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:** \( q(t) \approx q_h(t) = Q_c \cos(\Omega t) + Q_s \sin(\Omega t) \)

(only one harmonic here)
Duffing oscillator: Harmonic Balance

Time derivatives of ansatz:
\[ q_h = +Q_c \cos(\Omega t) + Q_s \sin(\Omega t) \]
\[ \dot{q}_h = -Q_c\Omega \sin(\Omega t) + Q_s\Omega \cos(\Omega t) \]
\[ \ddot{q}_h = -Q_c\Omega^2 \cos(\Omega t) - Q_s\Omega^2 \sin(\Omega t) \]

Expansion of nonlinear term
\[ q_h^3 = (Q_c \cos(\Omega t) + Q_s \sin(\Omega t))^3 \]
\[ = Q_c^3 \cos^3(\Omega t) + 3Q_c^2Q_s \cos^2(\Omega t) \sin(\Omega t) + 3Q_cQ_s^2 \cos(\Omega t) \sin^2(\Omega t) + Q_s^3 \sin^3(\Omega t) \]
\[ = \frac{3}{4} \left( Q_c^3 + Q_cQ_s^2 \right) \cos(\Omega t) + \frac{3}{4} \left( Q_s^3 + Q_sQ_c^2 \right) \sin(\Omega t) + (\ldots) \cos(3\Omega t) + (\ldots) \sin(3\Omega t) \]

Substitute into Duffing equation and collect harmonics
\[ \left[ (1 - \Omega^2) Q_c + \delta\Omega Q_s + \frac{3}{4} \gamma \left( Q_c^3 + Q_cQ_s^2 \right) - P \right] \cos(\Omega t) \]
\[ + \left[ (1 - \Omega^2) Q_s - \delta\Omega Q_c + \frac{3}{4} \gamma \left( Q_s^3 + Q_sQ_c^2 \right) \right] \sin(\Omega t) + [\ldots] \cos(3\Omega t) + [\ldots] \sin(3\Omega t) = 0 \]

We neglect harmonics with index higher than the ansatz (>1) and balance the harmonics:
\[
\begin{align*}
R_c &:= (1 - \Omega^2) Q_c + \delta\Omega Q_s + \frac{3}{4} \gamma \left( Q_c^3 + Q_cQ_s^2 \right) - P = 0 \\
R_s &:= (1 - \Omega^2) Q_s - \delta\Omega Q_c + \frac{3}{4} \gamma \left( Q_s^3 + Q_sQ_c^2 \right) = 0
\end{align*}
\]

With trigonometric identities
\[
\begin{align*}
\cos^3 x &= \frac{3}{4} \cos x + \frac{1}{4} \cos 3x \\
\cos^2 x \sin x &= \frac{1}{4} \sin x + \frac{1}{4} \sin 3x \\
\cos x \sin^2 x &= \frac{1}{4} \cos x - \frac{1}{4} \cos 3x \\
\sin^3 x &= \frac{3}{4} \sin x - \frac{1}{4} \sin 3x
\end{align*}
\]
Duffing oscillator: Frequency response

*Frequency response:* We are interested in how the solution evolves with $\Omega$.

We can formulate this as a **continuation problem**, with $\Omega$ as a **free parameter**:

\[ \mathbf{R}(X) = \begin{bmatrix} R_c \\ R_s \end{bmatrix} = 0 \]

with respect to \[ X = \begin{bmatrix} Q_c \\ Q_s \\ \Omega \end{bmatrix} \]

with \[ R \in \mathbb{R}^2, \ X \in \mathbb{R}^3 \]

in the interval \[ \Omega^{(s)} \leq \Omega \leq \Omega^{(e)} \]

- 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

\[ Q_c = a \cos \theta \]
\[ Q_s = a \sin \theta \]
\[ Q_c^2 + Q_s^2 = a^2 \]

Substitution into algebraic equations

\[
(1 - \Omega^2) a \cos \theta + \delta \Omega a \sin \theta + \frac{3}{4} \gamma \left( a^3 \cos^3 \theta + a^3 \cos \theta \sin^2 \theta \right) = P \quad (1)
\]
\[
(1 - \Omega^2) a \sin \theta - \delta \Omega a \cos \theta + \frac{3}{4} \gamma \left( a^3 \sin^3 \theta + a^3 \cos^2 \theta \sin \theta \right) = 0 \quad (2)
\]

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

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

It is easier to solve Eq. (5) for \( \Omega \):

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

We can have zero, one or two real-valued solutions \( \Omega_{1,2}^2 \).
Duffing oscillator: Frequency response

\[ P = 0.02 \]
Duffing oscillator: Frequency response

\[ P = 0.05 \]
Duffing oscillator: Frequency response

\[ P = 0.10 \]
For a given excitation frequency, multiple steady-state oscillations can be reached, depending on the initial conditions.
Multiple steady states in reality

model

\[ m\ddot{q} + d\dot{q} + kq + k_{nl}q^3 \approx kE \cos(\Omega t) \]
Duffing oscillator: Summary

Idea of Harmonic Balance

• find periodic solutions of ODEs
• ansatz: truncated Fourier series
• balancing of harmonics \( \rightarrow \) 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 \texttt{NLvib}, application examples
  - limitations, ongoing research
- summary, questions, discussion
Nonlinear mechanical system

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

\[ M \ddot{q} + D \dot{q} + K q + f_{nl}(q, \dot{q}) = f_{ex}(t) \]

with \( q, f_{nl}, f_{ex} \in \mathbb{R}^{n \times 1} \), \( M, D, K \in \mathbb{R}^{n \times n} \), \( M = M^T > 0 \)

and \( f_{ex}(t) = f_{ex}(t + T) \)

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

**Ansatz:** \( q_h(t) = Q_0 + \sum_{k=1}^{\infty} Q_{c,k} \cos(k\Omega t) + Q_{s,k} \sin(k\Omega t) \)

We will use this representation.

**with**
\( Q_0, Q_{c,k}, Q_{s,k} \in \mathbb{R}^{n \times 1} \)
\( \tilde{Q}_k, Q_k \in \mathbb{C}^{n \times 1} \) \( \forall k \neq 0 \)
\( \tilde{Q}_k = \tilde{Q}_{-k}^* \)

mathematically equivalent representations of truncated Fourier series
Nonlinear mechanical system

Time derivatives of ansatz:

\[ q_h = \mathcal{R}\{ \sum_{k=0}^{\infty} Q_k e^{ik\Omega t} \} \quad \dot{q}_h = \mathcal{R}\{ \sum_{k=0}^{\infty} ik\Omega Q_k e^{ik\Omega t} \} \quad \ddot{q}_h = \mathcal{R}\{ \sum_{k=0}^{\infty} -(k\Omega)^2 Q_k e^{ik\Omega t} \} \]

Substitute into equations of motion

\[ M \mathcal{R}\{ \sum_{k=0}^{\infty} -(k\Omega)^2 Q_k e^{ik\Omega t} \} + D \mathcal{R}\{ \sum_{k=0}^{\infty} ik\Omega Q_k e^{ik\Omega t} \} + K \mathcal{R}\{ \sum_{k=0}^{\infty} Q_k e^{ik\Omega t} \} + f_{nl} - f_{ex} =: \mathbf{r} \neq 0 \]

Residual due to Fourier approximation of possibly nonsmooth function

\[ \mathcal{R}\{ \sum_{k=0}^{\infty} \left[ -(k\Omega)^2 M + ik\Omega D + K \right] Q_k e^{ik\Omega t} \} + f_{nl}(q_h, \dot{q}_h) - f_{ex} = \mathbf{r} \]

\[ \mathcal{R}\{ \sum_{k=0}^{\infty} \left( \left[ -(k\Omega)^2 M + ik\Omega D + K \right] Q_k + F_{nl,k} - F_{ex,k} \right) e^{ik\Omega t} \} = \mathbf{r} \]

with \( f_{ex} = \mathcal{R}\{ \sum_{k=0}^{\infty} F_{ex,k} e^{ik\Omega t} \} \)

\[ \frac{1}{\pi} \int_0^{2\pi} f_{nl}(q_h, \dot{q}_h) e^{-ik\Omega t} d\Omega t = \begin{cases} 2F_{nl,0} & \text{if } k \neq 0 \\ F_{nl,k} & \text{if } k = 1, \ldots, H \end{cases} \]

Harmonic Balance:

\[ R_k = 0 \quad k = 0, \ldots, H \]

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

\[ \begin{bmatrix} R_0(Q_0, Q_1, \ldots, Q_H) = 0 \\ R_1(Q_0, Q_1, \ldots, Q_H) = 0 \\ \vdots \\ R_H(Q_0, Q_1, \ldots, Q_H) = 0 \end{bmatrix} \]

\( n(2H+1) \) algebraic equations

in \( n(2H+1) \) unknowns

with

\( R_k, Q_k \in \mathbb{R}^{n \times 1} \)

\( k = 1, \ldots, H \)
Harmonic Balance as a Galerkin method

Weighted residual method

\[
\int_0^T r \left[ q_h(t), \dot{q}_h(t) \right] \psi_k^*(t) \, dt = 0 \quad k = 0, 1, \ldots
\]

Galerkin method: take ansatz functions as weight functions

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

\[
\int_0^T r \left[ q_h(t), \dot{q}_h(t) \right] e^{-i k \Omega t} \, dt = 0
\]

\[
\int_0^T \Re \left\{ \sum_{\ell=0}^{\infty} R_{\ell} e^{i \ell \Omega t} \right\} e^{-i k \Omega t} \, dt = 0
\]

\[
\int_0^{2\pi} \Re \left\{ \sum_{\ell=0}^{\infty} R_{\ell} e^{i \ell \tau} \right\} e^{-i k \tau} \, d\tau = 0 \quad \text{with} \quad \tau = \Omega t
\]

\[
\int_0^{2\pi} \sum_{\ell=0}^{\infty} \left( \frac{R_{\ell} e^{i \ell \tau}}{2} + \frac{R_{\ell}^* e^{-i \ell \tau}}{2} \right) e^{-i k \tau} \, d\tau = 0
\]

since

\[
\int_0^{2\pi} e^{im \tau} \, d\tau = \begin{cases} 2\pi & m = 0 \\ 0 & m \neq 0 \quad m \in \mathbb{Z} \end{cases}
\]

\( R_k = 0 \)
Harmonic Balance vs. numerical integration

numerical integration
successive forward time stepping until transient approaches periodic state
\[ q(t_\ell) \rightarrow q(t_{\ell+1}) \]

Harmonic Balance
solve algebraic equation system in Fourier coefficients
\[ q \approx \sum_k Q_{c,k} \cos k\Omega t + Q_{s,k} \sin k\Omega t \]

ansatz
Nonlinear mechanical system

Interpretation of Harmonic Balance

**dynamic stiffness matrix**

\[ S(k\Omega) \]

\[ R_k = \left[ - (k\Omega)^2 M + ik\Omega D + K \right] Q_k + F_{nl,k} - F_{ex,k} = 0 \]

- linear internal forces
- nonlinear internal forces
- external forces

The true challenge is the calculation of the nonlinear force harmonics

\[ F_{nl,k}(Q_0, \ldots, Q_H) ! \]

formal definition:

\[
\frac{1}{\pi} \int_0^{2\pi} f_{nl}(q_h, \dot{q}_h) e^{-ik\Omega t} d\Omega t = \begin{cases} 
2F_{nl,0} \\
F_{nl,k} \quad k = 1, \ldots, H
\end{cases}
\]

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

\[ \{ F_{n1,k} \} = \text{FFT} \left[ f_{n1} \left( \text{iFFT} \left[ \{ Q_k \} \right] \right) \right] \]

**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 $R(x) = 0$
with respect to $x$
with $R, x \in \mathbb{R}^{n(2H+1) \times 1}$

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

**Idea of Newton method**
Linearization of residual

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

Iteration step

$$x^{(j+1)} = x^{(j)} - \frac{\partial R}{\partial x}\bigg|_{x^{(j)}}^{-1} R(x^{(j)})$$

for Harmonic Balance, we have

$$
\begin{bmatrix}
R_0 \\
\Re \{R_1\} \\
\Im \{R_1\} \\
\vdots \\
\Re \{R_H\} \\
\Im \{R_H\}
\end{bmatrix}
\begin{bmatrix}
x_0 \\
\Re \{x_1\} \\
\Im \{x_1\} \\
\vdots \\
\Re \{x_H\} \\
\Im \{x_H\}
\end{bmatrix}
= 0$$
Solution of the algebraic equation system

**Task**

solve \( R(x) = 0 \)

with respect to \( x \)

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

**Solvers**

- pseudo-time solver
- Newton-like solver
- secant solver
- ...

**Idea of Newton method**

Linearization of residual

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

Iteration step

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

for Harmonic Balance, we have

\[
R = \begin{bmatrix}
R_0 \\
\Re \{R_1\} \\
\Im \{R_1\} \\
\vdots \\
\Im \{R_H\}
\end{bmatrix},
\quad
x = \begin{bmatrix}
Q_0 \\
\Re \{Q_1\} \\
\Im \{Q_1\} \\
\vdots \\
\Im \{Q_H\}
\end{bmatrix}
\]

\[
\| R \left( x^{(2)} \right) \| = 2 \cdot 10^0
\]

\[
\begin{array}{c}
0 \\
10 \\
-5 \\
0 \\
-10
\end{array}
\]

\[
\begin{array}{c}
0 \pi/2 \\
\pi \\
3\pi/2 \\
2\pi
\end{array}
\]

\[
\begin{array}{c}
\Omega t
\end{array}
\]
Solution of the algebraic equation system

**Task**

\[
\text{solve } R(x) = 0
\]

with respect to \( x \)

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

**Solvers**

- pseudo-time solver
- Newton-like solver
- secant solver
- ...

**Idea of Newton method**

Linearization of residual

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

Iteration step

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

for Harmonic Balance, we have

\[
R = \begin{bmatrix}
R_0 \\
\Re\{R_1\} \\
\Im\{R_1\} \\
\vdots \\
\Re\{R_H\} \\
\Im\{R_H\}
\end{bmatrix}
\]

\[
x = \begin{bmatrix}
Q_0 \\
\Re\{Q_1\} \\
\Im\{Q_1\} \\
\vdots \\
\Re\{Q_H\} \\
\Im\{Q_H\}
\end{bmatrix}
\]

\[
\| R(x^{(3)}) \| = 0.5 \cdot 10^0
\]
Solution of the algebraic equation system

**Task**

\[ R(x) = 0 \]

with respect to \( x \)

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

**Solvers**

- pseudo-time solver
- Newton-like solver
- secant solver
- ...

**Idea of Newton method**

Linearization of residual

\[
R \left( x^{(j+1)} \right) \approx R \left( x^{(j)} \right) + \frac{\partial R}{\partial x} \bigg|_{x^{(j)}} \left( x^{(j+1)} - x^{(j)} \right) = 0
\]

Iteration step

\[
x^{(j+1)} = x^{(j)} - \frac{\partial R}{\partial x} \bigg|_{x^{(j)}}^{-1} R \left( x^{(j)} \right)
\]

for Harmonic Balance, we have

\[
R = \begin{bmatrix}
R_0 \\
\Re \{ R_1 \} \\
\Im \{ R_1 \} \\
\vdots \\
\Im \{ R_H \}
\end{bmatrix}
\quad x = \begin{bmatrix}
Q_0 \\
\Re \{ Q_1 \} \\
\Im \{ Q_1 \} \\
\vdots \\
\Im \{ Q_H \}
\end{bmatrix}
\]

\[
\| R \left( x^{(4)} \right) \| = 0.2 \cdot 10^0
\]
Solution of the algebraic equation system

**Task**

solve \( R(x) = 0 \)

with respect to \( x \)

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

**Solvers**

- pseudo-time solver
- Newton-like solver
- secant solver
- ...

**Idea of Newton method**

Linearization of residual

\[
R\left(x^{(j+1)}\right) \approx R\left(x^{(j)}\right) + \frac{\partial R}{\partial x}\bigg|_{x^{(j)}} \left(x^{(j+1)} - x^{(j)}\right) = 0
\]

Iteration step

\[
x^{(j+1)} = x^{(j)} - \left(\frac{\partial R}{\partial x}\bigg|_{x^{(j)}}\right)^{-1} R\left(x^{(j)}\right)
\]

for Harmonic Balance, we have

\[
R = \begin{bmatrix}
R_0 \\
\Re\{R_1\}
\end{bmatrix}
\]

\[
x = \begin{bmatrix}
Q_0 \\
\Re\{Q_1\}
\end{bmatrix}
\]

\[
\|R\left(x^{(5)}\right)\| = 7 \cdot 10^{-2}
\]
Solution of the algebraic equation system

**Task**

solve \( \mathbf{R}(\mathbf{x}) = 0 \)

with respect to \( \mathbf{x} \)

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

**Solvers**

- pseudo-time solver
- Newton-like solver
- secant solver
- ...

**Idea of Newton method**

Linearization of residual

\[
\mathbf{R}\left(\mathbf{x}^{(j+1)}\right) \approx \mathbf{R}\left(\mathbf{x}^{(j)}\right) + \frac{\partial \mathbf{R}}{\partial \mathbf{x}}\bigg|_{\mathbf{x}^{(j)}} \left(\mathbf{x}^{(j+1)} - \mathbf{x}^{(j)}\right) = 0
\]

Iteration step

\[
\mathbf{x}^{(j+1)} = \mathbf{x}^{(j)} - \frac{\partial \mathbf{R}}{\partial \mathbf{x}}\bigg|_{\mathbf{x}^{(j)}}^{-1} \mathbf{R}\left(\mathbf{x}^{(j)}\right)
\]

for Harmonic Balance, we have

\[
\mathbf{R} = \begin{bmatrix}
\mathbf{R}_0 \\
\Re\{\mathbf{R}_1\} \\
\Im\{\mathbf{R}_1\} \\
\vdots \\
\Re\{\mathbf{R}_H\} \\
\Im\{\mathbf{R}_H\}
\end{bmatrix}
\quad \mathbf{x} = \begin{bmatrix}
\mathbf{Q}_0 \\
\Re\{\mathbf{Q}_1\} \\
\Im\{\mathbf{Q}_1\} \\
\vdots \\
\Re\{\mathbf{Q}_H\} \\
\Im\{\mathbf{Q}_H\}
\end{bmatrix}
\]

\[
\|\mathbf{R}\left(\mathbf{x}^{(6)}\right)\| = 3 \cdot 10^{-2}
\]

\[
\begin{array}{c c c c}
0 & 0.1 & 0.05 & 0
\end{array}
\]

\[
\begin{array}{c c c c}
0 & -0.1 & -0.05 & 0
\end{array}
\]

\[
\Omega t
\]

\[
\begin{array}{c c c c c c c}
0 & \pi/2 & \pi & 3\pi/2 & 2\pi
\end{array}
\]
Solution of the algebraic equation system

**Task**

solve \( R(x) = 0 \)

with respect to \( x \)

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

**Solvers**

- pseudo-time solver
- Newton-like solver
- secant solver
- ...

**Idea of Newton method**

Linearization of residual

\[
R\left(x^{(j+1)}\right) \approx R\left(x^{(j)}\right) + \frac{\partial R}{\partial x}\bigg|_{x^{(j)}} \left(x^{(j+1)} - x^{(j)}\right) = 0
\]

Iteration step

\[
x^{(j+1)} = x^{(j)} - \left(\frac{\partial R}{\partial x}\right)^{-1} x^{(j)} R\left(x^{(j)}\right)
\]

for Harmonic Balance, we have

\[
\begin{bmatrix}
R_0 \\
\Re\{R_1\} \\
\Im\{R_1\} \\
\vdots \\
\Im\{R_H\} \\
\Re\{Q_1\} \\
\Im\{Q_1\} \\
\vdots \\
\Im\{Q_H\}
\end{bmatrix} \quad \begin{bmatrix}
x_0 \\
\Re\{x_1\} \\
\Im\{x_1\} \\
\vdots \\
\Im\{x_H\} \\
\Re\{q_1\} \\
\Im\{q_1\} \\
\vdots \\
\Im\{q_H\}
\end{bmatrix}
\]

\[
\|R\left(x^{(7)}\right)\| = 2 \cdot 10^{-3}
\]
Solution of the algebraic equation system

**Task**

solve \( R(x) = 0 \)

with respect to \( x \)

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

**Solvers**

- pseudo-time solver
- Newton-like solver
- secant solver
- ...

**Idea of Newton method**

Linearization of residual

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

Iteration step

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

for Harmonic Balance, we have

\[
\begin{align*}
R &= \begin{bmatrix} R_0 & \mathbb{R}\{R_1\} & \ldots & \mathbb{R}\{R_H\} \\
\mathbb{R}\{R_1\} & \ldots & \mathbb{R}\{R_H\} \\
\vdots & & \ddots & \vdots \\
\mathbb{R}\{R_H\} & \ldots & \ldots & \mathbb{R}\{R_H\} \\
\end{bmatrix} \\
x &= \begin{bmatrix} Q_0 \\
\mathbb{R}\{Q_1\} \\
\vdots \\
\mathbb{R}\{Q_H\} \\
\end{bmatrix}
\end{align*}
\]

\[
\| R\left(x^{(8)}\right) \| = 3 \cdot 10^{-5}
\]

\[
\begin{array}{c}
\text{\( y \)} \\
\begin{array}{cccc}
0 & 0.05 & 0.1 & -0.05 & -0.1 \\
\hline
0 & \pi/2 & \pi & 3\pi/2 & 2\pi
\end{array}
\end{array}
\]
Solution of the algebraic equation system

Task

solve $R(x) = 0$

with respect to $x$

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

Solvers

• pseudo-time solver
• Newton-like solver
• secant solver
• ...

Idea of Newton method

Linearization of residual

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

Iteration step

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

for Harmonic Balance, we have

$R = \begin{bmatrix} R_0 \\ \mathbb{R}\{R_1\} \\ \mathbb{I}\{R_1\} \\ \vdots \\ \mathbb{I}\{R_H\} \end{bmatrix}$

$x = \begin{bmatrix} Q_0 \\ \mathbb{R}\{Q_1\} \\ \mathbb{I}\{Q_1\} \\ \vdots \\ \mathbb{I}\{Q_H\} \end{bmatrix}$

$\|R \left( x^{(9)} \right) \| = 4 \cdot 10^{-9}$

- 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 \( R(X) = 0 \)
with respect to \( X = \begin{bmatrix} x \\ \Omega \end{bmatrix} \)
with \( R, x \in \mathbb{R}^{n(2H+1) \times 1} \)
in the interval \( \Omega^{(s)} \leq \Omega \leq \Omega^{(e)} \)

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](http://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
  - (distributed) polynomial stiffness nonlinearity
- **analysis types**
  - frequency response
  - nonlinear modal analysis

\[
\begin{align*}
  f_{nl} &= \sum_e w_e f_{nl,e} (w_e^T q, w_e^T u) \\
  f_{nl} &= \sum_k E_{ki} \prod_j q_j^{p_{kj}}
\end{align*}
\]
**NLvib: Duffing oscillator**

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

% 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**

```matlab
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

$$M \ddot{q} + D \dot{q} + K q + f_{nl} = \Re \{ F_{ex,1} e^{i \Omega t} \}$$

with the nonlinear force

$$f_{nl} = \omega f_{fric} (q_{nl}, \dot{q}_{nl}) = \omega f_{fric} (\omega^T q, \omega^T \dot{q})$$

and the regularized Coulomb dry friction law

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

**Computation times:**

- HB, whole frequency response: ~1.3 s
- Numerical integration, single frequency: ~30 s
Vibration prediction of bladed disks with friction joints

- **Nonlinear Vibration Analysis**
  - frequency response analysis
  - nonlinear modal analysis
  - flutter analysis

- **Component Mode Synthesis**
  - resonances
  - damping
  - deflection shapes
  - dynamic stress

- **Measurements**
  - $\{w_e\}$

- **Validation**

- **CFD**
  - $K_a, D_a, F_{ex}$

- **Static FEA**
  - $M_s, K_s$
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 \texttt{NLvib}, application examples

• limitations, ongoing research

• summary, questions, discussion
Harmonic Balance has to main limitations

Limitation 1: Only periodic oscillations $\rightarrow$ no quasi-periodic, broadband or chaotic ones

Extensions

- for quasi-periodic oscillations: multi-frequency HB [SCHI06, KRAC16], adjusted HB [GUSK12]
- for broadband or chaotic oscillations: ongoing research

\[ q_h(t) = \Re\left\{ \sum_{n \in \mathcal{H}} Q_n e^{i(n \cdot \omega) t} \right\}, \mathcal{H} \subset \mathbb{Z}^p, \omega \in \mathbb{R}^p \]

quasi-periodic oscillations on an invariant torus
Harmonic Balance has to main limitations

**Limitation 2:** *Harmonic base functions* $\rightarrow$ Gibbs phenomenon near discontinuities

**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*

\[
R(x) = \begin{bmatrix} q(T) - q_0 \\ u(T) - u_0 \end{bmatrix} = 0
\]

with respect to \( x = \begin{bmatrix} q_0 \\ u_0 \end{bmatrix}^T \)

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

where \( q(T), u(T) \) 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 \( R(x) = \begin{bmatrix} q(T) - q_0 \\ u(T) - u_0 \end{bmatrix} = 0 \)

with respect to \( x = \begin{bmatrix} q_0^T \\ u_0^T \end{bmatrix}^T \)

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

where \( q(T), u(T) \) are determined by forward numerical integration
Alternatives to Harmonic Balance

Periodic oscillations

• Fourier-based alternatives
  – trigonometric collocation
  – time spectral method
• Shooting methods
• …
Alternatives to Harmonic Balance

Periodic oscillations
- Fourier-based alternatives
  - trigonometric collocation
  - time spectral method
- Shooting methods
- ...

General oscillations:
Forward numerical integration

Shooting problem
\[
R(x) = \begin{bmatrix}
q(T) - q_0 \\
u(T) - u_0
\end{bmatrix} = 0
\]
with respect to \( x = \begin{bmatrix} q_0^T & u_0^T \end{bmatrix}^T \)
with \( R, x \in \mathbb{R}^{2n \times 1} \)
where \( q(T), u(T) \) 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 \texttt{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


Treatment of particular nonlinearities, Asymptotic Numerical Method


Computation of quasi-periodic oscillations with Harmonic Balance


Computation of non-smooth periodic oscillations


Questions?

Topics for discussion?
### Appendix: Overview of basic examples in NLvib

<table>
<thead>
<tr>
<th>name</th>
<th>n</th>
<th>HB</th>
<th>Shooting</th>
<th>FRF</th>
<th>NMA</th>
</tr>
</thead>
<tbody>
<tr>
<td>01_Duffing</td>
<td>1</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>02_twoDOFoscillator_cubicSpring</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>03_twoDOFoscillator_unilateralSpring</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>04_twoDOFoscillator_cubicSpring_NM</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>05_twoDOFoscillator_tanhDryFriction_NM</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>06_twoSprings_geometricNonlinearity</td>
<td>2</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>07_multiDOFoscillator_multipleNonlinearities</td>
<td>3</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>08_beam_tanhDryFriction</td>
<td>16</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>09_beam_cubicSpring_NM</td>
<td>38</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>0</td>
</tr>
</tbody>
</table>

- 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**

**Equations of motion**

\[ M \ddot{q} + D \dot{q} + K q + f_{nl} = \Re \{ F_{ex1} e^{i \Omega t} \} \]

**Local nonlinear elements**

\[ f_{nl} = \sum e w_e f_{nl,e} \left( w_e^T q, w_e^T u \right) \]

**Matlab syntax**

```matlab
% 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);
```

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).
Appendix: Local nonlinear elements in NLvib

Some of the already available local nonlinear elements

\[
f_{nl,e}(q_{nl}, u_{nl}) = \begin{cases} 
\text{nonlinear_elements}\{e\}.\text{stiffness} & q_{nl}^3 \\
\text{nonlinear_elements}\{e\}.\text{damping} & q_{nl}^2 u_{nl} \\
\text{nonlinear_elements}\{e\}.\text{friction\_limit\_force} & \text{tanh} \left( \frac{u_{nl}}{\text{nonlinear_elements}\{e\}.\text{eps}} \right) \\
\text{nonlinear_elements}\{e\}.\text{stiffness} & (q_{nl} - \text{nonlinear_elements}\{e\}.\text{gap}) 
\end{cases}
\]

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 = ...
  end

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**

\[
M = \text{diag}\{m_i\} = \begin{bmatrix}
    k_0 + k_1 & -k_1 & 0 & \cdots & 0 \\
    -k_1 & k_1 + k_2 & -k_2 & \cdots & \vdots \\
    0 & -k_2 & \ddots & \ddots & \vdots \\
    \vdots & \vdots & \ddots & \ddots & 0 \\
    0 & \cdots & 0 & -k_{n-1} & k_{n-1} + k_n
\end{bmatrix}
\]

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

\[
D \text{ analogous to } K
\]

**Matlab syntax**

```matlab
% 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

\[ q_{\text{full}} = \begin{bmatrix} w_1 \\ w'_1 \\ \vdots \\ w_{n_{\text{nod}}} \\ w'_{n_{\text{nod}}} \end{bmatrix} = Lq \]

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

Matlab syntax

```matlab
% 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

$$q_{\text{full}} = \begin{bmatrix} x_1 \\ \vdots \\ x_{n_{\text{nod}}} \end{bmatrix} = Lq$$

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

Matlab syntax

```matlab
% 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

\[ f_{nl} = E^T z = \sum_k E_{ki} z_k \quad \text{with} \quad z_k = \prod_j q_j^{p_{kj}} \]

Matlab syntax

```matlab
% 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

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

\[
\ddot{q}_2 + 2\zeta_2 \omega_2 \dot{q}_2 + \omega_2^2 q_2 + \frac{\omega_1^2}{2} q_1^2 + \omega_2^2 q_1 q_2 + \frac{3\omega_1^2}{2} q_2^2 + \frac{\omega_1^2 + \omega_2^2}{2} q_1^3 q_2 + \frac{\omega_1^2 + \omega_2^2}{2} q_1^2 q_2 = 0
\]
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 function
   - 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 $R(X) = 0$

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

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

Matlab syntax

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

- columns are solution points
- initial guess
- residual function $R(X)$
- interval limits
- nominal step size
- array of structures returned by postprocessing functions
- postprocessing functions $F(X)$ (optional)
- fsolve options (optional)
- per solution step

Solinfo.FCtotal
Solinfo.ctime
Solinfo.iEx
Solinfo.NIT
Solinfo.FC

total function evaluation count
total computation time
fsolve exit flag
number of iterations
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 \( R(x) = \begin{bmatrix} (x_1 - 1)^2 \\ (10^7 x_2 - 0.75)^2 \end{bmatrix} \) with solution \( x_0 = \begin{bmatrix} 1 \\ 0.75 \cdot 10^{-7} \end{bmatrix} \)

Rescaled problem \( \tilde{R}(\tilde{x}) := R(D_{\text{scale}} \tilde{x}) \) with solution \( \tilde{x}_0 = \begin{bmatrix} 1 \\ 0.75 \end{bmatrix} \)

where the scaling matrix \( D_{\text{scale}} = \text{diag}\{|\hat{x}_i|\} = \begin{bmatrix} 1 & 0 \\ 0 & 10^{-7} \end{bmatrix} \)

attempts to achieve similar orders of magnitude among the new variables \( \tilde{x} = D_{\text{scale}}^{-1} x \)

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:

\[ \text{cond} \left( \frac{dR}{dx} \right) \sim 10^7 \quad \Rightarrow \quad \text{cond} \left( \frac{d\tilde{R}}{d\tilde{x}} \right) \sim 10^0 \]

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.

![Graph showing convergence of residuals with and without scaling](image)
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

\[
solve \quad R(X) = \begin{bmatrix}
R_0 \\
\Re\{R_1\} \\
\Im\{R_1\} \\
\vdots \\
\Im\{R_H\}
\end{bmatrix} = 0
\]

where

\[
R_k = \left[ - (k\omega)^2 M + i k\omega D + K \right] Q_k + F_{nl,k} - F_{ex,k}
\]

with respect to

\[
X = \begin{bmatrix}
Q_0^T \\
\Re\{Q_1^T\} \\
\Im\{Q_1^T\} \\
\vdots \\
\Im\{Q_H^T\} \\
\Omega
\end{bmatrix}^T
\]

in the interval

\[
\Omega^{(s)} \leq \Omega \leq \Omega^{(e)}
\]
Appendix: Frequency response analysis with NLvib

Shooting formulation

\[
\begin{align*}
\text{solve } R(X) &= \left[ \begin{array}{l}
(q(T) - q_0) \\
(u(T) - u_0)
\end{array} \right] \frac{1}{q_{\text{scl}}} = 0 \\
\text{with respect to } X &= \begin{bmatrix} q_0^T & u_0^T & \Omega \end{bmatrix}^T \\
\text{in the interval } \Omega^{(s)} &\leq \Omega \leq \Omega^{(e)} \\
\text{where } q(T), u(T) &\text{ are determined by forward numerical integration}
\end{align*}
\]

$q_{\text{scl}}$ 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

\[
R(X) = \begin{bmatrix}
R_0 \frac{f_{sc\ell}}{a} \\
\Re\{R_1\} \frac{f_{sc\ell}}{a} \\
\Im\{R_1\} \frac{f_{sc\ell}}{a} \\
\vdots \\
\Re\{R_H\} \frac{f_{sc\ell}}{a} \\
\Re\{\sum_{k=0}^{H} Q_k^H M Q_k\} / a^2 - 1 \\
\dot{q}_{inorm}(0) / (\omega a)
\end{bmatrix} = 0
\]

where

\[
R_k = \left[ -(k\omega)^2 M + i k \omega (D - 2 \delta \omega M) + K \right] Q_k + F_{nl,k}
\]

with respect to

\[
X = \begin{bmatrix}
\frac{Q_T^T}{a} \\
\Re\{\frac{Q_T^T}{a}\} \\
\Im\{\frac{Q_T^T}{a}\} \\
\vdots \\
\Re\{\frac{Q_T^T}{a}\} \\
\omega \\
\delta \\
\log_{10} a
\end{bmatrix}^T
\]

in the interval

\[
\log_{10} a^{(s)} \leq \log_{10} a \leq \log_{10} a^{(e)}
\]

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**

\[
\begin{align*}
\text{solve } & \quad R(X) = \left[ \begin{array}{c}
(q(T) - q_0) \\
(u(T) - u_0)
\end{array} \right] \frac{1}{q_{scl}} = 0 \\
\text{with respect to } & \quad X = \left[ \begin{array}{c}
\frac{q_0 T}{a} \\
\frac{u_0 T}{\omega a} \\
\omega \\
D \\
\log_{10} a
\end{array} \right]^T \\
\text{in the interval } & \quad \log_{10} a^{(s)} \leq \log_{10} a \leq \log_{10} a^{(e)} \\
\text{where } & \quad q_0, u_0 \text{ are } q_{0-}, u_{0-} \text{ only with } q_0, i_{\text{norm}} = \alpha \\
\text{where } & \quad u_0, i_{\text{norm}} = 0 \\
\text{are determined by forward numerical } & \quad \text{integration}
\end{align*}
\]

\[q_{scl} \text{ positive real-valued scalar}\]

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 $q(t_0), u(t_0)$, 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 $t_{\ell+1}$.

$$ q(t_{\ell+1}) = g_q(t_{\ell+1}, q(t_{\ell+1}), u(t_{\ell+1}), q(t_\ell), \ldots) $$
$$ u(t_{\ell+1}) = g_u(t_{\ell+1}, q(t_{\ell+1}), u(t_{\ell+1}), q(t_\ell), \ldots) $$

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.

Appendix: Newmark method implemented in NLvib

Equation of motion evaluated at end of a time level

\[ M \ddot{u}^E + D \dot{u}^E + K q^E + f_{\text{nl}}^E - f_{\text{ex}}^E = 0 \]  

(1)

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

\[ u^E = u^S + \frac{\ddot{u}^S + \ddot{u}^E}{2} \Delta t \]  

(2)

\[ q^E = q^S + \frac{u^S + u^E}{2} \Delta t \]  

(3)

\[ \Rightarrow \dot{u}^E = \frac{4}{\Delta t^2} (q^E - q^S) - \frac{4}{\Delta t} u^S - \dot{u}^S \]  

(4)

\[ \Rightarrow u^E = \frac{2}{\Delta t} (q^E - q^S) - u^S \]  

(5)

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

\[ \left( \frac{4}{\Delta t^2} M + \frac{2}{\Delta t} D + K \right) q^E \bigg|_s + f_{\text{nl}} (q^E, u^E (q^E)) = \]

\[ f_{\text{ex}} (t^E) + M \left( \frac{4}{\Delta t^2} q^S + \frac{4}{\Delta t} u^S + \dot{u}^S \right) + D \left( \frac{2}{\Delta t} q^S + u^S \right) \]

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

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

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

Normalized time \( \tau := \Omega t , \ d\tau = \Omega dt \)

Reformulation of time derivatives

\[
\begin{align*}
\dot{u} &= \dot{q} = \frac{dq}{dt} = \frac{dq}{d\tau} \cdot \frac{d\tau}{dt} = \Omega q' \\
\ddot{u} &= \ddot{q} = \ldots = \Omega^2 q''
\end{align*}
\]

Equations of motion in non-normalized and normalized time

\[
\begin{align*}
M\ddot{q} + D\dot{q} + Kq + f_{nl}(q, \dot{q}) &= f_{ex}(t) \\
\frac{M\Omega^2}{\tilde{M}} q'' + \frac{D\Omega}{\tilde{D}} q' + Kq + f_{nl}(q, \Omega q') &= f_{ex}(\tau)
\end{align*}
\]
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;