Institut für Luftfahrtantriebe

Universität Stuttgart

Introduction to Harmonic Balance and application to nonlinear vibrations

M. Krack

Head of the Structural Dynamics group

Institute of Aircraft Propulsion Systems

Department of Aerospace Engineering

University of Stuttgart

What is Harmonic Balance?

Harmonic Balance is an approximation method for computing periodic solutions of ordinary differential equations (ODEs).

The dynamics of many systems (structures, fluids, electrical circuits, …) can be described by ODEs.

The method is only interesting if we do not know the exact solution → nonlinear ODEs.

Periodic oscillations are often of primary technical relevance.

Advantages of Harmonic Balance vs. numerical integration

- computational efficiency (usually a few orders of magnitude faster, because: no integration of long transients, good approximation often already for small number of variables)

- fewer problems with numerical instability or damping

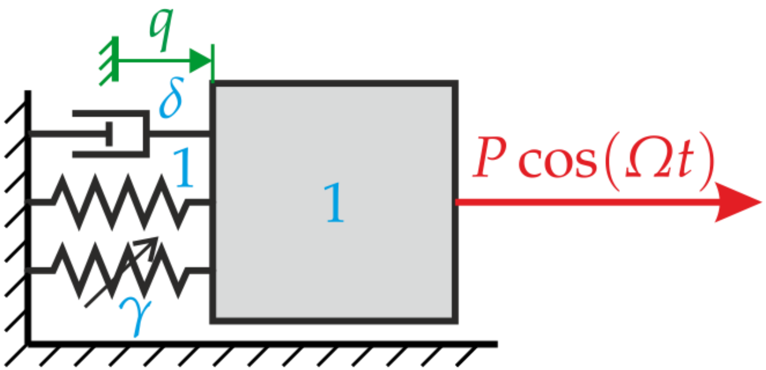
- consideration of phase-lag boundary conditions

- computation also of physically unstable oscillations

Outline of talk

- introductory example: Duffing oscillator
- generalization to nonlinear mechanical systems
- implementation in a simple Matlab tool **NLvib,** application examples
- limitations, ongoing research
- summary, questions, discussion

Duffing oscillator



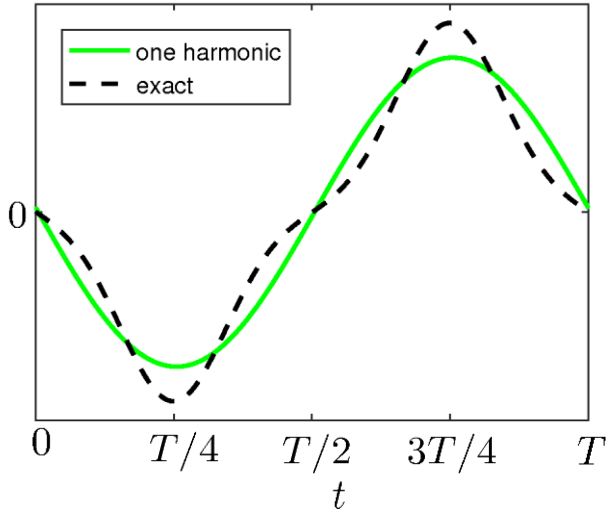
Equation of motion of a single-degree-of-freedom oscillator with cubic spring (Duffing oscillator), with damping and harmonic forcing:

$$\ddot{q} + \delta\dot{q} + q + \gamma q^3 = P\cos(\Omega t)$$

Goal: Compute periodic solutions $q(t+T) = q(t)$ with $T = \frac{2\pi}{\Omega}$

Ansatz: $q(t) \approx q_{\mathrm{h}}(t) = Q_{\mathrm{c}}\cos(\Omega t) + Q_{\mathrm{s}}\sin(\Omega t)$

(only one harmonic here)



Duffing oscillator: Harmonic Balance

Time derivatives of ansatz:

$$
\begin{aligned}
q_\mathrm{h} &= +Q_\mathrm{c} \cos(\Omega t) + Q_\mathrm{s} \sin(\Omega t) \\
\dot{q}_\mathrm{h} &= -Q_\mathrm{c}\Omega \sin(\Omega t) + Q_\mathrm{s}\Omega \cos(\Omega t) \\
\ddot{q}_\mathrm{h} &= -Q_\mathrm{c}\Omega^2 \cos(\Omega t) - Q_\mathrm{s}\Omega^2 \sin(\Omega t)
\end{aligned}
$$

With trigonometric identities

$$
\begin{aligned}
\cos^3 x &= \tfrac{3}{4}\cos x + \tfrac{1}{4}\cos 3x \\
\cos^2 x \ \sin x &= \tfrac{1}{4}\sin x + \tfrac{1}{4}\sin 3x \\
\cos x \ \sin^2 x &= \tfrac{1}{4}\cos x - \tfrac{1}{4}\cos 3x \\
\sin^3 x &= \tfrac{3}{4}\sin x - \tfrac{1}{4}\sin 3x
\end{aligned}
$$

Expansion of nonlinear term

$$
\begin{aligned}
q_\mathrm{h}^3 &= \left(Q_\mathrm{c}\cos(\Omega t) + Q_\mathrm{s}\sin(\Omega t)\right)^3 \\
&= Q_\mathrm{c}^3\cos^3(\Omega t) + 3Q_\mathrm{c}^2 Q_\mathrm{s}\cos^2(\Omega t)\sin(\Omega t) + 3Q_\mathrm{c}Q_\mathrm{s}^2\cos(\Omega t)\sin^2(\Omega t) + Q_\mathrm{s}^3\sin^3(\Omega t) \\
&= \tfrac{3}{4}\left(Q_\mathrm{c}^3 + Q_\mathrm{c}Q_\mathrm{s}^2\right)\cos(\Omega t) + \tfrac{3}{4}\left(Q_\mathrm{s}^3 + Q_\mathrm{c}^2 Q_\mathrm{s}\right)\sin(\Omega t) + (\ldots)\cos(3\Omega t) + (\ldots)\sin(3\Omega t)
\end{aligned}
$$

Substitute into Duffing equation and collect harmonics

$$
\begin{aligned}
&\left[\left(1-\Omega^2\right)Q_\mathrm{c} + \delta\Omega Q_\mathrm{s} + \frac{3}{4}\gamma\left(Q_\mathrm{c}^3 + Q_\mathrm{c}Q_\mathrm{s}^2\right) - P\right]\cos(\Omega t) \\
+&\left[\left(1-\Omega^2\right)Q_\mathrm{s} - \delta\Omega Q_\mathrm{c} + \frac{3}{4}\gamma\left(Q_\mathrm{s}^3 + Q_\mathrm{c}^2 Q_\mathrm{s}\right)\right]\sin(\Omega t) + [\ldots]\cos(3\Omega t) + [\ldots]\sin(3\Omega t) = 0
\end{aligned}
$$

We neglect harmonics with index higher than the ansatz (>1) and balance the harmonics:

$$
\left.
\begin{aligned}
R_\mathrm{c} &:= \left(1-\Omega^2\right)Q_\mathrm{c} + \delta\Omega Q_\mathrm{s} + \frac{3}{4}\gamma\left(Q_\mathrm{c}^3 + Q_\mathrm{c}Q_\mathrm{s}^2\right) - P = 0 \\
R_\mathrm{s} &:= \left(1-\Omega^2\right)Q_\mathrm{s} - \delta\Omega Q_\mathrm{c} + \frac{3}{4}\gamma\left(Q_\mathrm{s}^3 + Q_\mathrm{c}^2 Q_\mathrm{s}\right) = 0
\end{aligned}
\right\}
$$

2 algebraic equations $R_\mathrm{c}, R_\mathrm{s}$ in 2 unknowns $Q_\mathrm{c}, Q_\mathrm{s}$

Duffing oscillator: Frequency response

Frequency response: We are interested in how the solution evolves with Ω.

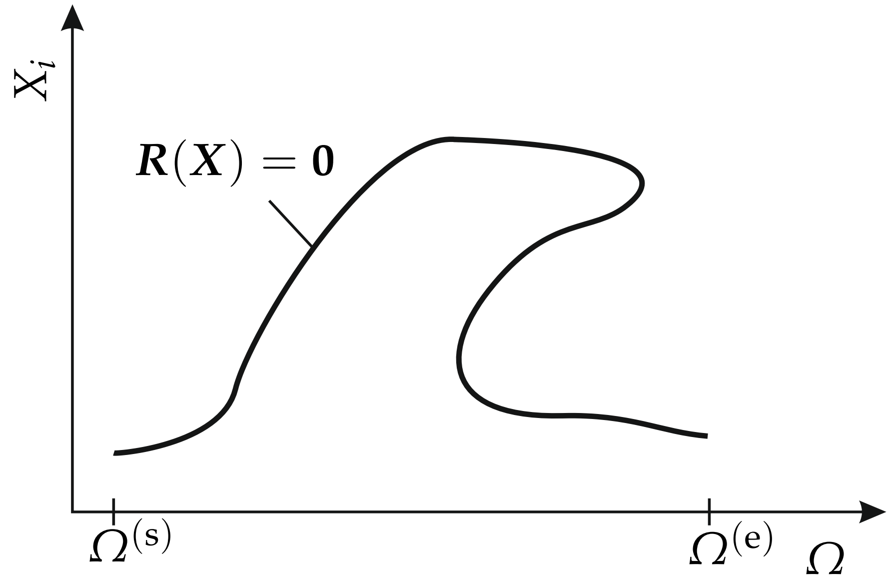
We can formulate this as a continuation problem, with Ω as a free parameter:

solve $\boldsymbol{R}(\boldsymbol{X}) = \begin{bmatrix} R_\mathrm{c} \\ R_\mathrm{s} \end{bmatrix} = \boldsymbol{0}$

with respect to $\boldsymbol{X} = \begin{bmatrix} Q_\mathrm{c} \\ Q_\mathrm{s} \\ \Omega \end{bmatrix}$

with $\boldsymbol{R} \in \mathbb{R}^2, \boldsymbol{X} \in \mathbb{R}^3$

in the interval $\Omega^{(\mathrm{s})} \le \Omega \le \Omega^{(\mathrm{e})}$



- ➢ We will later apply numerical solution and continuation methods.
- ➢ For this simple case, let us develop an analytical solution.

Duffing oscillator: Frequency response

Transform to polar coordinates

$$Q_{\mathrm{c}} = a\cos\theta$$
$$Q_{\mathrm{s}} = a\sin\theta$$
$$\Rightarrow \quad Q_{\mathrm{c}}^2 + Q_{\mathrm{s}}^2 = a^2$$

Substitution into algebraic equations

$$\begin{aligned}
\left(1-\Omega^2\right) a\cos\theta + \delta\Omega a\sin\theta + \frac{3}{4}\gamma\left(a^3\cos^3\theta + a^3\cos\theta\sin^2\theta\right) &= P \qquad (1)\\
\left(1-\Omega^2\right) a\sin\theta - \delta\Omega a\cos\theta + \frac{3}{4}\gamma\left(a^3\sin^3\theta + a^3\cos^2\theta\sin\theta\right) &= 0 \qquad (2)
\end{aligned}$$

Algebraic manipulations of Eq. (1)-(2)

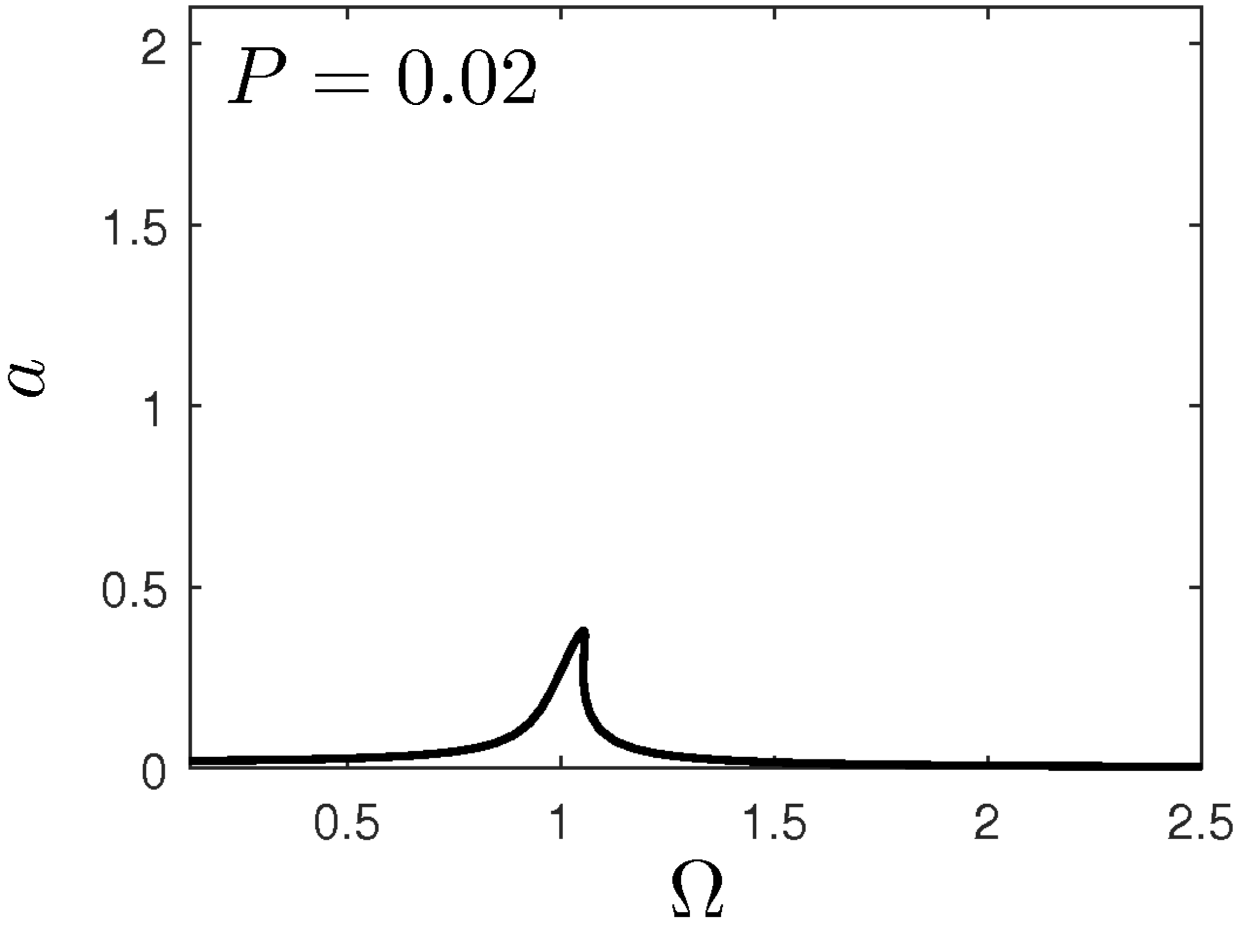
$$\begin{aligned}
\left(1-\Omega^2\right) a + \frac{3}{4}\gamma a^3 &= P\cos\theta \qquad (3)\\
\delta\Omega a &= P\sin\theta \qquad (4)\\
\left[1-\Omega^2+\frac{3}{4}\gamma a^2\right]^2 a^2 + \delta^2\Omega^2 a^2 &= P^2 \qquad (5)\\
\sin\theta &= \frac{\delta\Omega a}{P} \qquad (6)
\end{aligned}$$

It is easier to solve Eq. (5) for Ω:

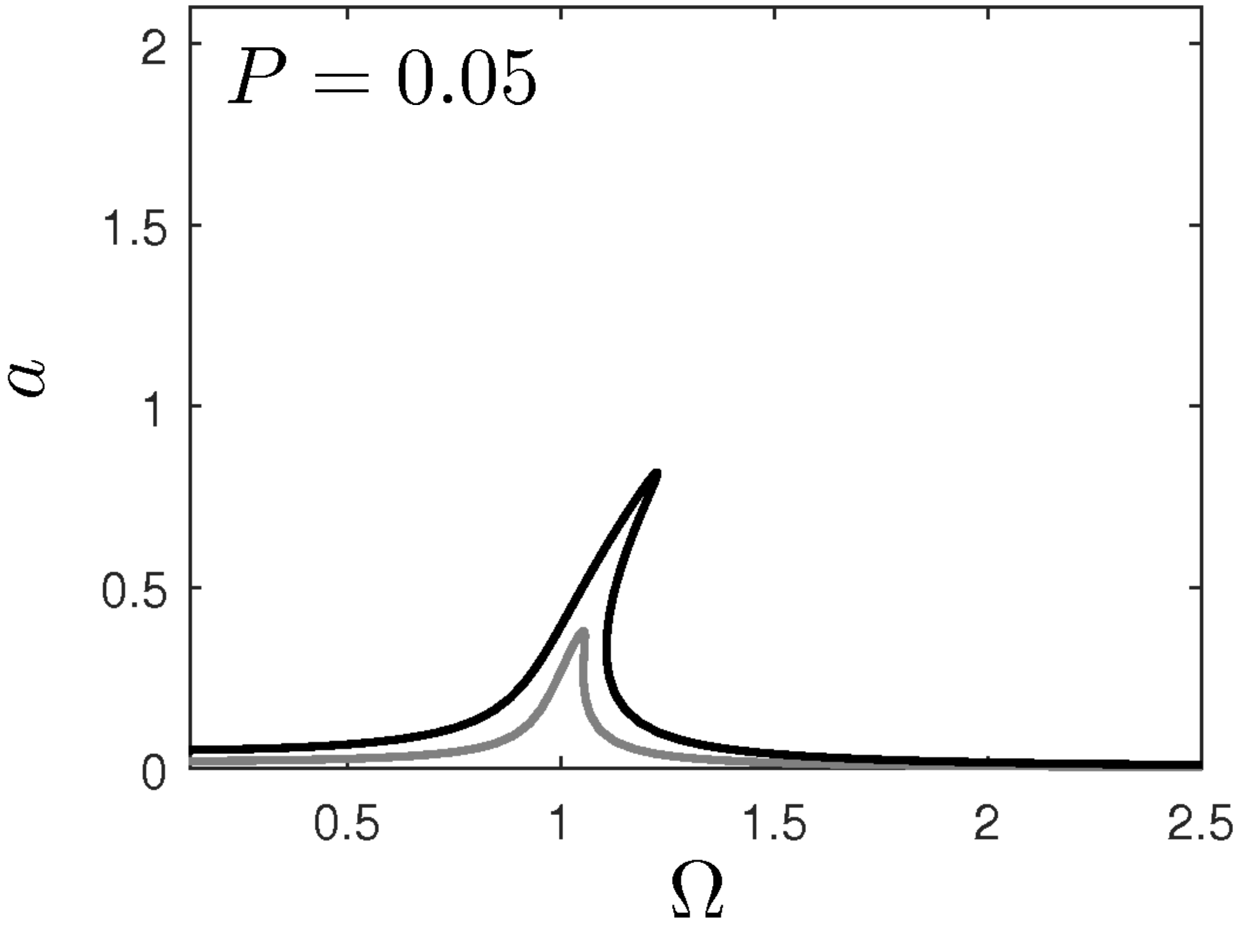
$$\Omega_{1,2}^2 = 1 - \frac{\delta^2}{2} + \frac{3\gamma a^2}{4} \pm \sqrt{\frac{P^2}{a^2} + \frac{\delta^4}{4} - \delta^2 - \frac{3\delta^2\gamma a^2}{4}}$$

We can have zero, one or two real-valued solutions $\Omega_{1,2}^2$.

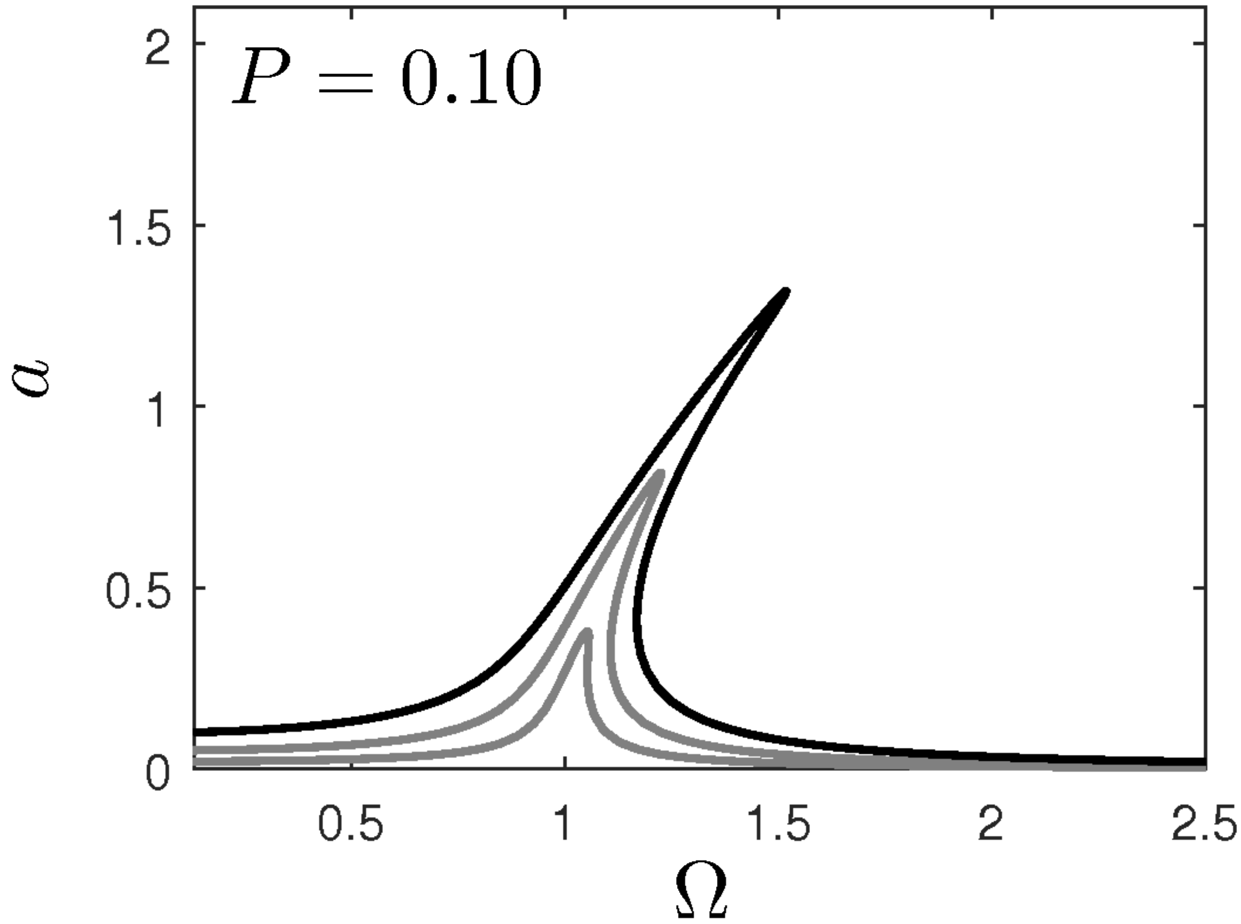
Duffing oscillator: Frequency response



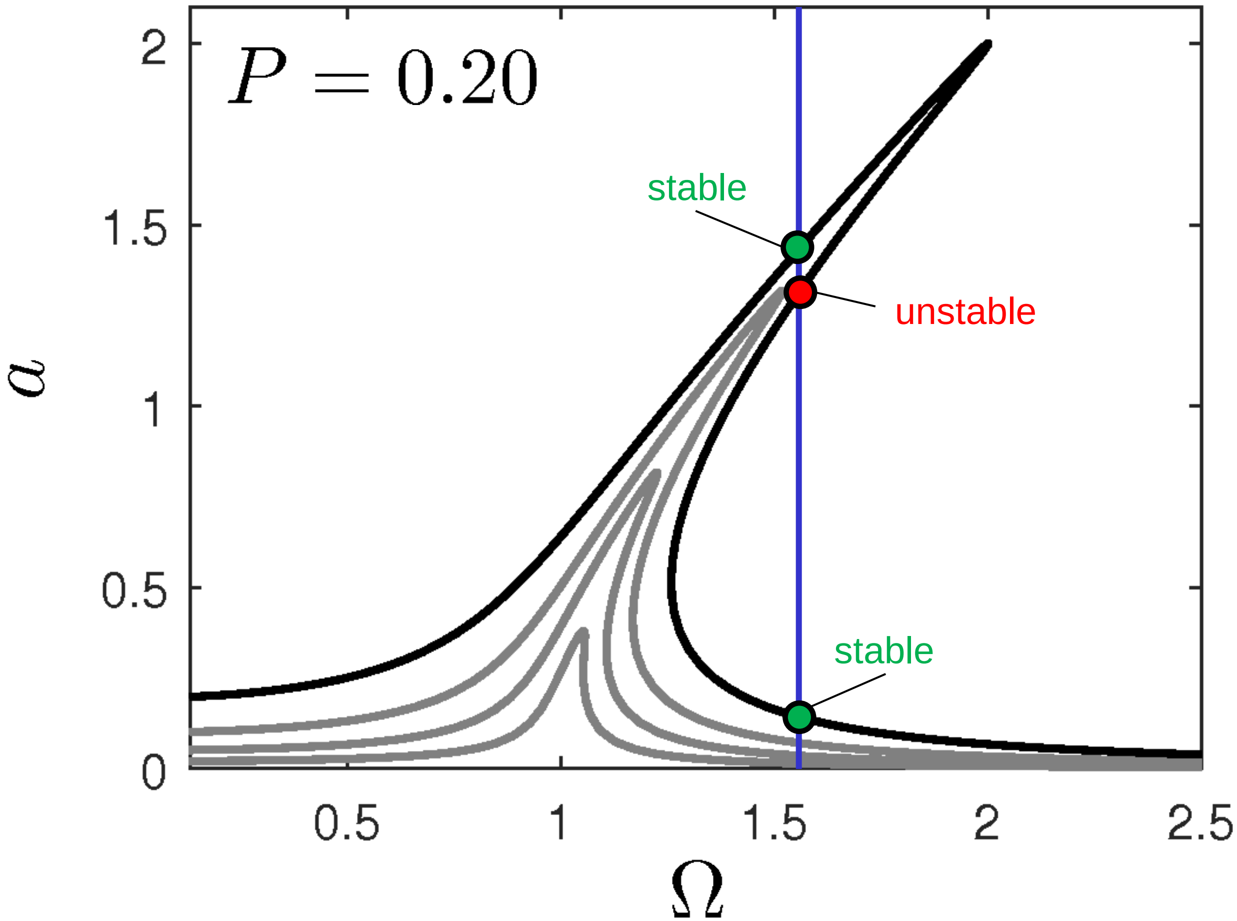
Duffing oscillator: Frequency response



Duffing oscillator: Frequency response



Duffing oscillator: Frequency response

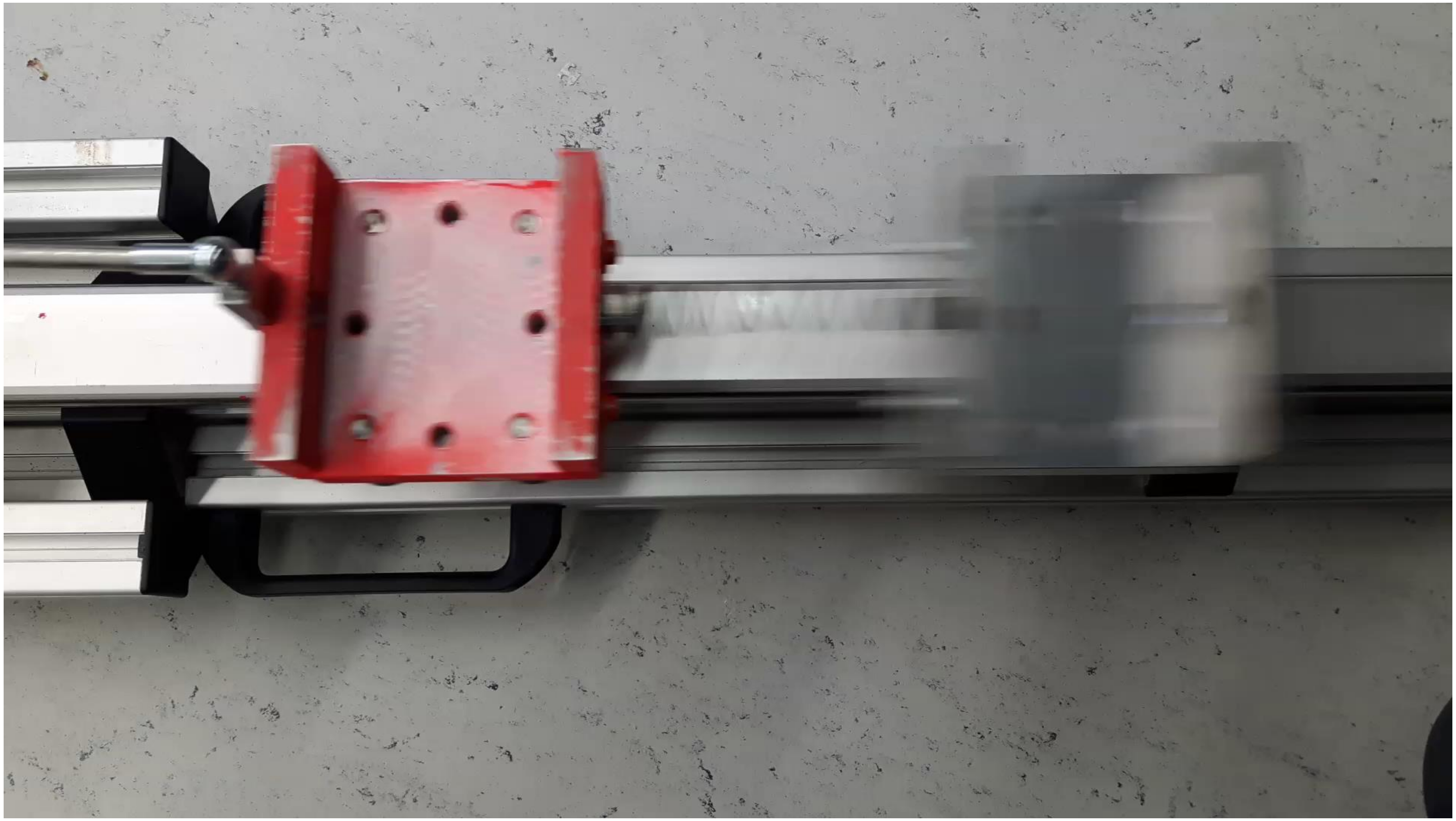


- For a given excitation frequency, multiple steady-state oscillations can be reached, depending on the initial conditions.

Multiple steady states in reality

model

$$m\ddot{q} + d\dot{q} + kq + k_{\mathrm{nl}}q^3 \approx kE\cos(\Omega t)$$



Duffing oscillator: Summary

Idea of Harmonic Balance

- find periodic solutions of ODEs
- ansatz: truncated Fourier series
- balancing of harmonics → algebraic equation system in Fourier coefficients

To be discussed further

- generalization to multiple harmonics
- systematic derivation of equation system
- treatment of generic nonlinearities
- numerical solution

We will focus here on mechanical systems.

Outline of talk

- introductory example: Duffing oscillator
- generalization to nonlinear mechanical systems
- implementation in a simple Matlab tool **NLvib,** application examples
- limitations, ongoing research
- summary, questions, discussion

Nonlinear mechanical system

Equations of motion of a time-invariant mechanical system with periodic forcing

$$\boldsymbol{M}\ddot{\boldsymbol{q}} + \boldsymbol{D}\dot{\boldsymbol{q}} + \boldsymbol{K}\boldsymbol{q} + \boldsymbol{f}_{\mathrm{nl}}\left(\boldsymbol{q}, \dot{\boldsymbol{q}}\right) = \boldsymbol{f}_{\mathrm{ex}}(t)$$

with $\boldsymbol{q}, \boldsymbol{f}_{\mathrm{nl}}, \boldsymbol{f}_{\mathrm{ex}} \in \mathbb{R}^{n\times 1}$ $\boldsymbol{M}, \boldsymbol{D}, \boldsymbol{K} \in \mathbb{R}^{n\times n}$ $\boldsymbol{M} = \boldsymbol{M}^{\mathrm{T}} > \boldsymbol{0}$

and $\boldsymbol{f}_{\mathrm{ex}}(t) = \boldsymbol{f}_{\mathrm{ex}}(t+T)$

Goal: Compute periodic solutions $\boldsymbol{q}(t) = \boldsymbol{q}(t+T)$ with $T = \frac{2\pi}{\Omega}$

Ansatz:
$$\boldsymbol{q}_{\mathrm{h}}(t) = \boldsymbol{Q}_0 + \sum_{k=1}^{\infty} \boldsymbol{Q}_{\mathrm{c},k}\cos(k\Omega t) + \boldsymbol{Q}_{\mathrm{s},k}\sin(k\Omega t)$$

$$= \sum_{k=-\infty}^{\infty} \tilde{\boldsymbol{Q}}_k \mathrm{e}^{\mathrm{i}k\Omega t}$$

$$\boxed{= \Re\{\sum_{k=0}^{\infty} \boldsymbol{Q}_k \mathrm{e}^{\mathrm{i}k\Omega t}\}}$$

We will use this representation.

with
$\boldsymbol{Q}_0, \boldsymbol{Q}_{\mathrm{c},k}, \boldsymbol{Q}_{\mathrm{s},k} \in \mathbb{R}^{n\times 1}$
$\tilde{\boldsymbol{Q}}_k, \boldsymbol{Q}_k \in \mathbb{C}^{n\times 1}\ \forall\ k \neq 0$
$\tilde{\boldsymbol{Q}}_k = \tilde{\boldsymbol{Q}}^*_{-k}$

mathematically equivalent representations of truncated Fourier series

Nonlinear mechanical system

Time derivatives of ansatz:

$$\boldsymbol{q}_\mathrm{h} = \Re\{\sum_{k=0}^{\infty} \boldsymbol{Q}_k \mathrm{e}^{\mathrm{i}k\Omega t}\} \qquad \dot{\boldsymbol{q}}_\mathrm{h} = \Re\{\sum_{k=0}^{\infty} \mathrm{i}k\Omega \boldsymbol{Q}_k \mathrm{e}^{\mathrm{i}k\Omega t}\} \qquad \ddot{\boldsymbol{q}}_\mathrm{h} = \Re\{\sum_{k=0}^{\infty} -(k\Omega)^2 \boldsymbol{Q}_k \mathrm{e}^{\mathrm{i}k\Omega t}\}$$

Substitute into equations of motion

residual due to Fourier approximation of possibly nonsmooth function

$$\boldsymbol{M}\,\Re\{\sum_{k=0}^{\infty} -(k\Omega)^2 \boldsymbol{Q}_k \mathrm{e}^{\mathrm{i}k\Omega t}\} + \boldsymbol{D}\,\Re\{\sum_{k=0}^{\infty} \mathrm{i}k\Omega \boldsymbol{Q}_k \mathrm{e}^{\mathrm{i}k\Omega t}\} + \boldsymbol{K}\,\Re\{\sum_{k=0}^{\infty} \boldsymbol{Q}_k \mathrm{e}^{\mathrm{i}k\Omega t}\} + \boldsymbol{f}_\mathrm{nl} - \boldsymbol{f}_\mathrm{ex} =: \boldsymbol{r} \neq \boldsymbol{0}$$

$$\Re\{\sum_{k=0}^{\infty} \left[-(k\Omega)^2 \boldsymbol{M} + \mathrm{i}k\Omega \boldsymbol{D} + \boldsymbol{K}\right] \boldsymbol{Q}_k \mathrm{e}^{\mathrm{i}k\Omega t}\} + \boldsymbol{f}_\mathrm{nl}(\boldsymbol{q}_\mathrm{h}, \dot{\boldsymbol{q}}_\mathrm{h}) - \boldsymbol{f}_\mathrm{ex} = \boldsymbol{r}$$

$$\Re\{\sum_{k=0}^{\infty} \underbrace{\left(\left[-(k\Omega)^2 \boldsymbol{M} + \mathrm{i}k\Omega \boldsymbol{D} + \boldsymbol{K}\right] \boldsymbol{Q}_k + \boldsymbol{F}_{\mathrm{nl},k} - \boldsymbol{F}_{\mathrm{ex},k} \right)}_{\boldsymbol{R}_k} \mathrm{e}^{\mathrm{i}k\Omega t}\} = \boldsymbol{r}$$

with $\boldsymbol{f}_\mathrm{ex} = \Re\{\sum_{k=0}^{\infty} \boldsymbol{F}_{\mathrm{ex},k} \mathrm{e}^{\mathrm{i}k\Omega t}\}$ $\quad \frac{1}{\pi}\int_0^{2\pi} \boldsymbol{f}_\mathrm{nl}(\boldsymbol{q}_\mathrm{h}, \dot{\boldsymbol{q}}_\mathrm{h})\, \mathrm{e}^{-\mathrm{i}k\Omega t} \mathrm{d}\Omega t = \begin{cases} 2\boldsymbol{F}_{\mathrm{nl},0} & \\ \boldsymbol{F}_{\mathrm{nl},k} & k \neq 0 \end{cases}$

Harmonic Balance:
$\boldsymbol{R}_k = \boldsymbol{0} \; k = 0, \dots, H$

(truncating to order H and setting the Fourier coefficients of the residual to zero)

$$\left.\begin{array}{lcl} \boldsymbol{R}_0(\boldsymbol{Q}_0, \boldsymbol{Q}_1, \dots, \boldsymbol{Q}_H) & = & \boldsymbol{0} \\ \boldsymbol{R}_1(\boldsymbol{Q}_0, \boldsymbol{Q}_1, \dots, \boldsymbol{Q}_H) & = & \boldsymbol{0} \\ & \vdots & \\ \boldsymbol{R}_H(\boldsymbol{Q}_0, \boldsymbol{Q}_1, \dots, \boldsymbol{Q}_H) & = & \boldsymbol{0} \end{array}\right\}$$

n(2*H*+1) algebraic equations in *n*(2*H*+1) unknowns

with
$\boldsymbol{R}_0, \boldsymbol{Q}_0 \in \mathbb{R}^{n\times 1}$
$\boldsymbol{R}_k, \boldsymbol{Q}_k \in \mathbb{C}^{n\times 1}$
$k = 1, \dots, H$

Harmonic Balance as a Galerkin method

Weighted residual method

weight function

$$\int_0^T \boldsymbol{r}\left[\boldsymbol{q}_\mathrm{h}(t), \dot{\boldsymbol{q}}_\mathrm{h}(t)\right] \psi_k^*(t)\,\mathrm{d}t = \boldsymbol{0} \quad k = 0, 1, \ldots$$

Galerkin method: take ansatz functions as weight functions

In our case, the ansatz functions are $\psi_k^* = \mathrm{e}^{-\mathrm{i}k\Omega t}$. We thus obtain:

$$\begin{aligned}
\int_0^T \boldsymbol{r}\left[\boldsymbol{q}_\mathrm{h}(t), \dot{\boldsymbol{q}}_\mathrm{h}(t)\right] \mathrm{e}^{-\mathrm{i}k\Omega t}\,\mathrm{d}t &= \boldsymbol{0} \\
\int_0^T \Re\{\sum_{\ell=0}^{\infty} \boldsymbol{R}_\ell \mathrm{e}^{\mathrm{i}\ell\Omega t}\}\, \mathrm{e}^{-\mathrm{i}k\Omega t}\,\mathrm{d}t &= \boldsymbol{0} \\
\int_0^{2\pi} \Re\{\sum_{\ell=0}^{\infty} \boldsymbol{R}_\ell \mathrm{e}^{\mathrm{i}\ell\tau}\}\, \mathrm{e}^{-\mathrm{i}k\tau}\,\mathrm{d}\tau &= \boldsymbol{0} \qquad \text{with } \tau = \Omega t \\
\int_0^{2\pi} \sum_{\ell=0}^{\infty} \left(\boldsymbol{R}_\ell \frac{\mathrm{e}^{\mathrm{i}\ell\tau}}{2} + \boldsymbol{R}_\ell^* \frac{\mathrm{e}^{-\mathrm{i}\ell\tau}}{2}\right) \mathrm{e}^{-\mathrm{i}k\tau}\,\mathrm{d}\tau &= \boldsymbol{0} \\
\boxed{\boldsymbol{R}_k = \boldsymbol{0}}
\end{aligned}$$

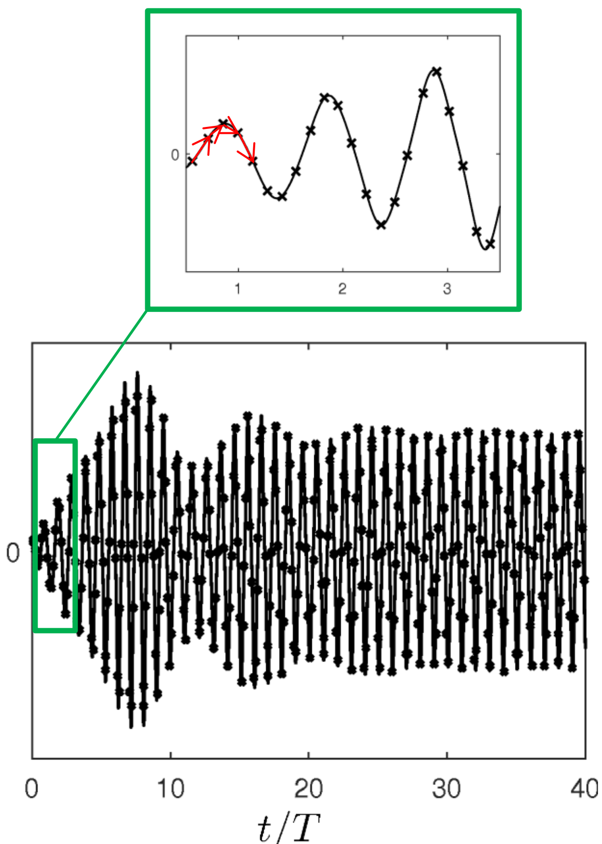
since $\int_0^{2\pi} \mathrm{e}^{\mathrm{i}m\tau}\mathrm{d}\tau = \begin{cases} 2\pi & m = 0 \\ 0 & m \neq 0 \end{cases} \quad m \in \mathbb{Z}$

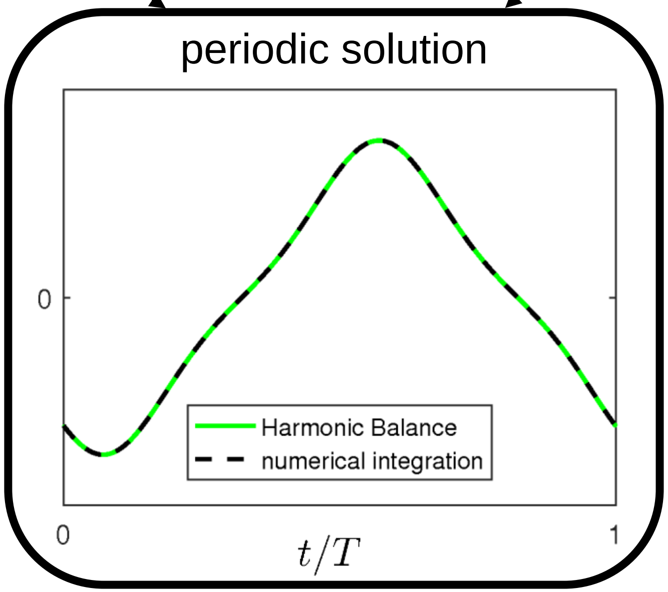
Harmonic Balance vs. numerical integration

numerical integration

successive forward time stepping until transient approaches periodic state

$$\boldsymbol{q}(t_\ell) \longrightarrow \boldsymbol{q}(t_{\ell+1})$$



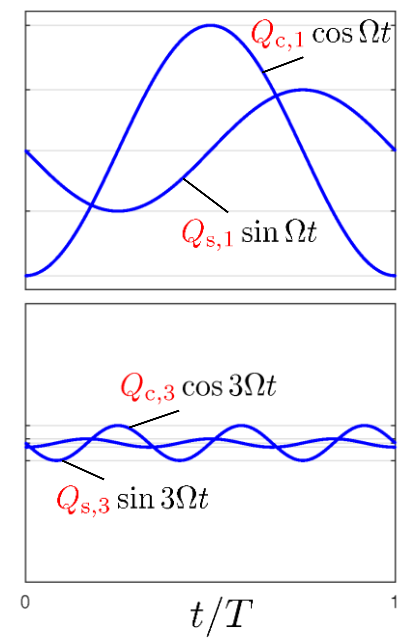


Harmonic Balance

solve algebraic equation system in Fourier coefficients

ansatz

$$\boldsymbol{q} \approx \sum_k Q_{\mathrm{c},k} \cos k\Omega t + Q_{\mathrm{s},k} \sin k\Omega t$$



Nonlinear mechanical system

As both the force equilibrium and the unknowns are Fourier coefficients, Harmonic Balance is a frequency-domain method.

Interpretation of Harmonic Balance

dynamic stiffness matrix

$$\boldsymbol{R}_k = \underbrace{\overbrace{\left[-\left(k\Omega\right)^2 \boldsymbol{M} + \mathrm{i}k\Omega\boldsymbol{D} + \boldsymbol{K}\right]}^{\boldsymbol{S}(k\Omega)} \boldsymbol{Q}_k}_{\text{linear internal forces}} + \underbrace{\boldsymbol{F}_{\mathrm{nl},k}}_{\text{nonlinear internal forces}} - \underbrace{\boldsymbol{F}_{\mathrm{ex},k}}_{\text{external forces}} = \boldsymbol{0}$$

dynamic force equilibrium in the frequency domain

The true challenge is the calculation of the nonlinear force harmonics $\boldsymbol{F}_{\mathrm{nl},k}(\boldsymbol{Q}_0, \dots, \boldsymbol{Q}_H)$!

formal definition: $$\frac{1}{\pi}\int_0^{2\pi} \boldsymbol{f}_{\mathrm{nl}}\left(\boldsymbol{q}_{\mathrm{h}}, \dot{\boldsymbol{q}}_{\mathrm{h}}\right)\ \mathrm{e}^{-\mathrm{i}k\Omega t}\mathrm{d}\Omega t = \begin{cases} 2\boldsymbol{F}_{\mathrm{nl},0} & \\ \boldsymbol{F}_{\mathrm{nl},k} & k = 1, \dots, H \end{cases}$$

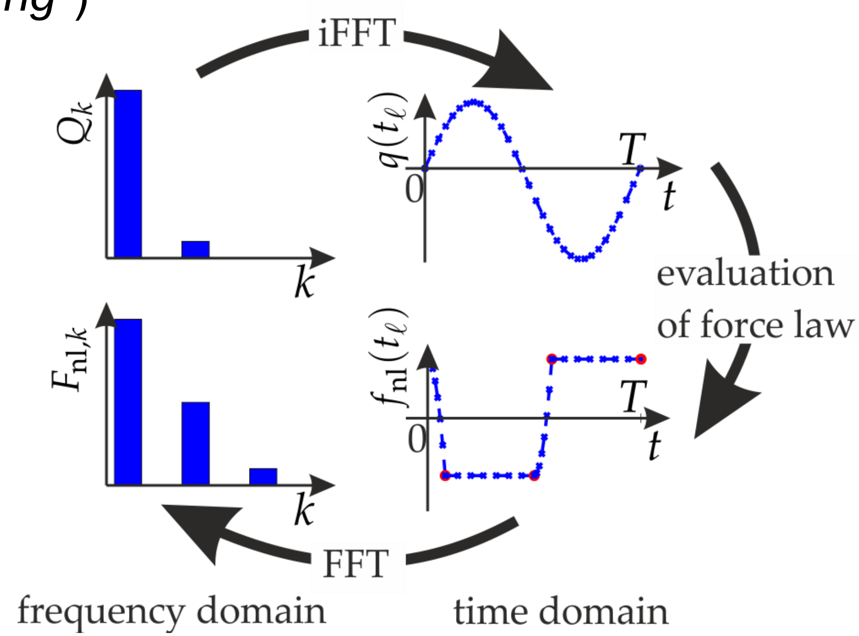
Treatment of nonlinear forces

Opportunities

- **polynomial** forces: closed formulation using convolution theorem
- **piecewise polynomial** (incl. piecewise linear) forces: determine transition times, then as above [PETR03,KRAC13]
- **generic** nonlinear forces: Alternating-Frequency-Time Scheme (AFT)

Alternating-Frequency-Time Scheme ("*sampling*")

$$\{\boldsymbol{F}_{\mathrm{nl},k}\} = \mathrm{FFT}\left[\, \boldsymbol{f}_{\mathrm{nl}}\left(\mathrm{iFFT}\left[\, \{\boldsymbol{Q}_k\} \,\right]\right) \,\right]$$



Number of samples per period:

- theoretical lower limit given by Nyquist-Shannon theorem
- generic nonlinear, perhaps even non-smooth forces: generously oversample if you can afford it

Solution of the algebraic equation system

Task

solve $\boldsymbol{R}(\boldsymbol{x}) = \boldsymbol{0}$

with respect to $\boldsymbol{x}$

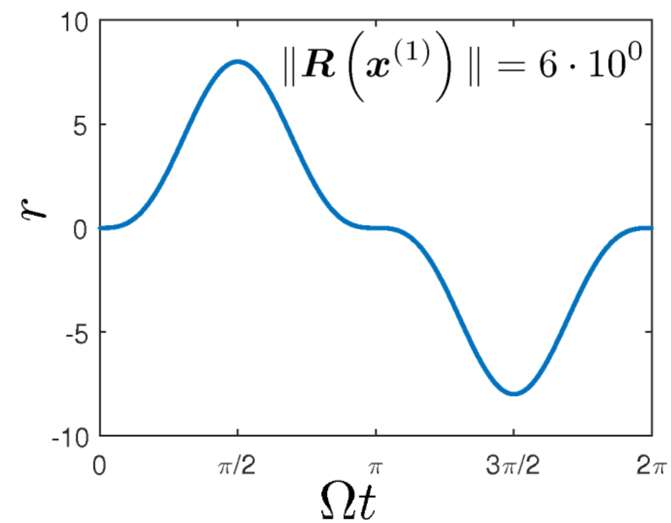
with $\boldsymbol{R}, \boldsymbol{x} \in \mathbb{R}^{n(2H+1)\times 1}$

for Harmonic Balance, we have

$$\boldsymbol{R} = \begin{bmatrix} \boldsymbol{R}_0 \\ \Re\{\boldsymbol{R}_1\} \\ \Im\{\boldsymbol{R}_1\} \\ \vdots \\ \Im\{\boldsymbol{R}_H\} \end{bmatrix} \quad \boldsymbol{x} = \begin{bmatrix} \boldsymbol{Q}_0 \\ \Re\{\boldsymbol{Q}_1\} \\ \Im\{\boldsymbol{Q}_1\} \\ \vdots \\ \Im\{\boldsymbol{Q}_H\} \end{bmatrix}$$

Solvers

- pseudo-time solver
- Newton-like solver
- secant solver
- …



Idea of Newton method

Linearization of residual

$$\boldsymbol{R}\left(\boldsymbol{x}^{(j+1)}\right) \approx \boldsymbol{R}\left(\boldsymbol{x}^{(j)}\right) + \left.\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}^{(j)}} \left(\boldsymbol{x}^{(j+1)} - \boldsymbol{x}^{(j)}\right) = \boldsymbol{0}$$

Iteration step

$$\boldsymbol{x}^{(j+1)} = \boldsymbol{x}^{(j)} - \left.\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}^{(j)}}^{-1} \boldsymbol{R}\left(\boldsymbol{x}^{(j)}\right)$$

Solution of the algebraic equation system

Task

solve $\boldsymbol{R}(\boldsymbol{x}) = \boldsymbol{0}$

with respect to $\boldsymbol{x}$

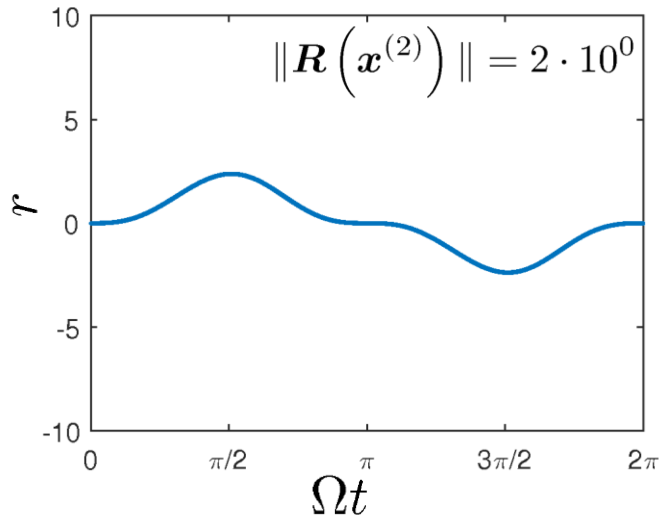
with $\boldsymbol{R}, \boldsymbol{x} \in \mathbb{R}^{n(2H+1)\times 1}$

for Harmonic Balance, we have

$$\boldsymbol{R} = \begin{bmatrix} \boldsymbol{R}_0 \\ \Re\{\boldsymbol{R}_1\} \\ \Im\{\boldsymbol{R}_1\} \\ \vdots \\ \Im\{\boldsymbol{R}_H\} \end{bmatrix} \quad \boldsymbol{x} = \begin{bmatrix} \boldsymbol{Q}_0 \\ \Re\{\boldsymbol{Q}_1\} \\ \Im\{\boldsymbol{Q}_1\} \\ \vdots \\ \Im\{\boldsymbol{Q}_H\} \end{bmatrix}$$

Solvers

- pseudo-time solver
- Newton-like solver
- secant solver
- …



Idea of Newton method

Linearization of residual

$$\boldsymbol{R}\left(\boldsymbol{x}^{(j+1)}\right) \approx \boldsymbol{R}\left(\boldsymbol{x}^{(j)}\right) + \left.\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}^{(j)}} \left(\boldsymbol{x}^{(j+1)} - \boldsymbol{x}^{(j)}\right) = \boldsymbol{0}$$

Iteration step

$$\boldsymbol{x}^{(j+1)} = \boldsymbol{x}^{(j)} - \left.\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}^{(j)}}^{-1} \boldsymbol{R}\left(\boldsymbol{x}^{(j)}\right)$$

Solution of the algebraic equation system

Task

solve $\boldsymbol{R}(\boldsymbol{x}) = \boldsymbol{0}$

with respect to $\boldsymbol{x}$

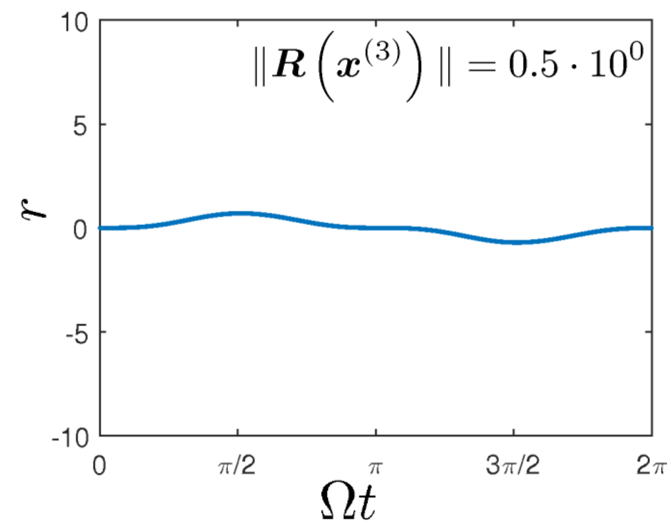
with $\boldsymbol{R}, \boldsymbol{x} \in \mathbb{R}^{n(2H+1)\times 1}$

for Harmonic Balance, we have

$$\boldsymbol{R} = \begin{bmatrix} \boldsymbol{R}_0 \\ \Re\{\boldsymbol{R}_1\} \\ \Im\{\boldsymbol{R}_1\} \\ \vdots \\ \Im\{\boldsymbol{R}_H\} \end{bmatrix} \quad \boldsymbol{x} = \begin{bmatrix} \boldsymbol{Q}_0 \\ \Re\{\boldsymbol{Q}_1\} \\ \Im\{\boldsymbol{Q}_1\} \\ \vdots \\ \Im\{\boldsymbol{Q}_H\} \end{bmatrix}$$

Solvers

- pseudo-time solver
- Newton-like solver
- secant solver
- …



Idea of Newton method

Linearization of residual

$$\boldsymbol{R}\left(\boldsymbol{x}^{(j+1)}\right) \approx \boldsymbol{R}\left(\boldsymbol{x}^{(j)}\right) + \left.\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}^{(j)}} \left(\boldsymbol{x}^{(j+1)} - \boldsymbol{x}^{(j)}\right) = \boldsymbol{0}$$

Iteration step

$$\boldsymbol{x}^{(j+1)} = \boldsymbol{x}^{(j)} - \left.\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}^{(j)}}^{-1} \boldsymbol{R}\left(\boldsymbol{x}^{(j)}\right)$$

Institut für Luftfahrtantriebe | Universität Stuttgart

Solution of the algebraic equation system

Task

solve $\boldsymbol{R}(\boldsymbol{x}) = \boldsymbol{0}$

with respect to $\boldsymbol{x}$

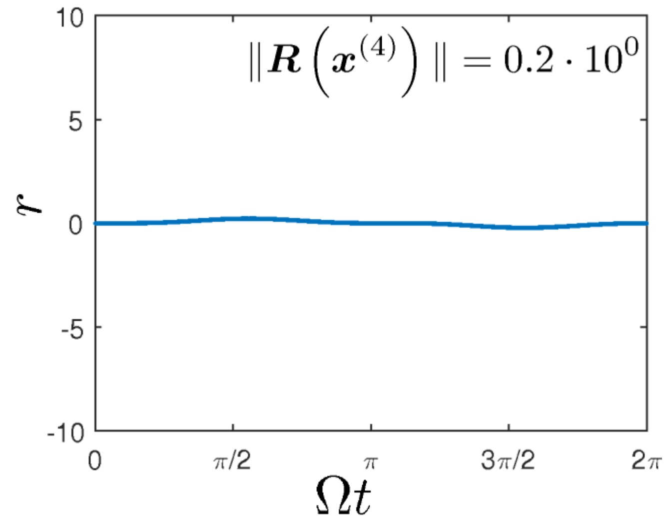
with $\boldsymbol{R}, \boldsymbol{x} \in \mathbb{R}^{n(2H+1)\times 1}$

for Harmonic Balance, we have

$$\boldsymbol{R} = \begin{bmatrix} \boldsymbol{R}_0 \\ \Re\{\boldsymbol{R}_1\} \\ \Im\{\boldsymbol{R}_1\} \\ \vdots \\ \Im\{\boldsymbol{R}_H\} \end{bmatrix} \quad \boldsymbol{x} = \begin{bmatrix} \boldsymbol{Q}_0 \\ \Re\{\boldsymbol{Q}_1\} \\ \Im\{\boldsymbol{Q}_1\} \\ \vdots \\ \Im\{\boldsymbol{Q}_H\} \end{bmatrix}$$

Solvers

- pseudo-time solver
- Newton-like solver
- secant solver
- …



Idea of Newton method

Linearization of residual

$$\boldsymbol{R}\left(\boldsymbol{x}^{(j+1)}\right) \approx \boldsymbol{R}\left(\boldsymbol{x}^{(j)}\right) + \left.\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}^{(j)}} \left(\boldsymbol{x}^{(j+1)} - \boldsymbol{x}^{(j)}\right) = \boldsymbol{0}$$

Iteration step

$$\boldsymbol{x}^{(j+1)} = \boldsymbol{x}^{(j)} - \left.\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}^{(j)}}^{-1} \boldsymbol{R}\left(\boldsymbol{x}^{(j)}\right)$$

Solution of the algebraic equation system

Task

solve $\boldsymbol{R}(\boldsymbol{x}) = \boldsymbol{0}$

with respect to $\boldsymbol{x}$

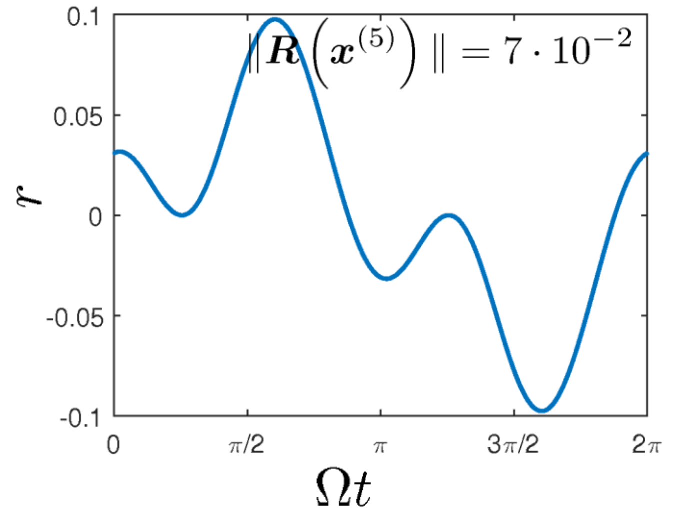
with $\boldsymbol{R}, \boldsymbol{x} \in \mathbb{R}^{n(2H+1)\times 1}$

for Harmonic Balance, we have

$$\boldsymbol{R} = \begin{bmatrix} \boldsymbol{R}_0 \\ \Re\{\boldsymbol{R}_1\} \\ \Im\{\boldsymbol{R}_1\} \\ \vdots \\ \Im\{\boldsymbol{R}_H\} \end{bmatrix} \quad \boldsymbol{x} = \begin{bmatrix} \boldsymbol{Q}_0 \\ \Re\{\boldsymbol{Q}_1\} \\ \Im\{\boldsymbol{Q}_1\} \\ \vdots \\ \Im\{\boldsymbol{Q}_H\} \end{bmatrix}$$

Solvers

- pseudo-time solver
- Newton-like solver
- secant solver
- …



Idea of Newton method

Linearization of residual

$$\boldsymbol{R}\left(\boldsymbol{x}^{(j+1)}\right) \approx \boldsymbol{R}\left(\boldsymbol{x}^{(j)}\right) + \left.\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}^{(j)}} \left(\boldsymbol{x}^{(j+1)} - \boldsymbol{x}^{(j)}\right) = \boldsymbol{0}$$

Iteration step

$$\boldsymbol{x}^{(j+1)} = \boldsymbol{x}^{(j)} - \left.\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}^{(j)}}^{-1} \boldsymbol{R}\left(\boldsymbol{x}^{(j)}\right)$$

Solution of the algebraic equation system

Task

solve $\boldsymbol{R}(\boldsymbol{x}) = \boldsymbol{0}$

with respect to $\boldsymbol{x}$

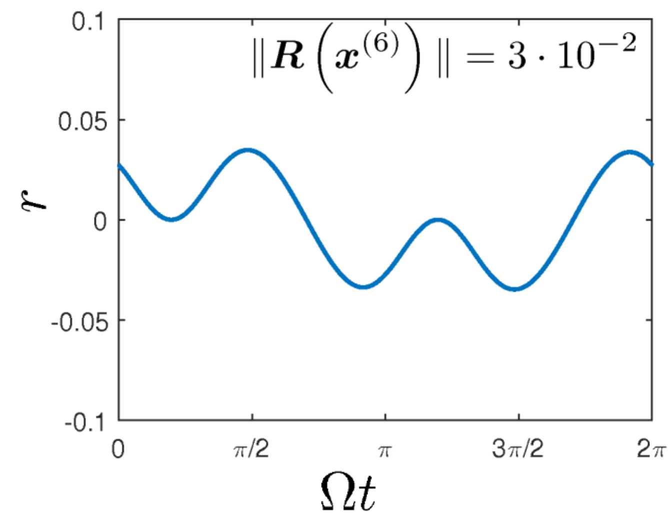
with $\boldsymbol{R}, \boldsymbol{x} \in \mathbb{R}^{n(2H+1)\times 1}$

for Harmonic Balance, we have

$$\boldsymbol{R} = \begin{bmatrix} \boldsymbol{R}_0 \\ \Re\{\boldsymbol{R}_1\} \\ \Im\{\boldsymbol{R}_1\} \\ \vdots \\ \Im\{\boldsymbol{R}_H\} \end{bmatrix} \quad \boldsymbol{x} = \begin{bmatrix} \boldsymbol{Q}_0 \\ \Re\{\boldsymbol{Q}_1\} \\ \Im\{\boldsymbol{Q}_1\} \\ \vdots \\ \Im\{\boldsymbol{Q}_H\} \end{bmatrix}$$

Solvers

- pseudo-time solver
- Newton-like solver
- secant solver
- …



Idea of Newton method

Linearization of residual

$$\boldsymbol{R}\left(\boldsymbol{x}^{(j+1)}\right) \approx \boldsymbol{R}\left(\boldsymbol{x}^{(j)}\right) + \left.\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}^{(j)}} \left(\boldsymbol{x}^{(j+1)} - \boldsymbol{x}^{(j)}\right) = \boldsymbol{0}$$

Iteration step

$$\boldsymbol{x}^{(j+1)} = \boldsymbol{x}^{(j)} - \left.\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}^{(j)}}^{-1} \boldsymbol{R}\left(\boldsymbol{x}^{(j)}\right)$$

Solution of the algebraic equation system

Task

solve $\boldsymbol{R}(\boldsymbol{x}) = \boldsymbol{0}$

with respect to $\boldsymbol{x}$

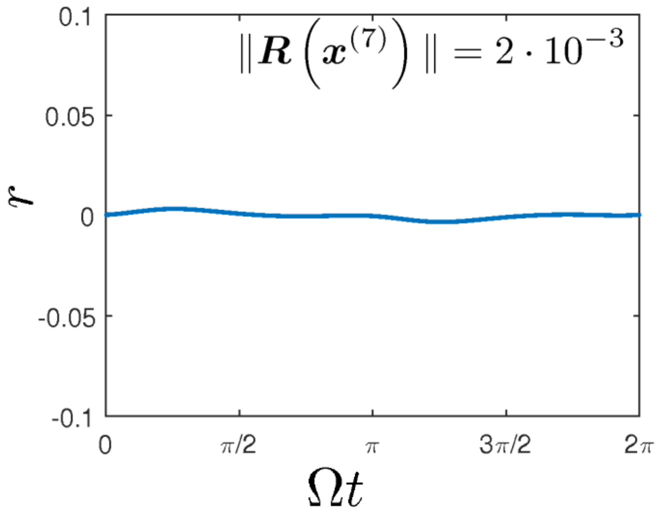
with $\boldsymbol{R}, \boldsymbol{x} \in \mathbb{R}^{n(2H+1)\times 1}$

for Harmonic Balance, we have

$$\boldsymbol{R} = \begin{bmatrix} \boldsymbol{R}_0 \\ \Re\{\boldsymbol{R}_1\} \\ \Im\{\boldsymbol{R}_1\} \\ \vdots \\ \Im\{\boldsymbol{R}_H\} \end{bmatrix} \quad \boldsymbol{x} = \begin{bmatrix} \boldsymbol{Q}_0 \\ \Re\{\boldsymbol{Q}_1\} \\ \Im\{\boldsymbol{Q}_1\} \\ \vdots \\ \Im\{\boldsymbol{Q}_H\} \end{bmatrix}$$

Solvers

- pseudo-time solver
- Newton-like solver
- secant solver
- …



Idea of Newton method

Linearization of residual

$$\boldsymbol{R}\left(\boldsymbol{x}^{(j+1)}\right) \approx \boldsymbol{R}\left(\boldsymbol{x}^{(j)}\right) + \left.\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}^{(j)}} \left(\boldsymbol{x}^{(j+1)} - \boldsymbol{x}^{(j)}\right) = \boldsymbol{0}$$

Iteration step

$$\boldsymbol{x}^{(j+1)} = \boldsymbol{x}^{(j)} - \left.\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}^{(j)}}^{-1} \boldsymbol{R}\left(\boldsymbol{x}^{(j)}\right)$$

Solution of the algebraic equation system

Task

solve $\boldsymbol{R}(\boldsymbol{x}) = \boldsymbol{0}$

with respect to $\boldsymbol{x}$

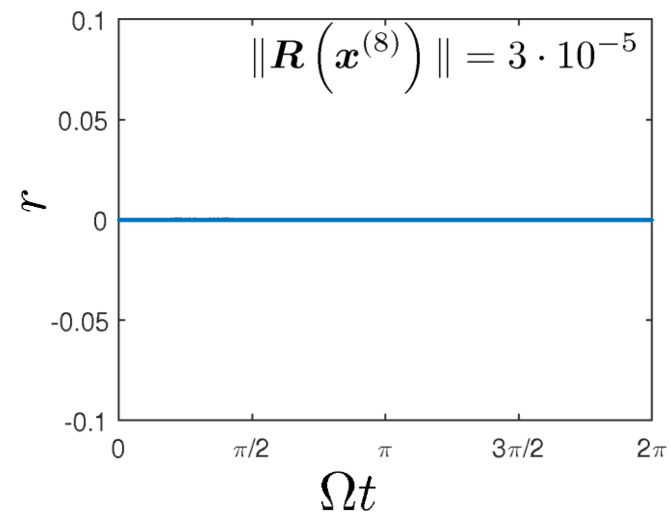
with $\boldsymbol{R}, \boldsymbol{x} \in \mathbb{R}^{n(2H+1)\times 1}$

for Harmonic Balance, we have

$$\boldsymbol{R} = \begin{bmatrix} \boldsymbol{R}_0 \\ \Re\{\boldsymbol{R}_1\} \\ \Im\{\boldsymbol{R}_1\} \\ \vdots \\ \Im\{\boldsymbol{R}_H\} \end{bmatrix} \quad \boldsymbol{x} = \begin{bmatrix} \boldsymbol{Q}_0 \\ \Re\{\boldsymbol{Q}_1\} \\ \Im\{\boldsymbol{Q}_1\} \\ \vdots \\ \Im\{\boldsymbol{Q}_H\} \end{bmatrix}$$

Solvers

- pseudo-time solver
- Newton-like solver
- secant solver
- …



Idea of Newton method

Linearization of residual

$$\boldsymbol{R}\left(\boldsymbol{x}^{(j+1)}\right) \approx \boldsymbol{R}\left(\boldsymbol{x}^{(j)}\right) + \left.\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}^{(j)}} \left(\boldsymbol{x}^{(j+1)} - \boldsymbol{x}^{(j)}\right) = \boldsymbol{0}$$

Iteration step

$$\boldsymbol{x}^{(j+1)} = \boldsymbol{x}^{(j)} - \left.\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}^{(j)}}^{-1} \boldsymbol{R}\left(\boldsymbol{x}^{(j)}\right)$$

Solution of the algebraic equation system

Task

solve $\boldsymbol{R}(\boldsymbol{x}) = \boldsymbol{0}$

with respect to $\boldsymbol{x}$

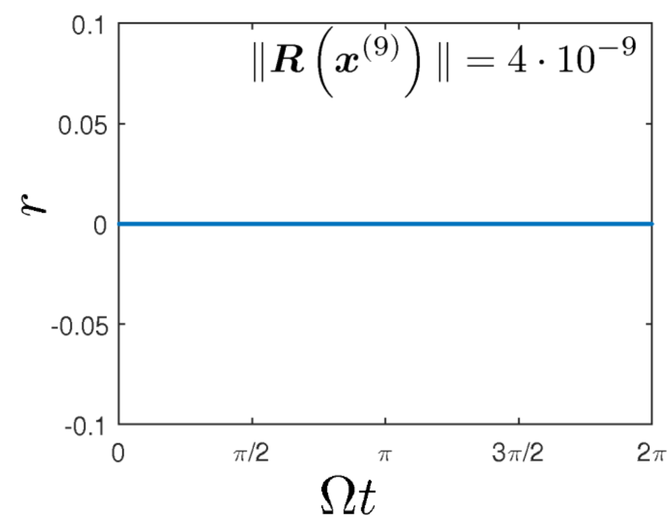
with $\boldsymbol{R}, \boldsymbol{x} \in \mathbb{R}^{n(2H+1)\times 1}$

for Harmonic Balance, we have

$$\boldsymbol{R} = \begin{bmatrix} \boldsymbol{R}_0 \\ \Re\{\boldsymbol{R}_1\} \\ \Im\{\boldsymbol{R}_1\} \\ \vdots \\ \Im\{\boldsymbol{R}_H\} \end{bmatrix} \quad \boldsymbol{x} = \begin{bmatrix} \boldsymbol{Q}_0 \\ \Re\{\boldsymbol{Q}_1\} \\ \Im\{\boldsymbol{Q}_1\} \\ \vdots \\ \Im\{\boldsymbol{Q}_H\} \end{bmatrix}$$

Solvers

- pseudo-time solver
- Newton-like solver
- secant solver
- …



Idea of Newton method

Linearization of residual

$$\boldsymbol{R}\left(\boldsymbol{x}^{(j+1)}\right) \approx \boldsymbol{R}\left(\boldsymbol{x}^{(j)}\right) + \left.\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}^{(j)}} \left(\boldsymbol{x}^{(j+1)} - \boldsymbol{x}^{(j)}\right) = \boldsymbol{0}$$

Iteration step

$$\boldsymbol{x}^{(j+1)} = \boldsymbol{x}^{(j)} - \left.\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}^{(j)}}^{-1} \boldsymbol{R}\left(\boldsymbol{x}^{(j)}\right)$$

- ➢ fast convergence near solution
- ➢ adjustments required for global convergence
- ➢ analytical gradients greatly reduce computation time

Computation of a solution branch (under variation of a free parameter)

Example: Frequency response analysis

solve $\boldsymbol{R}(\boldsymbol{X}) = \boldsymbol{0}$

with respect to $\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x} \\ \Omega \end{bmatrix}$

with $\boldsymbol{R}, \boldsymbol{x} \in \mathbb{R}^{n(2H+1)\times 1}$

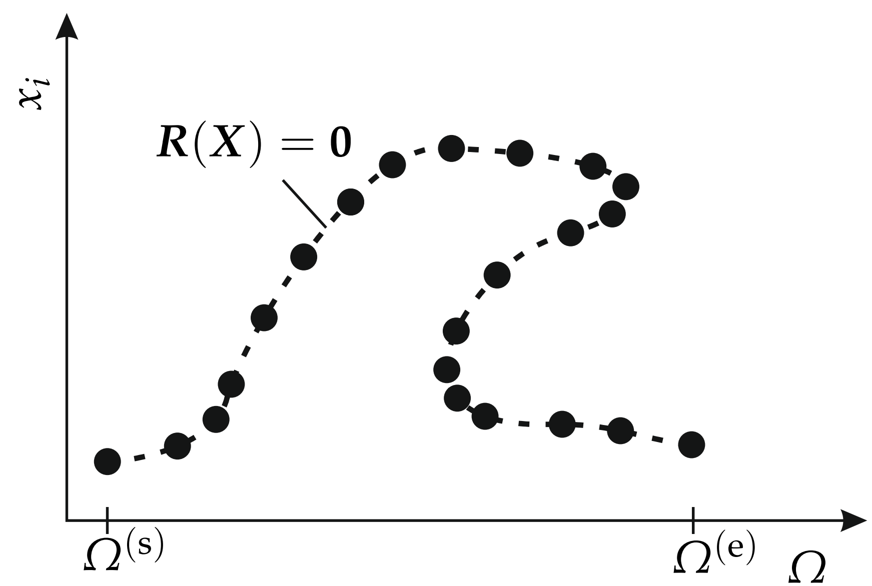
in the interval $\Omega^{(\mathrm{s})} \leq \Omega \leq \Omega^{(\mathrm{e})}$

➢ This is a job for a continuation method!

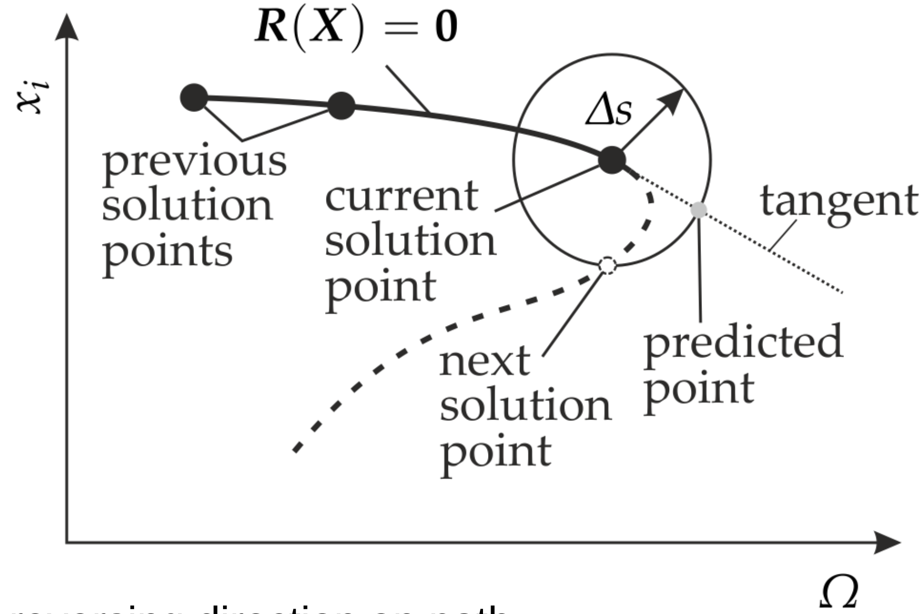
Numerical path continuation: generate a sequence of suitably spaced solution points within the given parameter range, and go around turning points (if any).

similar analyses

- analysis of self-excited limit cycles
- nonlinear modal analysis
- tracking of resonances
- tracking of bifurcation points



Ingredients of a continuation method



Predictor

popular variants:

- tangent
- secant
- power series expansion

Parametrization: Quo vadis?

- avoid returning to same solution point or reversing direction on path
- in most cases: additional equation, free parameter as additional unknown
- popular variants: arc length; local; orthogonal

Corrector: apply solver!

Step length control:

- as small as necessary to ensure convergence and not overlook important characteristics of the solution
- as large as possible to avoid spurious computational effort
- empirical rules depending on solver and tolerances, upper and lower bounds

Outline of talk

- introductory example: Duffing oscillator
- generalization to nonlinear mechanical systems
- implementation in a simple Matlab tool **NLvib,** application examples
- limitations, ongoing research
- summary, questions, discussion

NLvib – a Matlab tool for nonlinear vibration analysis

Source code and documentation available via www.ila.uni-stuttgart.de/nlvib

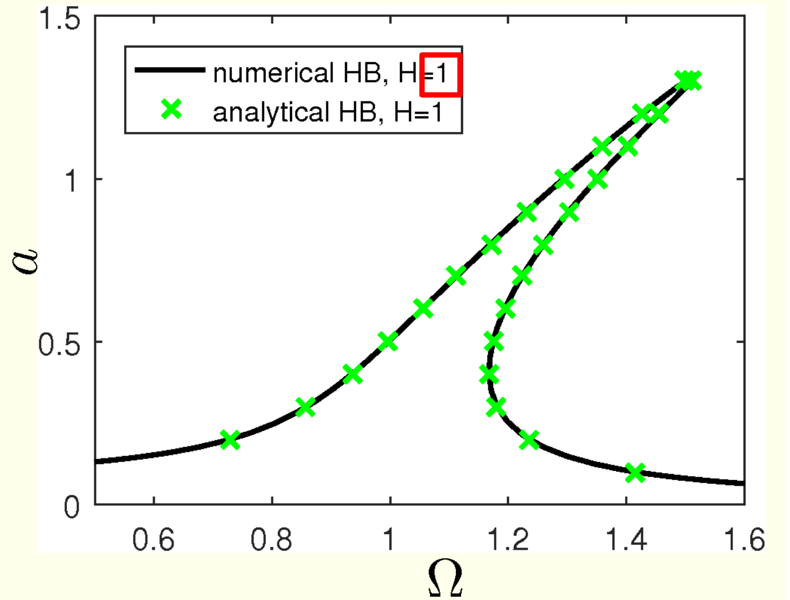
Features

- Harmonic Balance with Alternating Frequency-Time Scheme
- Shooting, Newmark numerical time step integration
- *solver:* predictor-corrector continuation with Newton-like corrector (‘fsolve’), analytical gradients
- *nonlinearities*
  - local generic nonlinear elements $\boldsymbol{f}_{\mathrm{nl}} = \sum_e \boldsymbol{w}_e \, f_{\mathrm{nl},e}\left(\boldsymbol{w}_e^{\mathrm{T}}\boldsymbol{q},\, \boldsymbol{w}_e^{\mathrm{T}}\boldsymbol{u}\right)$
  - (distributed) polynomial stiffness nonlinearity $\boldsymbol{f}_{\mathrm{nl}} = \sum_k E_{ki} \prod_j q_j^{p_{kj}}$
- *analysis types*
  - frequency response
  - nonlinear modal analysis

NLvib: Duffing oscillator

$$\mu\ddot{q} + \delta\dot{q} + \kappa q + \gamma q^3 = P\cos(\Omega t)$$

singleDOFoscillator_cubicSpring.m



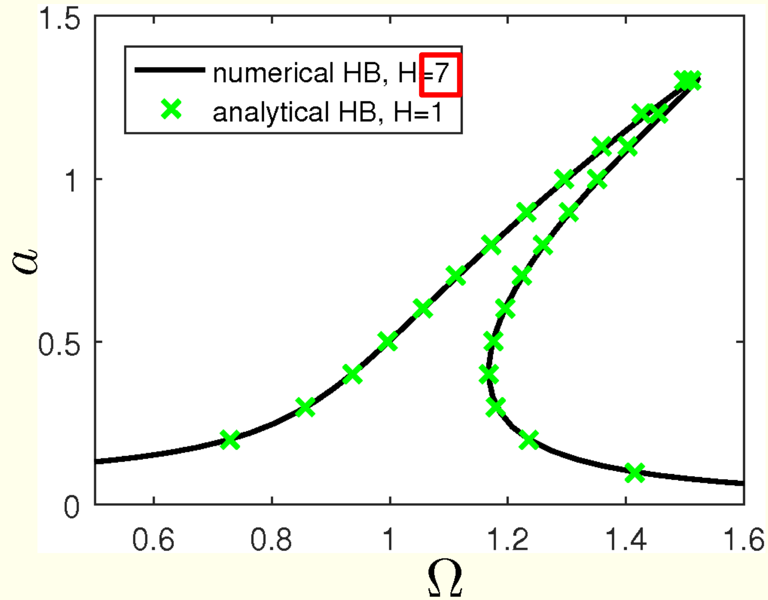

```
% Parameters of the Duffing oscillator
mu = 1;
delta = 0.05;
kappa = 1;
gamma = 1;
P = .1;

% Analysis parameters
H = 1;        % harmonic order
N = 2^7;      % number of time samples per period
Om_s = .5;    % start frequency
Om_e = 1.6;   % end frequency

% Initial guess (from underlying linear system)
Q = (-Om_s^2*mu+1i*Om_s*delta+kappa)\P;
x0 = [0;real(Q);-imag(Q);zeros(2*(H-1),1)];

% Solve and continue w.r.t. Om
ds = .01;                           % Path continuation step size
Sopt = struct('jac','none');        % No analytical Jacobian provided here
X = solve_and_continue(x0,...
    @(X) HB_residual_singleDOFcubicSpring(X,mu,delta,kappa,gamma,P,H,N),...
    Om_s,Om_e,ds,Sopt);

% Determine excitation frequency and amplitude (magnitude of fundamental
% harmonic)
Om = X(end,:);
a = sqrt(X(2,:).^2 + X(3,:).^2);
```

NLvib: Duffing oscillator

$$\mu\ddot{q} + \delta\dot{q} + \kappa q + \gamma q^3 = P\cos(\Omega t)$$

singleDOFoscillator_cubicSpring.m

```
% Parameters of the Duffing oscillator
mu = 1;
delta = 0.05;
kappa = 1;
gamma = 1;
P = .1;

% Analysis parameters
H = 7;          % harmonic order
N = 2^7;        % number of time samples per period
Om_s = .5;      % start frequency
Om_e = 1.6;     % end frequency

% Initial guess (from underlying linear system)
Q = (-Om_s^2*mu+1i*Om_s*delta+kappa)\P;
x0 = [0;real(Q);-imag(Q);zeros(2*(H-1),1)];

% Solve and continue w.r.t. Om
ds = .01;                          % Path continuation step size
Sopt = struct('jac','none');       % No analytical Jacobian provided here
X = solve_and_continue(x0,...
    @(X) HB_residual_singleDOFcubicSpring(X,mu,delta,kappa,gamma,P,H,N),...
    Om_s,Om_e,ds,Sopt);

% Determine excitation frequency and amplitude (magnitude of fundamental
% harmonic)
Om = X(end,:);
a = sqrt(X(2,:).^2 + X(3,:).^2);
```



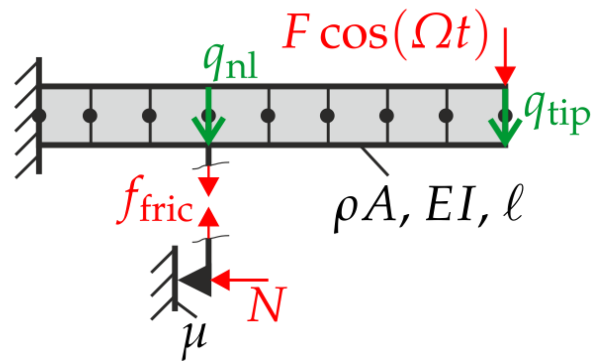
NLvib: Duffing oscillator

$$\mu\ddot{q} + \delta\dot{q} + \kappa q + \gamma q^3 = P\cos(\Omega t)$$

HB_residual_singleDOFcubicSpring.m

```
function R = HB_residual_singleDOFcubicSpring(X,mu,delta,kappa,gamma,P,H,N)
% Conversion of real-valued to complex-valued harmonics of generalized coordinates q
Q = [X(1);X(2:2:end-1)-1i*X(3:2:end-1)];

% Excitation frequency
Om = X(end);

% P is the fundamental harmonic of the external forcing
Fex = [0;P;zeros(H-1,1)];

% Specify time samples along period and apply inverse discrete Fourier transform
tau = (0:2*pi/N:2*pi-2*pi/N)';
qnl = real(exp(1i*tau*(0:H))*Q);

% Evaluate nonlinear force in the time domain
fnl = gamma*qnl.^3;

% Forward Discrete Fourier Transform, truncation, conversion to half spectrum notation
Fnlc = fft(fnl)/N;
Fnl = [real(Fnlc(1));2*Fnlc(2:H+1)];

% Dynamic force equilibrium (complex-valued)
Rc = ( -((0:H)'*Om).^2 * mu + 1i*(0:H)'*Om * delta + kappa ).*Q+Fnl-Fex;

% Conversion from complex-valued to real-valued residual
R = [real(Rc(1));real(Rc(2:end));-imag(Rc(2:end))];
```

NLvib: friction-damped beam



Equation of motion

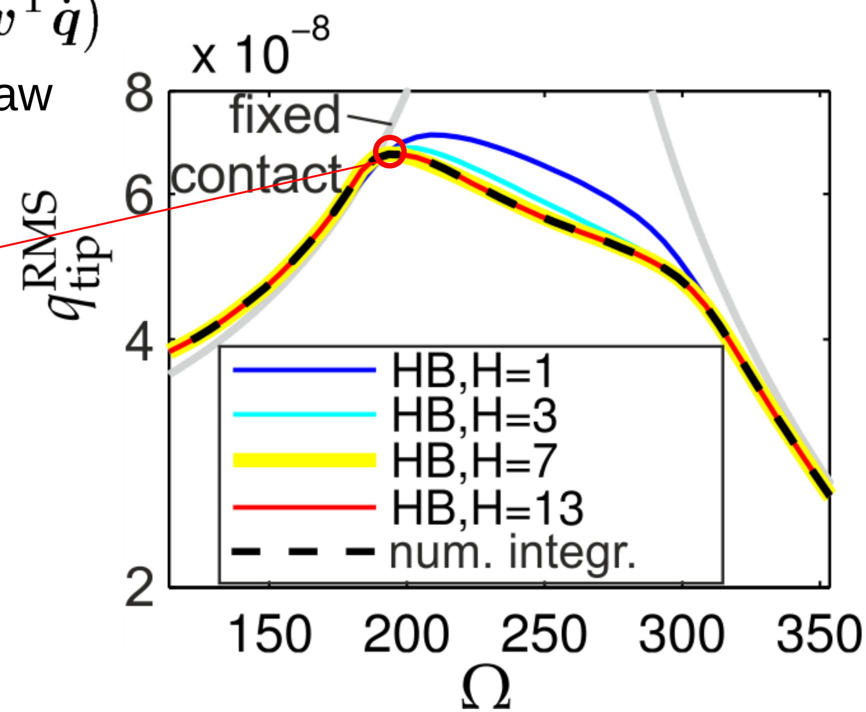
$$\boldsymbol{M}\ddot{\boldsymbol{q}} + \boldsymbol{D}\dot{\boldsymbol{q}} + \boldsymbol{K}\boldsymbol{q} + \boldsymbol{f}_{\mathrm{nl}} = \Re\{\boldsymbol{F}_{\mathrm{ex},1}\mathrm{e}^{\mathrm{i}\Omega t}\}$$

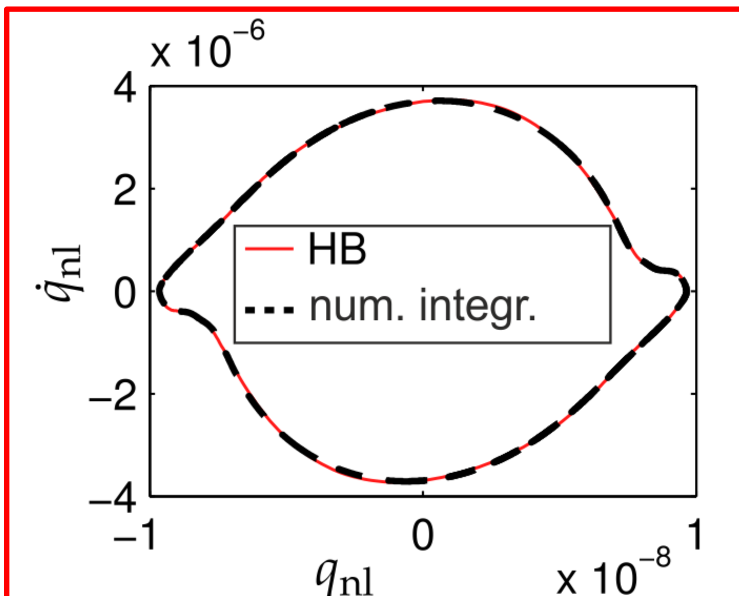
with the nonlinear force

$$\boldsymbol{f}_{\mathrm{nl}} = \boldsymbol{w} f_{\mathrm{fric}}\left(q_{\mathrm{nl}}, \dot{q}_{\mathrm{nl}}\right) = \boldsymbol{w} f_{\mathrm{fric}}\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{q}, \boldsymbol{w}^{\mathrm{T}}\dot{\boldsymbol{q}}\right)$$

and the regularized Coulomb dry friction law

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

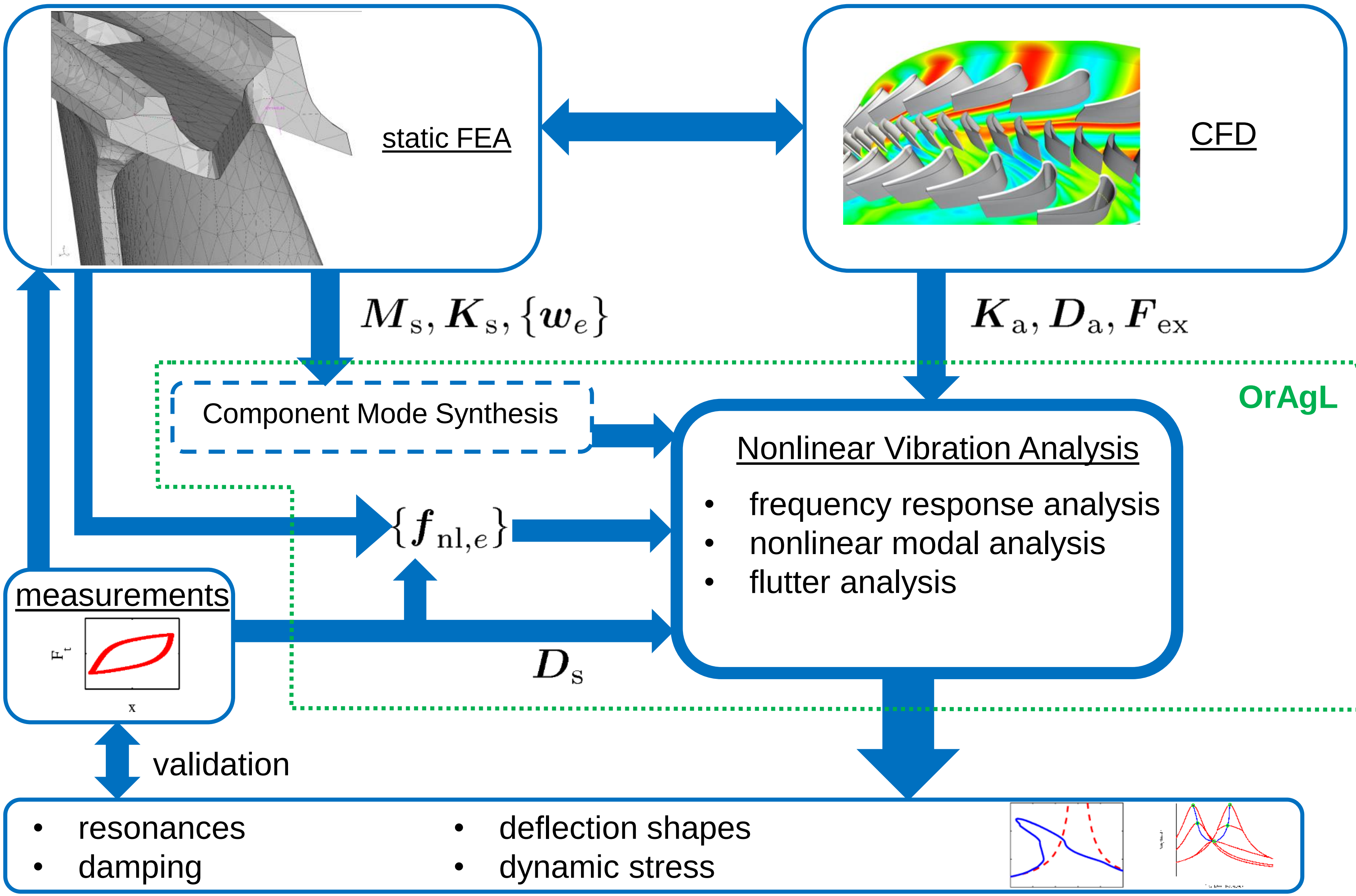




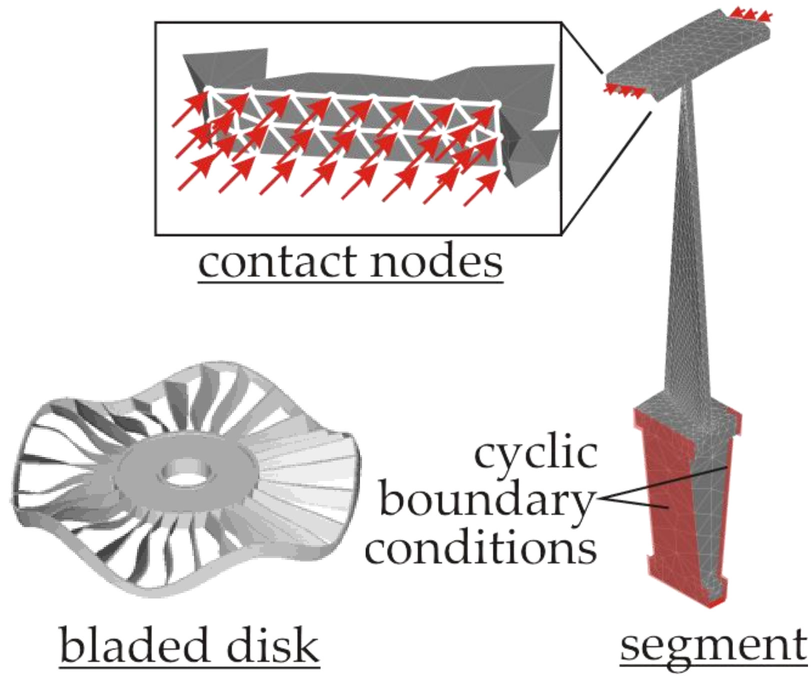
computation times:

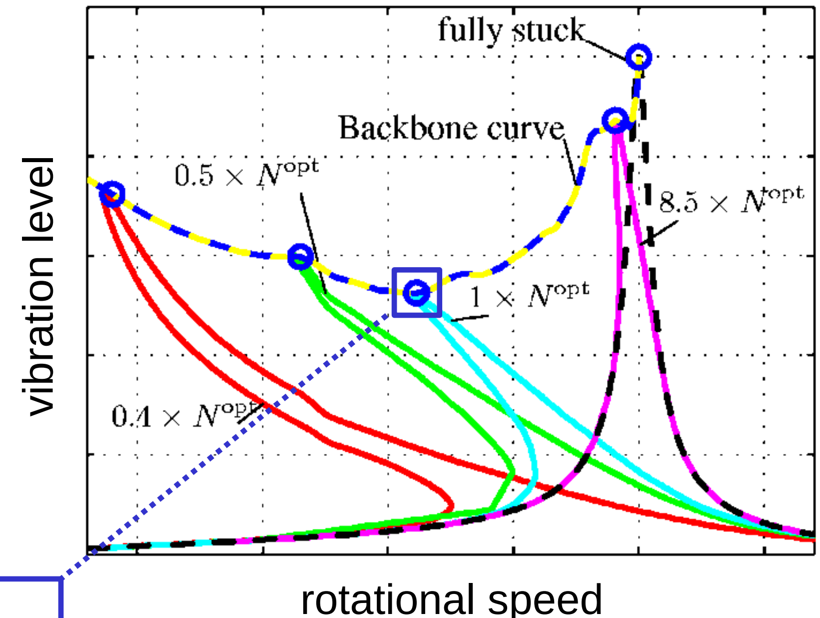
- HB, whole frequency response: ~1.3 s
- numerical integration, single frequency: ~30 s

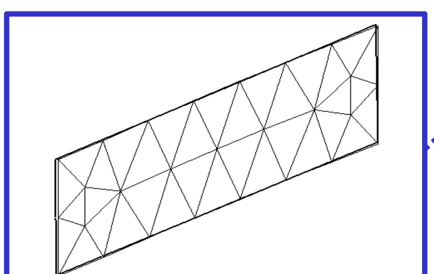
Vibration prediction of bladed disks with friction joints

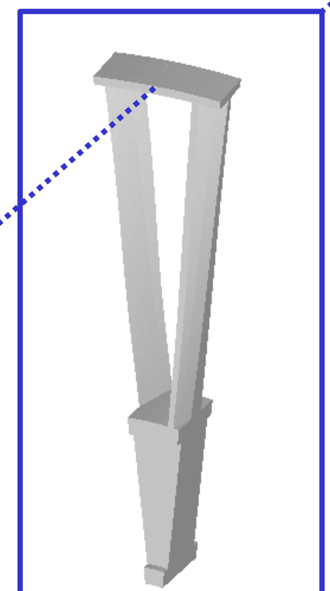


Frequency response of a bladed disk with shroud contact









For comprehensive rig and engine validation (with a much more realistic model), attend the presentation of paper GT2018-75186.

Outline of talk

- introductory example: Duffing oscillator
- generalization to nonlinear mechanical systems
- implementation in a simple Matlab tool **NLvib,** application examples
- limitations, ongoing research
- summary, questions, discussion

Harmonic Balance has to main limitations

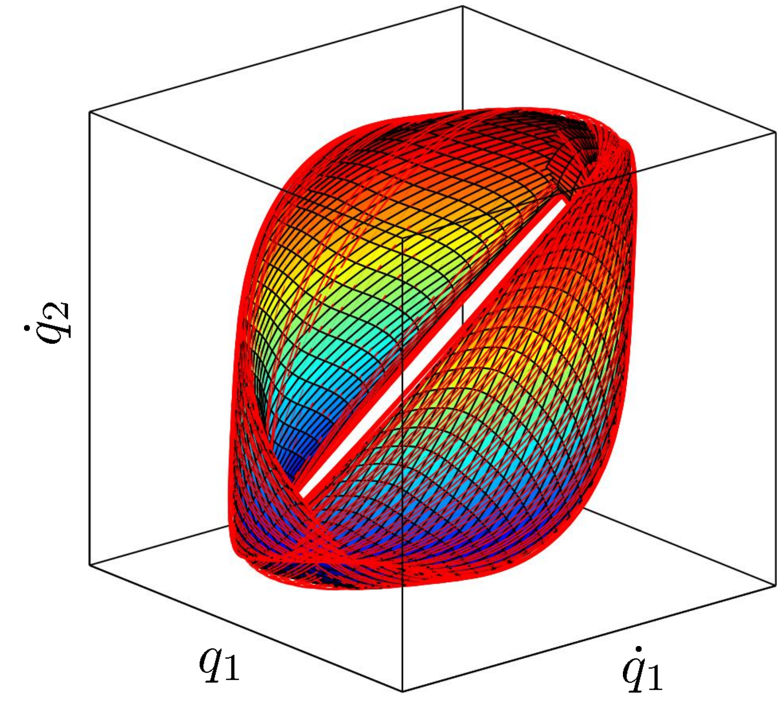
Limitation 1: *Only periodic oscillations* → no quasi-periodic, broadband or chaotic ones

multi-frequency HB ansatz

$$\boldsymbol{q}_{\mathrm{h}}(t) = \Re\{\sum_{\boldsymbol{n}\in\mathcal{H}} \boldsymbol{Q}_{\boldsymbol{n}} \mathrm{e}^{\mathrm{i}(\boldsymbol{n}\cdot\boldsymbol{\omega})t}\},\ \mathcal{H}\subset\mathbb{Z}^p,\ \boldsymbol{\omega}\in\mathbb{R}^p$$

Extensions

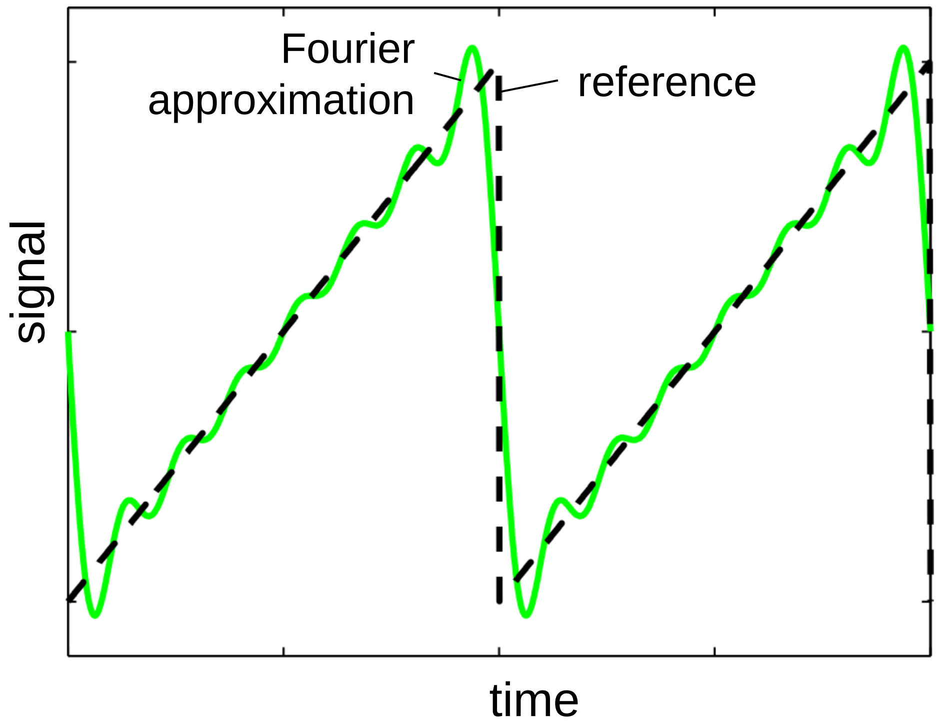
- for quasi-periodic oscillations: multi-frequency HB [Schi06,Krac16], adjusted HB [Gusk12])
- for broadband or chaotic oscillations: ongoing research



quasi-periodic oscillations on an invariant torus

Harmonic Balance has to main limitations

Limitation 2: *Harmonic base functions* → Gibbs phenomenon near discontinuities



examples:
- impacts
- stick-slip transitions

Extensions

- enrichment by non-smooth base functions (wavelet balance [JONE15,KIM03])
- numerical integration of non-smooth states (mixed-Shooting-HB [SCHR16])

Alternatives to Harmonic Balance

Periodic oscillations

- Fourier-based alternatives
  - trigonometric collocation
  - time spectral method
- Shooting methods
- …

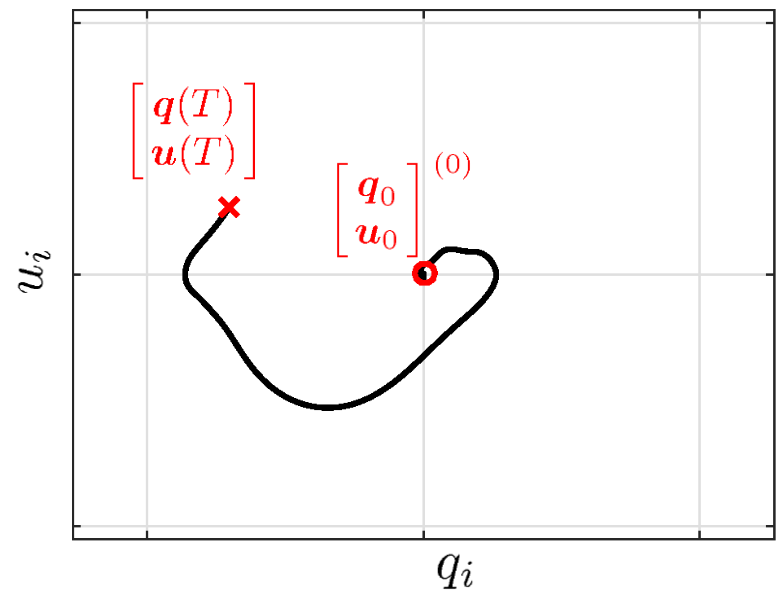
Shooting problem

solve $\boldsymbol{R}(\boldsymbol{x}) = \begin{bmatrix} \boldsymbol{q}(T) - \boldsymbol{q}_0 \\ \boldsymbol{u}(T) - \boldsymbol{u}_0 \end{bmatrix} = \boldsymbol{0}$

with respect to $\boldsymbol{x} = \begin{bmatrix} \boldsymbol{q}_0^{\mathrm{T}} & \boldsymbol{u}_0^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}$

with $\boldsymbol{R}, \boldsymbol{x} \in \mathbb{R}^{2n\times 1}$

where $\boldsymbol{q}(T), \boldsymbol{u}(T)$ are determined by forward numerical integration



Alternatives to Harmonic Balance

Periodic oscillations

- Fourier-based alternatives
  - trigonometric collocation
  - time spectral method
- Shooting methods
- …

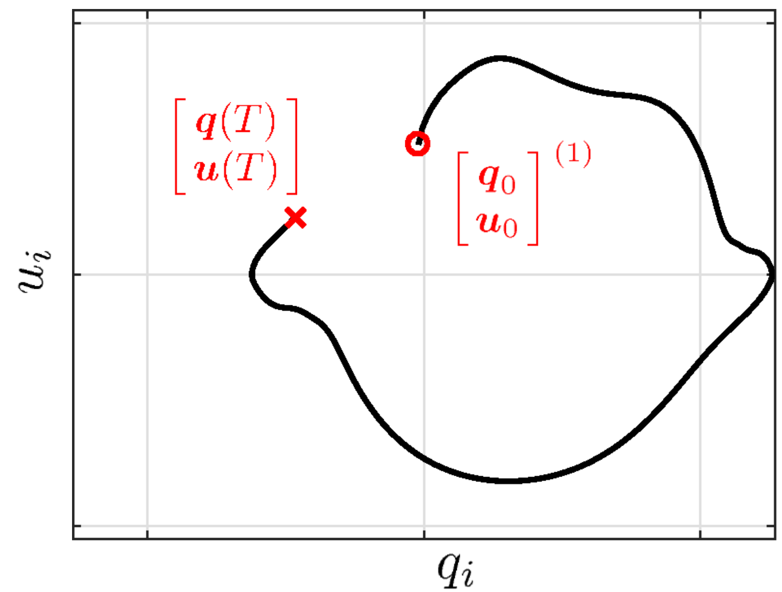
Shooting problem

solve $\boldsymbol{R}(\boldsymbol{x}) = \begin{bmatrix} \boldsymbol{q}(T) - \boldsymbol{q}_0 \\ \boldsymbol{u}(T) - \boldsymbol{u}_0 \end{bmatrix} = \boldsymbol{0}$

with respect to $\boldsymbol{x} = \begin{bmatrix} \boldsymbol{q}_0^{\mathrm{T}} & \boldsymbol{u}_0^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}$

with $\boldsymbol{R}, \boldsymbol{x} \in \mathbb{R}^{2n \times 1}$

where $\boldsymbol{q}(T), \boldsymbol{u}(T)$ are determined by forward numerical integration



Alternatives to Harmonic Balance

Periodic oscillations

- Fourier-based alternatives
  - trigonometric collocation
  - time spectral method
- Shooting methods
- …

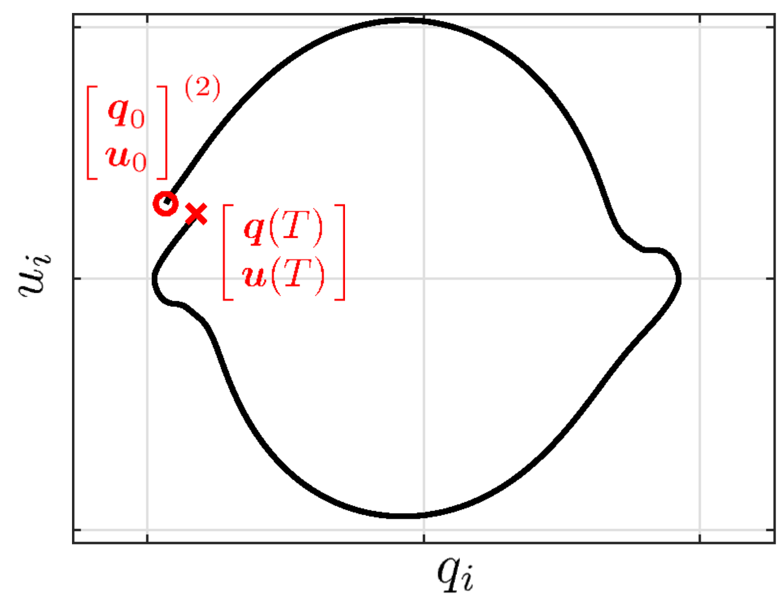
Shooting problem

solve $\boldsymbol{R}(\boldsymbol{x}) = \begin{bmatrix} \boldsymbol{q}(T) - \boldsymbol{q}_0 \\ \boldsymbol{u}(T) - \boldsymbol{u}_0 \end{bmatrix} = \mathbf{0}$

with respect to $\boldsymbol{x} = \begin{bmatrix} \boldsymbol{q}_0^{\mathrm{T}} & \boldsymbol{u}_0^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}$

with $\boldsymbol{R}, \boldsymbol{x} \in \mathbb{R}^{2n\times 1}$

where $\boldsymbol{q}(T), \boldsymbol{u}(T)$ are determined by forward numerical integration



Institut für Luftfahrtantriebe Universität Stuttgart

Alternatives to Harmonic Balance

Periodic oscillations

- Fourier-based alternatives
  - trigonometric collocation
  - time spectral method
- Shooting methods
- …

General oscillations:
Forward numerical integration

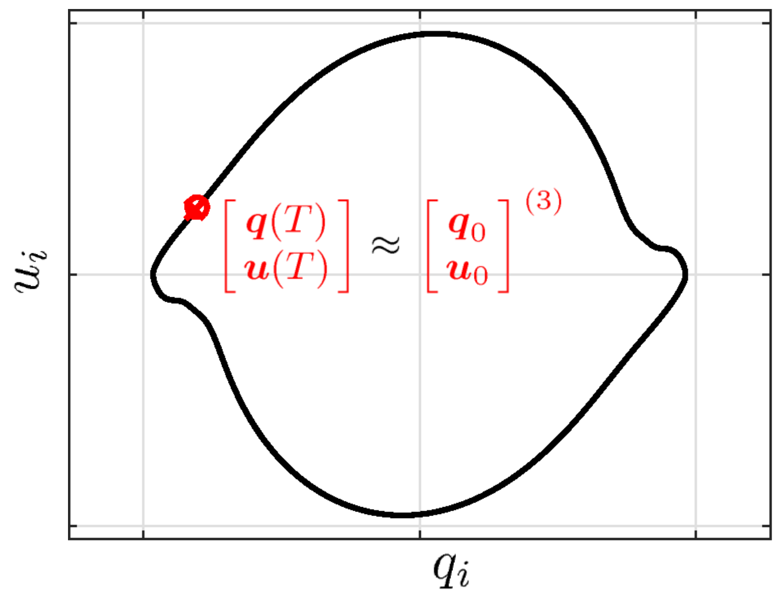
Shooting problem

solve $\boldsymbol{R}(\boldsymbol{x}) = \begin{bmatrix} \boldsymbol{q}(T) - \boldsymbol{q}_0 \\ \boldsymbol{u}(T) - \boldsymbol{u}_0 \end{bmatrix} = \boldsymbol{0}$

with respect to $\boldsymbol{x} = \begin{bmatrix} \boldsymbol{q}_0^{\mathrm{T}} & \boldsymbol{u}_0^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}$

with $\boldsymbol{R}, \boldsymbol{x} \in \mathbb{R}^{2n\times 1}$

where $\boldsymbol{q}(T), \boldsymbol{u}(T)$ are determined by forward numerical integration



Ongoing research on Harmonic Balance

- robust and efficient methods for branching behavior
- reliable and efficient methods for stability assessment (local and global!)
- multi-physical problems
- improvements for non-smooth problems

Outline of talk

- introductory example: Duffing oscillator
- generalization to nonlinear mechanical systems
- implementation in a simple Matlab tool **NLvib,** application examples
- limitations, ongoing research
- summary, questions, discussion

Summary

Harmonic Balance is a numerical method for the efficient computation of periodic solutions of nonlinear ordinary differential equations.

It can be interpreted as Galerkin method. It yields an algebraic equation system, which can be solved using e.g. Newton-like methods and numerical path continuation.

A relatively simple Harmonic Balance code is implemented in the Matlab tool **NLvib**, available via www.ila.uni-stuttgart.de/nlvib.

Further reading

Harmonic Balance, Alternating Frequency-Time Scheme

[CAME89] Cameron, T. M.; Griffin, J. H.: An Alternating Frequency/Time Domain Method for Calculating the Steady-State Response of Nonlinear Dynamic Systems. Journal of Applied Mechanics 56(1):149-154, 1989.

[CARD94] Cardona, A.; Coune, T.; Lerusse, A.; Geradin, M.: A Multiharmonic Method for Non-Linear Vibration Analysis. Int. J. Numer. Meth. Engng 37(9):1593-1608, 1994.

[URAB65] Urabe, M.: Galerkin's Procedure for Nonlinear Periodic Systems. Archive for Rational Mechanics and Analysis 20(2):120-152, 1965.

Treatment of particular nonlinearities, Asymptotic Numerical Method

[COCH09] Cochelin, B.; Vergez, C.: A High Order Purely Frequency-Based Harmonic Balance Formulation for Continuation of Periodic Solutions. Journal of Sound and Vibration 324(1-2):243-262, 2009.

[KRAC13] Krack, M.; Panning-von Scheidt, L.; Wallaschek, J.: A High-Order Harmonic Balance Method for Systems With Distinct States. Journal of Sound and Vibration 332(21):5476-5488, 2013.

[PETR03] Petrov, E. P.; Ewins, D. J.: Analytical Formulation of Friction Interface Elements for Analysis of Nonlinear Multi-Harmonic Vibrations of Bladed Disks. Journal of Turbomachinery 125(2):364-371, 2003.

Computation of quasi-periodic oscillations with Harmonic Balance

[GUSK12] Guskov, M.; Thouverez, F.: Harmonic Balance-Based Approach for Quasi-Periodic Motions and Stability Analysis. Journal of Vibration and Acoustics 134(3):11pp, 2012.

[KRAC16] Krack, M.; Panning-von Scheidt, L.; Wallaschek, J.: On the interaction of multiple traveling wave modes in the flutter vibrations of friction-damped tuned bladed disks. J. Eng. Gas Turbines Power 139(4):9pp, 2016.

[SCHI06] Schilder, F.; Vogt, W.; Schreiber, S.; Osinga, H. M.: Fourier Methods for Quasi-Periodic Oscillations. Int. J. Numer. Meth. Engng 67(5):629-671, 2006.

Computation of non-smooth periodic oscillations

[JONE15] Jones, S.; Legrand, M.: Forced vibrations of a turbine blade undergoing regularized unilateral contact conditions through the wavelet balance method. Int. J. Numer. Meth. Engng 101(5):351-374, 2015.

[KIM03] Kim, W.-J; Perkins, N. C.: Harmonic Balance/Galerkin Method for Non-Smooth Dynamic Systems. Journal of Sound and Vibration 261(2):213-224, 2003.

[SCHR16] Schreyer, F.; Leine, R. I.: A Mixed Shooting – Harmonic Balance Method for unilaterally constrained mechanical systems. Archive of Mechanical Engineering 63:298-313, 2016.

Questions?

Topics for discussion?

Appendix: Overview of basic examples in **NLvib**

name	n	HB	Shooting	FRF	NMA
01_Duffing	1	o	-	o	-
02_twoDOFoscillator_cubicSpring	2	o	o	o	-
03_twoDOFoscillator_unilateralSpring	2	o	o	o	-
04_twoDOFoscillator_cubicSpring_NM	2	o	o	-	o
05_twoDOFoscillator_tanhDryFriction_NM	2	o	o	-	o
06_twoSprings_geometricNonlinearity	2	o	-	o	o
07_multiDOFoscillator_multipleNonlinearities	3	o	-	o	-
08_beam_tanhDryFriction	16	o	o	o	-
09_beam_cubicSpring_NM	38	o	-	-	o

n: number of degrees of freedom
HB: Harmonic Balance
FRF: nonlinear frequency response analysis
NMA: nonlinear modal analysis

Run times depend on your computing environment, but should not exceed a minute per example for a standard computer (2017).

Appendix: Definition of Mechanical Systems in **NLvib**

For simplicity, the code comes with harmonic forcing. Note that you can easily generalize the external force to multiple harmonics (which is actually a good exercise to get familiar with the code).

Equations of motion

$$\boldsymbol{M}\ddot{\boldsymbol{q}} + \boldsymbol{D}\dot{\boldsymbol{q}} + \boldsymbol{K}\boldsymbol{q} + \boldsymbol{f}_{\mathrm{nl}} = \Re\{\boldsymbol{F}_{\mathrm{ex},1}\mathrm{e}^{\mathrm{i}\Omega t}\}$$

Local nonlinear elements

$$\boldsymbol{f}_{\mathrm{nl}} = \sum_e \underbrace{\boldsymbol{w}_e}_{\text{force direction}} \, f_{\mathrm{nl},e}\left(\boldsymbol{w}_e^{\mathrm{T}}\boldsymbol{q},\, \boldsymbol{w}_e^{\mathrm{T}}\boldsymbol{u}\right)$$

Matlab syntax

specific properties and values

```
% Define properties
M = ...    % n x n matrix
D = ...    % n x n matrix
K = ...    % n x n matrix
Fex1 = ...% n x 1 vector
w1 = ...   % n x 1 vector
...

% Define nonlinear elements
nonlinear_elements{1} = struct('type',...,'force_direction',w1,['p1',v1,'p2',v2,...]);
nonlinear_elements{2} = ...

% Define mechanical system
mySystem = MechanicalSystem(M,D,K,nonlinear_elements,Fex1);
```

Appendix: Local nonlinear elements in NLvib

Some of the already available local nonlinear elements

$$f_{\mathrm{nl},e}\left(q_{\mathrm{nl}}, u_{\mathrm{nl}}\right) = \begin{cases} \text{nonlinear\_elements}\{e\}.\text{stiffness } q_{\mathrm{nl}}^3 & \text{`cubicSpring'} \\ \text{nonlinear\_elements}\{e\}.\text{damping } q_{\mathrm{nl}}^2 u_{\mathrm{nl}} & \text{`quadraticDamper'} \\ \text{nonlinear\_elements}\{e\}.\text{friction\_limit\_force } \tanh\left(\frac{u_{\mathrm{nl}}}{\text{nonlinear\_elements}\{e\}.\text{eps}}\right) & \text{`tanhDryFriction'} \\ \text{nonlinear\_elements}\{e\}.\text{stiffness } \langle q_{\mathrm{nl}} - \text{nonlinear\_elements}\{e\}.\text{gap}\rangle & \text{`unilateralSpring'} \end{cases}$$

(*type* column: 'cubicSpring', 'quadraticDamper', 'tanhDryFriction', 'unilateralSpring')

$$\langle x \rangle = \begin{cases} x & x \geq 0 \\ 0 & x < 0 \end{cases}$$

➢ *You can easily add new nonlinearities analogous to the already available ones.*

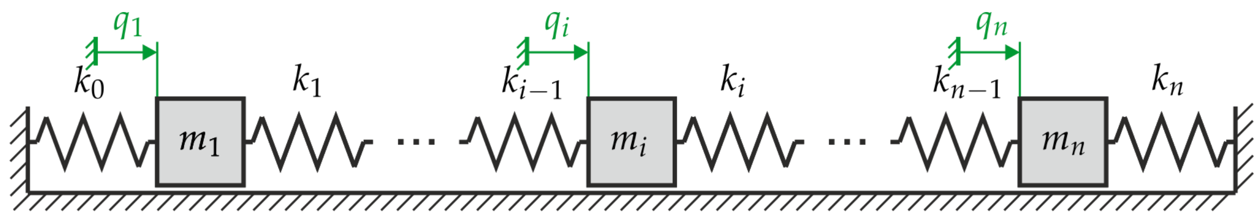
Implementation of nonlinear elements in HB_residual.m (analogous in shooting_residual.m)

```
...
%% Evaluate nonlinear force in time domain
        switch lower(nonlinear_elements{nl}.type)
            case 'cubicspring'
                fnl = nonlinear_elements{nl}.stiffness*qnl.^3;
                dfnl = ...
            case 'mynewnonlinearity'
                fnl = ...
                dfnl = ...
          ...
```

analytical derivative

Tip: When you add a new nonlinearity, run the solver with 'jac' option 'none'. If everything is working properly, you can later accelerate the code by providing (correct!) analytical derivatives. If you encounter convergence problems at that point, your derivatives are wrong.

Appendix: A chain of oscillators in **NLvib**



Structural matrices

$$\boldsymbol{M} = \mathrm{diag}\{m_i\}$$

$$\boldsymbol{K} = \begin{bmatrix} k_0+k_1 & -k_1 & 0 & \cdots & 0 \\ -k_1 & k_1+k_2 & -k_2 & \ddots & \vdots \\ 0 & -k_2 & \ddots & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & -k_{n-1} \\ 0 & \cdots & 0 & -k_{n-1} & k_{n-1}+k_n \end{bmatrix}$$

$\boldsymbol{D}$ analogous to $\boldsymbol{K}$

Matlab syntax

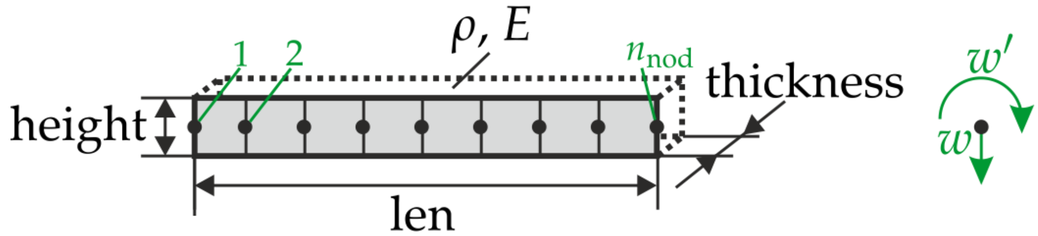
```
% Define properties
mi = ...  % vector with length n
ki = ...  % vector with length n+1
di = ...  % vector with length n+1
Fex1 = ...% n x 1 vector
...

% Define nonlinear elements
nonlinear_elements{1} = ...

% Define chain of oscillators
myChain = ChainOfOscillators(mi,di,ki,nonlinear_elements,Fex1);
```

Appendix: An FE model of an Euler-Bernoulli beam in **NLvib**

Coordinates

$$\boldsymbol{q}_{\mathrm{full}} = \begin{bmatrix} w_1 \\ w_1' \\ \vdots \\ w_{n_{\mathrm{nod}}} \\ w_{n_{\mathrm{nod}}}' \end{bmatrix} = \boldsymbol{L}\boldsymbol{q}$$

matrix containing columns of the identity matrix, so that the generalized coordinates are compatible with the boundary conditions (BCs)



Matlab syntax

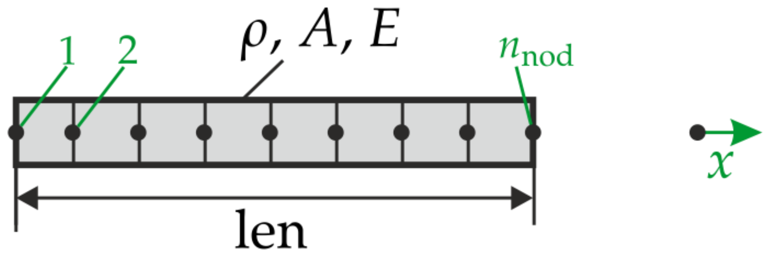
```
% Define properties
...
BCs = 'clamped-free';  % example with clamping on the left and free end on the right;
                       % pinned is also possible; arbitrary combinations are allowed
n_nod = ...            % positive integer

% Define beam (rectangular cross section)
myBeam = FE_EulerBernoulliBeam(len,height,thickness,E,rho,BCs,n_nod);

% Add external forcing (add_forcing works in an additive way)
inode = ...                    % node index
dof = ...                      % degree of freedom specifier ('rot' or 'trans')
Fex1 = ...                     % complex-valued scalar
add_forcing(myBeam,inode,dof,Fex1);

% Add nonlinear attachment (only grounded nonlinear elements for transversal DOF)
inode_nl = ...                 % see above
dof_nl = ...                   % see above
add_nonlinear_attachment(myBeam,inode,dof,type,['p1',v1,'p2',v2,...]);
```

Note that you can add non-grounded elements (as in the general MechanicalSystem case, but you will have to set up the force direction manually.

Appendix: An FE model of an elastic rod in **NLvib**

Coordinates

$$\boldsymbol{q}_{\text{full}} = \begin{bmatrix} x_1 \\ \vdots \\ x_{n_{\text{nod}}} \end{bmatrix} = \boldsymbol{L}\boldsymbol{q}$$



matrix containing columns of the identity matrix, so that the generalized coordinates are compatible with the boundary conditions (BCs)

Matlab syntax

```
% Define properties
...
BCs = 'pinned-free';    % example with pinning on the left and free end on the right;
                        % arbitrary combinations are allowed
n_nod = ...             % positive integer

% Define rod
myRod = FE_ElasticRod(len,A,E,rho,BCs,n_nod);

% Add external forcing (add_forcing works in an additive way)
inode = ...                       % node index
Fex1 = ...                        % complex-valued scalar
add_forcing(myRod,inode,Fex1);

% Add nonlinear attachment (only grounded nonlinear elements)
inode_nl = ...                    % see above
add_nonlinear_attachment(myRod,inode,type,['p1',v1,'p2',v2,...]);
```

Note that you can add non-grounded elements (as in the general MechanicalSystem case, but you will have to set up the force direction manually.

Appendix: A system with polynomial stiffness in NLvib

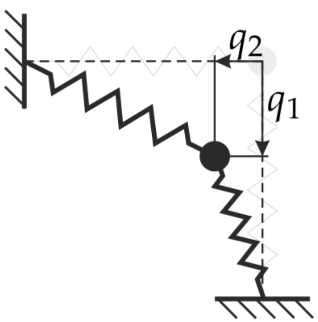
Equations of motion of MechanicalSystem, but with nonlinear force

$$\boldsymbol{f}_{\mathrm{nl}} = \boldsymbol{E}^{\mathrm{T}}\boldsymbol{z} = \sum_k E_{ki} z_k \quad \text{with} \quad z_k = \prod_j q_j^{p_{kj}}$$

Matlab syntax

```
% Define properties
p = ... % Nz x n vector of nonnegative integers
E = ... % Nz x n vector of real-valued coefficients
...

% Define system
myPolyStiffSys = System_with_PolynomialStiffnessNonlinearity(M,D,K,p,E,Fex1);
```

Example: system with geometrical nonlinearity



color scheme E_{ki} z_k

$$\ddot{q}_1 + 2\zeta_1\omega_1\dot{q}_1 + \omega_1^2 q_1 + \frac{3\omega_1^2}{2} q_1^2 + \omega_2^2 q_1 q_2 + \frac{\omega_1^2}{2} q_2^2 + \frac{\omega_1^2+\omega_2^2}{2} q_1^3 + \frac{\omega_1^2+\omega_2^2}{2} q_1 q_2^2 = 0$$

$$\ddot{q}_2 + 2\zeta_2\omega_2\dot{q}_2 + \omega_2^2 q_2 + \frac{\omega_1^2}{2} q_1^2 + \omega_2^2 q_1 q_2 + \frac{3\omega_1^2}{2} q_2^2 + \frac{\omega_1^2+\omega_2^2}{2} q_1^2 q_2 + \frac{\omega_1^2+\omega_2^2}{2} q_2^3 = 0$$

Appendix: Some practical hints on using NLvib

Strongly simplify your problem first and then successively increase complexity!

- Always analyze the linearized problem first.
  - ➢ Do the system matrices have the expected dimensions, symmetries, eigenvalues?
  - ➢ Derive a suitable initial guess for the nonlinear analysis.
  - ➢ Derive reference values for linear scaling (‘Dscale’).
- Always start the nonlinear HB analysis with H=1.
- Then increase H successively until the results converge (do not waste resources by setting it unreasonably high).

Appendix: Some practical hints on using NLvib

What shall I do if I encounter one or more of the following difficulties?

a) *Initial guess not within basin of attraction.*
   - start in ‘more linear’ regime
   - improve the initial guess (e.g. from a suitable linearization or numerical integration)
   - if analytical gradients are used, validate them (or run with ‘jac’ parameter set to ‘none’)

b) *No convergence during continuation.*
   - ensure suitable scaling variables (‘Dscale’ vector) and residual funcations
   - reduce step length parameter ‘ds’
   - ensure numerical path continuation is activated (‘flag’ set to ‘on’ (default))
   - increase AFT scheme sampling rate
   - analytical gradients, cf. above

c) *The computation time is very large.*
   - scaling, cf. above
   - increase step length parameter ‘ds’
   - use (correct!) analytical gradients
   - lower your expectations ☺

Appendix: solve_and_continue in NLvib

Problem statement

solve $\boldsymbol{R}(\boldsymbol{X}) = \boldsymbol{0}$

with respect to $\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x} \\ \lambda \end{bmatrix}$

in the interval $\lambda^{(\mathrm{s})} \leq \lambda \leq \lambda^{(\mathrm{e})}$

Matlab syntax

initial guess *residual function R(X)* *interval limits* *nominal step size*

```
[X,Solinfo,Sol] = solve_and_continue(x0,fun_residual,lam_s,lam_e,ds [,Sopt, ...
                                         fun_postprocess,opt_fsolve]);
```

continuation options structure (optional)

columns are solution points

array of structures returned by postprocessing functions

postprocessing functions F(X) (optional)

fsolve options (optional)

Solinfo.FCtotal	total function evaluation count
Solinfo.ctime	total computation time
per solution step	
Solinfo.iEx	fsolve exit flag
Solinfo.NIT	number of iterations
Solinfo.FC	function count

Appendix: solve_and_continue in NLvib

*Most common continuation options (*Sopt*)*

.flag	*flag whether actual continuation is performed or trivial (sequential) continuation is employed (default: 1)*
.predictor	*tangent or secant predictors can be specified ['tangent'|'secant'] (default: 'tangent')*
.parametrization	*parametrization constraint ['arc_length'|'pseudo_arc_length'|'local'| 'orthogonal'] (default: 'arc_length')*
.dsmin	*minimum step size (default: ds/5)*
.dsmax	*maximum step size (default: ds*5)*
.stepadapt	*flag whether step size should be automatically adjusted (default: 1; recommended if Sopt.flag = 1)*
.stepmax	*maximum number of steps before termination*
.termination_criterion	*cell array of functions (X) returning logic scalar 1 for termination*
.jac	*flag whether analytically provided residual derivatives (Jacobian) should be used (default), or a finite difference approximation should be computed (.jac = 'none')*
.Dscale	*linear scaling to be applied to vector X*

Appendix: solve_and_continue in NLvib

Why should you apply linear scaling?

Example problem $\boldsymbol{R}(\boldsymbol{x}) = \begin{bmatrix} (x_1 - 1)^2 \\ \left(10^7 x_2 - 0.75\right)^2 \end{bmatrix}$ with solution $\boldsymbol{x}_0 = \begin{bmatrix} 1 \\ 0.75 \cdot 10^{-7} \end{bmatrix}$

Rescaled problem $\tilde{\boldsymbol{R}}(\tilde{\boldsymbol{x}}) := \boldsymbol{R}(\overbrace{\boldsymbol{D}_{\text{scale}}\tilde{\boldsymbol{x}}}^{\boldsymbol{x}})$ with solution $\tilde{\boldsymbol{x}}_0 = \begin{bmatrix} 1 \\ 0.75 \end{bmatrix}$

where the scaling matrix $\boldsymbol{D}_{\text{scale}} = \operatorname{diag}\{|\hat{x}_i|\} = \begin{bmatrix} 1 & 0 \\ 0 & 10^{-7} \end{bmatrix}$

attempts to achieve similar orders of magnitude among the new variables $\tilde{\boldsymbol{x}} = \boldsymbol{D}_{\text{scale}}^{-1}\boldsymbol{x}$

This suggest that one should scale with the (not a priori known) solution. In practice, one can achieve good results with typical values for the respective variable, derived e.g. from a solution of a linearized problem.

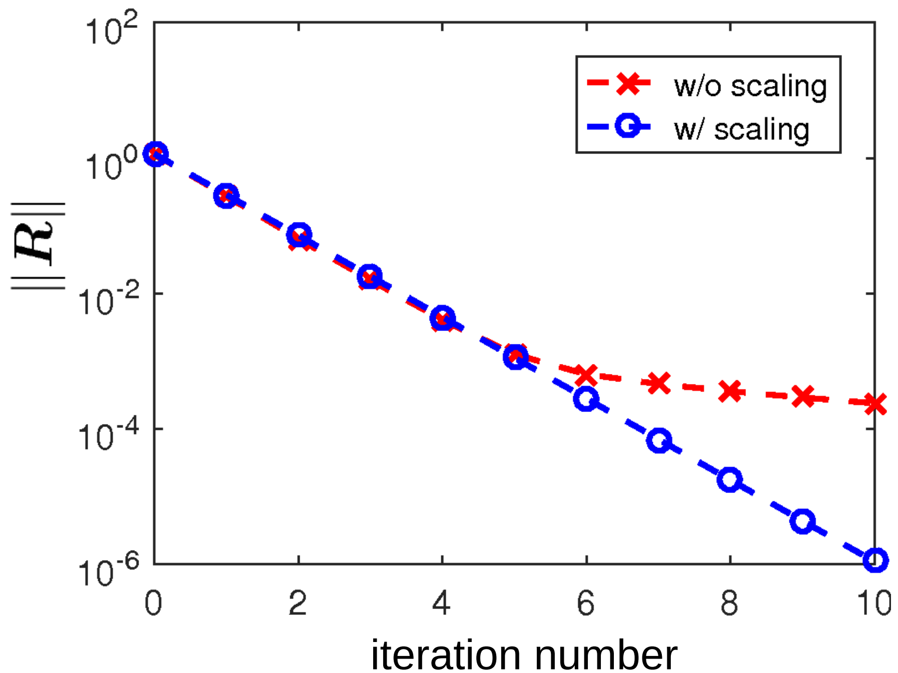
Appendix: solve_and_continue in NLvib

With the suggested scaling, the condition number of the Jacobian change:

$$\mathrm{cond}\left(\frac{\mathrm{d}\boldsymbol{R}}{\mathrm{d}\boldsymbol{x}}\right) \sim 10^7 \quad \Rightarrow \quad \mathrm{cond}\left(\frac{\mathrm{d}\tilde{\boldsymbol{R}}}{\mathrm{d}\tilde{\boldsymbol{x}}}\right) \sim 10^0$$

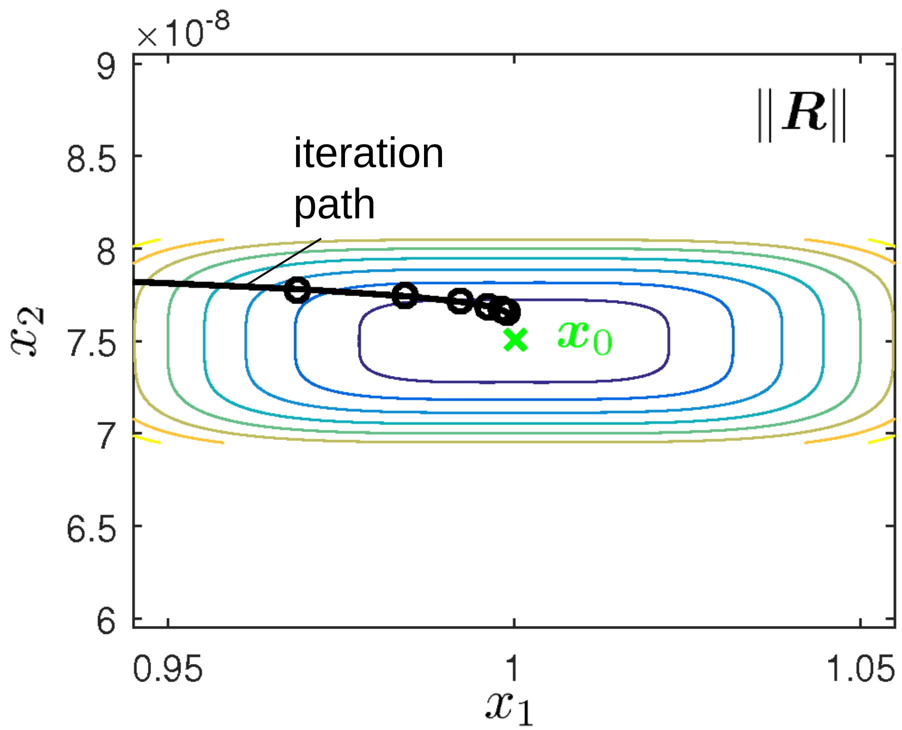
A high condition number is likely to cause convergence problems within the Newton method in the presence of numerical imprecisions. To illustrate this, we apply Matlab's *fsolve* to both problems. Numerical imprecisions are introduced by letting *fsolve* approximate the Jacobian using finite differences.

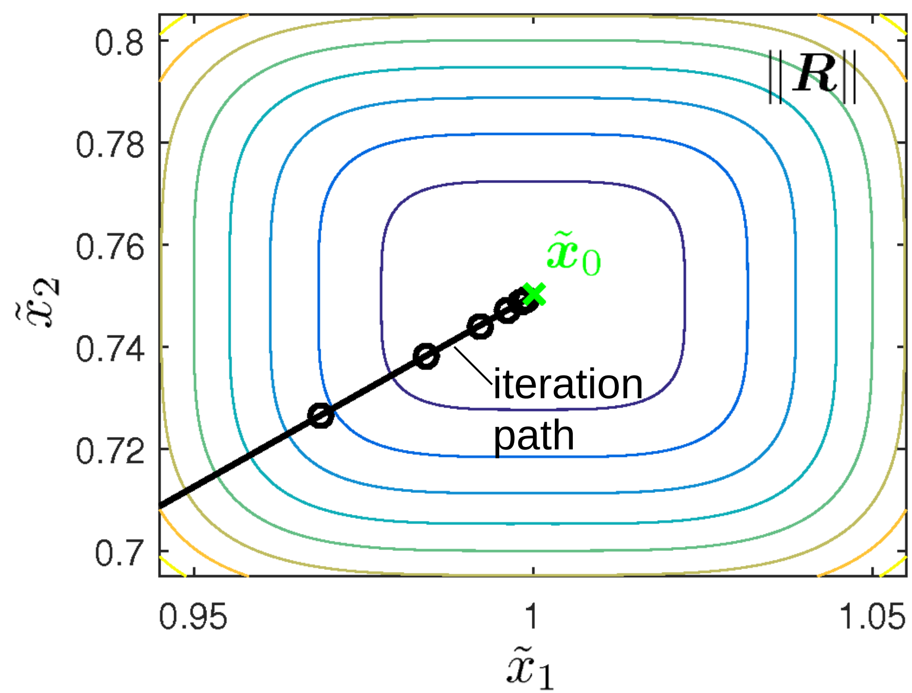
Without scaling, the solver has apparent difficulties.



Appendix: solve_and_continue in NLvib

From the viewpoint of the solver, the problem is stretched, making it hard to numerically approximate the true solution in the unscaled variable space.





Appendix: Frequency response analysis with NLvib

Harmonic Balance formulation

solve $\boldsymbol{R}(\boldsymbol{X}) = \begin{bmatrix} \boldsymbol{R}_0 \\ \Re\{\boldsymbol{R}_1\} \\ \Im\{\boldsymbol{R}_1\} \\ \vdots \\ \Im\{\boldsymbol{R}_H\} \end{bmatrix} = \boldsymbol{0}$

where $\boldsymbol{R}_k = \left[-(k\omega)^2\boldsymbol{M} + \mathrm{i}k\omega\boldsymbol{D} + \boldsymbol{K}\right]\boldsymbol{Q}_k + \boldsymbol{F}_{\mathrm{nl},k} - \boldsymbol{F}_{\mathrm{ex},k}$

with respect to $\boldsymbol{X} = \left[\boldsymbol{Q}_0^{\mathrm{T}} \quad \Re\{\boldsymbol{Q}_1^{\mathrm{T}}\} \quad \Im\{\boldsymbol{Q}_1^{\mathrm{T}}\} \quad \ldots \quad \Im\{\boldsymbol{Q}_H^{\mathrm{T}}\} \quad \Omega\right]^{\mathrm{T}}$

in the interval $\Omega^{(\mathrm{s})} \leq \Omega \leq \Omega^{(\mathrm{e})}$

Appendix: Frequency response analysis with NLvib

Shooting formulation

solve $\boldsymbol{R}(\boldsymbol{X}) = \begin{bmatrix} \left(\boldsymbol{q}(T) - \boldsymbol{q}_0\right)\frac{1}{q_{\mathrm{scl}}} \\ \left(\boldsymbol{u}(T) - \boldsymbol{u}_0\right)\frac{1}{q_{\mathrm{scl}}} \end{bmatrix} = \boldsymbol{0}$

with respect to $\boldsymbol{X} = \begin{bmatrix} \boldsymbol{q}_0^{\mathrm{T}} & \frac{\boldsymbol{u}_0^{\mathrm{T}}}{\Omega} & \Omega \end{bmatrix}^{\mathrm{T}}$

in the interval $\Omega^{(\mathrm{s})} \leq \Omega \leq \Omega^{(\mathrm{e})}$

where $\boldsymbol{q}(T), \boldsymbol{u}(T)$ are determined by forward numerical integration

q_{scl} *positive real-valued scalar*

Rationale behind scaling of residual: achieve similar orders of magnitude for quite different vibration levels. Otherwise the solver might misinterpret e.g. a small value as a converged residual.

Appendix: Nonlinear modal analysis with **NLvib**

Harmonic Balance formulation

f_{scl} *positive real-valued scalar*

solve
$$\boldsymbol{R}(\boldsymbol{X}) = \begin{bmatrix} \boldsymbol{R}_0 \, \frac{f_{\mathrm{scl}}}{a} \\ \Re\{\boldsymbol{R}_1\} \, \frac{f_{\mathrm{scl}}}{a} \\ \Im\{\boldsymbol{R}_1\} \, \frac{f_{\mathrm{scl}}}{a} \\ \vdots \\ \Im\{\boldsymbol{R}_H\} \, \frac{f_{\mathrm{scl}}}{a} \\ \Re\{\sum\limits_{k=0}^{H} \boldsymbol{Q}_k^{\mathrm{H}} \boldsymbol{M} \boldsymbol{Q}_k\}/a^2 - 1 \\ \dot{q}_{i_{\mathrm{norm}}}(0)/(\omega a) \end{bmatrix} = \boldsymbol{0}$$

amplitude normalization

phase normalization

where $\boldsymbol{R}_k = \left[-(k\omega)^2\boldsymbol{M} + \mathrm{i}k\omega\left(\boldsymbol{D} - 2\delta\omega\boldsymbol{M}\right) + \boldsymbol{K}\right]\boldsymbol{Q}_k + \boldsymbol{F}_{\mathrm{nl},k}$

with respect to $\boldsymbol{X} = \left[\frac{\boldsymbol{Q}_0^{\mathrm{T}}}{a} \quad \Re\{\frac{\boldsymbol{Q}_1^{\mathrm{T}}}{a}\} \quad \Im\{\frac{\boldsymbol{Q}_1^{\mathrm{T}}}{a}\} \quad \ldots \quad \Im\{\frac{\boldsymbol{Q}_H^{\mathrm{T}}}{a}\} \quad \omega \quad \delta \quad \log_{10} a\right]^{\mathrm{T}}$

in the interval $\log_{10} a^{(\mathrm{s})} \le \log_{10} a \le \log_{10} a^{(\mathrm{e})}$

Rationale behind scaling of residual: achieve similar orders of magnitude of typical values. Otherwise the dynamic force equilibrium or the normalization conditions would have unreasonably strong weight, which could have a negative influence the convergence of the solver.

Appendix: Nonlinear modal analysis with NLvib

Shooting formulation

solve $\boldsymbol{R}(\boldsymbol{X}) = \begin{bmatrix} \left(\boldsymbol{q}(T) - \boldsymbol{q}_0\right) \frac{1}{q_{\mathrm{scl}}} \\ \left(\boldsymbol{u}(T) - \boldsymbol{u}_0\right) \frac{1}{q_{\mathrm{scl}}} \end{bmatrix} = \boldsymbol{0}$

with respect to $\boldsymbol{X} = \begin{bmatrix} \frac{\boldsymbol{q}_{0-}^{\mathrm{T}}}{a} & \frac{\boldsymbol{u}_{0-}^{\mathrm{T}}}{\omega a} & \omega & D & \log_{10} a \end{bmatrix}^{\mathrm{T}}$

in the interval $\log_{10} a^{(\mathrm{s})} \leq \log_{10} a \leq \log_{10} a^{(\mathrm{e})}$

where $\boldsymbol{q}_0, \boldsymbol{u}_0$ are $\boldsymbol{q}_{0-}, \boldsymbol{u}_{0-}$ only with
$q_{0,i_{\mathrm{norm}}} = a$ — *amplitude normalization*
$u_{0,i_{\mathrm{norm}}} = 0$ — *phase normalization*

where $\boldsymbol{q}(T), \boldsymbol{u}(T)$ are determined by forward numerical integration

q_{scl} *positive real-valued scalar*

Rationale behind scaling of residual: achieve similar orders of magnitude for quite different vibration levels. Otherwise the solver might misinterpret e.g. a small value as a converged residual.

Appendix: Numerical time step integration (for details see textbooks e.g. [GER15])

The purpose of numerical time step integration is to approximate the solution of an ordinary differential equation, from given initial values $\boldsymbol{q}(t_0), \boldsymbol{u}(t_0)$, up to a given end time. Most of the methods are based on finite difference approximations with respect to time and yield a quadrature formula governing the values of the unknown states at the next time level $t_{\ell+1}$.

$$\begin{aligned}
\boldsymbol{q}(t_{\ell+1}) &= \boldsymbol{g}_{\boldsymbol{q}}(t_{\ell+1}, \boldsymbol{q}(t_{\ell+1}), \boldsymbol{u}(t_{\ell+1}), \boldsymbol{q}(t_{\ell}), \ldots) \\
\boldsymbol{u}(t_{\ell+1}) &= \boldsymbol{g}_{\boldsymbol{u}}(t_{\ell+1}, \boldsymbol{q}(t_{\ell+1}), \boldsymbol{u}(t_{\ell+1}), \boldsymbol{q}(t_{\ell}), \ldots)
\end{aligned}$$

Some methods directly deal with the second-order differential equation of motion (Newmark, Hilbert-Hughes-Taylor), other methods require the re-formulation to first-order.

For explicit methods, the quadrature formula can be brought into an form where the unknown states at the next time level are determined by simplify evaluating a function once. Implicit methods require one to solve an algebraic equation system at each time level.

The method is (numerically) stable if the states remain finite for a finite step size. There are conditionally stable methods, which are only stable for sufficiently small step size, and unconditionally stable methods.

The approximation error depends, among others, on the quadrature formula and the step size. Important accuracy measures are numerical damping (non-physical decrease of energy), numerical dispersion (depending of error on contributing oscillation frequencies). Some numerical damping of higher frequencies can be desirable, particularly if their dynamics is not correctly modeled due to e.g. finite spatial discretization.

[GER15] Geradin, M.; Rixen, D.J.: Mechanical Vibrations, Wiley, 2015

Appendix: Newmark method implemented in NLvib

Equation of motion evaluated at end of a time level

$$\boldsymbol{M}\dot{\boldsymbol{u}}^{\mathrm{E}} + \boldsymbol{D}\boldsymbol{u}^{\mathrm{E}} + \boldsymbol{K}\boldsymbol{q}^{\mathrm{E}} + \boldsymbol{f}_{\mathrm{nl}}^{\mathrm{E}} - \boldsymbol{f}_{\mathrm{ex}}^{\mathrm{E}} = \boldsymbol{0} \quad (1)$$

$t^{\mathrm{E}} = t_{\ell+1}, t^{\mathrm{S}} = t_{\ell},$
$\Delta t = t^{\mathrm{E}} - t^{\mathrm{S}}, \boldsymbol{q}^{\mathrm{E}} = \boldsymbol{q}(t^{\mathrm{E}}), \dots$

Time discretization *(constant average acceleration Newmark scheme, see e.g. [GER15])*

$$
\begin{aligned}
\boldsymbol{u}^{\mathrm{E}} &= \boldsymbol{u}^{\mathrm{S}} + \frac{\dot{\boldsymbol{u}}^{\mathrm{S}} + \dot{\boldsymbol{u}}^{\mathrm{E}}}{2}\Delta t && (2)\\
\boldsymbol{q}^{\mathrm{E}} &= \boldsymbol{q}^{\mathrm{S}} + \frac{\boldsymbol{u}^{\mathrm{S}} + \boldsymbol{u}^{\mathrm{E}}}{2}\Delta t && (3)\\
\Rightarrow \quad \dot{\boldsymbol{u}}^{\mathrm{E}} &= \frac{4}{\Delta t^2}\left(\boldsymbol{q}^{\mathrm{E}} - \boldsymbol{q}^{\mathrm{S}}\right) - \frac{4}{\Delta t}\boldsymbol{u}^{\mathrm{S}} - \dot{\boldsymbol{u}}^{\mathrm{S}} && (4)\\
\Rightarrow \quad \boldsymbol{u}^{\mathrm{E}} &= \frac{2}{\Delta t}\left(\boldsymbol{q}^{\mathrm{E}} - \boldsymbol{q}^{\mathrm{S}}\right) - \boldsymbol{u}^{\mathrm{S}} && (5)
\end{aligned}
$$

Note that in the actual implementation, we introduce a time normalization, see next slide.

Substitution of (4) and (5) into (1) gives an implicit equation in displacements at end of time step,

$$\underbrace{\left(\frac{4}{\Delta t^2}\boldsymbol{M} + \frac{2}{\Delta t}\boldsymbol{D} + \boldsymbol{K}\right)}_{\boldsymbol{S}}\boldsymbol{q}^{\mathrm{E}} \quad + \quad \boldsymbol{f}_{\mathrm{nl}}\left(\boldsymbol{q}^{\mathrm{E}}, \boldsymbol{u}^{\mathrm{E}}\left(\boldsymbol{q}^{\mathrm{E}}\right)\right) =$$

$$\underbrace{\boldsymbol{f}_{\mathrm{ex}}\left(t^{\mathrm{E}}\right) + \boldsymbol{M}\left(\frac{4}{\Delta t^2}\boldsymbol{q}^{\mathrm{S}} + \frac{4}{\Delta t}\boldsymbol{u}^{\mathrm{S}} + \dot{\boldsymbol{u}}^{\mathrm{S}}\right) + \boldsymbol{D}\left(\frac{2}{\Delta t}\boldsymbol{q}^{\mathrm{S}} + \boldsymbol{u}^{\mathrm{S}}\right)}_{\boldsymbol{b}}$$

which is solved using Newton iterations with Cholesky factorization of the Jacobian.

[GER15] Geradin, M.; Rixen, D.J.: Mechanical Vibrations, Wiley, 2015

Appendix: Time normalization (for implementation of Newmark method) in NLvib

Normalized time $\tau := \Omega t,\ \mathrm{d}\tau = \Omega \mathrm{d}t$

Reformulation of time derivatives

$$\begin{aligned} \boldsymbol{u} &= \dot{\boldsymbol{q}} = \frac{\mathrm{d}\boldsymbol{q}}{\mathrm{d}t} = \underbrace{\frac{\mathrm{d}\boldsymbol{q}}{\mathrm{d}\tau}}_{\boldsymbol{q}'} \underbrace{\frac{\mathrm{d}\tau}{\mathrm{d}t}}_{\Omega} = \Omega \boldsymbol{q}' \\ \dot{\boldsymbol{u}} &= \ddot{\boldsymbol{q}} = \ldots = \Omega^2 \boldsymbol{q}'' \end{aligned}$$

Equations of motion in non-normalized and normalized time

$$\begin{aligned} \boldsymbol{M}\ddot{\boldsymbol{q}} + \boldsymbol{D}\dot{\boldsymbol{q}} + \boldsymbol{K}\boldsymbol{q} + \boldsymbol{f}_{\mathrm{nl}}(\boldsymbol{q}, \dot{\boldsymbol{q}}) &= \boldsymbol{f}_{\mathrm{ex}}(t) \\ \underbrace{\boldsymbol{M}\Omega^2}_{\tilde{\boldsymbol{M}}}\ \boldsymbol{q}'' + \underbrace{\boldsymbol{D}\Omega}_{\tilde{\boldsymbol{D}}}\ \boldsymbol{q}' + \boldsymbol{K}\boldsymbol{q} + \boldsymbol{f}_{\mathrm{nl}}(\boldsymbol{q}, \Omega\boldsymbol{q}') &= \boldsymbol{f}_{\mathrm{ex}}(\tau) \end{aligned}$$

Appendix: An almost foolproof approach to analytical gradients (as used in **NLvib**)

```
function [R,dR] = my_function(X,param1,param2)
% Define auxiliary variables from input variables X
x1 = X(1);
x2 = X(2);
Om = X(3);

% Initialize derivative of auxiliary variables ('Seeding')
dX = eye(length(X));
dx1 = dX(1,:);
dx2 = dX(2,:);
dOm = dX(3,:);

% Operate on auxiliary variables, determine derivatives in each step using elementary
calculus
z = x1*Om^2;
dz = dx1*Om^2 + x1*2*Om*dOm;
R = z/x2 - x2;
dR = dz/x2 - z/x2^2*dx2 - dx2;
```