Published in: Communications in Nonlinear Science and Numerical Simulation, 42, pp. 607-622 (2017)

Clustered Simple Cell Mapping: an extension to the Simple Cell Mapping method

Gergely Gyebrószki[a,1,*], Gábor Csernák[2]

[a] *Budapest, Műegyetem rkp. 5, H-1111, Hungary*

Abstract

When a dynamical system has a complex structure of fixed points, periodic cycles or even chaotic attractors, Cell Mapping methods are excellent tools to discover and thoroughly analyse all features in the state space. These methods discretize a region of the state space into cells and examine the dynamics in the cell state space. By determining one or more image cells for each cell, the global behaviour within the region can be quickly determined. In the simplest case – Simple Cell Mapping (SCM) method – only one image corresponds to a cell and usually a rectangular grid of cells is used. In typical applications the grid of cells is refined at specific locations.

This paper, however, introduces a different approach, which is useful to expand the analysed state space region to include all features which properly characterize the global dynamics of the system. Instead of refining the initial cell state space, we start with a small initial state space region, analyse other interesting regions of the state space and incorporate them into a cluster of cell mapping solutions. By this approach, trajectories escaping the original state space region can be followed automatically and additional objects in the state space can be discovered.

To illustrate the benefits of the method, we present the exploration of the phase-space of the *micro-chaos map* – a simple model of digitally controlled systems.

Keywords: `discretized state space, adaptive global analysis, algorithm, micro-chaos`

Highlights:

- Extension of the Simple Cell Mapping method by adaptively expanding cell state space.
- New state space regions can be attached to the cluster of SCM solutions.
- Clustering makes parallel computation trivial.

1. Introduction

Cell Mapping methods (or shortly CM methods) were introduced by C.S. Hsu [1], in order to make the quick and thorough global analysis of nonlinear systems possible. CM methods discretize a region of the state space, thus creating the so called cell state space. For each cell one or more image cell is assigned (to where the dynamics lead from that cell), and by analysing the resulting graph or Markov-chain, periodic orbits, fixed points and their domains of attraction can be found.

The simplest CM method is the Simple Cell Mapping (SCM) and in the simplest case the cell state space is an n-dimensional grid of cells of the same size. The basic idea of the SCM method is that each cell has a single image, which is usually determined using the Centre Point Method [1], namely, a single trajectory

*Corresponding author
Email address: `gyebro@mm.bme.hu` (Gergely Gyebrószki)
[1]Department of Applied Mechanics, Budapest University of Technology and Economics
[2]MTA-BME Research Group on Dynamics of Machines and Vehicles

from the centre of the *cell domain* is examined. In other words, all states within a cell are *mapped* to a single cell. Due to this property, the method is able to classify cells either as *periodic* cells (belonging to a periodic group) or *transient* cells (leading to a periodic group). Successful classification of all cells forms the *solution of the SCM*.

There are many variation of the CM methods, usually a relatively fast CM method (for example SCM) is applied to the initial state space region, then further analysis is carried out at certain locations, using more advanced methods (Generalized Cell Mapping, for instance), typically with refined cell state space [2], [3], [4]. These methods are excellent if the *interesting region* of the state space is known, but if that'sthat is not the case, a method capable of automatically extending the analysed state space region could be more suitable. Our goal is to extend the Simple Cell Mapping with such capability.

To emphasize the relevance of adaptive state space extension, one could recall the following situations:

- The dynamical system has an expectedly complex state space and the enclosing region of state space objects is not known.
- The dynamical system has more than one attractors, and not all of them are found in the initial state space region. Escaping trajectories indicate the possible direction of other attracting structures.
- A lower dimensional state space object, e.g., a basin boundary is being followed.
- Examination of global bifurcations or crises in dynamical systems in cases when the structure and/or the size of state space objects change abruptly during the variation of certain parameters. This situation is typically encountered in piecewise smooth systems.
- Analysing diffusion-like processes, for example intermittent maps [5].

Our approach to solve the problem of state space extension is to find an adjacent region to the initial state space, to where most of the trajectories escape. Afterwards, a separate CM solution is calculated on that region and the two solutions are joined. Upon the joining procedure, new state space objects residing on the boundary of the two cell state spaces are also discovered. This paper introduces this extension, particularly for the Simple Cell Mapping method, because it is the simplest adequate method to discover all objects in the state space [1]. The method of joining separate SCM solutions to a cluster of SCM solutions is referred to as Clustered SCM method. Based on these results, optional later analysis can be carried out using more advanced CM methods [6].

As an example of application, we show the analysis of the so-called *micro-chaos map* [7], where multiple disconnected attractors – possibly consisting of distinguishable communicating repellers – are present in the state space. The behaviour of this piecewise smooth system fits into moremost of the aforementioned situations, as it exhibits a pattern of chaotic attractors and crisis phenomena with the appearance or disappearance of chaotic attractors/repellers [8].

1.1. Definitions and abbreviations

This section describes the basics terms, definitions and properties related to the Simple Cell Mapping, which are used throughout the paper. Also some auxiliary subroutines are presented, which are necessary for the implementation of the method (see Figure 1).

- *Cell state space* (CSS): the bounded and discretized state space region, which is continuously covered by arbitrary *cell domains*. In the simplest case n-dimensional rectangular cuboids of the same size can be used to discretize an n-dimensional state space.
- *Cell domain*: bounded domain of the state space, part of the *cell state space*. In the simplest case it can be represented by a centre point in the state space and lengths along each dimension.
- *Cell*: object having its unique *index* referencing to a *cell domain* and various properties (e.g. *image*, *pre-image*).

- *Cell index* (or shortly *index*): cell property; a unique identifier.
- *Image*: property of a cell, one or more reference to other cells. The dynamics from the *cell domain* corresponding to the cell lead to the *cell domain(s)* indexed by the *image(s)*.
- *Pre-image*: property of a cell, one or more reference to other cells. The dynamics from the *cell domain(s)* indexed by the *pre-image(s)* lead to the *cell domain* corresponding to the cell.
- *Sink cell* (SC): a special *cell* indexing the unbounded region of the state space outside the CSS. Once a trajectory enters the sink, its evolution is no longer followed, to express this, the *image* of the sink is itself by definition.
- State-to-index (or shortly *index()*) function: is a surjective function returning the *index* corresponding to the *cell domain* covering the given point in the state space.
- Index-to-domain (or shortly *domain()*) function: is a bijective function returning the *cell domain* representation for the given *index*.
- *Cell sequence*: A set of cells formed by tracking the *image* of cells subsequently. (See cells $\{7, 2, 4, 11, 18, 24, 16\}$ in Figure 1.)
- *Periodic group* (PG): A part of a cell sequence, that might constitute a periodic motion. A periodic cycle of n cells forms a periodic group, with periodicity n (or shortly an n-P group). Each cell within the PG is a *periodic cell* with period n, or shortly n-P cell [1]. (For example, the sink cell is a 1-P cell and forms a 1-P group.)
- *Transient cell*: Cell sequences leading to an n-P cell contain an n-P group at the end of the sequence. All other cells within the sequence are *transient* cells leading to that periodic group, forming a *transient cell sequence*.
- *Transient cell sequence*: cell sequences with their destination n-P cells removed form a *transient cell sequence*, see Figure 1.
- *Group number* (g): For each periodic group a unique group number is assigned. All periodic cells within a PG and all transient cells leading to that PG have the same specific group number assigned.
- *Step number* (s): property of a cell, the number of steps required to reach a PG. *Periodic cells'* step number is $s = 0$, while *transient cells'* step number is $s > 0$.
- *Domain of Attraction* (DoA): the DoA of a PG with group number g is the set of (transient) cells with the same group number g and step number $s > 0$. The Domain of Attraction can be thought as the discretization of the *Basin of Attraction* (see [9], [10] and for its numerical exploration [11].)
- *SCM solution*: After the successful execution of the SCM method, besides the initial cell properties, the *group number* and *step number* properties are assigned to each cell. At this stage all periodic groups and their domain of attraction are found, and we call the cell state space and its properties the *SCM solution*.

2. Joining two SCM solutions

This section describes the procedure of joining two SCM solutions with non-overlapping cell state spaces. No other restrictions apply to the cell state spaces, even non-adjacent regions can be joined. First, the possible relationships between cells of the SCMs are examined, then the algorithm of joining is explained supported by a pseudo-code of the procedure.

We adopt the following conventions regarding the *SCM solutions* to aid the joining procedure.

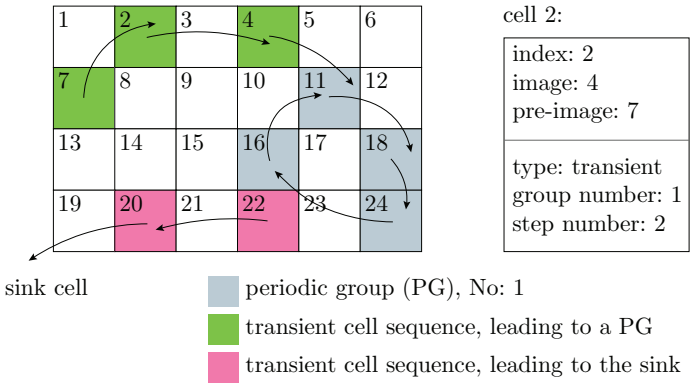

Figure 1: Explanation of the definitions introduced in Section 1.1

- Group number $g = 0$ is assigned to the sink cell(which is a 1-P cell by definition). Also the sink cell's index is 0.
- A new property, called *cell mapping index* (shortly: *cmid*) is assigned to each cell as an extension to its *group number* indicating which SCM contains the group referenced by the *group number*. Initially all SCM solutions have a unique *cmid*, and all cells within an SCM solution hashave that same *cmid*.
- Cells have an auxiliary *state* property, which can take any of the following three values: UNTOUCHED, UNDER_PROCESSING, PROCESSED. This property is used to keep track of the solution procedure.

2.1. Relationship of two SCM solutions

Upon joining two SCM solutions, transient cell sequences leading to the sink cell are examined, because these cell sequences might enter the other SCM's cell state space and lead to an object within the *united cell state space* – the union of the two cell state spaces. The state space region outside the *united cell state space* is called *reduced sink*. While examining an SCM solution's transient cell sequences leading to the original sink, the following cases can occur (Figure 2).

1. The *transient cell sequence* leads to a known destination:
   (a) the *reduced sink* or
   (b) a *periodic* or *transient* cell with group number $g > 0$ of the other SCM.
2. The *transient cell sequence* leads to a cell of the other SCM, which belongs to the domain of attraction of the sink (so the cell's group number is $g = 0$). This means that the final destination of the sequence is not known yet.

Considering the above cases, only Case 2 requires further analysis. Otherwise, transient cell sequences can be updated with a new group and step number (along with a new cell mapping id), corresponding to their new destination.
The procedure of joining two SCM solutions is therefore divided into two stages. Stage 1 enumerates all *transient cell sequences* and also updates those corresponding to Case 1. Stage 2 analyses the remaining sequences of Case 2.

2.2. Cell tree mapping

It is clear, that cell sequences leading to the other SCM's sink cell's domain of attraction (See Case 2 in Section 2.1) will eventually have one of the already existing periodic groups (including the *reduced sink*) as their destination, or they might form a new periodic group possibly with some extra transient cells leading to that PG.
This calls for the idea of mapping these remaining transient cell sequences onto each other (or some already

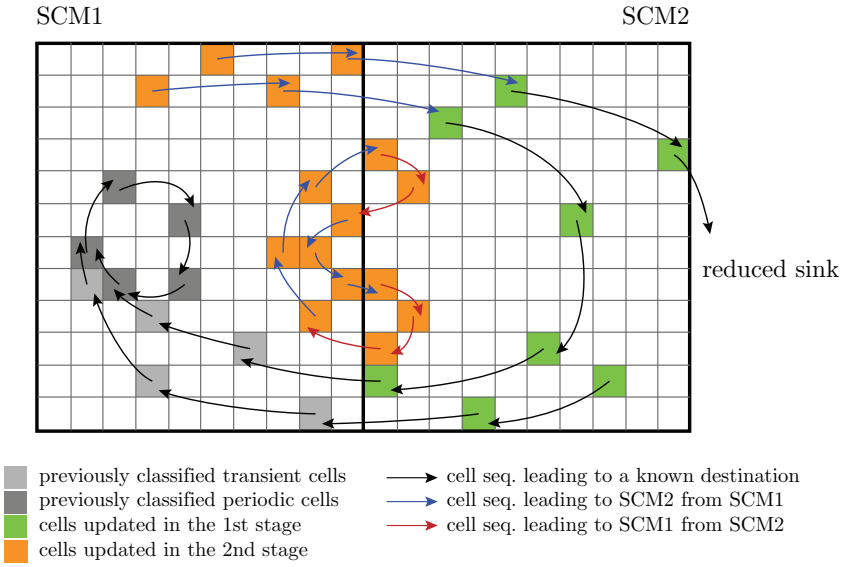

Figure 2: Joining of previously calculated adjacent SCM solutions. Cell sequences which lead to a known destination can be updated in Stage 1 (green cells), while sequences leading to another unclassified sequence or transient cell need further analysis in Stage 2 (orange cells). As a result, new periodic groups can be found close to the boundary of SCM1 and SCM2.

determined cell). Transient cell sequences form trees called *cell trees* having a single cell as destination (which belongs to the other SCM), therefore these trees can be handled just like cells in SCM. The *image* of a *cell tree* is either a cell which was updated in the first stage of the joining procedure (Case 1 in Section 2.1), or alternatively a member cell of another *cell tree* of the other SCM. Tracking the images of *cell trees* creates *tree sequences*. A *tree sequence* either leads to an already existing periodic group or forms a new periodic group and some transient cells leading to that group. Figure 3 illustrates two cell trees mapped to each other.

Shortly, the SCM method can be applied to the *cell trees*. If a *tree sequence* leads to a previously processed cell, all of its member cells can be tagged with the appropriate *cmid, group* and *step* numbers. Otherwise the trees form a graph containing a single cycle – the new periodic group – and branches which are transient cells belonging to that group, hence the *cmid, group* and *step* numbers can be updated. (The new periodic groups obtained this way must be added to one of the SCM solutions to have a valid *cell mapping index*.)

2.3. The algorithm of joining

This subsection describes the algorithm of joining adjacent SCM solutions. The algorithm is divided into preprocessing and two stages of classifying cell sequences which previously led to the sink cell.

Throughout the presentation of the algorithm, multiple SCM solutions will be examined. For the sake of simplicity, object oriented notation is used, with simple classes for describing the cell and SCM solution including the cell state space. See Algorithms 1 and 2 for these classes. In the pseudo codes the `.` (dot) operator is used to access data or function members of these objects. For instance `scm.cells[i].index` accesses the *index* of the i-th cell of the `scm` object. Furthermore, $\triangleright$ indicates clarifying comments.

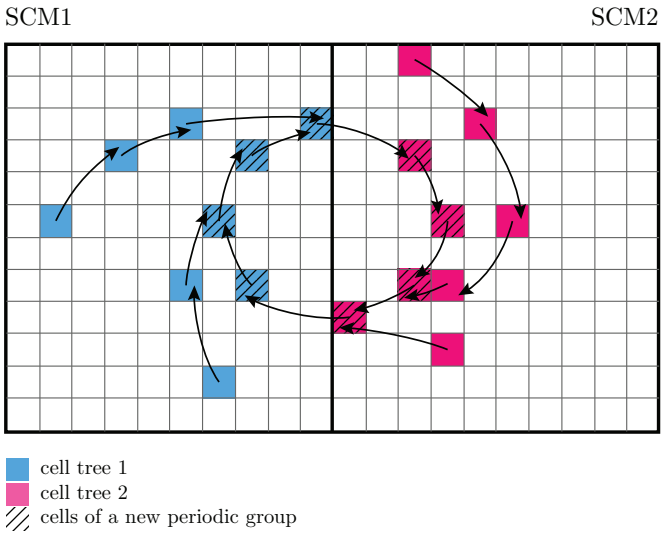

Figure 3: Illustration of the notion of cell tree mapping. Cell trees 1 and 2 are mapped to each other. The graph formed by them contains a cycle (new periodic group), and all other branches are transient cells leading to that group.

Algorithm 1 Class for cell

class CELL
  index $\subset \mathbb{N}$
  image $\subset \mathbb{N}$
  domain
  group $\subset \mathbb{N}$
  step $\subset \mathbb{N}$
  type $\subset$ { UNKNOWN, TRANSIENT, PERIODIC }
  state $\subset$ { UNTOUCHED, UNDER_PROCESSING, PROCESSED }
end class

Algorithm 2 Class for simple cell mapping

class SCM
  cell array of CELL objects
  cellCount $\subset \mathbb{N}$ ▷ the number of cells in the cell state space
  periodicGroupCount $\subset \mathbb{N}$ ▷ the number of periodic groups in the SCM solution
  index(...)
  domain(...)
end class

During the preprocessing the cells corresponding to the domain of attraction of the sink cell for both SCM solutions are identified. This can be done by selecting cells with group number 0, which belong to the 1-P group of the sink cell. Checking the step number is not necessary, since all cells with 0 group number must be transient cells. For the pseudo code of preprocessing see Algorithm 3 and 4.

Algorithm 3 Identification of sink cell's domain of attraction

Input: *scm* object representing an SCM solution
Output: array of indices of sink cell's domain of attraction

```
1: function GetSinkDomainOfAttraction(scm)
2:     sinkDoA ← ∅
3:     for i ← 1, scm.cellCount do
4:         if scm.cell[i].group = 0 then
5:             sinkDoA ← sinkDoA ∪ i
6:             scm.cell[i].state ← UNTOUCHED ▷ invalidate previously processed cell
7:         end if
8:     end for
9:     return sinkDoA
10: end function
```

Algorithm 4 Preprocessing of two SCM solutions

Input: objects representing SCM solutions
Output: array of indices for both sink's domain of attraction

```
1: function Preprocess(scm1, scm2)
2:     sinkDoA1 ← GetSinkDomainOfAttraction(scm1)
3:     sinkDoA2 ← GetSinkDomainOfAttraction(scm2)
4:     return {sinkDoA1, sinkDoA1}
5: end function
```

Once the domain of attraction of the sink cell is identified for each SCM solution, the first stage of joining examines *transient cell sequences* and updates cells in Case 1 of Section 2.1, see Algorithm 5. The **for** loop in line 3 starts a new cell sequence by taking the next UNTOUCHED cell from the domain of attraction of the sink cell. The **while** loop in line 10 builds the cell sequence and updates all cells accordingly. If the condition in line 12 is true, then the cell sequence is still within the original cell state space. In this case the *cmid* is checked in line 14. If the currently examined cell has the same *cmid*, the current cell sequence either touches another cell sequence (line 16) and prepended to that cell sequence (thus forming a cell tree), or touches an already processed cell (line 23) in which case the cell sequence can be updated accordingly, or touches an UNTOUCHED cell (line 29) which results in continuing the current sequence by examining that cell's image.

If the condition in line 14 (*cmid* check) yields false, the cell sequence touches another cell sequence transiting to the other SCM's state space, therefore the current sequence can be updated accordingly. In cases, when $imz = 0$ is fulfilled (line 40), the cell sequence leaves the cell state space. Line 45 checks whether the current cell sequence enters the cell state space of the other SCM. In this case the sequence either touches a cell with $g \neq 0$ (line 48) when the current sequence is updated, or touches a cell with $g = 0$ (line 52) when the current cell sequence (*seq*) is stored in the array of cell trees (*cellTrees*) for further analysis. Lastly, if both cell state space have 0 (sink) index for the cell (see line 58), the current sequence leads to the reduced sink.

Algorithm 5 Stage 1 of the joining procedure

Input: Examined SCM solution and its DoA of sink, other SCM solution
Output: Updated SCM solution object scm, cell trees which require further processing

```
1: function STAGE1(scm, sinkDoA, otherScm)
2:     cellTrees ← ∅
3:     for i ← 0, sinkDoA.size do
4:         seq ← ∅
5:         z ← sinkDoA[i]
6:         if scm.cell[z].state = UNTOUCHED then
7:             ▷ Create new cell sequence
8:             seq ← seq ∪ z
9:             left ← false
10:            while left = false do
11:                imz ← scm.cell[z].image
12:                if imz ≠ 0 then
13:                    cmimz ← scm.cell[imz].cmid
14:                    if cmimz = cmid then
15:                        if scm.cell[imz].state = UNDER_PROCESSING then
16:                            ▷ This sequence touches another sequence under processing
17:                            left ← true
18:                            ct ← scm.cell[imz].cellTreeIndex
19:                            Tag cells in seq as UNDER_PROCESSING, assign ct as cellTreeIndex
20:                            ▷ The current sequence is prepended to cell sequence/tree with index ct
21:                            cellTrees[ct] ← seq ∪ cellTrees[ct]
22:                        else if scm.cell[imz].state = PROCESSED then
23:                            ▷ This sequence touches an already processed cell (Case 1.b)
24:                            left ← true
25:                            g ← scm.cell[imz].group
26:                            cm ← scm.cell[imz].cmid
27:                            Tag cells in seq as PROCESSED and assign new group number g and cmid cm
28:                        else
29:                            ▷ Append cell to sequence and continue
30:                            seq ← seq ∪ imz
31:                            z ← imz
32:                        end if
33:                    else
34:                        ▷ This sequence touches another sequence transiting to the other SCM (Case 1)
35:                        left ← true
36:                        g ← scm.cell[imz].group
37:                        cm ← scm.cell[imz].cmid
38:                        Tag cells in seq as PROCESSED and assign new group number g and cmid cm
39:                    end if
```

```
40:                 else
41:                     ▷ This sequence leaves the cell state space (imz = 0)
42:                     left ← true
43:                     ▷ Get image using the other SCM's cell state space
44:                     imz ← otherScm.index(step(scm.cell[z].center))
45:                     if imz ≠ 0 then
46:                         ▷ This sequence enters other SCM solutions cell state space
47:                         g ← otherScm.cell[imz].group
48:                         if g ≠ 0 then
49:                             ▷ This sequence touches a periodic group with g > 0 (Case 1.b)
50:                             cm ← otherScm.cell[imz].cmid
51:                             Tag cells in seq as PROCESSED and assign new group number g and cmid cm
52:                         else
53:                             ▷ This sequence touches a transient cell of the other SCM's sink,
54:                             ▷ save this sequence for further analysis (Case 2)
55:                             Tag cells in seq as UNDER_PROCESSING and assign new group g and cmid cm
56:                             cellTrees ← cellTrees ∪ seq
57:                         end if
58:                     else
59:                         ▷ This sequence leads to the reduced sink (Case 1.a)
60:                         Tag cells in seq as PROCESSED
61:                     end if
62:                 end if
63:             end while
64:         else
65:             ▷ skip cell
66:         end if
67:     end for
68:     return cellTrees
69: end function
```

In the second stage, for Case 2 in Section 2.1 a *cell tree mapping* is carried out (Algorithm 6). The **for** loop in line 3 starts examining an UNTOUCHED cell tree and the **while** loop in line 10 builds a sequence of cell trees (see variable: *treeSequence*). While examining the image tree ($ctImage$) of the current cell tree ($cellTrees[i]$), the following cases can occur:

- The image of the current cell tree is a cell which was updated in Stage 1 of the procedure (line 11). All cells in the sequence of trees can be updated.
- The image tree of the current cell tree is PROCESSED (line 18), therefore, the sequence of trees touches a known destination, and all cells in the sequence of trees can be updated accordingly.
- The image tree of the current cell tree is UNDER_PROCESSING (line 23), and a new periodic group and transient cells are found. All cells within the sequence of trees are examined and tagged as *periodic* (cycle in the sequence of trees) or *transient* (branches). See Figure 3.
- The image tree of the current cell tree is UNTOUCHED (line 29), the image tree is appended to the sequence of trees, and the examination of the tree sequence is continued.

In the end of Stage 2, all cell trees are processed and new periodic groups (if any) with their domain of attraction (transient cells) are found. The complete procedure of joining is summarized in Algorithm 7. The two SCM solutions joined this way form a cluster of cell mapping solutions, which can be further extended similarly.

Algorithm 6 Stage 2 of the joining procedure

Input: Cell Sequences Tree arrays and SCM objects
Output: Updated SCM solutions

function STAGE2($cellTrees1, cellTrees2, scm1, scm2$)
  $cellTrees \leftarrow cellTrees1 \cup cellTrees2$
  for $i \leftarrow 0, cellTrees.size$ **do**
    if $cellTrees[i].state =$ UNTOUCHED **then**
      ▷ Start examining sequence of cell trees
      $cellTrees[i].state \leftarrow$ UNDER_PROCESSING
      $treeSequence \leftarrow \emptyset \cup i$
      $processing \leftarrow True$
      $ctImage \leftarrow cellTrees[i].imageTree$
      while processing **do**
        if $ctImage = null$ **then**
          ▷ There is no sequence image, image cell must be already PROCESSED in Stage 1
          $imageCell \leftarrow cellTrees[i].cell[0].image$
          Update all cells in each cell tree of the current $treeSequence$
          Tag all cell tree in $treeSequence$ as PROCESSED
        else
          ▷ Cell tree mapping
          if $cellTrees[ctImage].state ==$ PROCESSED **then**
            ▷ The sequence of trees leads to a known destination
            Update all cells in each cell tree of the current $treeSequence$
            Tag all cell tree in $treeSequence$ as PROCESSED
            $processing \leftarrow False$
          else if $cellState[ctImage].state =$ UNDER_PROCESSING **then**
            ▷ New periodic group and transient cells are found
            $g \leftarrow nextGroupNumber()$
            Update all cells in each cell tree of the current $treeSequence$
            Tag all cell tree in $treeSequence$ as PROCESSED
            $processing \leftarrow False$
          else
            ▷ $cellTrees[ctImage].state ==$ UNTOUCHED
            ▷ Tag this cell tree as UNDER_PROCESSING,
            ▷ append to $treeSequence$ and continue
            $treeSequence \leftarrow treeSequence \cup ctImage$
            $cellTrees[ctImage].state \leftarrow$ UNDER_PROCESSING
          end if
          ▷ Get next image sequence
          $ctImage = cellTrees[ctImage].imageSeq$
        end if
      end while
    else if $cellTrees[i].state =$ PROCESSED **then**
      ▷ Skip already processed cell tree
    end if
  end for
  return $\{scm1, scm2\}$
end function

Algorithm 7 Procedure of joining two SCM solutions

Input: SCM objects representing SCM solutions
Output: updated SCM objects
1: **function** JOIN($scm1, scm2$)
2: $\{sinkDoA1, sinkDoA2\} \leftarrow$ PREPROCESS($scm1, scm2$) ▷ See Algorithm 4
3: $cellTrees1 \leftarrow$ STAGE1($scm1, sinkDoA1, scm2$) ▷ See Algorithm 5
4: $cellTrees2 \leftarrow$ STAGE1($scm2, sinkDoA2, scm1$)
5: $\{scm1, scm2\} \leftarrow$ STAGE2($cellTrees1, cellTrees2, scm1, scm2$) ▷ See Algorithm 6
6: **return** $\{scm1, scm2\}$
7: **end function**

3. Properties and possible extensions

3.1. Complexity of joining

It can be seen that the complexity of calculating an SCM solution is $O(n)$ where n is the number of cells in its cell state space [12]. This comes from the fact that every cell needs constant amount of operations for initialization, and their state changes twice, first to UNDER_PROCESSING then to PROCESSED (Algorithm 8). The complexity of preprocessing (Algorithm 3) is also linear, since the body of loop in line 3 contains constant amount of operations. For SCM solutions with cells n and m, the complexity of the preprocessing is $O(n+m)$.
The first stage of the joining procedure (Algorithm 5) contains an outer **for** loop (line 3) and an inner **while** loop (line 10), however, similarly to the SCM method, every cell is tagged with a new state maximum twice, therefore, the complexity of the first stage is $O(n)$ where n is the number of cells in the sink's domain of attraction.
Lastly, it can be seen that the complexity of the second stage (Algorithm 6) is also linear in terms of the number of total cells in the cell tree lists. This property can be shown with the same approach used in the previous case; every tree sequence is tagged with a new state maximum twice.
Introducing $n_{\text{sink}} \leq n$ and $m_{\text{sink}} \leq m$ for the number of cells in the domain of attraction of the sink cell, the complexity of the joining procedure can be written as $O(n_{\text{sink}} + m_{\text{sink}})$. The linear nature of the joining procedure can also be seen in the computation times presented in Table 1.

3.2. Simple continuous tiling of the state space

In Section 2 the procedure of joining two arbitrary SCM solutions was introduced. This section describes the simplesta simple algorithm for adaptively selecting an adjacent state space region (of the same shape and size as the original SCM solution) where most of the trajectories escape to. For convenience, the original cell state space is chosen to be an n-dimensional rectangular cuboid.
After selecting the initial state space region for the SCM solution one divides the sink cell into 3^n sub-regions.the unbounded outer state space region into adjacent subregions plus an unbounded non-adjacent region. To do this, the sink cell is divided into 3^n sub-regions. From these 3^n sub-regions, $3^n - 1$ are adjacent and equal size to the cellinitial state space and the remaining region – the rest of the sink cell – is non-adjacent to the cellinitial state space. These sub-regions are illustrated in Figure 4. During the calculation of the initial SCM solution, the number of cells entering these sub-regions can be counted.
Let us assume that the number of cells whose image belongs to the i-th adjacent sub-region r_i is c_i, where $i = 1, 2, ..., 3^n - 1$. Amongst the adjacent state space regions, the one with the largest number c_k is selected. The index of the selected new adjacent state space region is

$$k = \sigma(\max(\{c_i : i = 1, 2, ..., 3^n - 1\})),$$

where $\sigma(c_k) := k$ is an index function. After solving the new SCM belonging to the newly selected region, a cluster of two SCM solutions is formed, and the procedure can be continued similarly, leading to a continuous tiling of a state space region.

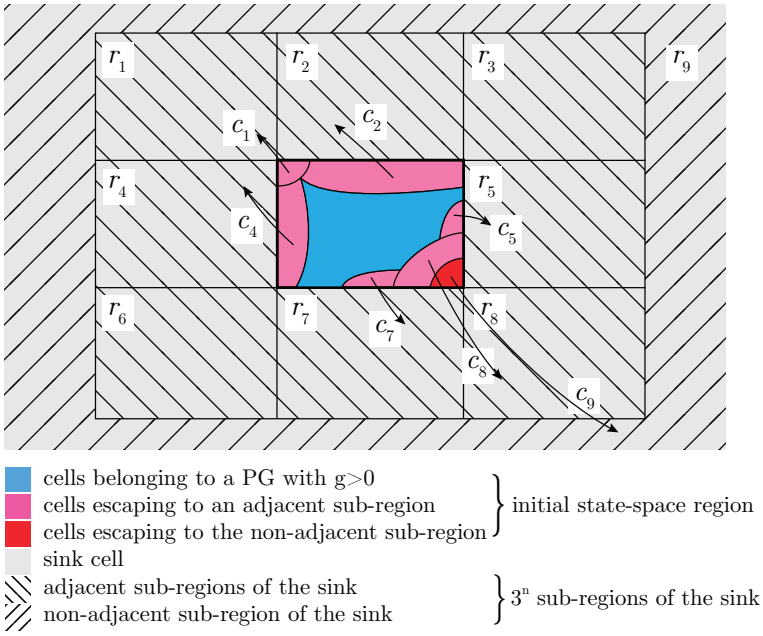

Figure 4: Adjacent regions of a 2D cell state spaceSub-regions of the sink cell in case of a 2D cell state space. Sub-regions $r_i \dots r_{3^n-1}$ are adjacent to the initial state space region, sub-region r_{3^n} is non-adjacent.

4. Application and Results

4.1. Analysis of the micro-chaos map

Although the Clustered SCM method is independent of the system's dimension, the results can be displayed most conveniently for systems with 2D state space. In search for a system that exhibits complex state space topology in 2D, simple problems of control engineering were considered. As the inverted pendulum is the archetype of stabilization problems in control theory, we have chosen a single degree of freedom inverted pendulum with a so-called proportional-derivative controller, where the quantized control force is calculated at sampling intervals τ, and kept constant within the interval (zero-order hold), as shown in Figure 5. The aforementioned digital effects – sampling and round-off – lead to the phenomenon of *micro-chaos*, i.e., small amplitude chaotic oscillations [13].

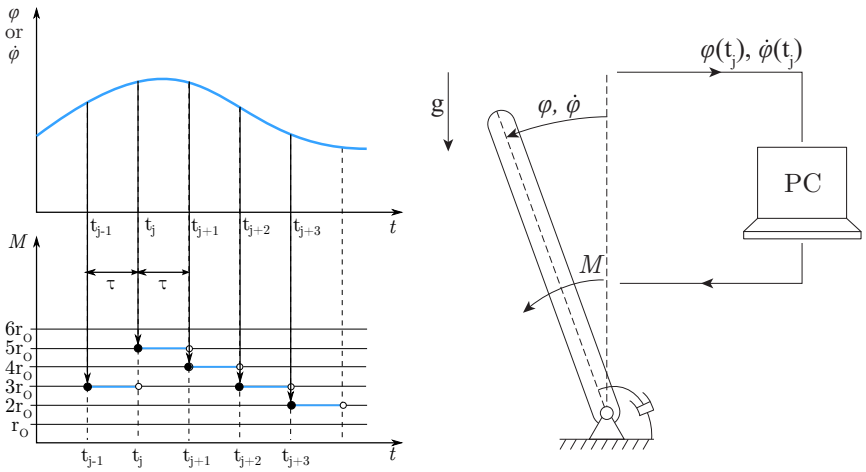

Figure 5: Illustration of sampling and quantization and a digitally controlled inverted pendulum.

The equation of motion of the controlled inverted pendulum is

$$\ddot{\varphi}(t) + 2\alpha\delta\dot{\varphi}(t) - \alpha^2\varphi(t) = -r_O \,\mathrm{Int}\left(\frac{P\varphi(t_j)}{r_O} + \frac{D\dot{\varphi}(t_j)}{r_O}\right), \qquad j = 1, 2, \ldots, \tag{1}$$

where α is related to a characteristic time constant, δ is the relative damping, P and D are control parameters and r_O is the resolution of the control torque (Figure 5). The resolution is taken into account with the Int() function, which denotes rounding towards the origin. According to the solution of the linearized, dimensionless equation of motion, the following mapping can be derived between the states at subsequent sampling instants [8]:

$$\mathbf{y}_{i+1} = \mathbf{U}\,\mathbf{y}_i + \mathbf{b}\,F_i, \tag{2}$$

where $F_i = \mathrm{Int}(\hat{P}x_i + \hat{D}x_i')$, $\mathbf{y} = [x_i \quad x_i']^T$, and:

$$\mathbf{U} = \frac{e^{-\hat{\alpha}\delta}}{\Gamma}\begin{bmatrix} \Gamma\cosh(\hat{\alpha}\Gamma) + \delta\sinh(\hat{\alpha}\Gamma) & \sinh(\hat{\alpha}\Gamma)/\hat{\alpha} \\ \hat{\alpha}\sinh(\hat{\alpha}\Gamma) & \Gamma\cosh(\hat{\alpha}\Gamma) - \delta\sinh(\hat{\alpha}\Gamma) \end{bmatrix}, \tag{3}$$

$$\mathbf{b} = \frac{1}{\hat{\alpha}^2\Gamma}\begin{bmatrix} \Gamma - e^{-\hat{\alpha}\delta}\left(\Gamma\cosh(\hat{\alpha}\Gamma) + \delta\sinh(\hat{\alpha}\Gamma)\right) \\ -\hat{\alpha}e^{-\hat{\alpha}\delta}\sinh(\hat{\alpha}\Gamma) \end{bmatrix}. \tag{4}$$

Hat symbols denote dimensionless variants of previously introduced quantities, and $\Gamma = \sqrt{1+\delta^2}$.
The quantization according to the Int() function introduces switching lines on the state space for every integer value. By examining the direction field of Equation (2), one can see an alternating pattern of unstable saddle points and switching lines [14], [15] see Figure 6.

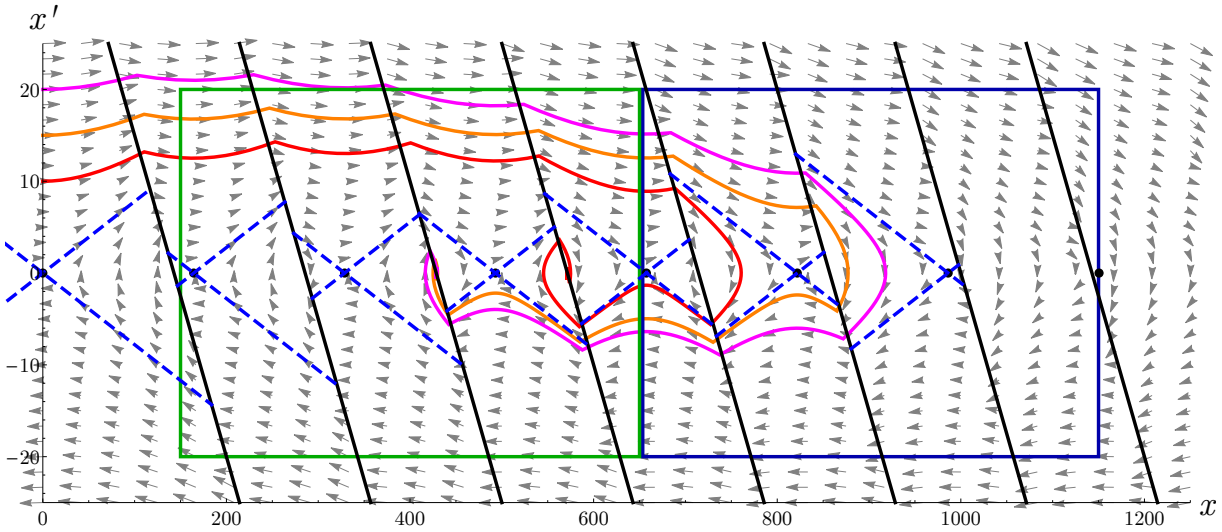

Figure 6: The state space of micro chaos map (2) at parameter values $\hat{\alpha} = 0.078$, $\delta = 0$, $\hat{P} = 0.007$, $\hat{D} = 0.02$. Dashed blue lines are the stable and unstable manifolds of saddle points. Three example trajectories leading to chaotic attractors are shown. The subsequent points of the trajectories are connected with line sections for better visibility. The green and blue rectangles show the initial and the adaptively chosen state space regions of the first example, respectively (see Figure 7).

The Clustered SCM method is applied to the micro-chaos map, and the resulting cluster of two SCM solutions is illustrated by coloured images in Figures 7-12. Red colour indicates transient cells leading to the sink, other coloured regions illustrate the domain of attraction of other periodic groups. The periodic groups residing at the intersections of the x-axis and the switching lines are denoted by black dots. These PGs correspond to very small chaotic attractors of the micro-chaos map. White lines indicate the switching lines and dashed white lines denote the stable and unstable manifolds of the saddle points of the map. The initial state space region is placed on the left and the new subregion is on the right side, since the right

adjacent state space region contains the most escaping trajectories.

In the first example, no periodic groups reside at the boundary of the two state space regions (see Figure 7). Therefore, during Stage 1, all cells can be updated, except transient cell sequences of the initial region leading to a member cell of the domain of attraction of the new region's sink cell (see Figure 8). These sequences also lead to an already existing PG, but are updated in Stage 2 (as shown in Figure 9). The parameters of the micro-chaos map are $\hat{\alpha} = 0.078$, $\delta = 0$, $\hat{P} = 0.007$, $\hat{D} = 0.02$.

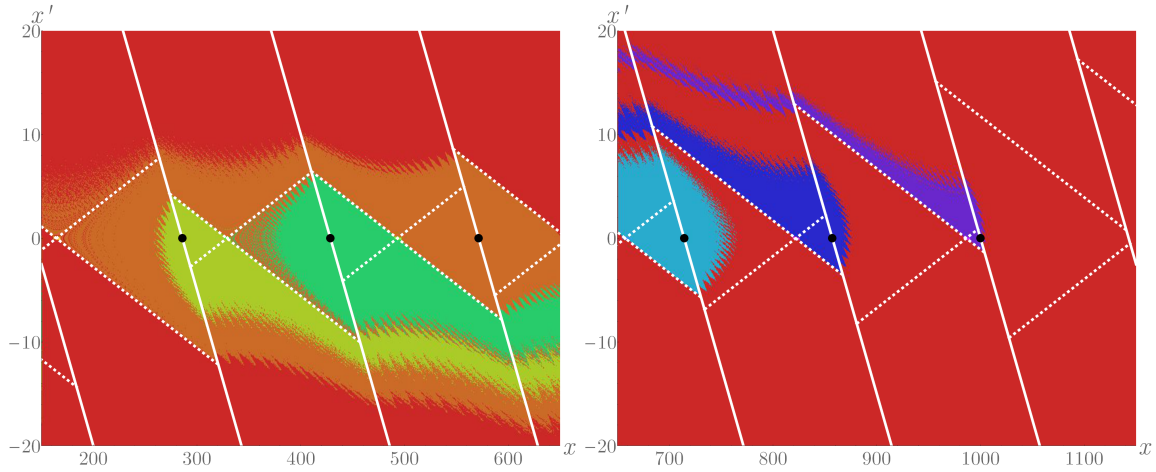

Figure 7: Example 1 – Illustration of initial SCM solutions before the joining procedure. The image on the left shows the initial state space region, the one on the right is the adaptively selected region. Both regions contain 3 chaotic attractors lying at the intersections of the x-axis and the switching lines.

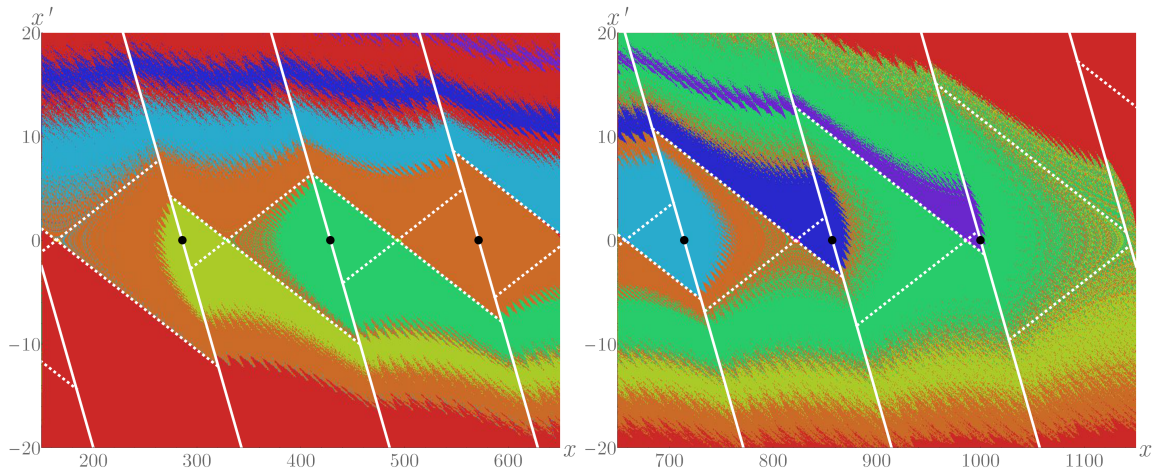

Figure 8: Example 1 – Illustration of SCM solutions after Stage 1 of the joining procedure. Cell sequences leading to a PG of the other SCM are updated (recoloured with the colour of the corresponding periodic group). The initial region contains some transient cell sequences which are stored for further processing in Stage 2. (See red bands at the top of the left image.)

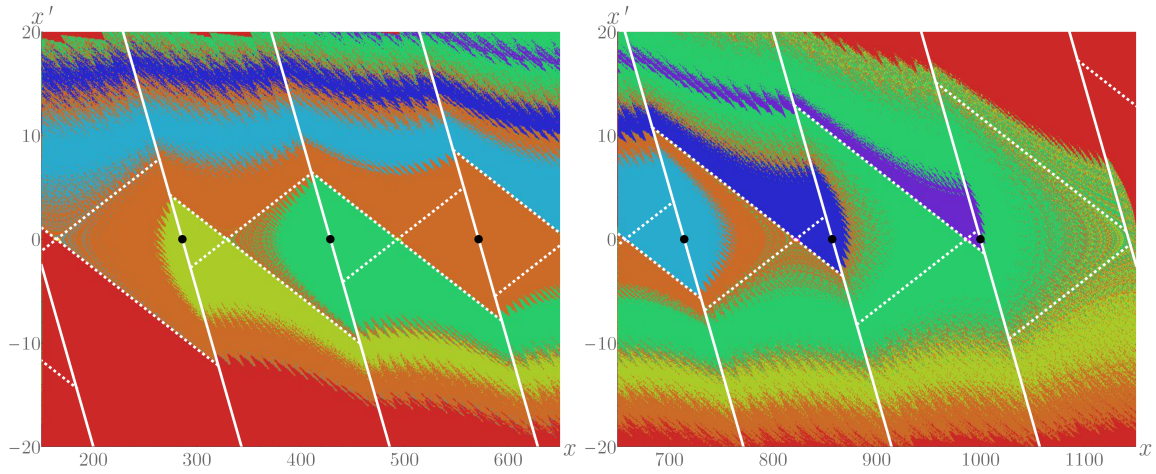

Figure 9: Example 1 – Illustration of SCM solutions after Stage 2 of the joining procedure. Examined cell trees are mapped to already processed cells (corresponding to the PGs with green and orange domain of attraction).

In order to show the creation of new periodic groups, another state space region is considered, for which a chaotic attractor of the map is just at the boundary of the region. The joining procedure is illustrated in Figures 10, 11 and 12. The parameters of the micro-chaos map are $\hat{\alpha} = 0.07$, $\delta = 0$, $\hat{P} = 0.007$, $\hat{D} = 0.02$. In the second example, a new periodic group and its domain of attraction are found during Stage 2.

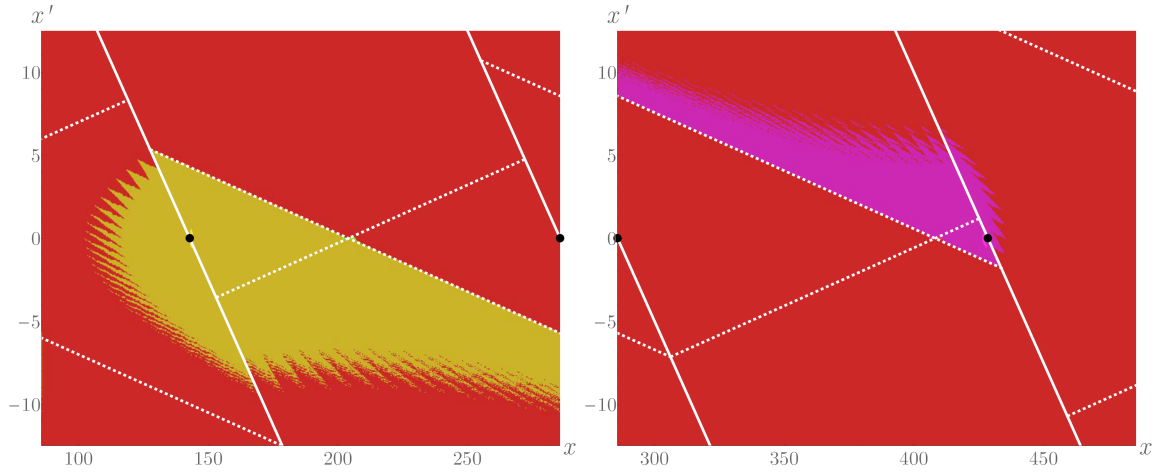

Figure 10: Example 2 – Illustration of initial SCM solutions before the joining procedure. The image on the left shows the initial state space region, the one on the right is the adaptively selected region. One chaotic attractor for each region is already detected (see yellow and pink domain of attractions). A third chaotic attractor is at the boundary of the two state space regions. (The black dot at the boundary of the state space regions denotes the third attractor's expected location.)

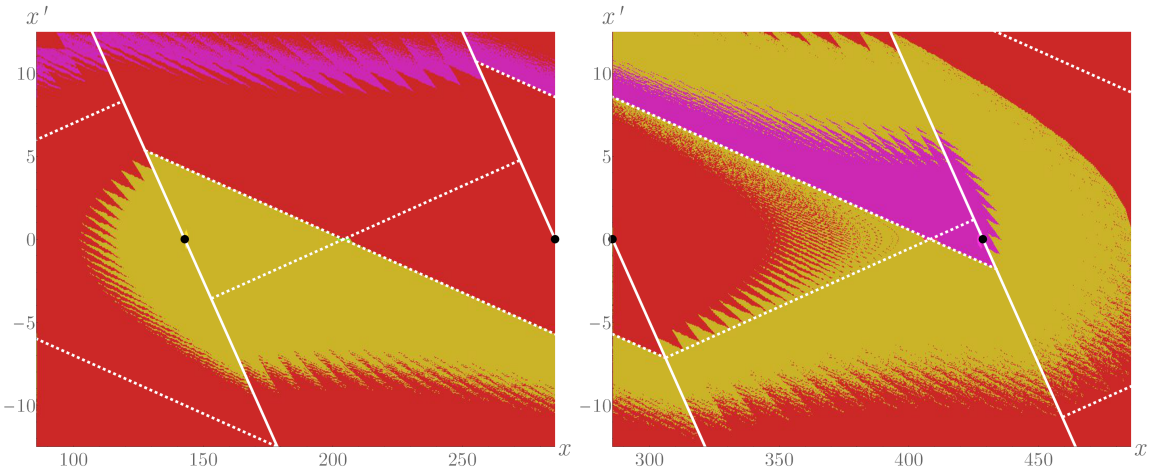

Figure 11: Example 2 – Illustration of SCM solutions after Stage 1 of the joining procedure. Cell sequences leading to the PG of the other SCM are updated (see yellow and pink cells). Both regions contain cell trees which are stored for further processing in Stage 2.

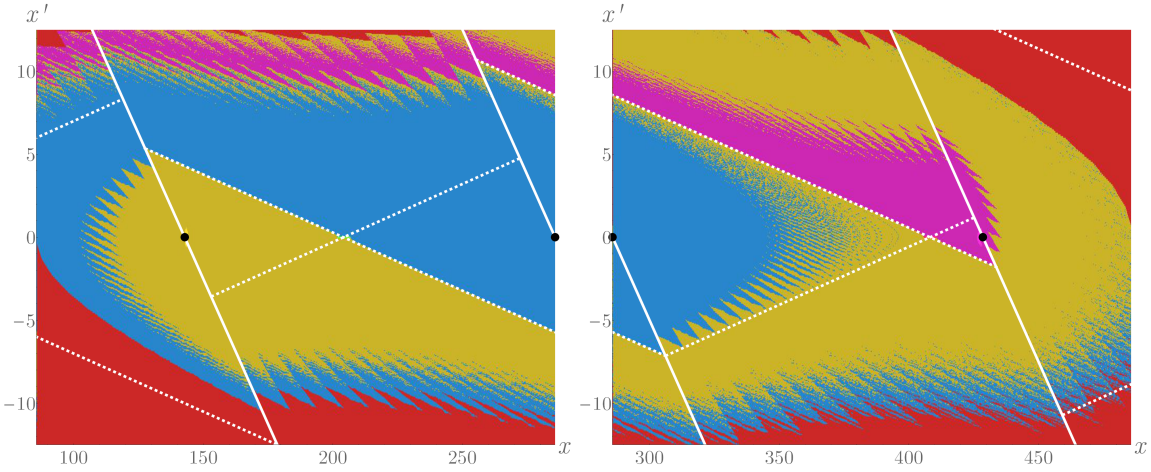

Figure 12: Example 2 – Illustration of SCM solutions after Stage 2 of the joining procedure. Examined cell trees are mapped to each other and a new periodic group is formed with its domain of attraction in blue.

4.2. Comparison of real computational efforts

To support the statements in Section 3.1, computation times for Example 1 are provided using the Clustered SCM and an SCM solution over the full region is calculated for comparison (see Table 1 and Figure 13). Since the calculation of SCM1 and SCM2 can be done in parallel, the total processing time is calculated as $t_{\text{total}} = \max(t_{\text{SCM1}}, t_{\text{SCM1}}) + t_{\text{joining}} t_{\text{total}} = \max(t_{\text{SCM1}}, t_{\text{SCM2}}) + t_{\text{joining}}$. (Computations were carried out using 2 cores of an Intel® Core™ i7-4700MQ CPU.)

In real situations it may happen that the two SCM solutions to be joined are of significantly different size. Consider the case when a 2D state space is displayed on the screen of a computer and the screen area is panned to move in the state space. Consequently, a separate SCM solution at the (narrow) state space region entering into the computer's screen must be calculated and joined to the already existing cluster. We checked the computation times for the case, when the original state space region is extended by 10%

towards an adjacent narrow state space region (see Table 2). The total processing time is calculated as $t_{\text{total}} = t_{\text{SCM2}} + t_{\text{joining}}$. One can see that the use of the Clustered SCM method makes nearly real-time application possible. Moreover, further optimizations can be introduced to the method specifically for the panning application, for example, adjacent state space regions can be joined in advance, to utilize idle CPU states.

The joining time only depends on the number of cells and state space topology, while the computation time of SCM solutions also depends on the effort needed to calculate the image cells. For systems, where greater effort is necessary for the calculation of images (e.g. flows), the computation time of joining is relatively smaller compared to the complete procedure.

Number of cells	CPU time [ms]				
	t_{SCM1}	t_{SCM2}	t_{joining}	t_{total}	SCM on full region
500000	395	386	89	484	844
1000000	780	791	190	981	1573
2000000	1550	1551	418	1969	3316
4000000	3234	3225	897	4131	6752
8000000	6638	6720	1935	8655	13389

Table 1: Computation times for Example 1. (See Figures 7-9.)

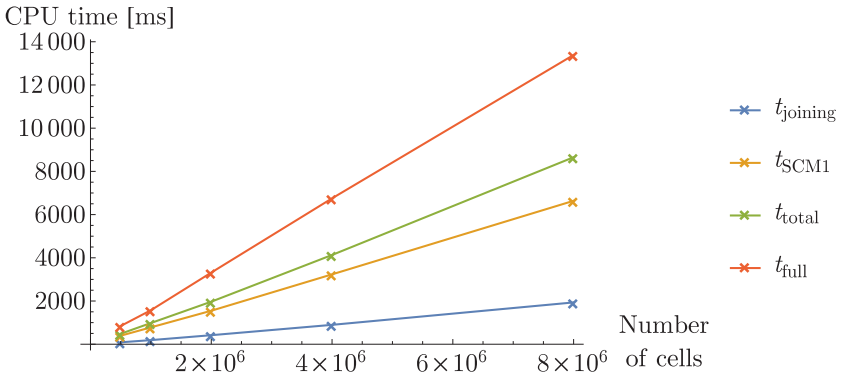

Figure 13: Comparison of computation times listed in Table 1.

Screen resolution	Number of cells		CPU time [ms]				
	n_{SCM1}	n_{SCM2}	t_{SCM1}	t_{SCM2}	t_{joining}	t_{total}	SCM on extended region
853×480	409440	40944	307	32	58	90	339
1280×720	921600	92160	661	66	129	195	740
1920×1080	2073600	207360	1581	188	361	549	1649
2880×1620	4665600	466560	3731	434	745	1179	4099
4320×2430	10497600	1049760	9689	753	1980	2733	11726

Table 2: Computation times for Example 1 in case of screen panning. Initially the whole computer screen is covered with the initial SCM solution (SCM1) and during panning a new SCM solution (SCM2) over a region with +10% width is added to the cluster. For comparison, the computation time of a single SCM solution on the extended state space region is included.

5. Conclusions

We have proposed the procedure of joining two SCM solutions – thus creating a cluster of SCMs – and described a simple way to select an adjacent state space region to be added to the cluster. We have shown, that the computational effort of the method is linear in terms of the total number of cells.
The method was applied to the micro-chaos map and two examples were presented to support the understanding of the stages of the procedure. Clustering has the following remarkable advantages.

- The method allows the continuation of the SCM solution after human assessment in cases when automatic state space extension is not used, but human supervision is conducted. Solving an SCM for a new region and incorporating it into the cluster is cheaper than solving an SCM over the whole extended state space (see Table 1).
- Parallelization is trivial as separate SCM solutions can be generated independently before the joining procedure. Also Stage 1 of the joining procedure (for each previously calculated SCM solution) can be done in parallel.
- The method is useful in real-time situations, where the region of interest is changing as a parameter is varied. Clustered cell mapping handles screen panning well, as a separate SCM solution at the (narrow) state space region entering into the computer's screen can be calculated quickly and joined to the already existing cluster (see Table 2).
- The proposed approach helps to overcome memory limitations by dividing large problems into smaller ones. During the generation of a clustered SCM solution, if all adjacent regions of a cluster are already examined, the SCM solution corresponding to the inner (fully surrounded) cluster can be written to disk and freed from memory.

The open-source C++ implementation of Clustered Cell Mapping method – along with materials in the topic of micro-chaos – are available at the website: `microchaos.com`.

Acknowledgements

This research was supported by the Hungarian National Science Foundation under grant no. OTKA K 83890.

References

[1] C. Hsu, Cell-to-Cell Mapping: A Method of Global Analysis for Nonlinear Systems, Vol. 64 of Applied Mathematical Sciences, Springer, Singapore, 1987.

[2] F.-R. Xiong, Z.-C. Qin, Y. Xue, O. Schtze, Q. Ding, J.-Q. Sun, Multi-objective optimal design of feedback controls for dynamical systems with hybrid simple cell mapping algorithm, Communications in Nonlinear Science and Numerical Simulation 19 (5) (2014) 1465–1473.

[3] H. Zou, J. Xu, Improved generalized cell mapping for global analysis of dynamical systems, Science in China Series E: Technological Sciences 52 (3) (2009) 787–800.

[4] B. de Kraker, J. A. W. van der Spek, D. H. van Campen, Extensions of cell mapping for discontinuous systems, in: M. Wiercigroch, B. de Kraker (Eds.), Applied Nonlinear Dynamics and Chaos of Mechanical Systems with Discontinuities, World Scientific, 2000, Ch. 4, pp. 61–102.

[5] R. Klages, Deterministic diffusion in one-dimensional chaotic dynamical systems, Ph.D. thesis, TU Berlin (1996).

[6] F.-R. Xiong, O. Schtze, Q. Ding, J.-Q. Sun, Finding zeros of nonlinear functions using the hybrid parallel cell mapping method, Communications in Nonlinear Science and Numerical Simulation 34 (2016) 23–37.

[7] G. Csernák, G. Stépán, Digital control as source of chaotic behavior, International Journal of Bifurcations and Chaos 5 (20) (2010) 1365–1378.

[8] G. Csernák, G. Gyebrószki, G. Stépán, Multi-baker map as a model of digital PD control, International Journal of Bifurcations and Chaos 26 (2) (2016) 1650023–1–11.

[9] H. E. Nusse, J. A. Yorke, Basins of attraction, Science 271 (1996) 1376–1380.

[10] J. Aguirre, R. L. Viana, M. A. F. Sanjun, Fractal structures in nonlinear dynamics, Rev. Mod. Phys 81 (2009) 333–386.

[11] H. E. Nusse, J. A. Yorke, Dynamics: Numerical Explorations, Vol. 101 of Applied Mathematical Sciences, Springer-Verlag, New York, 1998.

[12] J. A. W. van der Spek, Cell Mapping Methods: Modifications and extensions, Ph.D. thesis, Eindhoven University of Technology, Eindhoven (1994).
[13] G. Haller, G. Stépán, Micro-chaos in digital control, Journal of Nonlinear Science 6 (1996) 415–448.
[14] G. Csernák, G. Stépán, Sampling and round-off, as sources of chaos in PD-controlled systems, Proceedings of the 19th Mediterranean Conference on Control and Automation.
[15] G. Gyebrószki, G. Csernák, Methods for the quick analysis of micro-chaos, in: J. Awrejcewicz (Ed.), Applied Non-Linear Dynamical Systems, Springer International Publishing, 2014, Ch. 28, pp. 383–395.

6. Appendix A: Complexity of Simple Cell Mapping

Algorithm 8 Simple Cell Mapping

Input: Cell State space
Output: SCM solution — Number of execution, cost

1: $g \leftarrow 0$
2: **for** $z \leftarrow 0, n$ **do** — $n+1, 1$
3: **if** $state[z] =$ UNTOUCHED **then**
4: $processing \leftarrow True$
5: $sequence \leftarrow \emptyset \cup z$
6: $im \leftarrow z$
7: **while** *processing* **do** — $\sum_{z=0}^{n-1} t_z, 1$
8: **if** $state[im] =$ PROCESSED **then**
9: Tag cells in sequence as processed and transient — $\sum_{z=0}^{n-1} 1, s_z$
10: $processing \leftarrow False$
11: **else if** $state[im] =$ UNDER_PROCESSING **then**
12: ▷ New periodic group and possibily some transients found — $\sum_{z=0}^{n-1} 1, s_z$
13: Examine sequence, starting with im and tag cells as periodic, assign group g and step 0
14: Tag remaining cells as transient, assign $group \leftarrow g$ and calculate *step* numbers
15: $g \leftarrow g + 1$
16: $processing \leftarrow False$
17: **else**
18: ▷ $state[im] =$ UNTOUCHED, continue along the *image* track — $\sum_{z=0}^{n-1} t_z, 1$
19: $state[im] \leftarrow$ UNDER_PROCESSING
20: $sequence \leftarrow sequence \cup im$
21: $im \leftarrow image[im]$
22: **end if**
23: **end while**
24: **else**
25: ▷ Skip this cell
26: **end if**
27: **end for**

The number of times of execution and cost for some lines are denoted at line endings. The **for** loop is executed $n+1$ times, let t_z be the number of times the **while** loop is executed for that value of z. Let s_z be the length of the sequence accumulated starting with cell z.
By examining the algorithm, one can see, that $s_z = t_z$, since no branches of the **if-else** structure append new cell to the sequence or terminates the **while** loop at the same time. New cells are only appended to the sequence in line 20, while the processing of a sequence is either terminated at line 10 (reaching an already determined destination) or at line 16 (finding a new PG and transient cells). Therefore the cost of the algorithm is

$$C_{\mathrm{SCM}} = n\,c_1 + \sum_{z=0}^{n-1}(2\,s_z + t_z\,c_2) = (2 + c_1)\,n + \sum_{z=0}^{n-1} t_z\,c_2 = O(n),$$

where the sum of the length of sequences $\sum_{z=0}^{n-1} s_z = n$, c_1 is the total cost of constant-cost operations in the **for** loop outside the **while** loop, and c_2 is the total cost of constant-cost operations within the **while** loop.