

# Toolbox for Simulation and Parameter Identification of Nonlinear Mechanical Systems

Kjell Ahlin

Blekinge Institute of Technology, Department of Mechanical Engineering  
SE-371 79 Karlskrona, Sweden

Anders Brandt, Thomas Lagö

Axiom EduTech AB, SE-184 97 Ljusterö, Sweden

## ABSTRACT

Analysis of nonlinear mechanical systems is becoming increasingly important in many applications in automotive, aerospace and other industries. Many experimental techniques for analysis of nonlinear systems have been developed in recent research. It is important to be able to accurately simulate nonlinear systems for various input signals in order to understand dynamic effects of the nonlinearities. A new digital filter based method with high performance has therefore been developed and implemented in a new toolbox for simulation and experimental analysis of nonlinear systems in MATLAB®. Nonlinear forced response from general linear systems with known nonlinearities can be computed and analyzed with fast algorithms in this toolbox. Nonlinear parameters can further be identified from experimental data. Of particular practical interest in this case are the methods proposed by Julius Bendat, which have been generalized and implemented for identification of many different nonlinear systems. Simulation results as well as experimental results from some different practical applications are presented. The development of the toolbox has been partly funded by the Swedish Knowledge Foundation.

## INTRODUCTION

There exist several methods to obtain the time history response of a mechanical system to an arbitrary excitation. The Duhamel integral, also known as the convolution integral, is a well-established method in the literature, its main drawback is the computational burden associated with its non-recursive nature. However, the integral can be used to derive a recursive algorithm, a digital filter, which leads to a faster solution than the naive Duhamel integral method.

The method is described in detail in [1]. The mechanical system is characterized with its modal parameters, residues and poles. The parameters may come from a Finite Element Model, a lumped system description, an analytic model or from experimental modal analysis, see figure 1. When the residues and poles are known, the filter coefficients can be calculated, using different approaches. Two types of errors will occur, an aliasing error and a bias error. As we show in [1], the errors for the different methods are completely known.

As is also described in [1] we can add non-linearities to the mechanical system. The non-linearities may be of simple zero memory type, but they may also have memory, like stick-slip friction performance.

The simulation methods constitute the backbone of the described toolbox. There are also several functions for identification and parameter extraction for nonlinear systems.

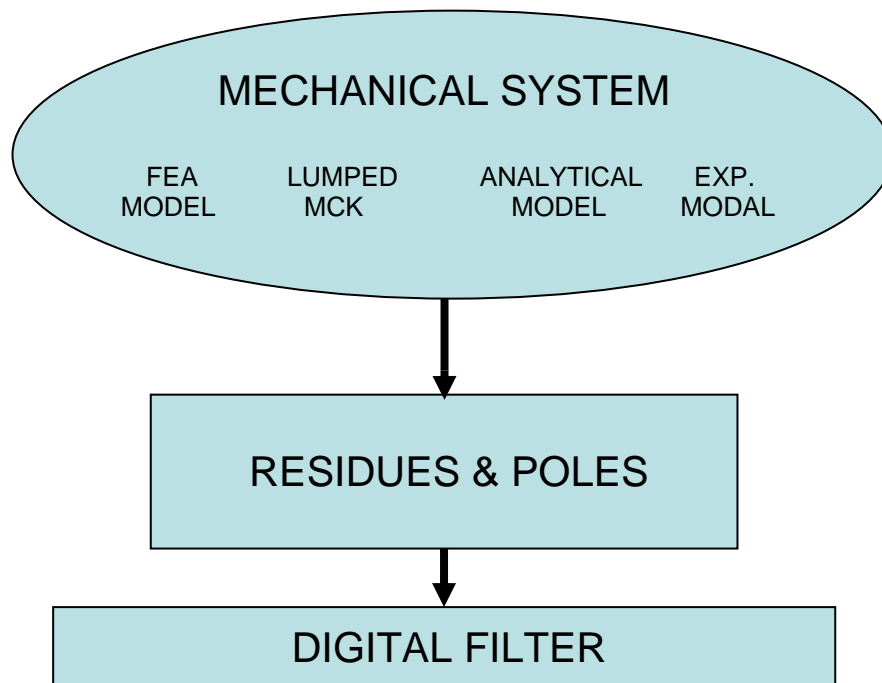


Figure 1: A linear mechanical system may be characterized by its residues and poles. These may come from different models of the system, or from experiments. The residues and poles then define the corresponding filters.

## FUNCTIONS FOR SIMULATION

The toolbox contains a family of functions to calculate the forced response of linear mechanical systems. As an example, we describe the function `timeresd`.

```
[y,t] = timeresd(F,fs,M,C,K,inno,outno,modes,type)
```

The input to the function is a force time history  $F$ , given with a sampling frequency  $fs$ . The linear mechanical system is described with its mass matrix  $M$ , damping matrix  $C$  and stiffness matrix  $K$ . The force is applied to degree of freedom, DOF,  $inno$ , and the response is calculated at output DOF  $outno$ . The input parameter  $modes$  tell the function what modes to use, and  $type$  determines the approximation method used.

$y$  is the output response as displacement and we get a corresponding time vector  $t$ .

The function has many options:

- $outno$  may be a vector, so many outputs can be calculated at the same time.
- the different types implemented are impulse invariant, step invariant, centered step invariant, ramp invariant, three point lagrange, etc.
- instead of  $M, C, K$  we may give mass and stiffness matrices and modal damping
- we may also describe the system with its residues and poles
- we may also calculate the output(s) as velocity instead of displacement

There are then many functions where non-linearities are added to the linear system. A simple example is `timeresd_cubic`.

```
x = timeresd_cubic(F,fs,M,C,K,const,inno,modes,responses)
```

The performance is quite similar to `timeresd`, but here a cubic spring is connected to ground at the DOF where the force is applied (inno). The parameter `const` tells the constant for the cubic spring. As for the linear `timeresd` function the linear mechanical system may also be described with `M`, `K` and modal damping or with residues and poles. The output response(s) may be calculated as displacement or as velocity.

The simulation functions are optimized for fast performance. To accomplish this, the more advanced functions have the description of the non-linearities explicitly written into the code. An example of such a function is `nonlindmck`

```
x = nonlindmck(F,fs,M,C,K,FDOF,nIDOFs,modes,responses)
```

In this function a force `F` is applied to DOF `FDOF` of a linear system described with `M`, `C` and `K`. We may then have any number of nonlinear functions connected to the linear system. The vector `nIDOFs` tells which DOFs are involved. In figure 2 we give an example.

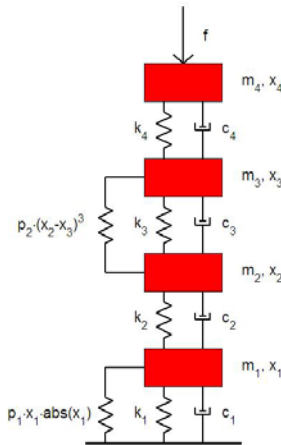


Figure 2: A four degree of freedom linear mechanical system with two nonlinear springs. A force is input to DOF 4 and all four responses are calculated.

The system in figure 2 is a four degree of freedom linear system. Between the masses  $m_2$  and  $m_3$  there is a cubic spring, and between  $m_1$  and ground there is a spring with the function “square with sign”. To use the function `nonlindmck` we type

```
x = nonlindmck(F,fs,M,C,K,4,[3,2,1],[1:4],[1:4])
```

Inside the function, we have to give the definition of the nonlinearities, in this example

```
f1 = 2.e16*[-(x1(1)-x1(2)).^3; (x1(1)-x1(2)).^3; -0.5e-4*x1(3).*abs(x1(3))];
```

To calculate all four responses to a given random force with 400 000 samples takes 65 seconds on a laptop computer.

As for the other forced response functions, there are many different options:

- the linear system can be described with `M`, `C`, `K` or with `M`, `K` and modal damping or with residues and poles.
- the output(s) can be calculated as displacement or as velocity
- approximation method may be selected (impulse, step, ramp,...)
- many examples of nonlinear functions are given, including nonlinearities with memory

In the toolbox, there are also some useful functions to create suitable input signals like swept sine and burst random.

## FUNCTIONS FOR ANALYSIS

A simple way to analyze a nonlinear system is to apply a swept sine and then calculate the response at the input frequency. The toolbox has a function to mimic a tracking filter to perform that, `trackfilt`. We can also use a function `harmbalance` to calculate the expected result under the assumption of harmonic balance. In figure 3 we show an example. The system under study is a cantilever beam connected to a cubic spring at the free end. The beam is described with its residues and poles. In the figure we compare the result from harmonic balance calculation with the result from a `trackfilt` analysis of a up-down sine sweep. The frequency region around the first resonance is shown. The performance looks like the one found in most textbooks.

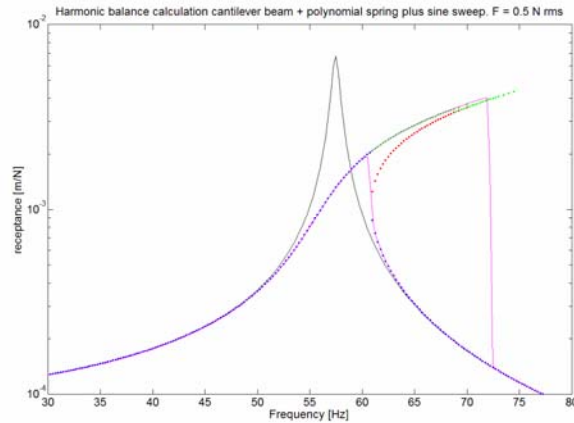


Figure 3. A cantilever beam with a cubic spring at the free end is simulated. In the figure the result from a harmonic balance calculation is shown together with the result from an up-down sine sweep.

With more complicated systems we can get a chaotic behavior, see an example in figure 4, where a waterfall diagram from a sweep is shown. The system is a four degree of freedom system with one cubic spring.

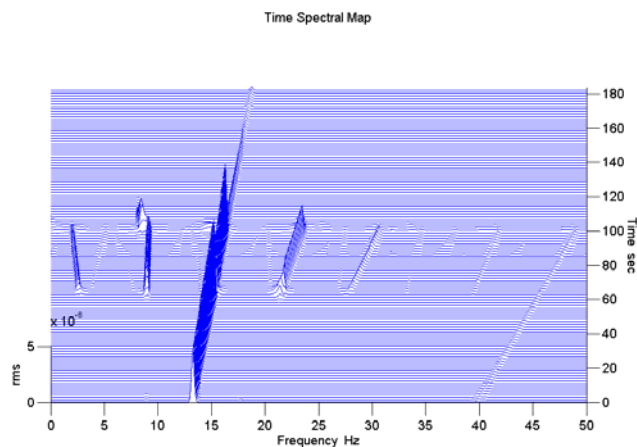


Figure 4. A 4 DOF system with a cubic spring is simulated. The system is excited with a swept sine. In some frequency ranges a chaotic behavior occurs.

One popular method to analyze nonlinear systems builds on the work of Julius Bendat [2]. Different ways to use the basic ideas are described in [3]. The common approach is that the nonlinear problem can be reformulated to a problem of multiple input / single output, MISO. To handle that, the toolbox contains efficient functions to handle MISO and MIMO, multiple input / multiple output situations.

The function *xmtrx* calculates the input and output spectral matrices and the cross spectral matrix. These matrices are then the input to the function *mimoest* that calculates the linear system Frequency Response Function, FRF, matrix and the multiple coherence function.

In the work of Bendat, the set of inputs, which may be correlated, are transformed into a set of uncorrelated inputs. That is not necessary to calculate the FRF matrix and the multiple coherence, but we have functions in the toolbox to perform the transformation for the interested user. We define a transform matrix  $\Phi$  that transforms the set of (correlated) inputs  $X$  into a set of uncorrelated inputs  $U$  by

$$U = \Phi \cdot X \tag{1}$$

The function *fmtrx* gives the  $\Phi$  matrix. That matrix may then be used to calculate the coherences used by Bendat to rank the importance of the inputs. The function for this is *ucohere*, which gives the individual coherences between the uncorrelated inputs and the output. The sum of these coherences equals the multiple coherence.

To simplify the use of Bendat's methods we have the function *julius*, which makes use of the MIMO functions above.

We give an example of the use of the functions.

A single degree of freedom system with a cubic spring is studied, see figure 5.

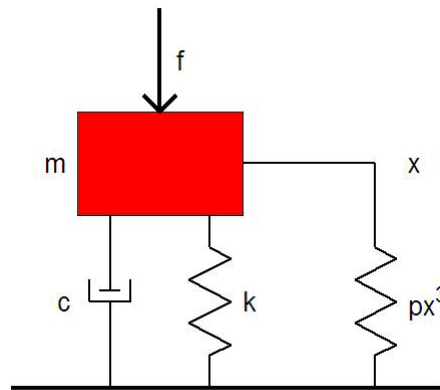


Figure 5. A single degree of freedom with a cubic spring is simulated. The system is excited with a random force.

The system is excited with a random force and the response is simulated with *timeresd\_cubic*. The raw Frequency Response Function (best linear approximation) between force and response is calculated. When that FRF is compared with the theoretical linear part of the system, we find a great difference, figure 6. We overestimate the resonance frequency of the linear system. The coherence, figure 7, clearly shows that the system is nonlinear.

We now use *julius* with a guess that the nonlinearity is of cubic type. The problem we solve is a MISO problem with  $x$  and  $x^3$  as inputs and the force  $f$  as output. This approach is often called "reverse path", as we calculate the real input  $f$  as output. As result, we get the inverse of the sought linear FRF and the unknown coefficient  $p$ . In figure 6 we find that we get an almost perfect estimate of the FRF in this way. As a quality measure, we plot the calculated multiple coherence in figure 8. It is very close to one, indicating that we have a good model.

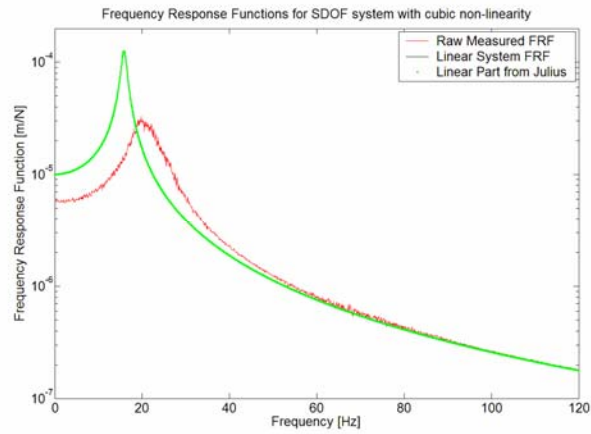


Figure 6. Frequency Response Functions for a single degree of freedom system with a cubic spring. The raw FRF differs from the linear FRF, but the FRF from Julius is nearly perfect.

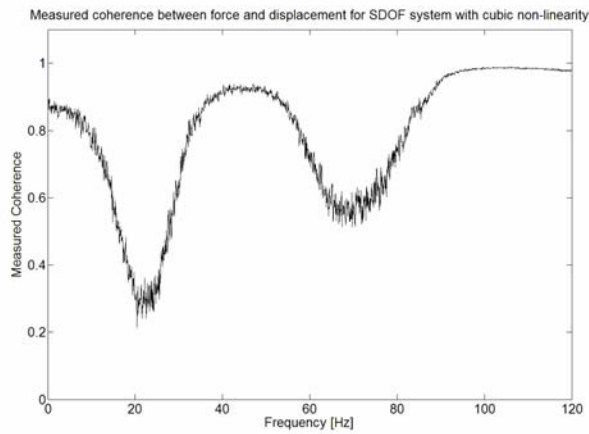


Figure 7. Coherence function corresponding to the raw FRF in figure 6. The coherence clearly shows that the system is nonlinear.

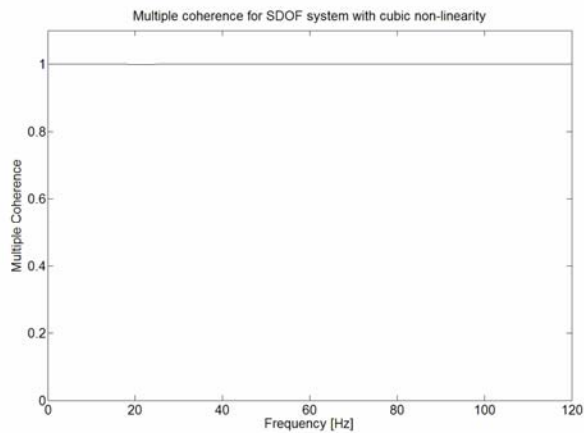


Figure 8. Multiple coherence function corresponding to the result in figure 6. The multiple coherence clearly shows that the used model is good.

To see what happens with a bad guess, we try *julius* with square with a sign ( $x \cdot \text{abs}(x)$ ) as the nonlinearity instead of the cubic function used before. The estimate of the FRF is then quite bad, figure 9.

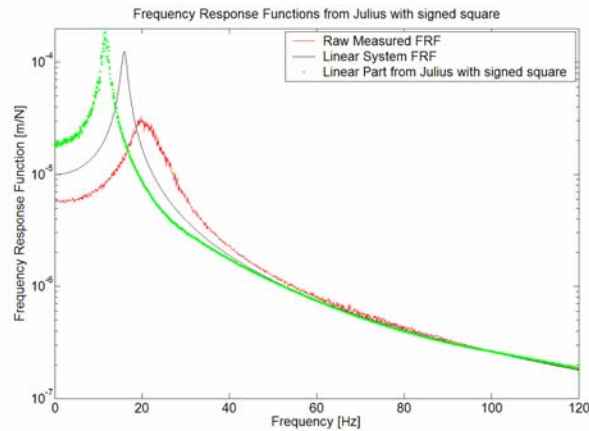


Figure 9. The FRF calculated using a bad guess for the nonlinearity is as bad as the raw FRF.

With the bad guess, we now underestimate the resonance frequency of the linear system. The multiple coherence, however, gives us a warning, see figure 10. This simple example shows that the estimates must be used with care in a real situation, when the measured data may be contaminated. It also shows the benefit of using the toolbox for simulations. We argue that you must try a proposed new method on simulations first before you try it on real data. It is only in simulations we have the middle peak in figure 9! In a real situation, you may be satisfied with the multiple coherence in figure 10. The example shows that this is not good enough!

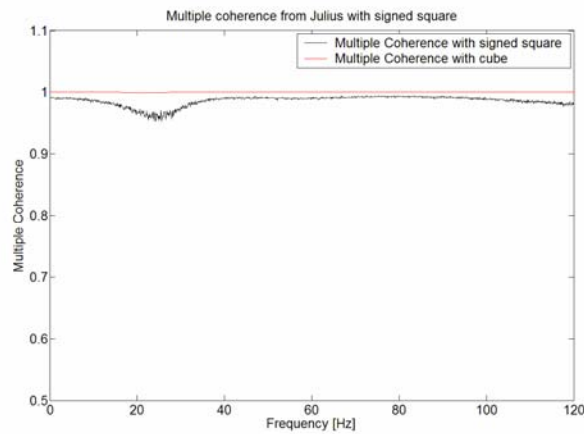


Figure 10. The multiple coherence with a bad guess for the nonlinearity gives a warning.

## **SUMMARY**

A MATLAB toolbox for simulation and identification of nonlinear mechanical system has been developed. The simulation functions are fast and accurate. The linear part of the system can be defined in many ways and many different nonlinearities, with or without memory, can be handled. Efficient tools to handle multiple input / multiple output (MIMO) situations are included to be used when a nonlinear identification problem is converted into an equivalent MIMO problem.

## **REFERENCES**

- [1] Ahlin, Magnevall, Josefsson: Simulation of Forced Response in Linear and Nonlinear Mechanical Systems Using Digital Filters. ISMA 2006, September 2006, Leuven, Belgium.
- [2] Julius S. Bendat: Nonlinear Systems Techniques and Applications. John Wiley & Sons, Inc 1998
- [3] Josefsson, Magnevall, Ahlin: On Nonlinear Parameter Estimation with Random Noise Signals IMAC XXV, Orlando, February 2007