UNIVERSITÀ POLITECNICA DELLE MARCHE, ANCONA, ITALIA
POLYTECHNIC UNIVERSITY OF MARCHE, ANCONA, ITALY

DOCTORAL THESIS

Basins of attraction and dynamical integrity of nonlinear dynamical systems in high dimensions

Author:
Nemanja ANDONOVSKI

Supervisor:
Prof. Stefano LENCI
Polytechnic University of Marche, Ancona, Italy

Co-supervisor:
Prof. Ivana Kovačić
CEVAS, Faculty of Technical Sciences, University of Novi Sad, Novi Sad, Serbia

A thesis submitted in fulfillment of the requirements for the degree of Doctor of Philosophy

February, 2020

Declaration of Authorship

I, Nemanja ANDONOVSKI, declare that this thesis titled, "Basins of attraction and dynamical integrity of nonlinear dynamical systems in high dimensions" and the work presented in it are my own. I confirm that:

- This work was done wholly in candidature for a research degree at Polytechnic University of Marche, Ancona, Italy.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

"Some dreams you know will never come true. But you keep dreaming anyway."

–

POLYTECHNIC UNIVERSITY OF MARCHE

Abstract

PhD School in Engineering Sciences
PhD in Civil, Environmental and Building Engineering and Architecture

Doctor of Philosophy

Basins of attraction and dynamical integrity of nonlinear dynamical systems in high dimensions

by Nemanja ANDONOVSKI

One of the consequences of rich dynamics in nonlinear systems is the multi-stability, i.e. the coexistence of various stable states (attractors), which can have different robustness. Typical example of stable attractors, but very unsafe for practical applications, are those with riddled basins of attraction. To examine and eventually quantify this sensitivity to initial conditions, we can study the shape, size and compactness of basins. However, when the dimension of the dynamical system increases, it is no longer sufficient to analyse arbitrary cross-sections to get the global overview of the dynamics, and a *full-dimensional* basins have to be determined. Due to lack of analytical methods, numerical computations are required to discover basins. Those numerical techniques applied to strongly nonlinear systems, with six or more state-space variables - that are main goal of this PhD thesis - demand considerable amounts of computing power, available only on High-Performance Computing platforms. With the aim to minimize computing requirements, we developed a software to adapt basin computations to small, affordable cluster computers, and with minor modifications utilizable also on larger ones. Our approach is based on Simple Cell Mapping method, modified to reduce the memory requirements, adjust integration time and overcome some discretization drawbacks. The resource intensive parts of computations are parallelized with hybrid OpenMP/MPI code, while less demanding operations are kept serial, due to their inherit sequential nature. The modifications we advised, that address disadvantages of originally proposed method, and the parallelization, are classified under common name - Not So Simple Cell Mapping (NSSCM). As intended, the program computes full six-dimensional basins of attraction within reasonable time-frame, with the adequate accuracy to distinguish compact parts of basins, but not to fully disclose fractal basin boundaries. Visual inspection of results can demonstrate how some basins that look robust in some cross-sections may be very narrow or do not exist in other dimensions, directly altering the robustness of corresponding attractors. Anyhow, the proper robustness analysis must be performed to include entire state-space window. Robustness is quantified by dynamical integrity measures, that exploit previously determined basins by, roughly speaking, computing the largest geometrical objects that can fit inside.

The whole development and computation approach, together with the results in form of basins of attraction and dynamical integrity, are the topic of this thesis, structured in a way to provide full insight into our research development. The first Chapter introduces the High-Performance Computing (hardware and software platforms, algorithm design methodology and overall efficiency), that is followed by the second one which deals with the serial and parallel methods to compute basins of

attraction, coronated with the main effort of our research - Not So Simple Cell Mapping. The third Chapter introduces the concepts and defines the dynamical integrity measures, which closes theoretical part of thesis. Then, on the test system consisting of three coupled Duffing oscillators, the computational efficiency and accuracy of NSSCM approach is discussed and validated by comparison with more accurate, but significantly slower Grid of Starts method.

Final chapters demonstrate applications of NSSCM and integrity analysis on two examples: sympodial tree model with first-level branches; and rotating hub with two pendulums. Dynamics of those strongly nonlinear systems is studied systematically to demonstrate why classical stability analysis is not sufficient to uncover the properties of global behaviour. Integrity analysis and inspection of certain low-dimensional cross-sections of full-dimensional basins showed the complex behavior of those systems, which cannot be determined otherwise.

With the approach described in this thesis, and when sufficient computing resources are available, one can compute dynamical integrity for an interval of relevant system parameter(s) and get the most valuable representation of system safeness - integrity erosion profiles.

Acknowledgements

First of all, I thank my parents Svetlana and Dragan, for their patience and support.

Thanks to Prof. Ivana Kovačić who encouraged me not to be afraid of things never tried before, and taught me the value of dedication.

Thanks to Prof. Stefano Lenci, with whom I learned that there are no such things that cannot be done, only those we choose not to.

And, many thanks go to my friends at archery club and University who made Ancona feel like home.

For the invaluable help with High-Performance Computing, I thank Professors Franco Moglie, Polytechnic University of Marche, Ancona, Italy, and Radu Serban and Dan Negrut from University of Wisconsin, Madison, USA.

Results presented in this thesis are computed under the CINECA ISCRA C-type project IsC66 DIAHIDD, thus we acknowledge the CINECA award under the ISCRA initiative, for the availability of high performance computing resources and support.

Contents

List of Figures

List of Tables

List of Abbreviations

2D	**2**(two)-**D**imensional
3D	**3**(three)-**D**imensional
6D	**6**(six)-**D**imensional
API	**A**pplication **P**rogramming **I**nterface
CH	**CH**aotic
CL-C	**CL**uster **C**ineca
CL-U	**CL**uster - **U**niversità Politecnica delle Marche
CM	**C**ell **M**apping
CPA	**C**onnecting **P**ost-processing **A**lgorithm
CPU	**C**entral **P**rocessing **U**nit
CUDA	**C**ompute **U**nified **D**evice **A**rchitecture
FDO	**F**orced **D**uffing **O**oscillators
FLOPS	**FL**oating point **O**peration **P**er **S**econd
GB	**G**iga **B**yte
GHz	**G**iga **H**er**z**
GIM	**G**lobal **I**ntegrity **M**easure
GPU	**G**raphical **P**rocessing **U**nit
GoS	**G**rid **o**f **S**tarts
HPC	**H**igh-**P**erformance **C**omputing
IF	**I**ntegrity **F**actor
LIM	**L**ocal **I**ntegrity **M**easure
MIMD	**M**ultiple **I**nstruction **M**ultiple **D**ata
MPI	**M**essage **P**assing **I**nterface
NC	**N**umber of **C**ells in cell space
NSSCM	**N**ot **S**o **S**imple **C**ell **M**apping
ODE	**O**rdinary **D**ifferential **E**quation
OpenMP	**O**pen **M**ulti-**P**rocessing application programming interface
OpenCL	**Open C**omputing **L**anguage
PR	**P**e**R**iodic
PR1	**P**e**R**iod**1** (one)
PR3	**P**e**R**iod**3** (three)
QP	**Q**uasi-**P**eriodic
RAM	**R**andom **A**ccess **M**emory
RHUB	**R**otating **HUB** (with two attached pendulums)
SCM	**S**imple **C**ell **M**apping
SIMD	**S**ingle **I**nstruction **M**ultiple **Data**
STREE	**S**ympodial **Tree** (model with first level branches)
TB	**T**era **B**yte

Chapter 1

INTRODUCTION

For engineering applications where change of some initial conditions may occur during the exploitation period (due to disturbances like noise, shocks, etc.), it is particularly important to determine the robustness of the steady state behaviour, which is governed by the size and shape of the regions where the basins of attraction (in further text simply **basins**) are compact. Quantitatively, how much basins are robust can be detected with several dynamical integrity measures [1–5].

The classical stability analysis methods [6–8] allow only to determine the *local* behaviour of the system dynamics, results that the basins are the key tool to determine *global* behaviour. Due to their complexity, the basins and derived integrity measures can be computed only numerically [9], in most cases. In this respect, trends are to determine them for strongly nonlinear systems with, at least, six state-space variables (dimensions). The increased dimensionality causes that the numerical computations become highly demanding for computer resources [9–12]. Thus, the contemporary research efforts are focused on developing methods for the exploitation of High Performance Computing (HPC) resources, as conventional computers cannot cope with such large-scale computing tasks.

For certain integrity measures (e.g. the Local Integrity Measure [4]), the fractal parts of the basins [13] can be overlooked, because the only relevant property is the distance from an attractor to the nearest point on the basin boundary. This is a consequence of the fact that in these regions the dynamics are practically unpredictable and thus they do not contribute to the robustness of the steady-state attractor (i.e., they do not belong to the safe basin, which is the compact subset of the basin that can be safely used in practical applications). Therefore, for basins computation oriented toward dynamical integrity analysis, there is no need to determine rigorously these regions of the phase-space. Under such conditions, it is possible to consider the full-dimensional basin of attraction computations with affordable HPC platforms, without resorting to expensive hardware. The accuracy of the computed results is rough, but can give meaningful information on the general structure of the volume occupied by the relevant basins. The computation and analysis of the full-dimensional basins is an aspect of nonlinear systems that has not yet been sufficiently developed. The value of basins for better understanding of dynamics is recognized and their analysis constitutes the main goal of our research.

Therefore, the first part of work presented in this thesis is dedicated to develop a software that can be used by scientists and engineers who have access to affordable HPC solutions and need to examine basins of attraction. It is envisioned to compute crude approximation of basins and to prepare the data for numerical analysis and visualization. The resulting approach is developed by the systematic examination of various numerical methods for the computation of full-dimensional basins. It offers balance between computing requirements and accuracy of basins, at low

development costs - which is a distinguished feature of the proposed approach and the second goal of the work.

Visual inspection of results can demonstrate how some basins that look robust in some cross-sections may be very narrow or not exist in other dimensions, directly altering the robustness of the corresponding attractors. Anyhow, the proper robustness analysis must be performed to include the entire considered part of state-space. Numerical measures of dynamical integrity (that exploit previously determined basins) quantify the robustness by computing the largest compact geometrical objects that can fit into the compact parts of basins. Those objects are defined by the distance metrics [14] used to measure the distance between discretization units (cells) and basin boundaries. To reduce computational efforts, we employ chessboard (Chebyshev) distance to get the edge length of hyper-cubes, instead of much more resource demanding computation of hyper-spheres with Euclidean distance - which is the last goal we intend to discuss in this thesis. In cases when sufficient computing resources are available, with the approach described in this thesis one can compute a dynamical integrity measure for the interval of system parameter(s) values and get the most valuable representation of system safeness - integrity erosion profiles.

The reader is invited to acknowledge this thesis as a summary of our work previously published in [15], [16] and [17], expanded with some additional content to provide full insight to our research and analysis process. The chapters are structured in such way to explain all theoretical requirements to understand and apply the methods used.

Foremost, the results presented herein are based on the computations on a large scale, and to understand the computing process, in Chapter 2 the reader is introduced to the concepts of High-Performance computing. Due to the variety of HPC platforms and basin computation methods, each with distinct advantages and drawbacks, we analyse the parallel computing architectures, their performance and program design methodology. Also, the programming models for two major platforms, clusters and computational graphic cards (GPU) are presented briefly.

In Chapter 3, we debate global analysis by the compatibility of basin computation methods with various HPC platforms. Upon establishing the necessary foundations, we proceed to the main part of our research - computation of full, six-dimensional basins of attraction. As there is never-ending discrepancy between accuracy and computational efficiency, we examined how to get useful results within reasonable time-frame. Considering the computational platforms that were available to us, we discuss the optimal method to compute basins on small clusters - The Not So Simple Cell Mapping (NSSCM). Moreover, it is described and demonstrated how NSSCM can be used also on larger clusters, by introducing the minor modifications.

Some applications of basins for dynamical integrity analysis is presented in the Chapter 4, where we define integrity measures and their computation process.

Within the Chapter 5, the computational efficiency and accuracy of NSSCM approach is discussed and validated. The NSSCM results are compared to those obtained with the method of Grid of Starts, which is more accurate but significantly slower method. The confrontation of results is performed on the test system consisting of three coupled Duffing oscillators [18].

Chapter 6 with certain examples demonstrates the applications of NSSCM and integrity analysis on two systems: a sympodial tree model with first-level branches [19]; and a rotating hub with two pendula [20]. Dynamics of those strongly nonlinear systems is studied systematically to demonstrate why classical stability analysis

is not sufficient to uncover the properties of their global behaviour. Integrity analysis and inspection of certain low-dimensional cross-sections of full-dimensional basins showed the complex behavior of those systems, which cannot be determined otherwise.

Final remarks, short summary of our work and potential future developments are discussed in the closing Chapter 7.

Chapter 2

HIGH-PERFORMANCE COMPUTING

In the recent decade, very powerful computing platforms became accessible to the scientific and engineering community. The complex problems that do not have closed form solutions, can now be directly solved numerically, by sheer force of super-computers. However, it is not an easy task to handle such power. Basin computations suffer from the *dimensionality curse*, which is an exponential growth of computational time caused by the increase of the problem dimensionality. This Chapter 2 is therefore dedicated to provide understanding of HPC platforms and programming models, which is absolutely necessary to understand the extent of computing problems caused by dimensionality, and how to face them. Without knowledge described herein, anybody who is not specifically educated in HPC programming would certainly be caught into one of the numerous traps of parallel computing.

2.1 High-Performance platforms

A general idea to speed-up the computations is to connect multiple computing entities into a network that acts as a single system, which can run one or more programs that are divided into numerous parallel tasks. This concept and its various implementations are usually referred as multiprocessors. In most cases, this type of massive parallelization satisfies the requirements necessary to compute most problems encountered in scientific and engineering applications. The parallelization aims to speed-up the serial execution by a dividing either instructions or data between concurrent processing units.

To efficiently carry out parallelization process of an algorithm, it is necessary to determine what type of parallelization can be achieved. *Task parallelism* [21, 22] occurs in cases where a sequence of instructions can be divided into multiple, parallel and independent operations - each task may run same or different code over same or different data. A counterpart situation, the *data intensive* computations are those where large number of data elements have to be processed in the same way [21, 22] - one sequence of instruction is executed over multiple parallel data elements.

Another analogous way to look at parallelism is through logical concepts of stream concurrency defined in Flynn's taxonomy [23–25], which coincide with multiprocessor organization. Namely, a program is a sequence of commands/instructions executed by processors - the *instruction stream*. The flow of data that is being manipulated by commands from an instruction stream is called a *data stream*. The execution model of the sequential computer is then referred as Single Instruction, Single Data stream (SISD), with the program flow like in Fig. 2.1a. In this manner,

the data parallelism is referred as Single Instruction, Multiple Data streams (SIMD), schematically presented in Fig. 2.1b. The Multiple Instruction, Single Data stream (MISD) model (Fig. 2.1c) is rarely used in practice, only for some special type of problems, like error control [23–25]. Although the MISD is an “opposite" concept to SIMD, it is not the equivalent of task parallelism. At various degrees, majority of actual programs and computation problems are not strictly data or task parallel. The problems described with Multiple Instruction, Multiple Data streams (MIMD) often combine parallelization on both data and instruction levels [26], illustrated in Fig. 2.1d. Certain problems, such as Fast Fourier transform or some sorting procedures, also benefit when a program can switch between data and task parallel execution models during runtime.

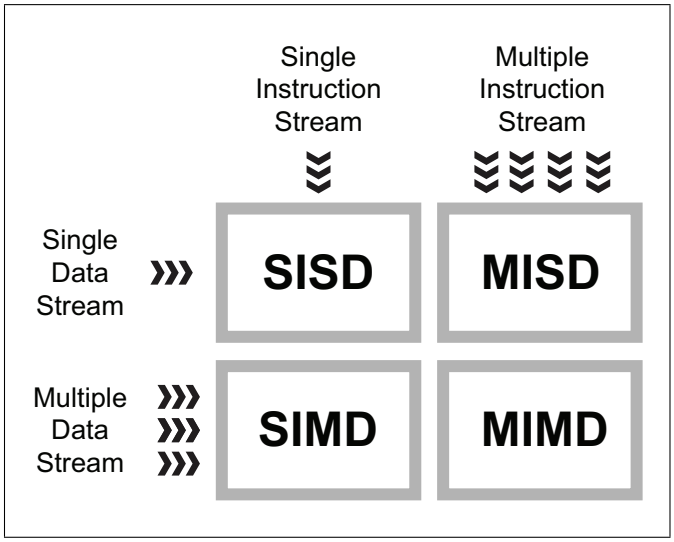

FIGURE 2.1: Flynn’s Taxonomy.

The hardware implementations of multiprocessors, which can efficiently execute data intensive computations are *general purpose graphics cards*. Multi-purpose platforms called *clusters* are much more versatile and can handle well the task parallel and MIMD execution models, and to a certain degree also the SIMD computations.

2.1.1 Cluster computers

Clusters are computers formed from a large number of closely centralized nodes connected with local high-speed interconnections. A node is an autonomous part of a cluster, which can be viewed as a stand-alone computer consisting of at least one CPU and its private Random Access Memory (RAM). Nodes can have multiple cores, CPUs and even a GPU attached to it. The CPUs of one node cannot access to memory of another node directly - data have to be exchanged through proper communication channels. This memory model is called distributed, therefore clusters are *distributed memory computers/systems*. Modern clusters often have multiple cores, within each node, which share common local memory. Cluster implementations with multi-core nodes are highly versatile computing platforms and their number and computing power increase each year. The list of currently most powerful clusters is available in [27].

2.1.2 General purpose graphic cards

Graphical image processing is a very data intensive operation [22, 28]. Accelerator graphic coprocessors are being intensively used to aid CPU in visualizing the graphic content. When CPU encounters a graphically intensive part of the program, it forwards data and instructions to GPU that have an electronic circuitry optimized for efficient image processing. Price for being specialized for certain tasks is that GPU is highly inflexible and unable to function on its own.

A special branch are Computational GPUs (or General Purpose GPUs) that utilize a high number of simple cores (stream processors, shaders) which are able to conduct general computations, while retaining high performance in data intensive applications [28, 29]. Graphical coprocessors are fabricated on a separate (graphic) card, supplied with high-performance RAM due to high demand for data.

In this thesis, when mentioning GPU, we will refer to the general purpose graphical coprocessor cards.

2.1.3 HPC platforms at our disposal

For our research, we had two HPC platforms available for exploitation. The first, smaller one is a part of University cluster with 16 single-core nodes (Intel® Broadwell, 2.4 GHz) with 2 GB RAM each. It was mainly used during a development period, as the computations with it is outside reasonable time-frame for systems where transient behaviour is either long or have high amplitudes.

When the software was completed, we had submitted a project proposal to get the access to CINECA [30] resources. The proposal was approved and the basins and integrity measures presented in this thesis are computed under the ISCRA C-type project IsC66 DIAHIDD. During the project life-time, we had access to the MARCONI A2 [31] cluster, composed of 3600 nodes, each with one Intel® Xeon Phi7250 (KnightLandings), 1.4 GHz processor (68 cores) and 96 GB of RAM.

For brevity, in the following text, the University cluster will be referred as *CL-U* and CINECA one as *CL-C*.

2.1.4 Performance of multiprocessors

A principal matter for high-power computing systems is how well extensive tasks can be accomplished. However, for high performance, it is not enough to have parallelization on massive scale [21, 24, 25, 28, 29, 32–34]. The utilization of resources depends on compatibility of algorithm with the hardware platform, implementation of algorithm, scheduling of tasks, rate of data transfers, etc. Relevant performance measures and most important factors that can downgrade performance are discussed in the following section. In cases of downgraded performance, the reader is referred to literature to get more precise information on how to detect and resolve particular issues.

2.1.4.1 CPU and GPU performance

When comparing the performance of processing units, usually their speed is a relevant factor. Unfortunately, the term speed of a processor does not have a determinate meaning. It may refer to several measures that quantify some of the processor characteristics. The reason for this is the rich complexity of micro-architectures that

differ even between processors of the same manufacturer and family. Current technologies of manufacturing processors with multiple cores, additionally complicates comparison of different processors.

The number of instructions per second [22, 24] is one of the measures used, but for processors with complex instruction set it is not an accurate measurement. Instructions have unequal size, resulting in variable execution time for each instruction. A more common and precise speed measure is the number of system state updates per second, namely *clock speed* or *frequency* (GHz) [22, 24]. A clock speed also does not give definite comparison quantities, since some processors may finish certain actions in fewer clock cycles than others. Thus, even equal clock speeds of different processors do not guaranty that they will do an equal amount of work.

Benchmark programs are used to get somewhat accurate comparison of performances for various systems or components. They may test performance of overall systems or a single component in various workload situations. In this way, processors can be compared on how well they perform at certain task. For scientific computing where data is mainly floating-point numbers, a measure how well system (or single component) perform is represented through floating-point operations per second (*FLOPS*) [21, 22, 24].

Graphical coprocessors are highly specialized hardware and their high performance limits the flexibility. Metrics for GPU performance are same as for processors, but direct comparison of processors versus general-purpose GPUs is vague because each is designed to be efficient at different type of tasks [22, 29].

2.1.4.2 Data transfer performance

Drops of overall computer performance is often caused by data transfer bottlenecks, where data is not supplied fast enough or in low quantities required for high percent of utilization of computing resources. Performance of data transfer is governed by two factors, *latency* and *channel width* [24]. Latency is time in seconds that takes to access the data. Width is the amount of information that can be transferred at once. Combined, is a measure called *bandwidth* that quantifies transfer capacity by the amount of data that can be transferred per unit of time.

2.1.5 Performance issues

Parallelization consequences, hardware restrictions and programming approaches influence performance, correctness of results and program propagation. Beside general cases, certain issues manifest only in operations with local memory, others during remote data communication.

2.1.5.1 Overheads

Overhead [22, 28, 33] is a common name for conditions that avert a processor from actual computations. Algorithmic overheads or excess computations are parts of the program that are difficult to or cannot be parallelized at all. In those cases, a parallel algorithm can be either much more complex than sequential or it must be run sequentially on only one processor. Communication overhead (interprocess interaction) is time that processors spend on data transfer instead of computations. It includes reading, writing and waiting for data.

Idling is a state of processing elements at which no useful computations are done. Reasons for idle time of processors might be synchronization requirements, overheads or load imbalance.

2.1.5.2 Bottlenecks

A situation where a processor efficiency downgrades as a result of insufficient memory bandwidth is called *memory wall* [22]. Computations where performance is constrained by transfer rates of memory are referred as *memory bound* [33]. In multiprocessor systems, bottlenecks [21, 24] can also be caused by poor load balance, where the whole system waits idle for one processor to finish computations.

2.1.5.3 Dependencies

Often for certain computations to advance, data from previous iterations is required. Such cases are called data dependent [21]. Parallelization is interrupted by the task that must wait for data to continue. Other types of dependencies exist, but are not significant from scientific programmer perspective.

2.1.5.4 Result preservation

In parallel program, order of execution is changed in comparison to sequential one [34]. Some operations that are sequentially executed one after another, now may be executed parallelly. Order of execution can cause change of accuracy on some systems due to number truncation or rounding off on different places in program. To avoid such errors, it is required to check if results are consistent during change in order of execution.

2.1.5.5 Synchronization errors

A very common type of errors manifest with shared memory when threads are not time synchronized [34]. A program executes correctly, since there is no actual bug, but the results may be incorrect. Timing of threads impacts if those errors happen or not, even in cases when synchronization is not explicitly imposed.

The first type of incorrect results may occur when one thread starts to work on data from another thread that have not yet finished with computations. A barrier can be instructed to ensure that the thread will wait for another one to finish.

Result inconsistency can also happen when a thread is scheduled to work on the part of data already processed by another thread. To avoid this issue, a programmer can explicitly protect access to data of another thread.

Race condition is a conflicting state when multiple threads try to simultaneously update the same variable. Without access synchronization, a variable update may not be as it is intended. To resolve race conditions, reading and writing to a shared variable should be enclosed in a critical section that permits only one thread at time to manipulate data. Critical sections are implemented as *atomic instructions*. One atomic instruction is a sequence of instructions that manipulate the shared variable. At any given time, only one atomic instruction is allowed to be executed by a whole computer system, and it must be executed entirely before any other instruction (breaking the parallel execution).

2.1.5.6 Variable scope issues

Certain variables may be declared to be shared between all threads or private to each thread. If declared incorrectly, the shared variable may not be updated when necessary and private may be updated by a wrong thread [34].

2.1.5.7 Program propagation impediments

In *deadlock* situations, two or more processes keep waiting for a resource from each other and none of them can make any progress [21, 28, 29]. The resource in those cases can be a message send/receive operation, synchronization instructions or access to remote device or memory. Conditions under which deadlocks happen are when the resource is mutually exclusive (only one process may use it), when process that use the resource requests another resource (hold and wait condition), in cases when the resource cannot be released without a process action (no pre-emption condition) and when multiple processes in a circular chain wait a message from another one.

Livelocks are similar situations, where two or more processes fail to progress because they indefinitely keep responding to resource requests, without actually granting access to the resources [22, 35].

The case when one process sends a message to another process that continuously denies it or when the process constantly tests if a condition is satisfied [35], is called *busy waiting*. The first process keeps resending the message or checking the condition state and cannot continue with useful work.

In cases where processes are scheduled by another entity, *starved process* [22, 35] is one that is ready to continue, but the scheduler ignores it.

2.2 Parallel program design

In multi-core environments there is no certainty that some parallel operations will be executed at exactly same time or in the exactly specified order without losing parallelization or introducing idle waiting time (explicit barriers). For example, in a parallel program, one core might supply certain data to another core too late or too early. The program code does not report any error (since there is none), yet it produces an incorrect output due to the loss of data coherency or processors stay unnecessary idle. Order of thread execution is scheduled by the operating system, and it can be different on each runtime. Consequently, a parallel program does not have the same order of instructions on each runtime, making the de-bugging process difficult [21, 22, 28, 29, 32]. Step-by-step debugging and data tracing is obviously not a feasible solution for checking errors and flow of parallel programs. Therefore, significantly more attention must be invested in a methodical design to prevent parallelization issues, mentioned in Section 2.1.5. The design methodology will be described in order to minimize parallel program flaws and bugs due to a common bad practice during programming while maintaining decent level of simplicity and efficiency [29, 32]. A more detailed approach to design of parallel algorithms may be found in [21, 25, 28].

2.2.1 Design methodology

To get from a problem specification to an effective parallel algorithm, it is crucially important to rely on design methodology and not on pure creativity of a programmer. Anyway, creativity is of great importance even when following a methodical approach. It allows to increase the range of the options considered, distinguish bad alternatives from good and to minimize backtracking from bad choices. Design flaws easily compromise parallel program performance. As most of (parallelizable) problems can be parallelized in multiple ways, the methodical design helps to characterize most favourable solution [21, 28, 29, 32, 33].

2.2.1.1 Partitioning - problem decomposition

The first step before starting to design a parallel program, is to determine if a problem is inherently parallel by its nature or if it may be parallelized [21, 32, 33]. Partitioning serve to recognize parallelization opportunities in the problem. Data and operations are decomposed into smaller (independent when possible) tasks. If decomposed tasks are not independent, they have to communicate data according to dependencies.

Domain decomposition is a technique where the data set of the problem is divided into independent pieces. Next step is to associate tasks to the partitioned data set. A complementary technique, the *functional decomposition*, focuses on dividing the computations into smaller disjoint tasks. In case when the decomposed tasks correspond to the partitioned data, the decomposition is complete. By nature of many problems, this is not possible. In those cases, the replication of data or task set (creating a private copy of "problematic" entity for each parallel process) must be considered. Even when it is not necessary, it might be worth to replicate data or instructions to reduce communication.

Before proceeding to the next steps of the parallelization, consult the following guidelines to ensure that there are no obvious design flaws:

1. to increase flexibility in the following design stages, there should be at least one order of magnitude more tasks than processors in the system;
2. consider both decomposition techniques and identify alternative options;
3. scalability can be compromised with larger problems if there are redundant computations or data input/output after partitioning;
4. it is hard to allocate the equal amount of load to each processor if tasks are not comparable in size;
5. to properly scale, with the increase in a problem size, the number of task should grow, rather than size of an individual task.

2.2.1.2 Communication design

A flow of data is specified in the communication stage [32] of design. In general, tasks can execute parallelly, but it is rare that they are independent. Proper communication structures are required to efficiently exchange data between parallel tasks. A goal of the communication design process is to allow efficient parallel execution, by acknowledging what communication channels and operations are required and eliminating those which are not necessary. Communication channels for parallel algorithms obtained by functional decomposition correspond to the data flow between

tasks. For domain decomposed algorithms, the data flow is not always straightforward, since some operations might require data from several tasks.

Local communication structures are used when a task communicate only with a small number of neighbouring tasks. Global communication protocols are more efficient when many tasks communicate with each other. Communication networks can be structured, where tasks form a regular composition and unstructured where the task are arbitrarily arranged. If an identity of communication pairs varies during the program execution, communication is dynamic and for unchangeable identities, it is static. In synchronous communication information exchange is coordinated, but for asynchronous communication structures, data is transferred with no mutual cooperation.

The following check-list is proposed to avoid overheads and scalability issues arising from an inefficient communication layout:

1. for a scalable algorithm, all tasks should perform a similar number of communication operations;
2. when possible, arrange the tasks so that global communication can be encapsulated in a local communication structure;
3. evaluate if communication operations are able to proceed parallelly;
4. evaluate if tasks can execute parallely and if communication prevents any of the tasks from proceeding.

2.2.1.3 Agglomeration

One of the principal requisites for an efficient execution is a level of matching between hardware and software. In an agglomeration stage [32] the algorithm from previous phases is adapted to be homologous to the computer system used for computation. It is known that it is useful to combine (agglomerate) a large number of small tasks into fewer task larger in size or to replicate either data or computation. Reduced number of tasks or replication can substantially reduce communication overheads.

A revision of parallel algorithm attained by a decomposition and communication design phase can be optimized by the following agglomeration procedure:

1. reduce communication cost by increasing task locality (ratio of remote/local memory access);
2. verify that benefits outweigh the costs of replication or limit scalability;
3. a task created by agglomeration should have similar communication costs as a single smaller task;
4. evaluate if an agglomerated algorithm with less parallel opportunities execute more efficiently than a highly parallel algorithm with greater communication overheads;
5. check if granularity (size of tasks) can be increased even further, since fewer large task are often simpler and less costly;
6. evaluate modification costs of parallelization and strive to increase possibilities of code reuse.

2.2.1.4 Mapping

The final stage of the design is to decide how to map task execution on processors [32]. Since there is no universal mechanism to assign a set of tasks and required communications to certain processors, two strategies are used to minimize execution time. The first option is to map tasks to different processors in order to increase a parallelization level. Other option that increase locality, is to map tasks that communicate often to the same processor/node. Those strategies are conflicting and a trade-off must be made to achieve optimal performance. A favoured strategy is problem specific and use of task-scheduling or load balancing algorithms can be used to dynamically manage task execution.

2.2.2 Design evaluation

Before starting to write the actual code, a parallel design should be briefly evaluated according to performance analysis criteria discussed in [32, 33]. For example, some simple performance analysis should be conducted to verify that the parallel algorithm meets performance requirements and that is the best choice among certain alternatives. Also, to be considered are the economic costs of implementing and possibilities for future code reuse or integration into a larger system.

2.2.3 Software solutions

The packages presented herein are not only solutions available on the market, but are widely used and supported by user community and developers. For larger jobs it might be useful also to get familiar with load balancing, task scheduling and management [36]. For other nonlinear phenomena, not considered in this thesis, e.g. *Computational Fluid Dynamics*, there are already well-developed software and customizable packages. We consider solutions for HPC programming in C/C++ and Fortran for implementation of the basin of attraction and integrity measure computations. Other parts of a global analysis and visualization of results is done with mathematical environments.

2.2.3.1 OpenMP

OpenMP (Open Multi-Processing) [28, 29, 37] is an API that supports programming with shared memory in C, C++ and Fortran languages. It offers an intuitive, multi-threading method of parallelization, where one main thread (master) forks when a parallelizable part of a code is encountered. The work is then divided among a number of secondary (slave) threads. There can also be multiple levels of forking. Threads of same level execute same code over a designated portion of total data. It is usually used in combination with other parallel software when is possible to parallelize work inside nodes.

2.2.3.2 Message Passing Interface

MPI [28, 29, 38] is a standard that defines syntax and semantics of library routines used for writing message-passing programs in C, C++ and Fortran. It operates on a variety of parallel architectures, but is major standard for programming of distributed memory systems, such as clusters. A message passing with MPI is not so intuitive approach to parallel programming and requires more attention than multi-threading approach with OpenMP.

Parallelization is achieved by creating one master and numerous slave tasks (*ranks*) at program runtime. Each rank runs own instance of a MPI program. Within the code, it is specified what parts are executed or skipped by certain ranks. In this way, each rank has own instance of data structures that are not shared with other ranks (although the structures are declared under the same name). To access some remote data, the rank have to explicitly request it.

The core of MPI is based on communication by passing messages between ranks. The simplest form of information exchange is by send/receive operations. One rank would request some data, other rank has to acknowledge this request and send required data back. The first rank then has to appropriately receive the message containing the requested information.

Beside point-to-point communications as send/receive, there are collective communication operations, where all (chosen) ranks participate. When large number of ranks have to exchange data, it is much more efficient to use collective message passing.

Synchronization of execution can be explicitly imposed by instructing barriers or implicitly by using blocking communication. Neither rank is allowed to proceed until all ranks execute the explicit barrier instruction. Blocking communication prevents receiving rank to continue until the message is received. Asynchronous communication can be achieved by using a non-blocking message passing or by using probe instructions. With probe instructions, ranks check if there is a pending message. When the message is there, a probing rank receives it, when not, rank continues with execution.

A message exchange is done within a communicator structure that defines communication privileges. It is used to specify what ranks will participate in certain communication operations.

A technique often used, also by us, is *hybrid programming*, where MPI ranks are used for inter-node parallelization and OpenMP threads for intra-node forking.

2.2.3.3 CUDA and OpenCL

Compute Unified Device Architecture (CUDA) [28, 29, 39] is programming environment developed to efficiently map a data parallel task to a GPU structure. The GPU program is separated in parts run by CPU (*host*) and data intensive functions (*kernels*) that are executed on GPU (*device*).

Beside memory allocation that hold transfer of data between CPU and GPU memories, a programmer has to specify how threads are organized inside the kernel. The kernel *grid* is organized in two levels. The top level is organization of thread *blocks* within the grid. On the second level threads are arranged inside a block. Each block of the same grid has the same number and structure of threads. Latest GPUs support three-dimensional organization of threads within the block. Execution configuration of the kernel is further divided into smaller units – *wraps*, which represent the collection of threads which executes at once. A mechanism called thread *scheduler* decided which wrap will be executed. This execution model efficiently exploits memory and core organization of GPU even in cases when a programmer poorly organize the kernel grid and memory allocation. Kernel execution requires large amounts of data and access to it is very time-expensive - a programmer should organize data so that neighbouring threads in wrap use equally organized data in memory (consecutive threads should use consecutive memory locations).

Open Computing Language (OpenCL) [28, 29, 40] is cross-platform programming environment that provide standardized support for computers with multiple

processors, GPUs and other computing units. It provides methods to efficiently assign tasks and exploit all the resources of heterogeneous computing platforms. An execution model of OpenCL programs are slightly more complex, but very similar to CUDA.

2.2.3.4 Mathematical and visualization software

Parts of a global analysis with low computing load are performed with software packages like Wolfram Mathematica [41] and MathWorks Matlab ® [42]. It is noteworthy to mention that Matlab offers possibility to massively parallelize certain functions with GPU. The low-dimensional cross-sections (2D/3D) can be visualized in, for example, ParaView [43] or similar programs for data analysis and presentation with parallel capabilities.

Chapter 3

GLOBAL ANALYSIS

Complex behaviour of nonlinear systems can occur even when small nonlinearities are introduced or exist in the system. Analyses of those motions can be very difficult since the mathematical tools that provide closed form solutions for nonlinear systems either do not exist, are valid only for specific families of systems or expressed with special functions [44, 45]. In higher dimensions, the analytical solutions can be obtained in even less cases, directly supporting the development of various numerical methods to compute the solutions.

The basins of attraction (required for dynamical integrity calculation) also have to be detected numerically in the majority of cases. A process to discover them consists of solving the systems of first-order differential equations for a large number of initial conditions. Due to a dimensionality curse, we were forced to search for the fastest computational way that is compatible with the computing platforms that were at our disposal. For that purpose, we have analysed basin computation methods already developed in the past. Due to numerous obstructions associated with HPC platforms, we discuss the principal method we used for computations, how to overcome some of its innate drawbacks and the parallelization of resource demanding parts according to the methodology described in the previous Chapter 2.

3.1 Analysis of dynamical systems

Majority of natural systems are in fact nonlinear [6, 7], but an initial clue of overall dynamics can be sometimes obtained by analyzing the corresponding linear system. With analytical methods available for linear systems, it is fairly easy (from computational point of view) to determine stable and unstable behaviour of the solution. However, the similarities between linear and nonlinear system depend on the magnitude of nonlinearities, where higher nonlinearity produce more diverse collection of behaviours, such as quasi-periodicity, deterministic chaos, solitons, fractals, riddled basins, pattern formation, etc. In order to determine which of those diverse behaviors are present in the system, a nonlinear analysis combines an analytical approximation, numerical calculations and experimental data. Another, important task is observation of system behavior during the change of some system parameters, since in many cases it can lead to the change in topology of the system (qualitative change).

Moreover, nonlinear systems may have arbitrary number of steady motions, some of which are stable and some of which are unstable. If trajectories converge towards a certain steady state, it is called attractor, while it is called a repellor if trajectories diverge away from it. A basin of an attractor consists of the all initial conditions that converge to the associated attractor forward in time. **The goal of a global analysis is to get a global behaviour of system, expressed in terms of attractors and their respective basins.** However, this is not a self-sufficient process -

it is accompanied with analysis of time series, frequency responses and parameter variation (bifurcations) [11].

Nonlinear systems can be represented mathematically through systems of partial or ordinary differential equations. For global analysis, the most interesting are dynamical systems which can be reduced to a system of differential equations of the first order [7]. The most important family are systems of second order differential equations, ensuing from the Newton's second law of motion, which can easily be reduced to the first order systems. The dimension of the resulting system (not to be confused with mechanical degrees of freedom) is equal to the number of first order differential equations. Each dimension in this case corresponds either to the coordinate or velocity that appears in the system. This representation gives possibility to use well developed numerical techniques [46] to integrate the equations. Solutions then can be analysed as trajectories in multi-dimensional state space.

Although the behaviour may be complex, numerical methods are able to compute fairly accurate results [9]. Difficulty comes with the increase in system dimension, as a number of required computations increases exponentially. Therefore, to numerically analyse dynamical systems with large dimension, it is necessary to resort on powerful computational computer systems, which heavily really on mass parallelization of computation. Currently, a multidimensional global analysis is focused on building basins in more than four dimensions. Six-dimensional systems are examined contemporary while eight-dimensional ones present a challenge for both computation and visualization.

3.1.1 Poincaré sections and maps

The analysis of a n-dimensional continuous system can be simplified by reducing it to a $(n-1)$-dimensional discrete one[8, 47]. By taking an $(n-1)$-dimensional surface (Poincaré section), traverse to the flow, the transformation rule, obtained by following trajectories from the current intersection to the subsequent one, is called a Poincaré map. In the associated discrete system, the properties of orbits are preserved and the dynamics of the original system can be analysed in lower-dimensional state-space.

Apart from some specific cases (for example the so called stroboscopic Poincaré map for periodic systems, see Fig. 3.1b), there is no strict procedure to choose the section and create a Poincaré map. In a general case, the one-sided Poincaré section (all intersection signs are equal) is favored wiht a respect to a two-sided one. A Poincaré section can be taken at different locations, consequently, different Poincaré maps are constructed for the considered system. However, a differentiable transformation from one Poincaré map to another usually exists, and the maps on the different sections exhibit the same qualitative dynamics (number and stability of steady states), because these are properties of the systems and do not depend on how we analyze them [8, 47].

Figure 3.1a shows how a Poincaré sections can be placed for autonomous systems with a periodic orbit. Maps constructed in this way are also referred to as return maps. For a non-autonomous system where the period of an excitation term is known, the stroboscopic map is obtained by taking the values of state variables at subsequent periods T, like in Figure 3.1b.

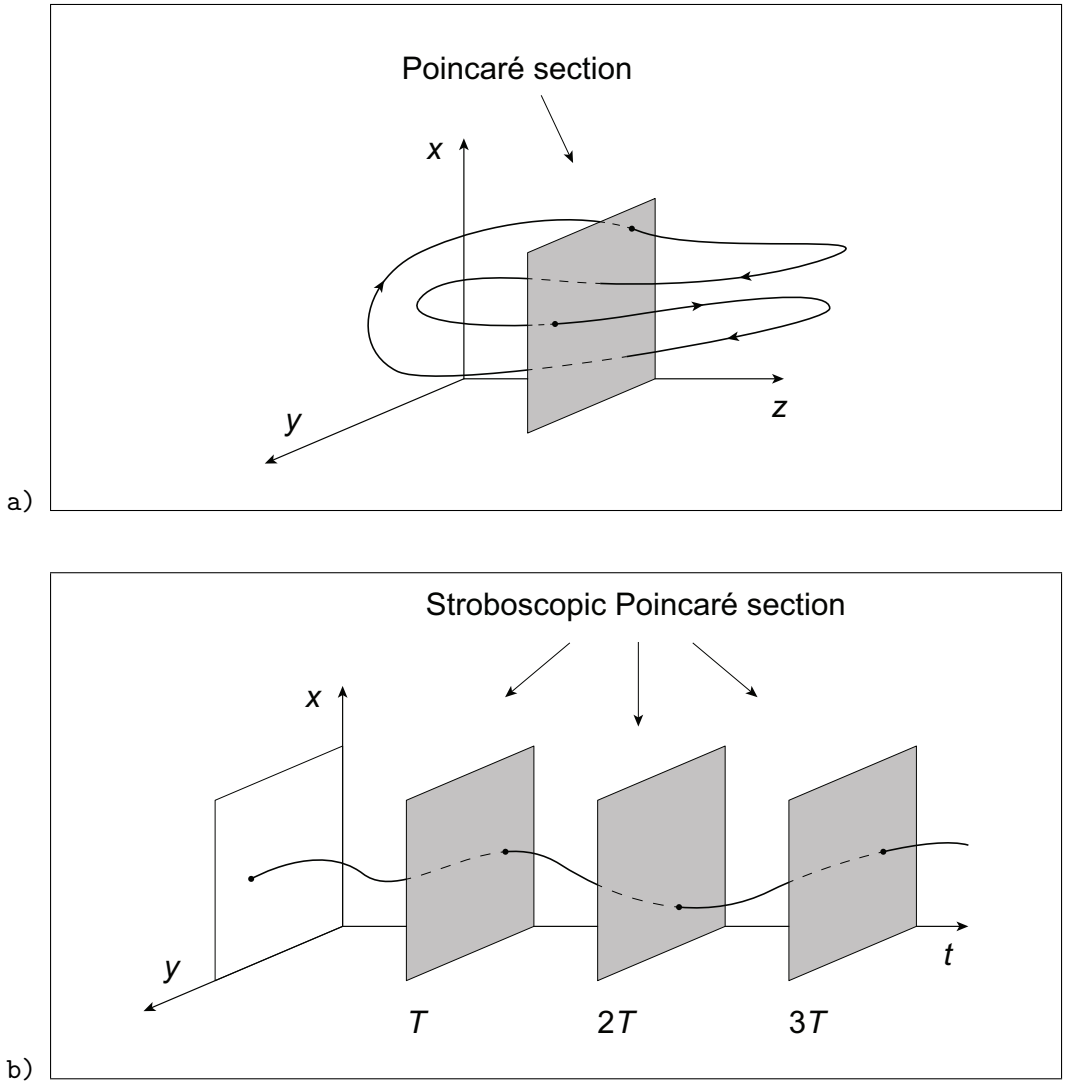

FIGURE 3.1: Poincaré sections for a) return map and b) stroboscopic map.

3.1.2 Classification of attractors and basin boundaries

Properties of fixed points are very well illustrated in literature and well know to engineers, therefore they are not discussed herein. In nonlinear systems, especially the driven ones, other type of steady state solutions also appear quite often - periodic, quasi-periodic and strange/chaotic. This happens in the systems we have examined and thus, they will be briefly presented (for rigorous definitions, the reader is referred to [6, 8, 47]). For the multi-stable systems, it is also important to examine the nature of boundaries that separate basins of attraction.

3.1.2.1 Periodic attractor (limit cycle)

The closed orbit of a flow $\mathbf{y}(t) = \mathbf{f}(\mathbf{x}_{initial}, t)$ that passes through the same point $\mathbf{x}_0$ in state-space on each successive time period T, so that $f(\mathbf{x}_0, t) = f(\mathbf{x}_0, t+T)$ and $f(\mathbf{x}_0, t) \neq f(\mathbf{x}_0, t+T)$ for $0 < t < T$, is a periodic orbit of period T. When there are no nearby periodic solutions, the particular periodic orbit is isolated, and if trajectories approach it forward in time, it is an attracting limit cycle (periodic attractor). Those correspond to a fixed point of a Poincaré map. Periodic solutions can also return to the same point after n multiple of driving term T_F for periodically excited systems, namely $T = nT_F$. Then the corresponding Poincaré map returns to the same point after n iterations, and the limit cycle is referred to as n-periodic limit cycle. In Fig. 3.2a a 3-period limit cycle is plotted for illustrative purposes. The (arbitrary) location of the Poincaré section is shown in Fig. 3.2b, and the resulting map (the intersection of the attractor trajectory with the Poincaré section) is plotted in Fig. 3.2c.

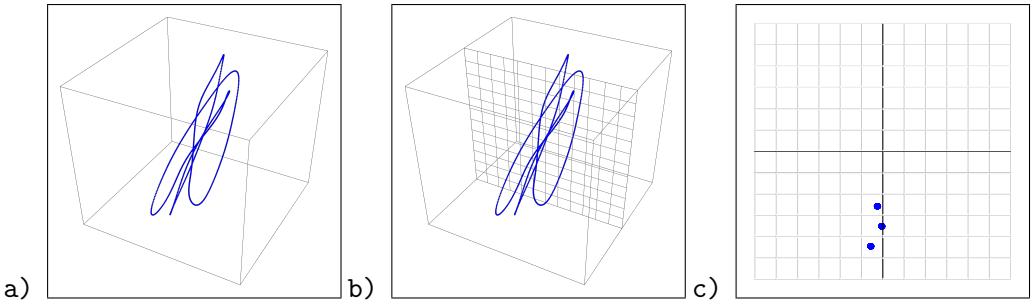

FIGURE 3.2: Periodic attractor (of period three): a) trajectory, b) location of Poincaré section and c) attractor intersection with Poincaré section.

3.1.2.2 Quasi-periodic attractors

Orbits that are characterized by at least two incommensurate frequencies (ω_i/ω_j is irrational) and are periodic separately for each one frequency are named quasi-periodic [6, 47]. The trajectory is often described by $\mathbf{y}(t) = \mathbf{f}(\mathbf{x}_0, \omega_1 t, ..., \omega_n t)$, where $m_1\omega_1 + ... + m_n\omega_n = 0$ only when all $m_i = 0$, and m_i are integers. Then, the trajectories lie on the surface of a hyper-torus and never repeat to themselves due to aperiodicity. The example of quasi-periodic trajectory in Fig. 3.3a intersects the Poincaré section located in the state-space as in Fig. 3.3b, providing the closed curve reported in Fig. 3.3c). The initial points on this curve do not return to themselves.

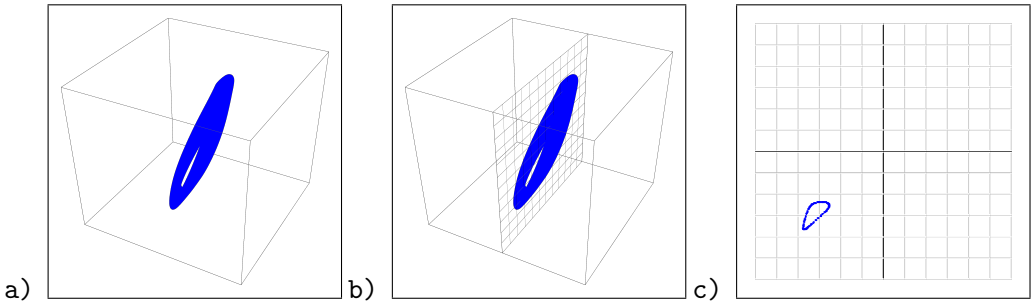

FIGURE 3.3: Quasi-periodic attractor: a) trajectory torus, b) location of Poincaré section and c) torus intersection with Poincaré section.

3.1.2.3 Strange and chaotic attractors

The attractors with a fractal (non-integer) dimension in state-space are called strange [6, 47]. They are mostly associated with chaotic behaviour, which is not always the case. The chaotic behavior is actually a property of a collection of trajectories, characterised with divergence of nearby trajectories and so-called sensitivity to initial conditions. Without rigorous definitions, the requirements for chaos are the following conditions:

- exponential divergence of nearby trajectories;
- trajectories never intersect;
- trajectories remain bounded within some part of state-space;
- recurrence (sooner or later the trajectories will get close - but never coincide with the initial conditions)
- density of periodic orbits (periodic orbits approach arbitrarily close to every point in the state-space)

An exemplary trajectory of a strange (also chaotic) attractor is presented in Figure 3.4a. The Poincaré section (here stroboscopic) of the chaotic attractor (Figure 3.4c) shows a structured cloud of points that is confined in state-space and points seam to randomly appear within it.

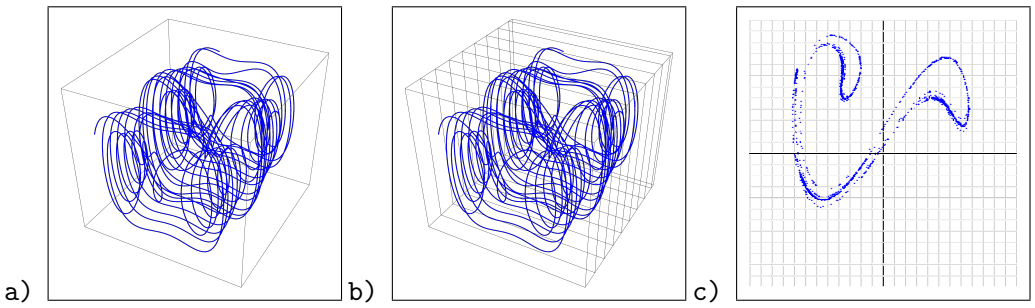

FIGURE 3.4: Chaotic attractor: a) trajectory, b) location of stroboscopic Poincaré section and c) trajectory intersection with Poincaré section.

3.1.2.4 Fractal structures

In nonlinear dynamics, another phenomena occurs, associated with non-integer dimension of its objects - the fractality [13]. Fractal curves are not smooth, but geometrically irregular or an uneven shape of non-integer dimension, repeated over all magnifications - from large to infinitesimally small. An example of a fractal structure is the*Koch snowflake* [48]. It is generated starting from a triangle, where on each iteration the middle third of each straight edge is replaced with two segments of equal length, and repeated infinitely. Figure 3.5 shows the first several iterations in the construction and how the segment are arranged.

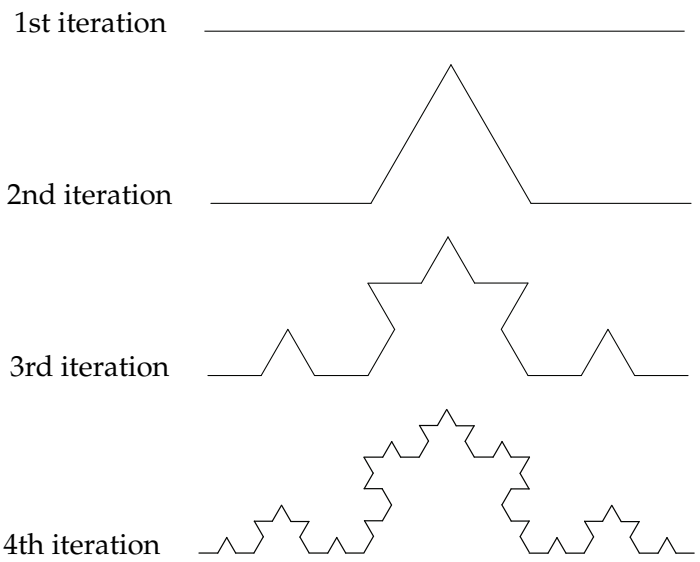

FIGURE 3.5: Fractal curve - *Koch snowflake*.

3.1.2.5 Basin separation

In cases where multiple attractors coexist, basins can be separated by smooth or fractal curves, or hyper-surfaces, depending the system dimension. Another possibility is that basins may be riddled [13, 49]. It means that the border between basins is not a curve (neither smooth nor fractal) and points in infinitesimally small hypersphere around certain initial condition do not necessarily converge to the same attractor. Numerically, fractal boundaries and riddled basins can only be assumed up to the computer precision. Figure 3.6 shows examples of numerical approximation of smooth and fractal boundaries (on the left) and riddled basins (on the right).

3.2 Basins of attraction computation

Due to finite precision of digital computers, numerical procedures use discrete representation of continuous spaces. Consequently, numerical methods used to compute basins can be divided into two major categories by the discretization technique. The first type of methods use points as discretization entities, which are often not suitable as the regions between points remain undefined. To avoid possible issues with undefined regions, the majority of methods (second category) divide the state-space

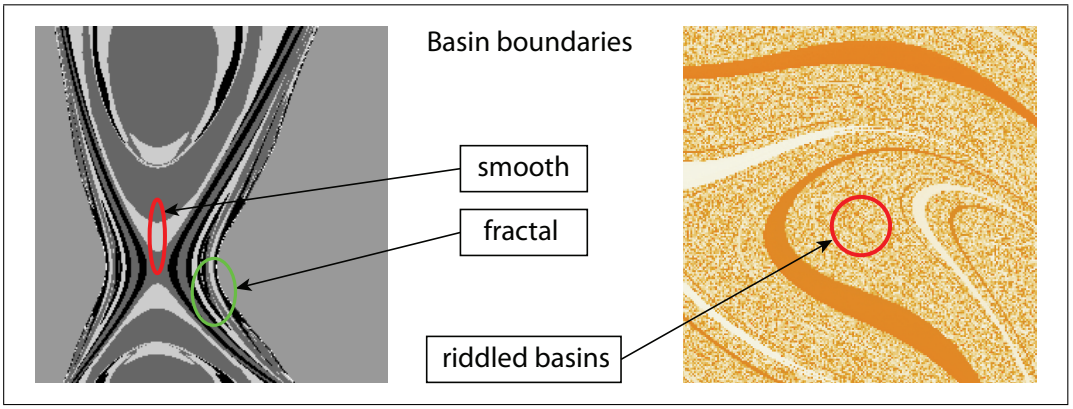

FIGURE 3.6: Basins with smooth and fractal boundaries (left), and riddled basins (rigth).

into the collection of small hyper-volumes, commonly called *cells* [11]. With the cell representation, whole hyper-volume inside a cell is approximated with only one point, usually its center (center-point method). Contemporarily, it is also required to choose the finite portion of the infinite state-space, the *window* or *cell-space*, where to build the basins. Improper choice of a window with respect to cell size may lead to incorrect results due to poor precision or to unrealistic computational time if the proposed precision is too high. In general, the **computation of basins highly depend on the balance between resource consumption and accuracy of results**.

It is noteworthy to mention the other contemporary approaches aimed in similar direction, namely the basin entropy [50, 51] and the basin stability [52–54]. Even so are modern and effective methods they are not within the scope of this thesis.

3.2.1 Grid of Starts

The method named Grid of Starts or Grid of Initial Conditions [55] is based on the very definition of basins of attraction. Upon the discretization process, each point or cell that exists in the state-space window is directly integrated. Each initial condition is separately (independently) integrated until its trajectory gets *close* to the attractor. The closeness is a property that can be defined arbitrarily, in a manner that satisfies the condition - the trajectory that is close to the attracting one will eventually converge to it. In this manner, the accuracy of basins depends only on the integration error.

Considering the definition of the method, integration time will be directly influenced with the length of transients. This is the main drawback of this approach - the process is very resource intensive (CPU time) with high-dimensional, although it is strongly parallelizable.

3.2.2 Cell Mappings

Despite the accuracy and reliability of the GS method, it is a very time-inefficient method in higher resolutions. The *Cell Mappings* [11, 12] are group of methods specifically developed to reduce the computation time of basins. A basic idea is to approximate the continuous trajectories with a Poincaré map for periodic and autonomous systems. For CM methods, the map is obtained by integration over short-time period

and recording the cells where trajectories intersect the Poincaré section. The collection of those *image* cells constitute a map, and any continuous trajectory (within a state-space window) can be followed by its iteration. Consequently, by the analysis of the map, the dynamics of the associated continuous system can be determined.

Simple Cell Mapping. It is the original concept on which many modifications are devised. In SCM, the (Poincaré) maps obtained as described in Section 3.1 are analysed to find a periodic loops, which correspond to discrete approximation of steady-state solutions of continuous systems. All other cells are then classified as transient, which correspond to basins of attraction. This method significantly speeds-up the computations for basins, on the expense of accuracy - the SCM is unable to disclose fractalities or chaos and basin boundaries are relatively crude. For non-autonomous systems, especially those periodically excited, the SCM often gives incorrect results due to the different periodicity of attractors and stroboscopic Poincaré section, low basin robustness and large transient amplitudes.

Generalized Cell Mapping. Contrary to the definition of maps, where each cell has single image cell, the GCM algorithm accepts multiple images. Transition from one to another cell is governed by a probability function which leads to Markov chains for a deterministic and stochastic systems. Abundant literature on Markov chains provides means necessary to examine GCM and discover invariant sets, manifolds of saddle-like equilibrium states, basins of attraction and their boundaries.

Interpolated Cell Mapping. The ICM method [56] uses the coarse SCM solutions to refine the invariant sets. Interpolation is performed without integrating the differential equation when the SCM solution is on a sufficiently small grid so that the underlying dynamics of the system is smooth enough for interpolation.

Improved and modified Cell Mapping methods. Due to high versatility of CM methods, many modification and improvements are devised, from which only few are listed. In [57], the ICM is improved to refine also the basin boundaries. An answer to the dimensionality problem is presented in [58], and some further examples of improved and modified cell mapping methods can be found in [59–61].

3.2.3 Discussion on parallelized basin computing methods

For higher-dimensional systems (more than 4D) it is inevitable to resort on HPC, considering the number of integrations required to do to discover basins of attraction. Consequently, the leading efforts to speed-up the basin computation explore possibilities to massively parallelize the integrations. This is a *data intensive* task [29], so the most (but not the only) attempts to parallelize computations are done for shared memory computers and GPUs, which are designed to handle such problems. Herein, we discuss parallelized methods, their advantages and drawbacks, in order to determine which one is the **best candidate** for computation of **full-dimensional** basins for **6D dynamical systems** on **cluster** HPC platforms.

One of those basin computation method aimed for HPC is described in [62], where the algorithm for *Parallel Multi-Degree of Freedom Cell Mapping* simultaneously examine multiple trajectory sequences. This method does not output the full basin of attraction - only the portrait of the selected 2D cross-section, hence it is not a good candidate for our purpose.

Clustered Simple Cell Mapping method [63] computes separate 2D solutions and joins them in the post-processing to form a global picture of dynamics. Solutions are joined by examining the border cells. By increasing the dimension number of cells that need examination drastically increases, since the touching region becomes

a hyper-surface. Consequently, the low scalability for higher-dimensional systems prevents us to implement this method.

One of the successful attempts to truly parallelize computations on massive, HPC scale is done in series of articles [10, 64–66], where the authors performed extensive analysis and provided an algorithm how to use GS efficiently on cluster computers, and its application for dynamical integrity analysis for 4D systems. However, the course of our research was directed by CL-U platform described in Section 2.1.3, which does not have enough computing power to handle the GS computations for 6D systems.

Currently, the most advanced and developed HPC method for a global analysis is *Hybrid Cell Mapping* [12], developed to work efficiently on GPUs. The mix of SCM and GCM algorithms uncover global dynamics, and then in backward manner, the subdivision and interpolation techniques refines images of attractors. As we are neither concerned in accurate representation of attractors, nor have access to GPU, the HCM algorithms, *in our case*, have no particular advantages over original SCM.

Therefore, the SCM is the best candidate since:

- it discovers full-dimensional basins of attraction for 6D systems (suitable to compute dynamical integrity measures);
- no additional resources are spent on precision of attractors and fractal phenomena;
- it computes the results within a reasonable time-frame even on small clusters, thanks to highly parallelizable map creation.

3.2.4 Simple Cell Mapping

The full process of SCM method was developed by C. S. Hsu and extensively analysed in [11]. Herein, we summarise it, since it is a foundation of our work. Readers familiar with SCM may skip this part and proceed to the next Section 3.2.5 where we discuss parallelization and ways to overcome some of the innate drawbacks.

3.2.4.1 Definition of cell-space

The SCM uses the cell discretization scheme, described in Section 3.2. Each regular cell is represented with a positive integer called *cell number*. A region outside the chosen window, namely the union of all outer attractors, including the infinity, is represented with the cell number 0 - a predefined cell called *sink*. Cell numbers can be transformed into cell coordinates and further to the real coordinates. The correlation between coordinates and cell numbers can be seen in Figure 3.7 for a generic example where $(-2, 2)$ intervals of continuous state-space are discretized with six cells per dimension. The choice of cell coordinates is not unique and may be chosen as wanted (e.g. $\{-2, -1, 0, 1, 2, 3\}$ or $\{1, 2, 3, 4, 5, 6\}$ instead of $\{0, 1, 2, 3, 4, 5\}$). As well, the initial corner and direction of cell indexing is arbitrarily chosen. It changes only the coordinate transformation functions, however, the system dynamics remains the same.

The *precision* (number of cells per dimension) is also an arbitrary choice. However, in 6D increase of accuracy (i.e. decreasing the size of a cell) leads to exponential increase in number of cells, which drastically add up to the computing requirements.

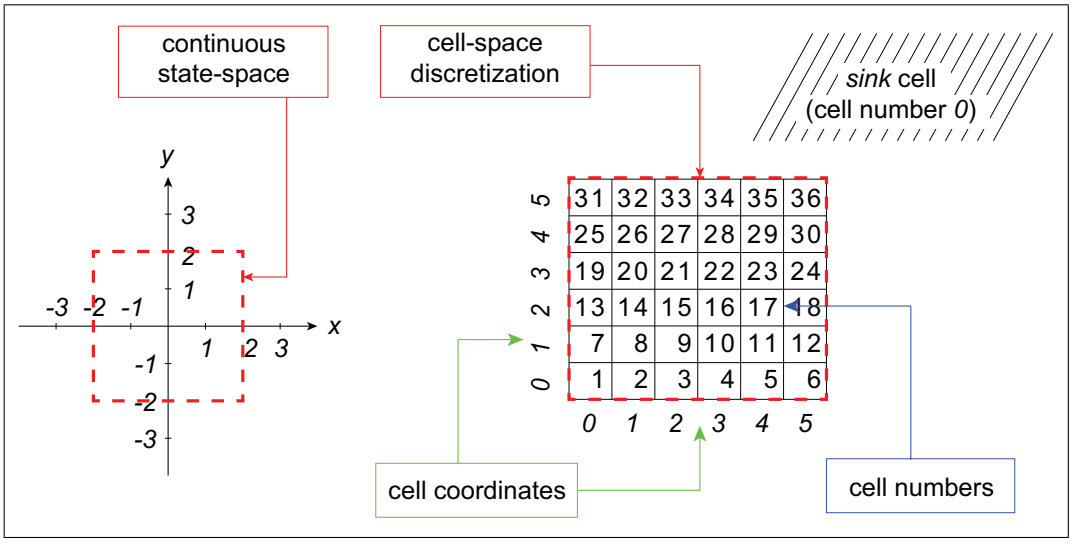

FIGURE 3.7: Cell discretization scheme.

3.2.4.2 Poincaré map creation

To create the map, all initial conditions (cell center-points) are integrated and the cells where trajectories intersect the Poincaré section are recorded. Those cells are named *image cells*, and the mapping is constituted when each cell have the corresponding image computed. On the left in Figure 3.8, is the initial cell-space with the (stroboscopic) Poincaré section where image cells are captured, and on the right is the resulting discrete mapping.

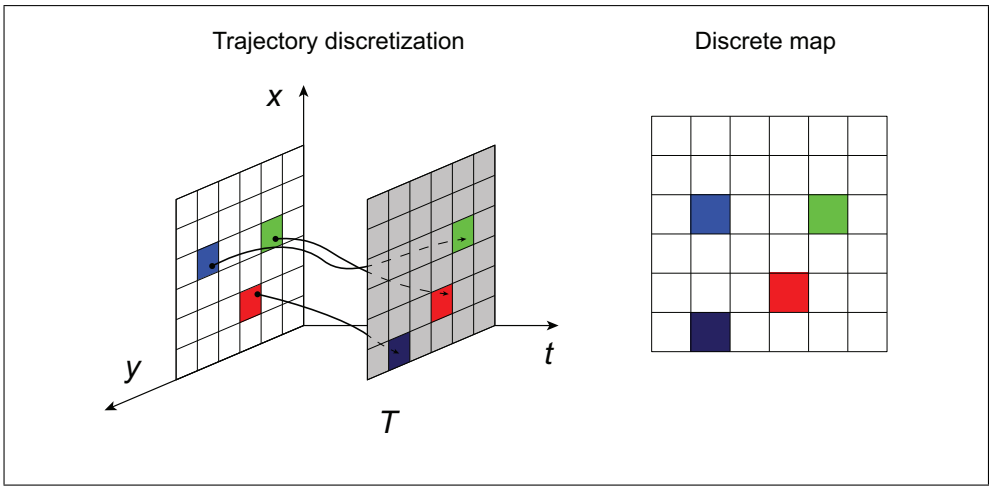

FIGURE 3.8: Process of creating the discrete map by capturing image cells on stroboscopic Poincaré section.

The result of the integration stage of SCM is a mapping function, constituted as a multi-dimensional square matrix, where the number of the image cell is reported in each entry. A 2D generic example for the mapping function is shown in Figure 3.9.

To store the map in computer memory it is more suitable to transform the image cell matrix into an one-dimensional array. Indices of the array then correspond to cell numbers and array elements to image cells. For example, matrix in Figure 3.9, can

14	11	11	0
10	11	11	11
6	11	11	11
6	6	7	8

FIGURE 3.9: Generic example of 2D mapping function.

be transformed into the array $\{6,6,7,8,6,11,11,11,10,11,11,11,14,11,11,0\}$, when the initial corner is lower left and direction of coordinate is from the left to the right.

3.2.4.3 Map analysis

By iterating the map it is now possible to follow the trajectory of each cell and analyse the global dynamics of the system. For example, iteration of the map from Fig. 3.8 correspond to the real trajectory shown in Fig 3.10.

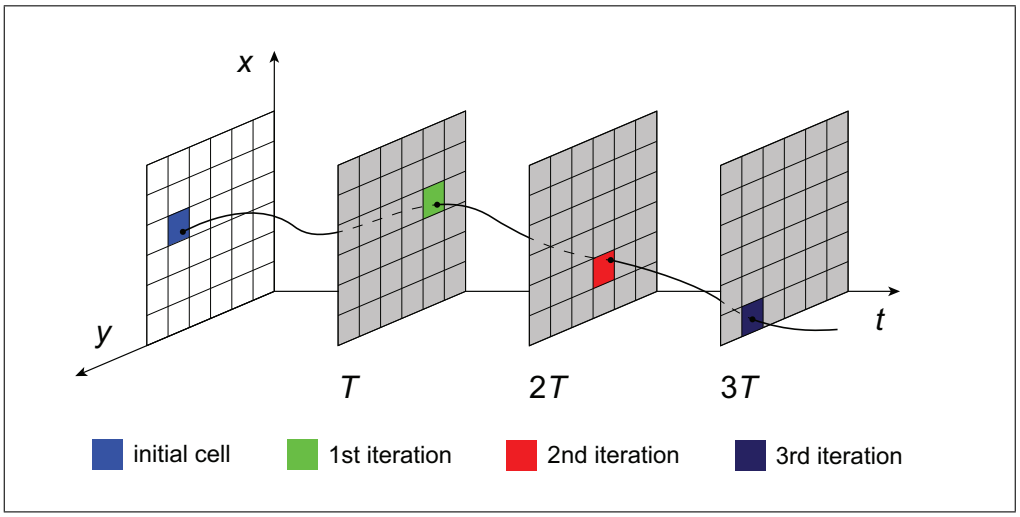

FIGURE 3.10: An example for approximation of the real trajectory with a discrete map.

The map analysis algorithm requires additional information to determine and house data about global dynamics. In the original SCM, those information are implemented as separate arrays or an array of structured data, by assigning the following attributes to each cell:

- *image cell number* - a positive integer number representing the next cell in the sequence;
- *periodicity number* - a number of iterations required for the associated attractor to make a full loop;
- *step number* - a number of iterations required to reach the attractor (transient behaviour);
- *group number* - a number representing periodic group association (basin of attraction).

TABLE 3.1: Unravelling algorithm - map iteration.

```
FOR(z = 1 ... NC){
    stp = 0
    IF(z.group == unprocessed){
        z.group = elaborating
        b = z
        WHILE(b.group == unprocessed){
            stp++
            b=b.image
        }
    }
    IF(b.group == elaborating) NEW_GROUP()
    ELSE OLD_GROUP()
}
```

The algorithm which determines the global behaviour of a Poincaré map is called *unravelling* or *Forward Sort* [11]. The analysis starts from the first unprocessed cell. The map is then iterated until a new periodic loop (periodic *group*) is discovered or an old one encountered. Then, the algorithm appropriately labels all cells in the current sequence. The analysis continues from the next unprocessed cell, terminating when all the cells have been elaborated. Pseudo-codes for unravelling algorithm is divided in three parts, in Table 3.1 for unravelling, Table 3.2 for an old group subroutine and in Table 3.3 for a new group subroutine. The algorithm uses the following variables and constants:

- `z` - currently examined cell
- `b`, `e` and `d` - auxiliary cell variables
- `NC` - total number of cells in the cell-space
- `stp` and `i` - auxiliary counters for number of steps
- `gmax` - number of currently discovered periodic groups

Cell attributes in the pseudo-code are accessed by naming the property after a cell number (e.g. `cell.attribute`). It is important to consider that the sink cell is always predefined: it maps to itself, its cell number is 0 and group number is 1.

At the end of the unravelling algorithm each cell has all its attributes determined. Cells with the step number equal to zero, are cells from the invariant sets, and all other are transient, constituting the basins of attraction. It is worth pointing out that **for the map, since it comes from a space discretization, all the attractors (periodic, quasi-periodic and chaotic) are approximated by periodic orbits of the map**. Of course, increasing the number of cells discretizion, will increase the period of the approximating orbits for quasi-periodic and chaotic attractors, but the orbits will remain periodic.

3.2.5 Not So Simple Cell Mapping for clusters

The nature of SCM, namely the discretization with short-time integration, results in some inconsistencies of results, especially with driven non-autonomous dynamical

TABLE 3.2: Unravelling algorithm - old group sub-routine.

```
OLD_GROUP(){
    z.group = b.group
    z.period = b.period
    z.step = stp + b.step
    e = z
    FOR(j = 1 ... stp - 1){
        e = e.image
        e.group = b.group
        e.period = b.period
        e.step = step - j + b.step
    }
}
```

systems. The impact of long and/or high-amplitude transients and aperiodic invariant sets, together with crude precision, can sometimes lead to inaccurate or even an incorrect global picture of the examined systems. Initially unaware of those drawbacks, we remained determined to compute dynamical integrity even on a small, low-cost cluster. Thus, we dedicated the first part of our research to overcome those drawbacks retaining computational efficiency of the original algorithm and parallelize very resource demanding map creation.

The resulting approach, named *Not So Simple Cell Mapping*, is a collection of modifications we devised to the original SCM method. The improvements are aimed to **speed-up the computation via parallelization** and **to extend the range of dynamical systems that can be examined**. In this section, thus, we discuss in detail: advantages of parallelization on cluster computers; memory restrictions; adaptability of integration time for map creation; transient problems; and discretization issues.

3.2.5.1 Integration parallelization

The goal of integrations is to compute the images of all cells contained in the selected state-space window. So the system has to be solved for a quite large number of initial conditions. Herein, we discuss parallelization on cluster computers according to the methodology presented in Section 2.2.1; for GPU algorithms the reader is referred to [12].

It is evident that parallel opportunities are found by domain decomposition, since the cell-space can be partitioned into smaller parts, and the tasks consist of associated cells for integration. Those integrations are mutually independent (SIMD computations), so it is possible to achieve very high levels of parallelization, meaning that scalability is very high.

Also, the consequence of mutually independent integration is that there is no communication between parallel tasks, excluding the need for design of communication structures and agglomeration.

Actual efficiency of parallelization strongly depends on the proper mapping of parallel tasks to the hardware - the number and distribution of tasks should be implemented according to the configuration of cluster nodes. For the clusters with one core per node, like CL-U, parallelization is implemented by assigning one MPI rank

TABLE 3.3: Unravelling algorithm - new group sub-routine.

```
NEW_GROUP(){
    i = 0
    d = z
    IF(d == b){
        gmax++
        z.group = gmax
        z.period = stp
        z.step = 0
        e=z
        FOR(j = 1 ... stp - 1){
            e = e.image
            e.group = gmax
            e.period = stp
            e.step = 0
        }
    }
    ELSE{
        WHILE(b \neq d){
            i++
            d = d.image
        }
        gmax++
        z.group = gmax
        z.period = stp - i
        z.step = i
        e=z
        FOR(j = 1 ... i - 1){
            e = e.image
            e.group = gmax
            e.period = stp - i
            e.step = i - j
        }
        FOR(j = i ... stp - 1){
            e = e.image
            e.group = gmax
            e.period = stp - i
            e.step = 0
        }
    }
}
```

to each node. In other words, the number of parallel tasks (MPI ranks) is equal to number of nodes.

Further parallelization can be achieved on clusters with multi-core nodes (e.g. CL-C), where the OpenMP multi-threading is initiated within each MPI rank. Then, the number of parallel tasks is equal to the total amount of cores - number of nodes multiplied with number of cores per node.

Some processors have the Hyper-Threading® technology [67], where the single core can manage multiple threads, which can increase the level of parallelization even more. Theoretically, the speed-up should be equal to the number of parallel tasks, but it is somewhat less due to the technical restrictions of the multi-core systems [28, 29].

3.2.5.2 HPC view on map analysis

The cell mapping array is a set of strongly connected components [12] and algorithms that disclose attractors work on inherently serial principle of the Depth-first Search [68], which cannot be parallelized easily. A possibility to parallelize map analysis is presented in [69]. The algorithm checks if every cell belongs to some invariant set or not. For each cell, the map is iterated until it returns to the initial cell, meaning that the cell is periodic, or the number of iterations exceeds the predefined maximal value, meaning that the cell is transient (or in some cases that belongs to a chaotic invariant set). However, this algorithm finds only periodic cells and their periodicity. Additional post-processing is required to sort those cells to the corresponding periodic groups, to discover if invariant sets are attracting or not, and the most important, to sort transient cells to the corresponding basins of attraction. The post-processing anyway **must be** executed sequentially for result consistency, to prevent multiple enumeration of an unique invariant set by multiple parallel tasks.

Moreover, the parallelization is achieved by allowing multiple tasks to access each cell. On shared memory systems, as GPU, this does not present large obstruction, as data is easily accessible to each parallel task (all data is stored in local memory). On distributed memory systems, all data which is stored in local memory of other nodes (remote data) have to be communicated through designated channels. For map analysis with large number of cells, the accessing to remote data creates overwhelming communication overheads. Contrary to the parallelization goal, with the increase of parallel tasks, the efficiency of the algorithm on clusters with this many communications, drops with the increase in parallelization level. A consequence of large redundancy is that a parallel map analysis is then drastically slower in comparison to the implementation of the sequential approach with the unravelling algorithm.

Up to our knowledge, an efficient parallel algorithm for map analysis does not yet exists, thus this part of global analysis is left to be sequential. Fortunately, it is considerably faster process than the map creation and it does not significantly prolong the global analysis procedure.

3.2.5.3 RAM requirements

As it is established what data is required to analyse the map, let us now discuss the memory requirements for it. First, we stress that we are not considering saving numbers on mass storage devices because it will require a lot of accesses that are very slow and unacceptably increase the computational time. Thus, to be efficient in time we can save only on RAM. Next, we observe that storing a map made of p

TABLE 3.4: RAM requirements (in GB) as a function of precision p and dimension d for 8-bytes data type and, within round brackets, for 4-bytes data type (where it is possible).

dimension	p=30	p=40	p=50	p=100
4	0.0064 (0.0032)	0.02 (0.01)	0.05 (0.025)	0.8 (0.4)
5	0.194 (0.097)	0.82 (0.41)	2.5 (1.25)	80
6	5.82 (2.91)	32.78 (16.39)	125	8′000
7	175	1′311	6′250	800′000
8	5′249	52′429	312′500	80′000′000

cells (precision), in dimension d requires saving (in RAM) p^d numbers (one per cell), each number in the range from 1 to p^d (the image cell number). This means that, theoretically, we need $p^d \times x \times 10^{-9}$ Giga-bytes (GB) if we use a x-bytes data type.

Limiting our considerations to an 8-bytes data type (which is typical for clusters), this entails having the minimum RAM requirements illustrated in Tab. 3.4. For example, if we have 16 GB of RAM in our cluster, and we need at least 40 cells precision, we cannot go above dimension six. To date, having 8 dimensions (still with $p = 40$) is impossible because 52.4 Tera-bytes (TB) of RAM are not available on common cluster.

We observe that for low precisions and/or low dimensions, it is sufficient to use 4-bytes data type. More precisely, this is true if $p^d < 2^{32}$. In this case, the RAM requirements are reported in the brackets in Tab. 3.4.

In addition to storing the map, we need to store the basins of attraction, too. Actually, the original SCM method beside basins, also stores periodicity and transient behaviour of each cell. Beside the image cell number, these require to save three integer numbers more for each cell containing, respectively, the label of the basin, the periodicity and the transient. For the first, a 1-byte number is sufficient (assuming that we have less than 256 attractors, which is commonly the case); for the second, a 2-bytes number is required (assuming that the transient will last less than 65′536 map iterations); for the third, at least 2-bytes are required (assuming that we can approximate a chaotic attractor with a 65′536 period attractor). This requires storing $p^d \times 5 \times 10^{-9}$ GB, which is comparable to the memory requirements for storing the map.

When only the basins are required, we can skip saving periodicity and transients, thus reducing the memory requirements to $p^d \times 1 \times 10^{-9}$ GB, which is just 1/8 of the values reported in Tab. 3.4 and it is a more affordable.

3.2.5.4 Prolonged integration time

For periodically excited systems with strong nonlinearities, the mapping function with image cells captured in early transient phase, is often unable to recreate the steady state dynamics accurately enough. It is a consequence of large transient amplitudes that escape a predefined state-space window, and/or low basin robustness due to fractalities or its narrow size in certain dimension(s).

When trajectories escape the predefined phase-space window during transient behaviour, it can happen that image cells are being captured in those external regions. Origin cells (and their external images) are then attributed to the basin of the

sink, regardless if their real corresponding trajectories converge (after longer time interval) to an attractor inside the window or not. Consequently, in this cases the mapping function does not have the valid information to emulate system dynamics.

For some systems, and for certain choices of investigated phase-space regions, this happens for the majority of initial conditions, and it is necessary to overcome this problem. An intuitive solution could be to extend the considered state-space window until every relevant initial condition has an image cell inside it. In the case that we examined, this was not practical solution - the RAM size prevented us from increasing the number of considered cells, needed to capture images of enough **relevant** cells. Consequently, it was not possible to properly reconstruct the system dynamics.

The feasible solution was to hybridise SCM with GoS, and prolong the integration time to allow the escaping trajectories to reenter the predefined region of state-space. The integration interval is extended to an integer multiple of system periods, which in the most cases is quicker than it is required for GoS to approach the attractor. The integration is extended up to the re-entry to the considered phase-space window or when the solution remains outside the window for a sufficiently large time - meaning that one has indeed approached an external attractor.

For certain dynamical systems (or specific parameter values), especially in cases when fractal basins boundaries occur (but not exclusively), compact parts of less robust basins can be very narrow in some phase-space dimensions. Consequently, the mapping created in an early transient phase often misses those narrow or fractal parts of a basin and assigns cells to a nearby (the more robust) basin. Possible remedy is to prolong the integration time to capture the image cells closer to the steady state, where trajectories pass close enough not to miss the narrow or fractal parts of basin.

Evidently, the robustness of the SCM method for strongly nonlinear systems particulary those periodically excited, depends collectively on the integration time and the cell size. Prolonged integration time can remedy some problems, but has to be maintained the shortest possible, or otherwise it would be more suitable to use the GoS method. Regarding the cell size, it has to be kept sufficiently small to represent dynamics of the system accurately enough and simultaneously not too small to overload memory with large cell-space. Those factors have to be considered carefully and brought into the balance; otherwise the benefits of fast computation with SCM could be lost.

3.2.5.5 Disconnected periodic orbits

As we use original SCM to analyse the mapping function, the ability of capturing invariant sets (also unstable ones) is inherited from the properties of discrete maps (thoroughly discussed in [11]). To compute those mapping functions, the SCM implements the center-point method [11], where the whole interior of a cell is represented by the point at its center. The image cells are thereby computed by the integrations originating from those points. Consequently, an error occurs, as the end point of the computed trajectory hardly coincides with the image cell center (that is the initial state for computation of another image cell). The errors can be minimized by reducing the cell size, which directly increases memory requirements, and therefore is impractical to achieve since we are yet exploiting near all the available hardware capabilities.

As the error can be only minimized, but not eliminated, it propagates and grows over the map iterations, to the magnitude that can outgrow the size of cell. When

it happens, the real (obtained with direct integration) and approximated (obtained with SCM) trajectory do not overlap any more. This difference culminates with the SCM reporting separate periodic orbits, which together represent a single attractor. We name them *disconnected periodic orbit*. The second source of the disconnected periodic orbits is the integration time, namely the position of a stroboscopic Poincaré section. Aperiodicity of quasi-periodic and chaotic attractors results in out of phase intersections with the Poincaré section. Consequently, all trajectories associated with those motions are also out of phase and the unravelling algorithm sorts them as separate, distinct periodic attractors. The example in Fig. 3.11 illustrates a quasi-periodic attractor reported with several separate SCM periodic orbits, each one colored differently.

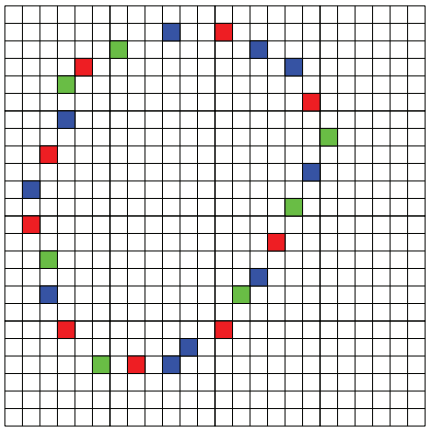

FIGURE 3.11: Disconnected periodic orbits: generic example of separate periodic SCM groups that should represent single quasi-periodic attractor.

It is worth emphasizing that while for periodic orbits this problem can be theoretically resolved by reducing the cell dimensions (i.e. increasing the accuracy), this is not the case with quasi-periodic and chaotic attractors. For higher resolutions, the number of cells that approximate trajectories of those orbits increases, consequently increasing also the number of disconnected periodic orbits.

Since with the clusters considered herein increasing the cell is not a feasible option (as we are yet exploiting almost the maximum accuracy allowed by available hardware), and since in any case for quasi-periodic and chaotic attractors this strategy does not solve the problem, we need another way to overcome it or, in other words, to connect disconnected periodic orbits. Practically, we need to a posteriori check all periodic orbits (as discovered by SCM) and to verify if two (or more) of them are actually the same real periodic orbits (as obtained by *direct* independent numerical integration). Once two (or more) periodic attractors of SCM are found to be the same, they are merged (*connected*) and their basins (previously discovered by SCM) merged as well.

To address the previous problem, we introduce the Connecting Post-processing Algorithm (CPA), which is a procedure that sequentially examines and joins the disconnected periodic orbits discovered by raw SCM, according to the pseudo-code in Table 3.5. The search starts on line `CPA_1`, from the periodic group with identifier 2, as the group number 1 is reserved for the sink, which does not require an analysis, up to the last periodic group enumerated with `g_max`. In the line `CPA_2` the initial conditions are prepared for the integration - the variable `current_x` takes the real

TABLE 3.5: Connecting Post-processing Algorithm.

```
CPA_1    FOR(current_g = 2 ... g_max)
CPA_2       current_x = p_cells[current_g]
CPA_3       FOR(1 ... connect)
CPA_4           current_x = integrate(current_x)
CPA_5           current_z = convert(current_x)
CPA_6           IF(current_z.grp > current_g)
                    correct_grp[current_z.grp]=current_g
CPA_7           IF(current_z.grp < current_g)
                    correct_grp[current_g]=current_z.grp
```

coordinates of arbitrary periodic cell center, from the periodic orbit scheduled for examination (value of `current_g`. The loop on line `CPA_3` then counts the periods of the real trajectory integration, previously specified by the user, controlled with the value of `connect` constant. On each loop iteration, one step of integration is executed in line `CPA_4`. If the integration error is smaller than the size of a cell, the group numbers of the initial condition `p_cells[current_g]` and the computed cell (`current_z` obtained on the line `CPA_5`) are (should be) equal, which is - the real and SCM approximated trajectories traverse same cells. The lines `CPA_6` and `CPA_7` test those conditions and assign correct groups otherwise. We have settled that when multiple groups are designated to merge, the group with lower identifier is the *correct* one.

Once the corrections have been determined, the array with calibrated groups is broadcasted to all ranks. From this point, each rank parallelly updates basin information for cells stored in own local memory.

As the CPA, after initial testing, has been proven to be tremendously faster than integration and the analysis, we decided to leave it not optimized for the sake of simplicity. We are well aware that it does some redundant work, as neither loop is interrupted in case when all correct groups are determined before the last iteration. Also, the CPA does not examine repelling and saddle orbits, which are represented as SCM periodic groups, but are unstable and thus do not have basins of attraction.

The CPA is the last modification to the original SCM that we devised. As those improvements are not changing the overall logic of SCM, they are all-together referred to as Not So Simple Cell Mapping. The differences between the SCM and NSSCM at each part of a computation process are summarized in Table 3.6. The nonlinear systems considered in it are those with the relatively short transient time and without a high transient amplitudes, namely, the systems for which we can find an adequate Poincaré section and state-space window for the computations.

3.2.5.6 NSSCM performance vs. accuracy discussion

Now, after considering all the aspects of HPC and basin computation method, it is natural to give some concluding comments on the NSSCM and basin computations. Foremost, it is the method aimed at **full high-dimensional basins of attraction**, which favors the accuracy sacrifice in order to get the results within a realistic time-frame. However, the **balance** between accuracy and computational load is the most important factor when the 6D basins are involved in the analysis. Moreover, for the nonlinear systems with long transients, it can happen that neither one contemporary method cannot deliver results within acceptable time-frame.

TABLE 3.6: Stages and properties of SCM and NSSCM computing process for dynamical systems with relatively short transient time and without high transient amplitudes. Results with asterisk sign (*) are available on demand.

Computing stage	SCM	NSSCM
Image cell computation (periods of integration)	single	adjustable
Map post-processing (result of analysis)	• basin • periodicity • transients	• basin • periodicity* • transients*
Attractor validation	none	CPA
Solvable dynamical systems	• autonomous • non-aut. periodic	• autonomous • non-aut. periodic • non-aut. driven

On clusters and distributed memory computers in general, it is even more difficult to make a proper balance, as the map analysis may require overwhelming communication overheads. A suggestion is to start with rough precision and refine the computations in case that preliminary results do not animate the dynamics properly. In other cases, the solution may be to increase integration time or precision. From computational point of view, sometimes is not possible to determine full 6D basins in reasonable time-frame, even with low precision.

However, even in low precision, the roughly determined basins can be very useful to determine dynamical integrity accurately enough. In the following chapter, we proceed to define integrity measures and discuss how basin computation with NSSCM can be applied to calculate them.

Chapter 4

DYNAMICAL INTEGRITY

In mechanical, civil and building engineering, the safety is traditionally addressed in a simple way - increase the safety factor for critical parts of the structure/machine. In modern applications, from micro/nano to macro scale, this way of safety design is becoming obsolete [70]. Moreover, it is not just enough to determine if some steady motion is stable or not. There are several rising questions: for what range of system parameters particular steady states exist; what is the impact of initial states on the steady-state dynamics; what is the magnitude of disturbances that will lead to change of steady-state behaviour; how actually large are the safe regions around the exploited steady-state; how to control imminent erosion of system safety? Answers to those questions can be interpreted through the robustness of the attractors. The higher the robustness, the safer an attractor is for practical applications.

A possible way to examine attractor robustness is via dynamical integrity. Evolution of integrity measures gives the erosion profiles, which show the variation of attractor robustness during the variation of some system parameter. The key tool to calculate integrity measures are basins of attraction. This chapter is therefore dedicated to summarize the most important dynamical integrity measures and procedures how to compute them. Further on, the reader might be interested in methods to control erosion of system robustness, for what we refer to the [4, 71], as this subject is out of the scope of the thesis.

4.1 Safe basin and erosion profiles

The very definition of basins of attraction encloses all the initial condition that converge to a particular attractor. However, not all of them are safe for practical applications. The regions of state space that encompass initial conditions which are preferred for exploitation are referred as *safe basins*. Nonetheless, a strict definition of the safe basin is not uniquely established. It can be defined according to tolerance to fractalities, by the analysis of transient or steady-state dynamics, etc. In general, it can be roughly defined as the set of initial condition (in phase-space), sharing a given (desirable) property. Converging to a given attractor is the property we consider when dealing with basins of attraction.

An example could be to address the safe basin as a collection of all initial conditions, which forward in time converge to the attractors inside a particular well. A drawback of this definition is that it ignores trajectories that temporarily escape the potential well during transient dynamics. An alternative definition [72] excludes those initial conditions whose trajectories are not confined inside the potential well during transient behaviour. Moreover, if multiple attractors co-exist inside the particular well, the safe basin in this case is comprised of an union of competing basins. Both definitions are based on a claim that the safe basins are property of potential wells, not of the attractors. The advantage of the first definition is that procedures to

determine those safe basins are straightforward, with already well-established methods described in Chapter 3. The second definition requires time consuming checking of the position of trajectories along transient dynamics. Lastly, the updated definition for a *true safe basin* is introduced in [73] to include the phase of excitation for driven systems. Hence, the true safe basin is the intersection of all safe basins when the phase ranges over the period of excitation.

Next, we are interested to explore the quantitative side of basin safety. Associated with the definition of the safe basin, there are several integrity measures established, of which we summarize and compare the three most widely used. Moreover, the integrity measures can be used to study the evolution of safe basins by varying the system parameter(s). The resulting plots of particular integrity measure evolution dependent on systems parameter (e.g. excitation frequency or amplitude) gives an *erosion profile*, which is useful to evaluate how the structural safety changes. This addresses the practical stability, where eroded regions of stable solutions are determined. Erosion of the safe basin in general is an unwanted phenomenon, and integrity analysis can ultimately help to control the penetration of a dangerous attractor basins and delay the erosion.

4.2 Global Integrity Measure

A normalized magnitude of the previously defined safe basin (e.g. the area of 2D basins) is the *Global Integrity Measure* (GIM). It is the most intuitive integrity measure, however, it neither considers the position of the attractor within the basin nor it rules out possible fractalities into account, in this sense not providing a measure of the usable compact part of the safe basin.

4.3 Local Integrity Measure

The *Local Integrity Measure* (LIM) is a normalized distance from the attractor to the nearest basin boundary. It is a local property of the attractor and rules out the fractal parts. A drawback is that for attractors with a complex structure (e.g. chaotic), LIM is hard to compute, since the distance has to be measured from all points and the minimium has to be taken. Also, there might be a larger compact part of the safe basin that is not centered on the attractor.

4.4 Integrity Factor

The *Integrity Factor* (IF) is the radius of the largest hyper-sphere entirely inside the safe basin. From engineering point of view this is the most useful integrity measure, since the fractalities are ruled out and the largest compact part of safe basin is taken into account.

4.4.1 Alternative distance metrics for Integrity Factor

To find the sphere radius that measures IF, it is necessary to compute the distance transform matrix [14], and extract the highest value for each safe basin. For high-dimensional systems, computing the Euclidean distance can be very computationally intensive. Herein, we consider the alternatives to reduce the computational load. Since it is unavoidable to compute the distance matrix, we discuss the different distance metrics.

Starting from the definition of the Minkowski distance between the vectors **x** and **y**, in a n-dimensional normed vector space

$$dist(\mathbf{x},\mathbf{y}) = \left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{\frac{1}{p}},$$

for the $p = 2$, we have the Euclidean distance (which corresponds to a hyper-sphere), namely

$$dist(\mathbf{x},\mathbf{y}) = \sqrt{\sum_{i=1}^{n} |x_i - y_i|^2}.$$

Now we are concerned with reducing the number of mathematical operations to compute the distance. For the Euclidean case, we have the n power of 2 and one square root operations. We can eliminate the root and the power operations by choosing either $p = 1$ or $p = \infty$.

The $p = 1$ case is called *Manhattan*, or alternatively *taxicab*, *city block*, *snake* or *rectilinear* distance. Formulated with

$$dist(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{n} |x_i - y_i|,$$

it is the sum absolute distance between components of two vectors.

The *Chebyshev* or *chessboard* distance is the limiting case when $p = \infty$

$$dist(\mathbf{x},\mathbf{y}) = \lim_{x\to\infty} \left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{\frac{1}{p}} = \max_{i=1}^{n} |x_i - y_i|,$$

where the distance is the maximal absolute distance between vector components.

The difference between the distance transform matrices can be seen on the example in Fig. 4.1. The distances are measured from the origin cell enumerated with 0.

a)

$\sqrt{2}$	1	$\sqrt{2}$
1	0	1
$\sqrt{2}$	1	$\sqrt{2}$

b)

2	1	2
1	0	1
2	1	2

c)

1	1	1
1	0	1
1	1	1

FIGURE 4.1: Example of distance transform matrices for a) Euclidean, b) Manhattan and c) chessboard distance metrics.

To make the correlation between the Euclidean and discrete metrics (Manhattan and chessboard), we adapt the formulation of a *circle* in discrete spaces as a set of points with a fixed distance (radius) from the origin. The comparison of the circles in Euclidean, Manhattan and chessboard metric spaces is presented in Fig. 4.2a-c, respectively, and they are compared to each other in Fig. 4.2d.

The equivalence between the Manhattan and chessboard metrics does not hold in dimensions higher than two. For example, a discrete sphere (generalization of the circle in 3D) with the chessboard distance metric is a cube, but with the Manhattan distance is an octahedron.

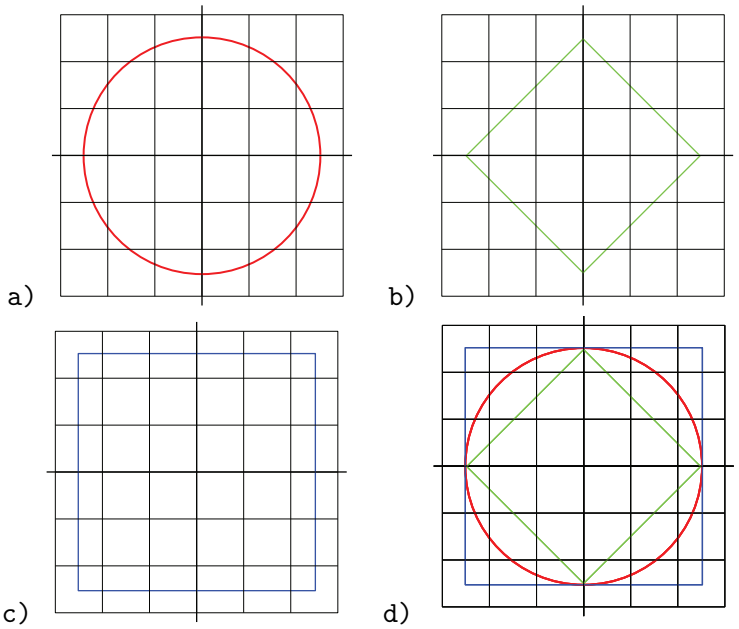

FIGURE 4.2: Circles in a) Euclidean, b) Manhattan, c) chessboard metric space, and d) comparison.

Considering the number of nested loops required to calculate the distance transform matrix in high-dimensional spaces, it is preferable to use the chessboard distance metric. Therefore, instead of a hyper-sphere for computation of IF, we use the **hyper-cube** to reduce computational load.

4.5 Low-dimensional example of integrity measures

Fig. 4.3 shows an example that demonstrates the differences between integrity measures. The relevant attractor is marked with the red point and the area covered with its basin is colored in black. The area covered with other colors represent the basins of unwanted attractors. If the entire black basin is considered safe, the integrity is quantified with GIM, which is measured as the area of a whole basin within the window. LIM is the radius of the largest circle that originates at the attractor and touches the nearest basin boundary, as plotted in Fig. 4.3b. Safe basins for IF with the Euclidean metric are shown in Fig. 4.3c and with the chessboard in 4.3d.

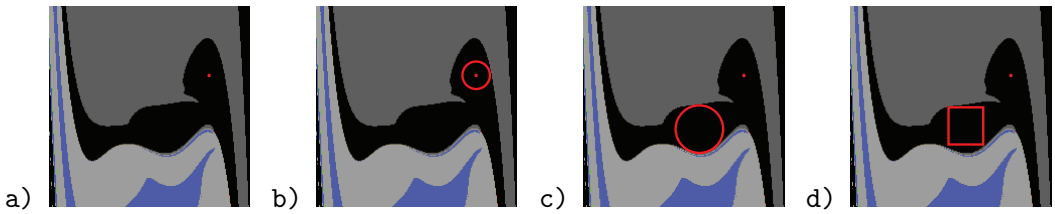

FIGURE 4.3: Integrity measures for green point attractor a) whole basin - GIM, b) LIM, c) IF (spherical) and d) IF (cube).

Chapter 5

NSSCM discussion: validation and performance

Development and application of the NSSCM approach is the central part of our work and this thesis. As it is not an entirely new method, but it represents an extension of SCM, it is useful that we discuss its performance, advantages over the original method and the settings under which NSSCM is a valid approach to compute full-dimensional basins for 6D nonlinear systems.

5.1 Test dynamical system

To validate the NSSCM approach and evaluate the resulting performance, we examine a paradigmatic dynamical system without a related concern about a precise engineering background of the parameters used in modeling. The test system is adopted from [18], and is composed of three coupled Duffing oscillators, one of which one is harmonically excited. It is well-known to exhibit rich nonlinear behaviour, allowing us to check if NSSCM is able to determine it. For more information on Duffing-like nonlinear systems, please see, for example [74–78].

5.1.1 Equations of motion

The three degree-of-freedom nonlinear system composed of three coupled Duffing oscillators (FDO) [18], one of which being forced by a periodic excitation, is mathematically modelled by the following system of six first-order differential equations:

$$\begin{aligned}
\dot{y_0} &= y_1, && (5.1)\\
\dot{y_1} &= F\sin(\omega t) - y_0 + \alpha y_2 - \epsilon(1 - \beta {y_0}^2 + {y_0}^4)y_1, && (5.2)\\
\dot{y_2} &= y_3, && (5.3)\\
\dot{y_3} &= -(1+\alpha)y_2 + \alpha(y_0 + y_4) - \epsilon(1 - \beta {y_2}^2 + {y_2}^4)y_3, && (5.4)\\
\dot{y_4} &= y_5, && (5.5)\\
\dot{y_5} &= -y_4 + \alpha y_2 - \epsilon(1 - \beta {y_4}^2 + {y_4}^4)y_5, && (5.6)
\end{aligned}$$

where y_0, y_2 and y_4 are the respective displacements of oscillators, and y_1, y_3 and y_5 are their respective velocities. The parameters are defined in [18], where $\epsilon > 0$ denotes the degree of nonlinearity, $\alpha \geq 0$ is the coupling factor ($\alpha = 0$ means decoupling, $\alpha = 1$ means maximum coupling) and β is the parameter that determines the magnitude of the term with the cubic nonlinearity. The parameters F and ω denote amplitude and frequency of the excitation.

5.1.2 Attractors

For testing, we consider a multi-stable region that exists for the excitation magnitude $F = 0.75$, frequency $\omega = 1.4$ and the parameter values $\alpha = 0.1$, $\epsilon = 0.5$ and $\beta = 3.1$. Within the state-space window $y_i = (-3, 3)$, five attractors are discovered with direct integration. To illustrate them, we capture the intersecting points of their trajectories and the Poincaré section. Since the FDO system is non-autonomous, the adequate choice for the Poincaré section is the stroboscopic one, with sampling time at integer multiples of the excitation term period. Furthermore, to visualize sampled points, we have to extract low-dimensional 2D or 3D cross-sections. Those are in Fig. 5.1a-d: a) y_0-y_1-y_2; b) y_0-y_2; c) y_0-y_1; and d) y_1-y_2.

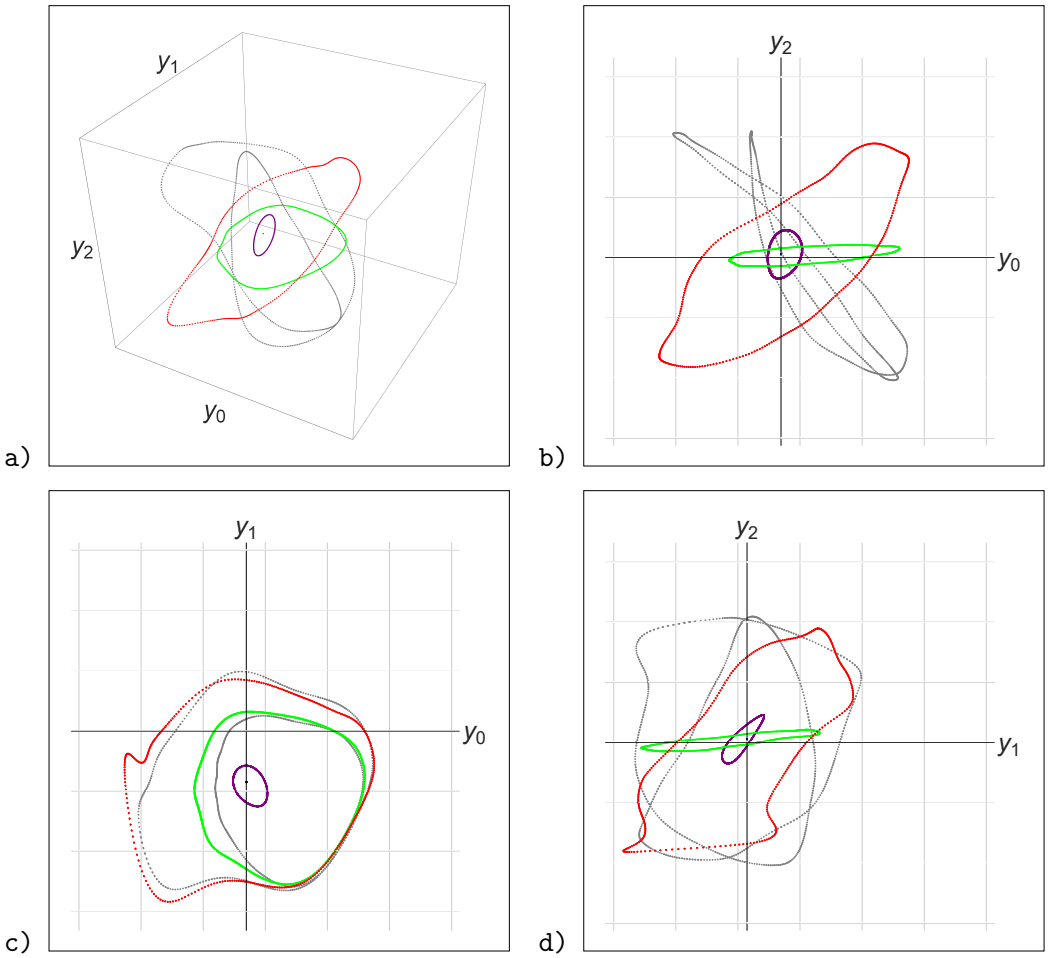

FIGURE 5.1: Stroboscopic Poincaré sections of FDO attractors.

It is evident that under harmonic excitation, the majority of FDO system attractors are quasi-periodic. However, to determine system behaviour more precisely, further conclusions can be made from the information that basins of attraction provide.

5.2 Computation accuracy vs. performance

Before analysing the basin's structure, first we examine the validity and efficiency of the NSSCM approach. To discuss it, we consider the following characteristics and

compare them to direct integration with GoS:

- the number and type of attractors;
- accuracy of basins;
- computational time-frame.

The fundamental factor that must be considered is that SCM (and to certain extent also the NSSCM) may not compute basins accurately enough due to low resolution or inadequate integration time, improper Poincaré section choice or simply the dynamical system is not suitable for CM methods. Moreover, for specific settings, SCM/NSSCM may accurately computes attractors, but not necessarily also their basins.

The GoS discovers only those attractors (and consequently basins) that exist in the considered low-dimensional cross-section(s). Thus, there are likely unaccounted attractors that remain undiscovered with the GoS, unless the full-dimensional basins is computed - which we aspire to avoid with the GoS, due to large computational requirements.

To validate efficiency we impose a standard to which the NSSCM and GoS are compared in terms of computational time: the time-performance is expressed with the number of excitation periods required to discover basins. For the GoS, it stands for the number of periods which a trajectory requires to approach close enough to the corresponding attractor, and for the SCM/NSSCM, the integration interval at which image cells are captured (the position of the stroboscopic Poincaré section). This is a valid hypothesis when the same numerical ODE solver is used for both cases.

In general, the GoS follows each trajectory until the distance between the trajectory and an the attractor becomes what is considered to be small enough, to rate it as convergent. In other words, the GoS integration time depends on the transient interval, which depends on the initial conditions. To decrease computational load, in our case, GoS was restricted to the integration time of maximum 75 excitation periods, meaning that a possibly less accurate criterion for the condition of convergence to an attractor was enforced.

To give a definitive conclusion if the NSSCM approach is valid and efficient, we compare attractors and basins computed with the SCM/NSSCM to the nominal GoS performance of 75-period long integration time and five discovered attractors at $\omega = 1.4$. The resolution of computations with NSSCM is 40 cells per dimension, so the considered full-dimensional window of the phase-space consists of total 40^6 (4'096'000'000) cells. The GoS computations are performed on a 2D cross-sections with grid of 40x40 initial conditions for low, and 150x150 for high resolution. The Table 5.1 summarises how the number of the discovered attractors with the SCM/NSSCM changes depending on the integration time, and how long it takes for the SCM/NSSCM to discover the basins accurately enough to compute IF.

With the plots of the 2D cross-section y_0-y_2 with fixed $y_1 = 0.825$, $y_3 = 0.825$, $y_4 = 1.575$, $y_5 = 0.825$, given in Fig. 5.2 we illustrate the data from Table 5.1. For the integration time of a single excitation period, the basins in Fig. 5.2a are not accurate enough to distinguish compact parts, required to compute IF, although the NSSCM precisely discovered the attractors. For the integration time of three periods the outcome is fairly the same, therefore it is not plotted. Slightly less fragmentation is achieved for the integration time of six periods, however, still with unsatisfactory accuracy for all basins, as it can be seen in Fig. 5.2b.

TABLE 5.1: Accuracy of basin computed with the SCM/NSSCM for test system, depending the integration time.

integration period	SCM attractors	NSSCM attractors	basin accuracy
1	6	5	very low
3	10	4	very low
6	17	5	low
12	47	4	average
15	8	5	good

The further increase of the integration time to 12 periods, gives significantly more accurate basins in terms of their compact parts. However, in this case the SCM/NSSCM nature and low accuracy resulted in that the two attractors were assimilated (red and purple), as shown in Fig. 5.2c. This might have happened for two reasons (which do not exclude each other). The first one is the consequence of low discretization accuracy: in certain dimension(s), trajectories of two attractors may be close to each other. If this distance is smaller than the cell size, the SCM cannot recognize them as separate ones.

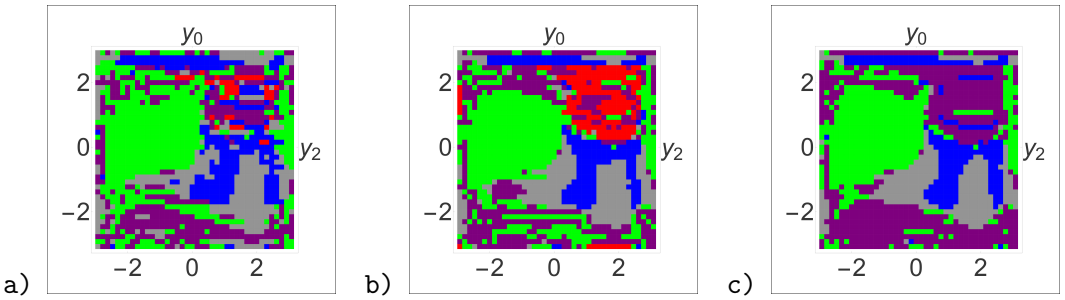

FIGURE 5.2: Impact of integration time on the SCM/NSSCM accuracy on cross-section y_0-y_2 with $y_1 = 0.825$, $y_3 = 0.825$, $y_4 = 1.575$, $y_5 = 0.825$ at $\omega = 1.4$, for the computational time of a) 1, b) 6 and c) 12 periods.

In some cases, quasi-periodic attractors have such SCM-periodicity that for a particular integration time, neither cell from their periodic groups is present on the stroboscopic Poincaré section chosen for SCM. Consequently, the basin of the missing attractor is assimilated within other basin(s).

For the integration time of 15 periods, the basin computed with the NSSCM (Fig. 5.3a) are in good agreement with the GoS in same resolution (Fig. 5.3b), considering the basin parts without fractalities. The high-resolution basins plotted in Fig. 5.3c are determined with drastically higher computational time. However, the structure of the compact parts in all resolutions is practically the same. Therefore, with respect to the integrity factor, having the results in a significantly higher resolution does not bring any particular advantage.

A further note is on the raw SCM and its improved version - NSSCM. One of the NSSCM features is prolonged integration time, which is illustrated with an example in Fig. 5.2. To compute an accurate basin, increasing integration time is an

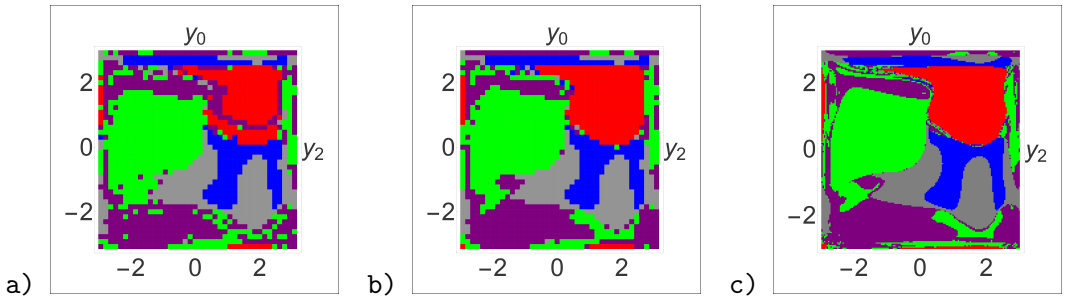

FIGURE 5.3: Basin accuracy on cross-section y_0-y_2 with $y_1 = 0.825$, $y_3 = 0.825$, $y_4 = 1.575$, $y_5 = 0.825$ at $\omega = 1.4$, computed with a) the NSSCM over 15 periods, and over 75 periods with GoS in b) low and c) high resolution.

TABLE 5.2: Execution time-frames of each NSSCM phase on CL-C for integration time of 15 excitation period.

NSSCM phase	**number of nodes**	**threads** (*per node*)	**parallel tasks** (*nodes*×*threads*)	**execution time** (in *seconds*)
integration	32	256	8192	18328
map analysis	1	1	1	360
CPA	1	1	1	224
file I/O	1	1	1	850
IF	1	256	256	3691

unavoidable necessity for driven non-autonomous systems, which is a consequence of transient behaviour.

By looking at the SCM column of Table 5.1, it is evident that without the CPA part of the NSSCM, basins would be indistinguishable and it would be very difficult to understand which SCM attractor/basin pairs correspond to the physical counterpart.

To justify the low-resolution computations with NSSCM, let us consider actual execution times required for each step of NSSCM, summarized in Table 5.2, for a FDO system integrated for the time interval of 15 periods.

By taking into account the 15-period long integration time of 5.1 hours with 8192 parallel tasks, it is clear how much computational resources are saved by using NSSCM instead of GoS. Also, even with significantly more parallel tasks, the integrations are far more demanding than the map analysis and CPA - together 10 minutes sequentially. To conclude, the NSSCM indeed is an efficient approach to determine a full-dimensional basin for 6D nonlinear systems by using cluster computers oriented to a dynamical integrity analysis with IF.

5.3 Structure of full-dimensional basins in 6D

It should be duly noted that a basin is a global characteristic of a system. Thus, information obtained from arbitrary low-dimensional cross-sections cannot be regarded as global; the full-dimensional basins must be considered. To illustrate it,

we examine the basins structure, with an emphasis on compact parts as the pivotal characteristic required to determine a dynamical integrity with IF.

The aforementioned is discussed by confronting: a full-dimensional basin computations with the NSSCM in resolution with 40 cells per dimension (total $40^6 = 4'096'000'000$ cells), where image cells were captured at stroboscopic Poincaré section located at time-step of 15 excitation periods; and GoS for specific 2D cross-sections with 40x40 and 150x150 initial condition grids, with the integration time restricted again to 75 periods.

Let us take first into the consideration the 2D cross-section y_0-y_1 with $y_2 = 2.775$, $y_3 = -0.225$, $y_4 = 1.125$, $y_5 = 0.075$. A plot in Fig. 5.4a is extracted from a full-dimensional NSSCM basin, and the plots computed with GoS in low and high resolution are shown in Fig. 5.4b-c. The cross-section shows two large compact basins, without the intrusion of any other basin or fractalities. Moreover, the border between them is what can be considered as a smooth curve from a numerical point of view.

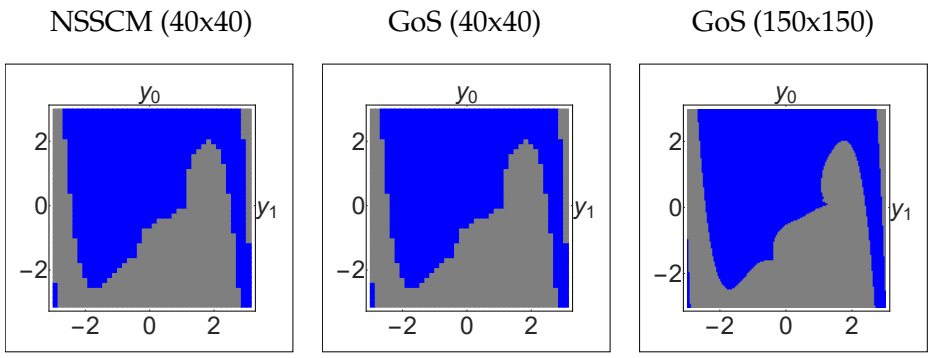

FIGURE 5.4: Basins of test system in 2D y_0-y_1 cross-sections with large compact parts at $y_2 = 2.775$, $y_3 = -0.225$, $y_4 = 1.125$, $y_5 = 0.075$.

The compactness is reduced within the regions where basins are tangled. Examples are shown in Fig. 5.5 (y_0-y_1 cross-section with fixed $y_2 = 0.825$, $y_3 = 0.825$, $y_4 = 0.825$, $y_5 = 0.825$) and Fig. 5.6 (y_0-y_1 cross-section with fixed $y_2 = -0.075$, $y_3 = 1.425$, $y_4 = -0.525$, $y_5 = 1.275$).

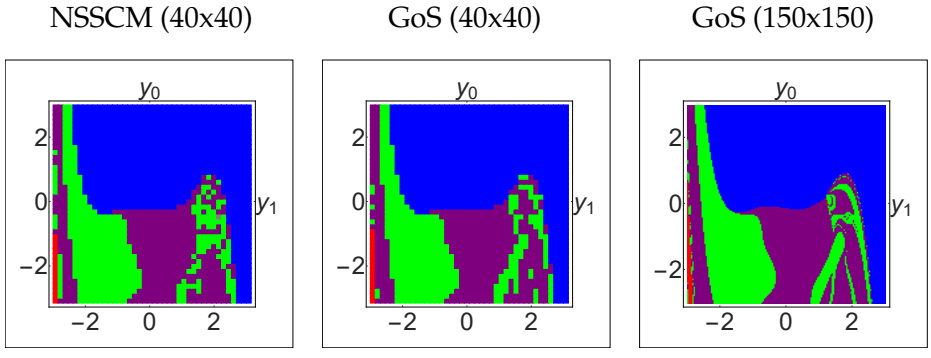

FIGURE 5.5: Cross-sections with both tangled and compact basins for $\omega = 1.4$ at 2D y_0-y_1 cross-sections with large compact parts at $y_2 = 0.825$, $y_3 = 0.825$, $y_4 = 0.825$, $y_5 = 0.825$.

Moreover, at the regions where basins are tangled, riddled and/or have fractal boundaries, cells can contain segments belonging to different basins, which is

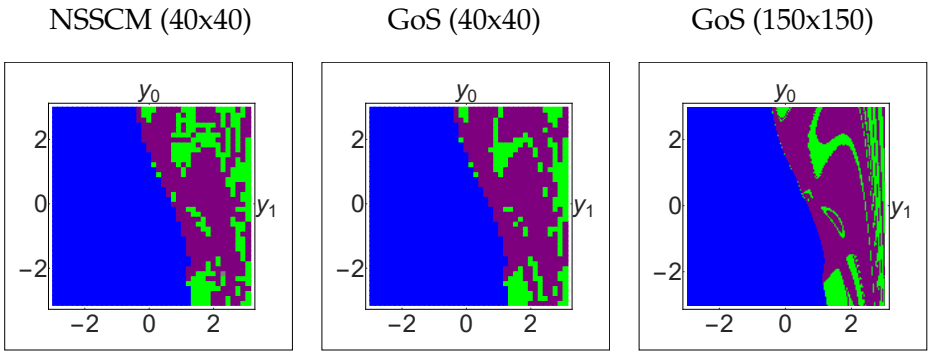

FIGURE 5.6: Cross-sections with both tangled and compact basins for $\omega = 1.4$ at 2D y_0-y_1 cross-sections with large compact parts at $y_2 = -0.075$, $y_3 = 1.425$, $y_4 = -0.525$, $y_5 = 1.275$.

a characteristic of a cell discretization procedure, and is not related to computing approaches. Thus, the accuracy inevitably decreases, as neither approach is able to precisely decide which one is the proper basin, for those cells that encompass fragments of multiple basins. Those circumstances are not necessarily disadvantageous, because the IF calculations does not consider disputable cells for the compact basin due to their vicinity to distinctly separate basin(s).

Such setting occurs in the cell-space region y_0-y_1-y_2, with remaining coordinates set to be $y_3 = -1.425$, $y_4 = -1.425$, $y_5 = -1.425$. For example, it is clear from Fig. 5.7 for $y_2 = -0.225$, Fig. 5.8 for $y_2 = 0.075$ and Fig. 5.9 for $y_2 = 0.375$, which neither basin (on those cross-sections) is noteworthy compact in comparison to those in Fig. 5.4.

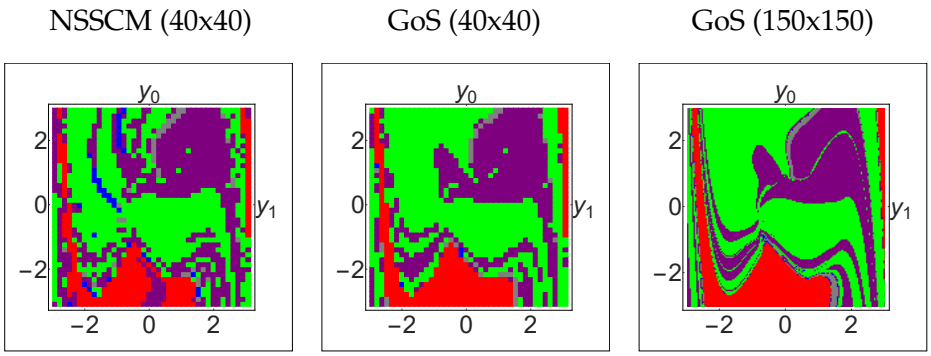

FIGURE 5.7: Basin for $\omega = 1.4$ at region with substantial fractal parts and basin intermittency, y_0-y_1-y_2 with $y_3 = -1.425$, $y_4 = -1.425$, $y_5 = -1.425$ at cross-section with $y_2 = -0.225$.

Complexity of high-dimensional systems also downgrades integrity with the likelihood of abrupt basins structure changes for low coordinate step in some direction. Moreover, this phenomenon often occurs together with tangled basins and fractal boundaries. For an example, lets inspect the basin's structure for the small change in y_2 direction of the y_0-y_1 cross-section, with fixed $y_3 = 1.725$, $y_4 = -1.575$, $y_5 = 0.975$. The basins for $y_2 = -2.625$ are plotted in Fig. 5.10.

How the basin's structure changes when a coordinate is varied by one cell ($\Delta y_2 = 0.15$), is shown in Fig. 5.11 for $y_2 = -2.475$. It can be seen that basins from cross-section in Fig. 5.10 are replaced with basins of other attractors. Those basins also have smaller compact parts.

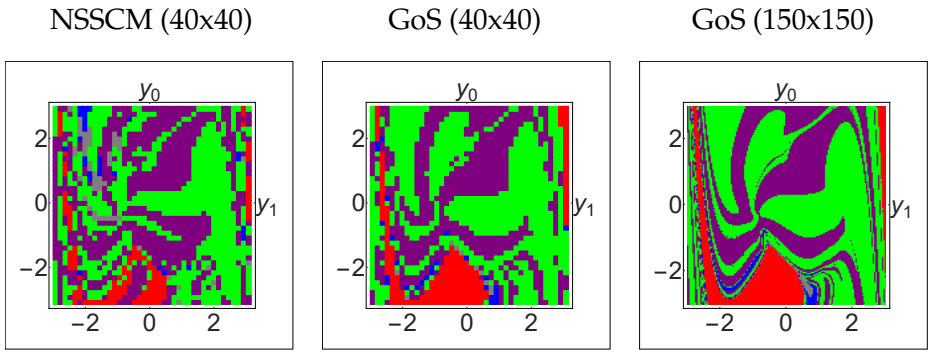

FIGURE 5.8: Basin for $\omega = 1.4$ at region with substantial fractal parts and basin intermittency, y_0-y_1-y_2 with $y_3 = -1.425$, $y_4 = -1.425$, $y_5 = -1.425$ at cross-section with $y_2 = 0.075$.

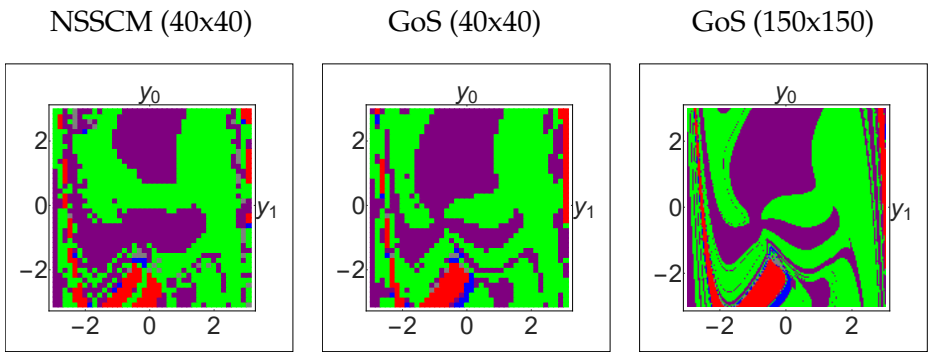

FIGURE 5.9: Basin for $\omega = 1.4$ at region with substantial fractal parts and basin intermittency, y_0-y_1-y_2 with $y_3 = -1.425$, $y_4 = -1.425$, $y_5 = -1.425$ at cross-section with $y_2 = 0.375$.

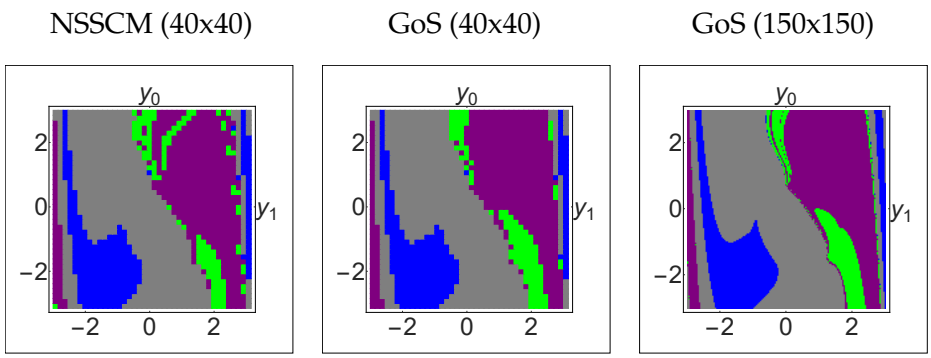

FIGURE 5.10: Basin for $\omega = 1.4$ at y_0-y_1 with $y_3 = 1.725$, $y_4 = -1.575$, $y_5 = 0.975$, where structure of basin changes abruptly, on cross-section $y_2 = -2.625$.

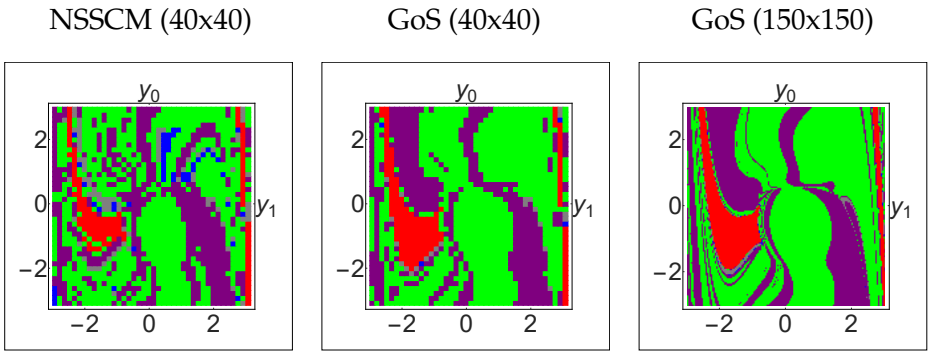

FIGURE 5.11: Basin for $\omega = 1.4$ at y_0-y_1 with $y_3 = 1.725$, $y_4 = -1.575$, $y_5 = 0.975$, where structure of basin changes abruptly, on cross-section $y_2 = -2.475$.

Another change of the y_2 coordinate for a single cell results in a drastic increase of basin robustness. This change is illustrated in Fig. 5.12, where $y_2 = -2.325$.

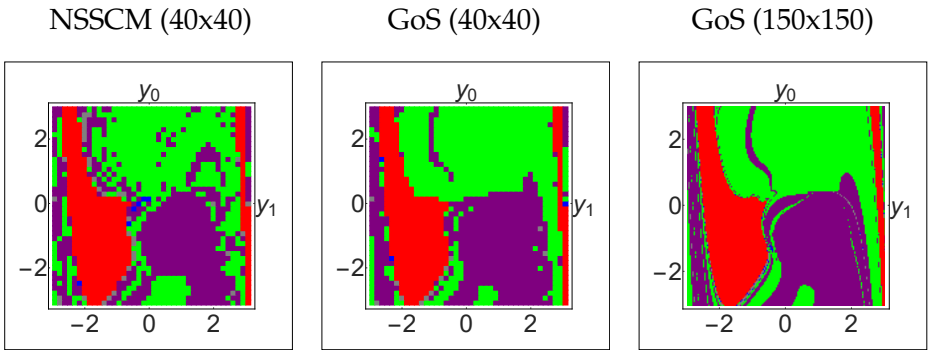

FIGURE 5.12: Basin for $\omega = 1.4$ at y_0-y_1 with $y_3 = 1.725$, $y_4 = -1.575$, $y_5 = 0.975$, where structure of basin changes abruptly, on cross-section $y_2 = -2.325$.

After analysis of the basin's structure, we can give a brief conclusion. Considering that basin boundary and cells close to fractal basin parts are not within a safe basin defined by IF, we are not concerned to precisely distinguish basins in those regions. Those cells are considered as unwanted initial conditions regardless to which basin they belong. Moreover, drastic changes in basin's structure demonstrate necessity to compute a full-dimensional basin for a proper global analysis. Consequently, the low resolution NSSCM method is a valid approach for computing full-dimensional basins oriented to the robustness analysis with IF.

5.4 Dynamical integrity

Up to this point, the discussion was about the computing approach for a discovery of full-dimensional basins of six-dimensional nonlinear systems. As it has been established that NSSCM is indeed a time-efficient and accurate enough method, now we discuss its application to determine dynamical integrity.

TABLE 5.3: Integrity factors (with chess-board distance metric) for FDO basin within state-space $y_i = (-3, 3)$, expressed as hyper-cube edge length (in number of cells).

attr. color	blue	gray	purple	green	red
IF	5	6	6	5	6

To supplement the claims from the previous Section 5.3 and to demonstrate the necessity to compute full-dimensional basins in order to determine dynamical integrity, we examine basin compactness on the 3D cross-section y_0-y_1-$y_z = 2$ with fixed $y_3 = -0.075$, $y_4 = -0.075$, $y_5 = -0.075$.

Let us first consider a 2D cross-section at $y_2 = -2.175$, where the basin colored in gray is compact and dominant over the blue one. A plot in Fig. 5.13a shows this cross-section extracted from the full-dimensional basin computed with the NSSCM. The high-resolution plot computed with GoS is shown in Fig. 5.13b. Its spatial position within the 3D cross-section is plotted in Fig. 5.13c.

Now, we can examine if the gray basin is indeed robust and compact. To do it, we traverse the 3D cross-section in y_0, y_1 and y_2 directions (pointed with black arrows in Fig. 5.13) and visually inspect the 2D slices highlighted with the red square contour.

From Figures 5.13d-f, with the slices at $y_0 = -1.425$, $y_0 = -0.075$ and $y_0 = 1.575$, it is apparent that the basin colored in blue is dominant and with a larger compact part than the gray one. Similar settings are also seen on the slices at $y_1 = -1.425$, $y_1 = -0.075$ and $y_1 = 1.575$, plotted in Figures 5.13g-i and $y_2 = -1.425$, $y_2 = -0.075$ and $y_2 = 1.575$ in Figures 5.13j-l. Moreover, in certain regions the gray basin is not present, like in Fig. 5.13k.

Furthermore, there is neither information about other basins not present in the examined cross-sections nor about the structure in other regions of state-space. Also, it is not possible to visually inspect a whole 6D state-space and it is clear that in general the analysis in low-dimensional cross-sections cannot provide all necessary information. Therefore, the only reliable way to determine the robustness of high-dimensional systems is a full-dimensional analysis.

Actual compactness of the basin for the FDO system is summarised in Table 5.3. It shows the edge length of the largest 6D cube that can be fitted inside each basin present in the state-space $y_i = (-3, 3)$.

However, the IF (or any relevant integrity measure) computed for a single value of a system parameter is not globally useful information by itself. A proper analysis comes in the form of erosion profiles - plots that illustrate how an integrity measure behaves when a system parameter is changed. Unfortunately, to generate those plots for a high-dimensional system, one needs to recompute full-dimensional basins for each parameter value, draining large amounts of computation resources. Although the plotting of erosion profiles would round-up this thesis, it was not possible to compute it with the limited computing resources. Nevertheless, computations demonstrated within this thesis can give us a valuable information which attractors are robust for the application at particular frequencies.

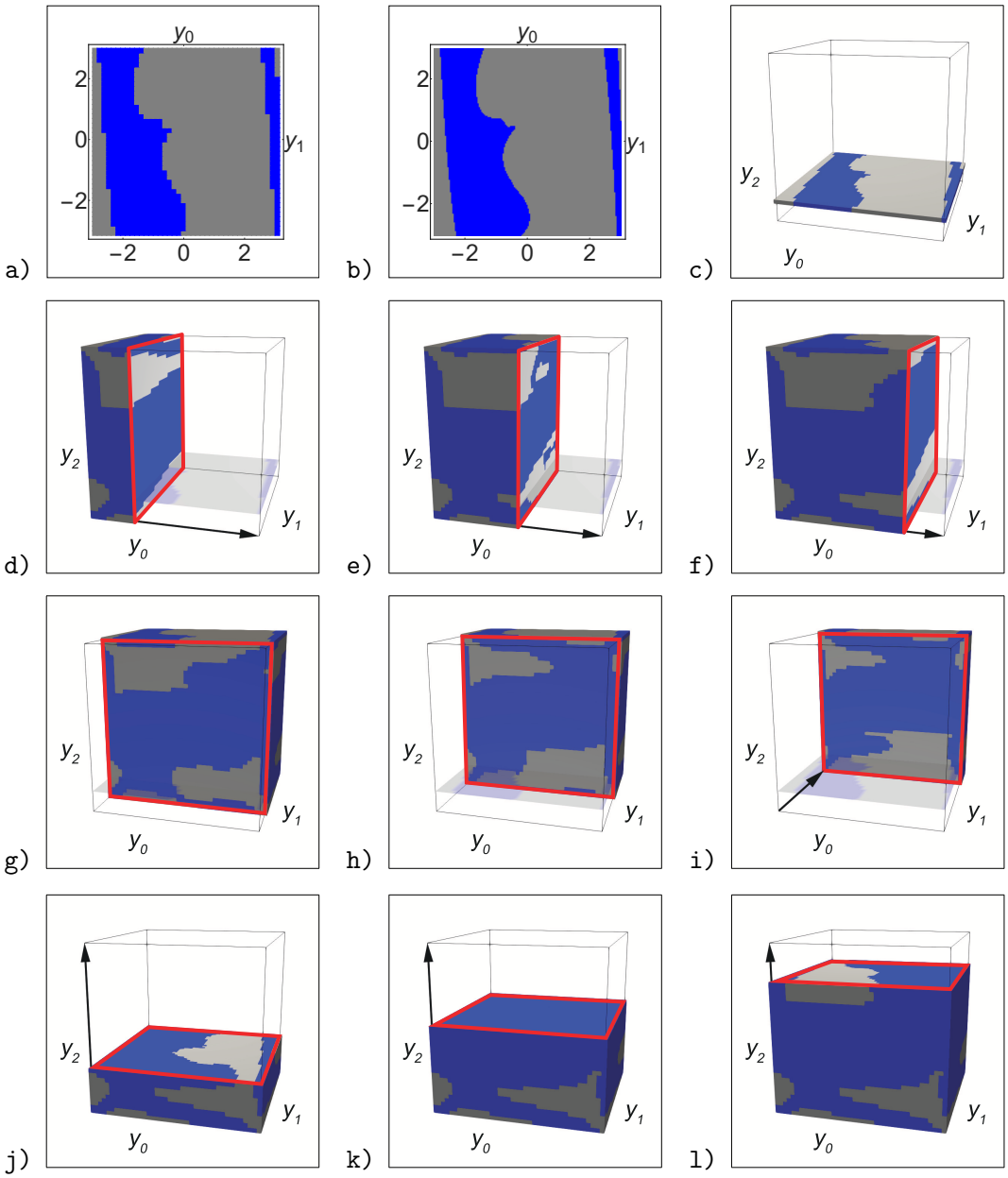

FIGURE 5.13: Compactness change of FDO basins within 3D cross-section y_0,y_1,$y_z = 2$ with fixed $y_3 = -0.075$,$y_4 = -0.075$,$y_5 = -0.075$.

Chapter 6

EXAMPLES

Aimed to determine the global behavior and crown it with computation of dynamical integrity, we examine dynamics of two examples in this chapter, the models of a **sympodial tree with first level branches** (STREE) and a **rotating hub with two attached pendula** (RHUB).

The sympodial tree model with first level branches is examined in collaboration with Prof. Ivana Kovačić[1], whose research group is working on theoretical and experimental research on tree behaviour.

In collaboration with PhD Zofia Szmit[2], we provide basins of attraction for the system examined in [20]. This system was the first step of the research of behaviour of helicopter blades that is an ongoing effort of this research group.

The key intention of this chapter is to illustrate basin complexity for those example dynamical systems. Foremost, brute force bifurcation diagrams are plotted to isolate multi-stable regions, convenient for building the basins of attraction. Those plots show intersecting points of steady-state trajectories with a stroboscopic Poincaré map, for an upsweep and downsweep excitation frequency quasi-statical change. Although the multi-stable regions are isolated this way, it is not certain if all the attractors are discovered. To capture all the attractors, we build a full-dimensional basin.

Then, the basin structure for each example is visually inspected on various low-dimensional cross-sections. However, the global properties of the basin are determined by computing the integrity factor (with a chessboard distance metric), which represents the magnitude of safe basins compactness.

6.1 Sympodial tree model with first-level branches

The interest in studying dynamics of trees comes from their ability to endure variety of detrimental natural conditions. To be appropriately exploited for engineering applications, this resilient behaviour needs to be understood first. So far, two main approaches to model trees are: as a combination of rigid body elements (e.g. [19, 79–81]); and as elastic bodies with finite element method, as in [82, 83]. Some of the experiments that supplement those theoretical studies are presented in [83], while certain engineering applications are summarized in [84].

We analyse the basins of attraction and dynamical integrity of the periodically excited nonlinear model of the sympodial tree with first-level branches [19]. The results of this analysis can be used as a foundation to examine behaviour of slender structures subjected to conditions analogous to those acting on trees in nature. The analysis presented herein is a summary of preliminary work published in [17] and

[1] Faculty of Technical Sciences, University of Novi Sad, Serbia
[2] Lublin University of Technology, Lublin, Poland

is still an ongoing research, which is related to some of the rich nonlinear behaviour of this model.

6.1.1 Equations of motion

The model under examination, presented in Fig. 6.1 is adopted from [19], with the addition of the periodic torque of magnitude M and angular frequency Ω, applied to the element representing the tree trunk. Two identical branches with mass m_1, length l_1 and diameter D_1 are attached to the trunk with the corresponding parameters m, l and D. The trunk can perform rotations around the axis of the hinge that attaches it to the base, and is connected to the ground by a spring with stiffness coefficient k and a viscous damper with damping coefficient b. Viscous dampers and springs also connect branches to the trunk, and their respective coefficients are labelled by b_1 and k_1. The generalized coordinates are defined as the absolute angles, φ for the trunk and ψ_1, ψ_2 for the branches, and are measured against the respective equilibrium positions, as shown in Fig. 6.1b. The equilibrium angle of the branches with respect to the trunk is defined by the *branching angle* α.

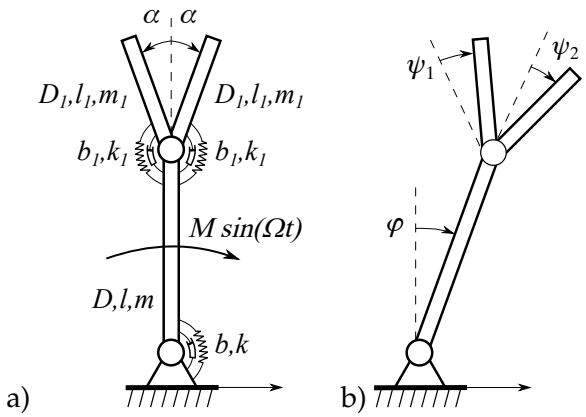

FIGURE 6.1: Model of sympodial tree with first-level branches, a) model properties, b) generalized coordinates.

To properly mimic characteristics of the sympodial tree, two additional parameters are introduced to the model: the *lateral branching ratio* $\lambda = 1/2$ [85], which describes the proportion of the cross-sectional area for successive segments; and the *slenderness coefficient* $s = 3/2$ [85, 86], which relates the length and the diameter of each element. Then, the remaining model parameters are defined as follows: the *diameter ratio* $D_1/D = \lambda^{1/2}$, the *length ratio* $l_1/l = \lambda^{1/2s}$, the *mass ratio* $m_1/m = \lambda^{4/3}$, the *stiffness ratio* $\kappa = k_1/k$, the dimensionless *damping coefficient* $\zeta = b/2l\sqrt{3/km}$ and the *damping ratio* $\beta = b_1/b$. Consequently, the governing system of equations is derived in the following dimensionless form:

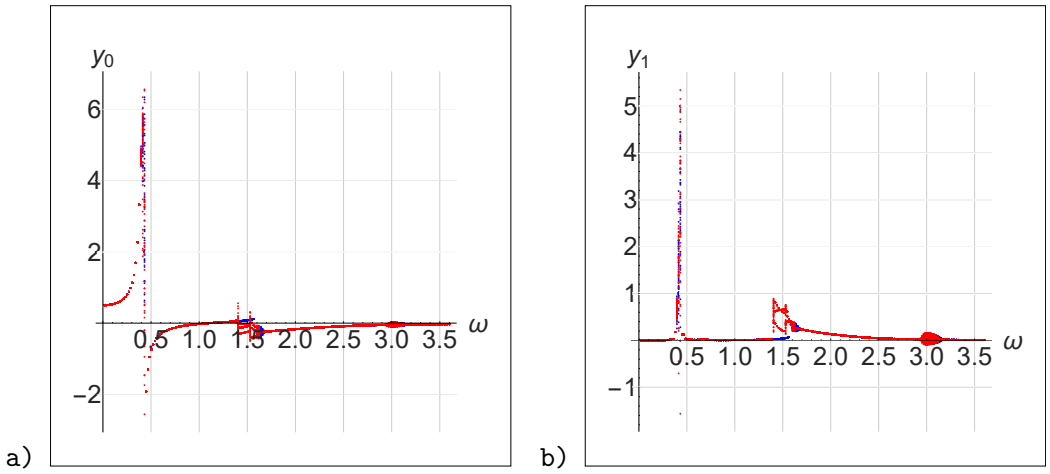

FIGURE 6.2: Trunk segment brute force bifurcation diagram for $\zeta = 0.02$ in excitation frequencies interval $\omega = (0, 3.6)$

$$
\begin{aligned}
&-2\kappa(\psi_1+\psi_2)-4\beta\zeta(\dot{\psi}_1+\dot{\psi}_2)\\
&-3\lambda^{5/3}\dot{\psi}_2^2\sin(\alpha-\varphi+\psi_2)+2(1+\kappa)\varphi+4(1+2\beta)\zeta\dot{\varphi}\\
&+3\lambda^{5/3}\dot{\psi}_1^2\sin(\alpha+\varphi-\psi_1)+2(1+6\lambda^{4/3})\ddot{\varphi}\\
&+3\lambda^{5/3}\ddot{\psi}_1 cos(\alpha+\varphi-\psi_1)-3\lambda^{5/3}\ddot{\psi}_2\cos(\alpha-\varphi+\psi_2) \;=\; 2M\cos(\Omega t), \qquad (6.1)\\
&2\kappa\varphi+4\beta\zeta\dot{\varphi}+3\lambda^{5/3}\dot{\varphi}^2\sin(\alpha+\varphi-\psi_1)-2\kappa\psi_1\\
&-4\beta\zeta\dot{\psi}_1-3\lambda^{5/3}\ddot{\varphi}\cos(\alpha+\varphi-\psi_1)-2\lambda^2\ddot{\psi}_1 \;=\; 0, \qquad (6.2)\\
&2\kappa\varphi+4\beta\zeta\dot{\varphi}-3\lambda^{5/3}\dot{\varphi}^2\sin(\alpha-\varphi+\psi_2)-2\kappa\psi_1\\
&-4\beta\zeta\dot{\psi}_2-3\lambda^{5/3}\ddot{\varphi}\cos(\alpha-\varphi+\psi_2)+2\lambda^2\ddot{\psi}_2 \;=\; 0. \qquad (6.3)
\end{aligned}
$$

To examine basins of attraction, the Eqs. (6.1-6.3) are transformed into a system of six first-order ordinary differential equations by introducing $y_0 = \varphi, y_1 = \dot{\varphi}, y_2 = \psi_1, y_3 = \dot{\psi}_1, y_4 = \psi_2, y_5 = \dot{\psi}_2$. The transformed system is more suitable for numerical simulations and the analysis of attractors and the corresponding basins in phase-space, herein not reported due to a cumbersome form caused by strong nonlinearities.

6.1.2 Bifurcation diagrams and attractors

With dimensionless damping coefficient values adopted from [19] in the interval $\zeta = (0.02, 0.03)$, we can realistically represent tree dynamics. Hence, we search for multi-stability regions by plotting the trunk segment frequency-amplitude diagrams for $\zeta = 0.02$ in Fig. 6.2, $\zeta = 0.025$ in Fig. 6.3 and $\zeta = 0.03$ in Fig. 6.4. The plots show the angular displacement y_0 and the angular velocity y_1 in the excitation frequency interval $\omega = (0, 3.6)$.

Considering that it is not possible (with HPC platforms at our disposal) to achieve acceptable resolution of full-dimensional basins for large 6D state-space windows, we look for multi-stable regions without high (resonant) amplitudes.

Such regions of interest exist within the excitation interval: $\omega = (1.4, 1.7)$ for $\zeta = 0.02$, plotted in Fig. 6.5; $\omega = (1.5, 1.65)$ for $\zeta = 0.025$ in Fig. 6.6; and $\omega = (1.54, 1.64)$ for $\zeta = 0.03$ in Fig. 6.7.

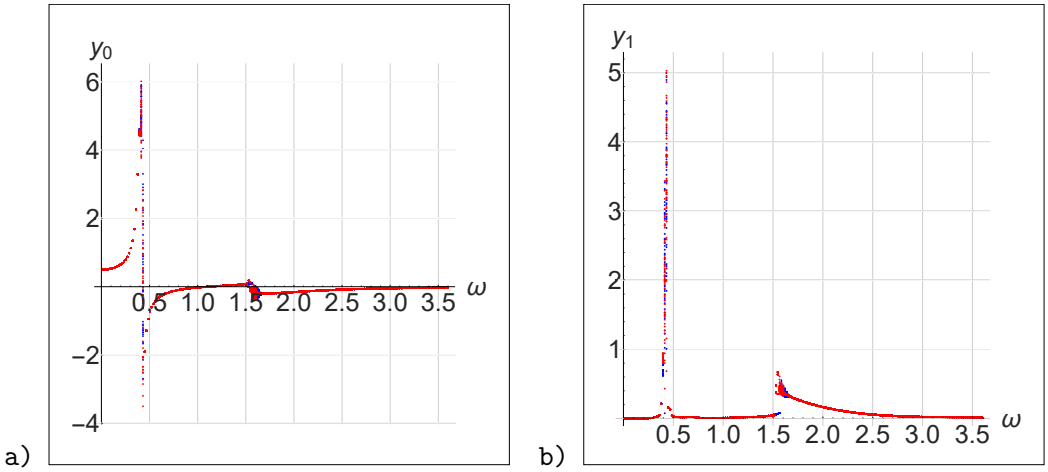

FIGURE 6.3: Brute force bifurcation diagram for $\zeta = 0.025$ in excitation frequencies interval $\omega = (0, 3.6)$

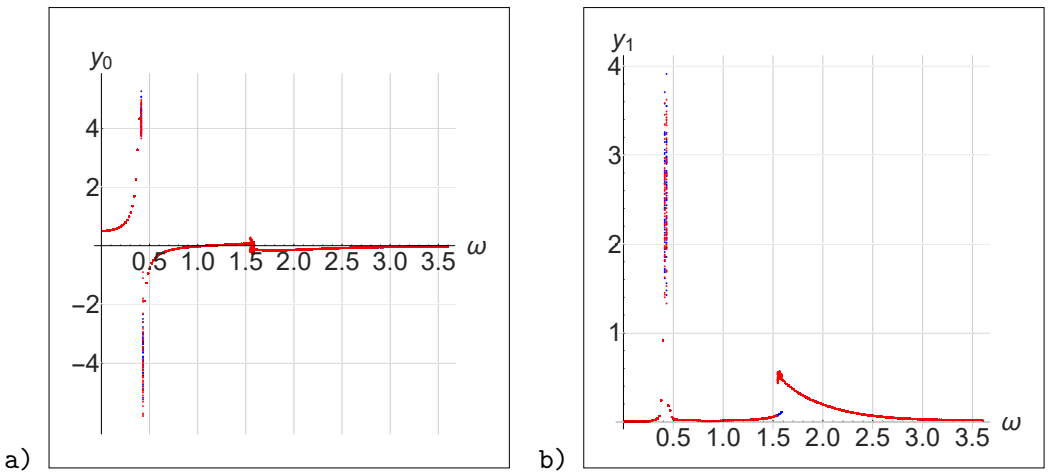

FIGURE 6.4: Brute force bifurcation diagram for $\zeta = 0.03$ in excitation frequencies interval $\omega = (0, 3.6)$

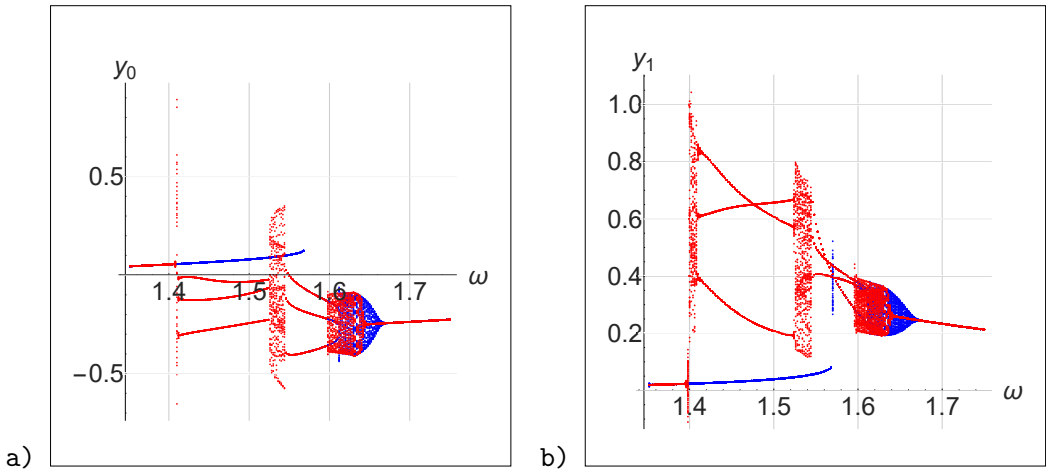

FIGURE 6.5: Brute force bifurcation diagram for $\zeta = 0.02$ in excitation frequencies interval $\omega = (1.35, 1.75)$

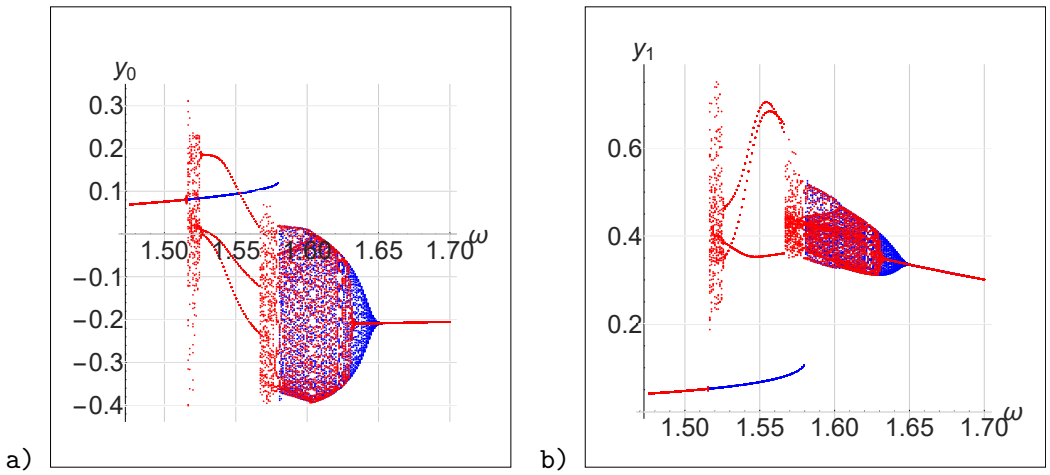

FIGURE 6.6: Brute force bifurcation diagram for $\zeta = 0.025$ in excitation frequencies interval $\omega = (1.47, 1.7)$

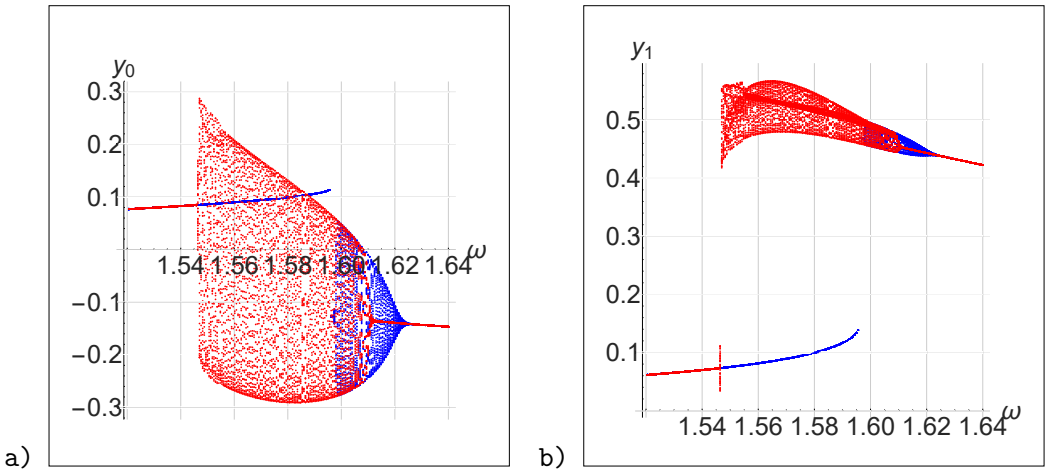

FIGURE 6.7: Brute force bifurcation diagram for $\zeta = 0.03$ in excitation frequencies interval $\omega = (1.52, 1.64)$

Before analysing the bifurcation diagrams in Figures 6.5-6.7, it is important to notice that the regions around bifurcation points are characterized with apparent multi-stability. Down-sweep trajectories have very long transients, which makes it very difficult to numerically compute the actual steady-state. Consequently, it falsely seams that a QP attractor coexist with a PR one, but in fact those trajectories are still being in a transient phase, which eventually settles on the PR attractor.

This (numerical) phenomenon resembles the case of ghost attractors that can happen near bifurcations when some attractor disappears catastrophically [87, 88]. Trajectories linger within the state-space where the attractor existed for a very long (transient) time. Those trajectories resemble the structure of the former attractors and it can falsely lead to the conclusion that a particular steady-state still exists. This conclusion was made by looking at the decrease rate of the transient amplitudes. Namely, the difference could not be noticed (neither numerically nor visually), unless the compared amplitudes were several thousand of periods apart. The very slow (but steady) decreasing tendency made us conclude that the trajectories converge to the a PR attractor. Theoretically, this and similar issues could be eliminated by a longer integration, however our numerical tools were not able to compute trajectories for such long time periods (more than hundred thousands of periods).

For example, some of the frequencies where this phenomenon occurs are in Fig. 6.5 near $\omega = 1.65$, in Fig. 6.6 near $\omega = 1.64$, in Fig. 6.7 near $\omega = 1.615$ and in Fig. 6.8 near $\omega = 2.95$ and $\omega = 3.155$.

Consequently, also the region in Fig. 6.8 is not multi-stable, and therefore not considered for basins building, although it has rich nonlinear behavior: a widespread QP motion with short PR windows for some frequencies.

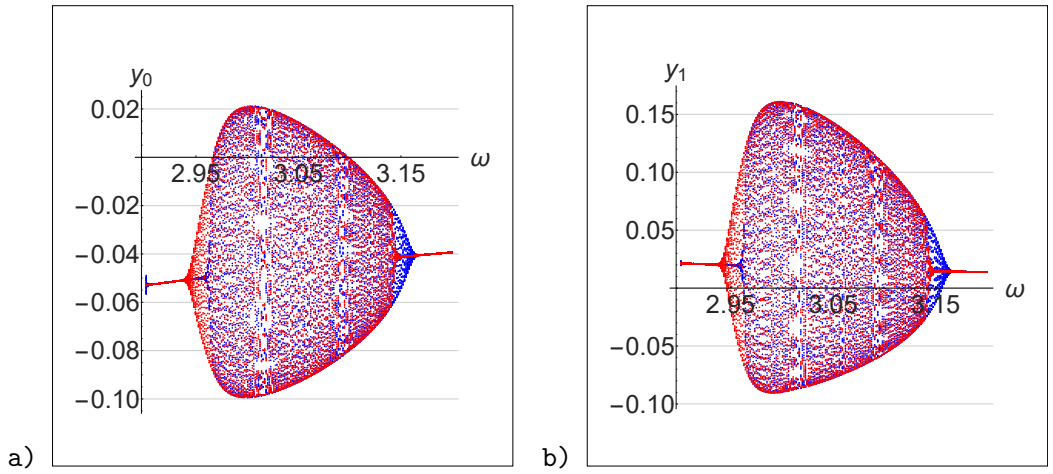

FIGURE 6.8: Brute force bifurcation diagram for $\zeta = 0.02$ in excitation frequencies interval $\omega = (2.9, 3.2)$

The similar phenomenon also occurs near the bifurcation points where two PR attractors with different periodicity are involved, as in Fig. 6.5 at $\omega = 1.4$ or in Fig. 6.6 near the frequencies $\omega = 1.52$ and $\omega = 1.58$.

By looking at the bifurcation diagrams, we see that for the system of equations (6.1-6.3), we have three types of steady-state responses: period one (PR1), period three (PR3) and quasi-periodic. Within multi-stable regions, we have PR1-PR3, PR1-QP and PR3-QP combinations of the coexisting attractors. Those multi-stable regions and attractor combinations that exist are summarised in Table 6.1. For $\zeta = 0.02$, near $\omega = 1.6$, we were not able to confirm if it is a ghost or the QP attractor - beside PR3 attractor - direct integration shows a trajectory that behaves as a QP steady state, while both NSSCM and GoS report only PR3 behaviour (no multi-stability). Consequently, leading to the conclusion that QP is a repellor, which often cannot be discovered with the basins of attraction searching methods. Some regions exist for very narrow band of frequencies, thus, they are represented with a single ω value only. Moreover, the exact values of frequencies where bifurcation points occur are numerically difficult to be precisely determined, thus the values of ω interval boundaries are approximate.

TABLE 6.1: Summary of multi-stable regions of STREE system and combinations of coexisting attractors.

$\zeta = 0.02$	$\omega = (1.4, 1.525)$	$\omega = (1.525, 1.545)$	$\omega = (1.545, 1.57)$
	PR1-PR3	PR1-QP	PR1-PR3
$\zeta = 0.025$	$\omega = (1.515, 1.575)$	$\omega = 1.5755$	-
	PR1-PR3	PR3-QP	-
$\zeta = 0.03$	$\omega = (1.545, 1.595)$	-	-
	PR1-QP	-	-

To illustrate the steady-state dynamics within the state-space window $y_i = (-3, 3)$, we plot the corresponding attractors for the values of ζ and ω that were considered for building the basins.

For $\zeta = 0.02$, at the frequency $\omega = 1.45$, the PR1 attractor coexists with a PR3 one. The projections of their trajectories (PR1 colored in blue and PR3 in red) on the y_0, y_1 and y_0, y_2 plane are plotted in Fig. 6.9.

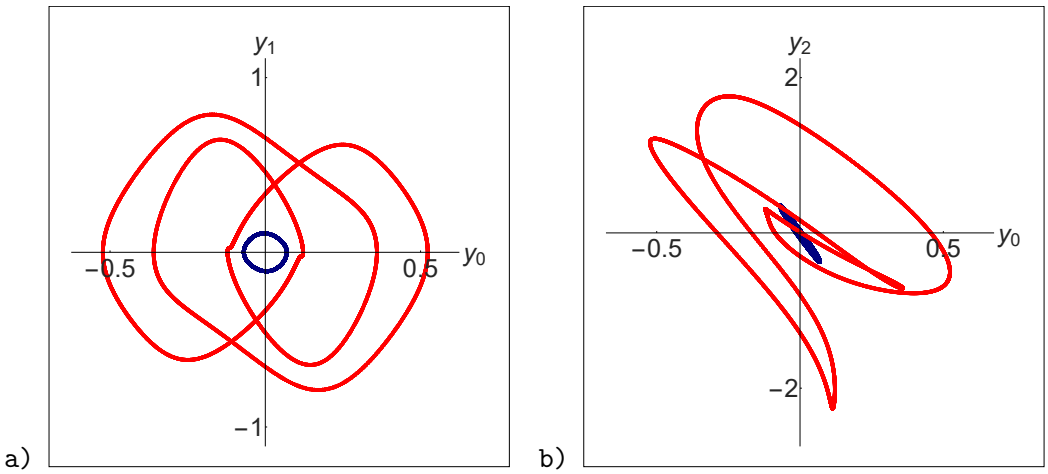

FIGURE 6.9: Trajectories of PR1 (blue) and PR3 (red) attractors, projected on a) y_0, y_1 and b) y_0, y_2 plane, obtained for $\zeta = 0.02$, $\omega = 1.45$.

For the frequency $\omega = 1.53$, there is a combination of PR1 and QP multi-stable behavior, which is plotted in Fig. 6.10 on the Poincaré sections in the y_0, y_1 and y_0, y_2 plane.

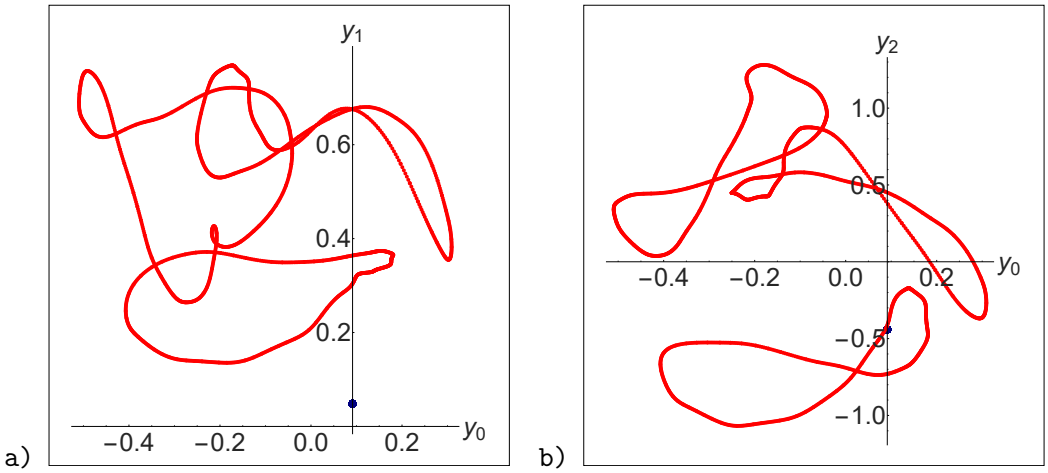

FIGURE 6.10: Poincaré sections of PR1 (blue) and QP (red) attractors in a) y_0, y_1 and b) y_0, y_2 plane, obtained for $\zeta = 0.02$, $\omega = 1.53$.

Another PR1-PR3 behaviour is observed for $\zeta = 0.025$ at the excitation frequency $\omega = 1.55$. In Fig. 6.11, which illustrates projections on the y_0, y_1 and y_0, y_2 planes, the trajectory of the PR1 attractor is colored in blue and of the PR3 in red.

The settings where the PR3 and QP attractors exist simultaneously happen for $\zeta = 0.025$ in a very narrow frequency band near $\omega = 1.5755$. The points at which those attractors intersect Poincaré sections in the y_0, y_1 and y_0, y_2 plane are plotted in Fig. 6.12, (PR3 colored i blue and QP in red).

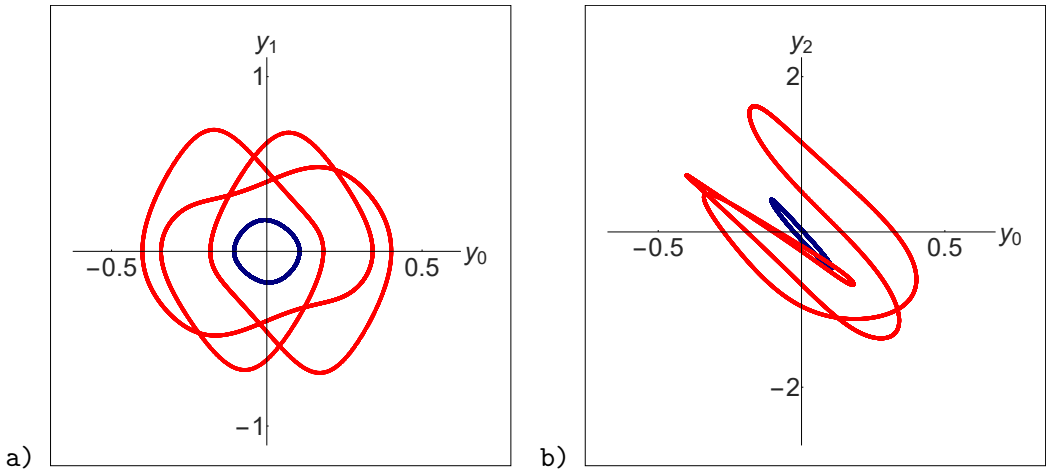

FIGURE 6.11: Trajectories of PR1 (blue) and PR3 (red) attractors, projected on a) y_0, y_1 and b) y_0, y_2 plane, obtained for $\zeta = 0.025$, $\omega = 1.55$.

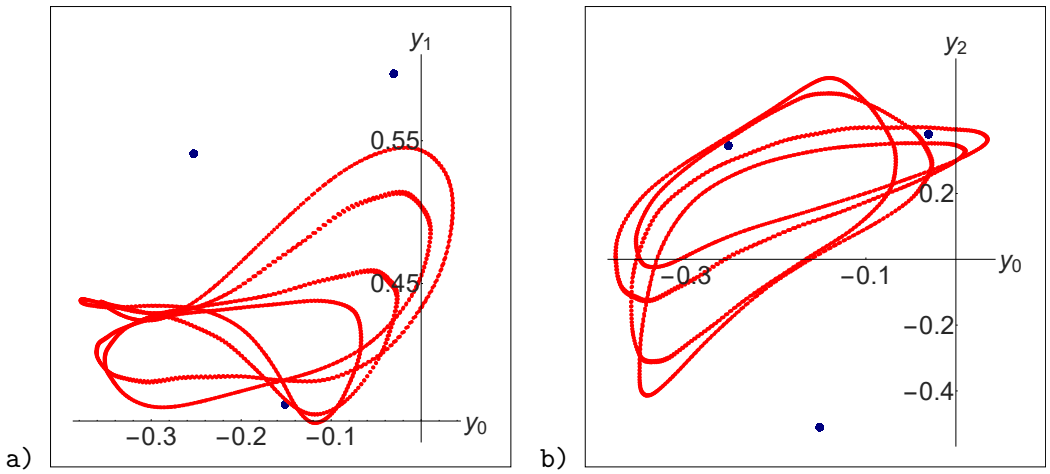

FIGURE 6.12: Poincaré sections of PR3 (blue) and QP (red) attractors in a) y_0, y_1 and b) y_0, y_2 plane, obtained for $\zeta = 0.025$, $\omega = 1.5755$.

For a higher value of damping, $\zeta = 0.03$, only one multi-stable region is detected. Within it, there is a PR1 attractor, whose intersections with the Poincaré section y_0, y_1 and the y_0, y_2 plane colored in blue in Fig. 6.13, while a QP is colored in red.

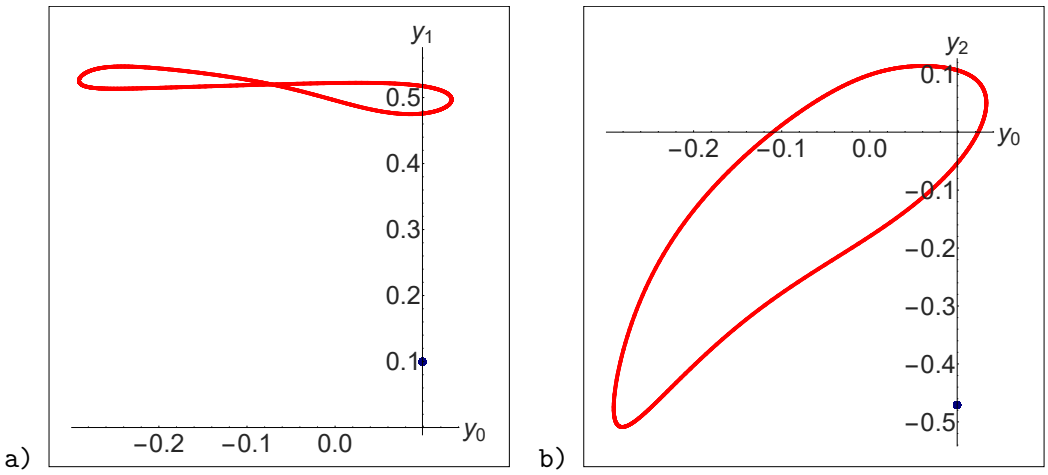

FIGURE 6.13: Poincaré sections of PR1 (blue) and QP (red) attractors in a) y_0, y_1 and b) y_0, y_2 plane, obtained for $\zeta = 0.03$, $\omega = 1.58$.

6.1.3 Basins of attraction: structure and integrity

Long transients with high amplitudes cause significant difficulties for precise computation of the global steady-state behaviour for STREE system. GoS requires on average 250 period long integrations to determine a basin with adequate certainty. It is utterly unacceptable in 6D for a full-dimensional basin analysis. Thus, we have to accept that there may be some cross-sections with less accurate basins than preferred. When this situation happens on a gross part of the basin, the computations must be prolonged. In our case, the sufficient basin accuracy was achieved for integrations over 24 periods.

Basin are computed on the CL-C cluster, with computational setting (the number of parallel tasks per a NSSCM stage) the same as in Table 5.2. Computations of image cells lasted about 7 *hours*, map post-processing 11 *minutes*, basin file writing 15 *minutes* and computation of integrity factor from 2 to 5 *hours* (longer time was required for more robust basins). Certain properties of the computed attractors (with the SCM, NSSCM and GoS) and the basin for examined settings are summarised in Table 6.2.

TABLE 6.2: Properties of STREE attractors and basins.

	attractors (SCM/NSSCM/GoS)	integrity factor (*number of cells*)
$\zeta = 0.02$	PR1 - PR3	PR1 (12)
$\omega = 1.45$	13 / 3 / 2	PR3 (4)
$\zeta = 0.02$	PR1 - QP	PR1 (12)
$\omega = 1.53$	10 / 4 / 2	QP (4)
$\zeta = 0.025$	PR1 - PR3	PR1 (15)
$\omega = 1.55$	4 / 2 / 2	PR3 (3)
$\zeta = 0.025$	PR1 - QP	PR1 (14)
$\omega = 1.5755$	6 / 3 / 2	QP (3)
$\zeta = 0.03$	PR1 - QP	PR1 (18)
$\omega = 1.58$	3 / 2 / 2	QP (3)

What the examination of the bifurcation diagrams showed and the integrity analysis confirmed (Table 6.2, integrity factor), is that the **PR1 motion is predominant behavior of the STREE system**. To visualize this setting, we examine basins structure for conditions that may occur in nature. With this simple tree model, we can consider wind as a source of non-zero initial angular velocities, beside its influence as harmonic excitation. This illustrates the impact of initial conditions on the global behaviour of the system (6.1-6.3) and how it influences the measures of dynamical integrity, and consequently its overall robustness.

Thus, we take into account the motion with zero initial angular displacements[3], which may be regarded as wind conditions. Now, we examine the behaviour of each segment by looking at its basin structure - the relationship of displacement and velocity (angular) under different initial configurations. For clarity of figures, the axes are not drawn in the following sections 6.1.3.1, 6.1.3.2 and 6.1.3.3, and all the plots show the $y_i = (-3, 3)$ state-space window.

6.1.3.1 Basin structure for PR1-PR3 attractor combination at $\zeta = 0.02$ and $\omega = 1.45$

It is evident from Fig. 6.14 that only small portion of initial conditions lead to PR3 steady-state. It requires to increase the magnitude of initial angular velocity of trunk itself (y_1) and right branch (y_5), while the left branch (angular velocity y_3) does not have particular influence.

The left branch y_2-y_3 cross-section area is, for higher initial condition magnitudes, roughly equally covered with both P1 and P3 basins, as shown in Fig. 6.15. However, it is the consequence of inaccuracies caused by large amplitudes, where SCM/NSSCM cannot precisely determine trajectories. In reality those cross-sections (validated with direct integration) show no traces of a P3 basin.

[3] the exact zero values could not be achieved as a consequence of the discretization scheme

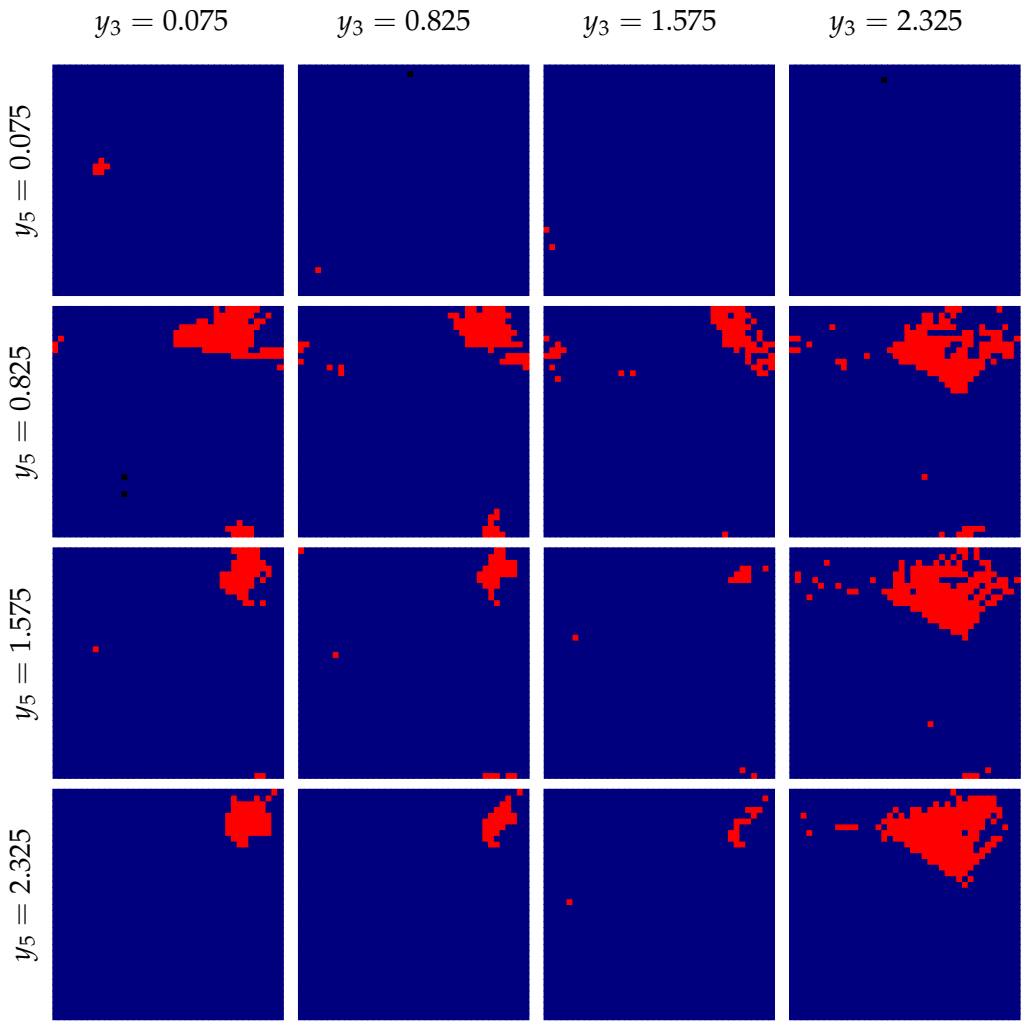

FIGURE 6.14: Basin structure of trunk segment in y_0-y_1 plane, for $\zeta = 0.02$ and $\omega = 1.45$), with the change in initial magnitude of branches angular velocities y_3 and y_5, while initial angular displacements are $y_{2,4} = 0.075$ (close to zero). In blue is basin of the PR1, in red of the PR3 attractor.

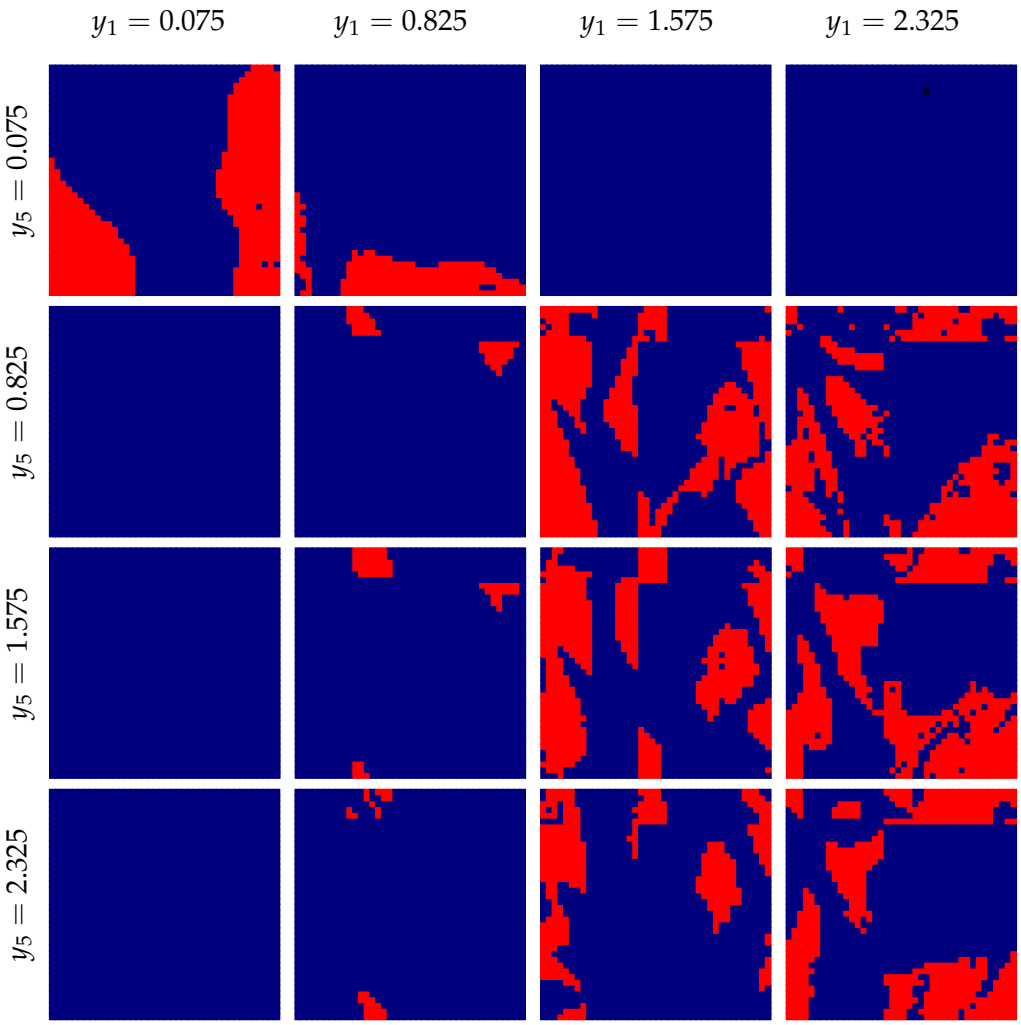

FIGURE 6.15: Basin structure of left branch segment in y_2-y_3 plane, for $\zeta = 0.02$ and $\omega = 1.45$, with the change in initial magnitude of trunk and right branch angular velocities y_1 and y_5, while initial angular displacements are $y_{0,4} = 0.075$ (close to zero). In blue is basin of the PR1, in red of the PR3 attractor.

It is evident from Fig. 6.16 that the magnitude of the initial angular velocity of the trunk has a major impact on steady-state behaviour of right branch segment. The motion is principally P1 with some P3 regions where the initial magnitudes of y_3 and y_5 are higher than of y_1.

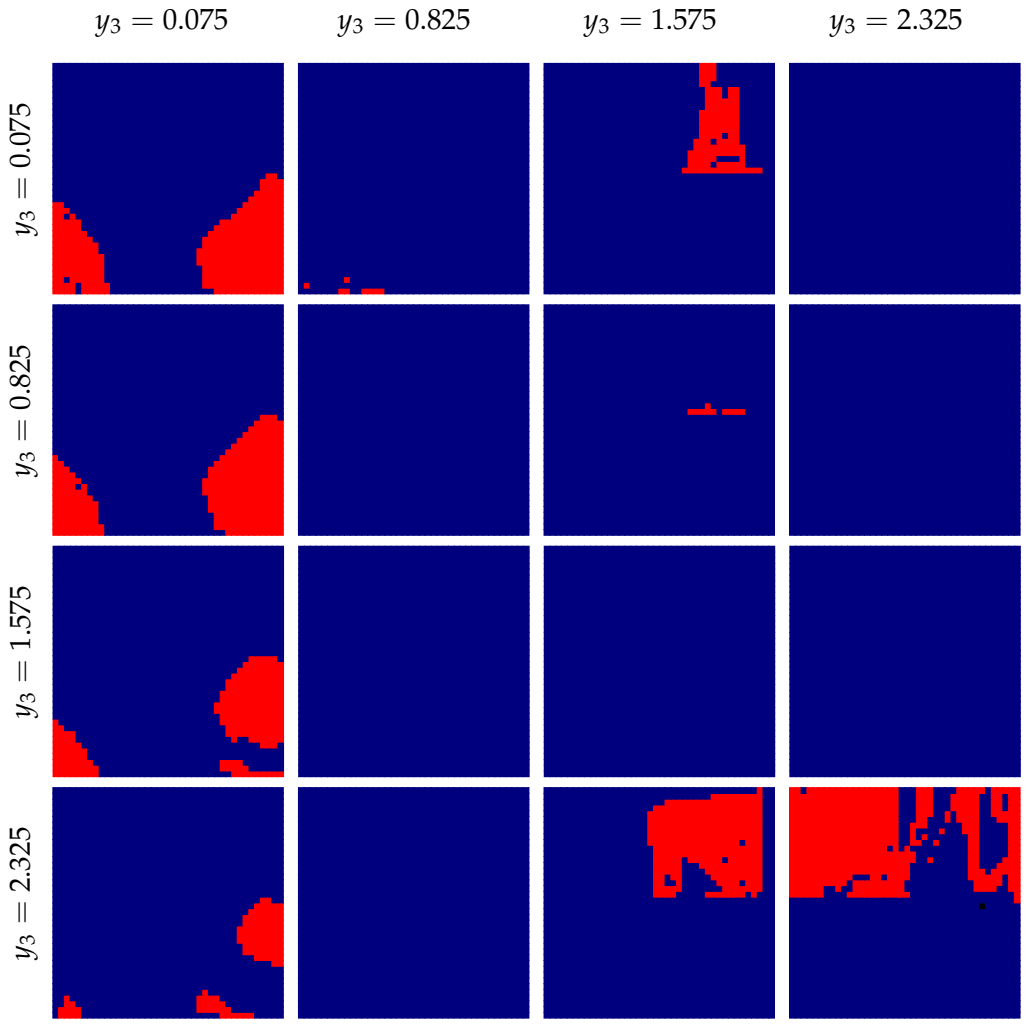

FIGURE 6.16: Basin structure of right branch segment in y_4-y_5 plane, for $\zeta = 0.02$ and $\omega = 1.45$, with the change in initial magnitude of trunk and left branch angular velocities y_3 and y_5, while initial angular displacements are $y_{0,2} = 0.075$ (close to zero). In blue is basin of the PR1, in red of the PR3 attractor.

6.1.3.2 Basin structure for PR3-QP attractor combination at $\zeta = 0.025$ and $\omega = 1.5755$

The specific case where the STREE system have a PR3 and QP attractors happens only in the very narrow frequency band around $\omega = 1.5755$ for $\zeta = 0.025$. The PR3 behaviour of the trunk segment is evidently dominant over the QP motion, which may be seen in Fig. 6.17.

The global behaviour of the branches is quite similar to each other. We can conclude from Figures 6.18 and 6.19 that the QP steady-state of both branches is on a par with PR3 only when the initial magnitude of the trunk angular velocity is close to zero. Otherwise, the PR3 is a predominant attracting behaviour.

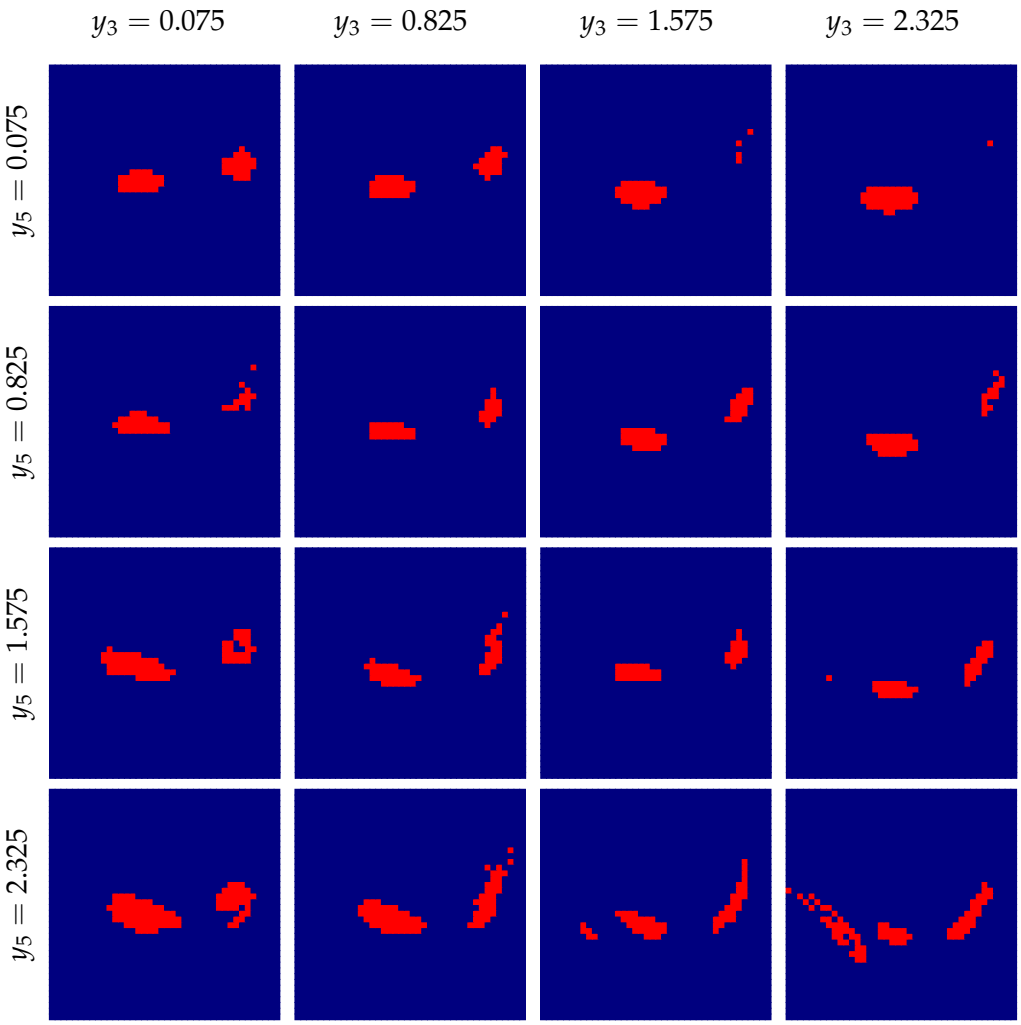

FIGURE 6.17: Basin structure of trunk segment in y_0-y_1 plane, for $\zeta = 0.025$ and $\omega = 1.5755$), with the change in initial magnitude of branches angular velocities y_3 and y_5, while initial angular displacements are $y_{2,4} = 0.075$ (close to zero). In blue is basin of the PR3, in red of the QP attractor.

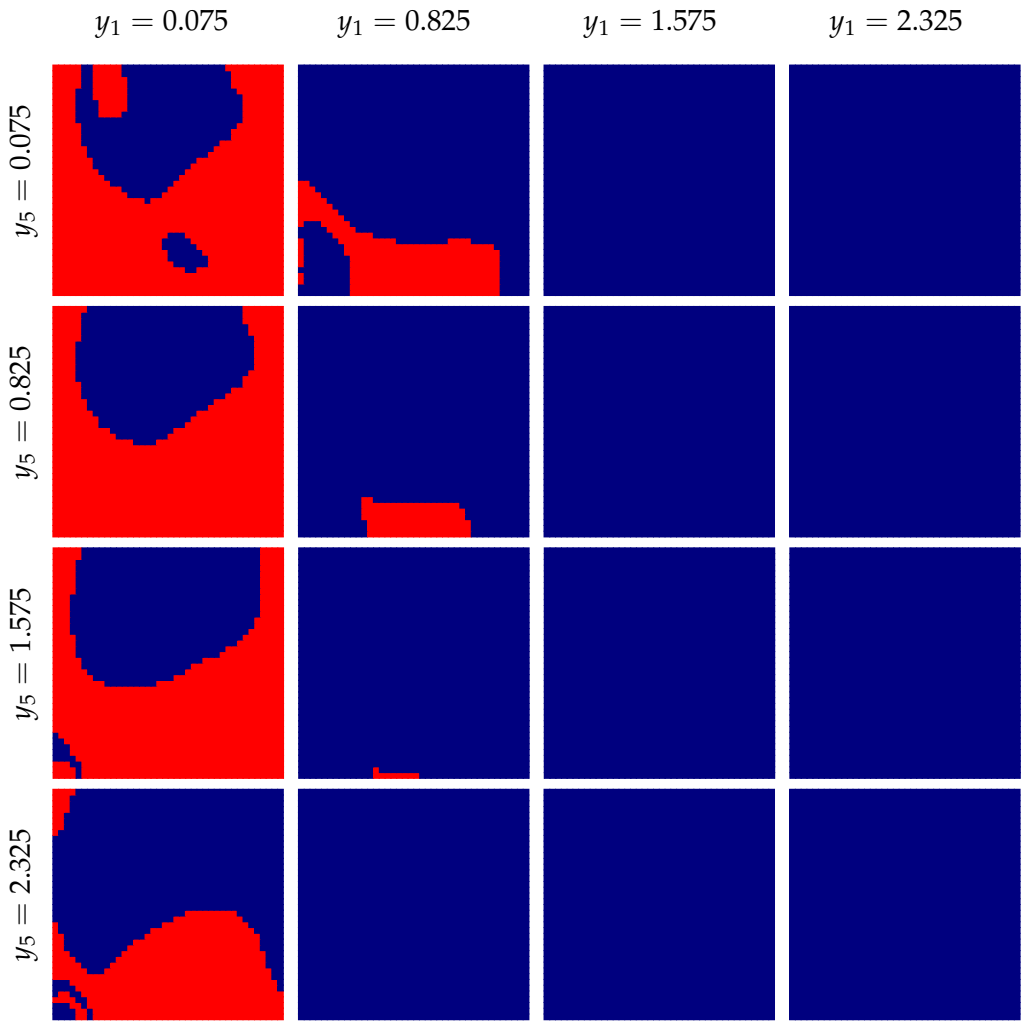

FIGURE 6.18: Basin structure of left branch segment in y_2-y_3 plane, for $\zeta = 0.025$ and $\omega = 1.5755$, with the change in initial magnitude of trunk and right branch angular velocities y_1 and y_5, while initial angular displacements are $y_{0,4} = 0.075$ (close to zero). In blue is basin of the PR3, in red of the QP attractor.

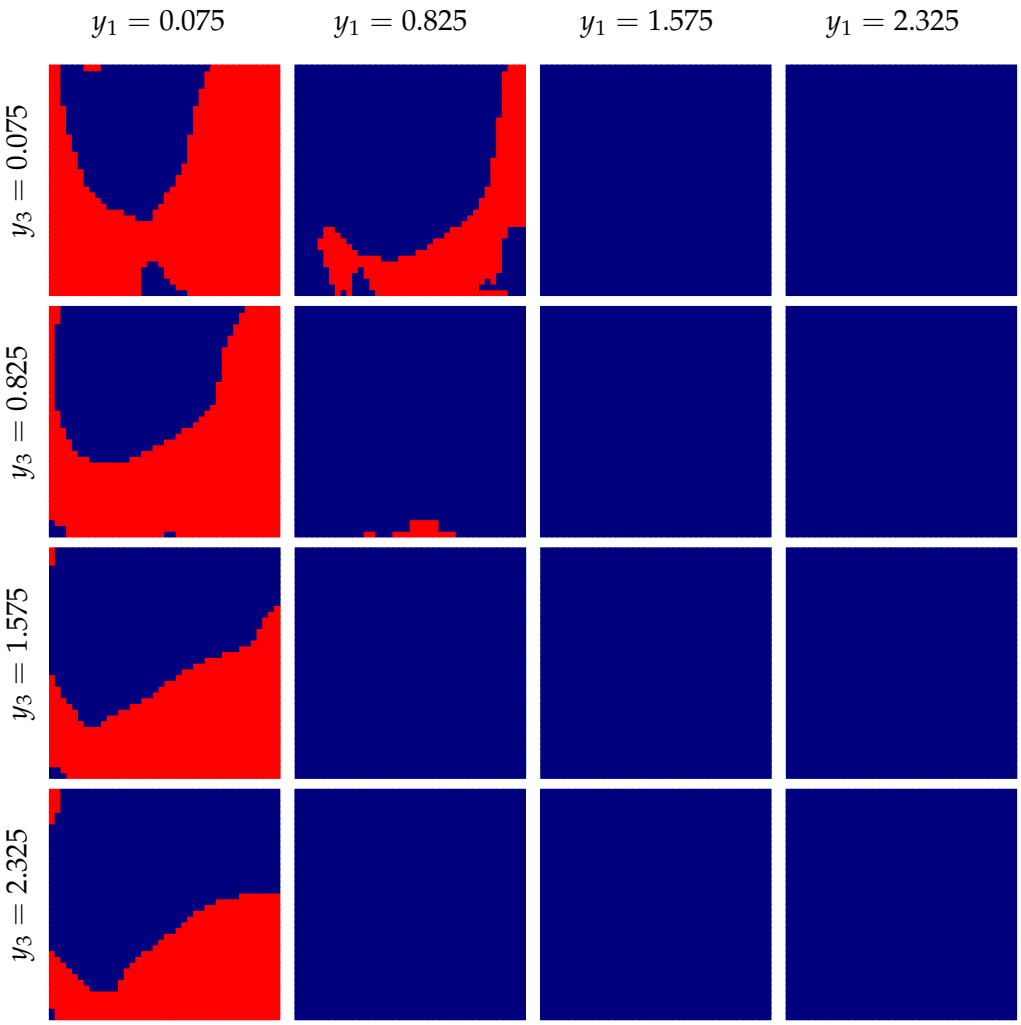

FIGURE 6.19: Basin structure of right branch segment in y_4-y_5 plane, for $\zeta = 0.025$ and $\omega = 1.5755$, with the change in initial magnitude of trunk and left branch angular velocities y_3 and y_5, while initial angular displacements are $y_{0,2} = 0.075$ (close to zero). In blue is basin of the PR3, in red of the QP attractor.

6.1.3.3 Basin structure for PR1-QP attractor combination at $\zeta = 0.03$ and $\omega = 1.58$

The case with $\zeta = 0.03$ expectedly shows that higher damping greatly suppresses irregular motions, such as QP for $\omega = 1.58$. In Figure 6.20 plotted for the trunk segment steady-state behaviour we see only traces of the QP basin, which was small, but clearly present, in case with lower damping ($\zeta = 0.025$) and similar frequency ($\omega = 1.5755$) in Fig. 6.17.

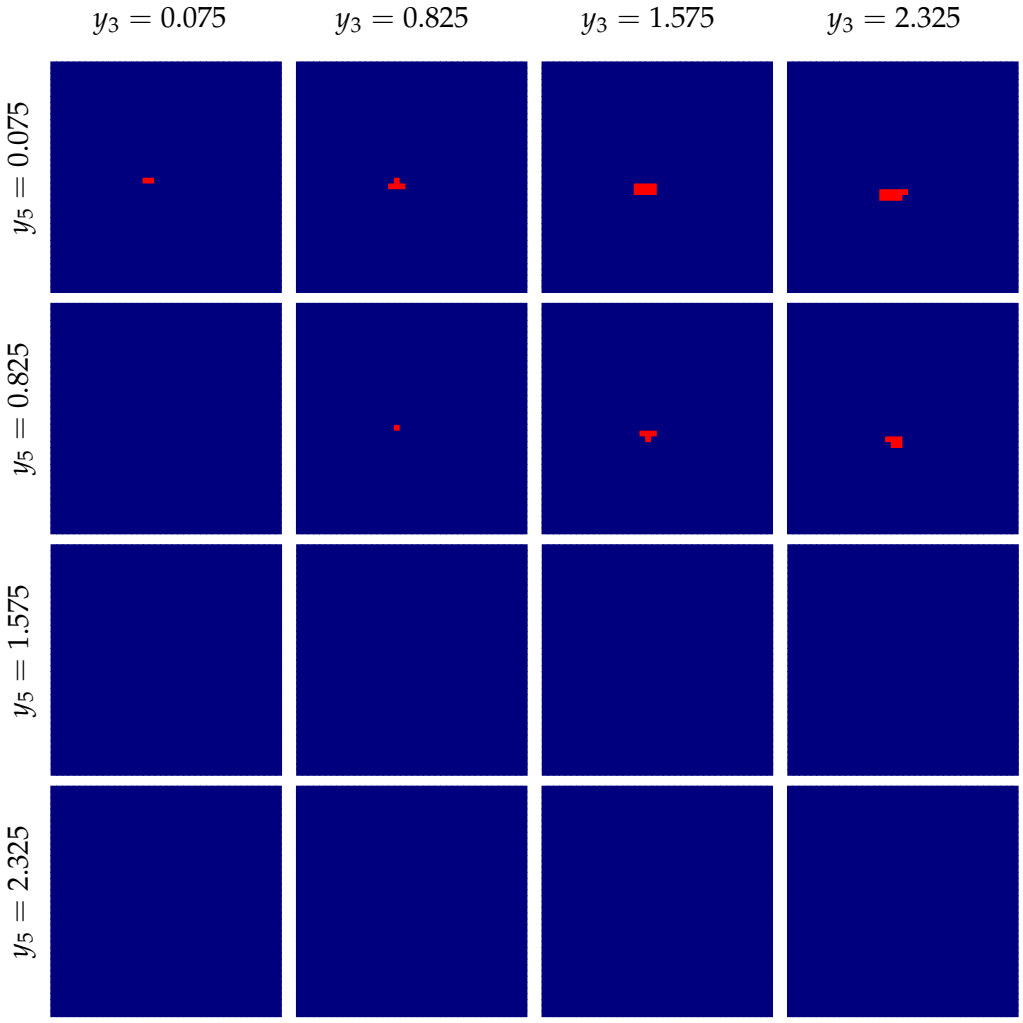

FIGURE 6.20: Basin structure of trunk segment in y_0-y_1 plane, for $\zeta = 0.03$ and $\omega = 1.58$), with the change in initial magnitude of branches angular velocities y_3 and y_5, while initial angular displacements are $y_{2,4} = 0.075$ (close to zero). In blue is basin of the PR1, in red of the QP attractor.

In comparison with a less damped case, we notice from Fig. 6.21 and 6.22 that QP basins (for low values of the trunk initial velocity) is still compact, however it covers a smaller part of the examined region.

6.2 Rotating hub with two pendulums

Rotating structures such as jet engine turbines, helicopter blades and wind turbines, are very important in aerospatial and mechanical engineering. The authors of [20]

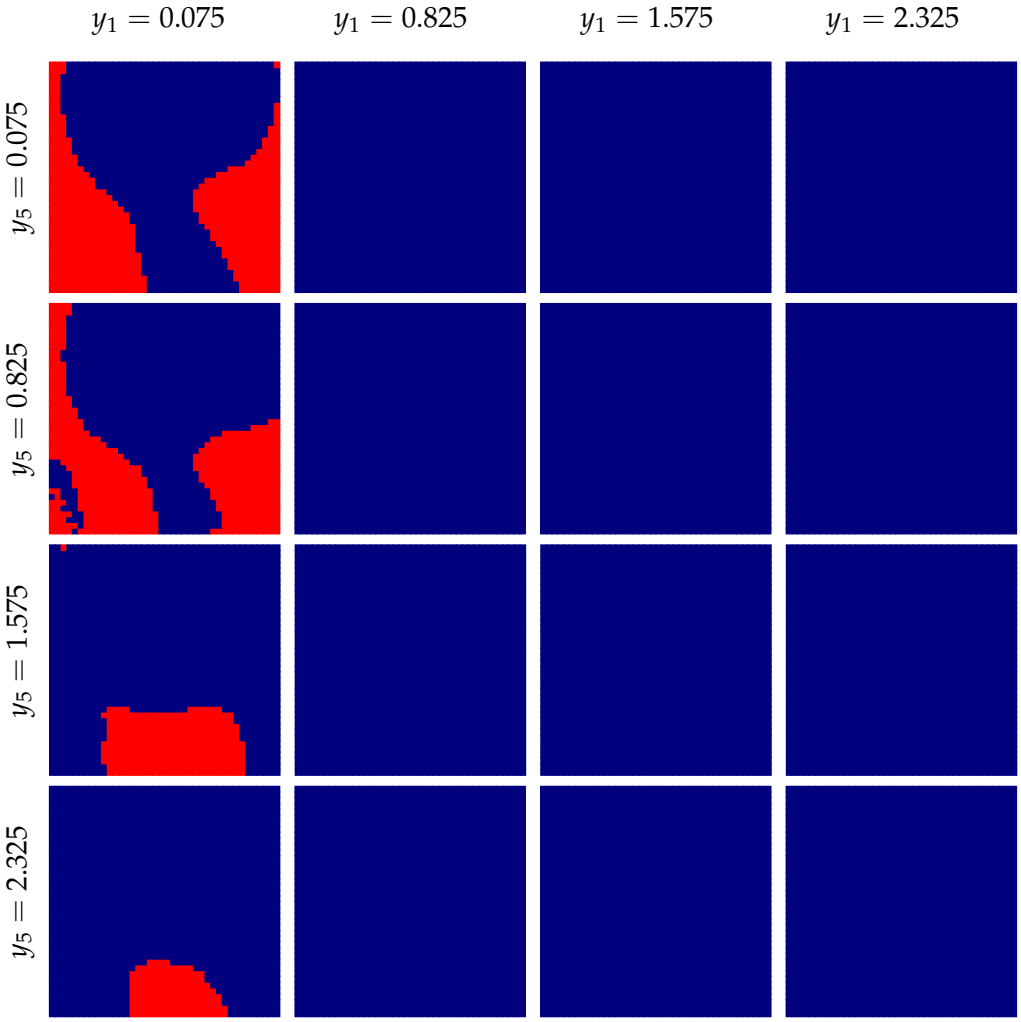

FIGURE 6.21: Basin structure of left branch segment in y_2-y_3 plane, for $\zeta = 0.03$ and $\omega = 1.58$, with the change in initial magnitude of trunk and right branch angular velocities y_1 and y_5, while initial angular displacements are $y_{0,4} = 0.075$ (close to zero). In blue is basin of the PR1, in red of the QP attractor.

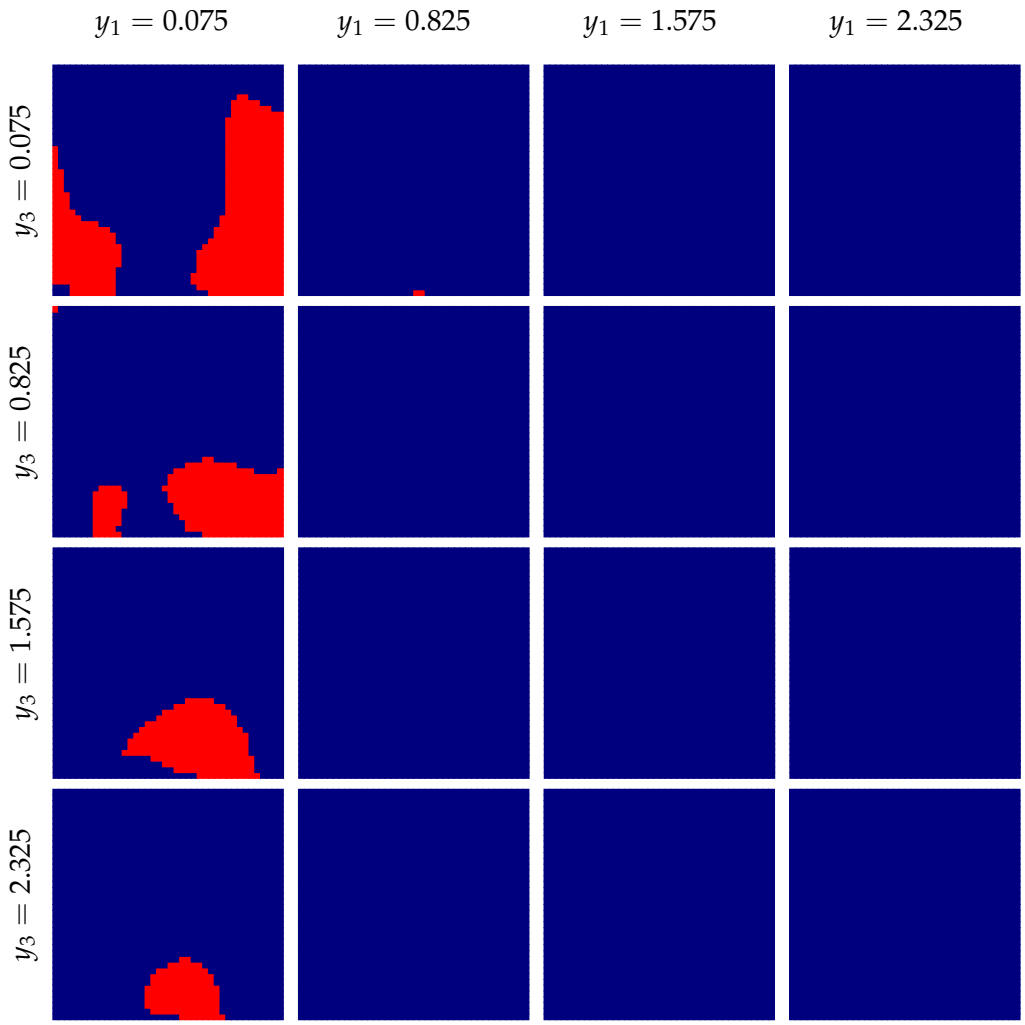

FIGURE 6.22: Basin structure of right branch segment in y_4-y_5 plane, for $\zeta = 0.03$ and $\omega = 1.58$, with the change in initial magnitude of trunk and left branch angular velocities y_3 and y_5, while initial angular displacements are $y_{0,2} = 0.075$ (close to zero). In blue is basin of the PR1, in red of the QP attractor.

examined a system composed of two pendula attached to a hub, which rotates in a horizontal plane. In addition to the published analysis, herein we provide the examples of basin and dynamical integrity for several values of the excitation frequency for this system.

6.2.1 Equations of motion

The considered model of a rotating structure is shown in Fig. 6.23, and is composed of two pendula symmetrically attached to a hub placed in a horizontal plane. The hub with a radius R and a moment of inertia J_0 is hinged in its center by a nonlinear spring with the stiffness coefficient k, k^* (k is the coefficient of the linear and k^* of the cubic polynomial terms). The pendula are modeled as lumped masses attached to a rigid and massless rods with length l_1 and l_2, connected to the hub by a hinged Duffing-like joint. Masses of pendula are m_1 and m_2, while viscosity and stiffness of the nonlinear spring in joints are denoted by c_1, k_1, k_1^* and c_2, k_2, k_2^*, respectively. The generalized coordinate ψ is defined as the absolute angle of hub rotation. The coordinates φ_1 and φ_2 are measured as relative angles of the pendula with respect to a coordinate system attached to the center of the hub. In addition, the hub is excited with a harmonic torque of magnitude M.

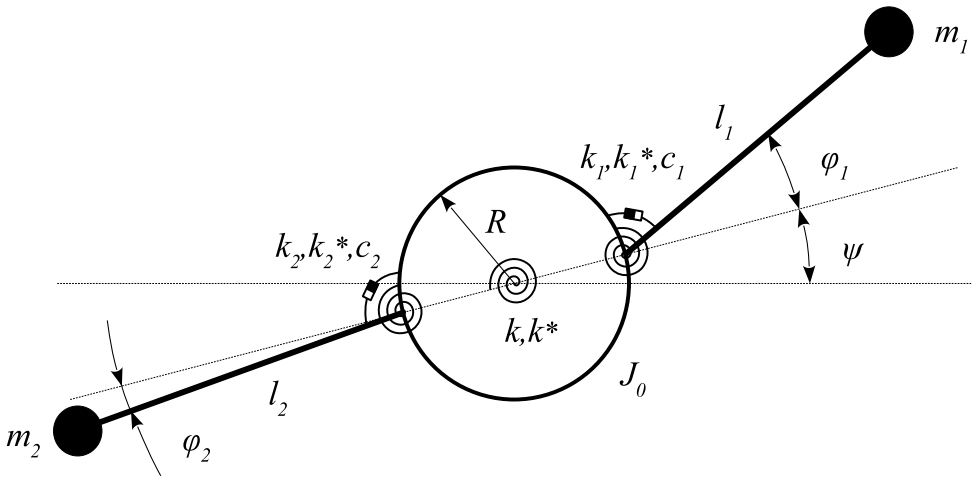

FIGURE 6.23: Model of rotating hub with two pendula.

By introducing dimensionless time $\tau = \omega_{01}^* t$, where $\omega_{01}^* = \sqrt{k_1/m_1 l_1^2}$ is natural frequency of the right pendula, we get the equations of motion in the following dimensionless form:

$$
\begin{aligned}
(1+\gamma_1+\gamma_2)\ddot{\psi}+\zeta_h\dot{\psi}+2\frac{\gamma_1}{\delta_1}\dot{\delta}_1\dot{\psi}& \\
+2\frac{\gamma_2}{\delta_2}\dot{\delta}_2\dot{\psi}+\frac{\gamma_1}{\delta_1}\cos\beta_1\ddot{\varphi}_1+\frac{\gamma_2}{\delta_2}\cos\beta_2\ddot{\varphi}_2+\frac{\gamma_1}{\delta_1^2}\cos\beta_1\dot{\delta}_1\dot{\varphi}_1& \\
+\frac{\gamma_2}{\delta_2^2}\cos\beta_2\dot{\delta}_2\dot{\varphi}_2+\frac{\gamma_1}{\delta_1}\frac{d}{d\tau}(\cos\beta_1)\dot{\varphi}_1+\frac{\gamma_2}{\delta_2}\frac{d}{d\tau}(\cos\beta_2)\dot{\varphi}_2+\kappa_h\psi+K_h\psi^3 &= \mu, \quad (6.4)\\
\ddot{\varphi}_1+\delta_1\cos\beta_1\ddot{\psi}+\delta_1\frac{d}{d\tau}(\cos\beta_1)\dot{\psi}+\cos\beta_1\dot{\delta}_1\dot{\psi}-\delta_1\frac{d\delta_1}{d\varphi_1}\dot{\psi}^2& \\
-\cos\beta_1\dot{\delta}_1\dot{\psi}-\delta_1\frac{d}{d\tau}(\cos\beta_1)\dot{\psi}+\zeta_1\dot{\varphi}_1+\omega_{01}^2\varphi_1+\kappa_1\omega_{01}^2\varphi_1^3 &= 0, \quad (6.5)\\
\ddot{\varphi}_2+\delta_2\cos\beta_2\ddot{\psi}+\delta_2\frac{d}{d\tau}(\cos\beta_2)\dot{\psi}+\cos\beta_2\dot{\delta}_2\dot{\psi}-\delta_2\frac{d\delta_2}{d\varphi_2}\dot{\psi}^2& \\
-\cos\beta_2\dot{\delta}_2\dot{\psi}-\delta_2\frac{d}{d\tau}(\cos\beta_2)\dot{\psi}+\zeta_2\dot{\varphi}_2+\omega_{02}^2\varphi_2+\kappa_2\omega_{02}^2\varphi_2^3 &= 0, \quad (6.6)
\end{aligned}
$$

where $j=1,2$ and

$$
\begin{aligned}
\omega_{0j}=\frac{\omega_{0j}^*}{\omega_{01}^*},\quad \delta_{0j}=\frac{R}{l_j},\quad \delta_j=\frac{R_j}{l_j}=\sqrt{\delta_{0j}^2+1+2\delta_{0j}\cos\varphi_j}& \\
\cos\beta_j=\sqrt{1-\frac{\delta_{0j}^2}{\delta_j^2}\sin^2\varphi_j},\quad \gamma_{0j}=\frac{m_jl_j^2}{J_0},\quad \gamma_j=\gamma_{0j}\delta_j^2,\quad \kappa_j=\frac{k_j^*}{k_1},& \\
\kappa_h=\gamma_{01}k,\quad K_h=\gamma_{01}k^*,\quad \zeta_j=\frac{c_j}{m_jl_j^2\omega_{01}^*},\quad \zeta_h=\frac{c_h}{J_0\omega_{01}^*},\quad \mu=\frac{M}{J_0\omega_{01}^{*2}}.& \quad (6.7)
\end{aligned}
$$

Equations (6.4-6.6) are transformed into the system of six first-order ordinary differential equations by introducing the following relations: $y_0=\psi, y_1=\dot{\psi}, y_2=\varphi_1, y_3=\dot{\varphi}_1, y_4=\varphi_2, y_5=\dot{\varphi}_2$. The transformed systems is not reported due to its very cumbersome form.

6.2.2 Bifurcation diagrams and attractors

The brute force bifurcation diagrams in Fig. 6.24 show two multi-stability regions for the parameter values $\delta_{11}=0.24$, $\delta_{22}=0.26$, $\gamma_{01}=0.0444$, $\gamma_{02}=0.0444$, $\kappa_1=0.5$, $\kappa_2=0.5$, $\omega_1=1$, $\omega_2=1$, $\zeta_1=0.1$, $\zeta_2=0.1$, $\zeta_h=0.1$, $\kappa_h=1$, $K_h=1$ and the excitation term in the form $\mu\cos\omega t$ with the amplitude $\mu=0.3$.

The multi-stable region (without resonant amplitudes) that can be examined by adequate resolution with the NSSCM exists in the excitation frequency interval $\omega=(1.2,1.65)$, as plotted in Fig. 6.25.

Three PR1 attractors exist within the $y_i=(-3,3)$ window. The projection of their trajectories on the y_0-y_1, y_2-y_3 and y_4-y_5 planes are plotted in Figures 6.26-6.28 for $\omega=1.4$, $\omega=1.5$ and $\omega=1.6$, respectively.

6.2.3 Basins of attraction: structure and integrity

It is evident that for the chosen parameter values, and the excitation frequency interval $\omega=(1.2,1.65)$, three PR1 distinct steady-state motions compete within the $y_i=(-3,3)$ window. To determine if any of them is predominant and what is their

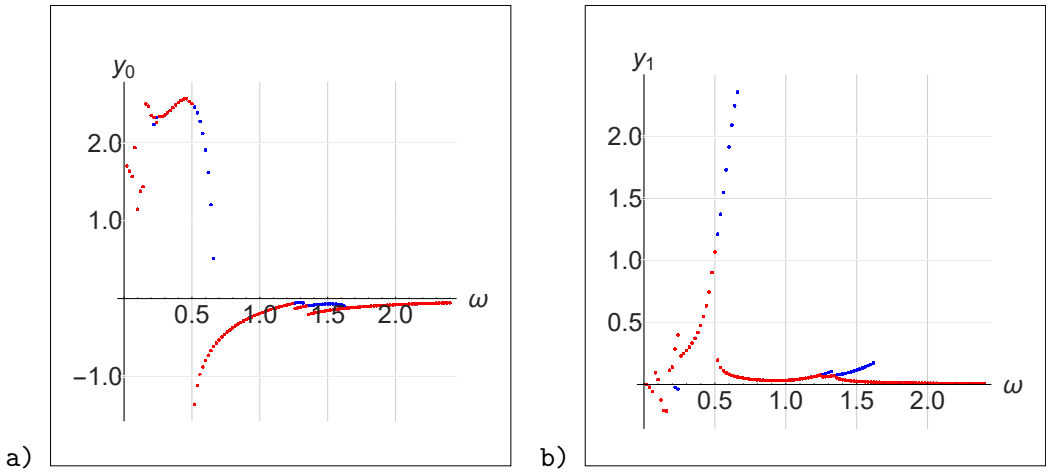

FIGURE 6.24: Brute force bifurcation diagrams of RHUB system for $\omega = (0, 2.4)$.

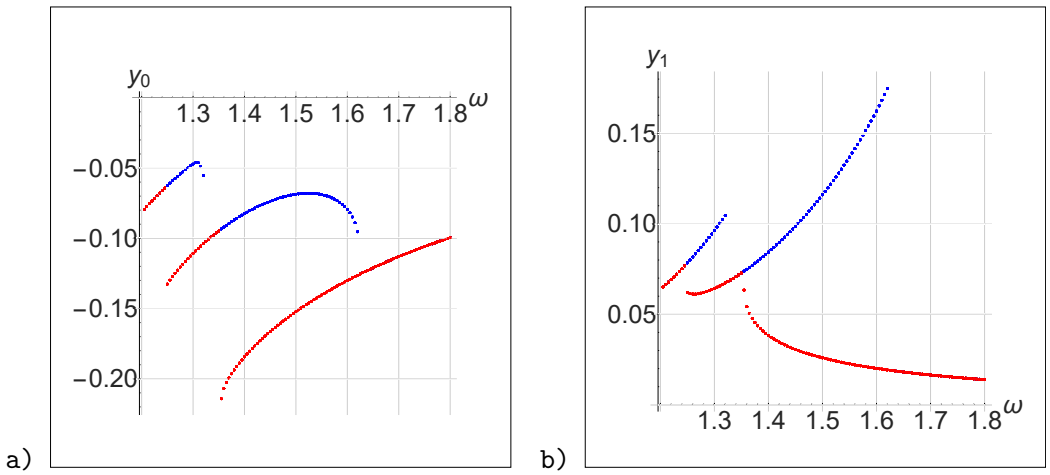

FIGURE 6.25: Brute force bifurcation diagrams of RHUB system for $\omega = (1.2, 1.65)$.

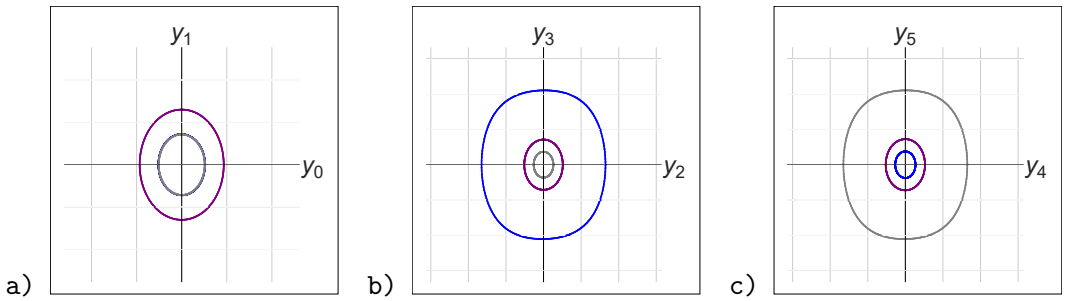

FIGURE 6.26: Trajectories of RHUB system attractors for $\omega = 1.4$ inside $y_i = (-3, 3)$ window, projected on a) y_0-y_1, b) y_2-y_3 and c) y_4-y_5 plane.

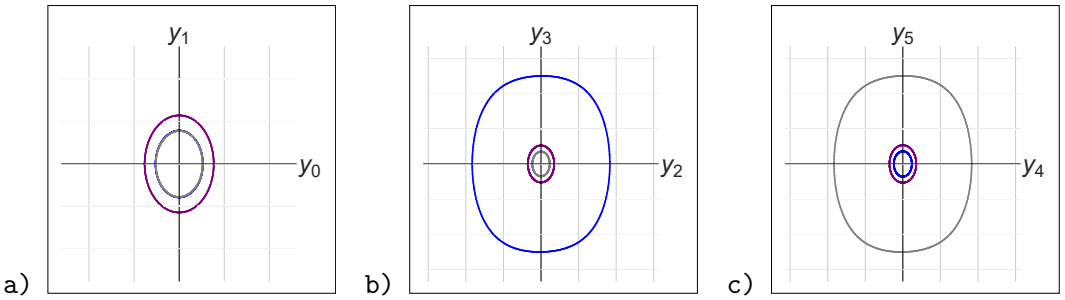

FIGURE 6.27: Trajectories of RHUB system attractors for $\omega = 1.5$ inside $y_i = (-3, 3)$ window, projected on a) y_0-y_1, b) y_2-y_3 and c) y_4-y_5 plane.

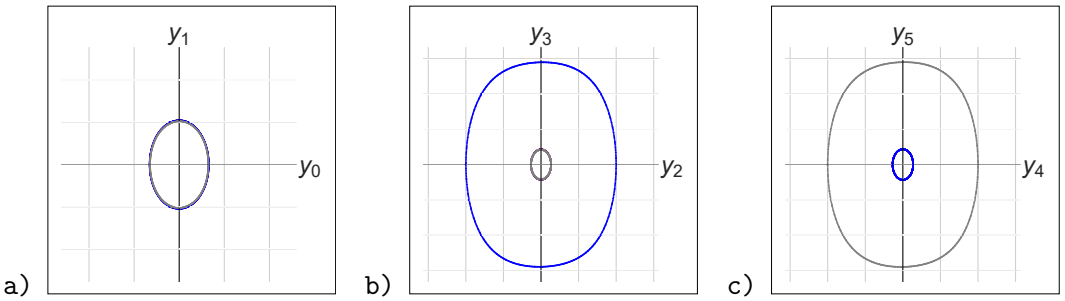

FIGURE 6.28: Trajectories of RHUB system attractors for $\omega = 1.6$ inside $y_i = (-3, 3)$ window, projected on a) y_0-y_1, b) y_2-y_3 and c) y_4-y_5 plane.

robustness relative to each other, we compute full-dimensional basins for the excitation frequencies $\omega = 1.4$, $\omega = 1.5$ and $\omega = 1.6$. Moreover, we can also observe the basin on particular low-dimensional cross-sections. An example of the basin cross-section evolution (for varying excitation frequency ω) and the comparison of NSSCM and GoS computations is shown in Fig. 6.29.

Measures of integrity factor computed with a chessboard distance metric in Table 6.3 show directly why integrity analysis and ultimately erosion profiles are the significant factor in development of an engineering system. Considering that the examined frequencies are from the middle of multi-stable interval (no bifurcations around), it is not intuitive that a sudden change in the attractor robustness can occur. However, it does happen. At the frequencies $\omega = 1.4$ and $\omega = 1.5$, the basins color-coded in blue and gray have larger compact parts than the purple one. With a small frequency change to $\omega = 1.6$, the purple basin suddenly becomes much more compact, compared to other basins and its previous state. An interesting fact is that the blue and gray basins have a matching level of basin compactness, even when the overall robustness changes. Those facts precisely tell how will the system behave in practical applications, which classical stability analysis cannot show.

Considering that the window size is 40 cells per dimension and the largest hypercube that can be accommodated within basin is 8 cells wide, it can be concluded that neither basins is very robust within the $y_i = (-3, 3)$ window. The cause(s) of low basin robustness/compactness (tangled basins, penetration of fractal tongues, etc.) can be examined visually on low-dimensional cross-sections.

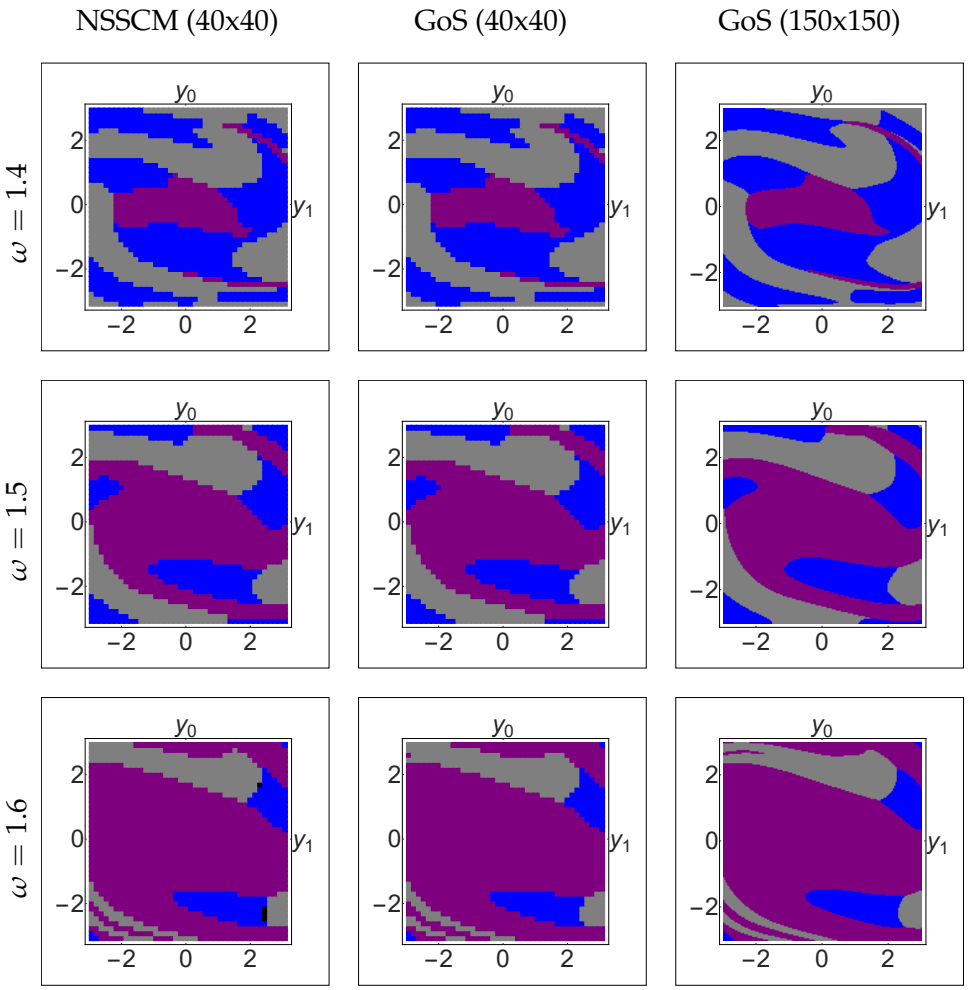

FIGURE 6.29: Comparison of basin computed with NSSCM (full-dimensional basins in resolution of 40 cells per dimension) and with GoS (2D cross-sections in low and high resolution) for $\omega = 1.4$, $\omega = 1.5$, $\omega = 1.6$ on y_0-y_1 cross-section with $y_2 = 0.375$, $y_3 = 0.375$, $y_4 = 0.375$, $y_5 = -0.375$.

TABLE 6.3: Integrity factors (with chess-board distance metric) for RHUB basin within state-space $y_i = (-3, 3)$, expressed as hyper-cube edge length (in number of cells).

attr. color	blue	gray	purple
IF for $\omega = 1.4$	7	7	4
IF for $\omega = 1.5$	7	7	6
IF for $\omega = 1.6$	4	5	8

6.2.3.1 Hub basin structure

The hub is the only segment subjected to the external excitation, thus we are interested in inspecting the basin structure at the cross-section y_0-y_1, where the initial states of the pendula segments are close to zero, namely $y_{2-5} = 0.075$. It is evident from Fig. 6.30a that for the frequency $\omega = 1.4$, the gray basin takes up most of the area, however, it has the compactness very similar to the purple one.

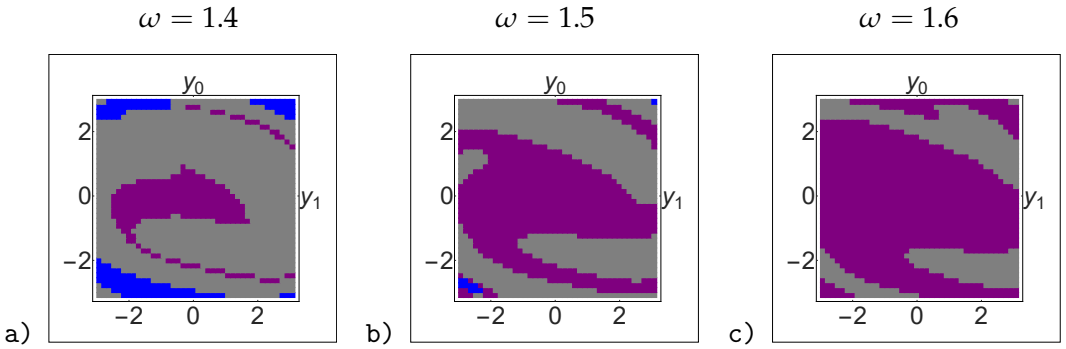

FIGURE 6.30: y_0-y_1 basin cross-section of hub segment with $y_{2-5} = 0.075$ for a) $\omega = 1.4$, b) $\omega = 1.5$ and c) $\omega = 1.6$.

For the frequencies $\omega = 1.5$ and $\omega = 1.6$, Figures 6.30b and 6.30c confirm that the compactness of the purple basin increases. Furthermore, Fig. 6.30 shows that hub motion will in most cases converge to the attractors colored in gray and purple, from which it can be expected that the initial states within the blue basin have significant impact on the pendula.

6.2.3.2 Pendula basin structure for $\omega = 1.4$

As the pendula segments represent very delicate machine elements and are often susceptible to failures (e.g. crack failures of helicopter rotor blades [89, 90]), it is compelling to examine more closely regions of the basin that testify about their steady-state behaviour. Thus, we consider the evolution of pendula basin structure on the respective y_2-y_3 and y_4-y_5 cross-sections with the change of the initial states of the hub segment.

The right pendula, with a shorter length of massless rod ($\delta_{11} = 0.24$) for the frequency $\omega = 1.4$ converges to periodic steady-states colored in purple for the majority of initial states plotted in (entire) Fig. 6.31. However, due to entanglement, its compactness is not significantly higher than the one of other basins.

The left pendula, with longer length of massless rod ($\delta_{11} = 0.26$) for the frequency $\omega = 1.4$ has qualitatively similar basins borders (Fig. 6.32). However, the basin within those borders are different. It means that the pendula will have different steady-state behaviour for the same respective initial states (it will not be symmetric).

6.2.3.3 Pendula basin structure for $\omega = 1.5$

For $\omega = 1.5$, the pendula repeat mirrored steady-state behaviour for same respective initial conditions. We see the increased presence of the basin colored in blue from Figures 6.33 and 6.34, so that all three basins are similarly robust at this frequency.

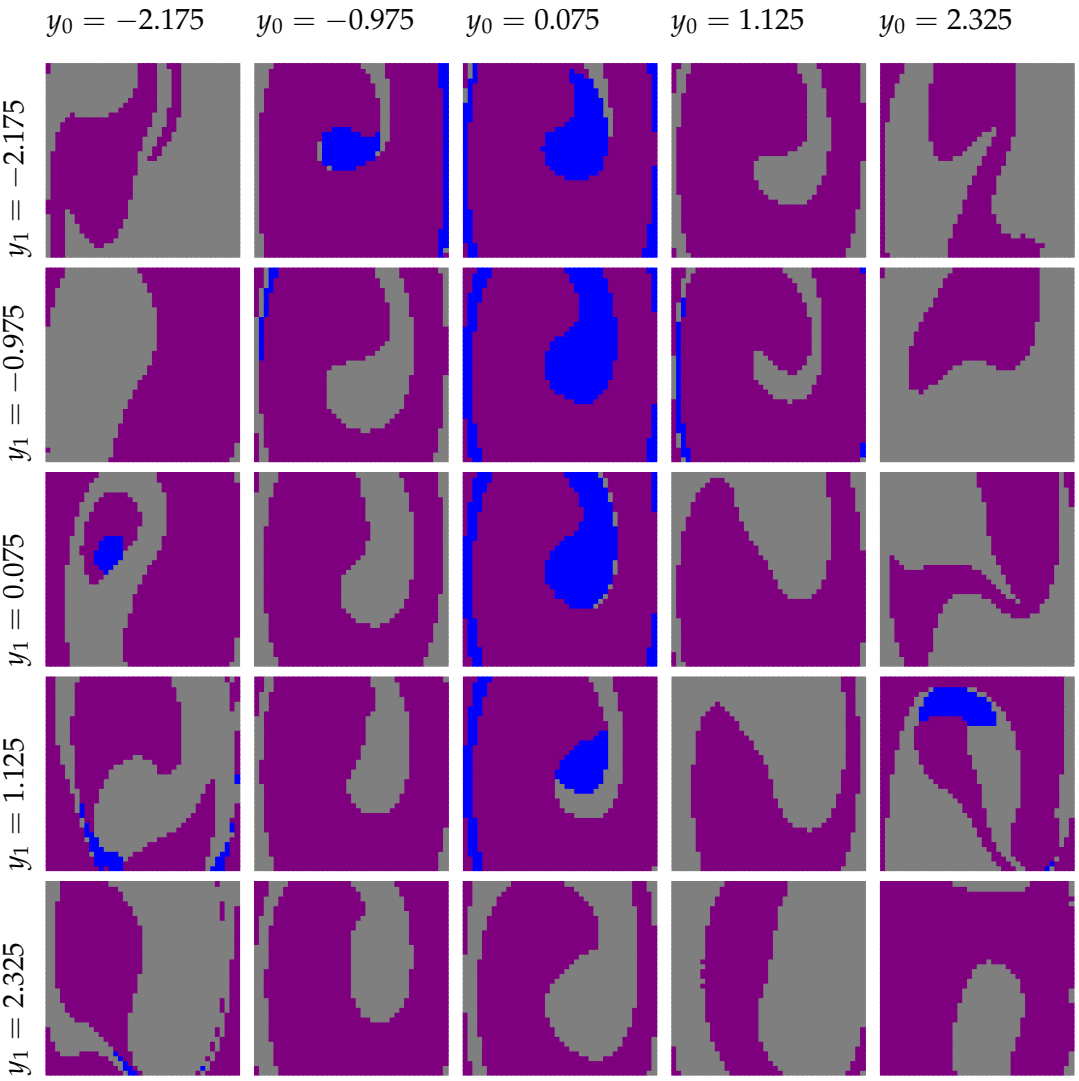

FIGURE 6.31: Basin structure of right pendula segment in y_2-y_3 plane, for $\omega = 1.4$, with the change in initial magnitude of hub initial states y_0 and y_1, while initial states of other pendula are close to zero $y_{4,5} = 0.075$.

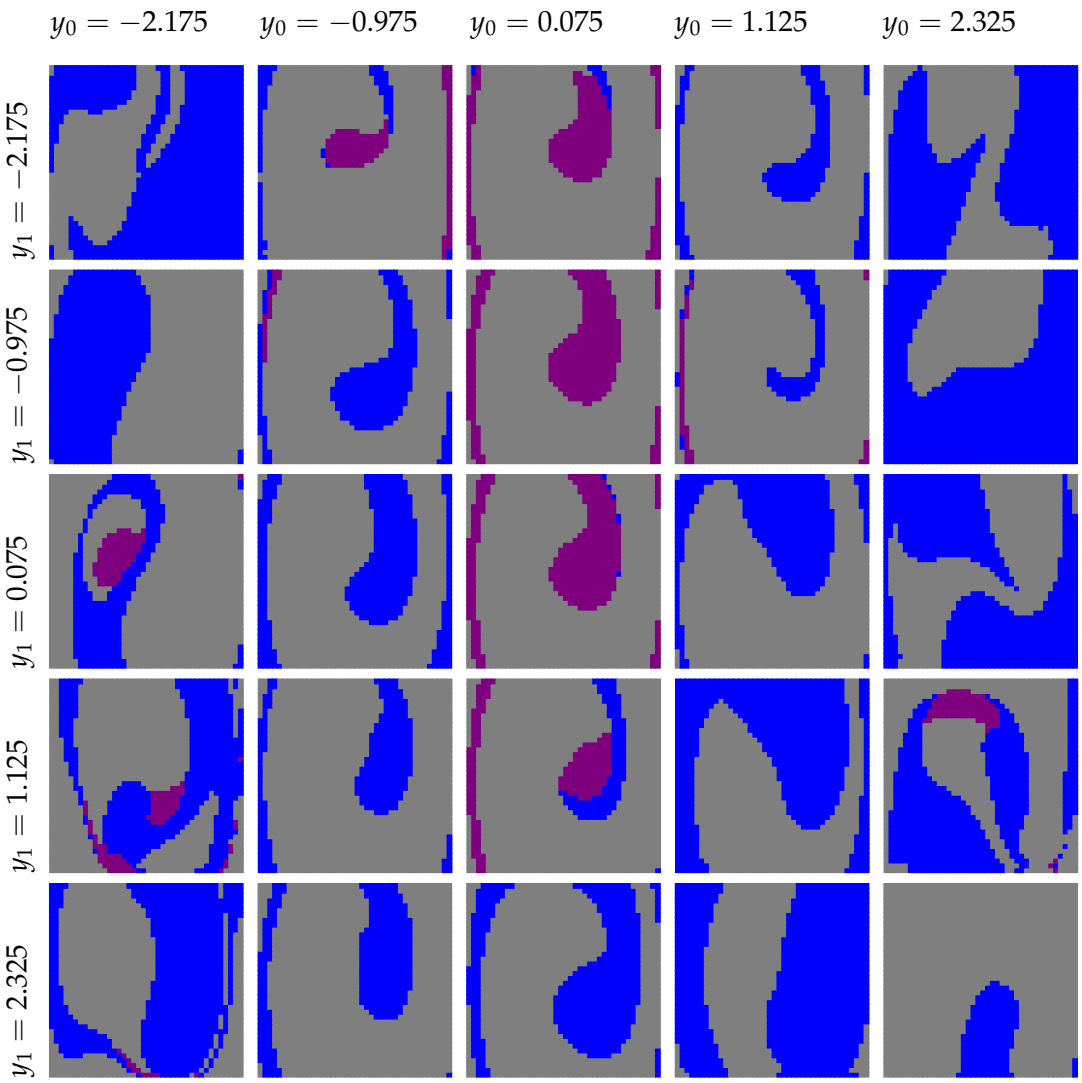

FIGURE 6.32: Basin structure of left pendula segment in y_2-y_3 plane, for $\omega = 1.4$, with the change in initial magnitude of hub initial states y_0 and y_1, while initial states of other pendula are close to zero $y_{2,3} = 0.075$.

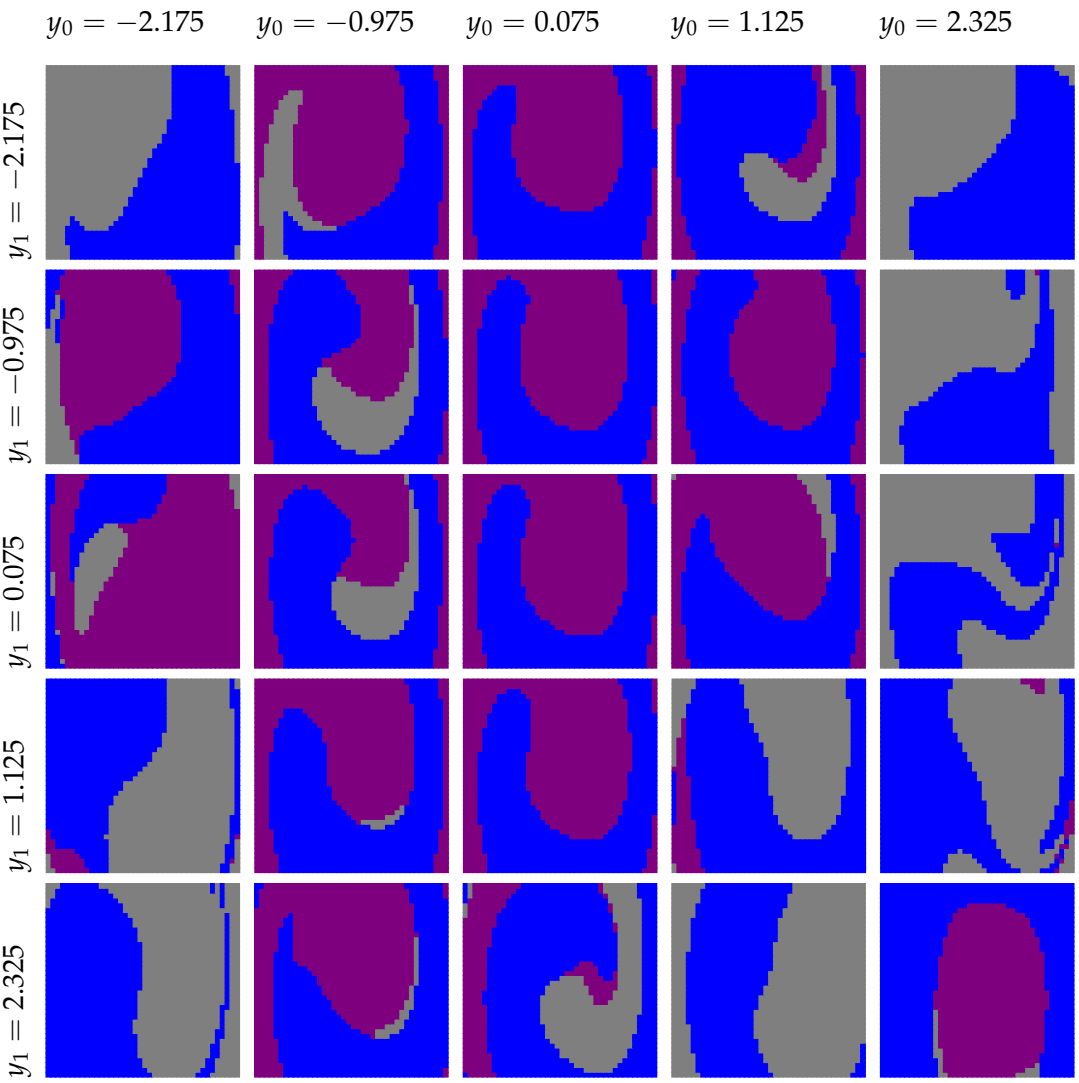

FIGURE 6.33: Basin structure of right pendula segment in y_2-y_3 plane, for $\omega = 1.5$, with the change in initial magnitude of hub initial states y_0 and y_1, while initial states of other pendula are close to zero $y_{4,5} = 0.075$.

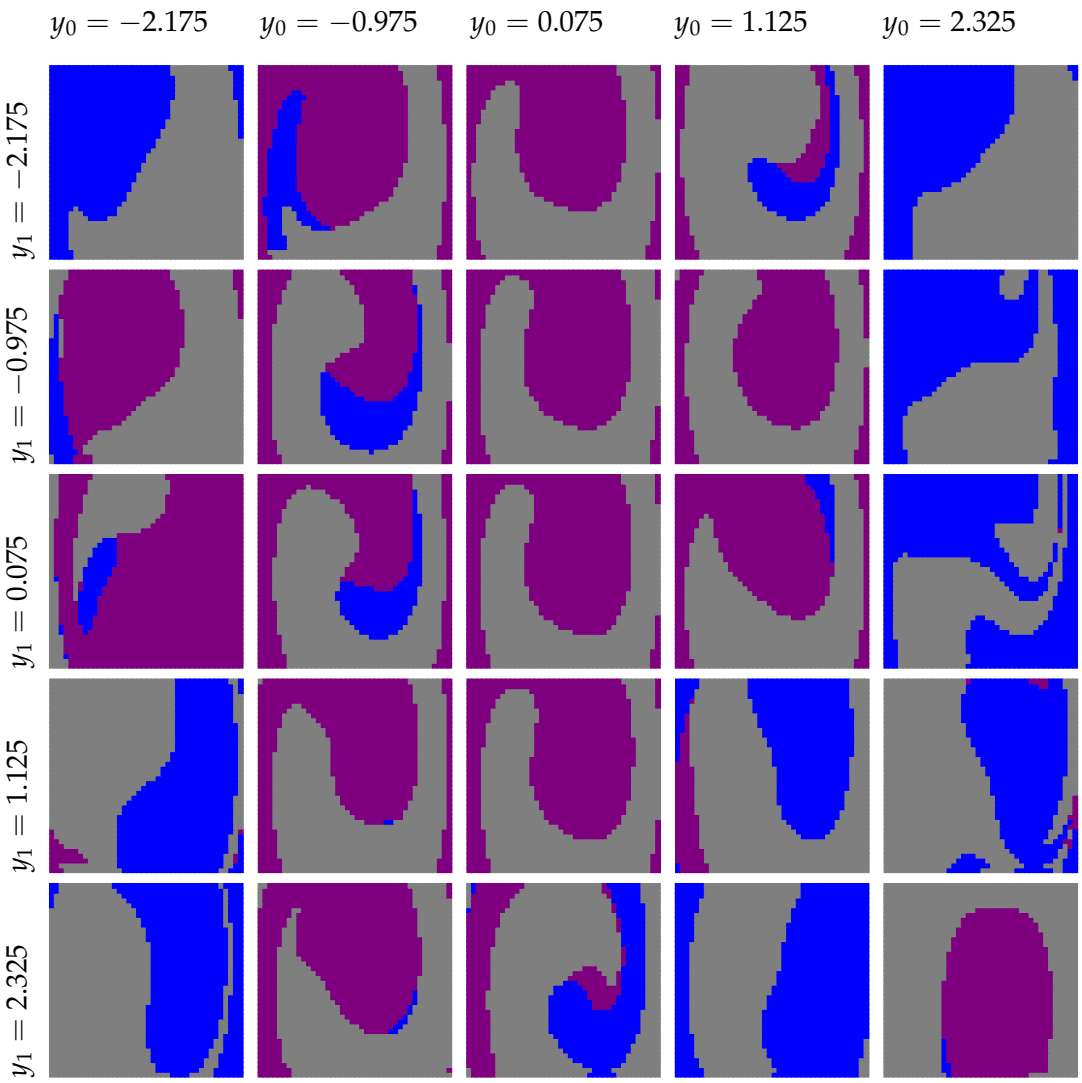

FIGURE 6.34: Basin structure of left pendula segment in y_2-y_3 plane, for $\omega = 1.5$, with the change in initial magnitude of hub initial states y_0 and y_1, while initial states of other pendula are close to zero $y_{2,3} = 0.075$.

6.2.3.4 Pendula basin structure for $\omega = 1.6$

At the frequency $\omega = 1.6$, the purple PR steady-state is the predominant motion for both pendula. Moreover, we see from Figures 6.35 and 6.36 that for this frequency, only the basins of blue and gray attractors are mirrored, confirming the stronger robustness of the purple basin reported in Table 6.3.

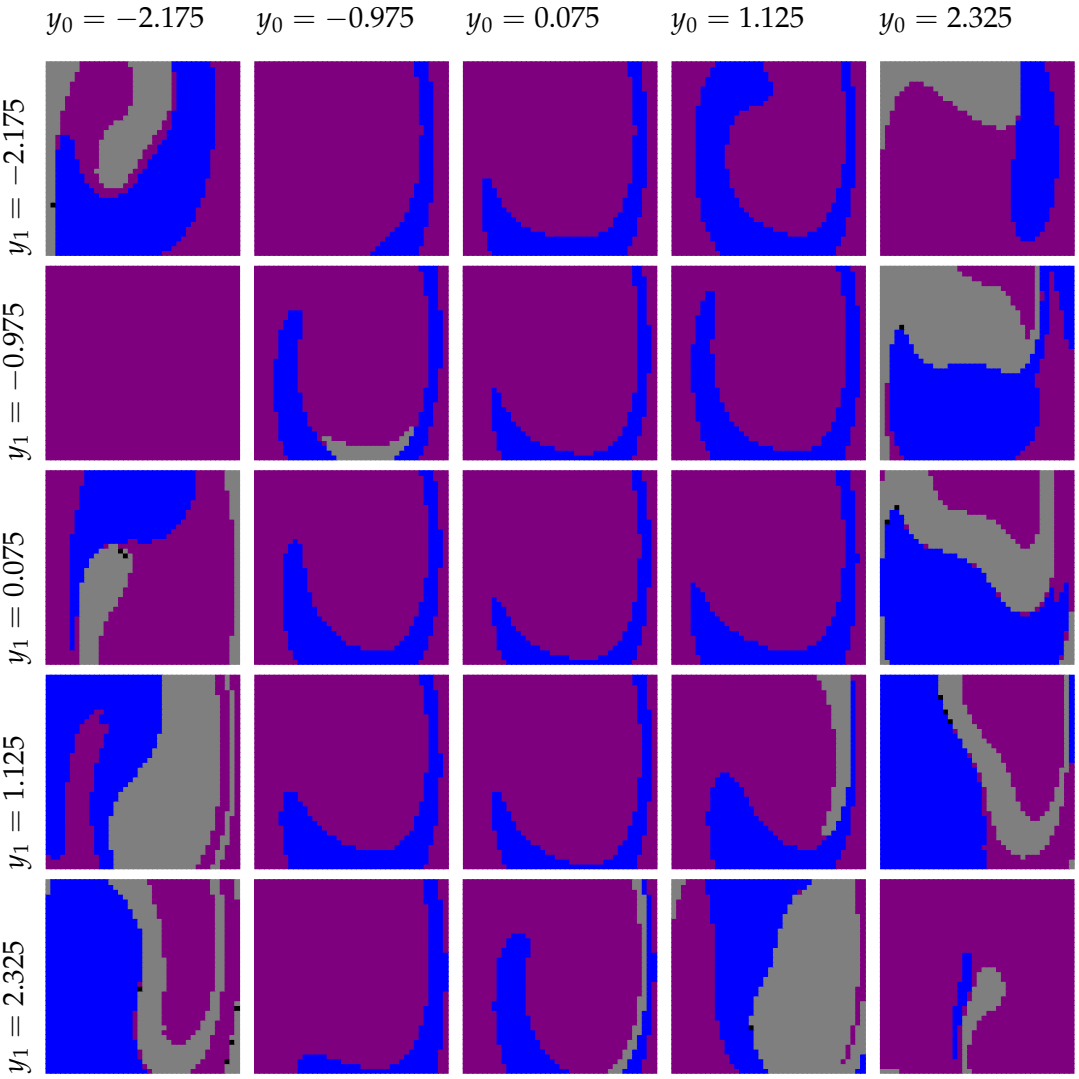

FIGURE 6.35: Basin structure of right pendula segment in y_2-y_3 plane, for $\omega = 1.6$, with the change in initial magnitude of hub initial states y_0 and y_1, while initial states of other pendula are close to zero $y_{4,5} = 0.075$.

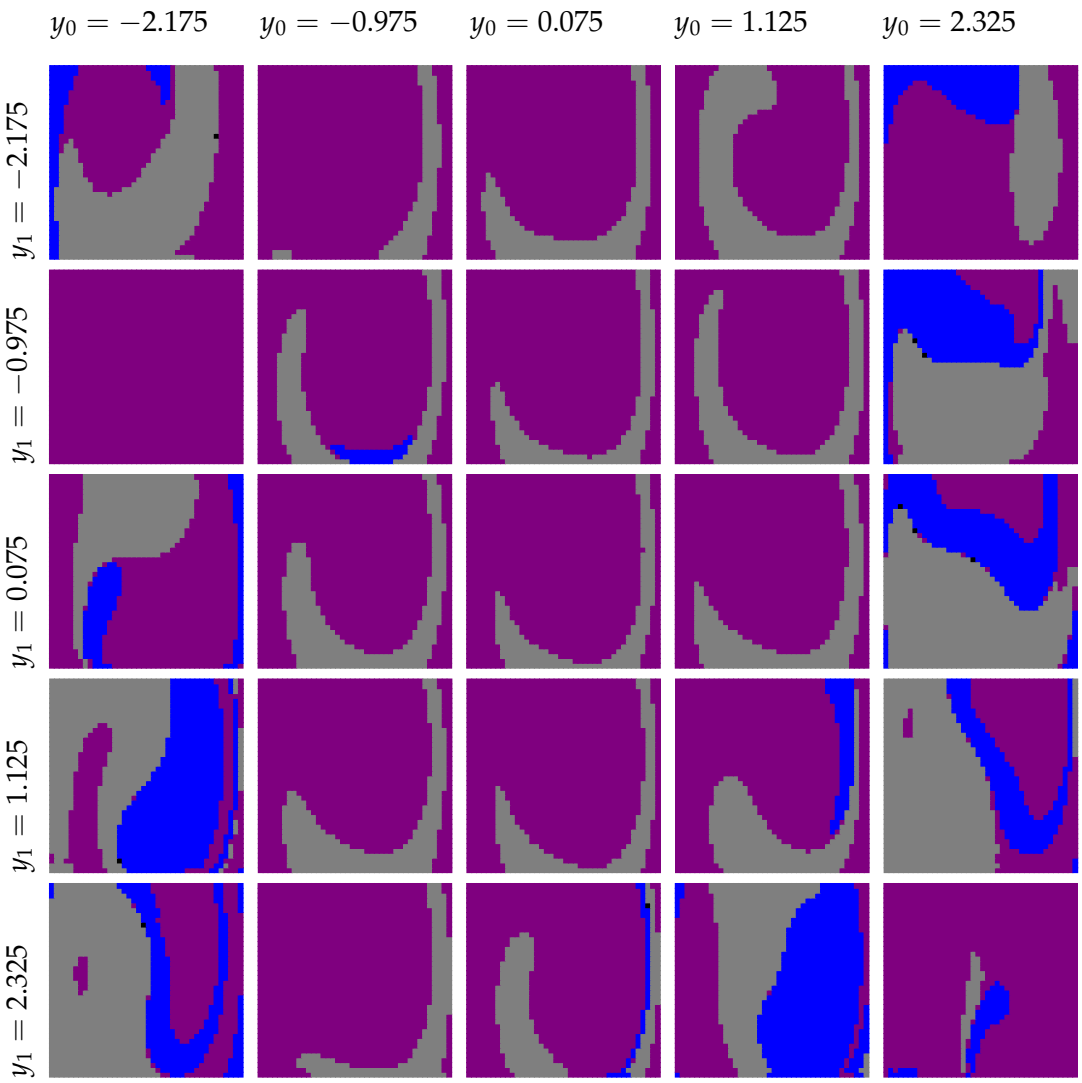

FIGURE 6.36: Basin structure of left pendula segment in y_2-y_3 plane, for $\omega = 1.6$, with the change in initial magnitude of hub initial states y_0 and y_1, while initial states of other pendula are close to zero $y_{2,3} = 0.075$.

Chapter 7

CONCLUSIONS

This thesis is a summary of efforts directed towards the goal to numerically examine basins of attraction for high-dimensional systems governed by first-order differential equations, present in nonlinear dynamics. The purpose of the basin analysis is to determine robustness of attractors and safeness of the dynamical system, which can be examined by using the approach of dynamical integrity. Computing of (dynamical) integrity measures is a very (resource and time) demanding numerical process that requires:

- to integrate a dynamical system for all initial conditions within the considered state-space;
- check to what attractor each initial condition converges;
- and compute distance transform matrix to get the largest compact part of basin.

Moreover, to precisely determine system safeness, this process has to be repeated for a relevant interval of a system parameter(s). The results are in the form of erosion profiles - the plots which show robustness of the attractors related to the system parameter values.

7.1 Development of NSSCM approach

Initial computations had taken unreasonable amount of time on modern personal computers, so the analysis had to be migrated to High-Performance Computing platforms. For the development of software, a small portion of University cluster was available. As the basin computations were still taking too much time with traditional Grid of Starts method, it was necessary to employ a faster approach, the Simple Cell Mapping. However, it is a two-stage method, with each computational phase being compatible with the opposite type of HPC platforms. This problem is addressed by parallelization of only a resource consuming part - the integration of dynamical system (map creation). Parallelization is implemented by assigning one MPI rank to each node of a cluster. Later, with the access to the CINECA cluster MARCONI, which had multi-core nodes, the parallelization is hybridized by multi-threading with OpenMP inside each MPI rank. The post-processing of the map and search for attractors and the corresponding basins is left sequential, as it is not a parallelizable algorithm, and more importantly - even sequentially is drastically faster than the map creation, which justifies this decision.

As SCM is an approximate method that originally creates a map during transient motion (to reduce computing costs), several problems occur. The issue of transient trajectories that escape a predefined state-space is solved by prolonging integration time to allow those trajectories to reenter the window. It happens in majority of

examined cases that trajectories reenter much before than they arrive on the neighbourhood of attractor. In other words, the integration time is longer, but still much shorter than required with GoS. Prolonged integration time introduced with NSSCM gives a possibility to compute more accurately the basins of externally excited systems.

The disconnected periodic orbit problem is a propagation of an approximation error or a discretization issue arising from the position of a stroboscopic Poincaré section. It causes the SCM to report multiple, separate periodic orbits that should represent a single physical attractor. The problem is resolved by introducing additional post-processing. It connects disconnected periodic orbits by following real (unapproximated) trajectories emerging from already discovered periodic cells and confront them with the encountered ones.

The collection of aforementioned improvements and the parallelization of the original SCM is named Not So Simple Cell Mapping, as the process of computing the attractor and basins is unchanged. The development of NSSCM constitutes the main goal of this thesis, which allows the user to accurately compute basins of attraction for driven nonlinear systems in 6D state-space. Without NSSCM improvements, the basins obtained with the original SCM would be practically indistinguishable due to a large number of disconnected attractors and incorrect image cells computed at an early transient phase.

7.2 Chessboard integrity factor

Once the basin computation approach is established, the attention was moved to a dynamical integrity analysis. The measure of an attractor robustness is defined as a largest geometrical object that can be found inside each basin. The intuitive approach is to search for a hyper-sphere, but this requires the calculation of a distance transform matrix with the Euclidean distances from each cell to the nearest basin boundary. In our work, a less compute intensive Chebyshev (chessboard) metric distance is used, so the integrity factor is defined with a hyper-cube instead of a sphere. A further decrease in computational time of the distance matrix transform is achieved by parallelization with OpenMP.

However, without the access to computing resources to compute erosion profiles, we are constrained to compute integrity factor for only few values of the relevant system parameter (excitation frequency). Nevertheless, this way did allow us to draw important conclusions about the system dynamics than the classical stability analysis would do.

7.3 Viability of NSSCM approach

The efficiency of NSSCM and the value of basins and integrity analysis is illustrated on the harmonically excited dynamical system composed of three coupled Duffing-like oscillators.

The reliability of the NSSCM is demonstrated by comparing the accuracy of basins computed via SCM, NSSCM and GoS in various settings. For shorter integrations, the overall basin accuracy obtained via SCM/NSSCM was low. The integration time of 15 excitation periods gave a satisfactory accuracy with NSSCM, which is significantly faster than GoS computations that required at least 50 periods

(for this dynamical system). Moreover, the SCM computed a high number of disconnected attractors making the basins indistinguishable, which was resolved with CPA of the NSSCM.

With the 8192 parallel tasks the integrations lasted about 5 hours with NSSCM. It clearly shows how much full-dimensional basin computation is actually demanding, and why it is crucial to avoid GoS for 6D system. On the other hand, the map post-processing (basin detection and CPA), implemented sequentially, lasted less than 10 minutes, proving not to be a resource and time consuming process. Computation time of the distance transform matrices could not have been anticipated, because it directly depends on the robustness of attractors. Namely, the larger the compact part of basins, the longer iterations are required to determine the integrity factor. Considering that the integrity analysis lasted more than one hour with 256 parallel tasks, it is evident that the choice of the chessboard distance instead of the Euclidean metric is justified.

The complex structure of basins is the main cause for the necessity to determine them throughout whole the state-space window, not just on an arbitrary cross-section. It has been shown that the basin can be large and compact on some cross-sections, while being noticeably fractalized in other regions of the state-space. Moreover, it has also been demonstrated that for a small coordinate change in some directions, the basins structure may change drastically. Altogether, it confirms that the computation of full-dimensional basins is unquestionably required to determine global behaviour.

As it is not possible to visually inspect basins in a 6D state-space, the values of the integrity factor were used to illustrate the robustness of attractors relative to each other.

7.4 Global dynamics of sympodial tree model with first level branches

A sympodial tree model has been analysed for the parameter values that correlate with the properties of a real tree. It was very resource consuming to determine steady-states precisely and the basins of attraction due to the long transients. With the NSSCM, an acceptable accuracy of basins was achieved for the integration time of 24 excitation periods. It is significantly faster than the GoS method, which required about 250 periods for accurate results.

The integrity analysis showed that a periodic motion with the period one is a predominant steady-state behaviour of this dynamical system. It is accompanied with either PR3 or QP attractor within multi-stable intervals. The influence of initial conditions on global behaviour that belong to secondary attractors is localized mostly near origin. Moreover, these claims are confirmed by the analysis of low-dimensional cross-sections and with low basin compactness in comparison to a much higher value of the integrity factor for the basin of a PR1 attractor.

7.5 Global dynamics of rotating hub model with attached pendulums

An important conclusion about the system composed of a rotating hub and two attached pendula with unequal lengths, is that some basin can be more robust than the others, but not necessarily in the regions where the system is exploited in practice.

Dynamics of this system is periodic in any combination of the parameter values that were examined and there were three competing PR1 attractors in multi-stable regions. The integrity analysis showed lesser compactness of one particular attractor, however, its basins were equally distributed on cross-section that were considered valuable for practical applications.

Moreover, it has been shown that the steady-states of pendula are mirrored, meaning that it will not have the same global behaviour for the same initial states. In other words, the basin structure of both pendula is qualitatively the same, but the respective areas are covered with the basins of another attractors.

7.6 Closure and future work

Hereby the thesis is concluded, with the strong belief that our work has made some progress toward more a efficient global analysis for examining safeness of nonlinear systems. However, several possibilities remain open for future efforts

- an efficient parallel algorithm for a map analysis, which would allow high scalability to large clusters, leading to a better accuracy and precision of basins;
- an efficient parallel algorithm for distance transform matrix computations, regardless the distance metric;
- computation of basin erosion profiles, which can provide better understanding of high-dimensional system safeness in practical applications.

Bibliography

[1] P. Brzeski, P. Belardinelli, S. Lenci, and P. Perlikowski. Revealing compactness of basins of attraction of multi-DoF dynamical systems. *Mechanical Systems and Signal Processing*, 111:348–361, 2018. doi: 10.1016/j.ymssp.2018.04.005.

[2] S. Lenci and G. Rega. Load carrying capacity of systems within a global safety perspective. part ii. attractor/basin integrity under dynamic excitations. *International Journal of Non-Linear Mechanics*, 46(9):1240 – 1251, 2011. doi: 10.1016/j.ijnonlinmec.2011.05.021.

[3] S. Lenci, G. Rega, and L. Ruzziconi. The dynamical integrity concept for interpreting/ predicting experimental behaviour: from macro- to nano-mechanics. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 371(1993), 2013. doi: 10.1098/rsta.2012.0423.

[4] G. Rega and S. Lenci. Identifying, evaluating, and controlling dynamical integrity measures in non-linear mechanical oscillators. *Nonlinear Analysis: Theory, Methods & Applications*, 63(5):902 – 914, 2005. doi: 10.1016/j.na.2005.01.084.

[5] M.S. Soliman and J.M.T. Thompson. Integrity measures quantifying the erosion of smooth and fractal basins of attraction. *Journal of Sound and Vibration*, 135(3): 453 – 475, 1989. doi: 10.1016/0022-460X(89)90699-8.

[6] R.C. Hilborn. *Chaos and nonlinear dynamics: an introduction for scientists and engineers*. Oxford University Press, 2nd edition, 2000. ISBN 9780198507239. doi: 10.1093/acprof:oso/9780198507239.001.0001.

[7] S.H. Strogatz. *Nonlinear dynamics and Chaos: with applications in physics, biology, chemistry, and engineering*. Addison-Wesley Pub, 1994. ISBN 0201543443.

[8] S. Wiggins. *Introduction to Applied Nonlinear Dynamical Systems and Chaos*. Texts in Applied Mathematics. Springer, New York, 2003. ISBN 9780387001777.

[9] H.E. Nusse, B.R. Hunt, E.J. Kostelich, and J.A. Yorke. *Dynamics: numerical explorations*. Applied mathematical sciences. Springer, New York, 2nd, rev. and enl. ed edition, 1998. ISBN 0387982647.

[10] P. Belardinelli and S. Lenci. A first parallel programming approach in basins of attraction computation. *International Journal of Non-Linear Mechanics*, 80:76 – 81, 2016. ISSN 0020-7462. doi: 10.1016/j.ijnonlinmec.2015.10.016.

[11] C.S. Hsu. *Cell-to-Cell Mapping: A Method of Global Analysis for Nonlinear Systems*. Springer-Verlag, New York, 1987. ISBN 978-1-4757-3892-6.

[12] J.Q. Sun, F.R. Xiong, O. Schütze, and C. Hernández. *Cell Mapping Methods: Algorithmic Approaches and Applications*. Springer, Singapore, 2018. ISBN 9789811304576.

[13] J. Aguirre, R.L. Viana, and M.A.F. Sanjuán. Fractal structures in nonlinear dynamics. *Reviews of Modern Physics*, 81:333–386, 2009. doi: 10.1103/RevModPhys.81.333.

[14] M.M. Deza and E. Deza. *Encyclopedia of Distances*, pages 1–583. Springer, Berlin, Heidelberg, 2009.

[15] Andonovski N., Lenci S., Moglie F. *Introduction to Scientific Computing Technologies for Global Analysis of Multidimensional Nonlinear Dynamical Systems*, pages 1–43. Mechanisms and Machine Science. Springer, Cham, 2019. ISBN 978-3-030-13316-0. doi: 10.1007/978-3-030-13317-7_1.

[16] N. Andonovski and S. Lenci. Six-dimensional basins of attraction computation on small clusters with semi-parallelized scm method. *International Journal of Dynamics and Control*, 2019. doi: 10.1007/s40435-019-00557-2.

[17] N. Andonovski, S. Lenci, and I. Kovacic. Basins of attraction for higher-dimensional nonlinear dynamical systems: Preliminary results on the case study of a sympodial tree. In *IUTAM Symposium on Exploiting Nonlinear Dynamics for Engineering Systems*, pages 27–36. Springer International Publishing, 2019. doi: 10.1007/978-3-030-23692-2_3.

[18] Y. Aruga, T. Endo, and A. Hasegawa. Bifurcation of modes in three-coupled oscillators with the increase of nonlinearity. In *2002 IEEE International Symposium on Circuits and Systems. Proceedings (Cat. No.02CH37353)*, volume 5, pages V–V. IEEE, 2002. doi: 10.1109/ISCAS.2002.1010702.

[19] I. Kovacic, M. Zukovic, and D. Radomirovic. Sympodial tree-like structures: from small to large-amplitude vibrations. *Bioinspiration & Biomimetics*, 13(2): 026002, 2018. doi: 10.1088/1748-3190/aa9d1c.

[20] J. Warminski, Z. Szmit, and J. Latalski. Nonlinear dynamics and synchronisation of pendula attached to a rotating hub. *The European Physical Journal Special Topics*, 223(4):827–847, 2014. doi: 10.1140/epjst/e2014-02143-9.

[21] E. Aubanel. *Elements of Parallel Computing*. Chapman and Hall CRC, 2016. ISBN 9781498727891.

[22] D. Padua, editor. *Encyclopedia of Parallel Computing*. Springer US, 2011. ISBN 978-0-387-09765-7.

[23] A. Elahi. *Computer Systems: Digital Design, Fundamentals of Computer Architecture and Assembly Language*. Springer, Cham, 2018. ISBN 978-3-319-66774-4.

[24] D. Comer. *Essentials of Computer Architecture, Second Edition*. Chapman and Hall CRC, 2nd edition, 2017. ISBN 1138626597, 9781138626591.

[25] J.L. Hennessy, D.A. Patterson, and K. Asanovi. *Computer architecture: A quantitative approach*. Morgan Kaufmann/Elsevier, Amsterdam; Boston, 5th edition, 2012. ISBN 9780123838735.

[26] S. Chakrabarti, J. Demmel, and K. Yelick. Modeling the benefits of mixed data and task parallelism. In *Proceedings of the Seventh Annual ACM Symposium on Parallel Algorithms and Architectures*, SPAA '95, pages 74–83, New York, NY, USA, 1995. ACM. ISBN 0-89791-717-0. doi: 10.1145/215399.215423.

[27] The Top500 List of June 2019, . URL www.top500.org/list/2019/06/. [Online; accessed 19-November-2018].

[28] P. Czarnul. *Parallel Programming for Modern High Performance Computing Systems*. CRC Press, Taylor and Francis Group, 2018. ISBN 9781138305953.

[29] T. Rauber and G. Rünger. *Parallel Programming for Multicore and Cluster Systems, 2nd Edition*. Springer-Verlag, Berlin, Heidelberg, 2013. ISBN 978-3-642-37800-3.

[30] Cineca, Italian super-computing centre (consortium). URL www.cineca.it/en/. [Online; accessed 25-August-2019].

[31] HPC at CINECA: User Documentation, UG3.1: MARCONI User Guide. URL wiki.u-gov.it/confluence/display/SCAIUS/UG3.1%3A+MARCONI+UserGuide. [Online; accessed 01-June-2019].

[32] I. Foster. *Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1995. ISBN 0201575949.

[33] A. Grama. *Introduction to parallel computing*. Addison-Wesley, Harlow, England; New York, 2nd edition, 2003. ISBN 0201648652.

[34] Y. Solihin. *Fundamentals of parallel multicore architecture*. Chapman-Hall, CRC, 1st edition, 2015. ISBN 9781498753418.

[35] W. Stallings and G. Paul. *Operating systems: internals and design principles*. Pearson, Boston, [Mass.]; London, 7th edition, 2012. ISBN 9780273751502.

[36] S. Pllana and F. Xhafa, editors. *Programming Multi-core and Many-core Computing Systems*. Wiley Publishing, 1st edition, 2014. ISBN 0470936908, 9780470936900.

[37] The OpenMP API specification for parallel programming, . URL www.openmp.org. [Online; accessed 19-November-2018].

[38] Message Passing Interface (MPI) Forum, . URL www.mpi-forum.org. [Online; accessed 19-November-2018].

[39] CUDA zone, . URL developer.nvidia.com/cuda-zone. [Online; accessed 19-November-2018].

[40] The OpenCL™ specification, . URL www.khronos.org/registry/OpenCL/specs/2.2/html/OpenCL_API.html. [Online; accessed 19-November-2018].

[41] Wolfram Mathematica. URL www.wolfram.com/mathematica/. [Online; accessed 15-September-2019].

[42] MathWorks MatLab. URL mathworks.com/products/matlab.html. [Online; accessed 15-September-2019].

[43] ParaView. URL www.paraview.org. [Online; accessed 15-September-2019].

[44] P.F. Byrd and M.D. Friedman. *Handbook of Elliptic Integrals for Engineers and Scientists*. Springer Berlin Heidelberg, 1971. doi: 10.1007/978-3-642-65138-0.

[45] P.A. Clarkson. Painlevé equations—nonlinear special functions. *Journal of Computational and Applied Mathematics*, 153(1):127 – 140, 2003. doi: 10.1016/s0377-0427(02)00589-7.

[46] M.K. Jain, S.R.K. Iyengar, and R.K. Jain. *Numerical Methods for Scientific and Engineering Computation*. Wiley Eastern Ltd., New Delhi etc., 1986. ISBN 0?85226?434?8.

[47] A.H. Nayfeh and B. Balachandran. *Applied Nonlinear Dynamics: Analytical, Computational, and Experimental Methods*. Wiley Series in Nonlinear Science. Wiley-VCH, Weinheim, 1995.

[48] A. Medio and M. Lines. *Nonlinear Dynamics: A Primer*. Cambridge University Press, 2001. doi: 10.1017/CBO9780511754050.

[49] P. M. Battelino, C. Grebogi, E. Ott, J. A. Yorke, and E. D. Yorke. Multiple coexisting attractors, basin boundaries and basic sets. *Physica D: Nonlinear Phenomena*, 32(2):296–305, 1988. doi: 10.1016/0167-2789(88)90057-7.

[50] A. Daza, A. Wagemakers, B. Georgeot, D. Guéry-Odelin, and M. Sanjuán. Basin entropy: a new tool to analyze uncertainty in dynamical systems. *Scientific Reports*, 6:3146, 2016. doi: 10.1038/srep31416.

[51] A. Daza, A. Wagemakers, B. Georgeot, D. Guéry-Odelin, and M. Sanjuán. Basin Entropy, a Measure of Final State Unpredictability and Its Application to the Chaotic Scattering of Cold Atoms, In *Chaotic, Fractional, and Complex Dynamics: New Insights and Perspectives. Understanding Complex Systems*, pages 9–34, Springer, Cham 2018. doi: 10.1007/978-3-319-68109-2_2.

[52] P. Ji and J. Kurths. Basin stability in complex oscillator networks. In *Nonlinear Dynamics of Electronic Systems*, pages 211–218. Springer International Publishing, 2014. doi: 10.1007/978-3-319-08672-9_26.

[53] P. Brzeski, M. Lazarek, T. Kapitaniak, J. Kurths, and P. Perlikowski. Basin stability approach for quantifying responses of multistable systems with parameters mismatch. *Meccanica*, 51(11):2713–2726, 2016. doi: 10.1007/s11012-016-0534-8.

[54] D. Dudkowski, K. Czołczyński, and T. Kapitaniak. Multistability and basin stability in coupled pendulum clocks. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 29(10):103140, 2019. doi: 10.1063/1.5118726.

[55] W. Szemplinska-Stupnicka and H. Troger. *Engineering Applications of Dynamics of Chaos*. CISM International Centre for Mechanical Sciences. Springer, Vienna, 1991. ISBN 9783709126103.

[56] B.H. Tongue and K. Gu. Interpolated cell mapping of dynamical systems. *Journal of Applied Mechanics*, 55(2):461–466, 1988. doi: 10.1115/1.3173700.

[57] B.H. Tongue and K. Gu. A higher order method of interpolated cell mapping. *Journal of Sound and Vibration*, 125(1):169 – 179, 1988. doi: 10.1016/0022-460x(88)90424-5.

[58] J.A.W. Spek, van der, D.H. Campen, van, and A. Kraker, de. Cell mapping for multi degrees of freedom systems. AMD, pages 151–159. ASME, 1994. ISBN 0-7918-1441-6.

[59] Z.M. Ge and S.C. Lee. A modified interpolated cell mapping method. *Journal of Sound and Vibration*, 199(2):189 – 206, 1997.

[60] B.H. Tongue and K. Gu. A higher order method of interpolated cell mapping. *Journal of Sound and Vibration*, 125(1):169 – 179, 1988.

[61] J.A.W. Spek, van der. *Cell mapping methods: modifications and extensions*. PhD thesis, Department of Mechanical Engineering, 1994.

[62] R. P. Eason and A. J. Dick. A parallelized multi-degrees-of-freedom cell mapping method. *Nonlinear Dynamics*, 77(3):467–479, 2014. doi: 10.1007/s11071-014-1310-8.

[63] G. Gyebrószki and G. Csernák. Clustered simple cell mapping: An extension to the simple cell mapping method. *Communications in Nonlinear Science and Numerical Simulation*, 42:607 – 622, 2017.

[64] P. Belardinelli and S. Lenci. An efficient parallel implementation of cell mapping methods for mdof systems. *Nonlinear Dynamics*, 86(4):2279–2290, 2016. doi: 10.1007/s11071-016-2849-3.

[65] P. Belardinelli and S. Lenci. Improving the global analysis of mechanical systems via parallel computation of basins of attraction. *Procedia IUTAM*, 22:192 – 199, 2017. doi: 10.1016/j.piutam.2017.08.028.

[66] P. Belardinelli, S. Lenci, and G. Rega. Seamless variation of isometric and anisometric dynamical integrity measures in basins,s erosion. *Communications in Nonlinear Science and Numerical Simulation*, 56:499 – 507, 2018. doi: 10.1016/j.cnsns.2017.08.030.

[67] Intel® Hyper-Threading Technology. URL www.intel.com/content/www/us/en/architecture-and-technology/hyper-threading/hyper-threading-technology.html. [Online; accessed 26-August-2019].

[68] J.H. Reif. Depth-first search is inherently sequential. *Information Processing Letters*, 20(5):229 – 234, 1985. doi: 10.1016/0020-0190(85)90024-9.

[69] J. Fernández, O. Schütze, C. Hernández, J.Q. Sun, and F.R. Xiong. Parallel simple cell mapping for multi-objective optimization. *Engineering Optimization*, 48 (11):1845–1868, 2016.

[70] Rega G. (eds) Lenci S. *Global Nonlinear Dynamics for Engineering Design and System Safety*, volume 588 of *CISM International Centre for Mechanical Sciences*. Springer International Publishing, 2019.

[71] S. Lenci and G. Rega. Optimal control of nonregular dynamics in a duffing oscillator. *Nonlinear Dynamics*, 33:71–86, 2003. doi: 10.1023/a:1025509014101.

[72] A.N. Lansbury, J.M.T. Thompson, and H.B. Stewart. Basin erosion in the twin-well duffing oscillator: two distinct bifurcation scenarios. *International Journal of Bifurcation and Chaos*, 02(03):505–532, 1992. doi: 10.1142/s0218127492000677.

[73] S. Lenci and G. Rega. Overturning thresholds of a rocking block subjected to harmonic excitation: Computer simulations and analytical treatment. volume 254. ASME, 2003. doi: 10.1115/IMECE2003-55600.

[74] I. Kovacic and M.J. Brennan, editors. *The Duffing Equation: Nonlinear Oscillators and their Behaviour*. John Wiley & Sons, Ltd, 2011. doi: 10.1002/9780470977859.

[75] H.J. Korsch, H.J. Jodl, and T. Hartmann. *The Duffing Oscillator*, pages 157–184. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. doi: 10.1007/978-3-540-74867-0.

[76] J. Sunday. The duffing oscillator: Applications and computational simulations. *Asian Research Journal of Mathematics*, 2:1–13, 2017. doi: 10.9734/arjom/2017/31199.

[77] L.I. Manevitch, A. Kovaleva, V. Smirnov, and Y. Starosvetsky. Duffing oscillators. In *Foundations of Engineering Mechanics*, pages 155–186. Springer Singapore, 2017. doi: 10.1007/978-981-10-4666-7_6.

[78] A.H. Nayfeh. Forced oscillations of the duffing oscillator. In *The Method of Normal Forms*, pages 161–173. Wiley-VCH Verlag GmbH & Co. KGaA, 2011. doi: 10.1002/9783527635801.ch5.

[79] B. Theckes, E. de Langre, and X. Boutillon. Damping by branching: a bioinspiration from trees. *Bioinspiration & Biomimetics*, 6(4):046010, 2011. doi: 10.1088/1748-3182/6/4/046010.

[80] T. Kerzenmacher and B. Gardiner. A mathematical model to describe the dynamic response of a spruce tree to the wind. *Trees*, 12(6):385–394, 1998. doi: 10.1007/s004680050165.

[81] K.D. Murphy and M. Rudnicki. A physics-based link model for tree vibrations. *American Journal of Botany*, 99(12):1918–1929, 2012. doi: 10.3732/ajb.1200141.

[82] J.R. Moore and D.A. Maguire. Simulating the dynamic behavior of Douglas-fir trees under applied loads by the finite element method. *Tree Physiology*, 28(1):75–83, 2008. doi: 10.1093/treephys/28.1.75.

[83] D. Sellier and T. Fourcaud. Crown structure and wood properties: Influence on tree sway and response to high winds. *American Journal of Botany*, 96(5):885–896, 2009. doi: 10.3732/ajb.0800226.

[84] H.C. Spatz and B. Theckes. Oscillation damping in trees. *Plant Science*, 207:66 – 71, 2013. ISSN 0168-9452. doi: 10.1016/j.plantsci.2013.02.015.

[85] Mathieu Rodriguez, Emmanuel de Langre, and Bruno Moulia. A scaling law for the effects of architecture and allometry on tree vibration modes suggests a biological tuning to modal compartmentalization. *American Journal of Botany*, 95(12):1523–1537, 2008. doi: 10.3732/ajb.0800161.

[86] T.A. McMahon and R.E. Kronauer. Tree structures: Deducing the principle of mechanical design. *Journal of Theoretical Biology*, 59(2):443 – 466, 1976. ISSN 0022-5193. doi: 10.1016/0022-5193(76)90182-X.

[87] E. Ott. *Chaos in Dynamical Systems*. Cambridge University Press, 2 edition, 2002. doi: 10.1017/cbo9780511803260.

[88] C. Grebogi, E. Ott, and J.A. Yorke. Crises, sudden changes in chaotic attractors, and transient chaos. *Physica D: Nonlinear Phenomena*, 7(1):181 – 200, 1983. doi: 10.1016/0167-2789(83)90126-4.

[89] M. Amura, L. Aiello, M. Colavita, F. De Paolis, and M. Bernabei. Failure of a helicopter main rotor blade. *Procedia Materials Science*, 3:726 – 731, 2014. doi: 10.1016/j.mspro.2014.06.119.

[90] R. Kieselbach and G. Soyka. Failure of a helicopter rotor. *Technology, Law and Insurance*, 5(3-4):141–146, 2000. doi: 10.1080/135993700750364369.