APPLIED MECHANICS 2018

April 9 - 11, 2018, Myslovice, Czech republic

POST-PROCESSING TECHNIQUES FOR RESPONSE OF NON-LINEAR SYSTEMS

L. Smolík[1]

Abstract: Challenges in modern engineering require implementation of more and more complex computational models. The computational models are often non-linear and even a slight modification of a design parameter can cause qualitative changes in a predicted response. Several visualisation methods which improve the understanding of such changes are presented in this paper. All proposed methods are illustrated with plots and diagrams and moreover, Matlab codes which were used in order to generate the illustrations are shown and discussed.

Keywords: post-processing; data visualisation; non-linear system; code samples; Matlab

1 Introduction

The visualisation and the evaluation of a response of non-linear systems is quite a challenging task. The response — or output — can be visualised employing standard 2-D and 3-D diagrams. However, such diagrams are not suitable for the analysis of qualitative changes of the response. This paper presents state-of-the-art techniques which can be used in order to post-process and visualise such data. Furthermore, some improvements of existing techniques are discussed and appropriate Matlab codes are presented.

2 Parameters of a sample non-linear system

The application of post-processing techniques that are introduced and discussed in this articles is shown for the case of a simple shaft supported on two journal bearings which is shown in Figure 1. The shaft's speed is regulated by a controller which is coupled to the shaft with a bellows coupling whose parameters are shown in Table 1. The non-linearity is introduced to the system through hydrodynamic forces acting in the bearings whose parameters are summarised in Table 2. The hydrodynamic forces are evaluated from hydrodynamic pressure in lubricant films, which is governed by the Reynolds equation. The Reynolds equation is derived and discussed in detail in [4].

Corresponding equations of motion are formulated employing the theory of dynamics of multi-body systems [2]. The equation of motion is in the form of a system of differential algebraic equations, which is introduced and solved in [3].

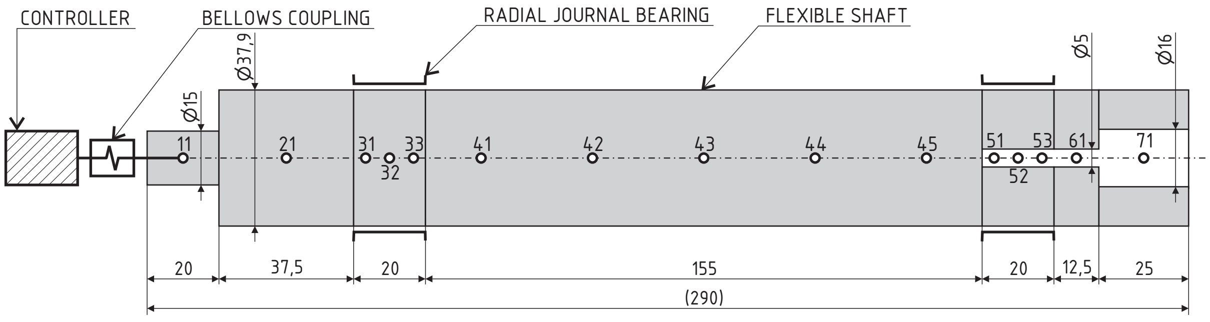

Figure 1: Geometry of the sample shaft.

[1] Luboš Smolík; Department of Mechanics, University of West Bohemia; carlist@kme.zcu.cz

Direction	*Stiffness*	*Damping*
torsional	9,000 N·m·rad^{-1}	4.5 N·m·rad^{-1}·s^{-1}
axial	30,000 N·m^{-1}	15 N·m^{-1}·s^{-1}
lateral	320,000 N·m^{-1}	160 N·m^{-1}·s^{-1}

Table 1: Parameters of the bellows coupling.

Parameter	*Value*	*Unit*	*Parameter*	*Value*	*Unit*
bearing diameter	38.0	mm	ambient pressure	1.0	bar
bearing length	20.0	mm	cavitation pressure	0.98	bar
radial clearance	40.5	μm	supply pressure	1.25	bar
lubricant temperature	50.0	°C	diameter of supply hole	5.0	mm
lubricant viscosity at temperature	28.3	mPa·s	angle to supply hole	270.0	°

Table 2: Parameters of the journal bearings.

3 Time series analysis

Let us assume that we analyse the vibrations of the shaft in the left bearing (see Figure 1). Shaft's horizontal and vertical displacements are stored in vectors x and y, respectively, vectors dotx and doty contains velocities, and vectors t and phi hold time the angle of shaft's rotation, respectively.

The most basic method for the visualisation of time dependent data is 2D plot with time t on the horizontal axis and a dependent variable on the vertical axis, which can be generated be plot(t, x) or line(t, x) commands. The later command is ca. $9\times$ faster than the first one [5].

Although widely used, the standard 2D plot might be unsuitable for the visualisation of coupled planar motions such as lateral vibrations of rotating structures and transverse vibrations of beams. Such motions can be visualised as trajectories by line(x, y) command. The resulting diagram, however, lacks any information about time patterns and trends. This problem can be solved if the trajectory is divided into individual cycles and the trajectory during each cycle is drawn with a different colour as shown in Figure 2a. If n cycles have to be visualised and each cycle starts at reference angle $\varphi = 2\,k\,\pi$, where $k \in \mathbb{Z}$, then Figure 2a is generated by the following code:

```
cmap = colormap(parula(n));        % use default 'parula' colormap
[~, locs] = findpeaks(cos(phi));   % find start of each cycle
figure; hold on;
for i = 1:n
    % plot a trajectory during a cycle
    plot(x(locs(end-n-1+i):locs(end-n+i)-2), ...
         y(locs(end-n-1+i):locs(end-n+i)-2), 'color', cmap(i, :));
    % plot the start of the cycle
    plot(x(locs(end-n-1+i)), y(locs(end-n-1+i)), 'k.');
end
```

Note that the trajectory is interrupted shortly before its corresponding cycle ends. This technique can be used in order to indicate the direction of the trajectory.

It is important to mention that the trajectories can be plotted not only in absolute coordinates but also in coordinates relative to a gross motion of a body (Figure 2b) or in a *state space* in the form of phase trajectories (Figures 2c and 2d). The relative and the phase trajectories are often useful for the testing of a stability, a periodicity, a quasi-periodicity or chaotic behaviour [1].

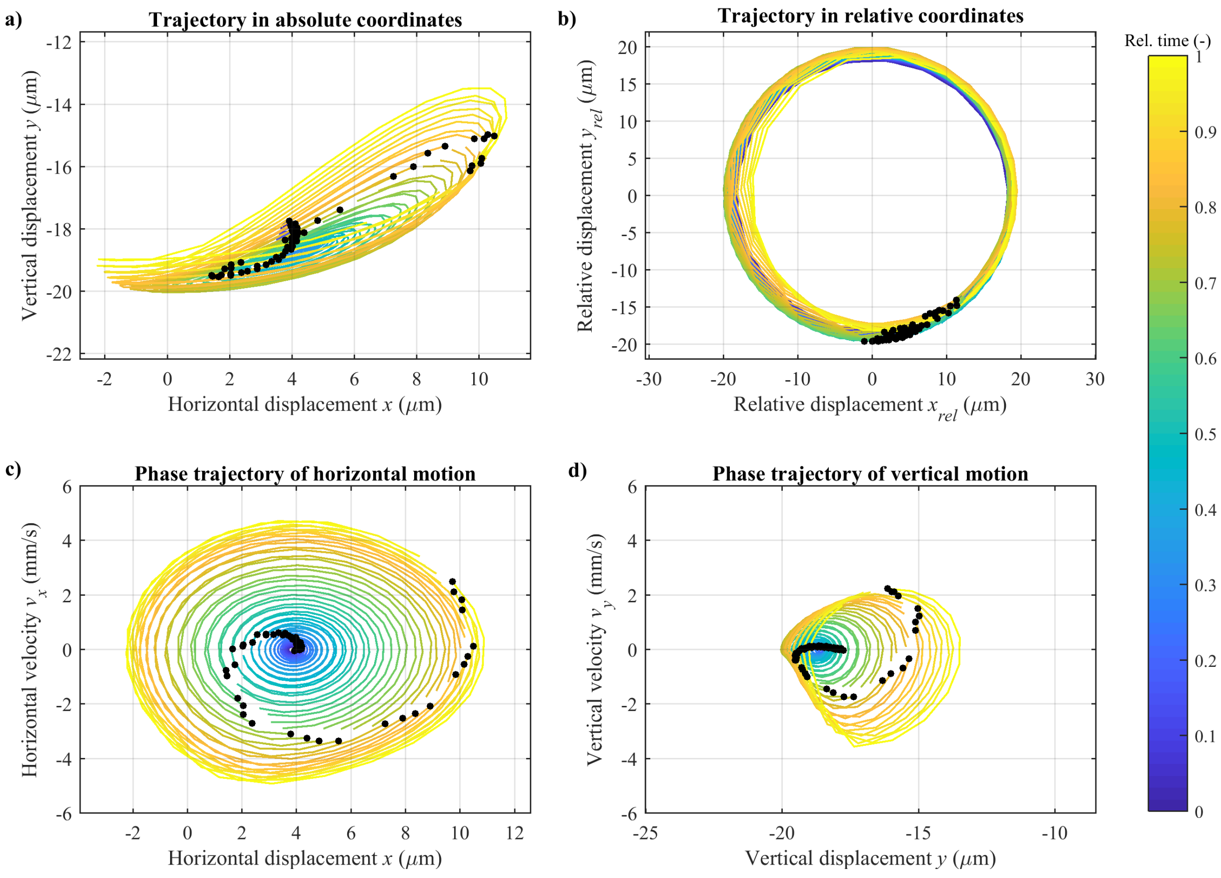

Figure 2: Sample trajectories in absolute a) and relative coordinates b) and phase trajectories for horizontal c) and vertical motions d) in the absolute coordinates. Black dots represent the position at the start of a cycle.

4 Parametric analysis of time series

Let us assume that we analyse the influence of an independent design parameter stored in vector `par` on vibrations of the shaft. Resulting displacements are stored in arrays `X` and `Y`, velocities in arrays `dotX` and `dotY` and the reference angles in array `Phi`. All arrays are arranged so that the i-th column contains a response for the i-th value of the design parameter. Data in the arrays can be visualised by many functions: `contour, mesh, surf` and `waterfall` to name a few. Resulting 3-D plots carry more information than 2-D plots but are less clear. When an appropriate dimensional reduction is used, important pieces of information can be preserved and data are presented in an easier form.

Dynamical systems can be effectively studied employing the *Poincaré maps*. The Poincaré map is the intersection of a phase trajectory with a certain lower-dimensional subspace called the *Poincaré section* [1]. The map with the section where derivative of the velocity is zero can be generated by

```
figure; hold on;
for i = 1:length(par)
    ddotX = diff(sign(dotX(:, i))); % find zero crossings
    indUp = find(ddotX > 0);         % crossings with positive slope
    indDown = find(ddotX < 0);       % crossings with negative slope
    plot(par(i) * ones(1, length(indDown)), X(indDown, i), 'k.');
    plot(par(i) * ones(1, length(indUp)), X(indUp, i), 'y.');
end
```

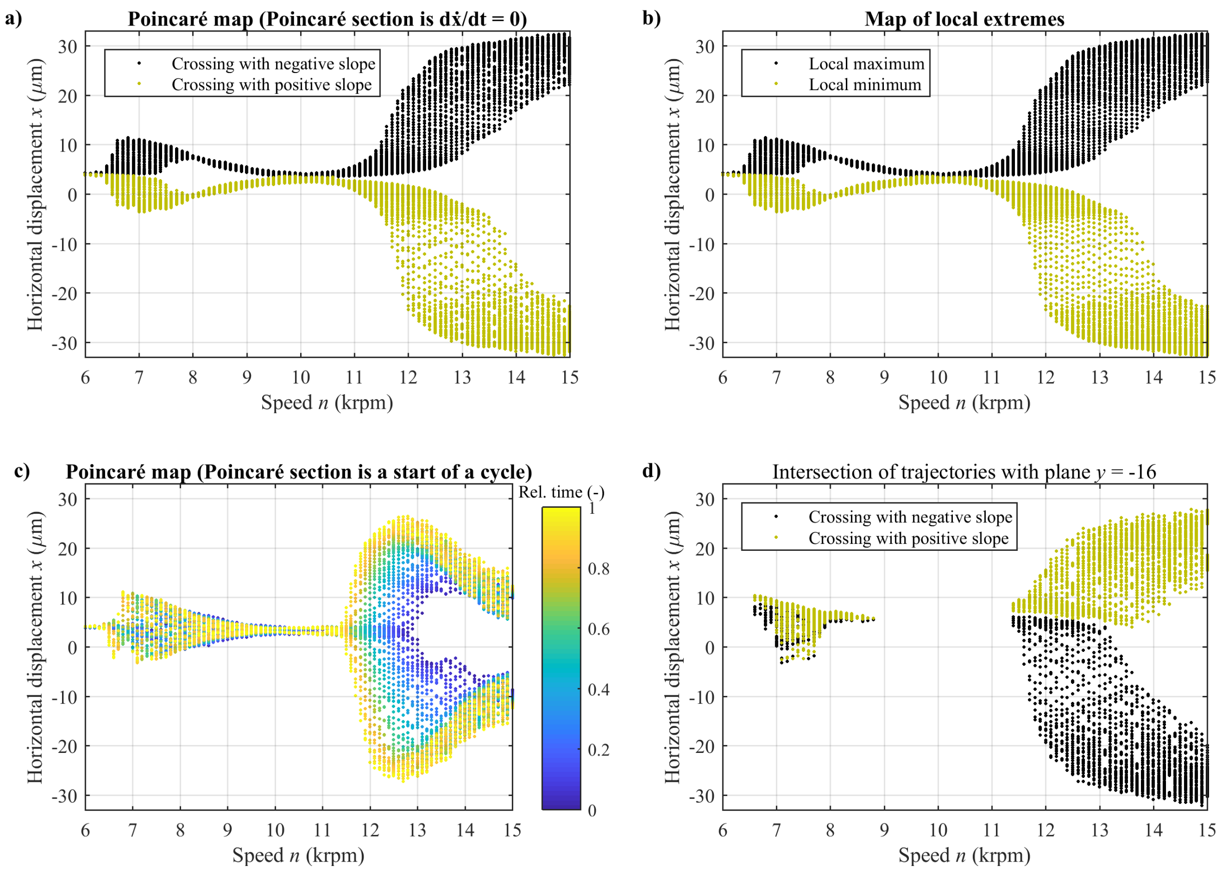

Figure 3: Several ways which can be used in order to visualise multi-parametric data.

The resulting map is shown in Figure 3a. Note that crossings of the Poincaré section with a positive slope are drawn with a different colour than crossings with a negative slope.

The Poincaré map from Figure 3a can be approximated if local extremes of the displacement are localised and depicted in dependence on the design parameter. Such a plot is shown in Figure 3b and can be generated by the following code:

```
figure; hold on;
for i = 1:length(par)
    locMax =   findpeaks(X(:, i));  % find local maxima
    locMin = - findpeaks(-X(:, i)); % find local minima
    plot(par(i) * ones(1, length(tempMax)), tempMax, 'k.');
    plot(par(i) * ones(1, length(tempMin)), tempMin, 'y.');
end
```

The points depicted in Figures 3a and 3b are coloured according to their type (positive/negative slope and local minimum/maximum). Another useful techniques is to appoint a colour in compliance with the order of appearance. It means that the point which is drawn as the first is for example black and subsequent points are brighter. The colour can be also changed with the reference angle, time or another user-defined reference. This type of diagram is depicted in Figure 3c and can be generated by

```
figure; hold on;
for i = 1:length(par)
    [~, locs] = findpeaks(cos(Phi(:, i))); % find start of each cycle
    cmap = colormap(parula(length(locs))); % generate colors
```

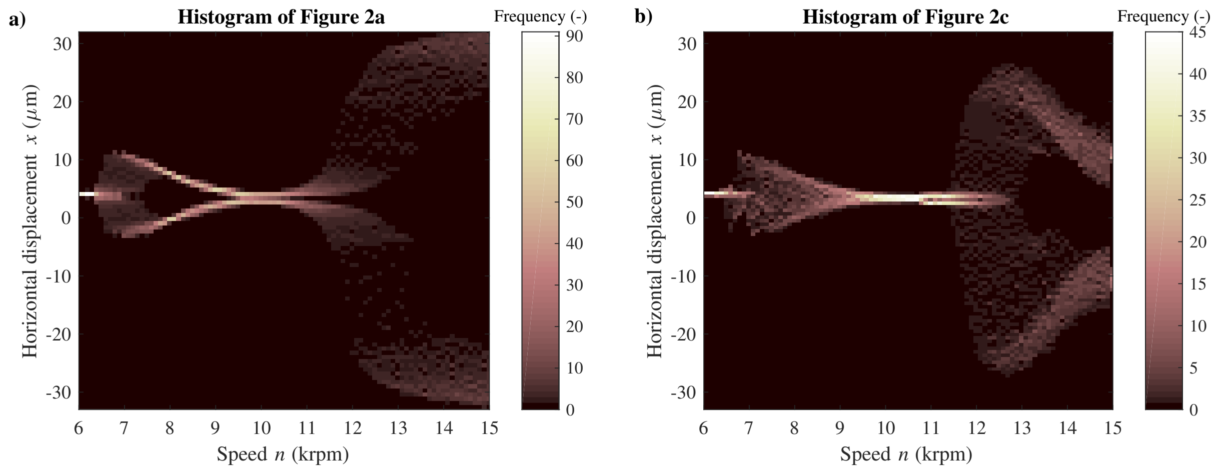

Figure 4: Depicted histograms estimates the frequency of occurrence of points with zero velocity a) and positions at the start of the cycle b).

```
    for j = 1:length(locs)
        % plot start of each cycle with different color
        plot(par(i), X(locs(j), i), '.', 'color', cmap(j, :));
    end
end
```

The diagram in Figures 3b illustrates that an improper dimensional reduction may cause that there are no data for some values of the design parameter. The following code was used in order to assemble the diagram:

```
figure; hold on;
for i = 1:length(par)
    diffY = diff(sign(Y(:, i) + 16)); % find crossings with Y = -16
    indUp = find(dotY > 0);           % crossings with positive slope
    indDown = find(dotY < 0);         % crossings with negative slope
    plot(par(i) * ones(1, length(indDown)), X(indDown, i), 'k.');
    plot(par(i) * ones(1, length(indUp)), X(indUp, i), 'y.');
end
```

The diagrams presented in Figure 3 have the following disadvantage: they hold little information about a frequency of occurrence of the depicted points. One can distinguish between areas with the lower and the higher frequency but it is difficult to determine where the frequency is the highest. The frequency of occurrence can be effectively estimated by histograms. Sample histograms are shown in Figure 4. The histogram which is depicted in Figure 4a is generated by the following code:

```
hor = cell(1, n); ver = cell(1, n);
for i = 1:length(par)
    ddotX = diff(sign(dotX(:, i))); % find zero crossings
    indUp = find(ddotX > 0);         % crossings with positive slope
    indDown = find(ddotX < 0);       % crossings with negative slope
    hor{i} = [par(i) * ones(1, length(indDown)), ...
              par(i) * ones(1, length(indUp))];
    ver{i} = [X(indDown, i)', X(indUp, i)']; % X(:,i) is column vector
end
```

```
% generate 2D histogram
h = histogram2(cell2mat(hor), cell2mat(ver), length(par), ...
               'DisplayStyle', 'tile', 'ShowEmptyBins', 'on');
colormap('pink');
```

Colour map `'pink'` offers the best contrast between areas with the low frequency and the zero frequency of occurrence, but is not suitable for printing because of large black areas. A map more suitable for printing can be generated by command `colormap(flipud('gray'))`.

The histogram depicted in Figure 4b can be generated similarly employing the code which was used for Figure 3c:

```
hor = cell(1, n); ver = cell(1, n);
for i = 1:length(par)
    [~, locs] = findpeaks(cos(Phi(:, i))); % find start of each cycle
    hor{i} = par(i) * ones(1, length(locs));
    ver{i} = X(locs, i)';                  % X(:,i) is column vector
end
% generate 2D histogram
h = histogram2(cell2mat(hor), cell2mat(ver), length(par), ...
               'DisplayStyle', 'tile', 'ShowEmptyBins', 'on');
colormap('pink');
```

5 Conclusions

This work has provided an overview of techniques which can be used in order to process output signals of non-linear systems. Methods for post-processing both single and multiple outputs were presented, shown in figures and discussed. Moreover, Matlab commands and codes were provided.

The methods were applied to time series that represent displacement and velocity of a shaft supported on non-linear journal bearings. The methods, however, can be applied also to other types of outputs: dynamic stress–strain, deformation, pressure and flow to name a few. If some changes are made, the methods are suitable also for the analysis of static quantities.

Acknowledgement

This publication was supported by project SGS-2016-038 of University of West Bohemia.

Simulations were performed in the AVL Excite software which is available in the framework of the University Partnership Program of AVL List GmbH, and whose usage is greatly acknowledged.

References

[1] TESCHL, Gerald. *Ordinary Differential Equations and Dynamical Systems*. Providence: American Mathematical Society, 2012. ISBN 978-0-8218-8328-0.

[2] SHABANA, Ahmed A. *Dynamics of Multibody Systems*. 3rd Ed. Cambridge: University Press, 2005. ISBN 978-1139446518.

[3] OFFNER, Guenter. Modelling of condensed flexible bodies considering non-linear inertia effects resulting from gross motions. *Proceedings of the Institution of Mechanical Engineers, Part K: Journal of Multi-body Dynamics*. 2011, **225**(3), 204-219.

[4] STACHOWIAK, Gwidon W. and Andrew W. BATCHELOR. *Engineering Tribology*. 4th Ed. Boston: Butterworth-Heinemann, 2014. ISBN 978-0-12-397047-3.

[5] ALTMAN, Yair M. *Accelerating MATLAB Performance: 1001 tips to speed up MATLAB programs*. Chapman and Hall, 2014. ISBN: 978-1482211290.