Professor
Prof. Dr.-Ing. Hartmut Hetzler

Current developer
Alexander Seifert (M.Sc.), Julian Vogelei (M.Sc.)

Former developer
Dr.-Ing. Simon Bäuerle, Dr.-Ing. Jonas Kappauf

CoSTAR

Website CoSTAR

The MATLAB Toolbox CoSTAR

Institute of Mechanics, Engineering Dynamics Group
Department of Mechanical Engineering (FB15)
University of Kassel (Germany)

INSTITUTE FOR MECHANICS

ENGINEERING DYNAMICS

Website Engineering Dynamics Group

Content

UNIKASSEL VERSITÄT

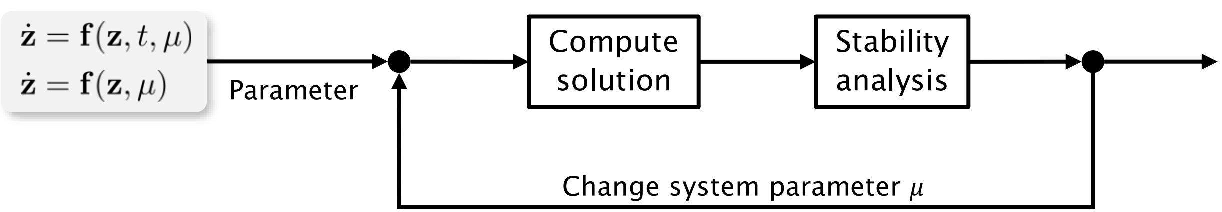
CoSTAR

Continuation of Solution Torus AppRoximations

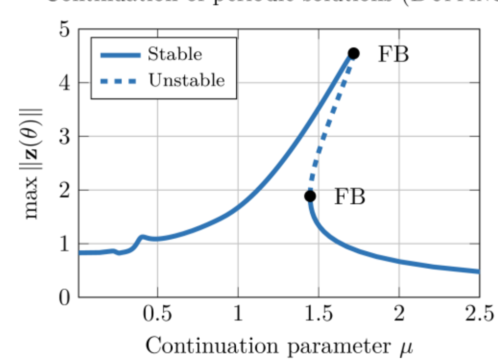
$$\dot{\mathbf{z}} = \mathbf{f}(t, \mathbf{z}, \mu)$$
$$\dot{\mathbf{z}} = \mathbf{f}(\mathbf{z}, \mu)$$

- **Computation of *stationary* solutions of dynamic systems**
  (*stationary* solution: solution type persists for infinite time interval)
  - Equilibrium solutions (**EQ**)
  - Periodic solutions (**PS**)
  - Quasi-periodic solutions (**QPS**) *(2 base frequencies)*
    - Rotordynamics:
      - External forcing (e.g. unbalance) and/or self-excitation (e.g. instabilities)
      - Bladed disks (in jet engines)

- ***Continuation* of stationary solution branches**



Continuation of periodic solutions (Duffing)



ENGINEERING DYNAMICS

Basic Theory: Solution Types

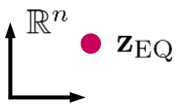
CoSTAR

Continuation of **S**olution **T**orus **A**pp**R**oximations

- **Computation of *stationary* solutions of dynamic systems**
  (*stationary* solution: solution type persists for infinite time interval)

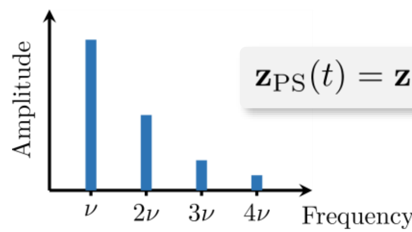
$$\dot{\mathbf{z}} = \mathbf{f}(t, \mathbf{z}, \mu)$$
$$\dot{\mathbf{z}} = \mathbf{f}(\mathbf{z}, \mu)$$

  - Equilibrium solutions (**EQ**)
    - Zero base frequencies

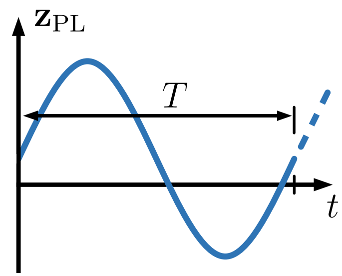


$\mathbf{z}_{\mathrm{EQ}} = \mathrm{const.}$

  - Periodic solutions (**PS**)
    - One *base frequency* ν
    - Higher harmonics $2\nu,\ 3\nu,\ \ldots$



$\mathbf{z}_{\mathrm{PS}}(t) = \mathbf{z}_{\mathrm{PS}}(t+T)$



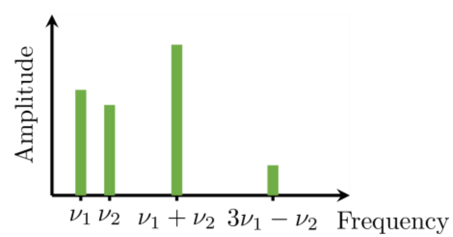
  - Quasi-periodic solutions (**QPS**)
    - $p \geq 2$ *incommensurable* (rationally independent) base frequencies

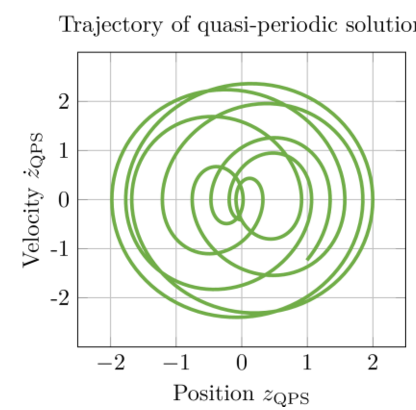
$$\frac{\nu_k}{\nu_j} \in \mathbb{R}\backslash\mathbb{Q} \quad (k \neq j)$$

e.g. $\nu_1 = 1$, $\nu_2 = \sqrt{2}$

$\Rightarrow$ "$T \to \infty$"

    - Linear combinations of base frequencies





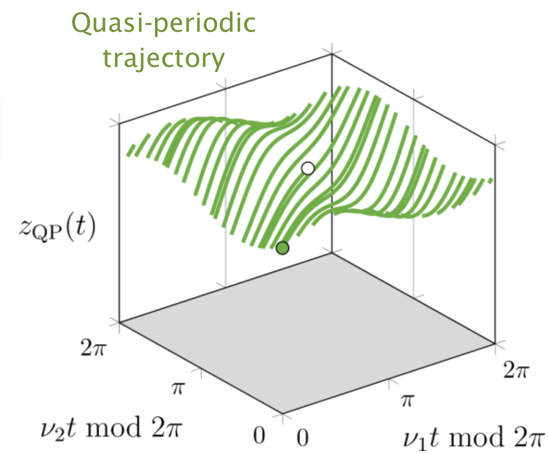
ENGINEERING DYNAMICS

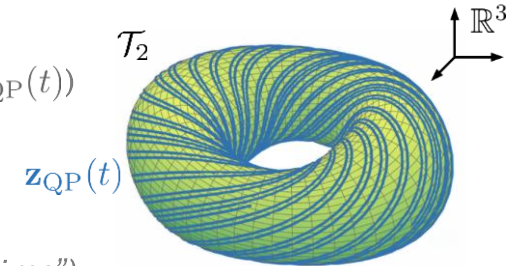
Basic Theory: Quasi-Periodic Solutions

- **Quasi-periodic solutions (QPS)**

  n: state space dimension

  - Quasi-periodic function: $\mathbf{z}_{\mathrm{QP}}(t) = \mathbf{Z}(\nu_1 t, \ldots, \nu_p t) : \mathbb{R} \to \mathbb{R}^n$
    - 2π-periodicity: $\mathbf{Z}(\ldots, \nu_k t, \ldots) = \mathbf{Z}(\ldots, \nu_k t + 2\pi, \ldots)$
  - Torus function: $\mathbf{Z}(\boldsymbol{\theta}) = \mathbf{Z}(\theta_1, \ldots, \theta_p) : \mathbb{T}^p \to \mathbb{R}^n$
    - Coordinate torus: $\mathbb{T}^p = (\mathbb{R}/2\pi\mathbb{Z})^p = [0, 2\pi)^p$
    - 2π-periodicity: $\mathbf{Z}(\ldots, \theta_k, \ldots) = \mathbf{Z}(\ldots, \theta_k + 2\pi, \ldots)$
  - Torus: $\mathcal{T}_p = \{\mathbf{Z}(\boldsymbol{\theta}) \mid \boldsymbol{\theta} \in \mathbb{T}^p\} \subset \mathbb{R}^n$
    - Filled *densely* by quasi-periodic trajectory $\mathbf{z}_{\mathrm{QP}}(t)$ for $t \to \infty$
      (every point on the torus is reached directly or arbitrarily close by $\mathbf{z}_{\mathrm{QP}}(t)$)
    - Carrier of QPS $\mathbf{z}_{\mathrm{QPS}}(t)$ and thus invariant manifold
      ➡ Compute torus $\mathcal{T}_p$ instead of trajectory of $\mathbf{z}_{\mathrm{QPS}}(t)$
    - Parametrisation using torus coordinates: $\boldsymbol{\theta}(t) = \boldsymbol{\nu} t \bmod 2\pi$ *("hyper-time")*





$$\begin{aligned}\dot{\mathbf{z}} &= \mathbf{f}(\mathbf{z}, t) \\ \dot{\mathbf{z}} &= \mathbf{f}(\mathbf{z})\end{aligned} \xrightarrow{\mathbf{z}(\boldsymbol{\theta}(t))} \sum_{k=1}^{p} \frac{\partial \mathbf{Z}(\boldsymbol{\theta})}{\partial \theta_k} \nu_k = \mathbf{f}(\mathbf{Z}(\boldsymbol{\theta}), \tilde{\boldsymbol{\theta}})$$

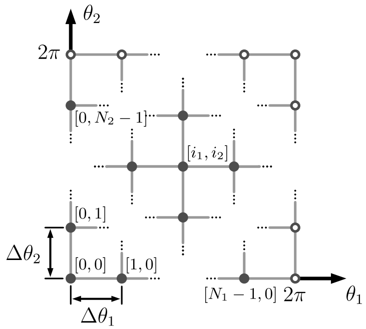
(Hyper-time) Invariance equation

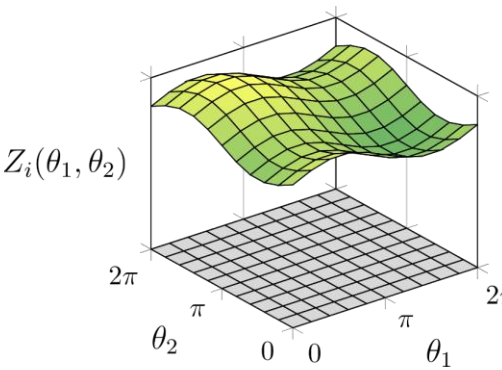
- $p = 0$: equilibrium solutions
- $p = 1$: periodic solutions
- $p = 2$: quasi-periodic solutions (2 base frequencies)

CoSTAR

Basic Theory: Approximation Methods

- **Approximation methods for (quasi-)periodic solutions in CoSTAR**
  - Finite Difference Method (**FDM**)
    - *Local* discretisation (mesh)
    - Approximation of derivatives by weighted sum of torus function values at the nodes

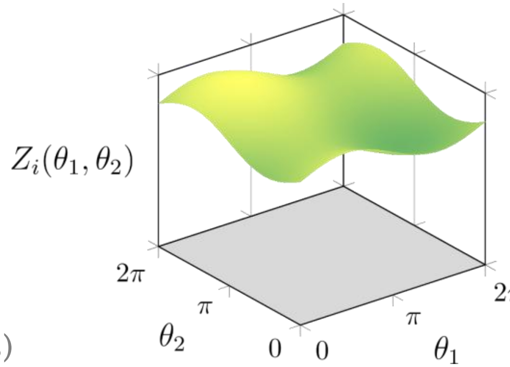




  - FOURIER-GALERKIN Method (**FGM**) *("harmonic balance")*
    - *Global* ansatz functions:
      Truncated (multi-dimensional) FOURIER series

$\mathbf{H}^\top\boldsymbol{\theta} = H_1\theta_1 + H_2\theta_2$

$$\mathbf{Z}(\boldsymbol{\theta}) \approx \mathbf{C}_0 + \sum_{\|\mathbf{H}\| \leq N} \left(\mathbf{C}_\mathbf{H} \cos\left(\mathbf{H}^\top\boldsymbol{\theta}\right) + \mathbf{S}_\mathbf{H} \sin\left(\mathbf{H}^\top\boldsymbol{\theta}\right) \right)$$



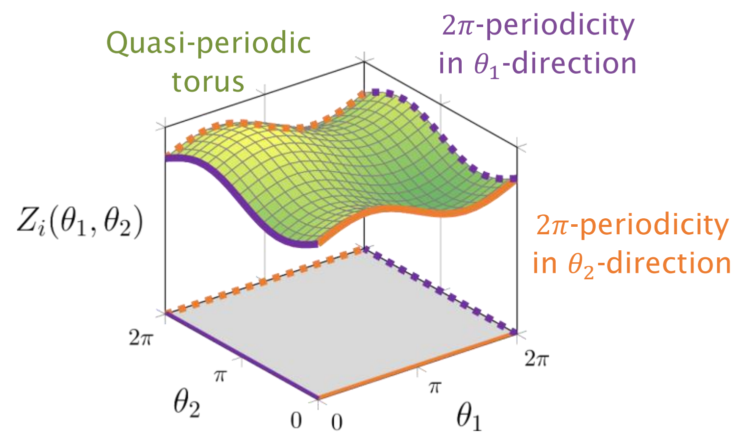
    - GALERKIN-projection of residual on ansatz functions
    - ➢ Error control available (adapting number of higher harmonics)
  - (Multiple) Shooting method (**SHM**)

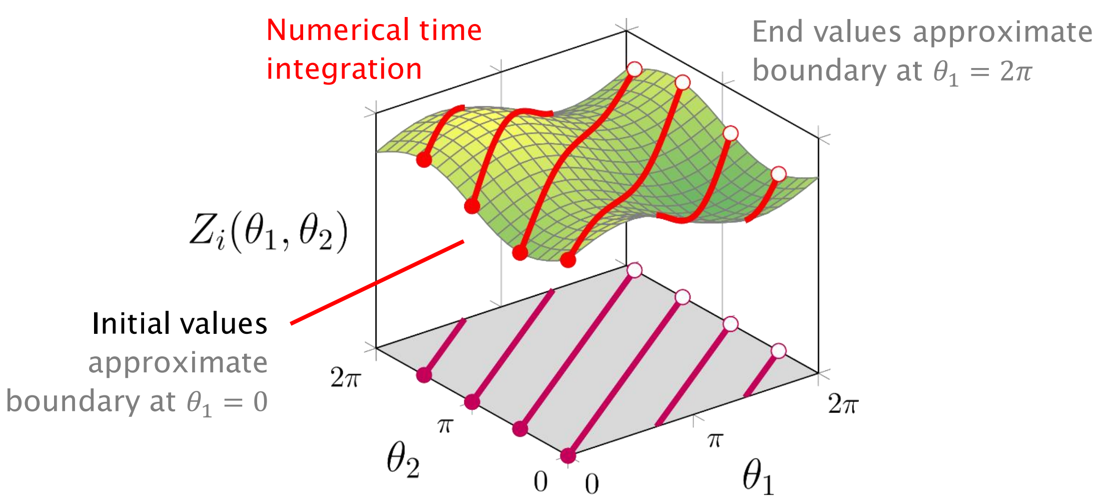
ENGINEERING DYNAMICS

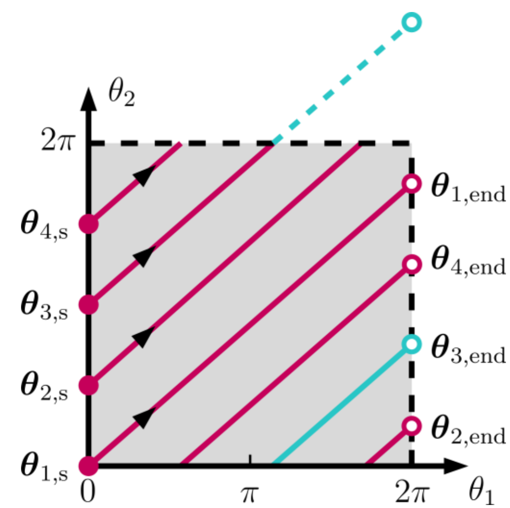
Basic Theory: Approximation Methods

UNI KASSEL VERSITÄT

- **Approximation methods for (quasi-)periodic solutions in** CoSTAR
  - (Multiple) Shooting Method (**SHM**)
    - **PS:** Multiple Shooting Method
    - **QPS:** Single Shooting Method
      - Utilise periodic boundaries & numerical time integration
      - Numerical time integration for $t \in [0, t_{\mathrm{end}}]$, $t_{\mathrm{end}} = 2\pi/\nu_1$ produces *characteristics*
      - Solver for non-linear equations: Find initial values so that boundaries in θ_1-direction are periodic



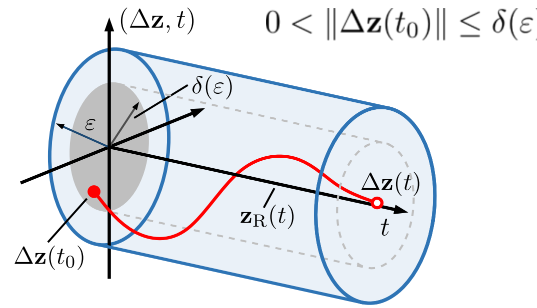




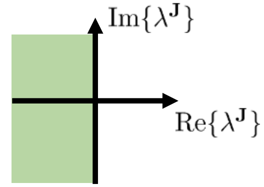
ENGINEERING DYNAMICS

Basic Theory: Stability Analysis

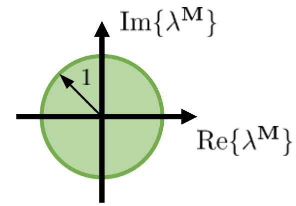
- **Stability computation of solutions in CoSTAR**
  - ➢ LYAPUNOV stability: $\|\mathbf{z}(t)-\mathbf{z}_{\mathrm{R}}(t)\| = \|\Delta\mathbf{z}(t)\| < \varepsilon, \quad \varepsilon > 0$



  - – Equilibrium solutions
    - • Eigenvalue theory
      (eigenvalues $\lambda^{\mathbf{J}}$ of JACOBIAN $\mathbf{J_f}$)

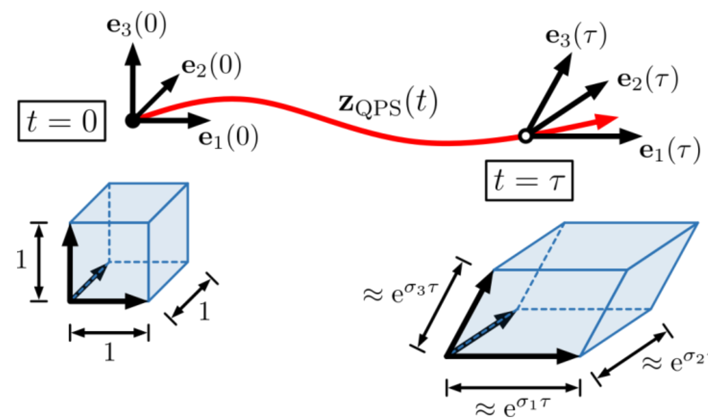


  - – Periodic solutions
    - • FLOQUET theory (eigenvalues λ^{M} of monodromy matrix)
    - ➢ Monodromy matrix can be extracted from **SHM**



  - – Quasi-periodic solutions
    - • Spectrum of 1st order LYAPUNOV exponents
      $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n$

$$\sigma_i = \limsup_{t\to\infty} \frac{1}{t} \ln \|\Delta\mathbf{z}(t)\| = \limsup_{t\to\infty} \frac{1}{t} \ln \|\boldsymbol{\psi}(t,0)\mathbf{e}_i(0)\|$$

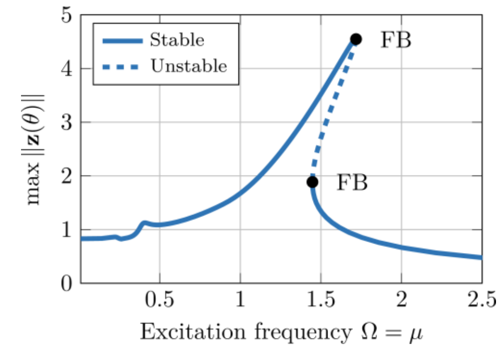


    - ➢ $\sigma_k > 0$ indicates unstable behaviour
    - ➢ Efficient computation possible if **SHM** is used
  - – **(Q)PS** approximated by **FDM** and **FGM**:
    - ➢ *"Reshoot"* the solution (compute again using **SHM**)
    - ➢ Initial values for shooting algorithm from **FDM** or **FGM** solution

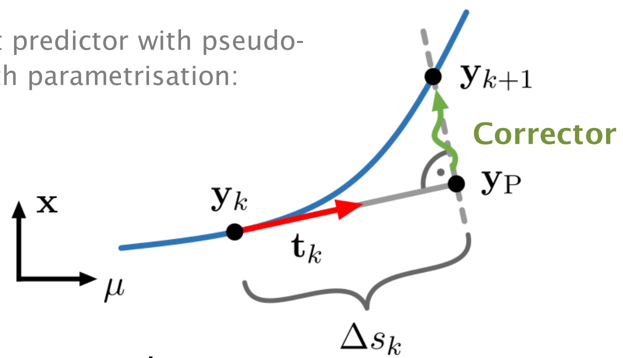
Basic Theory: Numerical Continuation

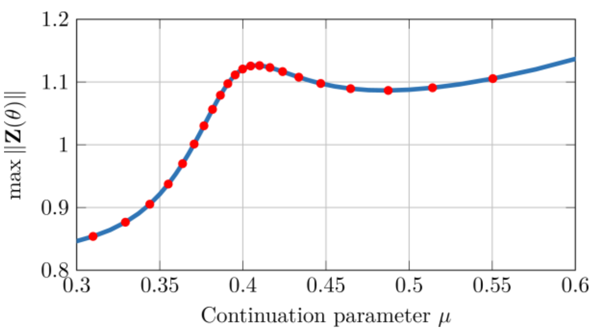
- **Continuation of solution branches**
  - Solutions for varying system parameter μ
    - ➢ Trace solution curve in higher dimensional space
  - *Predictor-corrector algorithm* in CoSTAR
    - Predictor: Predicting new solution
      - ➢ Tangent
      - ➢ Polynomials of order 1, 2 and 3
    - Parametrisation *(subspace constraint)*
      - ➢ Natural
      - ➢ Arclength and pseudo-arclength
      - ➢ 1 norm *(taxicab / Manhattan distance)*
    - Corrector
      - ➢ Solver for non-linear equation systems calculates new solution
  - Step control in CoSTAR
    - ➢ Adapt step length Δs_k to ensure convergence and to reduce overall computation time
    - ➢ Various algorithms based on geometrical information and solver iterations

$$\dot{\mathbf{z}} = \mathbf{f}(\mathbf{z}, t, \mu)$$
$$\dot{\mathbf{z}} = \mathbf{f}(\mathbf{z}, \mu)$$



Tangent predictor with pseudo-arclength parametrisation:

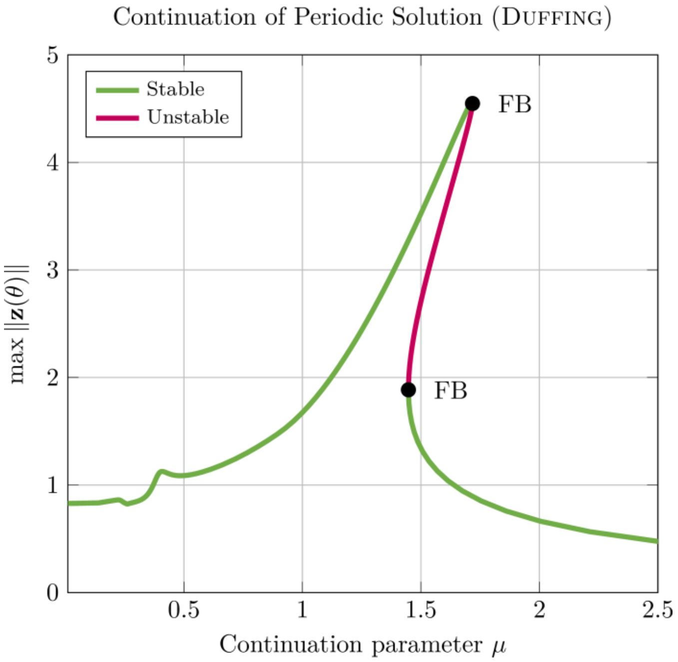




UNIKASSEL VERSITÄT

All features can be used as required

- **Stability computation of solutions**
  - Equilibrium solutions
  - Periodic solutions
    - **SHM** (monodromy matrix is extracted from here)
    - **FDM** & **FGM** (*reshooted* for monodromy matrix)
  - Quasi-periodic solutions
    - **SHM** (LYAPUNOV exponents are extracted from here)
    - **FDM** & **FGM** (*reshooted* for LYAPUNOV exponents)



- **Detection of bifurcation points**
  - Fold / Pitchfork / Transcritical (**FB**)
  - Period Doubling (**PDB**)
  - HOPF (**HB**)
  - NEIMARK-SACKER (**NSB**)

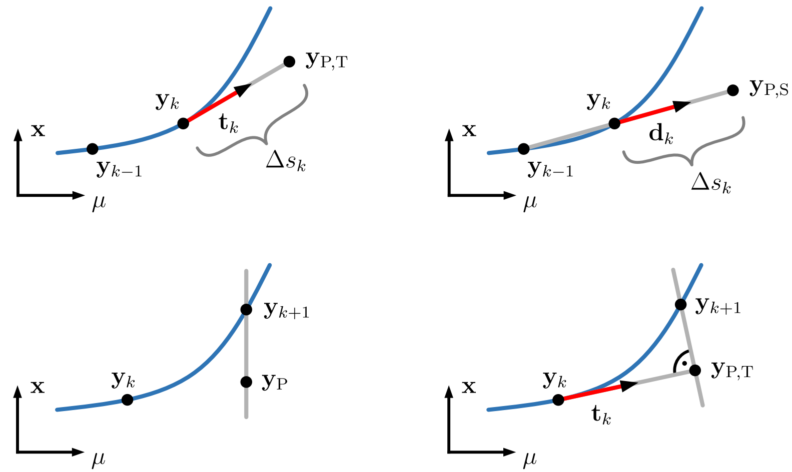
- **Error Control**
  - **FGM:** (PS & QPS) Automatic adaption of number of higher harmonics based on residual

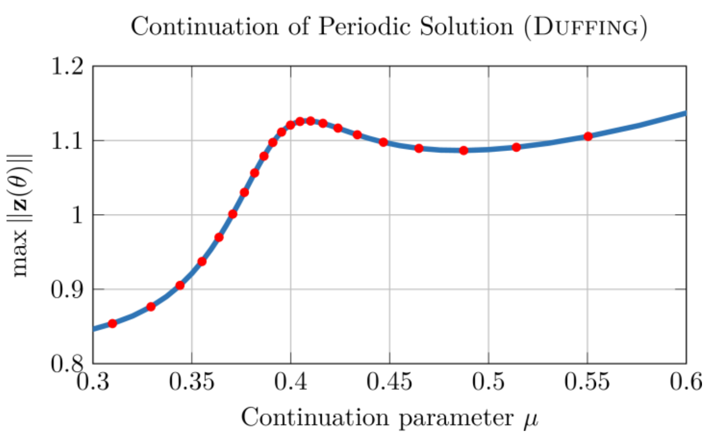
ENGINEERING DYNAMICS

Features

All features can be used as required

- **Continuation: Predictor-corrector algorithm**
  - Predictors
    - Tangent
    - Polynomials of order 1, 2 and 3
  - Parametrisations
    - Natural
    - Arclength and pseudo-arclength
    - 1 norm *(taxicab / Manhattan distance)*
  - Step control
    - Various algorithms based on geometrical information and solver iterations
  - Live plot
    - Creating continuation plot during computation

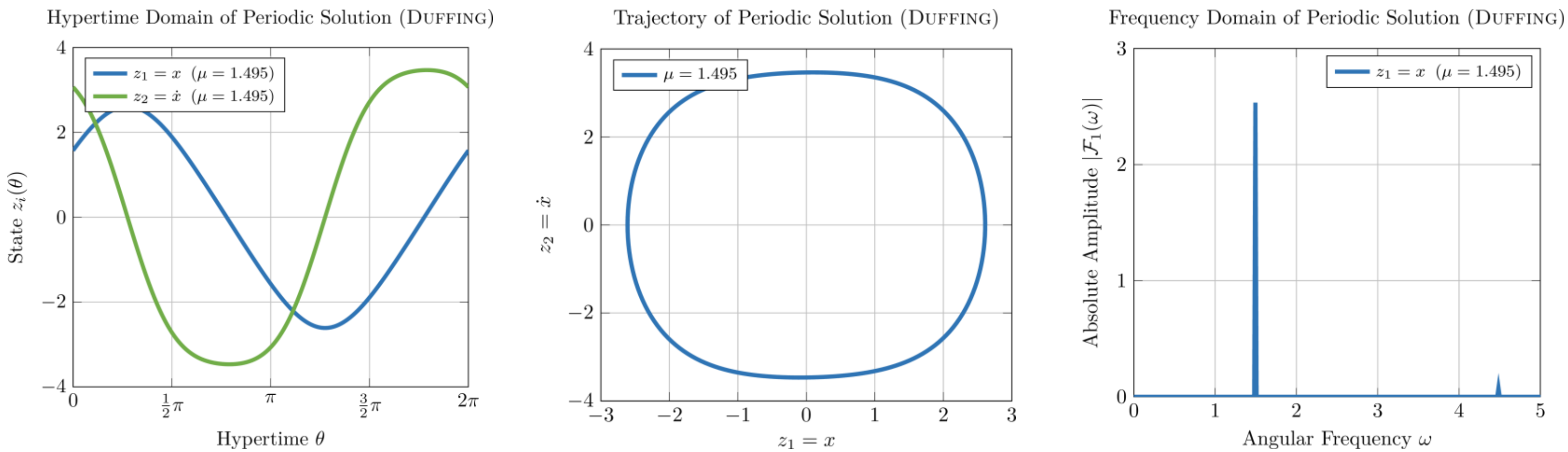




UNIKASSEL VERSITÄT

All features can be used as required

- **Postprocessing methods**
  - `contplot`
    - Creates continuation / bifurcation diagrams (plots solution branches with respect to μ)
  - `solplot`
    - Plots individual solutions in different *solution spaces*
      (Available *solution spaces*: time, hypertime, trajectory and frequency domain)



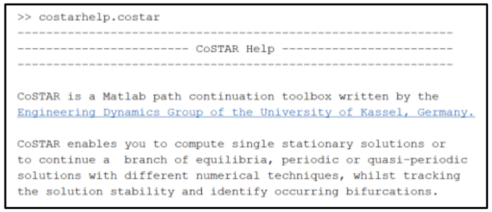
  - `solget`
    - Returns solution data in different solution spaces

ENGINEERING DYNAMICS

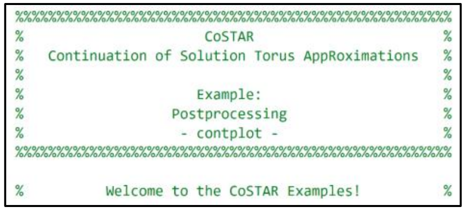
Help Features

- **Gatekeeper** ***cannot be bypassed***
  - Checks the input (the defined options) from the user
  - Reports errors in case of illogical or invalid input

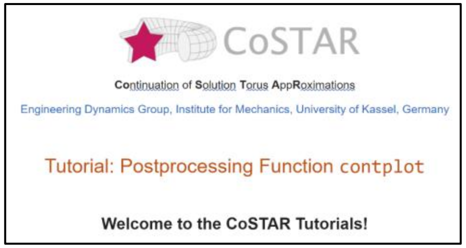
- **Help**
  - `costarhelp` function
    - Quick help in the command window
    - Overview of the available options with a short description
    - Type **`costarhelp.costar`** in the command window to start



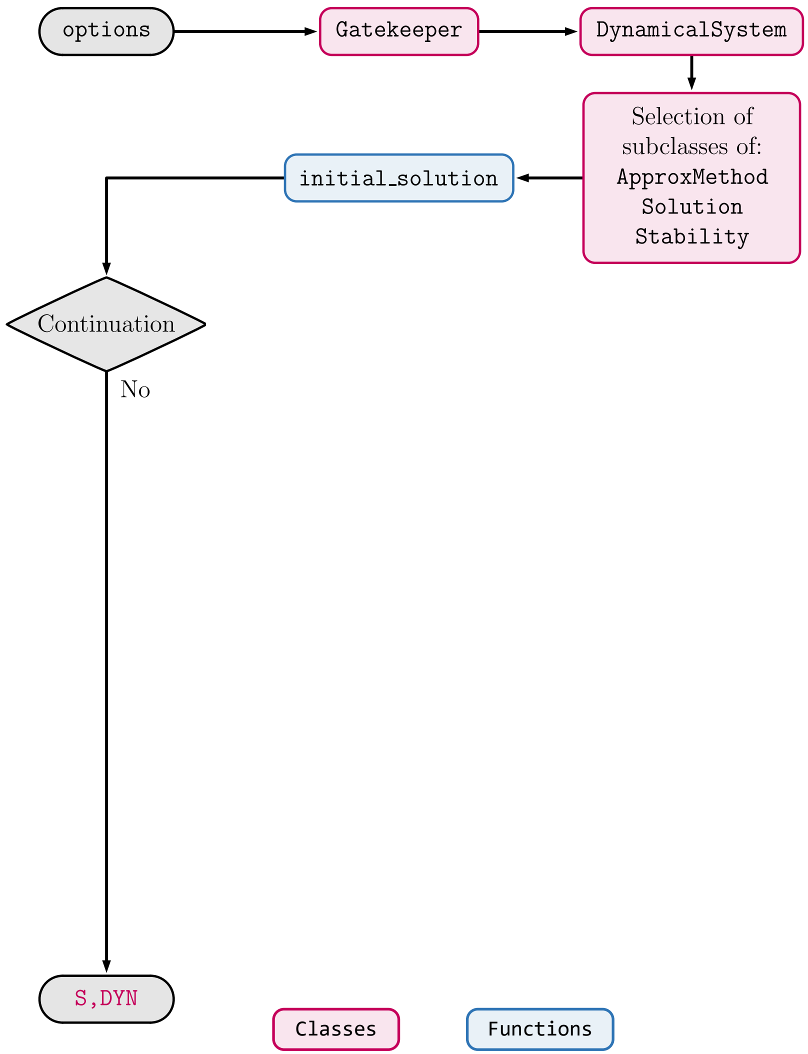
  - Examples
    - Short MATLAB scripts
    - Sample code showing usage of a certain **CoSTAR** module



  - Tutorials
    - MATLAB (live) scripts
    - There is one tutorial for each example (identical code)
    - Comprehensive explanations of a certain **CoSTAR** module



Flow Chart

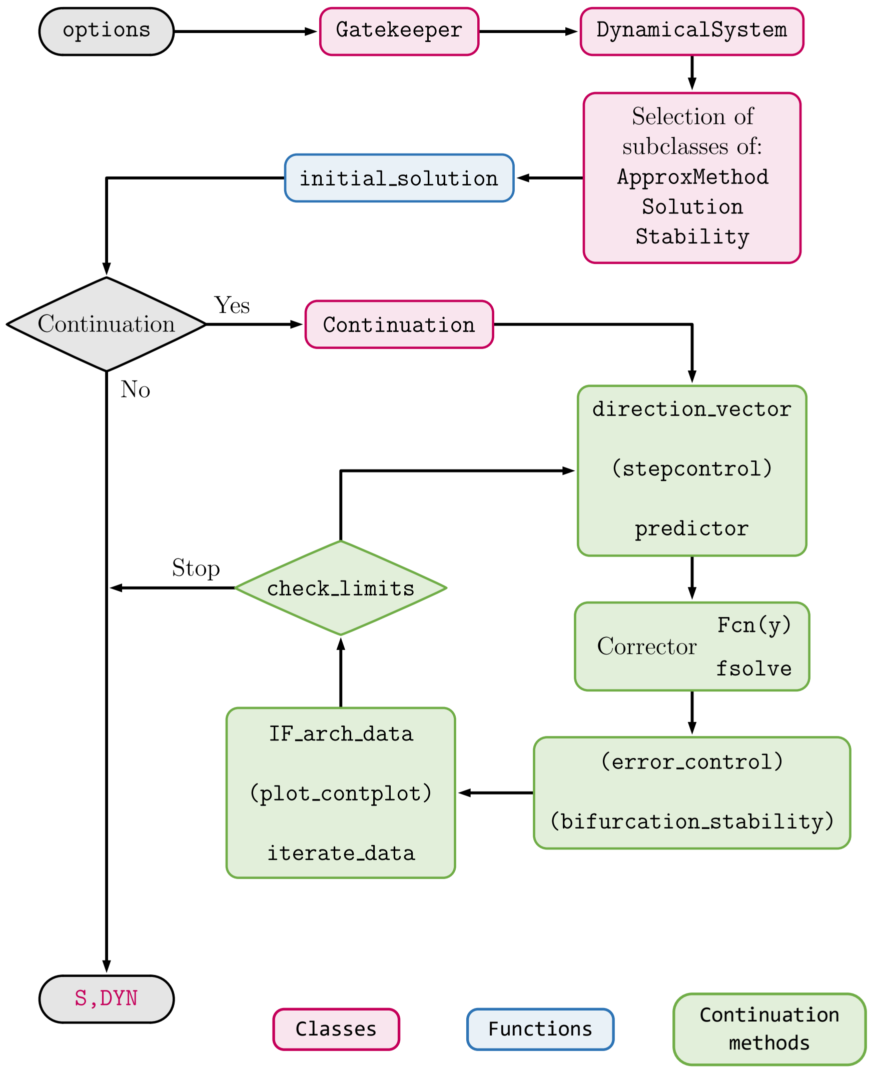


1. options
   - Structure array
   - Contains user-defined options for computation
2. Gatekeeper
   - Checks options
3. DynamicalSystem
   - Stores all options in object DYN
   - Can be used to restart the computation
4. Selection of subclasses
   - ApproxMethod
     - Applies approximation method
   - Solution
     - Stores solution data in object S
   - Stability
     - Used for stability computation
5. initial_solution
   - Computes the first *(initial)* solution

In case of no continuation:

6. Return S and DYN

Flow Chart



6. Continuation loop
   - 6.1. direction_vector
   - 6.2. (stepcontrol) (can be skipped)
   - 6.3. predictor
     - ➢ Computes direction vector, new step width and predictor point
   - 6.4. Corrector
     - ➢ fsolve solving `Fcn(y) = 0`
   - 6.5. (error_control) (can be skipped)
   - 6.6. (bifurcation_stability) (can be skipped)
     - ➢ Performs error control and computes stability as well as bifurcation point
   - 6.7. IF_arch_data
   - 6.8. (plot_contplot) (can be skipped)
   - 6.9. iterate_data
     - ➢ Stores data, updates live plot and performs iterations for next loop
   - 6.10. check_limits
     - ➢ Checks exit conditions

When exit condition is met:

7. Return S and DYN

Code Structure

Classes	➢	All classes and associated methods
Functions	➢	Functions not belonging to any class
RHS	➢	Functions defining right-hand side of $\dot{\mathbf{z}} = \mathbf{f}(t, \mathbf{z}, \mu)$ or $\dot{\mathbf{z}} = \mathbf{f}(\mathbf{z}, \mu)$
test	➢	Scripts to test the code
Tutorials	➢	Tutorial & example scripts
Version_Log	➢	Version log files documenting code development
costar	➢	Main **CoSTAR** function (to be called by the user)

Code Structure

UNIKASSEL VERSITÄT

- Classes — All classes and associated methods
  - @Continuation — Class and methods to perform the continuation
  - @costarhelp — Class and methods for the **`costarhelp`** feature
  - @DynamicalSystem — Class for storing the **`options`** structure
  - @Gatekeeper — Class and methods for the **`Gatekeeper`** feature
  - ApproxMethod_SC — Classes and methods to construct the residuum function
    - @AM_EQ
    - @AM_PS_FDM
    - @AM_PS_FGM
    - @AM_PS_SHM
    - @AM_QPS_FDM
    - @AM_QPS_FGM
    - @AM_QPS_SHM

      (@AM_EQ … @AM_QPS_SHM) — Subclasses and methods
    - @ApproxMethod — Superclass
  - Solution_SC
  - Stability_SC

ENGINEERING DYNAMICS

Code Structure

Classes	All classes and associated methods
@Continuation	➢ Class and methods to perform the continuation
@costarhelp	➢ Class and methods for the **`costarhelp`** feature
@DynamicalSystem	➢ Class for storing the **`options`** structure
@Gatekeeper	➢ Class and methods for the **`Gatekeeper`** feature
ApproxMethod_SC	➢ Classes and methods to construct the residuum function
Solution_SC	➢ Classes and methods to save computed data
@SOL_EQ …	➢ Subclasses and methods (analogous to ApproxMethod_SC)
@Solution	➢ Superclass
Stability_SC	➢ Classes and methods to compute stability
@ST_EQ	➢ Subclass and methods for equililibrium solutions
@ST_PS_SHM	➢ Subclass and methods for periodic shooting method
@ST_QPS_SHM	➢ Subclass and methods for quasi-periodic shooting method
@Stability	➢ Superclass and methods

costar

```
function [S,DYN] = costar(options)

    %% Gatekeeper
    GC = Gatekeeper();
    options = GC.m_gatekeeper(options);
    clear GC;

    %% Dynamical System class
    DYN = DynamicalSystem(options);

    %% Approximation Method class
    AM = ApproxMethod.s_method_selection(DYN);

    %% Solution class
    S = Solution.s_solution_selection(DYN,AM);

    %% Stability class
    ST = Stability.s_stability_selection(DYN,AM);

    %% Calculate initial solution
    [S,AM,DYN] = initial_solution(DYN,S,AM,ST);

    %% Continuation
    if strcmpi(DYN.cont,'on')
        CON = Continuation(options.opt_cont);
        S = CON.m_continuation(DYN,S,AM,ST);
    end
```

- **Input:** options structure
  Output: Solution object S, DynamicalSystem object DYN
- **Gatekeeper** checks all input options
- **Save** all options in DynamicalSystem object DYN
- **Create** ApproxMethod object AM
  (methods construct the residuum function)
- **Create** Solution object S
  (stores all solution data)
- **Create** Stability object ST
  (methods compute the stability of a solution)
- **Compute** the initial (first) solution
- **Create** Continuation object CON
- **Do** the continuation

Basic Use

1. Define important parameters and functions
(not necessarily needed, but it helps to keep the overview)

```
%% 1. Define important parameters and functions (not necessarily needed, but it helps to keep the overview)
D = 0.05;     kappa = 0.3;     g = 1;            % Parameters needed for the Duffing differential equation
mu_limit = [0.01, 2.5];                          % Limits of the continuation
eta0 = mu_limit(1);                              % Value of continuation parameter at start of continuation
param = {kappa, D, eta0, g};                     % Parameter array
active_parameter = 3;                            % Location of continuation parameter within the array
IC = [1; 0];                                     % Initial condition (point in state space) for fsolve

% Functions
non_auto_freq = @(mu) mu;                        % Non-autonomous excitation frequency
Fcn =  @(t,z,param) duffing_ap(t,z,param);       % Right-hand side of dz/dtau = f(tau,z,kappa,D,eta,g)
```

2. Define the options structure
(it comprises all information that CoSTAR needs)

```
%% 2. Define the options structure (it comprises all information that CoSTAR needs)
options.system = costaropts('order',1,'dim',2,'rhs',Fcn,'param',param,'info','continuation of Duffing equation');  % Properties of the system
options.opt_sol = costaropts('sol_type','periodic','approx_method','shooting','cont','on','stability','on', ...   % Properties of the solution
                             'non_auto_freq',non_auto_freq,'act_param',active_parameter);                        % Properties of the solution
options.opt_init = costaropts('ic',IC);                                                                           % Property for initial solution
options.opt_approx_method = costaropts('solver','ode45');                                                         % Properties of approximation method
options.opt_cont = costaropts('mu_limit',mu_limit);                                                               % Properties for continuation
```

3. Call CoSTAR (and do the continuation)

```
%% 3. Call CoSTAR and do the continuation
[S,DYN] = costar(options);                       % CoSTAR is called by costar(options)
```

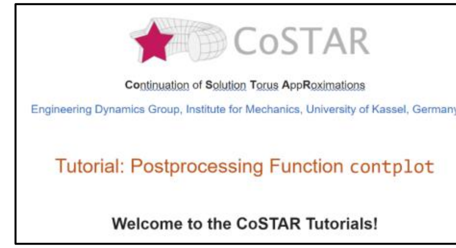
4. Individual postprocessing

```
%% 4. Individual postprocessing
% ...
```

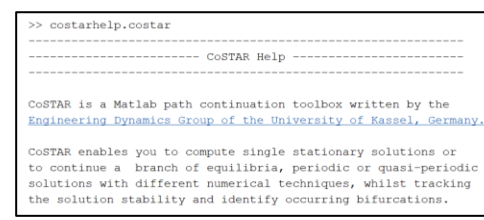
Where To Start

- **If you are new to CoSTAR or certain modules**
  - Tutorials
    - Comprehensive explanations of a certain **CoSTAR** module
    - Currently available:
      - ✓ Equilibrium solutions (`Tutorial_EQ`)
      - ✓ Periodic and quasi-periodic solutions approximated by FDM, FGM and SHM (`Tutorial_PS_FDM`, `Tutorial_PS_FGM`, …, `Tutorial_QPS_SHM`)
      - ✓ Postprocessing methods contplot, solplot and solget (`Tutorial_Postprocessing_contplot`, …)

Start with one of these if you have not used CoSTAR yet



- **If you already used CoSTAR**
  - Examples
    - Sample code showing usage of a certain **CoSTAR** module
    - Good rescue point to restart working with **CoSTAR**
    - There is one example for each tutorial (examples are labelled `Example_[…]`)
  - costarhelp feature
    - Overview of the available options with a short description
    - Quick help in the command window while using **CoSTAR**
    - Type **`costarhelp.costar`** in the command window to start



Outlook

Note: The following list may be incomplete and only lists ideas for future improvements to the toolbox. There is no guarantee for actual implementation.

- **Approximation methods**
  - Multiple Shooting Method for quasi-periodic solutions
- **Features**
  - **Stability computation**
    - **PS:** Directly from solution data when using FDM or FGM without JACOBIAN of SHM
  - **Error control**
    - Finite Difference Method (PS & QPS)
    - Handle different exit flags from `fsolve` when computing new solution with updated discretization
  - **Step control**
    - Algorithm(s) based on convergence of solver (when self-written solver is available)
  - **Postprocessing**
    - **FGM & SHM** (QPS hypertime plots): `[1x2]` array for `options` structure field `'resolution'`
  - **Tutorials & Examples**
    - Update default-script tutorials to live scripts
    - Tutorials and examples for continuation options, step control and stability computation

Outlook

Note: The following list may be incomplete and only lists ideas for future improvements to the toolbox. There is no guarantee for actual implementation.

- **Initial value** (for the solver to compute the initial solution)
  - Standardise the parameters, which create an initial value, for all approximation methods
  - Use of a solution of a different approximation method as initial value
  - Homotopy methods
- **Continuation**
  - Predictor: Polynomials of order > 3
  - Additionally compute the solution at specified (desired) μ-values
- **Computational effort**
  - Make parallel computing available to enhance performance
- **Solver**
  - Self-written solver to remove the need of MATLAB's Optimization Toolbox
- **Dynamic System**
  - Computation of non-hyperbolic manifolds (solutions of Hamiltonian systems)

Feedback and Download

- **Download**
  - CoSTAR is available for free as GitHub repository
  - CoSTAR is licenced under the *Apache 2.0* licence

- **Report of bugs**
  - Please create a GitHub issue, labelled as bug, if you experience a new bug
  - If a GitHub issue already exists for your bug, no action is required

- **Suggestions for improvement, wishes and ideas for future releases**
  - Please create a GitHub issue for any wishes, improvements and ideas for future releases and label it accordingly

CoSTAR

Website CoSTAR

ENGINEERING DYNAMICS

Website Engineering Dynamics Group

Publications

Theoretical basics can be found in following publications:

- FIEDLER, R., HETZLER, H. & BÄUERLE, S.
  Efficient numerical calculation of LYAPUNOV-exponents and stability assessment for quasi-periodic motions in nonlinear systems.
  Nonlinear Dyn **112**, 8299–8327 (2024). https://doi.org/10.1007/s11071-024-09497-9

- HETZLER, H. & BÄUERLE, S.
  Stationary solutions in applied dynamics: A unified framework for the numerical calculation and stability assessment of periodic and quasi-periodic solutions based on invariant manifolds.
  GAMM-Mitteilungen **46** (2023), e202300006. https://doi.org/10.1002/gamm.202300006

- BÄUERLE, S., SEIFERT, A., KAPPAUF, J. & HETZLER, H.
  A continuation framework for quasi-periodic solution branches based on different torus discretization strategies.
  Proceedings of ISMA Conference, Leuven, Belgien, 12.-14. September 2022.

- BÄUERLE, S., FIEDLER, R. and HETZLER, H.
  An engineering perspective on the numerics of quasi-periodic oscillations.
  Nonlinear Dyn **108** (2022), no. 4, 3927–3950. https://doi.org/10.1007/s11071-022-07407-5

ENGINEERING DYNAMICS