Frequency-Domain Analysis Methods for Modelica Models

Andreas Abel Tobias Nähring

ITI GmbH
Webergasse 1
01067 Dresden, Germany

{andreas.abel,tobias.naehring}@iti.de

Abstract

In addition to time-domain simulation methods, engineers from different application fields require further types of analysis to be performed on their systems. In particular results from frequency domain analysis play an important role – this includes the calculation of natural frequencies and vibration modes, but also the computation of transfer functions or the simulation of steady-state behaviour.

If the system equations are formulated using the Modelica language, there is the potential to use one and the same model for time-domain as well as frequency-domain computations.

In this paper we will show, how the different methods can be applied to a Modelica model, what kind of prerequisites and adjustments are required in order to perform the different types of analysis and how these methods can be seamlessly integrated into a Modelica simulation environment.

Keywords: Modelica, Steady State Simulation, Transfer Function Analysis, Natural Frequency Analysis

1 Introduction

In many engineering disciplines frequency-domain methods play an important role. Powertrain engineers for instance not only exploit transient simulations, but to a large extend assess the behaviour of their systems based on the natural frequencies, the resulting vibration models, and also in terms of steady state results, which show vibrations under stationary conditions resulting from the uneven and multi-order excitation of the driveline by the engine. Other engineering domains and tasks also require frequency-domain approaches.

However, all these tasks would typically be assigned to different software tools, which is not really necessary.

Modelica forms the ideal base also for frequency-domain analyses, since it provides complete system descriptions in an analytic form. However, so far Modelica is used almost exclusively for transient time-domain simulation.

In this paper we will show, how Modelica models are used in order to compute frequency-domain results and how these processes are integrated into the Modelica simulation environment SimulationX.

The paper will treat the following topics:

- Nonlinear periodic steady-state simulation and generation of spectral results based on harmonic balance
- Natural frequencies, vibration modes and energy distributions based on models linearized in an operating point
- Computation of transfer functions based on models linearized in an operating point

We focus on the periodic steady-state simulation since this is the most recent innovation in SimulationX.

2 Periodic Steady-State Simulation

2.1 Application to Modelica Models

The main area of application for the nonlinear periodic steady-state simulation in SimulationX is the vibration analysis of powertrains.

The example Modelica model in Fig. 1 is an adaption from [4] p. 246 with some added damping and cylinders including oscillating masses and driven by some typical combustion engine cylinder pressure.

The steady-state for a range of mean rotational speeds of the engine has to be computed. The oscillation time period is determined by the engine speed and the periodicity of the excitation over the crank angle.

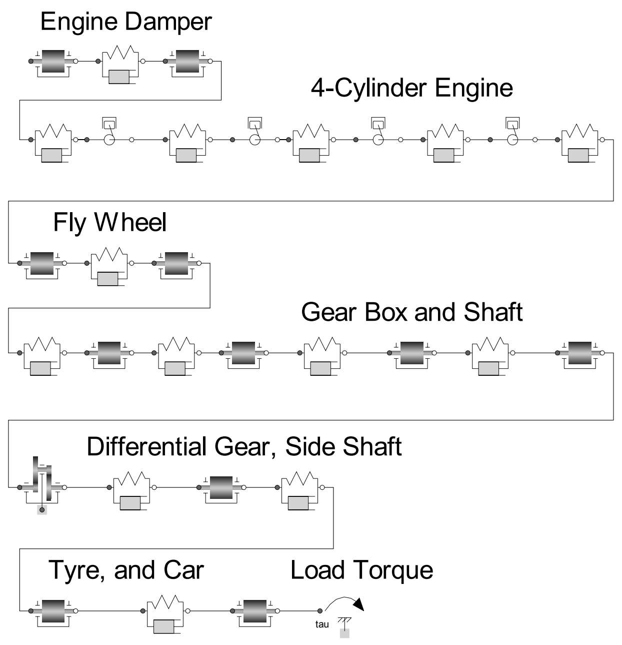


Fig. 1: Example of a Modelica model of a powertrain analyzed with the periodic steady-state simulation

But the method is also applicable to driven systems in other physical domains. For non-linear electronic amplifiers and filters most often the frequency or amplitude of the driving generator is swept and the period is measured at its phase. Therefore, a general approach is needed. In SimulationX the following procedure has been implemented: The user chooses the varying reference quantity (e.g. mean engine speed or generator frequency) and the period variable (e.g. crank angle or generator phase) from Modelica model trees containing all variables and parameters. For powertrains (or more general whenever the reference quantity is not a parameter but the mean value of a variable) the user also distinguishes some model parameter as compensation parameter - such as the load torque of the powertrain. The algorithm adjusts the compensation parameter for the steady-state, i.e. the mean engine torque and the load torque are balanced out by the algorithm. No special preparation of the Modelica model is needed to enable the steady-state simulation. The same model may be used for a simulation in time-domain too.

During the simulation the computed spectra of the Modelica variables are written to special steady-state protocols. Those results can be visualized in several different representations (amplitudes, phases, fluctuations, spectral powers and so on). For the powertrain example from Fig. 1 some of the amplitudes of the calculated harmonic torque components in the mass-damper spring are shown in Fig. 2.

In SimulationX the initial conditions corresponding to the results of the periodic steady-state simulation can be calculated and used to initialize a successive transient simulation.

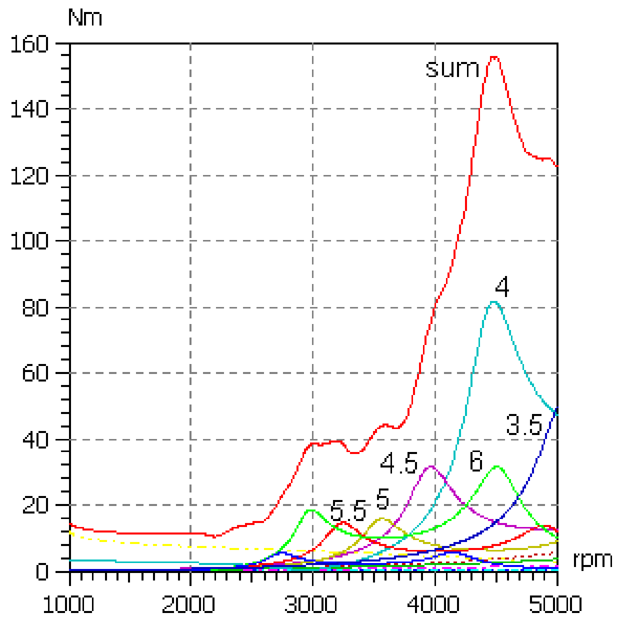


Fig. 2: Spectral results for the torque of the spring in the engine damper; the sum curve and the amplitudes of the first harmonic components are shown, the largest amplitudes are labelled with the oscillation orders

In this way the periodic solution can be recalculated with a transient simulation and the steady-state results can easily be checked. Fig. 3 shows a very good match of the steady-state simulation result with the transient simulation result.

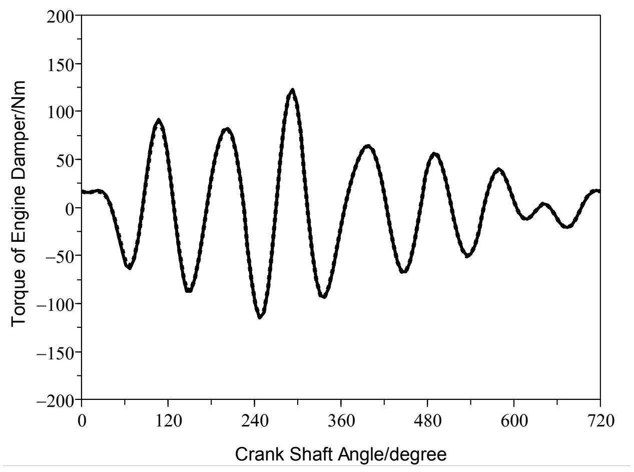


Fig. 3: Torque in the spring of the engine damper (full line: steady-state simulation, dashed line: transient simulation); the results are almost identical

2.2 Computational Background

In this subsection we give some insight in the computational background specific to the periodic steady-state simulation. If the reader is only interested in applications he may safely skip to subsection 2.3.

For the periodic steady-state simulation the harmonic balance method is employed. This method gives a high spectral precision of the results and prepares the numerical base for behavioural modelling in the frequency domain.

2.2.1 System Equations

The symbolic analysis compiles from the Modelica model a system of equations for the stationary simulation. If the simulation time appears explicitly in the model equations (for instance in a driven system) it is replaced by a state x_{time} with $dx_{\mathrm{time}} / dt = 1$ which leaves us with an autonomous algebraic differential equation system

$$f\left(x(t), \dot{x}(t), x_{\mathrm{C}}\right) = 0 \tag{1}$$

where x is the $\mathbf{R}^n$-valued state vector with corresponding time-derivative $\dot{x}$, and $x_{\mathrm{C}} \in \mathbf{R}$ is the compensation parameter (see section 2.1). It is convenient to represent oscillations not over time but over the phase angle $\varphi := \omega t$ for which the period length keeps constant at 2π independent of the period duration (ω is the phase velocity of the oscillation). Substituting the derivative w.r.t. time through the derivative w.r.t. phase $\dot{x}(t) = \omega\, x'(\varphi)$ in eq. (1) gives

$$f\left(x(\varphi), \omega\, x'(\varphi), x_{\mathrm{C}}\right) = 0\,. \tag{2}$$

Throughout the remainder of this section we represent x in dependence of the phase angle.

The system is assumed to be freely displaceable in one direction of the state space. Therefore, we chose a combination of a 2π-periodic function $\tilde{x}$ and a component linearly dependent on the phase angle as a solution ansatz

$$x(\varphi) = \frac{x_{\mathrm{P}}\varphi}{2\pi} + \tilde{x}(\varphi) \tag{3}$$

for the system equation (2) with a constant vector $x_{\mathrm{P}} \in \mathbf{R}^n$, called period vector in the sequel.

This setup is rather general. It includes freely rotating powertrains and periodically driven systems.

Solving (2) can now be divided into the two tasks

- computation of the period vector x_{P}
- computation of the periodic function $\tilde{x}$

which will be described in the following two sections.

2.2.2 Period Vector Computation

The user selects one model variable as the period variable (cf. section 2.1). We denote the index of that variable as $i\mathrm{P}$. For this variable the user specifies the period length p. The model equations (2) are then solved for the static case (i.e. $\omega = 0$) once with $\varphi = 0$ and once with $\varphi = 2\pi$. Because of the 2π-periodicity of $\tilde{x}$ the difference of these two solutions just gives the period vector

$$x_{\mathrm{P}} = x(2\pi) - x(0). \tag{4}$$

At $\varphi = 0$ the displacement of the system (e.g. the rotational position of a powertrain) is determined by the additional condition $x_{i\mathrm{P}}(0) = 0$. This together with (2) and (3) results in the overall system

$$f\left(x(0), 0, x_{\mathrm{C}}\right) = 0;\; x_{i\mathrm{P}}(0) = 0 \tag{5}$$

for the case $\varphi = 0$ which consists of $n+1$ equations for the $n+1$ unknowns composed of the n states $x(0)$ and the compensation quantity x_{C} (e.g. the load torque of a powertrain).

For $\varphi = 2\pi$ we use the user-defined periodicity of the state vector component $i\mathrm{P}$ and solve

$$f\left(x(2\pi), 0, x_{\mathrm{C}}\right) = 0;\; x_{i\mathrm{P}}(2\pi) = p\,. \tag{6}$$

The condition that x_{C} is the same in (5) and (6) offers a possibility to check the computed solutions.

For driven systems the equations in (5), (6) may not be simultaneously solvable. In that case in each of these systems the static equation

$$f\left(x, 0, x_{\mathrm{C}}\right) = 0;$$

is replaced by the condition

$$f\left(x, v, x_{\mathrm{C}}\right) = 0;\quad \|v\|_2 \to \min$$

where $\|v\|_2$ denotes the Euclidian norm of v.

In practice it has proven sufficient to solve the resulting restricted minimization problems by a modified Gauss-Newton algorithm.

2.2.3 Harmonic Balance

For the computation of the periodical part $\tilde{x}$ in the ansatz (3) equation (2) is reformulated as the variational equation

$$\frac{1}{2\pi}\int_0^{2\pi}\psi(\varphi)\cdot y(\varphi)\,d\varphi = 0 \qquad \textbf{(7)}$$

with ψ varying over all continuous $\mathbf{R}^n$-valued functions fulfilling the condition $\psi(0)=\psi(2\pi)$ and with

$$y(\varphi) := f\left(\frac{x_\mathrm{P}\varphi}{2\pi}+\tilde{x}(\varphi), \omega\left(\frac{x_\mathrm{P}}{2\pi}+\tilde{x}'(\varphi)\right), x_\mathrm{C}\right). \quad \textbf{(8)}$$

For a fixed phase velocity ω the solution $\tilde{x}$ is only determined up to a multiple of x_P and a corresponding phase shift (e.g. for a powertrain the arbitrary initial angular position). To formally fix the initial disposition, additionally the mean value of the period variable is balanced to zero:

$$\frac{1}{2\pi}\int_0^{2\pi}\tilde{x}_{i\mathrm{P}}(\varphi)\,d\varphi = 0\,. \qquad \textbf{(9)}$$

In some cases the user does not want to prescribe the phase velocity ω directly (e.g. for powertrains it is usual to prescribe the mean rotational speed of the engine instead). For that reason the user-chosen reference quantity was introduced in section 2.1. Let $i\mathrm{R}$ be the index of the reference quantity and r the wanted mean value for that variable. Then instead of a direct assignment to ω the equation

$$\frac{1}{2\pi}\int_0^{2\pi}\tilde{x}_{i\mathrm{R}}(\varphi)\,d\varphi = r \qquad \textbf{(10)}$$

is added to the variational system.

Following Galerkin for the numerical treatment of (**7**,**8**) the function space for ψ and $\tilde{x}$ is restricted to the finite-dimensional space spanned by the harmonic orthogonal system of base functions

$$\psi[k] := \exp(\mathrm{j}k\varphi) \text{ with } k=-N,\ldots,N\,. \qquad \textbf{(11)}$$

In the following we keep using lower indexes for the state vector components but we use Modelica index notation to organize the frequency components (as we have already done so by defining $\psi[k]$ above). Using the base (11) for the periodical part $\tilde{x}$ in (3) the ansatz becomes

$$\tilde{x}(\varphi) = \sum_{k=-N}^{N}\exp(\mathrm{j}k\varphi)\,\hat{x}[k] \qquad \textbf{(12)}$$

where $\hat{x}[k]$ is the k-th frequency component of the state space vector (we use a hat $\hat{x}$ or $(x)^\wedge$ to denote complex amplitudes). Since $\tilde{x}$ is real $\hat{x}[k]$ is the complex conjugate of $\hat{x}[-k]$. Thus, the values of $\hat{x}$ are determined by $n(2N+1)$ real numbers. With ψ replaced by $\psi[k]$ for $k=-N,\ldots,N$ the resulting $2N+1$ left-hand sides of (**7**) become the first $2N+1$ Fourier coefficients $\hat{f}(\hat{x},\omega,x_\mathrm{c})[k]$ of the left-hand side of (2), i.e. Fourier coefficients of the time-domain residuals. Equations (**7**,**8**,9,10) together then give the harmonic balance equation system

$$\begin{aligned}&\hat{f}(\hat{x},\omega,x_\mathrm{C}) = 0\\ &\qquad \hat{x}_{i\mathrm{P}}[0] = 0\\ &\qquad \hat{x}_{i\mathrm{R}}[0] = r\end{aligned} \qquad \textbf{(13)}$$

of $n(2N+1)+2$ scalar equations for the $n(2N+1)$ unknowns in $\hat{x}$ and the additional two unknowns ω, x_C. The fast Fourier transformation (FFT) is used to approximate the Fourier-coefficients of y. Because of the nonlinearities in f the spectrum of y is wider than that one of x and some oversampling is needed for the FFT to keep the aliasing error low.

For solving system (13) Newton's algorithm is applied. Deriving the Newton corrector equation in time-domain and then transforming it into frequency-domain gives good insight into the structure of the resulting system of equations. A first order Taylor approximation of (2) in the current numerical approximation of $(\tilde{x},\omega,x_\mathrm{C})$ yields the equation

$$\begin{aligned}f + \partial_1 f\cdot\delta x + \partial_2 f\cdot(\omega\delta x' + x'\,\delta\omega) + \\ + \partial_3 f\cdot\delta x_\mathrm{C} = 0\end{aligned} \qquad \textbf{(14)}$$

which determines with (3) the Newton correction $(\delta\tilde{x},\delta\omega,\delta x_\mathrm{C})$ (note: (i) here $\partial_k f$ stands for the derivative of f w.r.t. the kth argument, and (ii) for clarity we have omitted the arguments $(x,\omega x',x_\mathrm{C})$ of f, (iii) $x,\delta x,\delta\tilde{x}$ are functions of φ). The time-domain products in (**14**) correspond to frequency-domain convolutions. E.g., the FFT transforms $\partial_1 f\cdot\delta x$ into

$$\left((\partial_1 f)^\wedge * \delta\,\hat{x}\right)[k] = \sum_{l=-N}^{N}(\partial_1 f)^\wedge[k-l]\,\delta\,\hat{x}[l]\,. \qquad \textbf{(15)}$$

With $I:(I\,\hat{x})[k] := k\hat{x}[k]$ the spectrum of the derivative $\dot{x}$ can be written as $(x')^\wedge = \mathrm{j}I\,\hat{x}$. So, after shifting f to the right-hand side (**14**) is transformed by the FFT into the equation

$$\begin{aligned}\left((\partial_1 f)^\wedge * \delta\,\hat{x}\right) + \mathrm{j}\,\omega\left((\partial_2 f)^\wedge * (I\delta\,\hat{x})\right) + \\ + (\partial_2 f\cdot x')^\wedge\,\delta\omega + (\partial_3 f)^\wedge\,\delta\,x_\mathrm{C} = -\hat{f}\end{aligned} \qquad \textbf{(16)}$$

for the unknown Newton correction $(\delta\hat{x}, \delta\omega, \delta x_{\mathsf{C}})$ in the frequency domain. Together with (9) and (10) written as

$$\delta\,\hat{x}_{i\mathsf{P}}[0] = 0; \quad \delta\,\hat{x}_{i\mathsf{R}}[0] = 0 \qquad \textbf{(17)}$$

this system formally determines the Newton correction in the frequency domain completely.

With the number of $n(2N+1)+2$ real unknowns the system is rather large and the convolution operator in (16) causes large fill-in of the system matrix making direct solving infeasible in real-world applications. Therefore, the iterative GMRES algorithm is used instead (see e.g. [5]). This method only requires the evaluation of the left-hand side of (16) for known $(\delta\hat{x}, \delta\omega, \delta x_{\mathsf{C}})$. This also makes it possible to replace the frequency-domain convolutions in (16) by the cheaper corresponding time-domain products in (**14**) (together with the therefore needed FFT-operations). GMRES only works well with an appropriate pre-conditioner. Thus, one must be able to roughly solve systems with the left-hand side of (16) fast. For this end the block-diagonal preconditioner is used (see e.g. [6]). This approximates the convolutions by only retaining the mean value component of $(\partial_k f)^\wedge$:

$$\begin{aligned}\left((\partial_1 f)^\wedge * \delta\,\hat{x}\right)[k] &\approx (\partial_1 f)^\wedge[0]\cdot\delta\hat{x}[k]\\ \left((\partial_2 f)^\wedge * (I\delta\,\hat{x})\right)[k] &\approx (\partial_2 f)^\wedge[0]\cdot k\delta\hat{x}[k]\end{aligned} \qquad \textbf{(18)}$$

The so approximated system (16) can be solved frequency-component wise.

If the dynamical system is linear then the Jacobians $\partial_1 f, \partial_2 f$ are constant in time and the corresponding higher spectral components in the convolutions (e.g. $(\partial_1 f)^\wedge[k-l]$ with $k-l\neq 0$ in (15)) are zero. In this case `$\approx$' in (18) can be replaced by `=' and the approximations are exact. For increasing nonlinearities the higher spectral components of $\partial_1 f, \partial_2 f$ omitted in the preconditioner gain influence, the approximations become more coarse. In general one can say that with stronger nonlinearities the number of GMRES iterations per Newton step and the number of Newton-iterations increase.

If the local Newton method does not converge fast enough then the Newton-algorithm with backward-error minimization via backtracking (see [1] and [7]) is applied. For a better numerical condition the states are automatically scaled during the computation.

In section 2.3 we will give an example of a nonlinear system with a turning point in its frequency response. To make the computation of such points possible a curve tracing algorithm with variable step-size is implemented in SimulationX. A short outline of this algorithm shall conclude this subsection.

Only at the starting value r_{Start} and the end value r_{Stop} of the interval for the reference quantity $x_{i\mathsf{R}}$ the full system (13) is solved. At intermediate points for $x_{i\mathsf{R}}$ the last equation determining the value of the reference quantity is removed resulting in

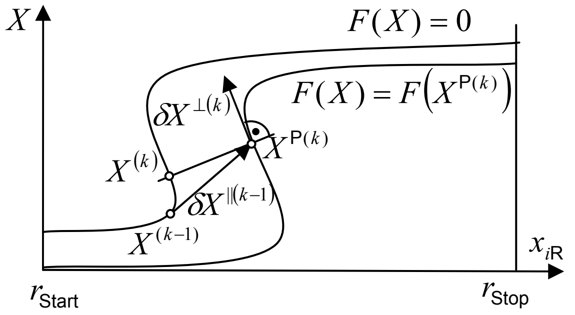


Fig. 4: Curve tracing algorithm (see text for details)

$$F(X)=0 \text{ with } F(X):=\begin{pmatrix}\hat{f}(\hat{x},\omega,x_{\mathsf{C}})\\ \hat{x}_{i\mathsf{P}}[0]\end{pmatrix} \qquad \textbf{(19)}$$

and with the unknowns collected in $X := (\hat{x}, \omega, x_{\mathsf{C}})$.

Since (**19**) has one scalar equation less than unknowns it formally defines a solution curve (see also upper branch in Fig. 4) instead of a single point.

Given the last solution point $X^{(k-1)}$ on the solution curve and the tangent direction $\delta X^{\|(k-1)}$ of the solution curve in that point a prediction

$$X^{\mathsf{P}(k)} = X^{(k)} + s\,\delta X^{\|(k-1)}$$

for the new solution point is computed. Thereby, the step size s is chosen in dependence of the estimated curvature of the solution path, the estimated distance of $X^{\mathsf{P}(k)}$ to the solution path, and the local convergence behaviour of Newton's algorithm (for details see [2]). In the predicted point a new estimation $\delta X^{\perp(k)}$ for the tangent vector is computed as the solution of the system

$$\begin{aligned}&\mathrm{D}F\left(X^{\mathsf{P}(k)}\right)\delta X^{\perp(k)} = 0,\\ &\left(\delta X^{\|(k-1)}\right)^T\cdot\delta X^{\perp(k)} = 1.\end{aligned}$$

This is not the tangent direction to the solution curve but to the curve defined by $F(X)=F\left(X^{\mathsf{P}(k)}\right)$ (see Fig. 4). Nevertheless, these curves and their tangents are supposed to be close to each other. The Newton correction for the computation of the next solution $X^{(k)}$ of (**19**) is then carried out in the affine plane with $X^{\mathsf{P}(k)}$ as origin and $\delta X^{\perp(k)}$ as normal direction. The point $X^{(k,0)} := X^{\mathsf{P}(k)}$ is used as an initial guess and the Newton corrections $\delta X^{(k,i)}$ as well as the iterated solution approximations $X^{(k,i)}$ $(i = 0,1,\dots)$ are defined by the system

$$\begin{aligned} DF\left(X^{(k,i)}\right)\cdot \delta X^{(k,i)} &= -F\left(X^{(k,i)}\right),\\ \left(\delta X^{\perp(k)}\right)^T \cdot \delta X^{(k,i)} &= 0,\\ X^{(k,i+1)} &= X^{(k,i)} + \delta X^{(k,i)}. \end{aligned} \tag{20}$$

As Fig. 4 suggests $X^{\perp(k)}$ is a better approximation of the tangent to the solution curve at the new solution point $X^{(k)}$ than $X^{\|(k-1)}$. Using $X^{\perp(k)}$ lets the Newton iterations run on nearly the shortest path to the solution curve, gives (20) a better numerical condition, and avoids jumping between different solution branches at sharp turning points of the solution path.

2.3 Example: Nonlinear Spring-Mass-System with Turning-Point in Frequency Response

Unlike linear systems nonlinear systems may exhibit turning points in the frequency characteristic. The curve tracing algorithm implemented in SimulationX makes the computation of such kind of frequency characteristics possible.

The simple mechanical system of Fig. 5 is a torque excited spring-mass-oscillator. The frequency of the sinusoidal torque source is chosen as the reference quantity and swept between $0.2\,\mathrm{Hz}$ and $0.7\,\mathrm{Hz}$. Since this reference quantity is a parameter and not a variable SimulationX chooses it automatically as compensation parameter as well. The phase of the sine oscillator is the period variable with period 2π. The quadratic term added to the spring characteristic makes the system nonlinear in such a way that it shows a turning point in the frequency characteristic (see Fig. 6).

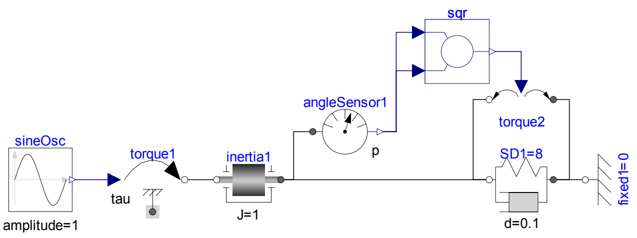


Fig. 5: Nonlinear Spring-Mass-system

In the interval from $0.397\,\mathrm{Hz}$ to $0.426\,\mathrm{Hz}$ the frequency characteristic is multi-valued. That corresponds to multiple periodic limit cycles at those excitation frequencies.

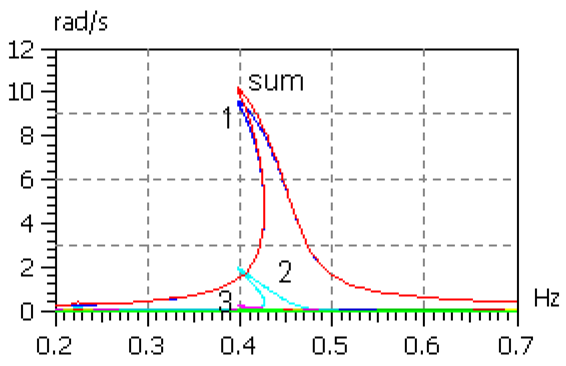


Fig. 6: Frequency response with turning-point for the angular speed of inertia1 in the nonlinear spring-mass-system; the sum curve and the first three harmonic components are distinguishable in this diagram

As an example in Fig. 7 the limit cycles from the two stable branches (lowest and highest) of the frequency characteristic at $0.405\,\mathrm{Hz}$ are shown.

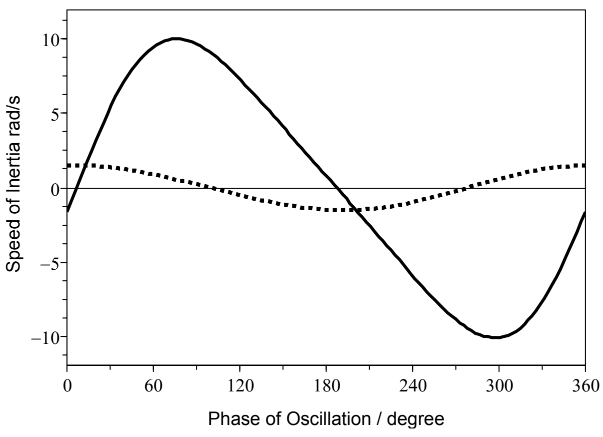


Fig. 7: Angular speed curves for the two possible stable limit cycles of the nonlinear spring-mass-system at excitation frequency $0.405\,\mathrm{Hz}$ represented over phase.

We kept this example simple to demonstrate that even very basic nonlinear systems may have frequency responses with turning-points. More complicated examples can be found in [8], and [9].

2.4 Example: Active Electronic Filter

The periodic steady state simulation is not restricted to mechanical systems. As an example the periodic steady state simulation is applied to a Modelica model for an active electronic pass-band filter (see Fig. 8). The reference and compensation quantity in this example is the frequency of the sinusoidal source `vin` and its phase is the phase variable.

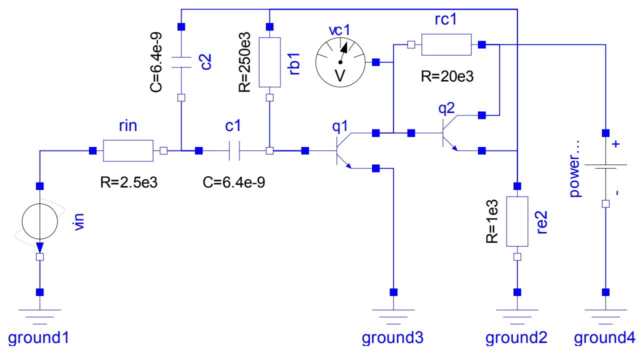


Fig. 8: Modelica model of the active electronical filter

At resonance frequency the transistor amplifier of the pass-band filter is overdriven which causes nonlinear harmonic distortions. The nonlinear frequency response of the collector voltage of transistor `q1` is shown in Fig. 9.

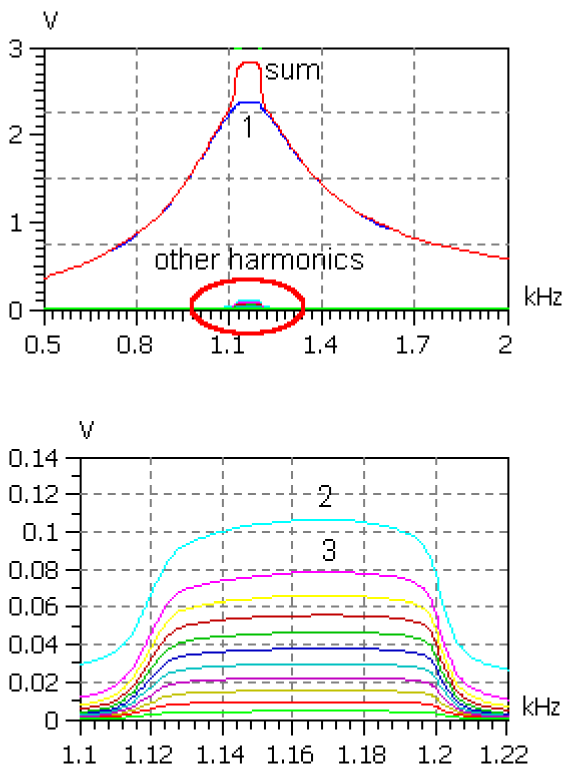


Fig. 9: Frequency response of the collector voltage of q1 in the active electronic filter; top: sum signal and first harmonic, bottom: zoomed view of the other harmonics in the resonance region where the amplifier is overdriven; the harmonics are decreasing with order, only the 2nd and 3rd harmonic are labelled

In Fig. 10 the periodic steady state result and the time domain result of this voltage over phase angle for an excitation frequency of 1.15 kHz are compared. At about 75° the base-emitter diode of q2 blocks and the voltage amplification of q1 grows which causes the spike in the collector voltage of q1.

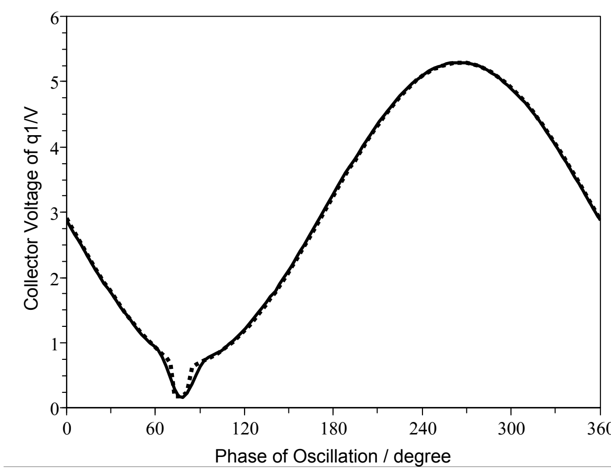


Fig. 10: Collector voltage of q1 in the active electronic filter at excitation frequency 1.15 kHz **represented over phase; full line: periodic steady state simulation, dashed line: transient simulation;**

The results are in good accordance. Nevertheless, a slight difference of the results from the periodic steady state simulation and the transient simulation is visible at about 75°. The steep slopes of the spike are somewhat smoothened by the limited number of equidistant sample-points for the steady state simulation (256 sample points per period were used).

3 Transfer Function Analysis and Natural Frequencies

3.1 Linear System Analysis

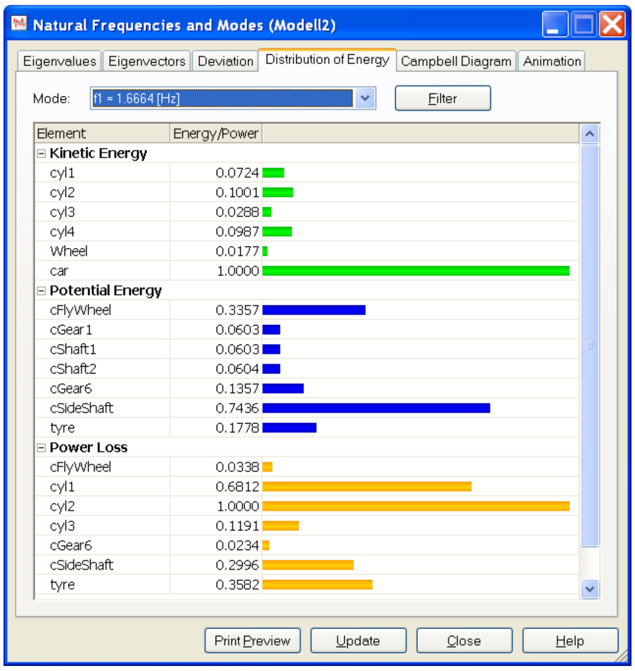
Beside the nonlinear algorithm for the steady-state simulation also linear frequency-domain analysis methods are applicable to Modelica models and are implemented in SimulationX. Those are based on the linear system which results from the linearization of the nonlinear system equations for the Modelica-model in the current operating point. The operating point may be determined by a previous transient simulation or an equilibrium computation (in electronics also called DC-analysis). Some of the algorithms may be applied to any Modelica model without changes by the user. This includes the computation of the eigensystems, the Campell diagram, and methods for the animation of the eigenmodes.

Other frequency-domain results such as the deviations in mechanical quantities (vibration modes) and the distribution of vibration energies and losses require special internal blocks that can be included into the Modelica-model. The following Modelica source code shows how the inertia from the standard Modelica library can be supplemented with an internal energy calculation block which SimulationX uses in order to compute the energy distribution.

```
model RotInertiaEnergyBlock
 import M=Modelica.Mechanics;
 extends M.Rotational.Inertia;
 Mechanics.Rotation.CalcEnergyBlock eb;
 equation
   eb.dom = w;
   eb.T = J*a;
end RotInertiaEnergyBlock;
```

The modification of the Type `SpringDamper` is similar. For a demonstration the (rotational and translational) masses and spring-dampers in the powertrain from Fig. 1 have been substituted by the modified types. The distribution of energy calculated by SimulationX for the eigenmode at 1.6664 Hz is shown in Fig. 12. In practical applications such representations show the engineer which masses, springs, and dampers dominate the behaviour in certain eigenmodes of the system, so he can take systematic countermeasures to avoid unwanted oscillations.

Up to now these blocks are not documented and only used for the internal element libraries of SimulationX. But this may change in future.



Natural Frequencies and Modes (Modell2)

Mode: f1 = 1.6664 [Hz]

Element	Energy/Power
Kinetic Energy	
cyl1	0.0724
cyl2	0.1001
cyl3	0.0288
cyl4	0.0987
Wheel	0.0177
car	1.0000
Potential Energy	
cFlyWheel	0.3357
cGear1	0.0603
cShaft1	0.0603
cShaft2	0.0604
cGear6	0.1357
cSideShaft	0.7436
tyre	0.1778
Power Loss	
cFlyWheel	0.0338
cyl1	0.6812
cyl2	1.0000
cyl3	0.1191
cGear6	0.0234
cSideShaft	0.2996
tyre	0.3582

Fig. 11: Distribution of energy for the powertrain example from Fig. 1

3.2 Input-Output Analysis

For the analysis of the input-output-behaviour the user must select the input and the output of the linearized system. Any result variable of the model may be used as the system output. SimulationX has a special class of signal inputs that may be open even for the top-level model. Those inputs may be used for the input-output-analysis. In Fig. 11 a cut-out of the powertrain from Fig. 1 is shown where a torque source with such an input has been added. The input-output behaviour is described by the frequency response function and the pole-zero diagram of the system.

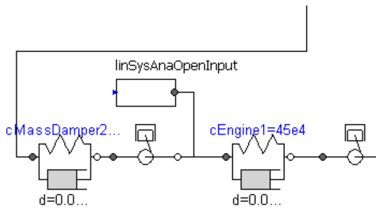


Fig. 12: Element linSysAnaOpenInput in the example from Fig. 1 with open input for the input-output-analysis

Fig. 13 and Fig. 14 show the pole-zero plot and the frequency characteristic, resp., for the powertrain from Fig. 1 with the torque at the first cylinder as input (Fig. 12) and the torque in the engine damper as output.

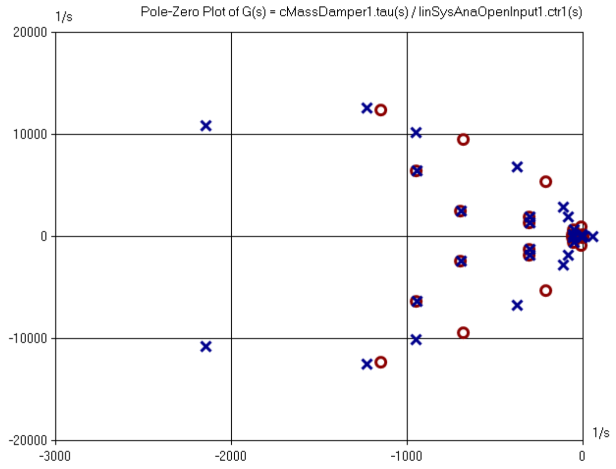


Fig. 13: Pole-zero plot of the system in Fig. 1; crosses: poles, circles: zeros

For further analysis in external tools the linearized system matrices may be exported in Modelica or MATLAB syntax.

4 Conclusions and Outlook

Periodic steady state simulation proves useful for the vibration analysis of nonlinear systems. SimulationX allows its application to Modelica models, in particular to powertrains, without the decomposition into nonlinear exciter and linear drivetrain. Furthermore, the method is applicable to driven systems of other physical domains since it is purely equation-based. Only very little knowledge of the system is required from the user. Two mechanical examples and one from electronics were given in the paper.

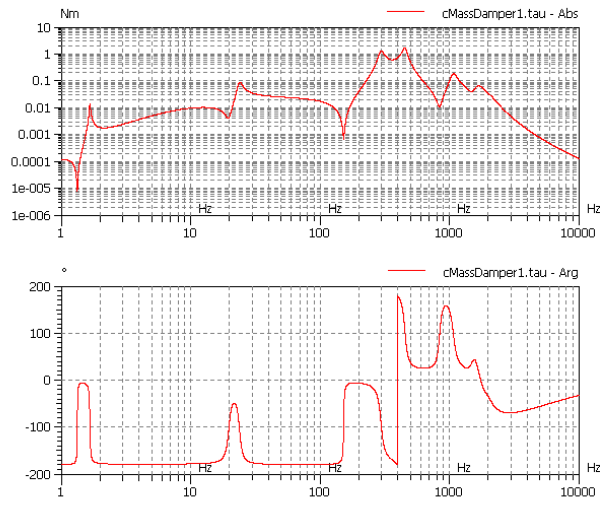


Fig. 14: Frequency response of the system in Fig. 1; top: amplitude, bottom: phase

Furthermore, we discussed methods for the small-signal analysis in the current operating point (resulting from a transient or equilibrium computation). Beside pole-zero plots and frequency response functions also some remarks about the deviation- and energy distribution analysis for oscillation modes were given. They are especially useful for the mechanical engineer to detect the powertrain elements which participate in selected oscillation modes.

• **Behaviour Description In Frequency Domain:** In future it is planned to include a behavioural description in frequency domain (e.g., for modeling of dynamic stiffness) for the periodic steady state simulation as well as for the frequency response computation, which was one main argument for the harmonic balance method to be preferred over the shooting method (see e.g. [11] for a short introduction and further references). One major reason for the frequency domain description not yet being implemented in SimulationX is that Modelica currently still lacks a standardized way for computations with complex numbers (even if some steps in this direction have already been taken, see e.g. [10]).

• **Event Iterations:** Event iterations are already embedded into the harmonic balance algorithm. But there remains still some work for the treatment of time-discrete variables in special cases.

• **Improved Convergence for Strongly Nonlinear Systems:** As long-term objective the convergence speed of the harmonic balance for strongly nonlinear systems can be improved by time domain preconditioners (see [6]).

• **Autonomous Systems:** The ansatz used for the harmonic balance also bears the potential for the simulation of autonomous systems. The required randomization of the start values for the harmonic balance could be implemented.

• **Detection of Stable/Unstable Limit Cycles:** Up to now there is no automatic discrimination of the stable and unstable branches in the nonlinear frequency response computed via harmonic balance. This can be implemented by an eigenvalue analysis of the monodromy matrix of the computed limit cycles.

References

[1] J. E. Jr. Dennis and Robert B. Schnabel: Numerical Methods for Unconstrained Optimization and Nonlinear Equations. SIAM 1996.

[2] E. L. Allgower and K. Georg: Numerical Continuation Methods: An Introduction. Springer-Verlag, 1990.

[3] http://www.simulationx.com

[4] H. Dresig and F. Holzweißig: Maschinendynamik. 5th ed., Springer-Verlag Berlin, 2004.

[5] A. Meister: Numerik linearer Gleichungssysteme. Vieweg-Verlag, Wiesbaden, 2005.

[6] Ognen J. Nastov: Methods for Circuit Analysis. PHD-theses, Massachusetts Institute of Technology, 1999.

[7] U. Feldmann, U. A. Wever, Q. Zheng, R. Schultz, and H. Wriedt: Algorithms for Modern Circuit Simulation. AEÜ, Vol. 46 (1992), No. 4.

[8] A. Al-shyyab and A. Kahraman: Non-linear dynamic analysis of a multi-mesh gear train using multi-term harmonic balance method: period-one motions. Journal of Sound and Vibration, 284 (2005) 151-172.

[9] Wen-I Liao, Tsung-Jen Teng, and Chau-Shioung Yeh: A method for the response of an elastic half-space to moving sub-Rayleigh point loads. Journal of Sound and Vibration 284 (2005) 173-188.

[10] Peter Aronsson at al.: Meta Programming and Function Overloading in OpenModelica. Modelica 2003, November 3-4, 2003.

[11] Kenneth S. Kundert: Introduction to RF Simulation and Its Application. IEEE Journal of Solid-State Circuits, Vol. 34, No. 9, September 1999.