Multi-Dimensional Bisection Method (MDBM) User's Guide

Dr. Daniel Bachrathy
Assistant Professor
Budapest University of Technology and Economics

February 2026

Contents

1 Introduction

The Multi-Dimensional Bisection Method (MDBM) is an efficient and robust root-finding algorithm for determining high-dimensional submanifolds of the roots of implicit non-linear equation systems. It is particularly powerful in cases where the number of unknowns surpasses the number of equations ($k \leq l$):

$$f_i(x_j) = 0, \quad i = 1 \dots k, \quad j = 1 \dots l \tag{1}$$

MDBM iteratively halves the parameter space and evaluates only those grid points where a root is likely to exist, significantly reducing the number of required function evaluations compared to brute-force methods.

2 Quick How-To-Use

To use MDBM, you need to define the parameter space, the vectorized function, and the number of iterations.

```
% 1. Define axes
ax(1).val = linspace(-3, 3, 10);
ax(2).val = linspace(-3, 3, 10);

% 2. Define vectorized function
foo = @(x) x(1,:).^2 + x(2,:).^2 - 4; % Unit circle

% 3. Run MDBM
Niteration = 4;
sol = mdbm(ax, foo, Niteration);

% 4. Plot
plot_mdbm(sol);
```

3 Detailed Options (mdbmset)

The behavior of the solver is controlled by the `mdbmset` structure. Each field is critical for fine-tuning the solver's performance and robustness.

Field	**Related Demo**	**Description**
`isconstrained`	Constrained Problems	If true, the last equation in the output is treated as a domain boundary. Solutions are only kept where this value is positive.
`interporder`	Interpolation Order	Order of root interpolation: 0 (raw grid points), 1 (linear), 2 (quadratic).
`bracketingdistance`	Bracketing n-cubes	Distance (in grid units) to look for sign changes. Helps recover branches in dense regions.
`bracketingnorm`	Bracketing n-cubes	The norm used to calculate distance for bracketing.
`checkneighbour`	Neighbor Check	Max consecutive neighbor check steps. Set to `Inf` for max robustness.

`refinetype`	Local Refinement	Refinement strategy: `'all'`, `'pos'` (range), `'object'` (near roots).
`zerotreshold`	Feature: Degenerate	Numerical value below which a function evaluation is considered exactly zero.
`connections`	Simplex Connections	If true, calculates the Delaunay connectivity needed for plotting surfaces and curves.
`timelimit`	Case: Mathieu	Max allowed time (seconds) for a single refinement step.
`funcallimit`	-	Max function evaluations allowed to prevent memory overflow.

4 Software Components (src)

The following files in the `src/` directory form the core of the MDBM ecosystem:

File	Description
`mdbm.p`	**Main Entry Point**. Orchestrates iterative bisection, calling refinement and interpolation.
`mdbmset.m`	**Configuration**. Utility to create the options structure with defaults.
`plot_mdbm.m`	**Visualization**. Intelligent plotting for 1D, 2D, and 3D submanifolds.
`extend_axis.m`	**Domain Extension**. Adds new parameter ranges to an existing solution.
`refine.p`	**Grid Management**. Implements the various refinement strategies.
`checkneighbour.p`	**Robustness Engine**. Scans adjacent grid cells to find missing branches.
`interpolating_cubes.p`	**Root Finder**. Performs root interpolation within a grid cell.
`DTconnect.p`	**Meshing**. Standard Delaunay triangulation for result visualization.
`interpplot.p`	**Smoothing**. Generates high-quality interpolation for plotting.
`plotthecomputedpoints.m`	**Debug Viz**. Visualizes all grid points evaluated by the solver.

5 History

The core idea of MDBM occurred during my MSc thesis (written during an Erasmus stay at the University of Bristol - thank you for that!) in 2006. It has been updated in larger and smaller bursts throughout the last 20 years, leading to my most cited paper.

The primary motivation was the solution of the characteristic equations of Delay Differential Equations (DDEs), which typically involve 2+1 parameters and 2 equations (real and imaginary parts of the characteristic equation). At that time, no direct method existed to solve these automatically in a robust way, especially for detecting closed manifolds of non-differentiable functions. As far as I know, there is still no other method that handles this as effectively.

6 Disclaimer

Note on Protected Files: I would like to protect my work, thus I use `*.p` files for the core engine (some of you may dislike it, sorry). In my opinion, the code is far too long and complex to understand; I spent more than 10 years on it, and sometimes it is hard even for me to understand the logic! (I am a mechanical engineer, not a professional programmer). However, all files that might need modification for special needs are provided as `*.m` files.

Julia Version: A newer version is available in Julia. Although not all features from the Matlab version have been migrated yet (as some are not critical), they have similar capabilities. Currently, only the Julia version is being actively maintained and updated, so great new features will come there first. For instance, sub-cube interpolation is already available, and the error-based non-uniform refinement ("best and most needed feature ever") is already in a usable beta version.

7 Known Alternatives

- **Mathematica (ContourPlot/Isosurface)**: Mathematica performs contour plots by iteratively refining the grid (triangle-based), which is similar to my method. As far as I know, it does this in 3D as well. However, it can generally handle only one equation (which is the "easy" problem).
- **Sequential Projections**: There are versions where you solve the first equation in higher dimensions, then triangulate the result, and solve the second equation on that mesh, and so on. While this creates a robust solution, the complexity is very, very high—almost the same as a brute-force search.
- **Interval Arithmetic**: Some methods provide guaranteed solutions (whereas in MDBM, it is inevitable that a small piece of the solution might be lost in higher dimensions—in contrast to the traditional 1D bisection problem). See the great works of **David P. Sanders** on Interval Arithmetic (e.g., `IntervalRootFinding.jl` in Julia).

8 Folder Descriptions and Gallery

8.1 Features (features/)

8.1.1 Simplex Connections

`connection_of_points_to_simplex.m` shows how points are connected (Figure 1).

8.1.2 Constrained Problems

Handling domain boundaries (Figure 2a and 2b).

8.1.3 Interpolation Order

Comparison of 0th, 1st, and 2nd order (Figure 3).

8.1.4 Local Refinement

Zooming into regions (Figure 4).

8.1.5 Neighbor Check

Recovering missing branches (Figure 5).

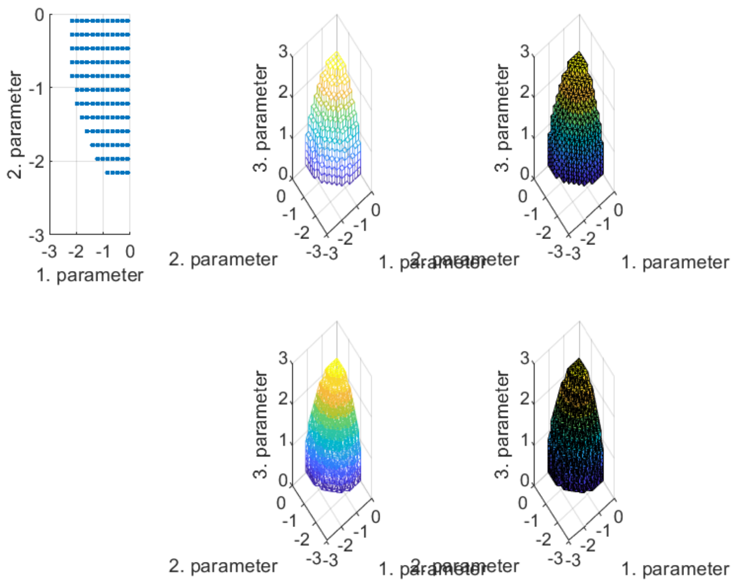
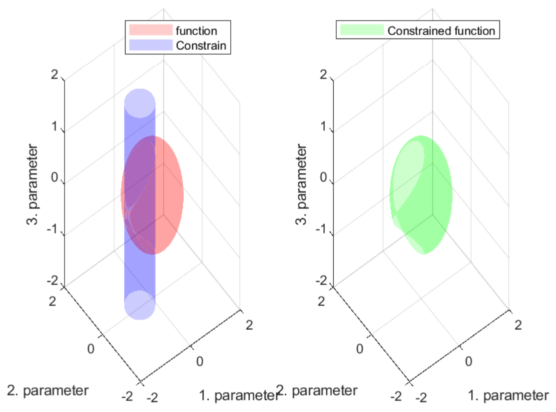
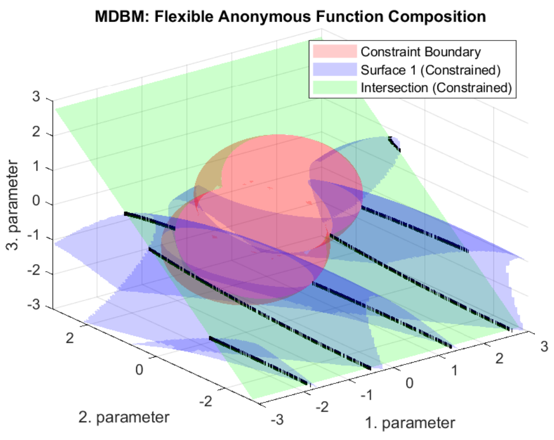


Figure 1: Connectivity of points in the parameter space.



(a) Simple constraint.



(b) Complex combined constraints.

Figure 2: Constraint handling in MDBM.

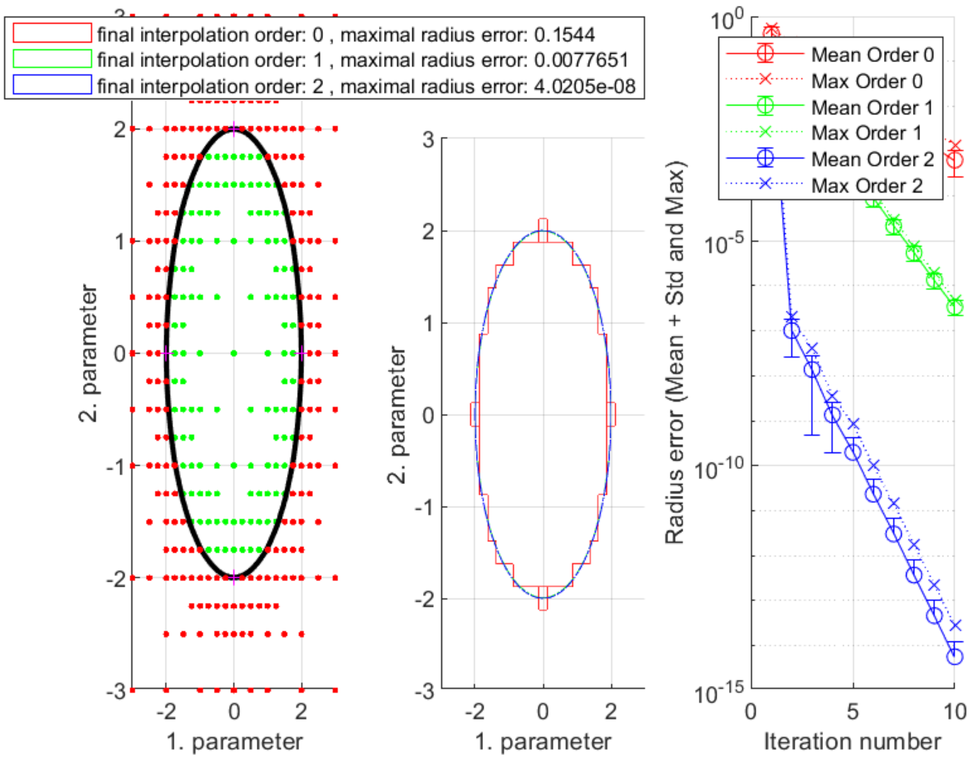


Figure 3: Effect of interpolation order on accuracy and convergence.

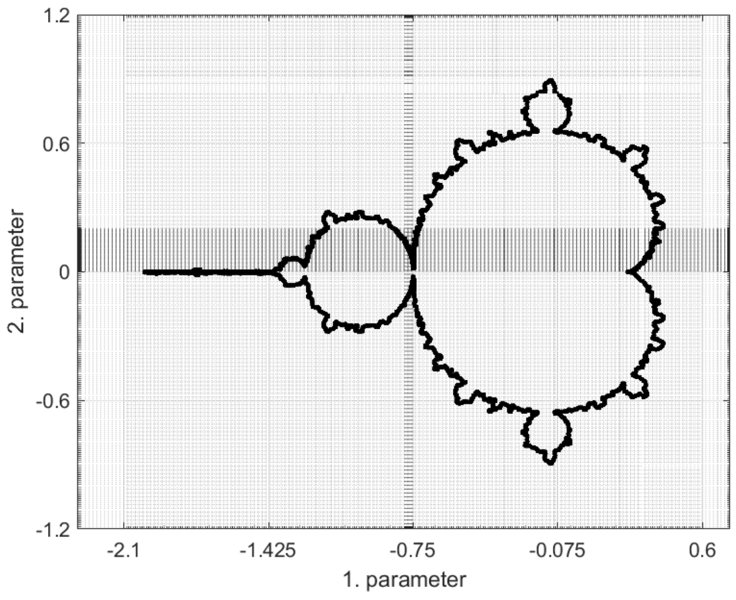


Figure 4: Local refinement of a non-smooth boundary.

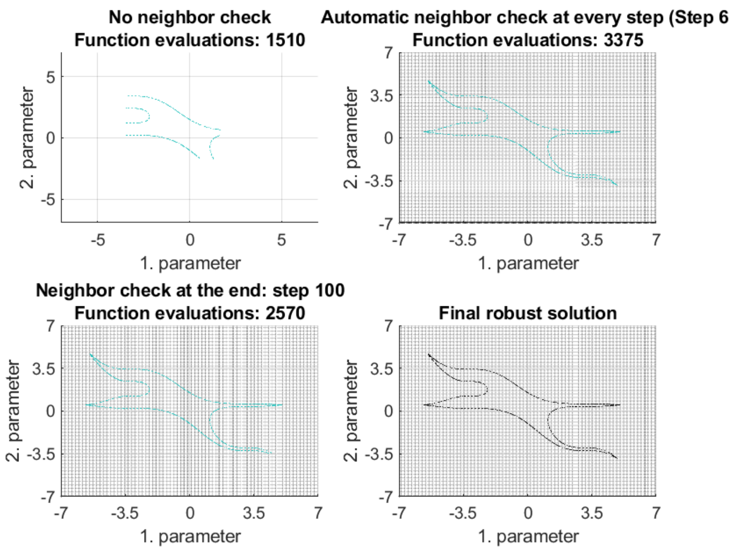


Figure 5: Neighbor check ensuring solution continuity.

8.1.6 Bracketing n-cubes

`bracketing_ncube_detection.m` shows how bracketing cells are identified (Figure 6). This script demonstrates the effect of the `bracketingdistance` parameter by performing a sweep across different values, highlighting the trade-off between capturing disconnected manifold components and the number of evaluated grid points.

8.1.7 Singularity Handling

`singularty_handling.m` illustrates the identification and removal of "false" solutions that arise due to function singularities (Figure 7). Since MDBM relies on sign changes, it may detect boundaries where a function jumps from $-\infty$ to $+\infty$. This script demonstrates how to filter these points by evaluating the function value at the interpolated roots or by using constraints to limit the function's magnitude.

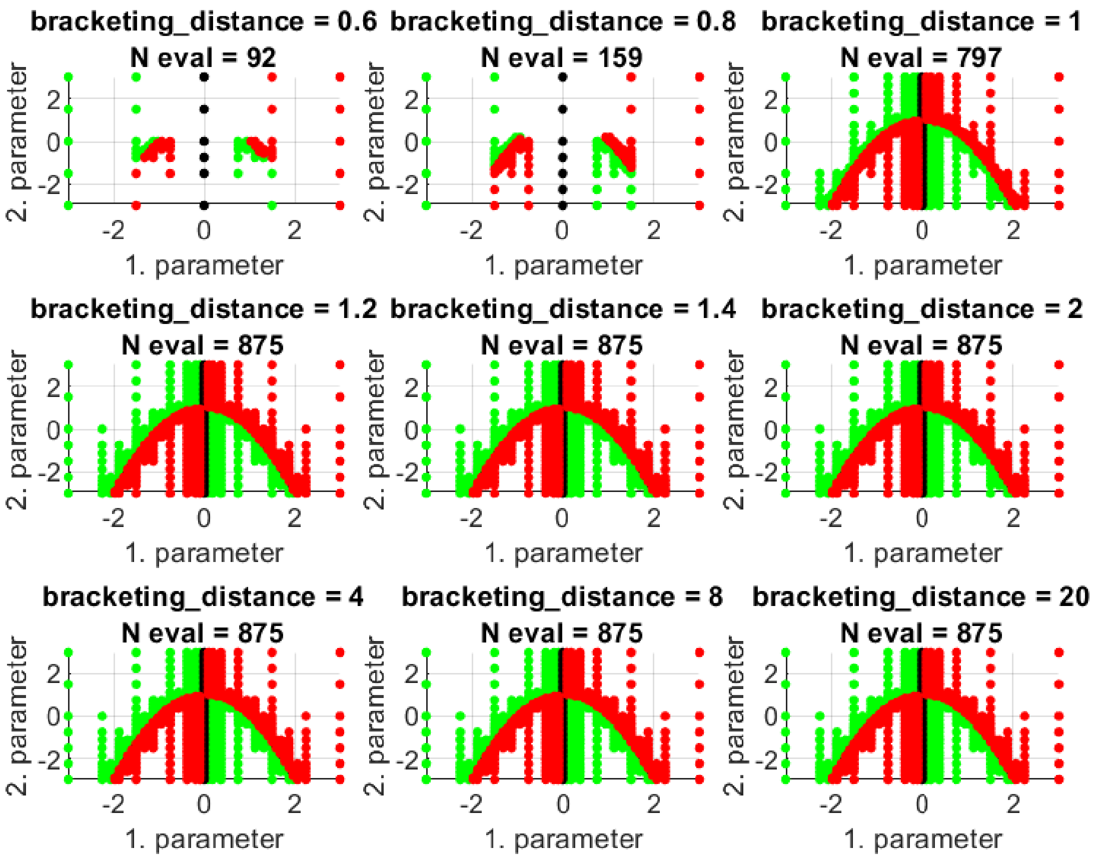


Figure 6: Bracketing n-cube detection with varying `bracketingdistance` values.

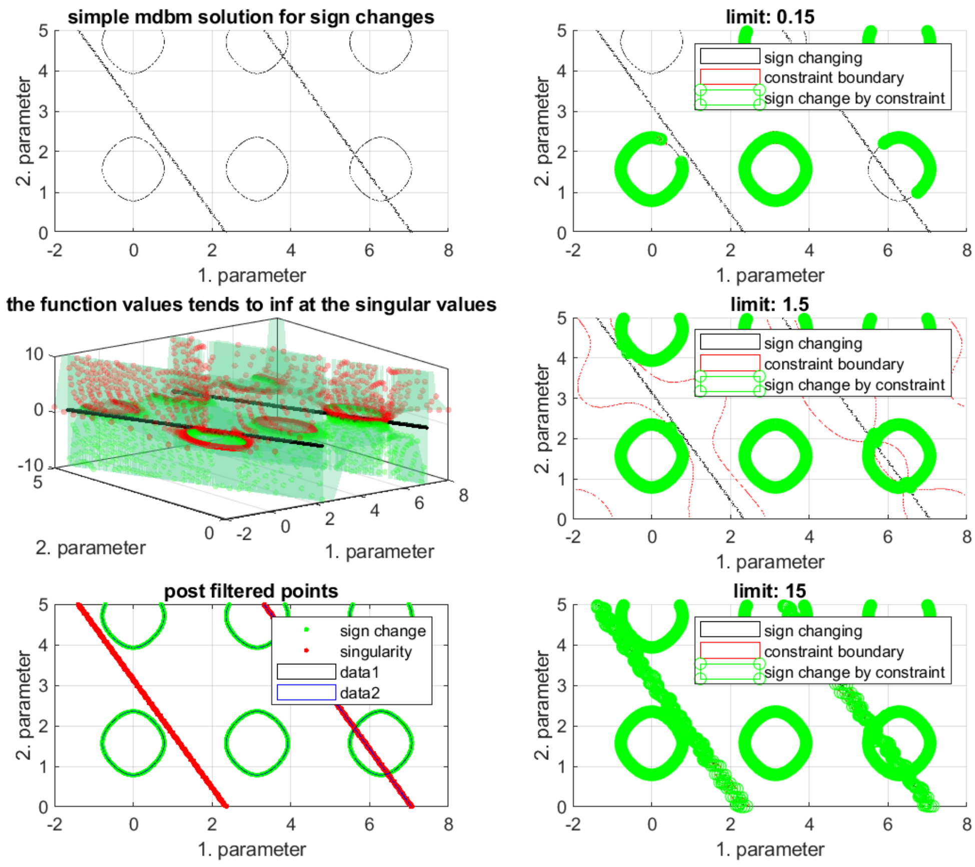
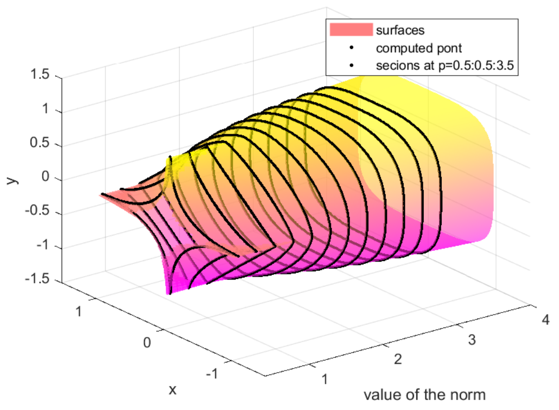
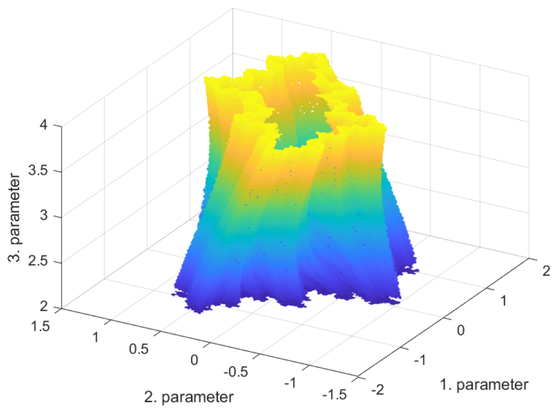


Figure 7: Detection and filtering of singularities.

8.2 Examples (examples/)



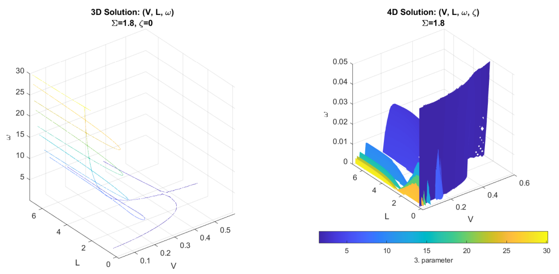
(a) Unit circles (L^p norms).



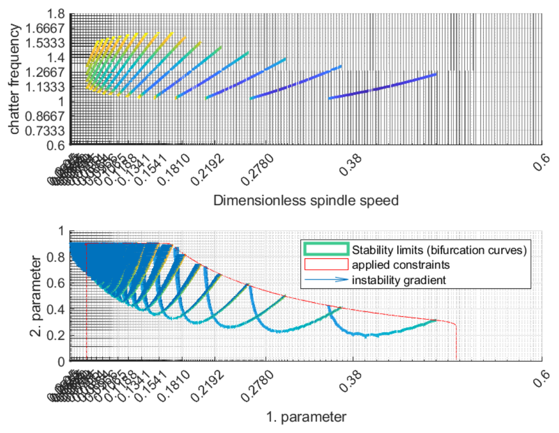
(b) 3D Julia set.

Figure 8: Geometric and fractal examples.

8.3 Case Studies (case_studies/)



(a) Shimmy vibration stability.



(b) Turning stability chart.

Figure 9: Engineering case studies.

9 Citing

If you use MDBM in your research, please cite: *Bachrathy, Daniel, and Gabor Stepan. "Bisection Method in Higher Dimensions and the Efficiency Number." Periodica Polytechnica Mechanical Engineering, 2012.*

A Detailed File Descriptions

A.1 Features (features/)

- `bracketing_ncube_detection.m`: Effect of `bracketingdistance` on manifold discovery.
- `connection_of_points_to_simplex.m`: Visualization of the mesh generation.
- `constrained_problems_simple.m`: Basic usage of the `isconstrained` option.

- `constrained_problems_complex.m`: Composition of multiple constraints and objective functions.
- `continuation_and_extension_1.m`: Extending the domain of a known solution.
- `continuation_and_extension_2.m`: Advanced domain extension for disconnected branches.
- `degenerate_function_1.m`: Handling functions with zero-gradients at roots.
- `degenerate_function_2_detecting_different_dimensions.m`: Detecting manifolds of mixed dimensions.
- `interpolation_of_the_results.m`: Showcase of different interpolation techniques.
- `interpolation_order.m`: Convergence analysis for 0th, 1st, and 2nd order interpolation.
- `local_refinement_nonsmooth_mandelbrot.m`: Targeted resolution increase in complex regions.
- `neighbor_check_demo.m`: Robustness via adjacent cell scanning.
- `plotting_exampels.m`: Comprehensive guide to the `plot_mdbm` function options.
- `singularty_handling.m`: Filtering false sign changes caused by poles.

A.2 Examples (examples/)

- `unit_circles.m`: Visualization of L^p norms in 3D.
- `catastrophe_surface_visualization_gradient.m`: Surface sections and fold lines.
- `convergence_transcendent.m`: Error analysis against iterative refinement.
- `delayed_pd_controler_stability.m`: Stability of systems with time-delay.
- `fun_with_floor.m`: Handling non-smooth (staircase) functions.
- `julia_set_3d.m`: Fractals in 3D.
- `noise_handeling.m`: Robustness against stochastic function values.

A.3 Case Studies (case_studies/)

- `catastrophe_surface/catastrophe_demo.m`: Pendulum catastrophe surface.
- `Claster_Treatment_of_Characteristic_Roots/main_CTCR.m`: Delay-dependent stability analysis.
- `Delay_Mathieu_simulation_based_stability_chart/DelayedMathieuStability_MDBM.m`: Stability of time-periodic delayed systems.
- `shimmy_problem/shimmy_stability`$_{s}urface.m : Wheelvibrationstability.$
- `Sweeping_envelope/sweeping_enveloper_only.m: Efficient envelope detection in parameter sweeps.`
- `turning_stability/turning_stability_instabilitygradient.m: Tool stability in machining.`

A.4 Templates (templates/)

- `template_pardim[N]_codim[M].m`: Standard starting scripts for N parameters and M equations ($N = 1..4, M = 1..3$).
- `superminimal_template_pardim3_codim1.m`: Concise template for high-dimensional problems.