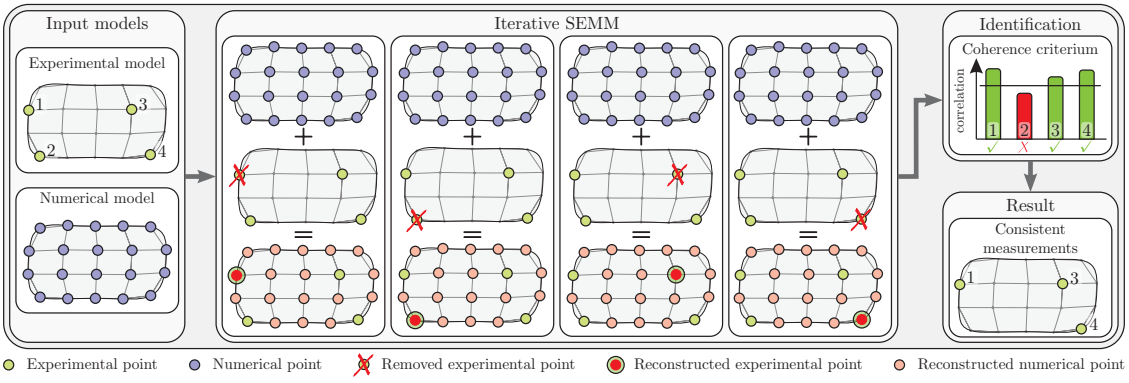


Fig. 2: Identification algorithm for the identification of inconsistent measurements in the frequency domain.

In the pyFBS library [4], the identification algorithm is performed completely automatically by calling the built-in function `pyFBS.expansion.identification_algorithm()`, which returns reconstructed FRFs and values of coherence criteria.

4 Experimental case study

The applicability of the proposed method for the identification of inconsistent measurements is demonstrated on a simple beam structure with a rectangular cross-section. The beam A-A represents the main system assembled from two equal beams A (Fig. 3). In this paper the dynamic response of the beam A-A will be obtained by coupling the dynamics of two identical beams A. On beam A, 26 equidistant measuring points were used (Fig. 3) for experimental and numerical model.

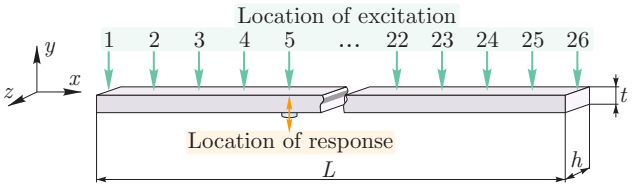


Fig. 3: Schematic presentation of beam A.

Parameter	Value
L	300 mm
h	40 mm
t	12 mm
E	205 GPa
ρ	7820 kg/m^3

Tab. 1: Parameters of beam A.

The result of identification criteria is the coherence diagram, shown in Fig. 4, where the boundary-coherence value was chosen to be 0.90. The measurements at points 15 and 24 are identified as inconsistent and will be removed from the existing experimental data set.

To show the efficiency of the proposed method for the identification of inconsistent measurements, two identical beams A were coupled using the LM FBS method to form a dynamic model of the coupled beam A-A. To validate the coupling results, the reference experimental dynamic model of the coupled

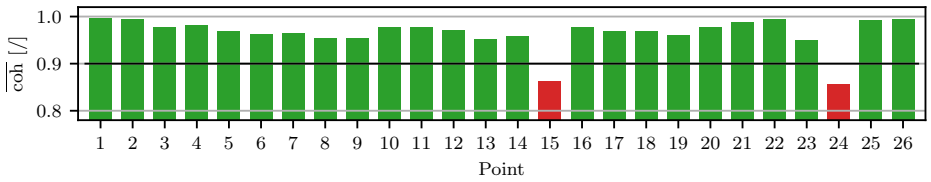


Fig. 4: Average value of the coherence and LAC criterion.
(■) - Consistent Measurement Coherence, (■) - Inconsistent Measurement Coherence,
(—) - Boundary Coherence

beam A-A was obtained independently. An example of the coupled structure's dynamic response is shown in Fig. 5. The beam was coupled using the entire experimental model as well as the experimental model, which contained only consistent measurements. The coupling results were significantly more accurate when we used the latter experimental model.

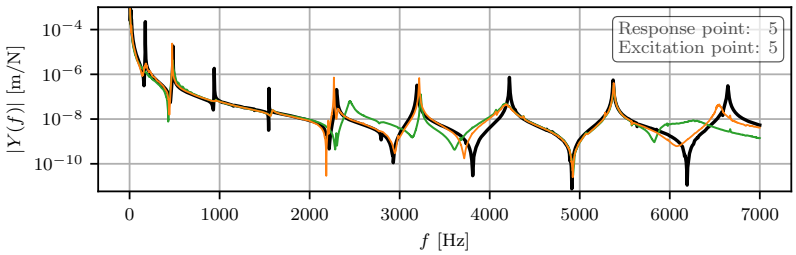


Fig. 5: FRF of coupled system.
(—) - Ref. Meas., (—) - SEMM - consistent Meas. (—) - SEMM - all Meas.

References

[1] Y. Chen, P. Avitabile, and J. Dodson, "Data consistency assessment function (DCAF)," *Mechanical Systems and Signal Processing*, vol. 141, p. 106688, 2020.

[2] S. W. Klaassen, M. V. van der Seijs, and D. de Klerk, "System equivalent model mixing," *Mechanical Systems and Signal Processing*, vol. 105, pp. 90–112, 2018.

[3] D. de Klerk, D. J. Rixen, and J. de Jong, "The frequency based substructuring (FBS) method reformulated according to the dual domain decomposition method," in *Proceedings of the 24th IMAC*, pp. 1–14, 2006.

[4] T. Bregar, A. El Mahmoudi, M. Kodrič, D. Ocepek, F. Trainotti, M. Pogačar, M. Göldeli, G. Čepon, M. Boltežar, and D. J. Rixen, "pyFBS: A python package for frequency based substructuring," *Journal of Open Source Software*, vol. 7, no. 69, p. 3399, 2022.

[5] M. V. van der Seijs, D. D. van den Bosch, D. J. Rixen, and D. de Klerk, "An improved methodology for the virtual point transformation of measured frequency response functions in dynamic substructuring," in *4th ECCOMAS Thematic Conference on Computational Methods in Structural Dynamics and Earthquake Engineering (COMPDYN)*, pp. 4334–4347, 2013.

LadiskDAQ: An Open-Source Python Package for Unified and Efficient Data Acquisition and Signal Generation

Tilen Košir Klemen Zaletelj Janko Slavič * Miha Boltežar

University of Ljubljana, Faculty of Mechanical Engineering

Abstract

In the field of structural dynamics, measuring and processing data is an everyday task. Python has become a popular tool for data processing, but its usage for data acquisition is currently limited or requires a significant effort in programming from the individual to use Python for data acquisition. The lack of a unified system for data acquisition in Python is a problem that affects the efficiency and ease of data acquisition and storage. `LadiskDAQ` is a data acquisition package developed in Python to solve this problem.

`LadiskDAQ` provides a platform for easy data acquisition and signal generation from various sources, including National Instruments, serial communication devices such as Arduino and ESP, FLIR thermal cameras, and more. It also allows for the easy addition of new data acquisition and signal generation sources that are not yet supported. LadiskDAQ enables real-time plotting of measured signals, as well as processed signals, such as FFT and arbitrary functions. The package allows for easy customization of real-time plotting. Being an open-source package, `LadiskDAQ` is available for everyone to use and improve.

The creation of `LadiskDAQ` has opened up opportunities for everyone to use the package in their data acquisition procedures and improve upon it. It allows for the unification of data acquisition systems, simplifies the process of data acquisition, and provides a platform for further improvement and customization. The benefits of `LadiskDAQ` include its ease of use, its compatibility with Python for further signal processing, and its ability to unify and simplify data acquisition systems, ultimately improving efficiency and productivity in data acquisition procedures.

Keywords: open source, data acquisition, signal generation, python

1 Statement of Need

In the field of structural dynamics, tasks related to the measurement and processing of data are commonplace. Python, a versatile high-level programming language, has gained popularity for data processing. However, the use of Python for data acquisition is currently limited to custom hardware solutions and requires significant effort for effective implementation. The lack of a unified system for data acquisition within the Python environment poses a challenge to the efficiency and simplicity of data acquisition and storage. This paper presents `LadiskDAQ`, an open-source Python package developed to address these challenges.

*Corresponding author, Email address: janko.slavic@fs.uni-lj.si

2 LadiskDAQ: Unified Data Acquisition and Signal Generation

LadiskDAQ provides a platform for unified data acquisition and signal generation from a variety of sources, including National Instruments devices, serial communication devices such as Arduino and ESP, FLIR thermal imaging cameras, and many others. In addition, LadiskDAQ's design allows for easy integration of other data acquisition and signal generation sources that are not currently supported. In addition to its versatility, LadiskDAQ supports the real-time display of both raw and processed signals (e.g., FFT and arbitrary functions), with the ability for the user to customize the display. The package contains the following main class objects: `LadiskDAQ.BaseAcquisition`, `LadiskDAQ.BaseGeneration`, `LadiskDAQ.Visualization` and `LadiskDAQ.Core`.

`LadiskDAQ.BaseAcquisition` serves as an acquisition template that takes care of proper acquisition. All acquisition sources currently supported in LadiskDAQ are implemented as child classes of `LadiskDAQ.BaseAcquisition`. Similarly, `LadiskDAQ.BaseGeneration` is a template class object that takes care of signal generation. `LadiskDAQ.Visualization` takes care of the real-time display of the acquired data. All of the above objects can then be combined into the `LadiskDAQ.Core` object, which combines the acquisition, generation, and visualization objects into an easy-to-use data acquisition system. All objects are completely decoupled from each other, making LadiskDAQ modular and adaptable to specific measurement applications. More information about the package can be found in the GitHub repository[1].

2.1 Using LadiskDAQ

The basic use of `LadiskDAQ` involves creating an acquisition and generation objects using the appropriate acquisition and generation classes depending on the hardware used for the measurement. Sample code Listing 1 shows a brief example of using the package when using National Instruments hardware. More detailed and advanced examples are available in the LadiskDAQ source code repository [2].

```
import LadiskDAQ
import numpy as np

#create excitation signals:
fs = 25600 # output sample rate
t = np.arange(fs * 10) / fs
signal1 = np.sin(2*np.pi*800*t)
signal2 = np.sin(2*np.pi*200*t)
excitation_signals = np.array([signal1, signal2]).T

# create acqusition and generation objects:
# assume 'input_task_name' is the name of the NI task created in NI MAX
# assume 'output_task_name' is the name of the NI task created in NI MAX
acq = LadiskDAQ.national_instruments.NIAcquisition('input_task_name',
    acquisition_name='NI')
gen = LadiskDAQ.national_instruments.NIGeneration('output_task_name',
    excitation_signals)

# create visualization object and configure live visualization::
vis = LadiskDAQ.Visualization(refresh_rate=100)
vis.add_lines(position=(0,0), source="NI", channels=[0, 1]) # Time signals
vis.add_lines(position=(1,0), source="NI", channels=[0, 1], function="fft") #
    Fourier transform
```

```
vis.config_subplot((0, 0), t_span=0.05, ylim=(-10, 10))
vis.config_subplot((1, 0), t_span=5.0,  ylim=(0, 10), xlim=(200, 1000))

# create core object and add acquisition sources:
ldaq = LadiskDAQ.Core(acquisitions=[acq], generations=[gen], visualization=vis)

# configure trigger:
ldaq.set_trigger(
    source='NI_data_source',
    level=100,
    channel=0,
    duration=10.,
    presamples=100)

# run acquisition:
ldaq.run()

# Retrieve the measurement data:
measurement = ldaq.get_measurement_dict()

# save the measurement data:
ldaq.save_measurement(name="TestName")
```

Listing 1: Example of `LadiskDAQ` usage.

3 Conclusion

`LadiskDAQ` represents an advance in open-source data collection. It not only streamlines the data collection process, but also opens up the opportunity for the community to contribute to its further development. By bridging the gap between Python and data acquisition tasks, `LadiskDAQ` can help develop measurement workflows in the field of structural dynamics and beyond more efficiently, consistently, and quickly.

References

[1] LADISK, "The LadiskDAQ source code repository." https://github.com/ladisk/LadiskDAQ, June 2023.

[2] LADISK, "The LadiskDAQ usage examples." https://github.com/ladisk/LadiskDAQ/tree/master/examples, June 2023.

Matlab-based Finite Element Method for Rotor Dynamics

Quankun Li[1*] Ruixian Ma[1] Mingfu Liao[1] Xingjian Jing[2*]

[1]*Northwestern Polytechnical University,* [2]*City University of Hong Kong*

Abstract

Currently, there are many different types of mathematical methods for the modelling and analysis of flexible rotor-bearing systems such as lumped mass method, transfer matrix method and finite element method (FEM). Compared with the first two methods, the FEM will be more convenient and effective especially for systems with complicated components, boundaries and loads. Even though there are many mature commercial FE software for rotor dynamics like ANSYS, ABAQUS, SAMCEF et al., writing FEM codes with Matlab software is essential and helpful for understanding the basic principle of FEM and rotor dynamic characteristics. Therefore, this paper will introduce procedures for the code writing, which includes parameter reading from Excel document, matrix formation with loop statement, equation calculation with matrix operation, result display with plot function et al. With these procedures and related Matlab codes, some basic characteristics of rotor-bearing systems like Campbell diagram for critical speeds, mode shapes and stable responses of disks and bearings will be obtained for the following analysis of flexible rotor-bearing systems, including dynamic design, vibration control, condition monitoring and fault diagnosis. On the other hand, codes in this paper can be applied and extended to other more complicated rotor-bearing systems with two or more spools or gearboxes.

Keywords: Matlab, Finite Element Method, Rotor Dynamics, Code Writing

1 Introduction

The rotor-bearing system, generally including compressor, shaft, turbine, bearing et al. plays a very important role in aero engines. To calculate rotor dynamic characteristics including critical speeds, mode shapes and output responses, the finite element method (FEM), which is more convenient and effective compared with lumped mass method and transfer matrix method, has been proposed and applied [1, 2]. Even through there are many mature commercial FE software for rotor dynamics like ANSYS, ABAQUS, SAMCEF et al., writing FEM codes with Matlab or other open source software is essential and valuable for students and researchers to understand the basic principle of FEM and rotor dynamic characteristics.

In this paper, detailed procedures of the Matlab-based FEM for rotor dynamics are introduced. As shown in Fig. 1, there are four steps in the method. Firstly, the finite element rotor model including generalized coordinates, element matrixes, load vectors et al. is established according to the principle of structural and dynamic similarities. Secondly, parameters of the rotor model including geometric dimensions, material properties, support damping and stiffness coefficients et al. are listed in the Excel sheet. And then, related codes for parameter reading, matrix formation, equation calculation,

* Corresponding author, Email address: quankun_li@hotmail.com, xingjing@cityu.edu.hk.

result display et al. are written with Matlab software. Finally, rotor dynamic characterizes of the rotor model such as critical speeds, mode shapes, output responses et al. are demonstrated and analyzed.

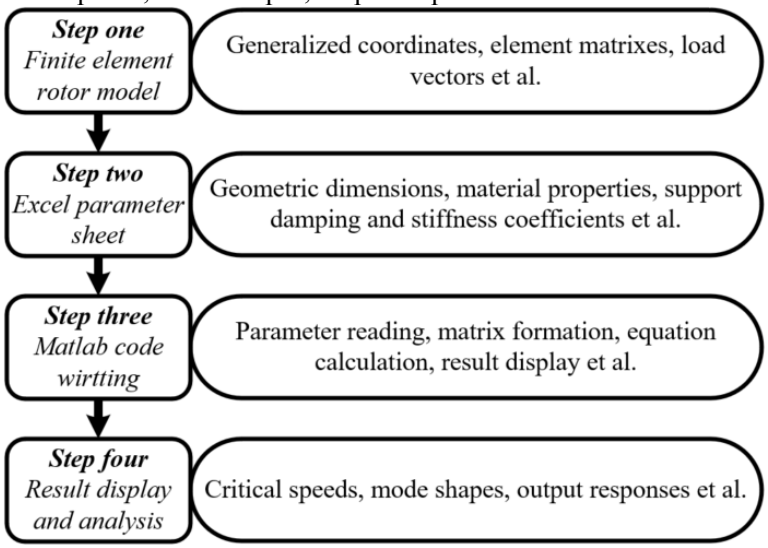


Fig. 1: Procedures of the FEM method.

2 Dynamic characteristics of the rotor

2.1 Finite element rotor model

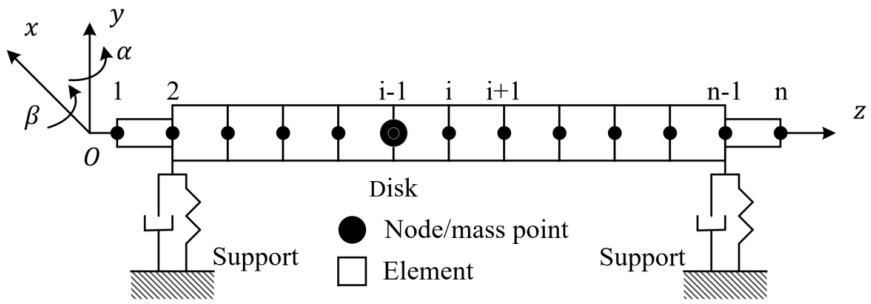


Fig. 2: The finite element rotor model.

In Fig. 2, a simply supported finite element rotor model with one disk is considered in this study. The rotor shaft is divided into several segments with elasticity and shaft mass is equivalently allocated to respective nodes. When the system rotates at a constant speed and only considering the translational and rotational vibration, each node of the model has four degrees of freedom, which can be described by the generalized coordinates [1, 2], as

$$o_i(t) = \{x_i(t), y_i(t), \beta_i(t), \alpha_i(t)\}, (i = 1,2,...,n) \tag{1}$$

where $x_i(t)$ and $y_i(t)$ represent translational coordinates, $\beta_i(t)$ and $\alpha_i(t)$ represent rotational coordinates.

Applying the Lagrange's equation and finite element method [1, 2, 3, 4], the differential equation of motion of the rotor model can be obtained as,

$$[M]\{\ddot{o}_i(t)\} + ([C] - \Omega[G])\{\dot{o}_i(t)\} + [K]\{o_i(t)\} = \{u_i(t)\} \tag{2}$$

where $[M]$, $[C]$, $[G]$ and $[K]$ represent linear mass, damping, gyroscopic and stiffness matrixes

respectively, their detailed expressions can be found in [1, 2]. Parameter Ω is the rotating speed of the rotor-bearing system. $\{\ddot{o}_i(t)\}$, $\{\dot{o}_i(t)\}$ and $\{o_i(t)\}$ represent time-domain acceleration, velocity and displacement vectors respectively. $\{u_i(t)\}$ represents the time-domain radial unbalance force vector.

2.2 Excel parameter sheet

In order to set and modify parameters conveniently, all parameters used in the rotor model including geometric dimensions, material properties, support damping and stiffness coefficients et al. are listed in the Excel sheet to be read (Fig. 3).

	A	B	C	D	E	F
1	01	1	2	3	4	5
2	Sha_D_o	2.50E-02	2.50E-02	0.00E+00	Outside diameter of the shaft	
3	Sha_d_o	0.00E+00	0.00E+00	0.00E+00	Inside diameter of the shaft	
4	Sha_l_o	2.40E-01	4.80E-01	0.00E+00	Length of the shaft	
5	Sha_l_e	8.00E-02	8.00E-02	8.00E-02	Length of the shaft element	Input
6	Sha_Ela_o	2.06E+11	2.06E+11		Elastic modulus of the shaft	
7	Sha_Den_o	6.97E+03	6.97E+03		Density of the shaft	
8	Sha_She_o	7.93E+10	7.93E+10		Shear modulus of the shaft	
9	Sha_Poi_o	2.59E-01	2.59E-01		Poisson ratio of the shaft	
10	Sha_N	3	6	0	Number of nodes	
11	Sha_A	4.91E-04	4.91E-04		Area of the shaft	
12	Sha_I	1.92E-08	1.92E-08		Second moment of area of the shaft	Calculated
13	Sha_Mp	3.42E+00	3.42E+00		Mass per length of the shaft	
14	Sha_Fai	2.15E-01	2.15E-01		Shear deformation coefficient of the shaft	
15	Dis_D_o		2.40E-01		Outside diameter of the disk	
16	Dis_d_o		0.00E+00		Inside diameter of the disk	
17	Dis_l_o		1.00E-01		Width of the disk	
18	Dis_Den_o		1.97E+03		Density of the disk	Input
19	Dis_Amp_o		1.00E-05		Unbalance force amplitude	
20	Dis_Pha_o		0.00E+00		Unbalance force phase	
21	Dis_Rub_o		0.00E+00		Rub impact fault coefficient	
22	Dis_M		8.90E+00		Mass of the disk	
23	Dis_Jp		6.39E-02		polar mass moment of inertia of the disk	Calculated
24	Dis_Jd		3.20E-02		Diameter mass moment of inertia of the disk	
25	Bea_Cx_o	0.00E+00		0.00E+00	x-direction bearing damping coefficient	
26	Bea_Cy_o	0.00E+00		0.00E+00	y-direction bearing damping coefficient	
27	Bea_Cn_o	0.00E+00		0.00E+00	z-direction bearing damping coefficient	Input
28	Bea_Kx_o	1.00E+08		1.00E+08	x-direction bearing stiffness coefficient	
29	Bea_Ky_o	1.00E+08		1.00E+08	y-direction bearing stiffness coefficient	
30	Bea_Kn_o	1.00E+08		1.00E+08	z-direction bearing stiffness coefficient	

Fig. 3: Excel parameter sheet.

From Fig. 3, it is shown that there are two types of parameters. When parameters in blue color are filled in, related parameters in white color will be calculated through functions in the Excel. All calculated parameters will be compared with that calculated in the Matlab for verification.

2.3 Matlab code writing

In the Matlab working window, creating a new script, and writing corresponding codes for the parameter reading, matrix formation, equation calculation, result display et al. All codes and related descriptions are shown as follows.

```
%%%Parameter
f_01_max=60; %%%Maximum frequency
f_01=0:0.2:f_01_max; %%%Frequency range
w_01=2*pi*f_01; %%%Angular frequency range
Num_ord=1;  %%%Calculated order
%%%Element matrix
```

```
    [Data,text]=xlsread('E:\04 Research\04 Diagnosis\0206 ROBO\01 Simulation\Rotor','20230410
','B1:AO120');
    if Data(2,1)>0
       ELE_01_O=Data(1:30,1:find(Data(10,:)==0));
       ELE_01_O(10,:)=round(ELE_01_O(4,:)./ELE_01_O(5,:)); %%%Sha_N
       ELE_01_O(11,:)=pi.*(ELE_01_O(2,:).^2-ELE_01_O(3,:).^2)./4; %%%Sha_A
       ELE_01_O(12,:)=pi.*(ELE_01_O(2,:).^4-ELE_01_O(3,:).^4)./64; %%%Sha_I
       ELE_01_O(13,:)=ELE_01_O(7,:).*ELE_01_O(11,:); %%%Sha_Mp
       ELE_01_O(14,:)=12*ELE_01_O(6,:).*ELE_01_O(12,:)./ELE_01_O(8,:)./ELE_01_O(5,:).^2
./(ELE_01_O(11,:)./((7+6.*ELE_01_O(9,:))./(6+6.*ELE_01_O(9,:)))./(1+(20+12.*ELE_01_O(9,:))
./(7+6*ELE_01_O(9,:)).*(ELE_01_O(2,:).*ELE_01_O(3,:)./(ELE_01_O(2,:).^2+ELE_01_O(3,:).^
2)).^2)); %%%Sha_Fai
       ELE_01_O(22,:)=ELE_01_O(18,:).*pi.*(ELE_01_O(15,:).^2-ELE_01_O(16,:).^2)./4.*ELE
_01_O(17,:); %%%Dis_M
       ELE_01_O(23,:)=0.5.*ELE_01_O(22,:).*(ELE_01_O(15,:).^2+ELE_01_O(16,:).^2)./4; %%
%Dis_Jp
       ELE_01_O(24,:)=ELE_01_O(23,:)./2; %%%Dis_Jd
       Num_01_ele=sum(ELE_01_O(10,:));
       Num_01_nod=Num_01_ele+1;
       ELE_01_C=zeros(30,Num_01_nod);
       %%%Shaft
       Sha_01_nod_sta=0;Sha_01_nod_end=0;
       for aaa=1:size(ELE_01_O,2)-1
          Sha_01_nod_sta=Sha_01_nod_end+1;
          Sha_01_nod_end=Sha_01_nod_sta+ELE_01_O(10,aaa)-1;
          for bbb=1:ELE_01_O(10,aaa)
             ELE_01_C(2:14,Sha_01_nod_sta+bbb-1)=ELE_01_O(2:14,aaa);
          end
       end
       %%%Disk
       Dis_01_nod_O=find(~isnan(ELE_01_O(15,:)));
       for aaa=1:length(Dis_01_nod_O)
          Dis_01_nod_C(aaa)=sum(ELE_01_O(10,1:Dis_01_nod_O(aaa)))-ELE_01_O(10,Dis_01_
nod_O(aaa))+1;
          ELE_01_C(15:24,Dis_01_nod_C(aaa))=ELE_01_O(15:24,Dis_01_nod_O(aaa));
       end
       %%%Bearing
       Bea_01_nod_O=find(~isnan(ELE_01_O(25,:)));
       for aaa=1:length(Bea_01_nod_O)
          Bea_01_nod_C(aaa)=sum(ELE_01_O(10,1:Bea_01_nod_O(aaa)))-ELE_01_O(10,Bea_01
_nod_O(aaa))+1;
          ELE_01_C(25:30,Bea_01_nod_C(aaa))=ELE_01_O(25:30,Bea_01_nod_O(aaa));
       end
    end
    %%%%%%%%%%%%System Matrix
    % (Num_nod,Sha_Ela,Sha_l_e,Sha_A,Sha_I,Sha_Mp,Sha_Fai)
    [M_Sha_01,G_Sha_01,K_Sha_01]=MCK_Sha(Num_01_nod,ELE_01_C(6,:),ELE_01_C(5,:),
ELE_01_C(11,:),ELE_01_C(12,:),ELE_01_C(13,:),ELE_01_C(14,:));
    % (Num_nod,Dis_M,Dis_Jp,Dis_Jd,Dis_Amp,Dis_Pha)
```

```
    [M_Dis_01,G_Dis_01,U_Dis_01]=MCK_Dis(Num_01_nod,ELE_01_C(22,:),ELE_01_C(23,:),
ELE_01_C(24,:),ELE_01_C(19,:),ELE_01_C(20,:));
    % (Num_nod,Bea_Cx,Bea_Cy,Bea_Kx,Bea_Ky)
    [C_Bea_01,K_Bea_01,U_Bea_01]=MCK_Bea(Num_01_nod,ELE_01_C(25,:),ELE_01_C(26,:
),ELE_01_C(28,:),ELE_01_C(29,:));
    Num_nod=Num_01_nod;
    M_sys=M_Sha_01+M_Dis_01;
    C_sys=C_Bea_01;
    G_sys=G_Sha_01+G_Dis_01;
    K_sys=K_Sha_01+K_Bea_01;
    U_Sys=U_Dis_01+U_Bea_01;
    %%%%%%%%%%%%%Modal property
    for aaa=1:length(w_01)
      A_sys=[zeros(Num_nod*4),eye(Num_nod*4);-M_sys\K_sys,-M_sys\(C_sys-w_01(aaa)*G_
sys)];
      [EV,ED]=eig(A_sys);
      [ED_S,ED_S_ind]=sort(imag(diag(ED)));
      Eng_D=imag(ED(ED_S_ind,ED_S_ind));
      Eng_V=EV(:,ED_S_ind);
      Num_pos=find(ED_S>0);
      Eng_D_T(:,aaa)=diag(Eng_D(Num_pos:length(ED_S),Num_pos:length(ED_S)));
      Eng_V_T(:,:,aaa)=Eng_V(:,Num_pos:length(ED_S));
    end
    %%%Critical speed
    for aaa=1:1:Num_ord
      figure (1000+aaa)
      if aaa==1
        cla;
      end
      plot(f_01,w_01,'b','LineWidth',2)
      hold on
      plot(f_01,Eng_D_T(aaa*2-1,:),'r','LineWidth',2)
      hold on
      plot(f_01,Eng_D_T(aaa*2,:),'k','LineWidth',2)
      hold on
      for bbb=1:length(w_01)-1
        if w_01(bbb)<Eng_D_T(aaa*2,bbb)&&w_01(bbb+1)>Eng_D_T(aaa*2,bbb+1)
          Eng_D_R(aaa,1)=(w_01(bbb)*Eng_D_T(aaa*2,bbb+1)-w_01(bbb+1)*Eng_D_T(aaa*2
,bbb))/(w_01(bbb)-w_01(bbb+1)-Eng_D_T(aaa*2,bbb)+Eng_D_T(aaa*2,bbb+1));
          Eng_V_R(aaa,:)=Eng_V_T(:,aaa*2,bbb);
          plot(Eng_D_R(aaa)/2/pi,Eng_D_R(aaa),'*','LineWidth',2)
          hold on
          grid on
          xlabel('Frequency (Hz)')
          ylabel('Angular frequency (Rad/s)')
          set(gca,'Fontname','Times New Roman','FontSize',16);
          legendR=legend('45^{\circ} line','Forward whirl','Backward whirl');
          set(legendR,'Box','off','Orientation','Vertical','Location','northwest','Fontname','Times N
ew Roman','FontSize',16)
```

```
            %%Mode shape
            for ccc=1:size(Eng_V_R,2)/2/4
               Eng_V_R_x(aaa,ccc)=Eng_V_R(aaa,(ccc-1)*4+1);
               Eng_V_R_y(aaa,ccc)=Eng_V_R(aaa,(ccc-1)*4+2);
            end
            figure(2000+aaa)
            cla;
            plot3(1:1:Num_nod,ones(Num_nod,1)*0,ones(Num_nod,1)*0,'b','LineWidth',2)
            hold on
            plot3(1:Num_nod,real(Eng_V_R_x(aaa,:)),imag(Eng_V_R_x(aaa,:)),'r','LineWidth',2)
            hold on
            for ccc=1:length(Dis_01_nod_O)
               plot3(sum(ELE_01_O(10,1:Dis_01_nod_O(ccc)))-ELE_01_O(10,Dis_01_nod_O(ccc
))+1,0,0,'*','LineWidth',2)
               hold on
            end
            for ccc=1:length(Bea_01_nod_O)
               plot3(sum(ELE_01_O(10,1:Bea_01_nod_O(ccc)))-ELE_01_O(10,Bea_01_nod_O(cc
c))+1,0,0,'^','LineWidth',2)
               hold on
            end
            for ccc=1:Num_nod
               plot3(ones(360,1)*ccc,abs(Eng_V_R_x(aaa,ccc))*cos((1:360)/180*pi),abs(Eng_V_R
_x(aaa,ccc))*sin((1:360)/180*pi))
               hold on
            end
            view(-45,45)
            grid on
            xlabel('z-direction')
            ylabel('x-direction')
            zlabel('y-direction')
            set(gca,'Fontname','Times New Roman','FontSize',16);
            legendR=legend('Center line','The first order mode shape');
            set(legendR,'Box','off','Orientation','Vertical','Location','northwest','Fontname','Times N
ew Roman','FontSize',16)
         end
      end
   end
   %%%%%%%%%%%%%%Output spectrum
   %%%Figure
   for aaa=1:length(w_01)
      FA_sys_P=-w_01(aaa)^2*M_sys+1i*w_01(aaa)*(C_sys-w_01(aaa)*G_sys)+K_sys;
      FU_sys_P=w_01(aaa)^2.*U_Sys(:,1);
      FQ_sys_P(:,aaa)=FA_sys_P\FU_sys_P;
      FA_sys_N=-w_01(aaa)^2*M_sys-1i*w_01(aaa)*(C_sys-w_01(aaa)*G_sys)+K_sys;
      FU_sys_N=w_01(aaa)^2.*U_Sys(:,2);
      FQ_sys_N(:,aaa)=FA_sys_N\FU_sys_N;
   end
   figure(3000)
```

```
cla;
plot(f_01,2*abs(FQ_sys_P((Dis_01_nod_C(1)-1)*4+1,:)),'b','LineWidth',2)
grid on
xlabel('Frequency (Hz)')
ylabel('Amplitude')
set(gca,'Fontname','Times New Roman','FontSize',16);
%%%%%%%%%%%%%Element matrix
%%%Shaft
function [M_Sha,G_Sha,K_Sha]=MCK_Sha(Num_nod,Sha_Ela,Sha_s_e,Sha_A,Sha_I,Sha_M
p,Sha_Fai)
  M_Sha=zeros(Num_nod*4,Num_nod*4);
  G_Sha=zeros(Num_nod*4,Num_nod*4);
  K_Sha=zeros(Num_nod*4,Num_nod*4);
  for aaa=1:Num_nod-1
    MT1=13/35+7/10*Sha_Fai(aaa)+1/3*Sha_Fai(aaa)^2;
    MT2=(1/105+1/60*Sha_Fai(aaa)+1/120*Sha_Fai(aaa)^2)*Sha_s_e(aaa)^2;
    MT3=(11/210+11/120*Sha_Fai(aaa)+1/24*Sha_Fai(aaa)^2)*Sha_s_e(aaa)^1;
    MT4=9/70+3/10*Sha_Fai(aaa)+1/6*Sha_Fai(aaa)^2;
    MT5=(13/420+3/40*Sha_Fai(aaa)+1/24*Sha_Fai(aaa)^2)*Sha_s_e(aaa)^1;
    MT6=(1/140+1/60*Sha_Fai(aaa)+1/120*Sha_Fai(aaa)^2)*Sha_s_e(aaa)^2;
    MT=[MT1 0 0 0 0 0 0 0;
      0 MT1 0 0 0 0 0 0;
      0 -MT3 MT2 0 0 0 0 0;
      MT3 0 0 MT2 0 0 0 0;
      MT4 0 0 MT5 MT1 0 0 0;
      0 MT4 -MT5 0 0 MT1 0 0;
      0 MT5 -MT6 0 0 MT3 MT2 0;
      -MT5 0 0 -MT6 -MT3 0 0 MT2];
    MT=MT+MT.'-diag([MT1,MT1,MT2,MT2,MT1,MT1,MT2,MT2]);
    MR1=6/5;
    MR2=(2/15+1/6*Sha_Fai(aaa)+1/3*Sha_Fai(aaa)^2)*Sha_s_e(aaa)^2;
    MR3=(1/10-1/2*Sha_Fai(aaa))*Sha_s_e(aaa)^1;
    MR4=-(1/30+1/6*Sha_Fai(aaa)-1/6*Sha_Fai(aaa)^2)*Sha_s_e(aaa)^2;
    MR=[MR1 0 0 0 0 0 0 0;
      0 MR1 0 0 0 0 0 0;
      0 -MR3 MR2 0 0 0 0 0;
      MR3 0 0 MR2 0 0 0 0;
      -MR1 0 0 -MR3 MR1 0 0 0;
      0 -MR1 MR3 0 0 MR1 0 0;
      0 -MR3 MR4 0 0 MR3 MR2 0;
      MR3 0 0 MR4 -MR3 0 0 MR2];
    MR=MR+MR.'-diag([MR1,MR1,MR2,MR2,MR1,MR1,MR2,MR2]);
    M_Sha((aaa-1)*4+1:(aaa-1)*4+8,(aaa-1)*4+1:(aaa-1)*4+8)=M_Sha((aaa-1)*4+1:(aaa-1)*
4+8,(aaa-1)*4+1:(aaa-1)*4+8)+Sha_Mp(aaa)*Sha_s_e(aaa)/(1+Sha_Fai(aaa))^2.*MT+Sha_Mp(aaa
)/Sha_A(aaa)*Sha_I(aaa)/(1+Sha_Fai(aaa))^2/Sha_s_e(aaa).*MR;
    G1=36;
    G2=3*Sha_s_e(aaa)-15*Sha_s_e(aaa)*Sha_Fai(aaa);
    G3=4*Sha_s_e(aaa)^2+5*Sha_s_e(aaa)^2*Sha_Fai(aaa)+10*Sha_s_e(aaa)^2*Sha_Fai(aa
a)^2;
```

```
            G4=Sha_s_e(aaa)^2+5*Sha_s_e(aaa)^2*Sha_Fai(aaa)-5*Sha_s_e(aaa)^2*Sha_Fai(aaa)^2
;
            G=[0 0 0 0 0 0 0 0;
               G1 0 0 0 0 0 0 0;
               -G2 0 0 0 0 0 0 0;
              0 -G2 G3 0 0 0 0 0;
               0 G1 -G2 0 0 0 0 0;
               -G1 0 0 -G2 G1 0 0 0;
              -G2 0 0 G4 G2 0 0 0;
               0 -G2 -G4 0 0 G2 G3 0];
            G=G-G.';
            G_Sha((aaa-1)*4+1:(aaa-1)*4+8,(aaa-1)*4+1:(aaa-1)*4+8)=G_Sha((aaa-1)*4+1:(aaa-1)*
4+8,(aaa-1)*4+1:(aaa-1)*4+8)+Sha_Mp(aaa)/Sha_A(aaa)*Sha_I(aaa)/(1+Sha_Fai(aaa))^2/15/Sha_
s_e(aaa).*G;
            KT1=12;
            KT2=(4+Sha_Fai(aaa))*Sha_s_e(aaa)^2;
            KT3=6*Sha_s_e(aaa);
            KT4=(2-Sha_Fai(aaa))*Sha_s_e(aaa)^2;
            KT=[KT1 0 0 0 0 0 0 0;
               0 KT1 0 0 0 0 0 0;
               0 -KT3 KT2 0 0 0 0 0;
               KT3 0 0 KT2 0 0 0 0;
               -KT1 0 0 -KT3 KT1 0 0 0;
               0 -KT1 KT3 0 0 KT1 0 0;
               0 -KT3 KT4 0 0 KT3 KT2 0;
               KT3 0 0 KT4 -KT3 0 0 KT2];
            KT=KT+KT.'-diag([KT1,KT1,KT2,KT2,KT1,KT1,KT2,KT2]);
            K_Sha((aaa-1)*4+1:(aaa-1)*4+8,(aaa-1)*4+1:(aaa-1)*4+8)=K_Sha((aaa-1)*4+1:(aaa-1)*
4+8,(aaa-1)*4+1:(aaa-1)*4+8)+Sha_Ela(aaa)*Sha_I(aaa)/(1+Sha_Fai(aaa))/Sha_s_e(aaa)^3.*KT;
        end
    end
    %%%Disk
    function [M_Dis,G_Dis,U_Dis]=MCK_Dis(Num_nod,Dis_M,Dis_Jp,Dis_Jd,Dis_Amp,Dis_Ph
a)
        M_Dis=zeros(Num_nod*4,Num_nod*4);
        G_Dis=zeros(Num_nod*4,Num_nod*4);
        U_Dis=zeros(Num_nod*4,4);
        for aaa=1:Num_nod
            M_Dis((aaa-1)*4+1:(aaa-1)*4+4,(aaa-1)*4+1:(aaa-1)*4+4)=diag([Dis_M(aaa),Dis_M(aaa
),Dis_Jd(aaa),Dis_Jd(aaa)]);
            G_Dis((aaa-1)*4+1:(aaa-1)*4+4,(aaa-1)*4+1:(aaa-1)*4+4)=[0 0 0 0;0 0 0 0;0 0 0 -Dis_Jp(
aaa);0 0 Dis_Jp(aaa) 0];
            U_Dis((aaa-1)*4+1:(aaa-1)*4+2,1)=Dis_Amp(aaa)/2*[exp(1i*Dis_Pha(aaa)/180*pi);exp(
1i*Dis_Pha(aaa)/180*pi)/1i];
            U_Dis((aaa-1)*4+1:(aaa-1)*4+2,2)=Dis_Amp(aaa)/2*[exp(-1i*Dis_Pha(aaa)/180*pi);-ex
p(-1i*Dis_Pha(aaa)/180*pi)/1i];
        end
    end
    %%%Bearing
```

```
function [C_Bea,K_Bea,U_Bea]=MCK_Bea(Num_nod,Bea_Cx,Bea_Cy,Bea_Kx,Bea_Ky)
    C_Bea=zeros(Num_nod*4,Num_nod*4);
    K_Bea=zeros(Num_nod*4,Num_nod*4);
    U_Bea=zeros(Num_nod*4,4);
    for aaa=1:Num_nod
        C_Bea((aaa-1)*4+1:(aaa-1)*4+4,(aaa-1)*4+1:(aaa-1)*4+4)=[Bea_Cx(aaa) 0 0 0;0 Bea_Cy
(aaa) 0 0;0 0 0 0;0 0 0 0];
        K_Bea((aaa-1)*4+1:(aaa-1)*4+4,(aaa-1)*4+1:(aaa-1)*4+4)=[Bea_Kx(aaa) 0 0 0;0 Bea_K
y(aaa) 0 0;0 0 0 0;0 0 0 0];
    end
```

2.4 Result display and analysis

With the excel parameter sheet and Matlab codes, some dynamic characteristics of the rotor model like the first order critical speed, related mode shape and stable output response are shown in Fig. 4, Fig. 5 and Fig. 6 respectively.

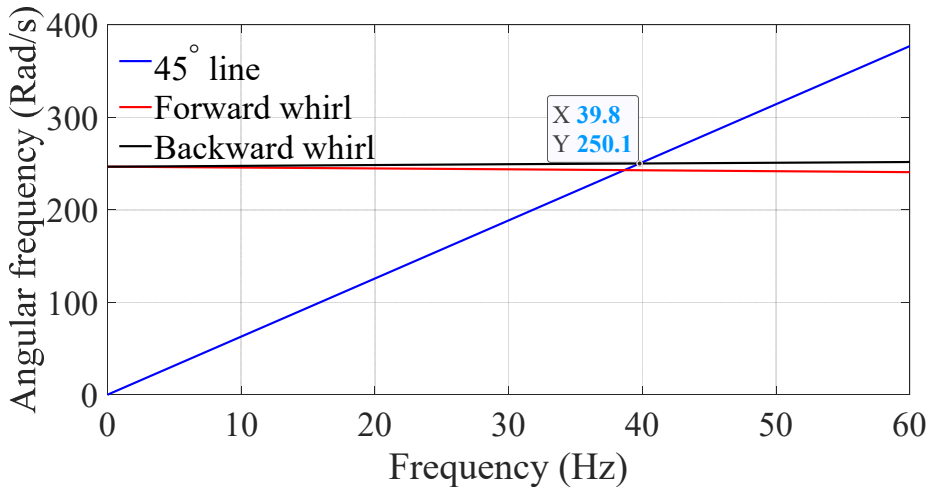


Fig. 4: Campbell diagram.

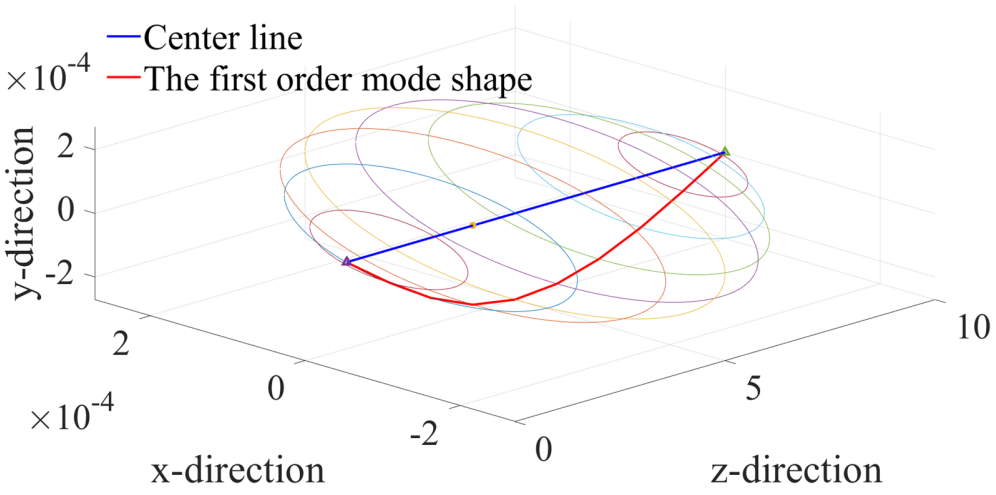


Fig. 5: Mode shape.

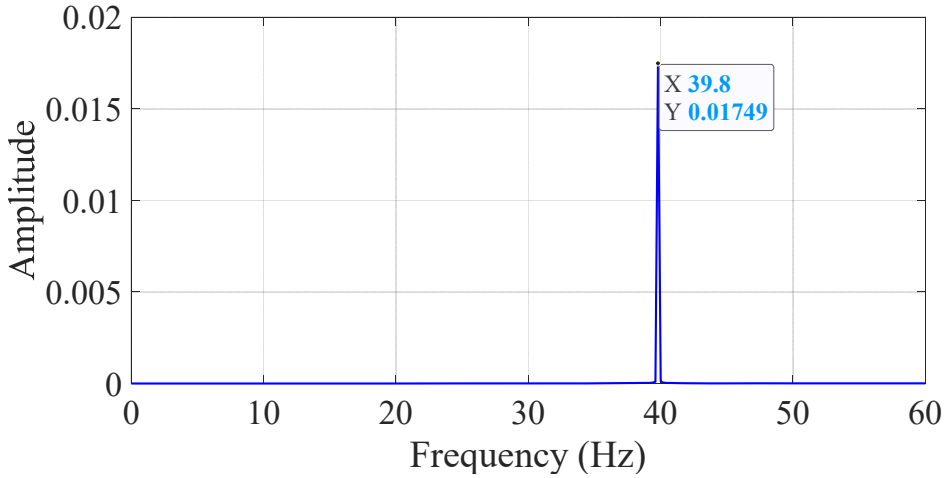


Fig. 6: Stable response of the disk.

From the Campbell diagram in Fig. 4, it can be seen that the first order critical speed is about 39.8Hz, and related mode shape shown in Fig. 5 demonstrates that the first order whirl orbit is the bending vibration of the shaft. The reason is that support stiffness coefficients are 1e8 shown in Fig. 3, which is very larger compared with the shaft stiffness. On the other hand, stable response of the disk in Fig. 6 also shows that the disk has a resonance peak at the critical speed (39.8 Hz).

3 Conclusions

Through the introduction of procedures of the Matlab-based FEM in previous sections, it can be concluded that the FEM can be easily applied through the Matlab codes. Compared with some commercial FE software, which is like a black box, writing codes with Matlab or other open source software would be more direct and clearer to show the basic principle of rotor dynamics and FEM for students and researchers.

References

[1] H.D. Nelson, J.M. McVaugh, "The dynamics of rotor-bearing systems using finite elements," *Journal of Engineering for Industry*, vol. 1, pp. 593-600, May. 1976.

[2] H.D. Nelson, "A finite rotating shaft element using Timoshenko beam theory," *Journal of Mechanical Design*, vol. 2, pp. 793-803, Oct. 1980.

[3] M.I. Friswell, J.E. Penny, S.D. Garvey, A.W. Lees, "Dynamics of rotating machines," *Cambridge university press*, 2010.

[4] Q.K. Li, M.F. Liao, X.J. Jing, "Transmissibility function-based fault diagnosis methods for beam-like engineering structures: a review of theory and properties," *Nonlinear Dynamics*, vol. 1, pp. 1-33, Nov. 2021.

A Python toolbox for the design of composite-made structural components

Edoardo Mancini * Massimiliano Palmieri Filippo Cianetti

University of Perugia *University of Perugia* *University of Perugia*

Abstract

Starting as materials for high-performance applications, composites are now commonly used in various engineering fields. Undoubtedly, the high mechanical performances and low density played a crucial role in their spreading. Nevertheless, one of the main advantages of composite is the material properties design and optimization. It is indeed possible for the structural engineer to create a custom material by arranging a certain number of like or unlike plies in different orientations. Despite this advantage, such peculiarity introduces new design challenges and complications. The consequence is an increased FE computational cost. Aiming to act on this specific problem, the authors developed an open-source toolbox for the structural analysis of layered materials. With the capability to determine material properties and stress states, the devised instrument should support the structural engineer and quicken the preliminary design phase.

Keywords: Composite structures, Open-source design tool, Python coded

1 Introduction / Statement of Need

Composite materials are widely and commonly used for structural and non-structural components. Indeed, these materials are often considered an alternative to the more classical metallic solution. The advantages of this solution allowed its spreading in almost all engineering fields. As it may be already known, composite materials have a very high stiffness-to-mass ratio (when compared to metallic materials), and most important their directional characteristics can be changed and adjusted in the design phase.

Nevertheless, the material properties design is both an interesting feature and a drawback. The former introduces a new and non-trivial design phase resulting in an increased workload and project intricacy. The authors have grown acquainted with the problem's complexity in the last two years through the collaboration with the Italian national institute of nuclear physics (INFN) concerning the design of composite-based supporting sandwich panels for space-borne particle physics experiments. The former project provided a real chance to face the composite design challenge and to develop the here-presented toolbox.

To increase the toolbox spread and ease its accessibility, the development took place in the Python environment.

*Corresponding author, Email address: edoardo.mancini2@studenti.unipg.it

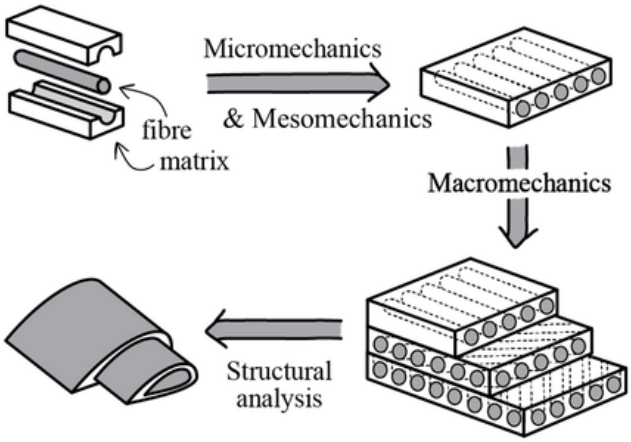


Fig. 1: Micro, meso and macro scale approach

2 The toolbox

Being the project mentioned in sec. 1 one of the first research group's approaches to the composite topic, the authors started from the basics([1], [2]) and put together a simple but effective toolbox.
The toolbox mainly divides into two parts: the micro-mechanics tools and the laminate tools. The two toolsets are completely independent of one another although the output from the first can be used as input for the second.
In composite classical literature, composite is approached on three scales: the micro-scale, the meso-scale, and the macro-scale (shown in fig. 1).

The first approach concerns the material's constituents and deals with them as merged but distinct entities. The second approach works with homogeneous composite materials laminae (making no distinction between the components). The third deals with the materials and the structures as a whole non differentiating constituents and layers. For the sake of clarity, consider a laminated wind turbine blade. The micro-mechanics of such object studies the interaction between the fiber and the matrix. The meso-mechanics concerns the stresses and deformations of the single laminae and the stress exchanges between adjacent laminae. Finally, macro-mechanics deals with overall blade parameters like the whole structural deformation. While the meso and macro mechanics are quite entangled together, the micro-mechanics is often dealt with as a separate topic, hence the split in the toolbox.
As previously stated, the micro-mechanics, and so the toolbox, starts from the materials constituents, their properties, and their relative proportions to extrapolate equivalent composite properties. To summarize, the code processes inputs on the constituents to provide equivalent composite mechanical properties.
The laminate tools start from the ply attributes (which can be either external or derived with the former tools) and the stack-up definition to compute properties as the stiffness matrices (A, B, D, and H) or the laminate equivalent orthotropic material constants. In addition, the code is capable of computing the laminate ply-by-ply stress state under a given set of actions (N, M). Lastly, the code can detect First Ply Failure and compute safety factors according to the Tsai-Wu, max stress, or max strain criteria.

Figure 2 provides an appreciative example: using as inputs the properties and percentages of the constituent, the lamination parameters, and the stress state the code provided a ply-by-ply stress analysis.

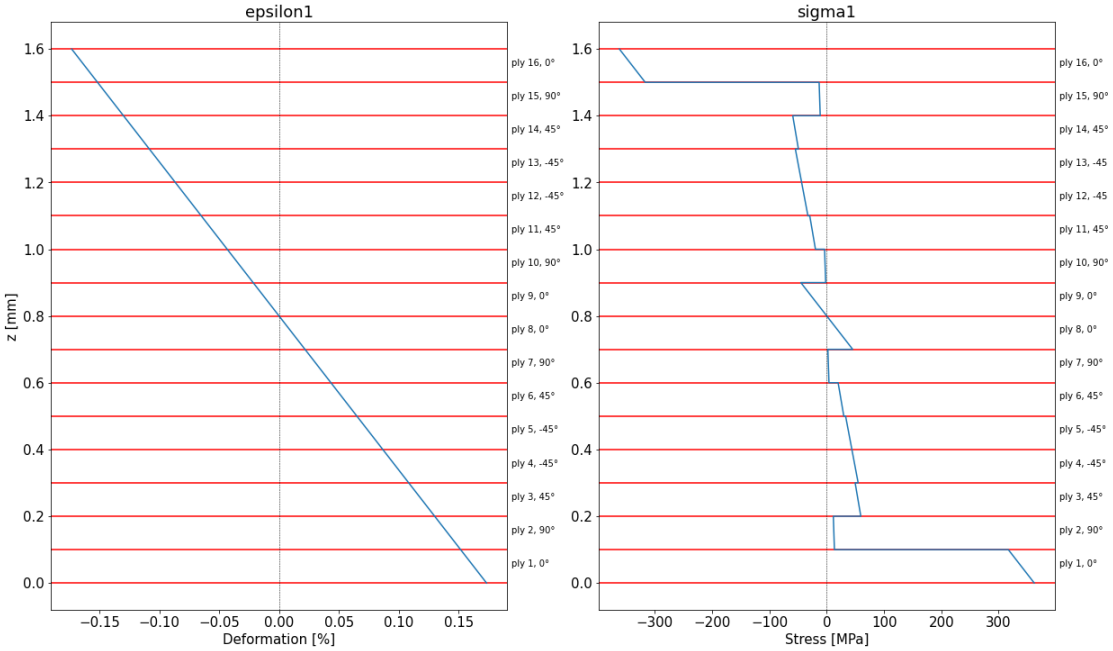


Fig. 2: Ply-by-ply stress analysis

3 The code itself

Before moving on to the toolbox applications and validation let us spend some words on the rationale behind the coding. The code was developed following the PEP-8 standard. Mainly four libraries were used: Numpy, Pandas, Scipy, and Matplotlib. The main outputs (materials and laminates) are defined as classes with various properties. Among those, there is also the list of dictionaries containing the plies characteristic. Additionally, several different methods were implemented (either internal or user accessible) and their outputs are stored as class properties. The code methods are designed to ease their expansion: when the method is called the user can choose which theory and equation he wants to use. The former choice allows not only the developers but also other users to implement in the code new, custom, or not-yet-available theories.

4 The applications

The application of such general tools can be multiple. Since covering every usage is not possible, the present section deals with the authors' application of the code and also represents a sort of introduction for the following section.
The most obvious and common usage for the code is the calculation of equivalent properties at all scales. As an example, meso scale properties (ply properties) could be needed as input for a detailed

composite structure FEA verification in which the meso-mechanics is managed at an FEA level. Moreover, equivalent material properties at a macro scale level could be used for a quicker and less refined FEA analysis.
In more sophisticated applications one could turn to the toolbox to carry on a part of the analysis. For example by using FEA stress resultants in a specific point as input to a Python routine to perform a preliminary laminate ranking (the work-flow is schematized in figure 3 .

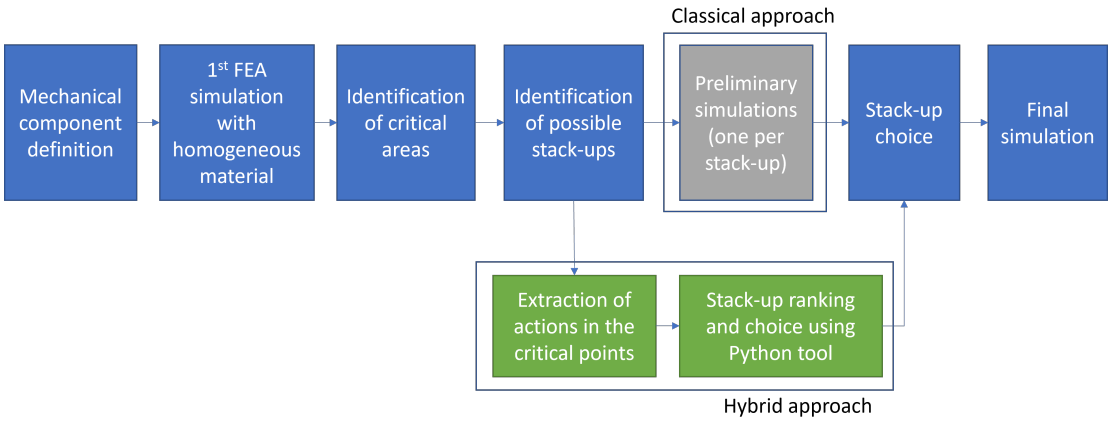


Fig. 3: Work-flow of a possible tool application

The former are merely some of the possible applications of the toolbox and come from the authors' experiences.

5 The validation

As specified in the former, the toolbox was often used together with the FEA. This tight correlation and the authors' experience with the FEA lead to the obvious usage of the latter for the toolbox validation.
For the validation, the Python toolbox and the FEA share the same settings (same material and laminate definition). After the successful validation of the toolbox-computed stiffness matrices against the ones from the FEA, the toolbox ply-by-ply stress calculation and failure capabilities were asserted against the FEA.
As an example tab. 1 reports the toolbox error at different points.

6 Conclusions

The presented document briefly describes the work done by the machine design group of the University of Perugia on an Open Source composite structural analysis toolbox. The code follows the PEP-8 standard and defines composite materials and laminates with classes. The group is planning to release the code on the GitHub platform, meanwhile the beta version can be accessed by contacting the corresponding author. The described approach allows the storage of a large variety of material and laminate properties as class attributes. Although the possible applications are various, the authors'

Tab. 1: Error committed by the toolbox on the stress calculation in different plies

Ply no. and FEA element	err. σ_x	err. σ_y	err. σ_{xy}
ply no. 0 element 1370	0.41%	0.33%	0.52%
ply no. 1 element 1370	0.40%	0.55%	0.52%
ply no. 2 element 1911	0.33%	0.31%	0.27%
ply no. 3 element 1911	0.00%	0.00%	0.00%
ply no. 4 element 2110	0.43%	0.43%	0.44%
ply no. 5 element 2110	0.33%	0.31%	0.27%
ply no. 6 element 2807	0.40%	0.55%	0.52%
ply no. 7 element 2807	0.41%	0.33%	0.52%

briefly dwelt on the toolbox-FEA interactions in terms of input provision or co-simulation. Regarding the validation, a toolbox to FEA results comparison assessed the accuracy of the developed tool.

References

[1] Jones, R. M. Mechanics of Composite Materials. Taylor and Francis, (1975).

[2] Barbero, Introduction to Material Composites Design Third Edition, CRC press (2017).

Addressing NVH Challenges with Open-Source Tools - Transfer Path Analysis and pyFBS

Domen Ocepek [a] Blaž Starc [b] Tomaž Bregar [b] Gregor Čepon [a*] Miha Boltežar [a]

[a] *University of Ljubljana, Faculty of Mechanical Engineering*
[b] *Gorenje, Predevelopment - Numerical simulations and Acoustics*

Abstract

This paper presents an open-source implementation of the transfer path analysis method as a part of the python package pyFBS. Transfer path analysis has been recently established as a reliable tool for modelling critical paths for the transmission of sound and vibration in assembly products. Its main advantages are the ability to distinguish the partial transfer path contributions and to predict the receiver's response even for its modifications. The implementation of the method within the pyFBS package is aimed at both researchers and engineers who tackle noise and vibration problems in their daily work. The intuitive use of the package is demonstrated by implementing a hybrid modelling approach aimed at evaluating different motor vibroisolation concepts for the new generation of ASKO washing machine. The approach incorporates experimental-based source characterisation of an electric motor and a numerical-based dynamic model of the virtual prototype of a washing machine. Merging the two modelling domains within the pyFBS package enables evaluation of the washing machine response in operation using transfer path analysis. In addition, the approach avoids the need for a physical prototype. Based on the calculated responses, the most suitable vibroisolation concept can be determined.

Keywords: Structural Dynamics, Python, Transfer Path Analysis, Rubber Mount Characterization

1 Introduction

The analysis of structural dynamics is an essential step in the development of high-tech mechanical systems. For modelling critical paths for the transmission of sound and vibration in an assembly, transfer path analysis (TPA) is a reliable and effective tool [1]. TPA replicates the source excitations using the set of equivalent forces acting at the interfaces between the source and the receiver structure of the assembly. Equivalent forces are an inherent property of the source and as such, transferable to any assembly with the modified receiver.

Recently, the release of the pyFBS python package [2] brought open-source implementation of the TPA and supplementary methods (such as virtual point transformation (VPT) and regularization techniques [3]) to the researches and acoustic engineers. All required methods, implemented using an object-orientated approach, can be used in an intuitive manner for consistent source characterization. In this paper, a procedure for evaluating assembly modifications is proposed that relies on the experimentally identified equivalent forces for the source, a structural admittance of the virtual prototype

*Corresponding author, Email address: gregor.cepon@fs.uni-lj.si

from a numerical model, and pyFBS package as an implementation tool. The proposed approach is applied to evaluate structural modifications by estimating the operational responses of a novel assembly in the virtual environment. In this case, it was applied to deduce the most suitable vibroisolation concept for the new generation of ASKO washing machine motor mount.

2 Theoretical background

In practise, most sources generate an input force that is impossible to measure or model. Therefore, a quantity is needed that enables a complete description of the source for a given operating condition. Treating generic structure as an assembly of source (A) and receiver (B) components (Fig. 1a), the following indirect source characterization can be applied. A set of equivalent forces acting at the interface is introduced that replicates the same responses at the receiver side as the source (Fig. 1b):

$$\mathbf{f}_2^{\mathrm{eq}} = \left(\mathbf{Y}_{42}^{\mathrm{AB}}\right)^{+} \mathbf{u}_4^{\mathrm{AB}}, \tag{1}$$

where $\mathbf{Y}_{42}^{\mathrm{AB}}$ is the admittance of the transfer paths and $\mathbf{u}_4^{\mathrm{AB}}$ are the responses at the receiver side measured while the assembly is in operation. Equivalent forces are a property of a source substructure only and are transferable to an assembly with a modified receiver where response $\mathbf{u}_3^{\mathrm{A\widetilde{B}}}$ is obtained:

$$\mathbf{u}_3^{\mathrm{A\widetilde{B}}} = \mathbf{Y}_{32}^{\mathrm{A\widetilde{B}}} \mathbf{f}_2^{\mathrm{eq}}. \tag{2}$$

(a) **(b)**

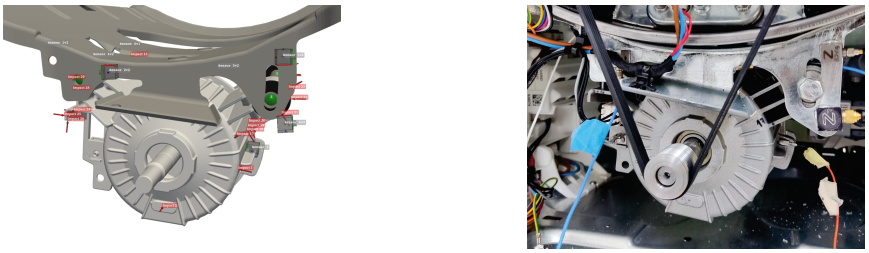
Fig. 1: Source characterization using in-situ TPA: a) assembly under operation; b) set of equivalent forces replicating the operational response.

Combining the concepts of TPA with the principles of dynamic substructuring has led to an approach in which the source is characterized using forces and moments in a virtual point (VP). The VP, typically used in frequency based substructuring, has the advantage of taking into account moments in the transfer paths that are otherwise not measurable with conventional force transducers and thus fulfill the requirement of full interface description in terms of significant degrees of freedom (DoFs).

3 Methodology

3.1 Source characterization

The first step of the proposed methodology comprises experimental identification of $\mathbf{f}_2^{\mathrm{eq}}$. The latter was performed on the existing ASKO WM75, with an operational regime of a motor run-up from 0 RPM to 20.000 RPM. Measurement campaign for acquiring structural admittance of the transfer paths was first planned using a *3D display* feature of pyFBS. Fig. 2a shows desired location of the sensors and impacts on the assembly, as well as virtual points needed for source characterization. After careful experiment planning, experimentalist in laboratory simply replicate experimental setup on a physical structure without difficulties in selecting suitable measurement locations (Fig. 2b).



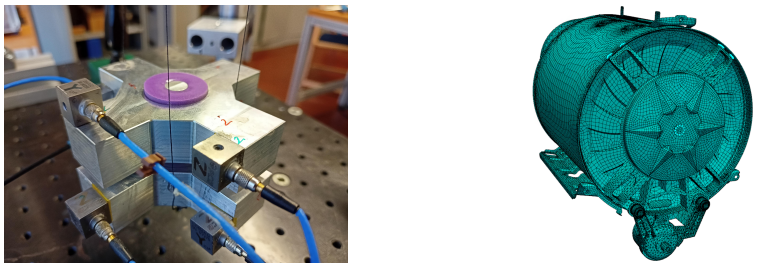
(a) (b)

Fig. 2: Experimental planning in advance: a) virtual positioning of impacts and sensors using pyFBS; b) actual measurement setup.

Next step required the use of VPT to obtain $\mathbf{Y}_{42}^{\mathrm{AB}}$. Implementation of *VPT* from pyFBS was exploited where only rigid interface deflection modes were considered. After the measurement of operational responses $\mathbf{u}_{4}^{\mathrm{AB}}$ the equivalent forces were estimated using Eq. (1).

3.2 Identification of rubber mount properties

To evaluate the performance of each vibroisolation concept, its valid numerical counterpart must first be established. For this, a material model of the used nitrile butadiene rubber (NBR) had to be identified. An inverse procedure was applied, where dynamic stiffness of the NBR was first identified experimentally on an existing motor vibroisolation (Fig.3a). The dynamic stiffness of the vibroisolation was measured by impact testing in free boundary conditions [4]. This procedure enables for characterization in all degrees of freedom (i.e., dynamic stiffness for translational and rotational degrees of freedom, again using *VPT* object from pyFBS). The experimentally identified dynamic stiffness was used to update the numerical model of the vibroisolation and thus obtain required material model.



(a) (b)

Fig. 3: a) Experimental setup for rubber mount characterization. b) Numerical model of a virtual prototype.

3.3 Response prediction

In order to simulate the responses in operation with new motor vibroisolations mounted, one must first obtain the FRFs of the novel assembly ($\mathbf{Y}_{32}^{\widetilde{\mathrm{AB}}}$). With the physical prototype unavailable at this stage of development, a numerical model of a virtual prototype was created (Fig. 3b) using ANSYS. Mass and stiffness matrices were then imported to pyFBS using *MCK* object to evaluate its operational

response, where its admittance was obtained using *FRF_ synth* function, followed by the response prediction using Eq. (2).

Each vibroisolation concept was incorporated into the numerical model and the response prediction procedure was repeated for each concept. The sum of all responses during the entire motor run-up $\mathbf{u}_3^{\widetilde{AB}}$ is presented in Fig. 4 on a 1/3 octave plot for all considered concepts.

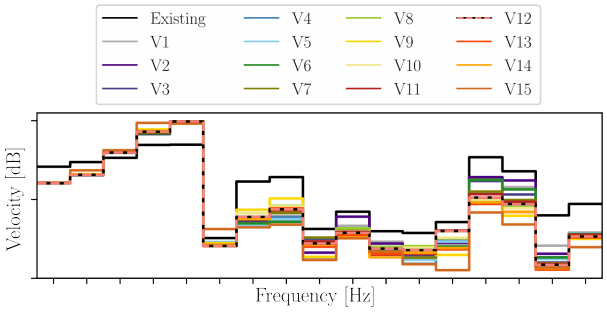
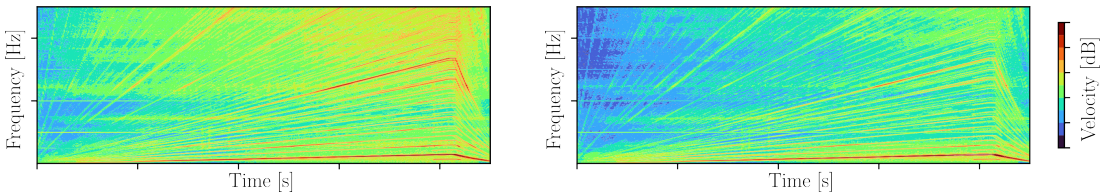


Fig. 4: Amplitude of the operating response for different vibroisolation concepts for motor run-up.

By analysing Fig. 4, a suitable solution is found with concept V12 (displayed using a dashed line). Fig. 5 compares the Campbell diagrams for the entire run-up of the BPM motor with the existing and the V12 vibroisolation concepts.



(a) **(b)**

Fig. 5: Campbell diagram: a) existing vibroisolation; b) V12 vibroisolation concept.

References

[1] M. V. van der Seijs, D. de Klerk, and D. J. Rixen, "General framework for transfer path analysis: History, theory and classification of techniques," *Mechanical Systems and Signal Processing*, vol. 68, pp. 217–244, 2016.

[2] T. Bregar, A. El Mahmoudi, M. Kodrič, D. Ocepek, F. Trainotti, M. Pogačar, M. Göldeli, G. Čepon, M. Boltežar, and D. J. Rixen, "pyfbs: A python package for frequency based substructuring," *Journal of Open Source Software*, vol. 7, no. 69, p. 3399, 2022.

[3] M. Häußler, *Modular sound & vibration engineering by substructuring*. PhD thesis, Technische Universität München, 2021.

[4] M. Häußler, S. Klaassen, and D. Rixen, "Experimental twelve degree of freedom rubber isolator models for use in substructuring assemblies," *Journal of Sound and Vibration*, vol. 474, p. 115253, 2020.

Development of an Open-Source device for the real time monitoring of fatigue life of mechanical components subjected to dynamic loads

Massimiliano Palmieri * Claudio Braccesi Filippo Cianetti

University of Perugia, Department of Engineering, Via G. Duranti 93, 06125 Perugia (Italy)

Abstract

The timely detection of faults in mechanical components is crucial for preventing system failures and minimizing maintenance costs. The relevance of this topic boosted the development of various fault detection techniques. Most of them evaluate the changes in on-filed measurements performed on the component to monitor a possible fault of the system. The present research aims to contribute to this field with a novel and economic fault detection tool for mechanical components subjected to dynamic loads. The proposed algorithm uses real-time data from a single sensor to estimate the fatigue damage and to compute the remaining life. To pursue simplicity and open accessibility, the authors implemented the algorithm in a low-cost commercial acquisition board. Lastly, an experimental campaign on a component subjected to random vibration demonstrated the tool's capability to predict fatigue damage.

Keywords: Vibration fatigue, Open Source, Structural Dynamics, Real-Time health monitoring

1 Introduction

Monitoring the structural behavior of mechanical components and systems has been widely used in engineering for many years since it offers several advantages such as an increase of safety and decrease maintenance and repair costs [1, 2]. Stuctural monitoring techniques are applied in several fields of engineer such as civil engineering [3, 4], mechanical engineering [5, 6], automotive and aerospace [7, 8]. The most commonly used approach for dynamic structures is to monitor any change in the dynamic response of the system using ultrasonic, acoustic emission or vibration analysis. In recent years, the diffusion of computer vision, artificial intelligence and machine learning has further extended the opportunities offered by damage monitoring methods to the damage localization and quantification [9, 10, 11, 12]

Recently, Cianetti [13] proposed a damage estimation method able to monitor the accumulated fatigue damage in a structure or potential damage in real-time. This method involves applying a cycle counting method and the Palmgreen-Miner damage accumulation rule [14] to a moving window of the signal. The significant advantage of this technique is that it can determine the cumulative damage on the entire structure with a single measurement, based on a numerically determined relationship.

*Corresponding author, Email address: massimiliano.palmieri@unipg.it

Indeed, assuming a linear behavior of the system, it is possible to easily determine the frequency response functions (by using a calibrated FE model) between the forcing function and any physical quantity aimed to evaluate the actual and/or potential damage for any point in the structure.
In this work, the monitoring technique proposed by Cianetti [13] has been implemented in an ad-hoc device, using easily available and low-cost components. A high-performance but low-cost processor and analog-to-digital converter were used for the hardware, while the damage calculation algorithm was developed in Python and it is freerly available.
Once the device was fabricated, it was experimentally tested on a Y-shaped component [15] subjected to vibration tests using different excitation profiles During the various tests, the potential damage was evaluated in real-time up to the predicted failure time.
The validation of the device and algorithm was then performed by comparing the estimated real-time lifetimes provided by the system with those obtained by monitoring the drop in natural frequency in post-processing analysis [15]. The results obtained demonstrate that the device provides sufficiently accurate results, with a percentage error committed always below 20%. The result of this work therefore enables the problem of identifying possible fatigue failure in a mechanical component to be addressed using a very simple algorithm and low-cost electronic components. Although the device was tested on a simple laboratory component, its potential is not diminished, as it can be applied to any mechanical component or system.

2 Evaluation of fatigue damage in real time

The evaluation of the fatigue damage of structures subjected to random vibrations in real time can be carried out by exploiting the classical methodologies in the time domain: given any measured signals, this can be processed with a cycle counting method and once the load spectrum and the S-N curve expressed in terms of the measured quantity are known, a damage accumulation law can be applied [14]. This means applying the cycle counting method over the entire time history, from the moment the component was installed to the moment the calculation is made. This approach does not go well with a real-time analysis as it would involve managing an enormous amount of data (especially for high-frequency signals), and therefore is de-facto inapplicable in real-time.

In this context, Cianetti [13] proposed a method that exploits the rainflow counting algorithm and the Palmgreen-Miner rule, but the calculation of damage is carried out only over a window, with appropriate characteristics (time length, sampling frequency). Since only small window of length ΔT are processed by the rainflow counting algorithm the real-time calculation is viable and it is possible determining the potential damage associate with the ith window as shown in equation 1.

$$d_{p_i} = \sum_{k=1}^{m_i} \frac{(n_k)_i}{\left(\frac{(x_{a_k})_i}{a}\right)^{1/b}} \tag{1}$$

In equation 1 a and b are the Wholer curve parameters, n_k is the extracted number of cycles in the ith window where k is the generic spectrum cycle counted, x_{a_k} represents the alternate value of the acquired signal while m_i is the total number of cycles counted in the window.
Once the potential fatigue damage associated to the ith window is known, the total accumulate damage at a predefined time instant D_p can be easily computed as the sum of the computed damages d_{p_i}. In this paper the authors define the damage as "potential" since it can be computed not only on stress process but on generic signals and in such case, the computed fatigue damage is potential and useful

more for comparison than to estimate the fatigue life.

3 The device

In order to have a device able to evaluate the real or potential cumulative damage within a component, it was necessary both to create an ad-hoc device capable of performing all the necessary operations and to implement an algorithm aimed to data acquisition and processing operation.

3.1 The hardware

The realized device is based on 5 elements: a transducer, a signal conditioner, an ADC converter, a buffer board (aimed to memorize the acquire data) and a processor. An accelerometer produced by PBC is used as sensor in this activity while the signal conditioner model 480E09 produced by PBC is adopted. The Raspberry Pi AD/DA Expansion Board ADS1256 with 8 channels at 24bit and sampling rate equal to 30ksps is used as ADC. For the buffer board, necessary to memorize the acquired data, the WeAct Black Pill V2.0 board has been used while the Raspberry PI4 is used as processor. The choice of these components has been made in order to minimize the total cost, that excluding the sensor and signal conditioner, is lower than 150 Euro. The final device is shown in figure 1.

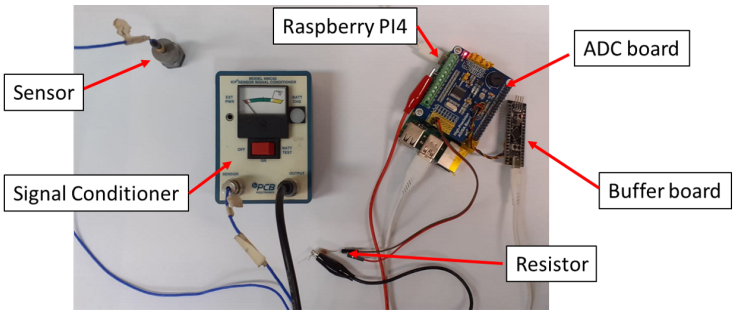


Fig. 1: The prototype of the device realized in this activity to monitor the fatigue damage in real-time

As visible from figure 1, an additional resistor was needed since the ADC converter can read only positive values in a range 0 - 3.3 Volts. By introducing an additional resistance, it was possible to insert a bias, thus a known mean value of voltage equal to 1.75V, to be removed after the acquisition. Without the resistor, negative values would be lost.

3.2 The software

The realized device should be able to acquire a signal and processing it simultaneously. To this aim, a Python algorithm has been implemented within the Raspberry PI4. This algorithm exploits the multi-threads logic. In such a way two threads operate at the same time. The first one is used to acquire portion (window) of the signal in real time while the second thread computes the fatigue damage, according to what stated in Sec. 2, of the previously acquired window. Figure 2 shows the multi-threads logic used to compute the fatigue damage in real time.

As visible from figure 2, the algorithm foresees to acquire a first window, secondly the two threads operate simultaneously. Both threads are inserted in a while loop that ends when the accumulated damage is equal or close to a predefined threshold defined critical for the analyzed component.

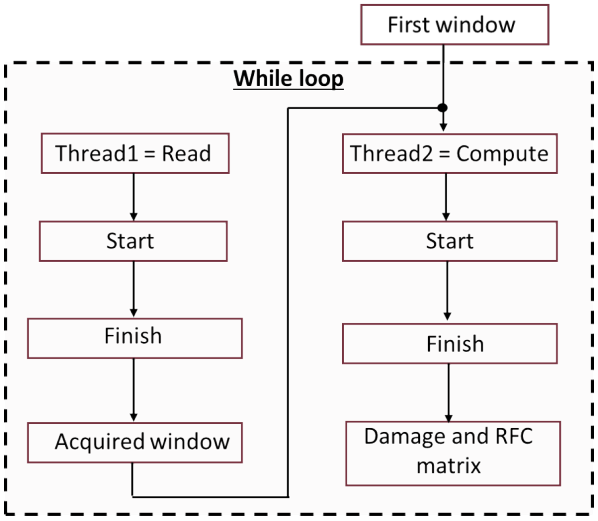


Fig. 2: Multi-thread logic to calculate fatigue damage in real-time

4 Validation of the device through vibration tests

In order to evaluate the ability of the device to operate in real time to monitor a fatigue failure induced by vibrations, experimental tests were performed on a simple laboratory component. The specimens were realized in additive manufacturing in Polylactic acid (White-Pearl PLA produced by ultimaker) for which the slope and the intercept of the S-N curve, defined as $S = aN^b$ had been determined in a previous activity [15]. These turn out to be: $a = 204.7$ MPa and $b = -0.1874$. Figure 3 shows the used experimental setup.

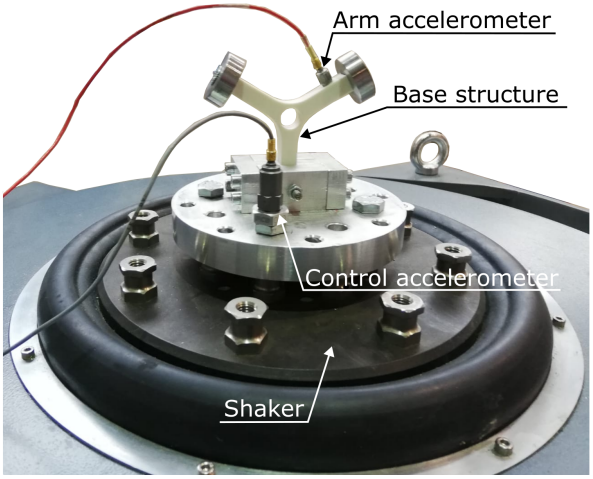


Fig. 3: Experimental setup used to validate the accuracy of the proposed device through vibration tests.

As shown in Figure 3, the chosen parameter to measure and then related to the fatigue damage was the acceleration measured on one arm of the specimen. However, in order to use this type of signal

and at the same time use the known parameters of the fatigue curve, it was necessary to determine a scale factor able to transform the acceleration measured on the arm of the specimen to the stress component at the failure point. The latter, given the simple geometry of the specimen, is in fact known and is located near the hole (Figure 3). The scale factor between acceleration and stress was determined by taking the maximum value of the frequency response function in terms of stress/acceleration, in correspondence of the resonance frequency and it results to be equal to 0.146 MPa/g. The frequency response function was determined by an experimentally calibrated finite element model. To obtain an experimental validated model, a set of vibration tests were performed and the results, in terms of frequency response function between arm and base acceleration, were compared to those numerically obtained [15]

Before proceeding with experimental tests, an intense numerical analysis has been performed in order to define the best setting parameters for the analysis. The parameters to be determine are the length of the window ΔT, the frequency sampling and the residue processing methodology [14]. The latter is indeed much important in this case since, applying the rainflow counting method to short time window there are many residues to be managed in order to determine the correct load spectrum. In particular four different residue processing methodologies have been considered: Counting the residues as half-cycles, considering all the residues as closed cycles, all the residues are discarded, all the residues are transferred to the next window. The analysis has been conducted numerically using several random signals such as non stationary signals, wide band stationary signals, variable mean value signals and bimodal processes. The results of the analysis led to define that the window time length should be not lower than 30 times the lowest frequency of the process, while (as expected) the sampling frequency should be at least ten times the maximum frequency of the signal. Moreover it comes out the best residues processing methodologies to obtain a fatigue damage comparable to the actual one is to transfer them to the next window over which the rainflow counting is further re-applied. The actual fatigue damage, used as benchmark for the fatigue damage obtained with the proposed algorithm, was computed applying the rainflow counting algorithm to the entire process, for each considered random signal, adopting the Cloorman-Seeger hypothesis [14].

Once the analysis parameters were determined, it was possible to perform the experimental test campaign. The specimen was tested with three different random loads, with a different frequency content and a different RMS. The properties of the signals used are shown in table 1. According to these three different inputs, the failure of the components were monitored with the device proposed in this activity. The results obtained are shown for each input in table 1.

Tab. 1: Comparison between fatigue life obtained with the proposed device and standard methods for three different loading conditions

Test ID	Frequency range	Input RMS	Estimated life	Real life	Percentage error
1	90-170	1	2020	2350	16.3
2	80-250	3	620	770	19.4
3	100-400	2	980	1180	17.1

To validate the accuracy of the results obtained with the proposed device, the acceleration signals were also acquired with a commercial instrument (Siemens Testlab) and were post-processed to compute the frequency drop, that foresees to monitor the resonance frequency of the specimen declaring a failure when the resonance frequency drop down below 5% compared to its initial value [15]. The results obtained with the proposed device and by monitoring the position of the natural frequency are

compared in table 1. As visible, there is a good agreement between the obtained fatigue life, indicating a good accuracy of the realized device. The committed error is in fact always lower than 20%. However, it should be noted that the device always anticipates the prediction of the component failure but this can be attributed to the stress/acceleration ratio numerically obtained.

5 Conclusions

This paper presents the development of a low-cost device for the monitoring of fatigue damage of structures subjected to vibrations in real-time. The device has been developed both in terms of hardware and software parts. While for the hardware component low-cost elements such as a Raspberry PI and a ADC converter were used, the software were developed in a Python environment in order to exploit a multithread logic needed to work in real-time. Indeed, to compute the fatigue damage in real-time, the developed software foresees two threads operating simultaneously. The first one is devoted to the acquisition of portion of the time-varying signals while the second one compute the fatigue damage on a previously acquired window. To check the accuracy of the proposed device to supply correct results in real-time, a set of vibration test were performed on a simple Y-shaped specimen excited with different random profiles until the failure occurs. The fatigue life of the component was monitored with the proposed device and compared with those obtained by monitoring the drop of the natural frequency of the specimen. The comparison shows that there is a good agreement between what obtained with the proposed device and with traditional method.
The authors have planned to release the code on the GitHub platform, meanwhile the beta version can be accessed by contacting the corresponding author.

References

[1] J. Vitola, F. Pozo, D. Tibaduiza, and M. Anaya, "A sensor data fusion system based on k-nearest neighbor pattern classification for structural health monitoring applications," *Sensors*, vol. 17, p. 417, 2017.

[2] A. Anaissi, M. Makki Alamdari, T. Rakotoarivelo, and N. Khoa, "A tensor-based structural damage identification and severity assessment," *Sensors*, vol. 18, p. 1, 2018.

[3] R. Hou and Y. Xia, "Review on the new development of vibration-based damage identification for civil engineering structures: 2010–2019," *Journal of Sound and Vibration*, vol. 419, p. 115741, 2021.

[4] J. Brownjohn, "Structural health monitoring of civil infrastructure," *Philosophical Transactions of the Royal Society A*, vol. 622, p. 365589, 2007.

[5] C. C. Ciang, J. Lee, and H. Bang, "Structural health monitoring for a wind turbine system: a review of damage detection methods," *Measurement Science and Technology*, vol. 19, p. 12, 2008.

[6] M. McGugan and L. Mishnaevsky, "Damage mechanism based approach to the structural health monitoring of wind turbine blades," *Coatings*, vol. 10, p. 1223, 2020.

[7] X. Qing, W. Li, Y. Wang, and S. Sun, "Piezoelectric transducer-based structural health monitoring for aircraft applications," *Sensors*, vol. 19, p. 545, 2019.

[8] K. Diamanti and C. Soutis, "Structural health monitoring techniques for aircraft composite structures," *Progress in Aerospace Sciences*, vol. 46, pp. 342–352, 2010.

[9] M. Azimi, A. Eslamlou, and G. Pekcan, "Data-driven structural health monitoring and damage detection through deep learning: State-of-the-art review," *Sensors*, vol. 20, p. 2778, 2020.

[10] A. Kesavan, S. John, and I. Herszberg, "Structural health monitoring of composite structures using artificial intelligence protocols," *Journal of Intelligent Material Systems and Structures*, vol. 19, pp. 63–72, 2008.

[11] Y. Yan, L. Cheng, Z. Wu, and L. Yam, "Development in vibration-based structural damage detection technique," *Mechanical Systems and Signal Processing*, vol. 21, pp. 2198–2211, 2007.

[12] A. Alavi, H. Hasni, N. Lajnef, K. Chatti, and F. Faridazar, "An intelligent structural damage detection approach based on self-powered wireless sensor data," *Automation in Construction*, vol. 62, pp. 24–44, 2016.

[13] F. Cianetti, "How to experimentally monitor the fatigue behaviour," *Strojniski vestnik–Journal of Mechanical Engineering*, vol. 66, pp. 557–566, 2020.

[14] J. Collins, *Failure of Materials in Mechanical Design*. Wiley - New Yorks, 1981.

[15] M. Palmieri, G. Zucca, G. Morettini, L. Landi, and F. Cianetti, "Vibration fatigue of fdm 3d printed structures: The use of fre-quency domain approach," *Materials*, vol. 15, 2022.

PyOMA and PyOMA_GUI: A Python module and software for Operational Modal Analysis

Dag Pasquale Pasca[a] * Angelo Aloisio[b] Marco Martino Rosso[c] Stefanos Sotiropoulos[c]

[a] *Norsk Treteknisk Institutt,*
Børrestuveien 3, 0373, Oslo, Norway

[b] *Department of Civil, Construction-Architectural and Environmental Engineering,*
Università degli Studi dell'Aquila, Piazzale Pontieri, 1, 67040, Monteluco di Roio, L'Aquila, Italy

[c] *Department of Structural, Geotechnical and Building Engineering,*
Politecnico di Torino, Corso Duca degli Abruzzi, 24, 10129, Torino, Italy

Abstract

Operational modal analysis (OMA) comprises several techniques and algorithms for estimating the dynamic characteristics of a structure in operational conditions from its vibration response. OMA methods has been spreading in the last years due to multiple advantages compared to input–output identification methods, especially for large and massive civil-engineering structures. In the current work the authors present the implementation of a Python module named PyOMA and its Graphical User Interface (GUI) PyOMA_GUI. This software provides a user-friendly framework, for the first time in the Python environment, for estimating the experimental modal parameters (natural frequencies, mode shapes, damping ratios) of a structure from output-only vibration measurements in operational conditions.

Keywords: Open Source, Structural Dynamics, Python, Operational Modal Analysis, vibration-based identification

1 Statement of Need

The practice of operational modal analysis (OMA) on civil structures and infrastructures has been growing significantly in the last decades [1]. OMA allows estimating the modal properties (natural frequencies, mode shapes and damping ratios) from output-only ambient vibration tests while the structure is in its operating conditions. Measuring the input ambient excitations for a civil engineering structure is nearly impossible. Neither can it adequately excite them using conventional devices, such as impact hammers and shakers. On the other hand, output vibration data acquired through accelerometers are relatively easy to obtain. In OMA, the deterministic knowledge of the input excitation is replaced by the assumption that the input is a realization of a stochastic process. Accordingly, OMA and output-only dynamic identification are considered synonyms.

*Corresponding author, Email address: dpa@treteknisk.no

To this date, a few commercial software implements OMA methods. The most known presumably are ARTeMIS [2], by Structural Vibration Solutions, and MACEC, a Matlab toolbox for modal testing and OMA [3]. However, to the author's best knowledge, there is no Python module nor any other open-source complete toolbox to perform output-only OMA. For this reason, the authors have developed the present PyOMA module. The API for PyOMA provides a set of functions for a quick and straightforward estimation of the natural frequencies, mode shapes and damping using the experimental data recorded by the user. Specifically, the user needs to specify only a minimal amount of input parameters in addition to the measurement data. For a complete description of the functionalities, please refer to the documentation page. The flowchart in Fig. 1 shows the general architecture of PyOMA. Furthermore, the greatest impact is provided by the development of the PyOMA_GUI, for which a general overview of the main functionalities is depicted in Fig. 2. The software provides a graphical user interface (GUI) approach designed to be adopted by both researchers, civil engineers and practitioners without requiring any Python expertise or Python coding knowledge prerequisite. The PyOMA_GUI aims at increasing the impact of the current open-source Python OMA module, which has already been used in several applications, as proved by several scientific publications: [4, 5, 6, 7, 8, 9, 10, 11, 12, 13].

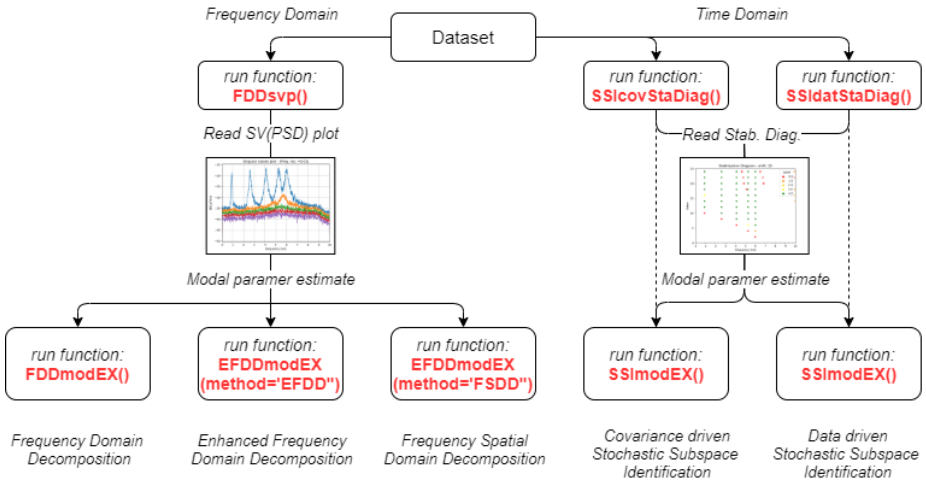


Fig. 1: "PyOMA" module flowchart.

2 Main Features

The pyOMA module offers the following identification techniques:

1. Frequency Domain Decomposition [14];
2. Enhanced Frequency Domain Decomposition [15];
3. Frequency Spatial Domain Decomposition [16];
4. Covariance driven Stochastic Subspace Identification [17];
5. Data driven Stochastic Subspace Identification [18];

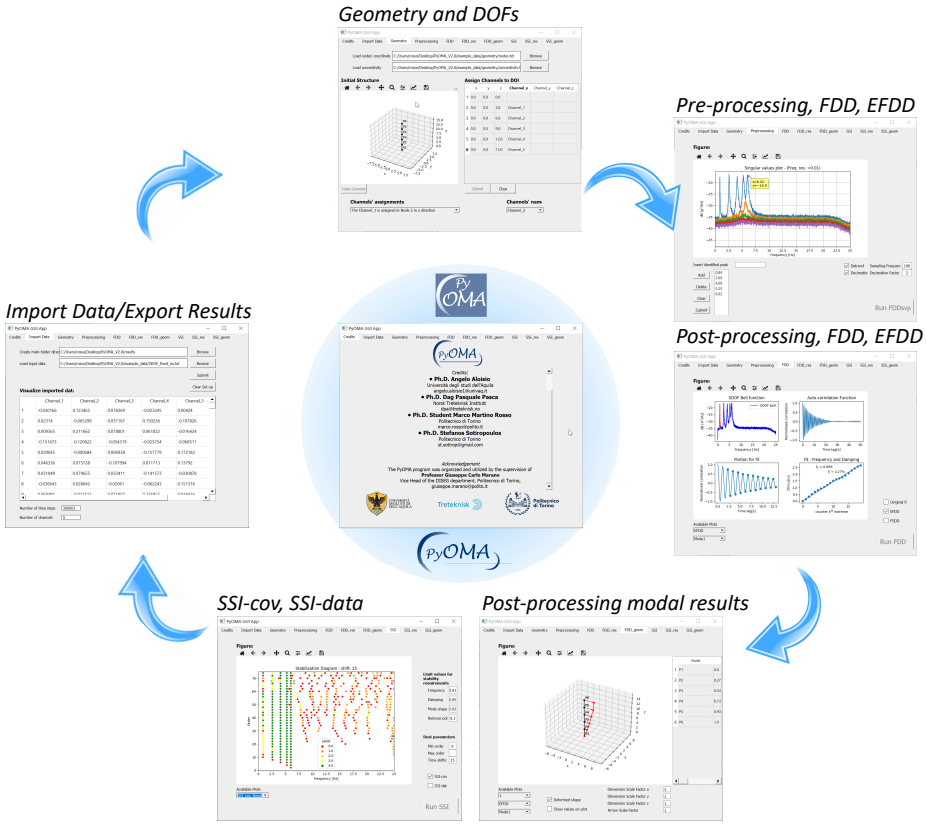


Fig. 2: "PyOMA_GUI" graphical user interface software general overview.

Through the implemented functions, it is possible to estimate the modal parameters of a civil structure using dynamic identification techniques derived from SSI and FDD [1].

In addition, the authors provided a graphical user interface software version of the current module, called PyOMA_GUI. The PyOMA_GUI aims to improve the appeal of the existing open-source Python OMA module, which has already been used in several applications. Not secondarily, the graphical user interface does not require any Python expertise or Python coding knowledge prerequisite. A general overview of the software functionalities is depicted in Fig. 2.

3 Conclusions

The authors developed a new python module and open-source software, PyOMA and PyOMA-GUI, respectively, to perform dynamic identification of structures from output-only vibration measurements. To the authors' knowledge, there are no currently structured python modules for Operational Modal Analysis (OMA). OMA represents a standard practice in the diagnosis phase of structures within any structural health monitoring paradigm. By providing a structured open-source software and framework to perform OMA, the authors hope to provide the scientific community and practitioners with a fundamental tool. For this reason, the authors developed a graphical user interface version of this python module to increase the impact of the software among users without particular expertise in python coding.

References

[1] C. Rainieri and G. Fabbrocino, "Operational modal analysis of civil engineering structures," *Springer, New York*, vol. 142, p. 143, 2014.

[2] S. V. Solutions, "Artemis extractor: Ambient response testing and modal identification software, user's manual," 2001.

[3] E. Reynders, M. Schevenels, and G. De Roeck, "Macec 3.2: A matlab toolbox for experimental and operational modal analysis," *Department of Civil Engineering, KU Leuven*, 2014.

[4] R. Alaggio, A. Aloisio, E. Antonacci, and R. Cirella, "Two-years static and dynamic monitoring of the santa maria di collemaggio basilica," *Construction and Building Materials*, vol. 268, p. 121069, 2021.

[5] A. Aloisio, A. Di Pasquale, R. Alaggio, and M. Fragiacomo, "Assessment of seismic retrofitting interventions of a masonry palace using operational modal analysis," *International Journal of Architectural Heritage*, pp. 1–13, 2020.

[6] A. Aloisio, L. D. Battista, R. Alaggio, E. Antonacci, and M. Fragiacomo, "Assessment of structural interventions using bayesian updating and subspace-based fault detection methods: The case study of s. maria di collemaggio basilica, l'aquila, italy," *Structure and Infrastructure Engineering*, vol. 17, no. 2, pp. 141–155, 2021.

[7] A. Aloisio, D. P. Pasca, R. Alaggio, and M. Fragiacomo, "Bayesian estimate of the elastic modulus of concrete box girders from dynamic identification: A statistical framework for the a24 motorway in italy," *Structure and Infrastructure Engineering*, pp. 1–13, 2020.

[8] A. Aloisio, I. Capanna, R. Cirella, R. Alaggio, F. Di Fabio, and M. Fragiacomo, "Identification and model update of the dynamic properties of the san silvestro belfry in l'aquila and estimation of bell's dynamic actions," *Applied Sciences*, vol. 10, no. 12, p. 4289, 2020.

[9] A. Aloisio, E. Antonacci, M. Fragiacomo, and R. Alaggio, "The recorded seismic response of the santa maria di collemaggio basilica to low-intensity earthquakes," *International Journal of Architectural Heritage*, pp. 1–19, 2020.

[10] A. Aloisio, R. Alaggio, and M. Fragiacomo, "Dynamic identification and model updating of full-scale concrete box girders based on the experimental torsional response," *Construction and Building Materials*, vol. 264, p. 120146, 2020.

[11] A. Aloisio, R. Alaggio, and M. Fragiacomo, "Time-domain identification of the elastic modulus of simply supported box girders under moving loads: Method and full-scale validation," *Engineering Structures*, vol. 215, p. 110619, 2020.

[12] I. Capanna, R. Cirella, A. Aloisio, R. Alaggio, F. Di Fabio, and M. Fragiacomo, "Operational modal analysis, model update and fragility curves estimation, through truncated incremental dynamic analysis, of a masonry belfry," *Buildings*, vol. 11, no. 3, p. 120, 2021.

[13] A. Aloisio, D. P. Pasca, L. D. Battista, M. M. Rosso, R. Cucuzza, G. C. Marano, and R. Alaggio, "Indirect assessment of concrete resistance from fe model updating and young's modulus

estimation of a multi-span psc viaduct: Experimental tests and validation," *Structures*, vol. 37, pp. 686–697, 2022.

[14] R. Brincker, L. Zhang, and P. Andersen, "Modal identification of output-only systems using frequency domain decomposition," *Smart materials and structures*, vol. 10, no. 3, p. 441, 2001.

[15] R. Brincker, C. E. Ventura, and P. Andersen, "Damping estimation by frequency domain decomposition," in *Proceedings of IMAC 19: A Conference on Structural Dynamics: februar 5-8, 2001, Hyatt Orlando, Kissimmee, Florida, 2001*, pp. 698–703, Society for Experimental Mechanics, 2001.

[16] L. Zhang, T. Wang, and Y. Tamura, "A frequency–spatial domain decomposition (fsdd) method for operational modal analysis," *Mechanical systems and signal processing*, vol. 24, no. 5, pp. 1227–1239, 2010.

[17] B. Peeters and G. De Roeck, "Reference-based stochastic subspace identification for output-only modal analysis," *Mechanical systems and signal processing*, vol. 13, no. 6, pp. 855–878, 1999.

[18] P. Van Overschee and B. De Moor, *Subspace identification for linear systems: Theory—Implementation—Applications*. Springer Science & Business Media, 2012.

An open-source vibration based structural health monitoring approach

Ana S. A. Pereira Andreia Mendes Tiago A. N. Silva*

NOVA School of Science and Technology, UNIDEMI
Department of Mechanical and Industrial Engineering
Campus da Caparica, 2829-516 Caparica, Portugal

LASI – Laboratório Associado de Sistemas Inteligentes
4800-058 Guimarães, Portugal

Abstract

Vibration based structural health monitoring is one of the most versatile approaches for damage assessment of large structures. Monitoring large structures encompasses several aspects related to structural dynamics, signal acquisition and processing, and challenges from damage detectability to measuring limitations. Open-source software tools are available in the referred context. This work aims at presenting a framework for vibration based structural health monitoring fully implemented in Python. Hence, our proposal has the goal to statistically compare experimental and simulated modal properties over time, in order to identify changes on the experimental modal properties, namely natural frequencies. Regarding the computational implementation, our proposal requires the ability: to manage and post-processing time responses, taken from accelerometers; to perform operational modal identification, using python implementations of stochastic subspace identification and enhanced frequency domain decomposition methods; to generate and update finite element models; to expand experimental responses to build virtual sensors, based on modal and transmissibility approaches; and to compute a damage detectability metric based on the area under the receiver operating characteristic curve. Note that the use of virtual sensors enhances the spatial resolution of the experimental acquisition setup, contributing for a better performance of damage localization techniques. Details on the proposed approach are given, including references to the used Python software packages and developed programming strategies. An experimental example is presented to show the applicability of the framework in the context of structural health monitoring.

Keywords: Structural Health Monitoring, Damage Identification, Data Expansion, Virtual Sensing, Python

*Corresponding author, Email address: tan.silva@fct.unl.pt

Component Mode Synthesis through pyFBS: Estimating Dynamic Properties of Assemblies with Component Models

Miha Pogačar Gregor Čepon * Miha Boltežar

University of Ljubljana, Faculty of Mechanical Engineering

Abstract

Estimating the dynamic properties of assemblies in experimental settings is a complex task that poses significant challenges. The accurate determination of these properties is crucial for the advancement of high-performance systems and components. This paper introduces a workflow that utilizes `pyFBS`, an open-source Python package, to estimate the dynamic properties of assemblies in the modal domain. `pyFBS` offers a range of useful functionalities that assist in various stages of the analysis. A case study is presented, which involves coupling a numerical substructure with a simulated experimental substructure. The workflow begins by importing the geometry and system matrices (mass, stiffness) of all substructures from commercial software. It then proceeds with interactive 3D visualization. For the numerical substructure, a modal model is created by spatially and modally reducing the imported numerical data. On the other hand, a noisy admittance matrix is synthesized for the simulated experimental substructure. A multi-reference modal identification technique is then employed to estimate the modal parameters. The next step in the workflow is the coupling procedure, which addresses the interface problem. Constraints are properly weakened to prevent interface locking, and established primal substructuring formulations are used to estimate the modal parameters of the assembly. Finally, the estimated coupled response is compared to the reference assembled model using `pyFBS` functionalities for graphical and quantitative comparison. The presented workflow provides an efficient and accurate method for estimating the dynamic properties of assemblies in the modal domain. The extension of the pyFBS library for modal substructuring opens up opportunities for researchers in a broader range of substructuring and related fields.

Keywords: Component mode synthesis, pyFBS, Open Source, Structural Dynamics, Python

1 Statement of Need

In the realm of structural dynamics, the sub-field of dynamics substructuring is witnessing significant advancements. However, it poses challenges and is rapidly evolving, lacking a comprehensive knowledge base of state-of-the-art methods. Initially, an open-source project was launched with the objective of developing a Python package for Frequency Based Substructuring, referred to as pyFBS. The aim was to facilitate collaborative progress among various research groups working in this subfield. As the project progressed, it became evident that the library needed expansion to encompass the interaction between different substructuring domains (Fig. 1). Recently, additional support functions have been incorporated to facilitate modal substructuring, including multi-reference experimental modal analysis. By expanding the package, it enables the incorporation of different numerical and experimental models, effectively leveraging the benefits of combining various substructuring domains.

*Corresponding author, Email address: gregor.cepon@fs.uni-lj.si

Fig. 1: New pyFBS logo, announcing expanded support beyond frequency-based methods.

The pyFBS was developed as part of a collaboration between the Laboratory for Dynamics of Machines and Structures at the University of Ljubljana, Faculty of Mechanical Engineering, and the Chair of Applied Mechanics at the Technical University of Munich. More information can be found on the documentation site [1] and the GitLab repository [2]. Additionally, a paper about pyFBS was submitted to the Journal of Open Source Software. If you are using pyFBS in your scientific research, please consider citing the paper [3].

2 Component mode synthesis through pyFBS

The main focus of this paper is dynamic substructuring in the modal domain, specifically demonstrated through a case study involving the simulation of coupling between two substructures. Substructure A, resembling a valve shape, is represented as a numerical model component, while Substructure B, a pipe-like structure with a cantilever support, is simulated as an experimental model (Fig. 2). This section solely concentrates on the practical implementation aspects of the process, without delving into theoretical discussions.[1]

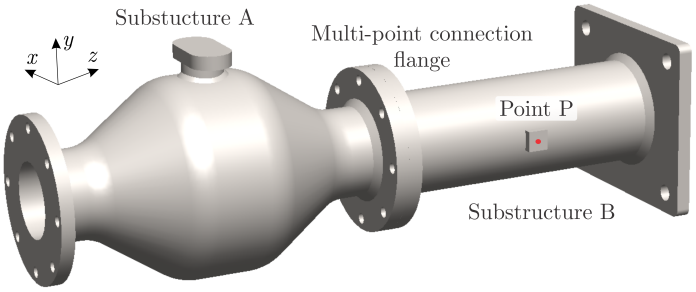


Fig. 2: Case study example as an assembly on substructures A (left) and B (right) with multi-point connection flange and reference point P.

2.1 Using pyFBS for experimental-analytical modal substructuring

Initially, we assume access to either the numerical model obtained from FEM software or the experimental model, usually provided as an admittance matrix. Additionally, we assume knowledge of the output and input DoFs locations and orientations, as well as the interface DoFs locations. The process

[1] However, for readers interested in the theoretical aspect of the methodology used, a detailed study can be found in [4].

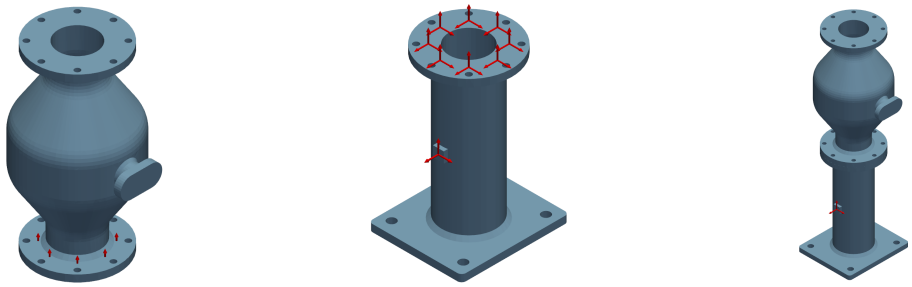
starts by importing this data for substructures A, B, and the reference assembly AB. Then, we initialize the MK class for each of the substructures:

```
# Import channel and impact data
df_chn_A = pd.read_excel(r'./chn_imp_data.xlsx', sheet_name='Channels_A')
df_chn_B = pd.read_excel(r'./chn_imp_data.xlsx', sheet_name='Channels_B')
df_imp_B = pd.read_excel(r'./chn_imp_data.xlsx', sheet_name='Impacts_B')
df_chn_AB = pd.read_excel(r'./chn_imp_data.xlsx', sheet_name='Channels_AB')

# Import mass, stiffness and result data from FEM models
MK_A = pyFBS.MK_model(r'./A/file.rst', r'./A/file.full', scale=1e3, no_modes=20, read_rst=1)
MK_B = pyFBS.MK_model(r'./B/file.rst', r'./B/file.full', scale=1e3, no_modes=20, read_rst=1)
MK_AB=pyFBS.MK_model(r'./AB/file.rst',r'./AB/file.full', scale=1e3, no_modes=20, read_rst=1)
```

Listing 1: Preprocessing - data import.

The resulting geometry, as well as the locations and orientations of channels and impacts, can be visualized and verified using an interactive 3D display (Fig. 3).



(a) **(b)** **(c)**

Fig. 3: Interactive 3D graphical interface: a) Substructure A, b) Substructure B, c) Substructure AB.

In the second step, pyFBS generates numerical and simulated experimental modal models for substructures A and B. These models are utilized for FRF generation and experimental estimation of modal parameters. The modal parameter estimation involves manual pole selection using a stabilization diagram (Fig. 3).

```
# Generate numerical modal model for substructure A
eig_val_A, xi_A, eig_vec_A = MK_A.transform_modal_parameters(df_chn_A, limit_modes = 20,
    modal_damping = 0.001, return_channel_only = True)

# Generate noisy FRFs to simulate experimental addmitance matrix of substructure B
MK_B.FRF_synth(df_chn_B, df_imp_B, f_start = 1, f_end = 8200, f_resolution = 1, limit_modes
    = 20, modal_damping = 0.001, frf_type = "accelerance")
MK_B.add_noise(n1 = 5e-5, n2 = 5e-5, n3 = 5e-5 ,n4 = 5e-5)

# Perform multi-reference modal identification
_id = pyFBS.modal_id(MK_B.freq, MK_B.FRF_noise)
_id.pLSCF(max_order=30)
_id.pLSFD(frf_type = 'accelerance', assume_proportional = True)
_id.normalize(output_dp_ind = [24,25,26], input_dp_ind = [4,5,6], check_dp = True)
```

Listing 2: Generation of modal models and modal parameter estimation.

The third step involves performing the coupling procedure after primal modal substructuring with weakened singular vector constraints, following the methodology outlined in [4]. This procedure results in the estimated natural frequencies and mode shapes of the assembly.

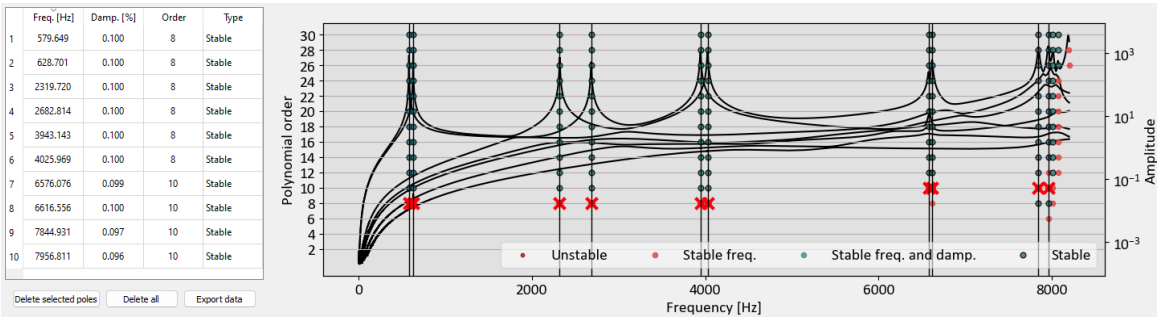


Fig. 4: Stability chart.

Once modal superposition using `pyFBS.FRF_synth` or `pyFBS.custom_FRF_synth` is performed, the FRF results can be quantitatively compared using the function `pyFBS.coh_frf`. The frequency response functions can also be graphically displayed using `pyFBS.plot_frequency_response`, as shown in the well-matched comparison of reference and coupled FRFs in Fig. 5 (coh. value 0.95).

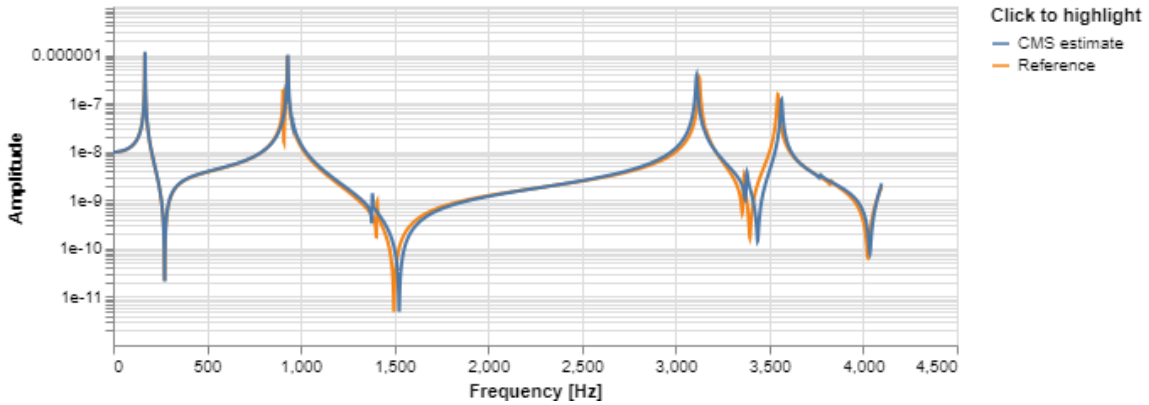


Fig. 5: Reference and estimated FRF coupled amplitude spectra in point P (y-direction driving point).

3 Conclusion

`pyFBS` is an open-source data tool that greatly enhances modal substructuring. It efficiently assists in multiple aspects as demonstrated in successful coupled response estimation in the case study.

References

[1] "pyFBS documentation." `https://pyfbs.readthedocs.io/en/latest/`.

[2] "pyFBS Gitlab repository." `https://gitlab.com/pyFBS/pyFBS`.

[3] T. Bregar, A. El Mahmoudi, M. Kodrič, D. Ocepek, F. Trainotti, M. Pogačar, M. Göldeli, G. Čepon, M. Boltežar, and D. J. Rixen, "pyFBS: A python package for frequency based substructuring," *Journal of Open Source Software*, vol. 7, no. 69, p. 3399, 2022.

[4] M. Pogačar, G. Čepon, and M. Boltežar, "Weakening of the multi-point constraints in modal substructuring using singular value decomposition," *Mechanical Systems and Signal Processing*, vol. 163, p. 108109, 2022.

A high level API for integrating vibrational data into machine learning algorithms

Guillermo Reyes-Carmenaty * Josep Font-Moré Marco A. Pérez

Universitat Ramón Llull - IQS School of Engineering

Abstract

A new python library called pymodal has been developed to facilitate the manipulation and preparation of vibrational data for machine learning applications. The library has a collection of classes, each designed to handle different quantities derived from vibrational data. Each class comes with a set of methods that enable users to modify the data contained within an instance (e. g. change the sampling rate of an acceleration measurement or the frequential resolution of a frequency response function) and to transform the data to another class (i. e. going from acceleration measurements of excitations and responses to frequency response functions, and from there to a modal assurance criterion matrix or any of the frequency domain assurance criteria). The highly modular nature of the library makes it so that new indicators and derived quantities can be added with full compatibility with the core features of the library. Apart from handling individual measurements, the library also allows users to create collections of measurements. These collections maintain the spatial and vibrational information in a labelled and ordered manner, making them easy to manipulate for machine learning applications. Additionally, the library includes methods for data augmentation and mass manipulation of all measurements contained in these collections . These collections are designed to store the data on disk so that the code can handle the big amounts of data machine learning applications need. All the code has been released under a MIT licence and is available in a GitHub repository, and extensive automatic testing has been prepared to ensure the good functioning of the code upon further development. Summarising, the library aims to provide a comprehensive toolbox that makes it easier for researchers and data scientists to work with vibrational data.

Keywords: vibration, modal analysis, experimental mechanics, machine learning, python, open source

1 Introduction

Structural Health Monitoring (SHM) is a field of study that aims to continuously monitor the integrity of structures in real-time and detect any signs of damage or degradation. Vibrational analysis is a commonly used technique in SHM, which involves monitoring the vibrations of a structure to identify any changes in behavior that may indicate damage [1]. Machine Learning (ML) algorithms have been used in vibration-based SHM applications since the 90s, and interest in ML applications relating to vibration-based SHM has only grown over the years [2].

*Corresponding author, Email address: second.author@example.com

This paper presents the development of a Python library focused on providing tools for preparing vibrational data for use in SHM machine learning applications. The library includes ways of storing labelled vibrational data in various forms, tools for manipulating said data and performing data augmentation operations and ways to interface with deep learning models built on pytorch.

There have been open source libraries which have dealt with the matter of treating vibrational data for modal and spectral analysis. Specifically, the openmodal project [3], integrated by the pyuff library, dedicated to dealing with UFF files, the pylvm library, similarly dealing with lvm files, and the pyFRF library, dealing with moving from the temporal to the frequential domain and manipulating the resulting frequency response function (FRF). Furthermore, the sdypy-EMA library and its predecessor, the pyEMA library [4], dealing with pole extraction and reconstruction given an FRF matrix, as well as modes and modal shapes. These libraries have been extremely useful and foundational to the library here presented.

The pymodal library is not concerned with recording and reading experimental results or analysis of the signal itself, since this is something that has been already adressed by the aforementioned open source software packages. The goal of pymodal is the structuring and en-masse storage and manipulation of vibrational data in various forms, not only FRFs, as will be detailed in 3. In order to do so, it is structured according to what has been exposed also in section 3. This extended abstract describes the intended and desired state of the library, but some of the described features are currently under development. This will be discussed in section 2.

2 Current State and Road Map

Currently pymodal 0.4 is a version of the library developed with a diferent philosophy in mind: the generation and manipulation of FRFs, as well as the transformation of FRFs into other useful vibrational data-related features (damage indicators such as the MAC, the FDAC or the CFDAC). For that purpose, it incorporated not only an FRF collection object capable of manipulating all FRFs contained within it, but also many standalone functions for converting said FRFs to othe vibrational features. It also incorporated a module with a high level pyansys API to generate spectra using ANSYS.

Given the use of propietary software for spectra generation and the narrow scope of the library, it was considered that an overhaul of the previous library and a more structured approach to goals that have not yet been addressed in the aforementioned open source libraries. Currently the library lack some key features still, namely solving the issue of storing great quantities of data and loading it into memory in an efficient way, as well as interfacing with pytorch.

3 Structure and Applications

Overview pymodal is structured so that there are parent classes for vector type vibrational features, such as the FRF (although the FRF is a matrix, it is often treated as a collection of vectors. That is the treatment afforded to it in the pymodal library); and bidimensional vibrational features, such as the CFDAC and FDAC. The library then has child classes for

every implemented feature, with methods particularized for that feature, as well as ways to convert one feature to another when possible.

The library also has a collection parent class. This collection parent class works in a way that allows for related vibrational features to be put in a container which can easily manipulate them en-masse, as well as keep them labelled and orderly for their use in ML applications. Last but not least, all these methods and classes are supported by standalone functions in a utilities file.

As has been stated in the introduction, the goals of this library are threefold: first of all the storage of labelled vibrational data collections, which is achieved through the use of hdf5 files; the manipulation of vibrational data using methods preprogrammed in the library, as well as some data augmentation routines found in other open source libraries such as audiomentations; and it's interfacing with AI and ML libraries such as TensorFlow [5] and PyTorch [6].

4 Conclusions

In conclusion, the pymodal library is an attempt to provide a framework within which vibrational data can be easily prepared and used for ML applications, and while it is still under development, its basic characteristics still makes it an addition worth having when dealing with ML applications relating to vibrational data.

5 Acknowledgements

This work has been partially supported by the Agency for Management of University and Research Grants (AGAUR) through the project 2019 DI 65 and by COMSA through the project Nuclis RD17-1-0105.

References

[1] A. Rytter, Vibrational Based Inspection of Civil Engineering Structures. PhD thesis, Aalborg University, Aalborg, 1993.

[2] O. Avci, O. Abdeljaber, S. Kiranyaz, M. Hussein, M. Gabbouj, and D. J. Inman, "A review of vibration-based damage detection in civil structures: From traditional methods to Machine Learning and Deep Learning applications," Mechanical Systems and Signal Processing, vol. 147, p. 107077, 2021.

[3] "OpenModal - Accelerating the implementation of your ideas."

[4] K. Zaletelj, T. Bregar, D. Gorjup, and J. Slavič, "ladisk/pyema: v0.24," Sept. 2020.

[5] T. Developers, "Tensorflow," Mar. 2023. Specific TensorFlow versions can be found in the "Versions" list on the right side of this page.
See the full list of authors on GitHub.

[6] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in Advances in Neural Information Processing Systems 32, pp. 8024–8035, Curran Associates, Inc., 2019.

Python tool to simulate thermo-mechanically induced interface stress in the solder connections of chips

Ward Rottiers * Babak Bozorgmehri Frank Naets

Department of Mechanical Engineering, KU Leuven, Belgium 'Flanders Make@KU Leuven

Abstract

The proposed substructuring approach in this work aims to circumvent the computational challenges associated with the dynamic analysis of coupled thermo-mechanical circuit models, which commonly have large numbers of degrees-of-freedom. The approach involves dividing the chip, solder, and PCB meshes into various substructures using Simcenter, and developing a novel assembly approach for the Reduced Order Models (ROMs) that is tailored to the structure of circuit boards. A novel coupling scheme is also developed to account for the non-conforming interface conditions between the different substructures using a penalty-based enforcement of the interface conditions. The resulting model can be used in an eigenvalue analysis or a time-domain simulation to predict the dynamic behavior of the system. The developed methodology is implemented in Python and utilizes the NX Open API to interact with Simcenter 3D.

Keywords: Substructuring, Interface reduction, Thermo-mechanical modelling, Python, Simcenter API

1 Introduction

A chip and printed circuit board (PCB) consists of many materials such as silicon (semi-conductor die), thermoset polymer (molding compound), copper (carrier), FR4 substrate (PCB substrate) and solder (connections). Their linearized material parameters such as Young's modulus (E) and coefficient of thermal expansion (CTE) cover a wide range from 1 to $169GPa$ and 2.5 to $250^{\mu mK}/_{m}$, respectively. The chip and PCB assembly is often subjected to global thermal cycling from environment changes, introducing the failure mode of solder interface rupture due to thermo-mechanically induced stress [1].

The dynamic analysis of coupled thermo-mechanical circuit models, commonly requires large computational loads. As these systems consist of multiple components, each of which commonly consist of a large number of degrees-of-freedom (DOFs), the resulting models easily have several million DOFs. Furthermore, the chip and PCB are often designed by different companies resulting in the exchange of (reduced) models which obfuscate intellectual property. Taking into account both challenges, a substructuring approach is proposed in this work.

*Corresponding author, Email address: ward.rottiers@kuleuven.be

2 Methodology

2.1 Assembled model equations

Starting from the Reduced Order Models (ROM) for the separate components, a novel assembly approach for the ROMs has been developed which is particularly tailored to the structure of circuit boards, as conceptually outlined in figure 1.

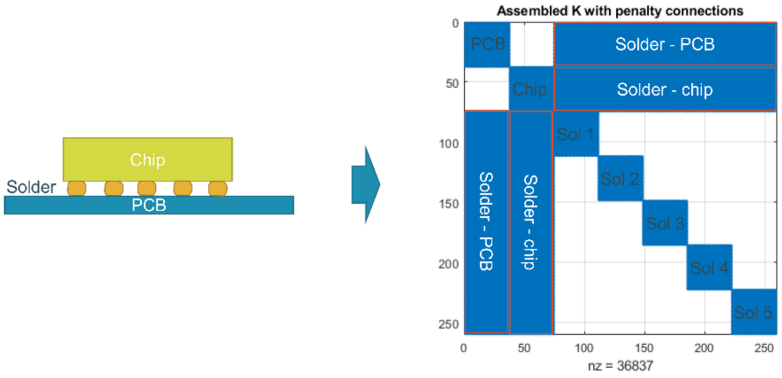


Fig. 1: Assembled stiffness matrix of the circuit board and chip connected via five solder balls.

To account for the non-conforming interface conditions between the different substructures, a novel coupling scheme has been developed starting from a penalty-based enforcement of the interface conditions between the different substructures. The corresponding Lagrangian L for a system consisting of two substructure A and B can be written as

$$L = \frac{1}{2}\left(\ddot{q}^A\right)^T \tilde{M}^A_{uu}\ddot{q}^A + \frac{1}{2}\left(\ddot{q}^B\right)^T \tilde{M}^B_{uu}\ddot{q}^B - \frac{1}{2}\left(q^A\right)^T \tilde{K}^A_{uu}q^A + \frac{1}{2}\left(q^B\right)^T \tilde{K}^B_{uu}q^B \\ - \left(q^A\right)^T \tilde{K}^A_{uT}T^A - \left(q^B\right)^T \tilde{K}^B_{uT}T^B + \left(q^A\right)^T \tilde{f}^A + \left(q^B\right)^T \tilde{f}^B - E_{penalty}, \quad (1)$$

with indices u and T referring to deformation and temperature, respectively.

The following equation impose the connection constraint on the interfaces between the components by means of penalty method

$$E_{penalty} = \frac{1}{2}p\left(\tilde{C}^B_{uu}q^B - \tilde{C}^A_{uu}q^A\right)^T\left(\tilde{C}^B_{uu}q^B - \tilde{C}^A_{uu}q^A\right) = 0 \quad (2)$$

The connection interface between the upper surface of the solder in contact with chip and the lower surface of the solder in contact with the PCB is described for the non-conforming interfaces resulting in non-binary $\tilde{C}$ matrices. However, using a strongly reduced interface description as proposed in the next paragraph, this approach would lead to locking of the model and an overly stiff description. In order to alleviate this issue, it is necessary to add some compliance in the interface, such that a slight mismatch is admitted between the interface DOFs of both substructures.

2.2 Model order reduction

Classical substructuring methods would set up a reduced order model by defining (at least) an interface mode for each set of interface points. However, for problems like these micro-electronics, this would

lead to an unacceptably large reduced order model as too many interface points are present. Moreover, the setup of this reduced order model would be prohibitively expensive as this also requires solving the response for all these interface points. In order alleviate this issue, we propose the use of a set of stochastic interface modes. The resulting basis for each subcomponent then consists of:

- A set of rigid body modes V_{rig} which are defined analytically from the nodal positions.
- A set of static thermal response loads V_{TM} which are determined from solving the static response for both the PCB and the chip for their respective thermal stresses.
- A set of stochastic interface loads where a user selected number m of randomized loads distributed over the connecting interface are automatically generated, and their corresponding static deformation (orthogonal to the rigid body modes) are computed.

3 Numerical case

The Infineon PG-TDSON-8 chip on a PCB is modelled in finite element. The interfaces are shown in figure 2.

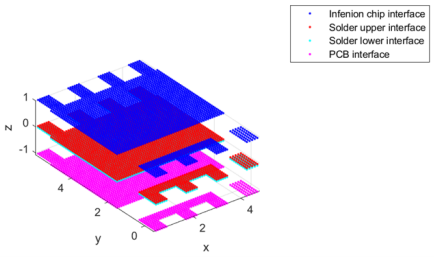


Fig. 2: Interface of the Infineon PG-TDSON-8 chip. Blue and pink dots respectively indicate the detected closest nodes of the solder parts to the chip and PCB.

Figure 3 shows the decay of the singular values of the stochastic interface modes for the Infineon chip, PCB and solder patches.

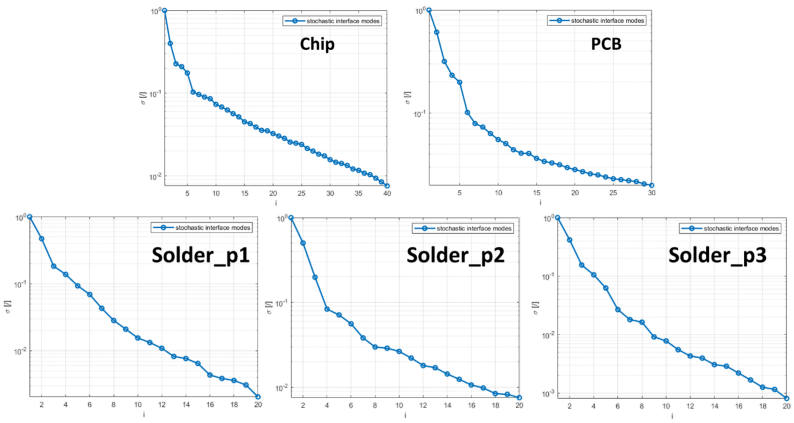


Fig. 3: Singular value decay in stochastic load modes.

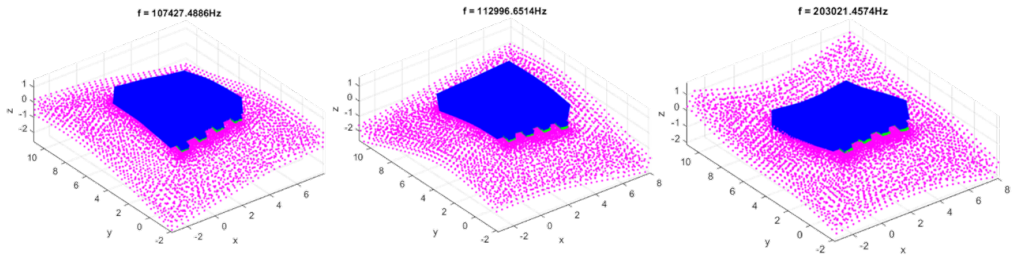


Fig. 4: Mode shapes obtained from system model

The eigenvalue analyses of the reduced and substructured system model is executed in less than 0.1 seconds on a regular laptop. The first three flexible modes are shown in figure 4.

4 Python toolchain

The core of the toolchain is Python scripting. Python allows to implemented the described methodology with the NumPy package [2]. Python also allows to interface with Simcenter 3D via the NX Open API to:

- parametrize the location of the chips on the PCB;
- call the NXNastran solver;
- post process the simulation results.

Acknowledgements

The Flanders Innovation & Entrepreneurship Agency within the COMPAS project is gratefully acknowledged for its support. This research was partially supported by Flanders Make, the strategic research centre for the manufacturing industry. Internal Funds KU Leuven are gratefully acknowledged for their support.

References

[1] IPC/JEDEC-9301:2018, “Numerical analysis guidelines for microelectronics packaging design and reliability,” standard, Institute for Interconnecting and Packaging Electronic Circuits, Bannockburn, Illinois, USA, 2018.

[2] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, “Array programming with NumPy,” *Nature*, vol. 585, pp. 357–362, Sept. 2020.

ROSS: An Open-Source Python Package for Rotordynamic

Vinicius Teixeira da Costa[1] * Marcus Filipe Sousa Reis[1] Raphael Timbó Silva[3]
André Albuquerque Thomas e Brandão[3] Thiago Gamboa Ritto[2]
Aldemir Aparecido Cavallini Junior[1]

[1] *LMEst - Structural Mechanics Laboratory, Federal University of Uberlandia (UFU), Brazil*
[2] *Dept. of Mechanical Engineering at the Federal University of Rio de Janeiro (UFRJ), Brazil*
[3] *Petrobras - Petróleo Brasileiro S.A*

Abstract

ROSS (Rotordynamic Open-Source Software) is the first library for modeling and analysis of dynamic rotors developed in open-source Python, providing users with a simulation environment for studying the behavior of complex machines. It is able to create elements such as shafts, disks, bearings, seals and various elements of a rotor, in addition to providing tools that allow evaluating the performance of the system in different operating conditions, stability, response in time and frequency, making accurate predictions by considering various situations. The library is configurable, adapting to specific requirements, being recommended for engineers and researchers interested in describing phenomena studied in rotating machines.

Keywords: Highly configurable, Ross, rotordynamic, simulation environment.

1 Statement of Need

Systems involving rotating components are increasingly required to model the dynamics of this equipment, which can make the representation extremely complex. These systems may operate at high frequencies, which enhances the probability of failures and accidents. It is the role of engineers and developers, in general, to understand how rotating systems work, and hence solve such problems. Thus, in addition to considering dynamic effects, which can help predict excessive vibration, fatigue and instability, it is necessary to anticipate each and every point to be considered as a possible design weakness. Many tools already exist in the world commercially that presents some rotor dynamics functionalities, both standalone tools [7, 6], and world-famous ones [1, 2], packages based on proprietary runtimes (ex. MATLAB) [4, 3]. These solutions require a license and were not developed in an open-source, collaborative way, restricting users to the environment with only graphical interactions, and often making more complex and extensive analyses impossible. In contrast, the Rotordynamic Open-Source Software (ROSS) library allows modeling and simulating the behavior of rotating systems under different operation conditions, with possible predictions which helps in research and enables the development of safe and reliable machines that work efficiently [9].

*Corresponding author, Email address: viniciustxc3@gmail.com

ROSS can be accessed at the website `https://ross.readthedocs.io/`, where its documentation can be found. It is possible also to visit its repository on GitHub, where the entire community is welcome to access and contribute to the library, being able to create and assign materials, instantiate the elements that make up the rotor, use and convert different units, and run different analyses.

In the numerical simulation in the construction of rotor models, ROSS uses the Timoshenko beam theory [5], in which rotational inertia and shear effects are considered. The rotor is discretized using the finite element method [4]. Disks are modeled as rigid bodies, considering its kinetic energy due to translation and rotation. It is possible include isotropic bearings in asymmetrical rotors, anisotropic bearings, cross-coupled bearings, isotropic bearings with damping, hydrodynamic bearings, cantilevered rotors, and hydrodynamic bearings.

To use this package, it is necessary to include the basic elements, such as shaft elements, bearing elements and disks, in list like format. If the shaft elements are not numbered, the class defines a number for each one, according to the position of the element in the list provided to the rotor constructor, as in the code example Listing 1 and represented in Fig. 1.

```
import ross as rs
import numpy as np

# Classic Instantiation of the rotor
shaft_elements = []
bearing_seal_elements = []
disk_elements = []
steel = rs.Material.load_material("Steel")
for i in range(6):
    shaft_elements.append(rs.ShaftElement(
      L=0.25, material=steel, n=i, idl=0, odl=0.05))

disk_elements.append(rs.DiskElement.from_geometry(
  n=2, material=steel, width=0.07, i_d=0.05, o_d=0.28))
disk_elements.append(rs.DiskElement.from_geometry(
  n=4, material=steel,  width=0.07, i_d=0.05, o_d=0.35))
bearing_seal_elements.append(rs.BearingElement(
  n=0, kxx=1e6, kyy=1e6, cxx=0, cyy=0))
bearing_seal_elements.append(rs.BearingElement(
  n=6, kxx=1e6, kyy=1e6, cxx=0, cyy=0))
rotor591c = rs.Rotor(shaft_elements=shaft_elements,
                     bearing_elements=bearing_seal_elements,
                     disk_elements=disk_elements,)

rotor591c.plot_rotor()
```

Listing 1: Rotor example with overlapping shaft elements

An example of possible to model complex machinery, for a centrifugal compressor modeled with ROSS, see Fig. 1. Shaft elements are shown in gray, disks are shown in red, and bearings are shown as springs and dampers.

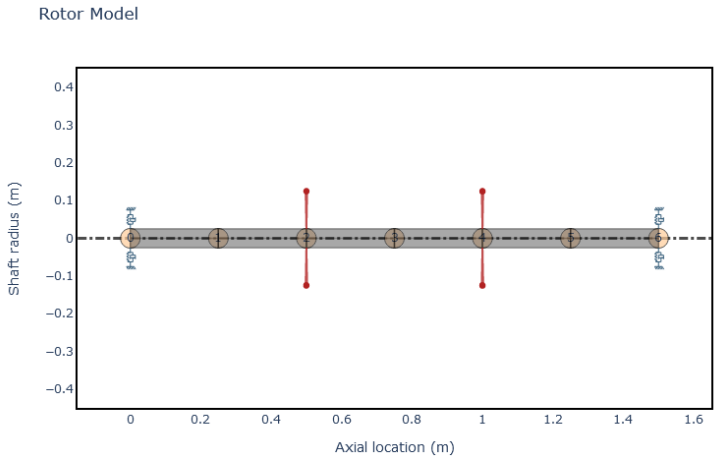


Fig. 1: Example rotor model

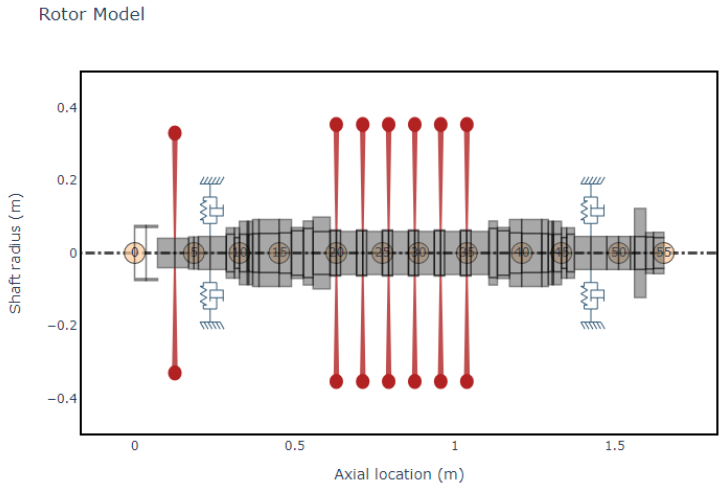


Fig. 2: Example centrifugal compressor modeled with ROSS

After formulating the model, it is possible to plot the rotor geometry, run simulations and get results in the form of graphs. ROSS can perform different analyses, such as static analysis, critical speed map, mode shape, frequency response, and time response. The user can run the function *campbell.plot(harmonics=[0.5, 1])*, to determine the Campbell diagram presented in Fig. 3, generated for the compressor given by Fig. 1, the backward and forward critical speeds can be observed in the obtained Campbell diagram.

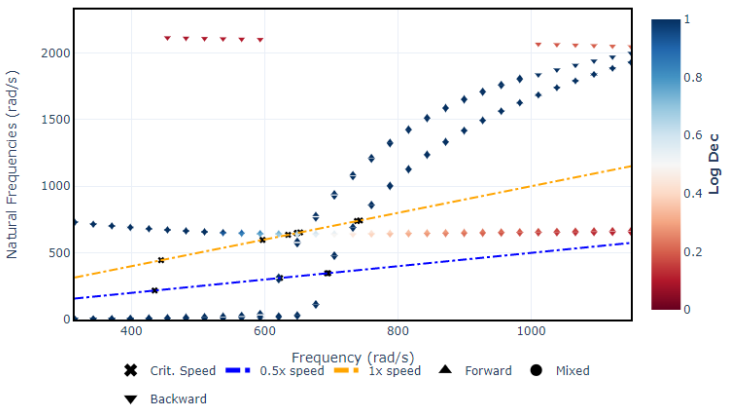


Fig. 3: Campbell Diagram

With this library, it is possible to simulate not only the rotor, but also the measurement configuration by using probe variables. These variables are a list of tuples with the node, where to observe the response, and orientation information. Figure 4 presents the Bode and Nyquist diagram, generated by the code in Listing 2.

```
# Unbalance 1
n1 = 29 # node
m1 = 0.003 # magnitude
p1 = 0 # phase
# Unbalance 2
n2 = 33 # node
m2 = 0.002 # magnitude
p2 = 0 # phase

frequency_range = np.linspace(315, 1150, 101)
results2 = rotor3.run_unbalance_response(
  [n1, n2], [m1, m2], [p1, p2], frequency_range)

probe1 = (15, 45) # node 15, orientation 45 degree
probe2 = (35, 45) # node 35, orientation 45 degree

results2.run_unbalance_response.plot(
  probe=[probe1, probe2], probe_units="degrees")
```

Listing 2: Unbalance response analysis with probe example

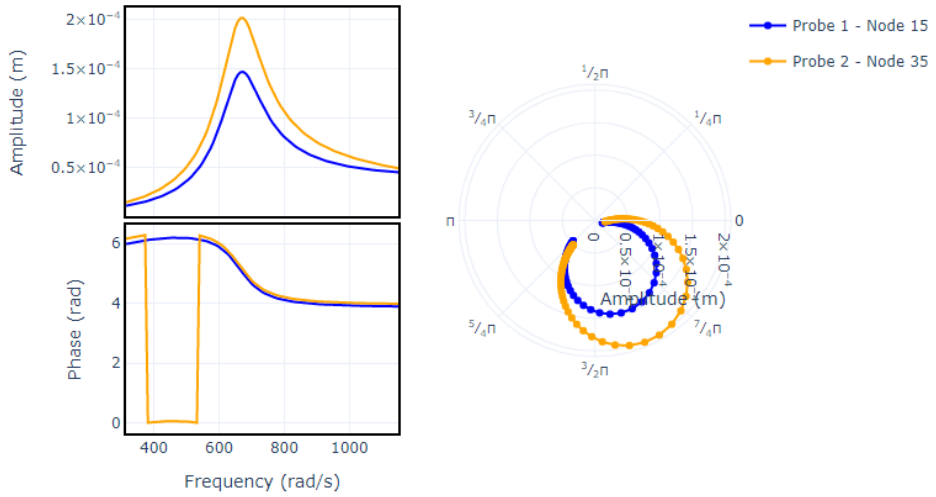


Fig. 4: Unbalance analysis results with Bode and Nyquist diagram

This library can also be used to evaluate the effects of seals on the dynamic behavior of rotors, and enables the extraction of the coefficients (mass, stiffness, and damping) with uncertainties [8]. Once it is open-source library coded in Python language, it facilitates the integration of codes with other programs, without dependence on commercial softwares. Additionally, the ROSS package has extensive documentation, and a set of examples in Jupyter Notebooks with the usage instructions.

2 Conclusion

ROSS package allows the numerical analyses of rotating machines operating under different conditions and encompassing various components, such as shafts, disks, bearings, and seals. This enables the simulation of several important problems in the field of rotordynamics, development, and industrial applications, making possible to work with fault diagnoses. Further work on the ROSS library will be dedicated to include learning algorithms for digital twin and fault identification applications.

3 Acknowledgements

The authors would like thanks CNPq, FAPEMIG, CAPES (INCT-EIE), and Petrobras for the financial support of the present contribution.

References

[1] ANSYS. *Mechanical Rotordynamics*. 2023. URL: https://www.ansys.com/training-center/course-catalog/structures/ansys-mechanical-rotordynamics.

[2] COMSOL. *Rotordynamics Module*. 2023. URL: https://www.comsol.com/blogs/tag/rotordynamics-module.

[3] DELTA-JS-AG. *MADYN 2000*. 2023. URL: https://www.delta-js.ch/en/software/.

[4] Michael Friswell. *Dynamics of rotating machines*. Cambridge University Press, 2010. DOI: 10.1017/CBO9780511780509.

[5] J. R. Hutchinson. "Shear coefficients for timoshenko beam theory". In: *Journal of Applied Mechanics, Transactions ASME* 68 (1 2001), pp. 87–92. ISSN: 15289036. DOI: 10.1115/1.1349417.

[6] Turbomachinery Laboratory. *XLTRC2*. 2023. URL: https://turbolab.tamu.edu/trc-software/.

[7] LaMCoS. *ROTORINSA*. 2023. URL: http://lamcos.insa-lyon.fr/logiciels_rotorinsa.php?eq=0&L=2.

[8] Raphael Timbó and Thiago G. Ritto. "Impact of damper seal coefficients uncertainties in rotor dynamics". In: *Journal of the Brazilian Society of Mechanical Sciences and Engineering* 41 (4 Apr. 2019). ISSN: 18063691. DOI: 10.1007/s40430-019-1652-8.

[9] Raphael Timbó et al. "ROSS - Rotordynamic Open Source Software". In: *Journal of Open Source Software* 5 (48 Apr. 2020), p. 2120. DOI: 10.21105/joss.02120.

MorletWaveModal: Python package for modal identification using Morlet-wave integral

Ivan Tomac[1,2] * Janko Slavič[1]

[1] *Faculty of Mechanical Engineering, University of Ljubljana, Aškerčeva cesta 6, SI-1000 Ljubljana, Slovenia*
[2] *Faculty of Electrical Engineering, Mechanical Engineering and Naval Architecture, University of Split, Ruđera Boškovića 32, HR-21000 Split, Croatia*

Abstract

This research deals with the identification of modal parameters from short and noisy signals. The identification is done using the free response signals where the high frequency modes suffer more from the influence of noise, which is even more evident when the displacement signal is used. To overcome these challenges, a method based on the Morlet wave integral was developed to identify the modal parameters. The method is based on the continuous wavelet transform method, from which it inherits all the advantages in terms of time-frequency separation, but does not suffer from the edge effect and is much faster. The method is implemented in the form of the Python package, named Morlet-Wave Modal, designed as an object-oriented and it is easy to use. For the basic use a free response signal and the initial natural frequency are required to identify modal parameters. The basics of the Morlet wave model method are presented.

Keywords: python package, open source, modal identification, Morlet-wave

1 Statement of Need

The optical methods based on the high-speed cameras for the measurements of the vibration responses from structures are fast developing [1]. Such measurements are typically used to identify full-field mode shapes, but damping and natural frequencies were identified using the high dynamic range sensor [2]. The reason is that the displacement measurements are very noisy and also suffer from low dynamic range which is critical for higher modes that fades out quickly into noise [3]. Therefore, a need to develop a method that will identify modal parameters from relatively short, noisy and low dynamic range signals arose. The Morlet wave damping identification method [4] was successfully applied to the high-speed camera based responses to perform full-field damping identification [5]. The Morlet wave method was then extended to amplitude and phase identification [6], leading to the development of the open source Python package MorletWaveModal [7].

The MWModal was developed as an object-oriented programme containing seven methods and eight auxiliary functions. It is based on the Numpy [8] package for arrays and numeric operations on arrays. The identification of modal parameters is based on the solution of an overdetermined system of

*Corresponding author, Email address: *itomac@fesb.hr*

equations using the least squares minimization which is established with the SciPy [9] package. The MW Modal method is an extension of the Morlet wave damping identification method [4] and is also implemented as a Python package MWDI [10] which is required for operation. Modal identification is performed by executing a single method that wraps methods for identification of: natural frequency, damping and residue (expressed as amplitude and phase angle). The modal identification workflow of the software package is presented in this presentation.

2 MorletWaveModal package – basic usage

The package is installed using the `pip` package manager from the command line:

Listing 1: Installation of the MorletWaveModal package

```
$ pip install morletwavemodal
```

Demonstration of the package assumes that the code is ran inside the Jupyter notebook [11]. The installed package is imported and the object identifier is assigned with:

Listing 2: Initialization of the object identifier

```
import mwmodal as mwm
identifier = mwm.MorletWaveModal(free_response=, fs=)
```

Where `free_response` – requires a vector, a NumPy [8] array containing time series from the single measurement point of the free response of the mechanical system with multiple degrees of freedom, and `fs` – is the sampling frequency of the time series signal. The next step is to call the wrapper method to perform the modal identification:

Listing 3: Identification of the modal parameters at the selected mode

```
omega, delta, X, phi = identifier.identify_modal_parameters(
   omega_estimated=)
```

where `omega_estimated` is the roughly estimated natural frequency in $\mathrm{rad\,s^{-1}}$ (typically picked from the magnitude plot in the frequency domain) of the identified mode. The identified modal parameters are returned as follows: `omega` – exact natural frequency, `delta` – damping ratio, `X` – amplitude, `phi` – phase angle. With this step the modal parameters are identified and the procedure can be iterated over the different modes.

3 Identification algorithm

In this section, the identification algorithm is explained in more detail. The wrapper method in List. 2 contains the following parameters, which are set by default: $k_{\mathrm{lo}} = 10$, $k_{\mathrm{hi}} = k_{\mathrm{lim}}$, $N_{\mathrm{k}} = 10$, $n_1 = 5$ and $n_2 = 10$. If necessary, these parameters can be changed during the initialisation phase:

Listing 4: Parameters of the method

```
identifier = mwm.MorletWaveModal(free_response=, fs=, n_1=5,
    n_2=10, k_lo=10, num_k=10)
```

where parameters n_1 and n_2 control the sensitivity of the method on identification of damping [6], parameters k_{lo} and k_{hi} determine the time range of the signal included in the analysis, and N_{k} controls

overdetermination. The parameter k_{hi} is determined by the value k_{lim}, which is the maximum theoretical limit for the Morlet wave integral [4, 6]. The parameter k_{lim} is controlled via estimated damping ratio which can be set when the wrapper method is executed:

Listing 5: Setting the the upper signal length used for identification

```
omega, delta, X, phi = identifier.identify_modal_parameters(
    omega_estimated=, damping_estimated=0.0025)
```

Where `damping_estimated` is the estimate of the damping ratio for the analysed mode. It is set to the default value, resulting in a maximum $k_{\mathrm{lim}} = 400$. The estimated damping can be set in the range $0.02\,\% \leq \delta \leq 2\,\%$ and an estimated damping of, *e.g.* $2\,\%$ would result in $k_{\mathrm{lim}} = 49$. For each mode at the original natural frequency, the wrapper method (List. 3, 5) executes the following four methods, which can be executed independently by the user in listed order:

1) `self.initialization(omega_estimated, damping_estimated)`

2) `self.identify_natural_frequency(omega_estimated)`

3) `self.identify_damping()`

4) `self.identify_amplitude_phase(damping)`

The method 1) determines k_{hi} and distributes N_{k} k values between k_{lo} and k_{hi} inclusive. The method requires the estimated natural frequency to check if the signal at the selected frequency has oscillations less than k_{hi}. In this case k_{hi} is adjusted to the signal length.

The method 2) searches the exact natural frequency at each k value in the range specified by the set with method 1). The method returns identified natural frequency.

The method 3) identifies damping using least squares minimisation. The method returns the identified damping ratio. When the method 3) is called by the wrapper method, the identified damping ratio is compared with the estimated damping ratio. If the identified damping ratio is higher than the estimated damping ratio, the estimated damping ratio is adjusted to the identified one and the identification is repeated from the method 1).

The method 4) identifies amplitude and phase using least square minimisation. The identified values are returned as two separate variables.

A detailed implementation of the method can be found in the repository where the source code is deposited [7].

4 Conclusion

An open source software package, developed in Python is presented. It is developed as a tool for modal identification of relatively short, noisy and low dynamic range signals, such as high-speed video recordings. The method is designed to work on SISO tests and extension to SIMO is straight forward. The source code is deposited in the online repository GitHub under the MIT licence.

Acknowledgements

The authors gratefully acknowledge partial financial support from the European Union's Horizon 2020 research and innovation programme under Marie Skłodowska-Curie Grant Agreement No. 101027829 and the Slovenian Research Agency (N2-0144, J2-3045).

References

[1] J. Baqersad, P. Poozesh, C. Niezrecki, and P. Avitabile, "Photogrammetry and optical methods in structural dynamics – a review," *Mechanical Systems and Signal Processing*, vol. 86, pp. 17–34, 2017.

[2] J. Javh, J. Slavič, and M. Boltežar, "The subpixel resolution of optical-flow-based modal analysis," *Mechanical Systems and Signal Processing*, vol. 88, pp. 89–99, 5 2017.

[3] J. Javh, J. Slavič, and M. Boltežar, "High frequency modal identification on noisy high-speed camera data," *Mechanical Systems and Signal Processing*, vol. 98, pp. 344–351, 1 2018.

[4] J. Slavič and M. Boltežar, "Damping identification with the morlet-wave," *Mechanical Systems and Signal Processing*, vol. 25, pp. 1632–1645, 7 2011.

[5] I. Tomac and J. Slavič, "Damping identification based on a high-speed camera," *Mechanical Systems and Signal Processing*, vol. 166, p. 108485, 3 2022.

[6] I. Tomac and J. Slavič, "Morlet-wave-based modal identification in the time domain," *Mechanical Systems and Signal Processing*, vol. 192, p. 110243, 6 2023.

[7] I. Tomac and J. Slavič, "ladisk/morletwavemodal: Mwmodal v0.6.3," 4 2023.

[8] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with numpy," *Nature*, vol. 585, pp. 357–362, 9 2020.

[9] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İlhan Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, A. Vijaykumar, A. P. Bardelli, A. Rothberg, A. Hilboll, A. Kloeckner, A. Scopatz, A. Lee, A. Rokem, C. N. Woods, C. Fulton, C. Masson, C. Häggström, C. Fitzgerald, D. A. Nicholson, D. R. Hagen, D. V. Pasechnik, E. Olivetti, E. Martin, E. Wieser, F. Silva, F. Lenders, F. Wilhelm, G. Young, G. A. Price, G.-L. Ingold, G. E. Allen, G. R. Lee, H. Audren, I. Probst, J. P. Dietrich, J. Silterra, J. T. Webber, J. Slavič, J. Nothman, J. Buchner, J. Kulick, J. L. Schönberger, J. V. de Miranda Cardoso, J. Reimer, J. Harrington, J. L. C. Rodríguez, J. Nunez-Iglesias, J. Kuczynski, K. Tritz, M. Thoma, M. Newville, M. Kümmerer, M. Bolingbroke, M. Tartre, M. Pak, N. J. Smith, N. Nowaczyk, N. Shebanov, O. Pavlyk, P. A. Brodtkorb, P. Lee, R. T. McGibbon, R. Feldbauer, S. Lewis, S. Tygier, S. Sievert, S. Vigna, S. Peterson, S. More, T. Pudlik, T. Oshima, T. J. Pingel, T. P. Robitaille, T. Spura, T. R. Jones, T. Cera, T. Leslie, T. Zito, T. Krauss, U. Upadhyay, Y. O. Halchenko, and Y. Vázquez-Baeza, "Scipy 1.0: fundamental algorithms for scientific computing in python," *Nature Methods*, vol. 17, pp. 261–272, 3 2020.

[10] I. Tomac and J. Slavič, "ladisk/mwdi: Mwdi v0.72," 4 2023.

[11] B. E. Granger and F. Pérez, "Jupyter: Thinking and storytelling with code and data," *Computing in Science & Engineering*, vol. 23, pp. 7–14, 2021.

Assessing non-stationary loading in structural dynamics: Python package for random time series analysis (pyRaTS)

Arvid Trapp * Peter Wolfsteiner

Munich University of Applied Sciences

Abstract

This article presents an open-source Python package, pyRaTS, an object-oriented framework designed to analyze and process time series in random vibration fatigue. Its presentation focuses on methods for analyzing non-stationary vibration loading in structural dynamics. This provides the basis for reliable assessments of fatigue strength, which is essential for determining the integrity of mechanical structures subjected to random vibration loading. A fatigue strength assessment can be based either on a statistical or a non-statistical approach of structural dynamics. The statistical approach uses power spectral densities to characterize loading and stresses, while the non-statistical approach involves the computationally costly processing of time-domain realizations. However, non-stationary vibration loading, commonly encountered in real applications, leads to significant deviations between both approaches. The pyRaTS methods presented herein determine effects of non-stationary loading on structural dynamics and fatigue strength. These methods include a modal perspective (the Fatigue Damage Spectrum — FDS), a spectral representation of kurtosis (the non-stationarity matrix), and a framework for abstracting damage-equivalent loading using a set of stationary Gaussian processes. Finally the article addresses a crucial limitation of the FDS related to modal decoupling and proposes a solution by applying linear systems theory to the non-stationarity matrix to estimate how non-stationary loading transfers into structural responses.

Keywords: non-stationary loading, structural dynamics, random vibration fatigue, Fatigue Damage Spectrum, non-stationarity matrix, open source, Python

*Corresponding author, Email address: arvid.trapp@hm.edu

Tab. 1: Abbreviations

Abbreviation	Definition	Abbreviation	Definition
DER	derive (new object)	QS	quasi-stationary
DK	Dirlik	RFC	rainflow counting
EST	estimate	SDOF	single-degree-of-freedom
f_D	damped natural frequency	STP	short-time periodogram
FD	frequency domain	SOE	system of equations
FDS	Fatigue Damage Spectrum	TD	time domain
NSM	non-stationarity matrix	TF	transfer function
PSD	power spectral density	TS	time series

1 Introduction

An accurate assessment of fatigue strength is critical for determining the integrity of mechanical structures subjected to random vibration loading [1]. A fatigue assessment involves calculating load spectra of critical cross-sections and comparing them with stress-life curves to predict structural lifetimes. Load spectra can be obtained following either a statistical or non-statistical approach of structural dynamics. The statistical approach uses power spectral densities (PSD) to characterize loading and stress tensors, while the non-statistical approach involves the computationally costly processing of realizations obtained from physical measurements or numerical models [2, 3]. Commonly, the former is referred to as the (statistical) frequency-domain and the latter as the (non-statistical, sampling-based) time-domain approach. Although the statistical approach seems to be the natural choice for loading driven by random components, it has a set of crucial limitations, mainly related to its PSD-based statistical characterization. PSDs provide spectral second-order averages that are incapable of representing the non-stationary nature of loading. Solely ob the PSD's basis, we cannot differentiate between non- and stationary processes. However, non-stationary loading is prevalent in many relevant engineering applications, leading to significant deviations between the statistical and non-statistical approaches in a fatigue assessment [4].
Previous research has primarily focused on expanding the statistical characterization and developing tools capable of capturing non-stationary and non-Gaussian effects on fatigue damage [5, 6]. To achieve a more detailed statistical characterization, central statistical moments of higher order and their normalized descriptors such as skewness and kurtosis are widely employed. These central moments provide concise statistical descriptions of the (average) probability density function and can be used to determine if loading conforms to Gaussianity and stationarity. However, to assess the effects of non-stationary loading on structural responses, i.e. including structural dynamics, more comprehensive characterizations are necessary. For applying linear systems theory we are required to turn to the frequency domain [7, 8, 9]. The following presentation gives an introduction into the Python pyRaTS (Random Time Series) package [10], with a specific emphasis on its ability to assess non-stationary loading in structural dynamics. The pyRaTS package is designed as an object-oriented framework for analyzing and processing time series in random vibration fatigue. It is set up to work with single-channel loading — the classic uniaxial approach. But it can flexible also process multi-channel and multi-process (quasi-stationary) loading. This includes the option for single-channel single-process (SCSP), single-channel multi-process (SCMP), multi-channel single-process (MCSP), and multi-channel multi-process (MCMP) configurations. Multi-channel configurations can also be useful for comparing different types of loading, as the example herein included presents. With this wide set of configurations the pyRaTS package was designed to be flexible, extensible, and easy to use. It may be employed to analyze random loading, develop and test new algorithms for fatigue evaluation, and compare different fatigue approaches. Therefore it integrates methods from other open-source vibration fatigue packages like FLife [11] and pyExSi [12]. In this presentation, we focus on pyRaTS methods that are useful in assessing non-stationary loading in structural dynamics. This includes a modal perspective (Fatigue Damage Spectrum) [5], a spectral representation of kurtosis (non-stationarity matrix) [13], and a framework for abstracting damage-equivalent loading using a set of stationary Gaussian processes [14].

2 Assessing non-stationary loading

In many engineering applications, it is necessary to consider the effects of non-stationary loading on structural components and their fatigue behavior. Non-stationary loading is characterized by changing loading conditions that have varying statistical properties over time. In this section, we will explore different approaches to analyze the effects of non-stationary loading on structural components and how they are incorporated in the pyRaTS package. To help visualize the subject of non-stationary loading in the context of structural dynamics and fatigue assessments, we will begin with introducing a synthetic dataset serving for illustration. As a synthetic dataset, it is easy to understand and reproduce, however, it symbolizes the same challenges we face in real non-stationary vibration loading, just in a much simplified form. We will then examine the limitations of using second-order averages, such as the PSD, to describe the statistical properties of non-stationary loading (Sec. 2.1). Next, we will discuss three methods for assessing fatigue damage under non-stationary loading conditions: (*i*) the Fatigue Damage Spectrum (FDS) that enables comparison of loads from a modal perspective (Sec. 2.2), (*ii*) the non-stationarity matrix that characterizes non-stationary loading spectrally (Sec. 2.3), and (*iii*) quasi-stationary load approximations that offer a practical approach to address non-stationary loading in a statistical fatigue assessment (Sec. 2.4).

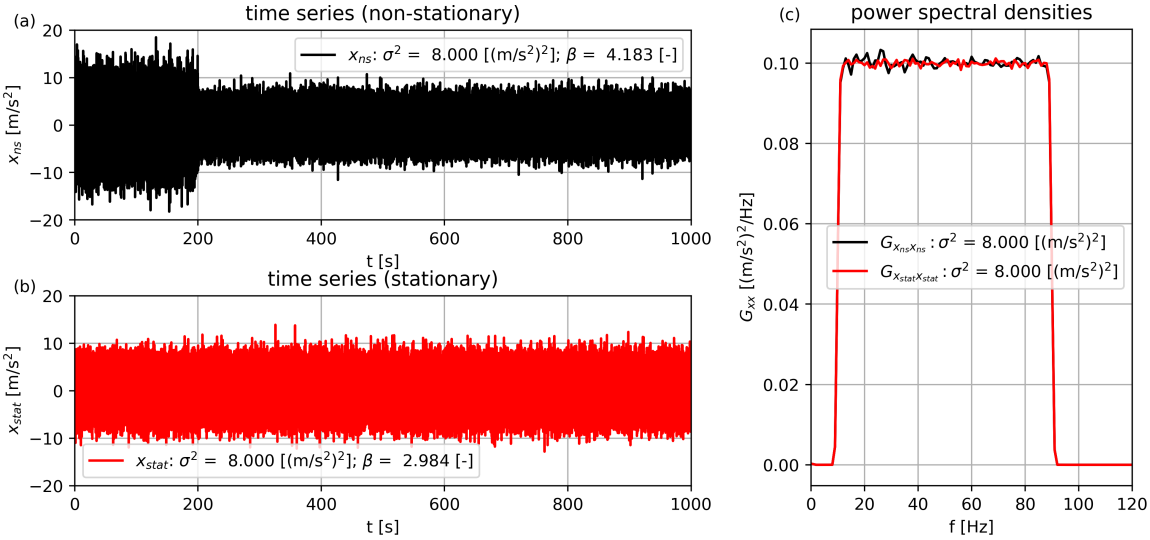


Fig. 1: Synthetically generated *(a)* non-stationary switching $x_{\mathrm{ns}}(t)$ and *(b)* stationary Gaussian $x_{\mathrm{stat}}(t)$ load series [pyRaTS command: `sig.plot()`], and *(c)* their PSDs [`sig.plot_psd(access = 'comp')`]

The example herein used for illustration compares a non-stationary and a stationary process having the same PSD (Fig. 1). The non-stationary process is constructed using one of the simplest models for generating non-stationary data — a switching process [15]. It consists of two stationary Gaussian sub-processes $X_{\mathrm{ns},p=1}(t)$ and $X_{\mathrm{ns},p=2}(t)$. These two sub-processes are assembled with proportions $T_1 = 0.2T$ and $T_2 = 0.8T$ to form the switching process, where $\{0.2, 0.8\}$ denote the portions to the total length T. The choice of a switching process is motivated by several factors, including its simplicity, its ability to design frequency-selective non-stationary behavior and its close relation to short-time periodogram (STP) analysis. Often, non-stationary processes are approximated by amplitude mod-

ulation [16], which is frequency-indifferent. But realistic processes are mostly frequency-selective in their non-stationary behavior. STP analysis, which resolves PSDs in time, provides one way to observe this. Hereby, the switching process is a reasonable simplification, that captures this continuous change of statistical and spectral descriptors in an easily understandable setting. It provides a useful framework for investigating the limitations of using classic second-order averages to describe non-stationary processes, understanding the effects of non-stationary loading on fatigue damage, and developing techniques to mitigate these effects in practical engineering applications.

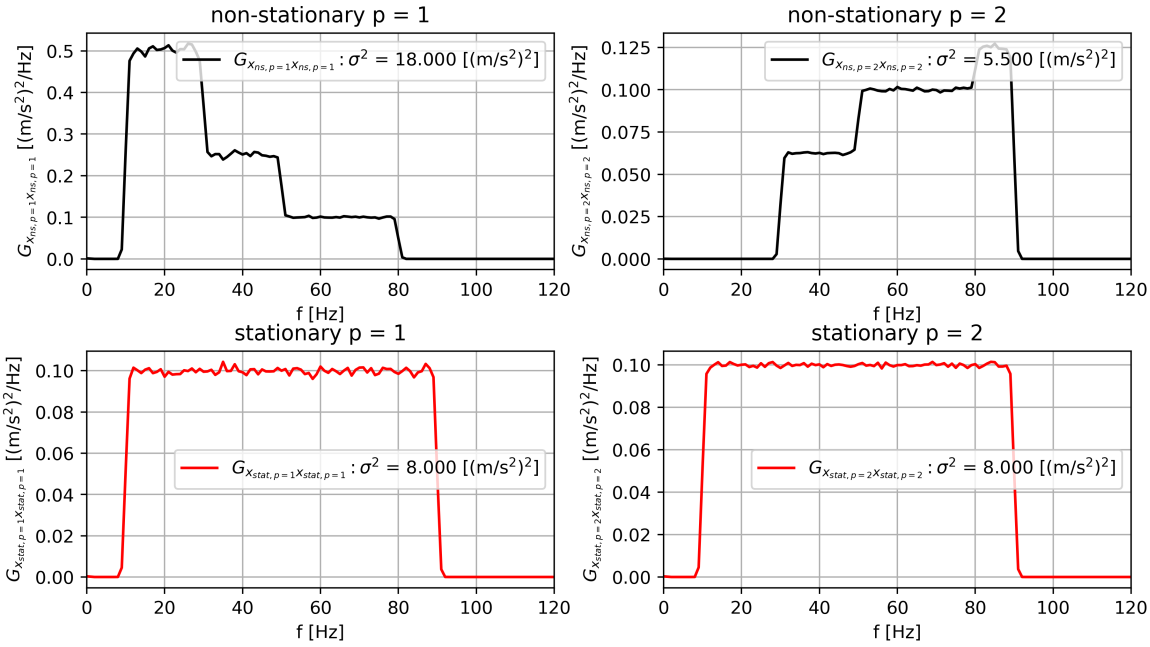


Fig. 2: Definition of exemplary switching process using different PSDs to set up a non-stationary switching and a stationary process of same (average) PSD [`sig.plot_psd(access = 'proc')`]

The switching process that serves for illustration (Fig. 1a) is created by assembling two stationary sub-processes (Figure 2, upper plots) using the Python code provided in Appendix A. The code defines the relevant PSD shapes and organizes them as a nested list of dictionaries containing PSDs and realization length T. This initializes a single timeseries object with two channels $\{x_{\mathrm{ns}}(t), x_{\mathrm{stat}}(t)\}$ and two sub-processes $p = \{1,2\}$ each, when passing it to the pyRaTS package. Almost all figures presented in the following sections are directly implemented in the Python package (Appendix B resp. figure captions). As such, Figure 2 is a pre-implemented plot that shows on the lower half the stationary process, which for implementation also consist of two stationary sub-processes that, in contrast to the switching process, are the same.

2.1 Second-order averages

Figure 1c compares the PSD $G_{xx}(f)$ of both processes. These are the same, which is a result of the inherent averaging that occurs when estimating a PSD. As a result, both processes also have the same (average) variance σ^2 resp. second-order central moment $\mu_2 = \sigma^2$ — the integral of the PSD

$\mu_2 = \sigma^2 = \int_0^\infty G_{xx}(f)df$. Thus, the switching process is not only composed of two different vibration states ('PSD shapes') but also embodies two different levels of intensity $\{\mu_{2,ns,p=1}, \mu_{2,ns,p=2}\}$. As each unique vibration state is stationary, we will refer to the switching process as a quasi-stationary (QS) process. While this example is synthetic, it provides a clear demonstration of the essential difficulties that arise when dealing with non-stationary loads. Although both processes have the same second-order statistical properties (Fig. 1), their kurtoses $\beta = \mu_4/\mu_2$ differ, with the non-stationary process having a larger kurtosis value than the stationary process. This larger value indicates a deviation from the Gaussian probability function, which has a kurtosis value of $\beta = 3$. Here, this deviation can be attributed to the non-stationarity of the process. The second-order and fourth-order moments of a QS Gaussian process can be calculated by averaging

$$\mu_{2,QS} = \sum_p \frac{T_p}{T}\mu_{2,p}; \qquad \mu_{4,QS} = \sum_p \frac{T_p}{T} 3\mu_{2,p}^2 \tag{1}$$

The latter's equation originates from the fourth-order central moment that defines a Gaussian process $\mu_{4,g} = 3\mu_2^2$. Consequently the kurtosis of a quasi-stationary Gaussian processes is provided by,

$$\beta_{QS} = \frac{\mu_{4,QS}}{\mu_{2,QS}^2} = \frac{\sum_p \frac{T_p}{T} 3\mu_{2,p}^2}{(\sum_p \frac{T_p}{T}\mu_{2,p})^2} \tag{2}$$

which essentially simply relates two normalized averages of different order. For the present example, this results in $\beta_{\text{ns}} = 3\frac{0.2\cdot 18^2 + 0.8\cdot 5.5^2}{(0.2\cdot 18 + 0.8\cdot 5.5)^2} = 3\frac{89}{64}$. This combination of second- and fourth-order moments provides a concise characterization of non-stationary processes, although additional higher-order moments may also be considered for a more detailed description. In contrast, a stationary Gaussian process is fully characterized by its second-order moment. However, in this context it is crucial to investigate the spectral content, which is certainly more relevant than additional higher orders in most cases.

2.2 The Fatigue Damage Spectrum

The Fatigue Damage Spectrum (FDS) [5] is a widely used tool in random vibration fatigue due to its simplicity, ability to account for non-stationary effects, and its relation to modal analysis. In structural dynamics, modal analysis can be employed to investigate how a structure responds to external forces or vibrations. The fundamental principle of modal analysis is to decompose the motion of a structure into a series of simpler, independent modes of vibration — its modal shapes. Each mode shape is characterized by a resonant frequency, a corresponding amplitude and can be complement by some damping characteristics. This modal perspective captures the dynamic behavior of structures with a limited set of modes, which is significantly more efficient than examining a structure by all degrees of freedom in the spatial domain.

The FDS founds on this principle but represents modes by fatigue damage potentials — so called pseudo-damages y_{equ}. It includes the process chain of a fatigue assessment, which involves calculating load spectra and carrying out linear damage accumulation. This can find a variety of applications, such as (*i*) comparing different load assumptions for arbitrary structures, (*ii*) comparing different approaches to a fatigue assessment (*iii*) estimating the effects of non-Gaussianity in random loading, and (*iv*) deriving damage-equivalent loading. These applications are based on the hypothesis that if two loads have the same fatigue damage potential for all modes (i.e. share the FDS), these loads would

cause the same fatigue damage in arbitrary dynamic structures represented as a superposition of independent modes. This hypothesis seems reasonable and promising, but the FDS is linked to highly simplified assumptions that rarely apply in reality [17]. However, when used with caution, it can provide helpful insights, as this section will demonstrate. It is worth noting that for comparing different approaches to a fatigue assessment, the pyRaTS FDS routine `est_fds()` accepts all damage estimators included in the Python FLife package [11].

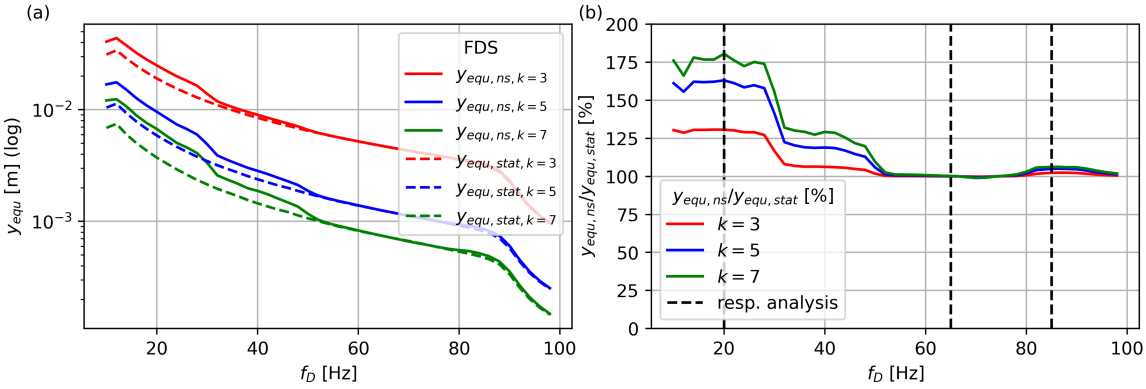
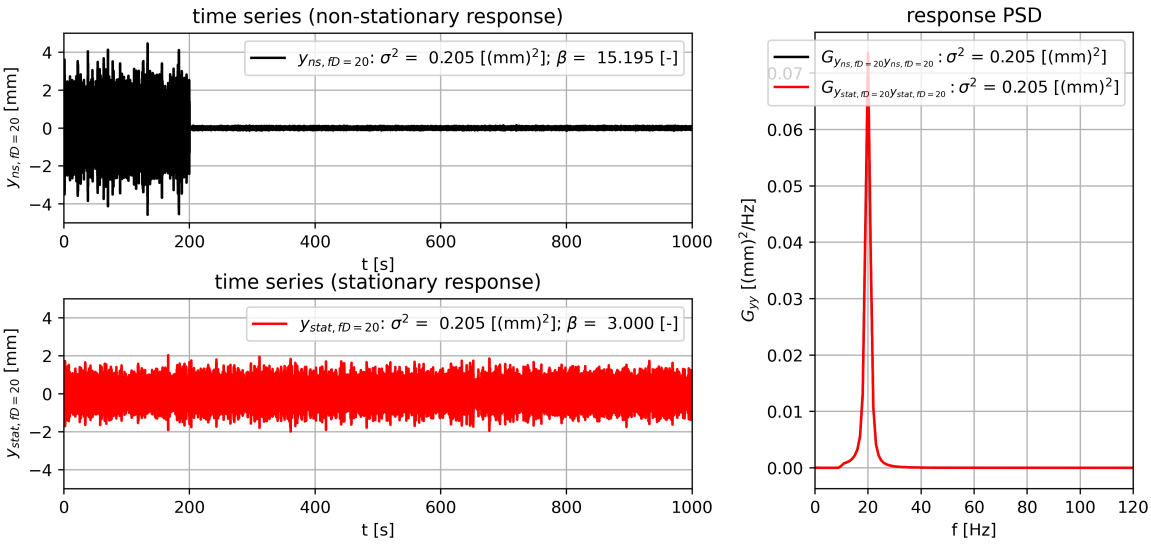


Fig. 3: FDS of example processes *(a)* and their comparison *(b)* for a set of different stress-life curve exponents $k = \{3,5,7\}$

Figure 3 depicts the FDS of the two example loads (Fig. 1). Each FDS 'sample' in Figure 3a is calculated in the pyRaTS package by generating the response of a single-degree-of-freedom system (SDOF — Fig. 4a shows one of them), applying rainflow counting (Fig. 9) and carrying out damage accumulation. The right-hand plot in Figure 3b compares the FDS with another. This comparison confirms that the non-stationary characteristics present in the non-stationary process exhibit strong frequency selectivity. For instance, the FDSs indicate that within the frequency range $f_D = [50, 80]$Hz, where there is no change in intensity (Fig. 2 resp. Appendix A), there is no difference in the fatigue damage potential of the two processes. Structures with resonant frequencies within this range would not be affected differently by the two loads, provided that no other relevant modes outside of the range contribute. Conversely, for the low-frequency interval $f_D = [10,30]$Hz, where the largest changes in the loading's intensity occur, Fig. 3b reveals the greatest discrepancies. To elaborate on this, Figures 4a and 4b display the underlying SDOF response for $f_D = 20$ Hz. They show that even though both processes generate the same second-order averages, the effects of the non-stationary load became larger in the structural response. This significantly affects the fatigue damage potential, which is governed by the potency law associated with the exponent k of the corresponding stress-life curve.

2.3 The non-stationarity matrix

To provide context for introducing the non-stationarity matrix (NSM), we can summarize our prior observations. So far we have used second-order averages to characterize the exemplary loads spectrally and found that linear systems respond proportionally to this characterization (Figure 4b). However, we have also observed that the non-stationary characteristics of the loading affect modes differently, as seen in the kurtosis increasing from $\beta_x = 4.18$ to $\beta_y = 15.20$ for mode $f_D = 20$ Hz (Fig. 1 and 4a). We also found that for modes in the interval $f_D = [50,80]$Hz, the kurtosis frequency-selective decreases to $\beta_y \approx 3$ since this interval was constructed to behave stationary (Fig. 2). These observations were



(a) response time series for $f_D = 20$ Hz **(b)** (average) response PSDs

Fig. 4: Uni-modal structural response for $f_D = 20$ Hz

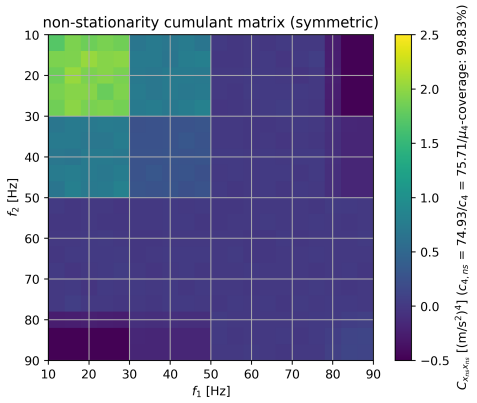
based on time-domain realizations, but we need a statistical characterization of the non-stationary effects that can be processed via linear systems theory to e.g. statistically estimate response kurtoses. This would be analogous to the frequency-domain decompositon of μ_2 — extending the PSD. Therefore, we introduce the NSM, providing a statistical characterization of fourth-order non-stationarity in loading [13].

The NSM is implemented into the pyRaTS package and can be accessed through the `est_nsm()` method. A brief background on the NSM is that it can be obtained either by applying correlation theory to STP (spectrogram) analysis or by framing a subset of the higher-order trispectrum [18] that is related to low-frequency modulation. While the former approach is easier to comprehend, the latter offers extensive knowledge of statistical stable estimation techniques. Expanding the previous equations, for a quasi-stationary process, the NSM has an expected value of

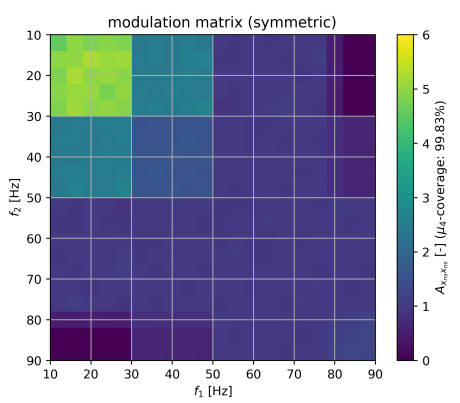
$$M_{xx,QS}(f_1, f_2) = \sum_p \frac{T_p}{T} 3\, G_{xx}(f_1) \otimes G_{xx}(f_2) \tag{3}$$

where the operator $\otimes$ represents the outer product and $G_{xx}(f)$ is the PSD in vector-form. Since the quasi-stationary process $X_{ns}(t)$ is composed of multiple PSD shapes, the two exemplary processes have different outcomes (Figure 5). When understanding the sum over p as sum over short-time periodograms (STPs), Eq. (3) weights all outer periodogram products, and thus $M_{xx}(f_1, f_2)$ can be interpreted as the correlation matrix of these periodograms over time. This moment NSM $M_{xx}(f_1, f_2)$ is a spectral representation of μ_4. More specifically, it should be interpreted as a simplification of the trispectrum, which provides the full spectral decomposition of μ_4 resp. kurtosis β. For low-frequency varying processes that are fundamentally Gaussian, the NSM captures the same fourth-order moment as the trispectrum. However, very abrupt changes, deterministic frequency components, and nonlinear influences are included in the trispectrum but may not appear in the NSM. Therefore, all visualizations of the NSM include a 'coverage' value indicating how much of the time-domain fourth-order moment is represented by the NSM (here 99.8% and 100.6%). The NSM has two advantages

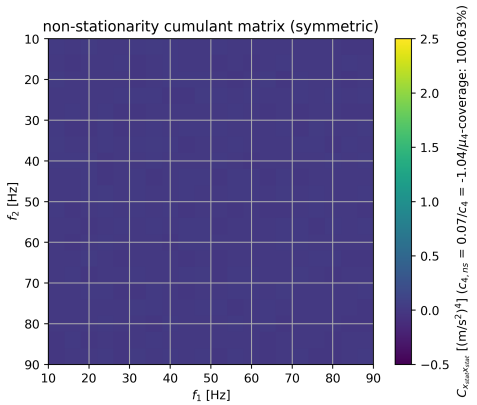
over the trispectrum (f_1, f_2, f_3), which are that its characterization requires one argument less than the trispectrum and these arguments provide clear interpretation. The NSM spectrally decomposes all contributions to μ_4 caused by a low-frequency non-stationary evolution for the frequency arguments f_1 and f_2. The diagonal values indicate non-stationarity for frequency intervals, while off-diagonals indicate whether these effects are correlated for f_1 and f_2 in time. This means the NSM shows whether potential modes are excited synchronously or not.



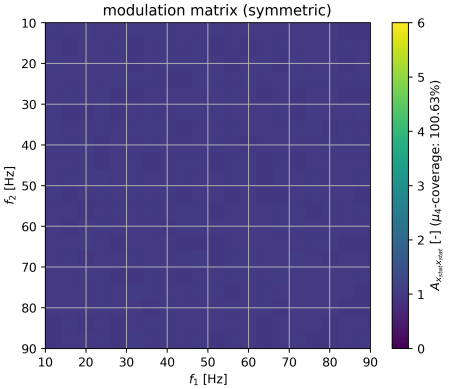
(a) cumulant matrix $C_{x_{\mathrm{ns}}x_{\mathrm{ns}}}(f_1, f_2)$



(b) modulation matrix $A_{x_{\mathrm{ns}}x_{\mathrm{ns}}}(f_1, f_2)$



(c) cumulant matrix $C_{x_{\mathrm{stat}}x_{\mathrm{stat}}}(f_1, f_2)$

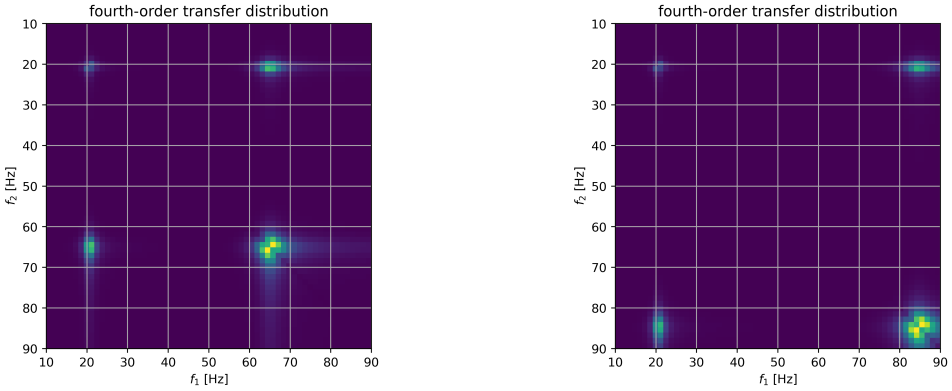


(d) modulation matrix $A_{x_{\mathrm{stat}}x_{\mathrm{stat}}}(f_1, f_2)$

Fig. 5: Non-stationarity matrix [`sig.plot_nsm(func = 'Axx',lim=[0,6])`]

Figures 5a and 5c display the cumulant NSM $C_{xx}(f_1, f_2)$ for the exemplary processes, which provides the spectral decomposition of the fourth-order cumulant $c_4 = \mu_4 - 3\mu_2^2$. The advantage of $C_{xx}(f_1, f_2)$ over $M_{xx}(f_1, f_2)$ is that it reveals deviations from a stationary Gaussian process, i.e. it decomposes the fourth-order moment that causes the kurtosis to deviate from $\beta = 3$, rather than $M_{xx}(f_1, f_2)$, which also includes all stationary Gaussian contributions and thus would be interpreted as displaying the moment that 'deviates' from $\beta = 0$. Consequently, the cumulant NSM of the stationary process in Fig. 5c consists of zero values. In contrast, Fig. 5a mostly displays non-zero values, which are larger for the frequencies that exhibit the largest changes over time (Fig. 2 resp. App. A). But also negative values appear on the off-diagonals between $f = [10, 30]$Hz and $f = [80, 90]$Hz, indicating an anti-

correlation over time between these intervals. This anti-correlation arises because the intervals are not active simultaneously in each of the two sub-processes. This aspect is relevant to the following discussion and is critical in understanding the limitations of the FDS. The interpretation of Figures 5b and 5d proceeds as follows: They present the modulation NSM, which is normalized and therefore represents the spectrally decomposed kurtosis β (Eq. 2). Stationary frequency components have a value of $A_{xx}(f_1, f_2) = 1$. Values above this indicate an increase in kurtosis, while values below one mean a decrease in kurtosis.



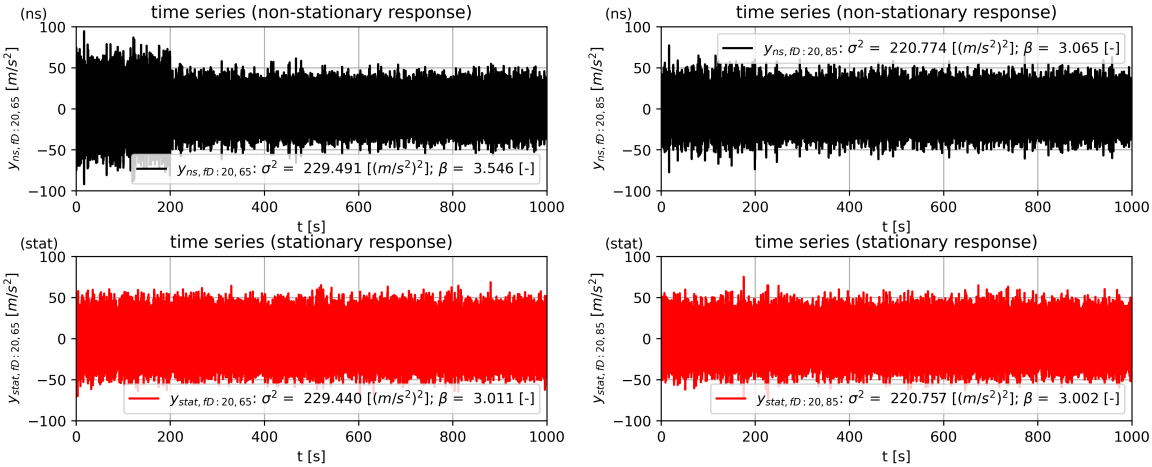
(a) fourth-order TF for $f_D = \{20, 65\}$Hz **(b)** fourth-order TF for $f_D = \{20, 85\}$Hz

Fig. 6: fourth-order transfer function [`tfobj = sig.der_tfObj(fpsd,tf); tfobj.plot_nsm()`]

Next, we consider linear systems theory. Likewise to the second-order transfer function $|H_{xy}(f)|^2$, which shows the dominant frequencies and damping of a structural response, the pendant — the fourth-order transfer characteristic — can also be defined for the NSM. This is given by the matrix $H_{xy}(f_1) \otimes H_{xy}(f_2)$, matching the dimensions of the corresponding NSM. Figure 6 illustrates this for two bi-modal transfer functions, created by superimposing a mode $f_D = 20$ Hz with $f_D = 65$ Hz and $f_D = 85$ Hz respectively. These plots indicate which parts of the excitational NSM dominate the structural response. Note that the transfer function including $f_D = 85$ Hz matches the anti-correlated cross terms, while the other one does not (Figure 5a and 5c). Figure 7 shows the structural responses for these two transfer functions. Only one of them exhibits a relevant kurtosis $\beta_{y,f_D=\{20,65\}} = 3.50$, while the other is also non-stationary but does not result in higher amplitudes due to its specific nature — the anti-correlation. This is interesting because one would interpret the FDS (Figure 3b) contrary. Here the individual deviations from the stationary Gaussian process are larger for $f_D = \{20, 85\}$ than for $f_D = \{20, 65\}$. However, the FDS by principle decouples modes and therefore is not sensitive to the favorable pairing of the specific non-stationary excitation and the dynamic transfer behavior, which becomes crucial for a correct interpretation.

Tab. 2: Response kurtosis for non-stationary excitation using statistical structural dynamics

bins before overlap	Δf	overlapping factor	dimensions	$f_D = \{20,65\}$ β_{FD}	β_{TD}	$f_D = \{20,85\}$ β_{FD}	β_{TD}
20	4 Hz	2	40x40	3.76	3.55	3.12	3.07
20	4 Hz	3	60x60	3.55	3.55	3.06	3.07
40	2 Hz	2	80x80	3.55	3.55	3.05	3.07



(a) Response time series for $f_D = \{20,65\}$Hz

(b) Response time series for $f_D = \{20,85\}$Hz

Fig. 7: Bi-modal structural responses [`sig.der_response(fpsd,tf)`]

Finally, Table 2 highlights important aspects. It presents the calculation of response kurtosis β_y using both, the statistical (`est_statResponse()`) and the time-domain approaches (Fig. 7), where the former relates response NSM to response PSD. While the statistical approach requires processing symmetric matrices of roughly speaking a thousand data points (dimensions in Tab. 2), the latter is computationally more costly, requiring to process realizations of over a million samples. However, as indicated by the parameter variation, the NSM and transfer matrix must be sufficiently resolved in frequency. It is also worth noting that, while the time-domain kurtosis would vary for each realization [19], the frequency-domain remains unaffected — another fundamental motivation for statistical considerations.

2.4 Quasi-stationary load approximation

After discussing the impact of non-stationary loading on structural dynamics, a critical question arises: how can we perform a statistical fatigue assessment when significant non-stationary elements are present in the loading? The pyRaTS package currently available provides two methods to derive replacement loads that can be processed using a statistical approach, which approximate the fatigue damage that non-stationary loading would cause in dynamic structures. The first method involves using the inverse Fatigue Damage Spectrum (iFDS) approach [20] to derive a single stationary Gaussian replacement process. The second method decomposes loading into a limited set of Gaussian processes to abstract a quasi-stationary (QS) process [14]. This approach can be viewed as reversing the construction of the synthetic example data used in this presentation (Fig. 2). The QS implementation is

based on approximating the load spectra that are obtained in the process of calculating an FDS. This section aims to briefly present the main ideas using the previous example.

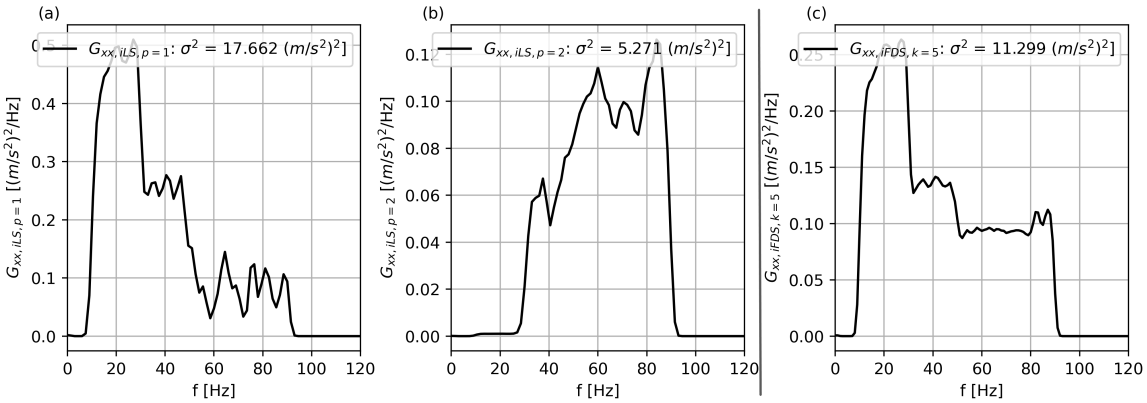


Fig. 8: PSDs derived for approximating the fatigue damage potential using *(a,b)* FDS load spectra approximation and *(c)* the inverse FDS for $k = 5$ [`sig.der_lsEquivalent(); sig.der_iFDS()`]

The method `der_lsEquivalent(R = integer inp.,...)` allows for the derivation of a new object that contains a set of R stationary processes, which can approximate the non-stationary behavior of the reference load. In our example (Fig. 1), when using $R = 2$ and applying this approach, the algorithm is able to closely approximate the original PSD shapes and their proportions (compare Figures 8a,b and 2), based solely on the FDS load spectra. It is important to note that this algorithm had been written to convey the approach's basic ideas, and it is recommended to refine and validate it before using it in real applications. Figure 8c shows the results of the inverse FDS `der_iFDS()` approach, which is based on obtaining a single stationary Gaussian process that generates the same FDS for a given stress-life curve exponent k. The resulting PSD shows an increase from the average PSD, which appears to be proportional to the variability of the two basis PSDs for frequency, resp., for the loading's FDS and NSM.

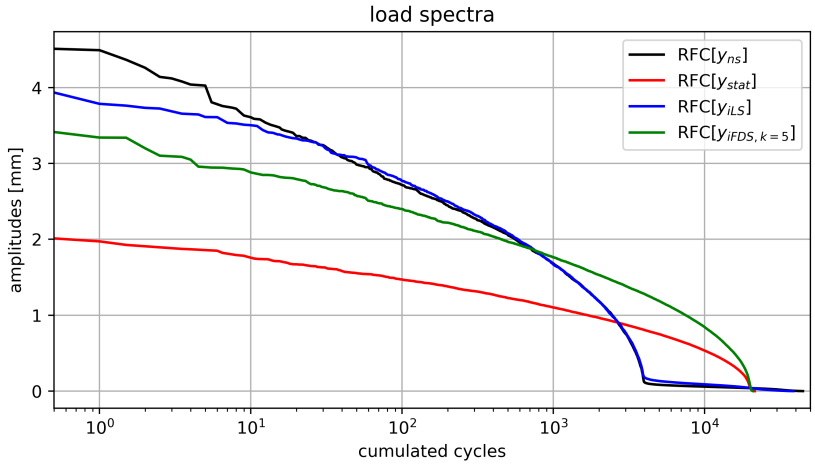


Fig. 9: Load spectra comparison of SDOF response $f_D = 20$ Hz subjected to different excitations [`sig.plot_ls()`]

Figure 9 summarizes these two approaches and compares them with the initial example using the load spectra obtained for the SDOF system with resonance frequency $f_D = 20$ Hz (Fig. 7). First of all it illustrates the deviations between the two response realizations $\{y_{\mathrm{ns}}(t), y_{\mathrm{stat}}(t)\}$ and thus between a statistical and a non-statistical assessment. The approximation methods introduced in this section offer improved results, but there are clear distinctions between them. The stationary iFDS method $y_{\mathrm{iFDS}}(t)$ is limited to stationary Gaussian processes, leading to a corresponding load spectra shape. While the FDS correspond, damage-equivalence is strictly limited to the specific stress-life curve exponent $k = 5$ used and to the simplified accumulation rule, the FDS is based on. In contrast, a quasi-stationary process $y_{\mathrm{iLS}}(t)$ can adapt significantly better to more realistic load spectra.

3 Conclusion

This article showcases the pyRaTS python package, which offers a set of tools for assessing the fatigue strength of mechanical structures under non-stationary vibration loading. It covers the limitations of the statistical and non-statistical approaches in structural dynamics and stresses the significance of incorporating non-stationary effects in a statistical assessment of structural dynamics. The methods presented, such as the Fatigue Damage Spectrum (FDS), non-stationarity matrix (NSM), and the framework for abstracting quasi-stationary (QS) replacement loads, provide detailed insights into the effects of non-stationary loading on structural dynamics and fatigue strength. The article covers a shortcoming of the popular FDS and proposes the solution of utilizing linear systems theory and the non-stationarity matrix to overcome the modal decoupling limitation of the FDS and to allow for estimating response kurtoses. Combining both may provide a viable future approach as it connects the information of how non-stationary loading affects fatigue strength for unique modes (FDS) and how their excitation is correlated in time (NSM). The pyRaTS package intends to be a valuable contribution to the field of structural dynamics by offering useful resources for researchers and engineers in the assessment of fatigue strength of mechanical structures.

References

[1] J. Slavič, M. Boltežar, M. Mršnik, M. Cesnik, and J. Javh, *Vibration fatigue by spectral methods: From structural dynamics to fatigue*. Elsevier, 2020.

[2] T. Dirlik, *Application of computers in fatigue analysis*. PhD Thesis, University of Warwick, 1985.

[3] D. Benasciutti and R. Tovo, "Spectral methods for lifetime prediction under wide-band stationary random processes," *International Journal of Fatigue*, no. 27, pp. 867–877, 2005.

[4] M. Palmieri, M. Česnik, J. Slavič, F. Cianetti, and M. Boltežar, "Non-Gaussianity and non-stationarity in vibration fatigue," *International Journal of Fatigue*, no. 97, pp. 9–19, 2017.

[5] C. Lalanne, *Specification delevopment: Mechanical vibration and shock analysis vol. 5*. London: ISTE, 2009.

[6] M. Decker, S. Kinscherf, N. Bauer, P. David, and M. Serifsoy, "Deriving fatigue equivalent power spectral density spectra for the vibration testing of engine components," *Materialwissenschaft und Werkstofftechnik*, vol. 49, no. 3, pp. 392–405, 2018.

[7] F. Kihm, S. A. Rizzi, N. S. Ferguson, and A. Halfpenny, "Understanding how kurtosis is transferred from input acceleration to stress response and its influence on fatigue life," *Recent Advances in Structural Dynamics 2013*, 2013.

[8] C. Braccesi, F. Cianetti, M. Palmieri, and G. Zucca, "The importance of dynamic behaviour of vibrating systems on the response in case of non-Gaussian random excitations," *Procedia Structural Integrity*, vol. 12, pp. 224–238, 2018.

[9] A. Zorman, J. Slavič, and M. Boltežar, "Short-time fatigue-life estimation for non-stationary processes considering structural dynamics," *International Journal of Fatigue*, vol. 147, p. 106178, 2021.

[10] A. Trapp and P. Wolfsteiner, "pyRaTS v0.2 (github.com/ArvidTrapp/pyRaTS)," 2023.

[11] A. Zorman and J. Slavič, "FLife v1.4.1 (10.5281/zenodo.7417587)," 2022.

[12] A. Zorman, D. Gorjup, and J. Slavič, "pyExSi v0.43 (github.com/ladisk/pyExSi)," 2021.

[13] A. Trapp and P. Wolfsteiner, "Frequency-domain characterization of varying random vibration loading by a non-stationarity matrix," *International Journal of Fatigue*, vol. 146, 2021.

[14] P. Wolfsteiner, A. Trapp, and L. Fertl, "Random vibration fatigue with non-stationary loads using an extended fatigue damage spectrum with load spectra," *International Conference on Noise and Vibration Engineering*, 2022.

[15] D. Benasciutti and R. Tovo, "Frequency-based fatigue analysis of non-stationary switching random loads," *Fatigue & Fracture of Engineering Materials and Structures*, no. 30, pp. 1016–1029, 2007.

[16] A. Trapp, M. J. Makua, and P. Wolfsteiner, "Fatigue assessment of amplitude-modulated non-stationary random vibration loading," *Procedia Structural Integrity*, vol. 17, pp. 379–386, 2019.

[17] P. Wolfsteiner and A. Trapp, "Fatigue life due to non-Gaussian excitation – An analysis of the fatigue damage spectrum using higher order spectra," *International Journal of Fatigue*, vol. 127, pp. 203–216, 2019.

[18] C. L. Nikias and A. P. Petropulu, *Higher-order spectra analysis: A nonlinear signal processing framework*. Englewood Cliffs, NJ: Prentice Hall, 1993.

[19] J. M. E. Marques, D. Benasciutti, and R. Tovo, "Variability of the fatigue damage due to the randomness of a stationary vibration load," *International Journal of Fatigue*, no. 141, 2020.

[20] M. Decker, "Vibration fatigue analysis using response spectra," *International Journal of Fatigue*, vol. 148, p. 106192, 2021.

A Python code

```
import numpy as np
import matplotlib.pyplot as plt
import pyRaTS as ts

''' Defining the synthetic example'''
fs    = 600             # sampling frequency [Hz]
dfpsd = 0.05            # frequency resolution
fpsd  = np.arange(0,fs/2+dfpsd,dfpsd)

''' PSD-shapes '''
psd1  = np.zeros(fpsd.shape)
psd1[(fpsd >= 10) & (fpsd < 30)] = 0.5
psd1[(fpsd >= 30) & (fpsd < 50)] = 0.25
psd1[(fpsd >= 50) & (fpsd < 80)] = 0.1
psd2  = np.zeros(fpsd.shape)
psd2[(fpsd >= 30) & (fpsd < 50)] = 0.0625
psd2[(fpsd >= 50) & (fpsd < 80)] = 0.1
psd2[(fpsd >= 80) & (fpsd < 90)] = 0.125
p = [0.2, 0.8]
avpsd = p[0]*psd1+p[1]*psd2

'''Generate realizations'''
T   = 1000
defswitch = []
defswitch.append({'psd': psd1,'fpsd':fpsd,'T':p[0]*T,'var': 'x_{ns,p=1}'})
defswitch.append({'psd': psd2,'fpsd':fpsd,'T':p[1]*T,'var': 'x_{ns,p=2}'})
defstat = []
defstat.append({'psd': avpsd,'fpsd':fpsd,'T':p[0]*T,'var': 'x_{stat,p=1}'})
defstat.append({'psd': avpsd,'fpsd':fpsd,'T':p[1]*T,'var': 'x_{stat,p=2}'})
sig    = ts.timeseries([defswitch,defstat])

''' Analysis (examples) '''
sig.plot()
# different access depending on configuration (multi-channel, multi-process)
sig.plot_psd()
sig.plot_psd(access='proc')
sig.plot_psd(access='comp')
# estimating and plotting non-stationarity matrix
ts.plot_nsm(sig.est_nsm(fl = 10,fu = 90))
# calculating and plotting FDS
ts.plot_fds(sig.est_fds(Nf = np.arange(10,100,2)))
# using FLife damage-estimators
sig.est_fds(method = FLife.method)
# deriving structural responses
respsig  = sig.der_sdofResponse(fD = 20,func = 'acc2dis4mm')
tf20     = ts.get_tfSDOF(fpsd,fD = 20, func = 'acc2acc')
tf65     = ts.get_tfSDOF(fpsd,fD = 65, func = 'acc2acc')
tf20_65  = tf20+tf65
respsigBiMod = sig.der_response(fpsd,tf20_65)
sig.der_iFDS(m = 5,plot = False,maxit = 200)
sig.der_lsEquivalent(R = 2, maxit = 200)
```

B Table of pyRaTS v0.2 methods

Tab. 3: Table of pyRaTS v0.2 methods

method	description	static	chan	proc	comp
`est_specMoms()`	calculate spectral moments for given PSD	x			
`est_dirlikD()`	EST fatigue damage via Dirlik method	x			
`get_tfSDOF()`	obtain basic SDOF-system's transfer function (TF)	x			
`get_tf()`	prepare linear TF	x			
`get_nsm()`	postprocess calculation of non-stationarity matrix	x			
`get_GaussianSeries()`	generate stationary Gaussian time series	x			
`get_weightFunc()`	get weighting function (optimization)	x			
`get_lsQsDirlik()`	get load spectra for QS loading using Dirlik	x			
`get_analyticalSignal()`	get analytical signal	x			
`soe_lsEquivalent()`	SOE for load spectral equivalence	x			
`soe_FDSEquivalent()`	SOE for FDS equivalence	x			
`plot()`	plot time series		x	x	x
`plot_psd()`	plot PSD that was lastly estimated via `est_psd()`		x	x	x
`plot_X()`	plot ABS of Fourier coefficients		x	x	x
`plot_tf()`	plot TF (interpretation: object is TF)		x	x	x
`plot_ls()`	plot load spectrum		x	x	x
`plot_nsm()`	plot non-stationarity matrix		x	x	x
`plot_fds()`	plot FDS of (multiple) signals		x	x	x
`est_psd()`	estimate power spectral density (PSD)		x	x	
`est_stats()`	estimate key statistical values		x	x	
`est_ls()`	DER/EST load spectrum via RFC/DK		x	x	
`est_dirlik()`	estimate Dirlik load spectrum and damage		x	x	
`est_fds()`	estimate Fatigue Damage Spectrum		x	x	
`est_nsm()`	estimate non-stationary matrix		x	x	
`der_statResponse()`	DER statistical response (PSD and NSM)		x	x	
`der_sdofResponse()`	DER response TS of SDOF system		x	x	
`der_sdofTransfer()`	DER single-degree-of-freedom system's TFs		x	x	
`der_response()`	DER TS of structural response for given TF		x	x	
`der_highpass()`	DER TS of ideal high pass filtered series		x	x	
`der_lowpass()`	DER TS of ideal low pass filtered series		x	x	
`der_bandpass()`	DER TS of ideal band pass filtered series		x	x	
`der_lsEquivalent()`	DER QS load definition using FDS load spectra		x	x	
`der_iFDS()`	DER load definition using inverse FDS		x	x	

Development of a multibody simulation framework with an emphasis on extensibility

Martijn Vermaut * Frank Naets

LMSD Research Group, Department of Mechanical Engineering, KU Leuven

Abstract

Novel developments in or tangent to the field of rigid and flexible multibody require a software platform to aid the development and to test and validate the algorithms in. The in-house MultiBody Research Code (MBRC) offers a Matlab-based modular object-oriented platform to satisfy this need. It was developed with mechanical engineering researchers in mind, as their expectations differ from those of software engineers. The extensibility focuses on three main areas. Firstly, it focuses on core structures such as easily switching out the body formulation without affecting the remainder of the model; it was e.g. used to develop the Flexible Natural Coordinates Formulation validated with the (also implemented) Floating Frame of Reference Formulation. Secondly, it focuses on additions within the field of mechanical modeling such as contact models, bushing models, etc. Thirdly, it focuses on additions in other fields such as e.g. (1) Kalman filtering for state-input-parameter estimation; (2) parameter and topology optimization such as the Adjoint Variable Method (exploiting the benefits of FNCF, hence implemented as FNCF-AVM); (3) co-simulation through the Functional Mockup Interface (FMI) or through a custom interface; and (4) extracting model definitions for use in other simulations tools such as the Bond-graph-based AMESim. The extensibility is guaranteed by standardizing the interface between the different model components; i.e. forces are always applied to mass-bearing components and intervention in the simulation (e.g. for co-simulation) is only possible between converged simulation steps. The platform is currently not open-source as the opportunities and practicalities of doing so are currently under investigation.

Keywords: simulation platform, multibody dynamics, flexible multibody dynamics

1 Statement of Need

Within the field of multibody research, it is typically a prerequisite to have a multibody simulation up and running, and preferably to have it as accessible as possible in terms of having open access to the equations and sensitivities. This seems like a trivial matter, but it is often overlooked. Commercial software is very accessible in terms of the user interface, but the equations are in turn shielded off from the user. Many multibody research codes already exist, but these are very often aimed towards one specific research goal as it often originates from one researcher and his research question. Researchers and starting PhD students more specifically thus typically start their research with developing a simplified framework to perform multibody simulations in which they can develop their novel research. There is thus a need for a generic simulation framework offering access to the underlying equations and their sensitivities, but also capable of providing the necessary tools for many different research questions. This led to the development of the LMSD in-house MultiBody Research Code (MBRC).

*Corresponding author, Email address: martijn.vermaut@kuleuven.be

2 Framework Development

The framework has been developed in Matlab as this is a scripting language that many mechanical engineers are familiar with. It also allows to link with many existing codes already developed within the research group. The framework has been developed to be modular, and this is partly achieved by implementing it as object-oriented code. As the framework is aimed to be used for multibody applications, the code is also structured as such. A multibody model contains many different elements. These are rigid bodies, flexible bodies, point masses, springs, dampers, contact models, co-simulated models, etc. The framework is implemented in such a way that all elements are handled in the same manner, i.e. the solver can call the same function on all of these elements without having to know what type of element it is. New elements can then also be implemented in a straightforward manner by having the same header for the evaluation function. This allows for a very powerful framework in which quite complex elements can be defined. It comes to no surprise that the MBRC contains element definitions for point masses, rigid bodies using the Cartesian coordinates and natural coordinates formulations, and flexible bodies using the floating frame of reference and flexible natural coordinates formulations [1]. It also contains element definitions for simple linear bushings (i.e. defined by local 6-by-6 constant stiffness and viscous damping matrices and a 6-element zero-length vector), but also a bushing library in which more advanced models can as easily be employed [2]. It also contains element definitions for a PID controller, a Dugoff tire model, and a NURBS-to-NURBS contact model. These three are discussed in more detail in the following section to highlight how the MBRC can be used to implement more advanced element definitions.

Since the MBRC uses generic functions calls to the elements to retrieve the required solver quantities (forces, constraint residual, etc.), it can just as easily be used to retrieve their sensitivities (i.e. tangent mass, damping, and stiffness matrices). Given that the code is completely open, and due to the modular nature of the element definitions, higher order sensitivities could also be implemented for the elements of interest without requiring everything to be implemented. This is especially useful when parametric sensitivities are required. The sensitivity of the generalized tangent stiffness matrix w.r.t. e.g. a bushing stiffness becomes quite easy to implement. This has been combined in an adjoint solver implementation with the flexible natural coordinates formulation to reduced the computational cost of these sensitivities even further [3]. The MBRC has also been used with the flexible natural coordinates formulation in a Kalman filter to estimate unknown forces [4].

Apart from evaluating elements in the context of the solver, the MBRC also provides generic evaluation functions to the bodies. In a multibody model it might be useful to have a collection of rigid and flexible bodies, as not all components may undergo equally important deformations. And these bodies may be implemented using different formulations depending on which is most suited for each. When implementing a connection element (e.g. a bushing or a kinematic constraint), the implementation should not depend on the rigid or flexible nature of the body or even the formulation used. The MBRC introduces `frames` as a generic interface class to the bodies. The function call to the frames are all the same, but internally quantities such as positions and rotational velocities are computed differently depending on the formulation.

3 Use Cases

In this section, three use cases are discussed to highlight how complex elements can be defined.

3.1 PID Controller

A first element definition to be discussed is a PID controller. This is not a very complicated model, but it should be noted that a mechanical system is typically represented by second order partial differential equations, and in this case kinematic constraint equations make it index-3 differential-algebraic equations; whereas a PID controller is typically represented with a single state and a first-order ordinary differential equation: $\varepsilon = x - x_{ref}$. This is actually quite easily mended by treating ε as a velocity coordinate, which automatically gives the integral error as the position coordinate. The main difficulty with this model in a multibody context, is its non-colocated nature. A sensor signal is retrieved at one location of the model and a force is applied at a different location of the model. This results in the tangent matrices to be non-symmetric, and the typical multibody solvers do not like this. But it also poses a programming challenge in how to connect everything together. As the MBRC offers `frames` as a generic interface to the bodies, this becomes quite easy to manage. As an example, figure 1 shows the response of a mass-spring-damper model to a PID controller in the MBRC with varying values for the I-parameter.

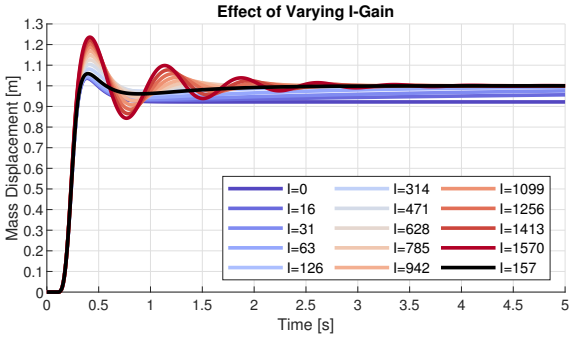


Fig. 1: MBRC simulation of a PID controller

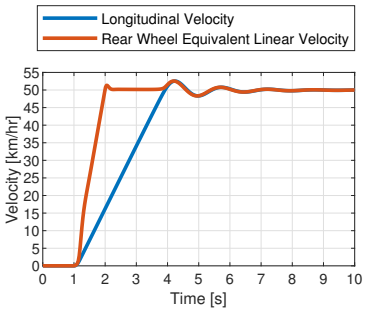


Fig. 2: MBRC simulation of a Dugoff tire model

3.2 Dugoff tire model

A second model to be discussed is a Dugoff tire model. In essence this is not a difficult model to implement, but a lot of things come together. Firstly, a planar constraint needs to be defined such that the tire does not drop through the floor. Secondly, a relaxation factor (and thus a state) needs to be implemented and defined by an equation to simulate the (lumped) delay effect of the rubber generating forces only after deforming. Lastly, forces need to be applied on two different bodies: the ground, and the wheel. This means that this model defines a constraint equation and thus a Lagrangian multiplier, it defines a state, and it applies forces to the states of two other elements in the MBRC model. For testing purposes, a simple 4-wheel vehicle model with independent vertically moving wheels was implemented. Figure 2 shows the linear velocity of the vehicle itself, and the equivalent linear velocity of the rear wheels at which the torque was being applied. It should be noted that this torque was being applied by the PID controller discussed before. This also highlights how these different models can all be used together without any interdependencies.

3.3 NURBS-to-NURBS contact model

Whereas the two previous examples could be considered lumped force elements in the multibody simulation, a final example shows that even very complicated models can be integrated in the framework. A rigid contact model based on NURBS surfaces, a Hertzian normal force computation, and a rolling contact model is implemented. The contact detection comprises of a Bounding Volume Hierarchy global contact search, and a local analytical refinement, all implemented in C++ as opposed to Matlab for efficiency. For the rolling contact the options are a Coulomb friction model, a linear Kalker model, a Polach model, and a FASTSIM2-based implementation of the non-linear Kalker model. This model is then used to model a train bogie and the contact forces between the tracks and the wheels. Figure 3 shows the normal forces of the four wheels during a cornering maneuver approaching a derailment. Whereas this model conceptually is nothing more than a complex spring, it highlights how even highly complicated models can be used in the modular framework of the MBRC.

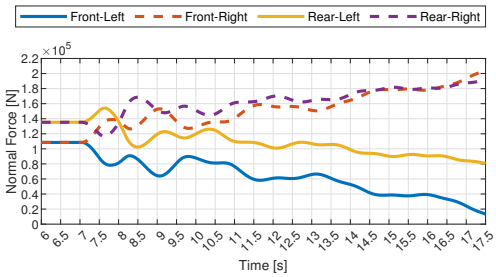


Fig. 3: Normal force of the four wheels of the train bogie

4 Conclusions and Open-Source Ambitions

The MBRC is currently an in-house only research code. Before making it open-source to a wider audience, it is first investigated within the research group what the benefits are of having access to the source code as opposed to using the MBRC as a black box to run simulations. The main challenge here is to identify the two roles someone may have when interacting with the MBRC: a user or a developer and how to make the code as accessible as possible for both roles. For instance, the documentation should be clearly structured accordingly in a user's manual and a developer's manual in order to facilitate picking up the code and working with it in a meaningful manner.

References

[1] M. Vermaut, W. Desmet, and F. Naets, "Flexible natural coordinates formulation (fncf): Towards simpler equations of motion for flexible multibody models," 2022-05-10.

[2] R. Adduci, W. Desmet, and J. Croes, "Multibody model-based virtual sensing. a generic indirect sensing approach for mechatronic systems with nonlinear bushing connections," 2022-01-31.

[3] S. Vanpaemel, W. Desmet, and F. Naets, "Sensitivity analysis of flexible multibody systems. with applications in parameter, topology and input optimization," 2022-01-27.

[4] E. Risaliti, W. Desmet, and B. Cornelis, "Model based virtual sensors for wheel center loads and full strain field on vehicle suspension components," 2019-11-28.

Multi-scale and full-field vision-based motion tracking for flexible multibody parameter identification

Thijs Willems * Frank Naets

Department of Mechanical Engineering, KU Leuven, Celestijnenlaan 300, B-3001 Heverlee, Belgium
Flanders Make@KU Leuven, Flanders Make, Belgium

Abstract

Nowadays many open source image processing packages are available implementing the latest state-of-the-art motion tracking algorithms (e.g., OpenCV, pyIDI). Some of these algorithms are implementing very accurate subpixel tracking for small motions (e.g., Lucas-Kanade optical flow) where others implement the, less accurate, detection of large motions (e.g., ArUco marker detection). However, for flexible multibody mechanisms, a combination of both is required as the components of these mechanisms undergo both large rigid body translations and rotations, and small deformations. Therefore, this research shows a multi-scale tracking approach that is able to track these combined motions with subpixel accuracy. In a first stage, the large rigid body translation and rotation motion is estimated by tracking discrete, well-positioned ArUco markers. Based on this estimated rigid body motion, a first guess can be extracted for the positions of the component feature points to track (e.g., speckle or checker pattern applied on the component). These guesses will then be refined to subpixel accuracy such that the component deformations are tracked on an accurate level. Allowing to track these motions up to an accuracy level which makes the individual dynamics of the components visible allows to extract relevant functional measurements of multibody mechanisms which would be impossible with current measurement techniques (e.g., accelerometers). In this research, two application cases were used: a weaving loom which includes the tracking of large translations and the handling of occlusions and a slider-crank mechanism which shows large component rotations.

Keywords: Vision-based motion tracking, Large translations and rotations, Flexible multibody mechanisms

1 Introduction

An important aspect in the field of mechanical system identification is the measurement of the dynamic response of components and systems. In the state-of-the-art many measurement techniques are already available for many years like accelerometers, laser-doppler vibrometers, strain gauges, etc. However, for measuring the dynamic response of (flexible) multibody mechanisms the measurement data is often limited to encoder data or data from discrete sensors on the system housing as large rigid body motions make a sensor placement on the components difficult. Here, the increasing use of vision-based measurements, which are contactless and full-field, can resolve this issue and provide a dense data set for multibody mechanisms.

*Corresponding author, Email address: thijs.willems@kuleuven.be

Previous research [1, 2] has shown the benefits of vision-based measurements for structural identification on single structural components by exploiting the spatial overdetermination of these measurements. It is the goal of the author to leverage this previous research for its use in flexible multibody mechanisms. However, to achieve this goal the previously used open-source motion tracking methodologies (e.g., the OpenCV Lucas-Kanande optical flow [3]) could not directly be reused as they failed to track the motion in the presence of the large translations and rotations (e.g., using an incremental instead of an absolute tracking will result in drifting errors [4]). Therefore, this research presents a motion tracking framework (Section 2) which is able to track the subpixel component deformations together with the rigid body translations and rotations and shows its application on two cases in Section 3. The last section (Section 4) gives the main conclusions of this research.

2 Multi-scale tracking method

An overview of the workflow of the proposed multi-scale motion tracking procedure is given in Fig. 1. The proposed tracking method relies on the presence of ArUco markers on the tracked components

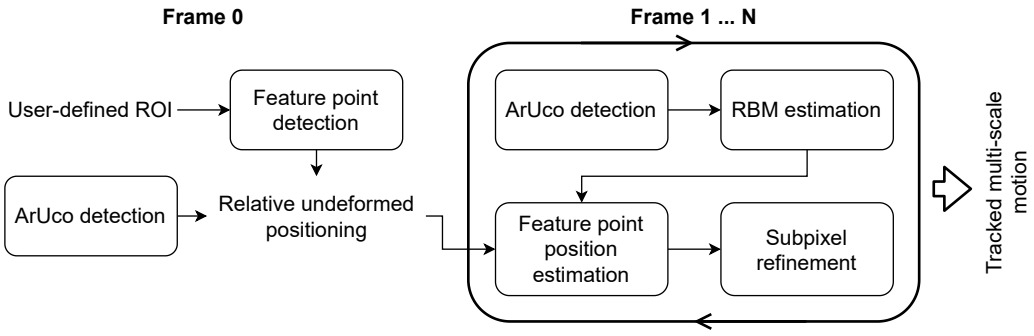


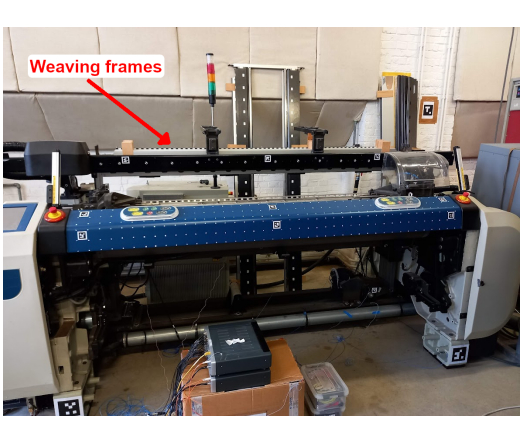
Fig. 1: Workflow of the multi-scale tracking approach.

(at least one ArUco marker for each component is needed). In the first frame of the image sequence ('Frame 0') these ArUco markers are detected through the open-source OpenCV ArUco detector [5] together with good feature points to track (e.g., Shi-Tomasi features [6]) within a user-defined region-of-interest (ROI). Based on these detected initial positions, the relative positions of the feature points with respect to the ArUco marker(s) are defined and stored for use during the processing of the subsequent frames ('Frames 1 ... N'). For these frames, the multi-scale tracking starts with detecting the ArUco marker(s) again in every frame to extract the rigid body motion of the component(s). By assuming the deformation of the component(s) is much smaller than the rigid body motion, a good guess of the feature points positions in every frame can be calculated from the ArUco marker(s) position(s) in the frame and the stored relative positions of the feature points from the first frame. Afterwards, a subpixel refinement method can be used to refine these guesses to subpixel accuracy in order to extract accurate component deformation motions (e.g., through the state-of-the-art OpenCV Lucas-Kanande optical flow [3] or subpixel corner refinement [7]). An additional benefit of the proposed methodology is the straight-forward handling of occlusions. Through the ArUco marker detection an occlusion of the component can easily be detected and after the occlusion resolves the tracking can easily be resumed without having to re-initiate the tracking algorithm. This will also become clear in the first application case in Section 3.

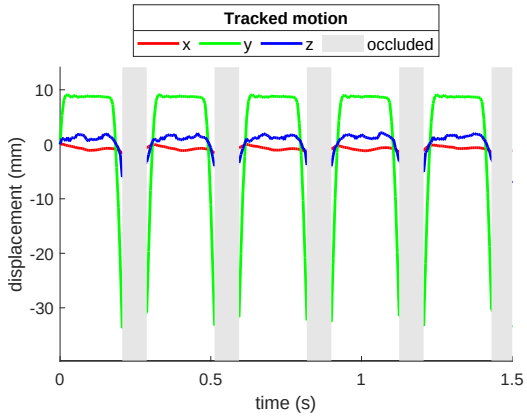
3 Application cases

In this research, the proposed multi-scale tracking method is applied on two cases: an industrial weaving loom and a slider-crank setup.

The first application case is shows in Fig. 2. Fig. 2a shows the weaving loom setup and indicates the weaving frames that are tracked during this research. On the weaving frames, a checker pattern was applied together with two ArUco markers (one on each end of the weaving frame) to perform the tracking. From the figure, as the checker pattern and ArUco markers are not fully visible, it can be seen that the frames are occluded by other components of the weaving loom during their motion. Fig. 2b shows the tracked motion in the camera reference frame of one of the checker corners in the middle of the weaving frame. The shaded areas in the figure indicate occluded time instances.



(a) Weaving loom setup



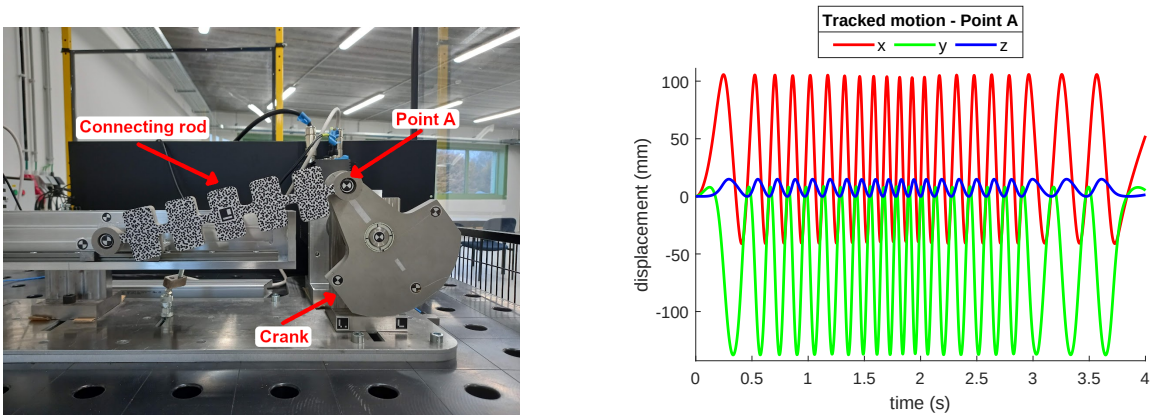
(b) Tracked motion

Fig. 2: Used weaving loom setup (a) and the motion tracking results of a selected feature point (b).

The second application case is shown in Fig. 3. Fig. 3a shows the slider-crank setup and indicates the crank, the connecting rod and point A of which the tracked motion is shown in Fig. 3b. Discrete markers were applied on the joints of the mechanism while a speckle pattern and one ArUco marker cover the connecting rod. This application case shows that the method is also able to track large rotations of the components.

4 Conclusion

This research presents a multi-scale tracking approach for the motion tracking of a flexible multibody mechanism. It combines an ArUco-based rigid body motion tracking with a subpixel refinement in order to track component deformations. Two application cases also show how the method behaves in the case of occlusions or large rotations. The method allows to extract full kinematic information in time of the studied multibody mechanism which enables the development of new experimental model identification methodologies in the future.



(a) Slider crank setup

(b) Tracked motion

Fig. 3: Used slider-crank setup (a) and the motion tracking results of a selected feature point (b).

5 Acknowledgements

Internal Funds KU Leuven are gratefully acknowledged for their support. The Research Foundation – Flanders (FWO) is gratefully acknowledged for its support through research grant no. G095120N. This research was partially supported by Flanders Make, the strategic research centre for the manufacturing industry.

References

[1] F. S. Egner, Y. Wang, T. Willems, M. Kirchner, B. Pluymers, W. Desmet, J. Palandri, B. Reff, and F. Wolf-Monheim, "High-speed camera based experimental modal analysis for dynamic testing of an automotive coil spring," pp. 278–288, 2022. [Online]. Available: https://doi.org/10.4271/2021-01-1119

[2] T. Willems, F. S. Egner, Y. Wang, M. Kirchner, W. Desmet, and F. Naets, "Time-domain model identification of structural dynamics from spatially dense 3d vision-based measurements," *Mechanical Systems and Signal Processing*, vol. 182, p. 109553, 2023. [Online]. Available: https://doi.org/10.1016/j.ymssp.2022.109553

[3] J.-Y. Bouguet, "Pyramidal Implementation of the Affine Lucas Kanade Feature Tracker," *Intel corporation, Microprocessor Research Labs*, 1999. [Online]. Available: http://robots.stanford.edu/cs223b04/algo_affine_tracking.pdf

[4] T. Willems, "Accurate camera based 3D displacement measurements in service to modal analysis: Algorithm development and experimental validation," M.S. Thesis, KU Leuven, 2020.

[5] OpenCV, "Detection of ArUco Markers." [Online]. Available: https://docs.opencv.org/4.x/d5/dae/tutorial_aruco_detection.html

[6] J. Shi and C. Tomasi, "Good features to track," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. CVPR-94*, vol. 178. Seattle, WA, USA: IEEE, 1994, pp. 593–600. [Online]. Available: https://doi.org/10.1109/CVPR.1994.323794

[7] W. Förstner and E. Gülch, "A fast operator for detection and precise location of distinct points, corners and centres of circular features," in *Proc. ISPRS intercommission conference on fast processing of photogrammetric data*, vol. 6. Interlaken, 1987, pp. 281–305.

EMA and UFF: Fundamental Tools in Structural Dynamics

Klemen Zaletelj * Janko Slavič Miha Boltežar

Faculty of Mechanical Engineering, University of Ljubljana

Abstract

Experimental modal analysis (EMA) is one of the fundamental methods in the field of structural dynamics. The modal parameters (natural frequencies, damping ratios and modal shapes), identified by EMA, play a vital role in various applications, such as structural health monitoring, model updating, and vibration fatigue. It is of great importance that the tools to perform the analysis are readily available, reliable and that the implementation is verifiable. An open-source Python package SDyPy-EMA that implements an efficient and widely used Least Squares Complex Frequency (LSCF) and Least Squares Frequency Domain (LSFD) methods is presented. The SDyPy-EMA is the successor of the pyEMA package and is the first 1st level package in the SDyPy framework. To transfer and store the experimental data and analysis results, the Universal File Format is used. The pyUFF package, for reading and writing UFF files was created. A short history of the SDyPy-EMA and pyUFF packages along with the basic usage is presented in this article.

Keywords: Experimental Modal Analysis, Universal File Format, SDyPy framework

1 Statement of Need

In the field of structural dynamics, the identification of modal parameters (natural frequencies, damping ratios, and modal shapes) from experimental data is one of the most important tasks. The modal parameters themselves are either the desired outcome (*e.g.*, determination of the optimal operating frequency) or they are needed for further analysis. Structural health monitoring, model updating [1] and vibration fatigue [2] are some examples where the modal parameters play an important role.

Numerous algorithms have been presented for estimating modal parameters based on the measured Frequency Response Functions (FRFs) [3]. To enable the development of new scientific methods and to ensure the repeatability of the research in which the methods are used, the implementation of the algorithms must be accessible, the code verifiable, and, if necessary, adaptable to the specific needs of the researchers. One of the most accurate, efficient, and widely used algorithms for modal identification is the Least Squares Complex Frequency (LSCF) method [4] in combination with Least Squares Frequency Domain (LSFD) method [5]. Since LSCF and LSFD are widely used in scientific research and industrial applications, open development was considered ideal for implementing these algorithms.

Without proper storage and convenient transfer of data, the results of experimental work and analysis are not reusable. The Universal File Format (UFF) [6] standard was introduced in the 1960s to

*Corresponding author, Email address: klemen.zaletelj@fs.uni-lj.si

address this issue. To ensure the correct implementation of the UFF standard, the open development of a Python package for reading/writing the UFF files is necessary.

2 SDyPy-EMA: road to 1st level package

Implementation of the LSCF and LSFD algorithms in Python began nearly a decade ago; however, the first open-source development began in 2018 with the creation of the Python package pyEMA [7]. Open development was initially seen as a convenient way to share the latest code within the laboratory but it soon became clear that the benefits were much greater (e.g., developer contributions, improved research repeatability). To extend the benefits of open-source development to a broader field of structural dynamics, the SDyPy framework was founded. SDyPy is a conglomerate of smaller, highly focused packages (such as pyEMA) in the field of structural dynamics.

The first five SDyPy Enhancement Proposals (SEPs) were assembled to create the basic structure of the framework. The SEPs are still being actively developed and improved. One of the most important SEPs is SEP 1, which defines the 4 levels of package integration (see the SDyPy documentation for a full explanation):

- 4th level: external packages associated with structural dynamics.
- 3rd level: packages that conform to the SDyPy template, have an MIT or similar license, and are available through the Python Package Index.
- 2nd level: namespace packages with code tests and all elements of 3rd level.
- 1st level: 2nd level packages, but developed under the SDyPy organization.

In 2018, when the SDyPy framework did not exist, pyEMA would have been a level 4 package. In 2019, when the documentation and MIT license were added, pyEMA would have become a level 3 package. With the introduction of the SDyPy framework, pyEMA was the first package to move up to the 2nd level by becoming a namespace package. PyEMA could then be installed and imported via SDyPy. Recently, the pyEMA repository was moved to the SDyPy organization and pyEMA became the first 1st level package in the SDyPy framework. As one of the fundamental modules in SDyPy (see SEP 4), pyEMA was renamed SDyPy-EMA.

3 pyUFF: the storage

To standardize the transfer and storage of data in structural dynamics, the Structural Dynamics Research Corporation (SDRC) has defined the Universal File Format (UFF) standard [6]. The UFF is particularly often used in the field of modal analysis. With the introduction of binary set 58, which is commonly used for storing FRFs, the UFF is equipped to store data from modern, large-scale dynamic tests.

To take advantage of the UFF standard, the Python package pyUFF was developed. pyUFF enables the reading and writing of sets. Recently, the package has been updated to simplify the preparation of data for writing. An important part of any open-source project is the documentation, as it allows the user to properly use the package and explore the available features. The pyUFF documentation defines all currently supported sets and gives examples of how to use them.

Since UFF is closely related to Experimental Modal Analysis, the SDyPy-EMA package has been updated to allow direct import of UFF files. An example can be seen in Sec. 4.

4 SDyPy-EMA usage example

Import the `EMA` module and create an instance of the `Model` class by inputing the arguments:

```
from sdypy import EMA

a = EMA.Model(
    frf_matrix,
    frequency_array,
    lower=10,
    upper=10000,
    pol_order_high=60,
    driving_point=3,
    frf_type='accelerance')
```

or read the UFF:

```
a = EMA.Model()
a.read_uff(filename)
```

The main inputs in the `Model` class are the FRFs, arranged in a two-dimensional numpy array with shape `(n_locations, n_frequencies)`, and the one-dimensional vector of frequencies. To set the frequency limits analysis, the `lower` and `upper` arguments are used. The `pol_order_high` argument determines the highest polynomial order used to approximate the `frf_matrix`. The `frf_matrix` will be approximated with increasing orders of the polynomial. `driving_point` argument is the index of the `frf_matrix` at the driving location. The `frf_type` argument gives information about what type of FRF is used (accelerance, mobility, receptance).

The poles for increasing polynomial orders are computed:

```
a.get_poles()
```

After the poles are computed, the stable ones and physically meaningful poles must be selected. To display the stability chart:

```
a.select_poles()
```

To identify the modal constants at the selected poles, the `get_constants()` method must be called. `get_constants()` returns the reconstructed FRFs H and the modal constants A.

```
H, A = a.get_constants(method='lsfd')
```

If the `driving_point` argument was passed to the `Model` class, the modal shapes are accessed:

```
modal_shapes = a.phi
```

5 Conclusions

The open development of the SDyPy-EMA and pyUFF packages was paramount for the reliable implementation of the LSCF/LSFD algorithms and the UFF standard. Further development of the packages continues in accordance with open-source guidelines to drive open, peer-reviewed, and repeatable research.

References

[1] M. Friswell and J. E. Mottershead, *Finite element model updating in structural dynamics*, vol. 38. Springer Science & Business Media, 2013.

[2] M. Mršnik, J. Slavič, and M. Boltežar, "Vibration fatigue using modal decomposition," *Mechanical Systems and Signal Processing*, vol. 98, pp. 548–556, 1 2018.

[3] N. M. M. Maia and J. M. M. e Silva, *Theoretical and Experimental Modal Analysis*. John Wiley & Sons, 1997.

[4] P. Guillaume, L. Hermans, and H. Van der Auwerer, "Maximum Likelihood Identification of Modal Parameters from Operational Data," *Proceedings of the 17th International Modal Analysis Conference (IMAC17)*, pp. 1887–1893, 1999.

[5] B. Cauberghe, *Applied frequency-domain system identification in the field of experimental and operational modal analysis*. PhD thesis, Vrije Universiteit Brussel, 2004.

[6] Structural Dynamics Research Corporation, "Universal file format." `https://www.ceas3.uc.edu/sdrluff/`, 2023. [Online; accessed 19-April-2023].

[7] K. Zaletelj, Tomaž Bregar, D. Gorjup, and J. Slavič, "ladisk/pyEMA: v0.24," 2020.

CIP - Kataložni zapis o publikaciji (CIP) pripravili v Narodni in univerzitetni knjižnici v Ljubljani

COBISS.SI-ID 156645379
ISBN 978-961-7187-00-7 (PDF)