Proceedings of the sixty-third European Study Group Mathematics with Industry

Enschede, The Netherlands, 28 January – 1 February, 2008

Editors:
Onno Bokhove
Johann Hurink
Gjerrit Meinsma
Chris Stolk
Michel Vellekoop

Preface

These proceedings contain the results that have been obtained during the Study Group Mathematics with Industry, which was held at the University of Twente in the Netherlands from January 28th until February 1st, 2008. During such Study Group weeks, mathematicians and other people with a strong interest in mathematics work on a several problems that have been formulated by industrial partners. The idea being that the participants try to solve these problems during one week with mathematical techniques.

In 2008 there were 78 participants from various countries, who together attacked six problems. These varied from problems on larger scales, such as the optimal arrangement of trains and large sensor networks, the modeling of aluminium alloys, and the drainage of rain water, to two problems on much smaller scales, one of which involved neurons in the brain and the other one mobile phone electronics. The reader will notice when studying the articles that substantial progress has been made on all problems and in some cases concrete solutions have already been proposed.

These scientific proceedings are accompanied by a separate booklet in Dutch, in which the results have been described for a wider non-scientific audience by journalist Bennie Mols.

It is a pleasure to take the opportunity to thank the sponsors of the Study Group for their generous donations. STW and NWO, our main sponsors, have been financing these events for many years and their continued support is very much appreciated. The research institutes CTIT and IMPACT, both part of the University of Twente, also contributed substantially. The CWI in Amsterdam has again sponsored the printing costs of these proceedings.

In closing, we thank all the participants for their outstanding contributions to an inspiring week of industrial mathematics.

The organization of SWI 2008,

Onno Bokhove,
Diana Dalenoord,
Dini Heres-Ticheler,
Johann Hurink,
Gjerrit Meinsma,
Marielle Plekenpol,
Chris Stolk,
Michel Vellekoop.

ISBN: 978-90-365-2779-8

The Study Group Mathematics with Industry 2008 and these proceedings were realized with financial support from:

Technology Foundation STW
The Netherlands Organization for Scientific Research (NWO)
Centrum Wiskunde & Informatica (CWI)
University of Twente (UT)
Institute of Mechanics, Processes and Control Twente (IMPACT)
Centre for Telematics and Information Technology (CTIT)
European Consortium for Mathematics in Industry (ECMI)

Contents

1 Shunting passenger trains: getting ready for departure 1

1.1 Introduction 1
1.2 First Approach: A Greedy Algorithm 7
1.3 Second Approach: A Dynamic Programming Algorithm 12
1.4 Alternative Approaches 17
1.5 Further research 18

2 Neural spike sorting with spatio-temporal features 21

2.1 Introduction 21
2.2 Spike classification 27
2.3 Regularity extraction 36
2.4 Concluding remarks 42

3 Math Fights Flooding 47

3.1 Introduction 47
3.2 A dynamical relation between precipitation and discharge 52
3.3 Rake model 55
3.4 Sensitivity analysis 59
3.5 Recommendations 61

4 Some studies on the deformation of the membrane in an RF MEMS switch 65

4.1 Introduction 66
4.2 Numerical Methods 70
4.3 The continuation problem 75
4.4 Analytical results 77

5 Increasing Detection Performance of Surveillance Sensor Networks 85

5.1 Introduction 85
5.2 Optimal coverage of the area 88
5.3 Statistical methods for intruder detection 94
5.4 Viterbi algorithm for intruder detection 102
5.5 Numerical results 107
5.6 Conclusions 111

6 Modeling and simulation of phase-transitions in multicomponent aluminum alloy casting 117
6.1 Introduction 118
6.2 Modeling transport and phase-transitions in multi-component aluminum casting 120
6.3 Thermodynamic representations and data reduction 125
6.4 Computational modeling of solidification fronts 134
6.5 Concluding remarks 136

1 Shunting passenger trains: getting ready for departure

Marjan van den Akker[1] Hilbrandt Baarsma[2] Johann Hurink[2*]
Maciej Modelski[3] Jacob Jan Paulus[2] Ingrid Reijnen[3]
Dan Roozemond[3] Jan Schreuder[2†]

Abstract

In this paper we consider the problem of shunting train units on a railway station. Train units arrive at and depart from the station according to a given train schedule and in between the units may have to be stored at the station. The assignment of arriving to departing train units (called matching) and the scheduling of the movements to realize this matching is called shunting. The goal is to realize the shunting using a minimal number of shunt movements.

For a restricted version of this problem an ILP approach has been presented in the literature. In this paper, we consider the general shunting problem and derive a greedy heuristic approach and an exact solution method based on dynamic programming. Both methods are flexible in the sense that they allow the incorporation of practical planning rules and may be extended to cover additional requirements from practice.

Keywords: shunting trains, greedy heuristic, dynamic programming

1.1 Introduction

In this paper we study a practical train shunting problem proposed by Dutch Railways. This problem has already been studied by Kroon et al. [7], but their work does not exploit the full potential of shunting trains.

Shunting of trains is a process that supports the execution of the train schedule at the station. Trains arrive at and depart from the station according to the train

[1]Utrecht University
[2]University of Twente
[3]Eindhoven University of Technology
*corresponding author, j.l.hurink@utwente.nl
†We thank Leo Kroon, Dutch Railways, for supplying this problem and his valuable insight and Daniël Roelfsema, Scar Groep, who also participated in the study group

schedule. Each arriving and departing train may consist of multiple and possibly different types of train units. This composition of the trains is specified in the train schedule. For an arriving train it now has to be decided what the next duties of the arriving train units are and for a departing train it has to be guaranteed that the corresponding train units are available on time on the platform. During rush hours almost all train units available are required to transport passengers and, thus, are on duty, but in between, and especially during the night, many train units are not needed for transporting passengers. Thus, train units may have to be parked at a shunt yard of a station for a certain period. An example of such a shunt yard is given in Figure 1.1, which represents the infrastructure of the station and shunt yard of Alkmaar.

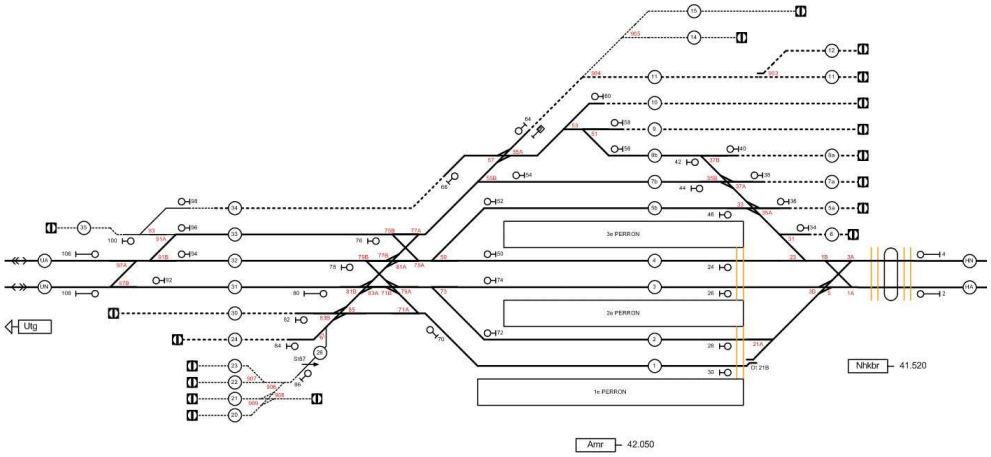


Figure 1.1: Shunt yard and station of Alkmaar.

The train units are classified according to their types and subtypes. Train units of the same type can be combined into longer trains, even if their subtypes differ. An example of a train unit is an ICM (Inter City Material) with 3 carriages, as shown in Figure 1.2. ICM denotes the type, and the subtype is specified by the number of carriages. There also exist ICMs with 4 carriages, which can be combined with the ICM with 3 carriages since they are of the same type, although not of the same subtype.

Figure 1.2: Train unit of type ICM with 3 carriages.

To park a train unit, a crew has to take several actions. If the train has to go only forward, the driver can stay on one side of the train and drive the train directly to the shunt yard. This is not always possible and it may e.g. be the case that the train

has to go forth, back, and forth again to be parked. In that case the engine driver has to switch places two times, since he always has to be in the front of the train. Each time going back or forth is called a shunt movement. So, if we only need to go forth, this is counted as one shunt movement and if we have to go forth, back, and forth then this is counted as three shunt movements.

When a train unit parked at the shunt yard is needed again in the schedule to transport passengers, it has to be taken out of the shunt yard and put at the platform from which the corresponding train will depart. Again it may happen that several shunt movements are needed to transport the unit to the platform. But it may even be worse in the sense that no train unit of the desired type is directly reachable on a shunt track. In this case, before getting the desired train unit, first some other blocking train units of another (sub)type have to be removed from a shunt track. This can also take several shunt movements.

As a consequence, to execute the train schedule, a feasible shunt schedule is required at each station. A shunt schedule consists of a list of actions that indicate which train units are shunted and between which places. Next to this, also the exact shunt movements of the train units can be specified. A shunt schedule is feasible if all arrivals and departures of the train schedule can be executed in the desired way. This implies for example that a platform has to be empty when a train is passing through or that train units of the desired (sub)types and in the desired order are at the right time at the right platform for a departing train.

However, not every feasible schedule is desirable: if the shunt schedule consists of many shunt movement, the schedule causes a high workload for the crew and is very sensitive for disruptions. This can cause delays in the train schedule, which should be avoided. Therefore, the goal is to have a shunt schedule with a minimal number of shunt movements.

Next to the main goal to create shunt schedules with a low number of shunt movements, some other practical aspects have influence on the quality of a schedule and lead to additional rules to be taken into account in creating shunt schedules. For example, for the crew it is convenient to have similar train units close together. This implies for instance that shunt tracks of the shunt yard should be used only for train units of the same type. Another practical aspect focuses on shunt movements just before a departure. Small disruptions in a shunt schedule with such movements directly may lead to delays of departing trains and, therefore, may disturb the train schedule. As a consequence, it is desirable that the number of shunt movements for a train that needs to depart is minimized. It would even be best if the train units are already waiting in the needed composition for the departure at the shunt yard.

1.1.1 Problem Description

The input for the shunting problem at a given railway station consists of the train schedule at that station and the layout of the station (platforms and shunt yard). The given train schedule prescribes the train arrivals and departures at the railway

station. Each of these events is characterized by a time, the composition of the train, its direction, required platform and whether the train arrives or departs. Since not all arriving train units are scheduled to leave the station immediately, the train units that stay behind may have to be stored at the shunt yard to clear the platform for the next train.

The shunt yard consists of a number of shunt tracks to store train units. Most of the shunt tracks are dead-end tracks. This implies that train units are blocked by train units parked at a later time. Thus, train units arrive and depart in *last in first out* (LIFO) order. The shunt tracks and platforms have a limited capacity for storing train units. There is a network of tracks connecting the shunt tracks with the platform tracks.

Between successive events of the train schedule, it may be necessary to move train units to make the next event possible. Such movements are called shunt movements. A one-directional movement is counted as one movement and every change of direction is counted as an extra movement. A solution is a list of shunt movements that take place between the events such that all events can take place. The objective is to find a solution with minimum number of shunt movements.

In this paper we assume a timeless model; i.e. we assume that a shunt movement takes zero time. This implies that an unlimited number of shunt movements can be performed between two events. However, it is possible to add extra constraints within the developed methods, which restrict the number of shunt movements between two events.

1.1.2 An Example

To illustrate the shunting problem we give a small example. Consider a railway station with the layout given in Figure 1.3. In this example we consider four types

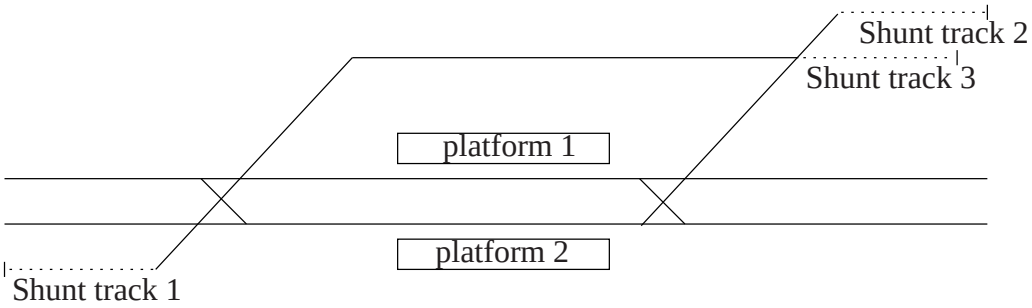


Figure 1.3: Layout of the example station.

of train units, denoted by A, B, C and D. Each train consist of some train units of these types. When we talk about a train AB we mean a train consisting of a train unit of type A and a train unit of type B, where the type A unit is positioned to the left of the type B unit. This is regardless of the direction the train is traveling in. Thus,

trains AB and BA are different in composition. We assume that the capacity of all shunt tracks and platform tracks is limited to accommodate 3 train units. According to the train schedule the following trains are arriving and departing in the given order.

e_1	Train AB arrives from the left-side at platform 1.
e_2	Train AA arrives from the right-side at platform 2.
e_3	Train CCC arrives from the right-side at platform 1.
e_4	Train CC departs from the platform 1 to the left-side.
e_5	Train AA departs from the platform 2 to the right-side.
e_6	Train DC arrives from the left-side at platform 1.
e_7	Train CDC departs from platform 1 to the right-side.
e_8	Train BA arrives from the right-side at platform 2.
e_9	Train BB departs from platform 2 to the right-side.
e_{10}	Train AA departs from platform 1 to the left-side.

In this small example there are already a number of non-trivial shunting decisions to make. It is not difficult to verify that the following solution is a valid shunt schedule.

Between e_2 and e_3	Shunt train AB from platform 1 to shunt track 2.
Between e_5 and e_6	Shunt train C from platform 1 to shunt track 1.
Between e_6 and e_7	Shunt train C from shunt track 1 to platform 1.
Between e_8 and e_9	Shunt train A from platform 2 to shunt track 2, and shunt train AA from shunt track 2 to platform 1, and shunt train B from shunt track 2 to platform 2.

The solution contains six shunt movements. In this example the choice whether to shunt to the tracks on the left-hand side or to the tracks on the right-hand side is the most important decision. Observe that shunting train unit C to any of the shunt tracks on the right-hand side is not a good decision. When moving the unit back, it has to go around the DC train, to connect to it from the left to form the CDC train. Going around the DC train implies a change of direction in the shunt movement and is counted as two shunt movements. Furthermore, if the AB train is shunted to the shunt track on the left-hand side, the efficient moves between e_8 and e_9 would not be possible. It turns out that the above solution is indeed optimal for the example.

1.1.3 Complexity of the Shunting Problem

The general problem of integrated matching (to which departing trains are the units of an arriving train matched?) and parking of train units is introduced in [7] and in [8] its computational complexity is determined. The general problem as well as the subproblem of matching the train units and the subproblem of parking the train units are shown to be NP-hard.

In the train matching problem we are given a set of arriving trains and a set of departing trains. We are supposed to partition the incoming trains into parts which can later be assembled into departing trains. Since each produced part is shunted separately, our main goal is to minimize the number of parts into which we partition the arriving trains. This problem is a generalization of the minimum common string partition problem known from computational biology. In [5] the minimum common string partition problem is shown to be NP-hard even if we restrict ourselves to instances with only two strings as input. This means that the train matching problem is NP-hard even if we are given just one arriving and one departing train.

Blasum et al. [1] introduce a problem of scheduling the departures of trams from a shunt yard in the morning. This problem turns out to be NP-hard and the authors provide a dynamic program for a special case of the problem with restricted number of shunt tracks. This problem can be seen as a subproblem of our shunting problem where all the trains are already placed in the shunting yard.

Cornelsen et al. [2] study the problem of generating shunt-free schedules in stations consisting of parallel two-sided tracks. They reduce the problem to a graph coloring problem of a conflict graph resulting from the train schedule. For most of the versions of the problem the conflict graph is perfect and can be colored in polynomial time. For other cases efficient approximations algorithms are presented.

In similar setting Dahlhaus et al. [3] consider a problem of grouping of train units. In this problem a sequence of incoming train units is given. Each train has to be sent to one of a given set of parallel tracks and later pulled out to the other side. The outgoing sequence has to be ordered in such a way that units of the same type are grouped together. Designing a schedule that minimizes the number of used tracks is shown to be NP-hard.

In freight train classification hump yards are commonly used for shunting. Jacob et al. [6] model the shunting task as a problem of finding a set of binary codes. It allows them to find optimal solutions for most versions of the problem. Some other versions are shown to be NP-hard.

1.1.4 Current Solution Approach

Currently there is no decision support system to aid the personnel in solving the shunting problem. However, Dutch Railways is testing the ILP-model proposed in [7] on small stations. However, this ILP-model has a number of drawbacks. First of all it does not cover all possible shunting moves. For example it does not allow trains to stay at a platform, waiting to be combined with a next train. It is clear that such a waiting possibility can be beneficial. Moreover, it does not model the possibility of moving train units between different shunt tracks. Whenever a train arrives, it either has to be shunted away or depart immediately.

Furthermore, in the current ILP-model the number of variables and constraints is already very large, and extending this model to cover the above shunting possibilities would increase the number of constraints and variables even further. Although

for a typical instance the current model can be solved within a reasonable time, one may expect that the extensions make the number of variables so large that the computation time required to solve the problem becomes unacceptably large.

1.1.5 Goal of the Research

The task of this paper is to present alternative approaches to the shunting problem which do allow waiting on platforms and rearrangements of train units between shunt tracks. In the following we describe two solution approaches, one which aims at finding fast a reasonable cost solution (a greedy algorithm) and one which aims at the optimal solution (a dynamic-programming algorithm). We conclude with an outlook on future research.

1.2 First Approach: A Greedy Algorithm

In this section, we present a heuristic approach for the shunting problem. This heuristic has to be fast and has to result in a reasonably good solution. The basic idea is to scan the event list and iteratively decide which actions to take. The decisions in each iteration are based on the situation resulting from the previous decisions and the current event. In this way, the approach tries to locally extend the given situation as good as possible and, therefore, falls in the category of *greedy* approaches.

From practice it is indicated that planners prefer situations where the train units of departing trains are already waiting somewhere (either on a platform track or a shunt track) in the composition they have to depart in. We take this philosophy, *be ready for departure*, as a guideline for building the greedy approach. As a consequence, we scan the event list backwards in time and make the decisions in such a way that they lead to the desired composition for the departing trains.

For the presentation, we assume that during the planning horizon the arriving train units correspond one to one to the departing trains. We assume that the train station is empty at the beginning and the end of the planning horizon. This can be justified by taking the planning horizon to be form one rush hour to the next, since during rush hour all material is needed in the train schedule. The presented heuristic can easily be adopted when this assumption is dropped.

Our algorithm consists of two main steps, step 2(a) and 2(b), which we explain in more detail later.

The Greedy Algorithm:

1. Start with empty platforms and shunt tracks
2. Scan the event list backwards in time, and for each event **DO**
    a) **IF** the event is a departure event, **THEN** assign the entire train to a shunt track

Events	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8
Arrivals	C	A	BB			AA		
Departures				B	BA		AA	C
Platform	2	1	2	2	1	2	1	2

Table 1.1: Event list Example 1.

b) **ELSE** the event is an arrival event **THEN** match the train units to train units already placed on the shunt track

The most important steps in our algorithm are steps 2(a) and 2(b). In these steps the main decisions are made. In step 2(a) we decide on which shunt track we set the train ready for departure. At this point, we do not care how these train units come to this shunt track, but just decide that the units wait on the assigned shunt track for departure. How these units arrive on their positions on the shunt track will be decided in subsequent iterations. Possible rules for assigning the trains to shunt tracks are given later. In step 2(b) we match the train units of arriving trains to train units that are already placed for departure from a certain track in one of the previous iterations. Again, concrete rules for this matching are given later.

Example 1 To get a better understanding of the basic idea of this greedy approach we present an example consisting of two platforms and two shunt tracks. The event list of this example is presented in Table 1.1 (in this table platform numbers are given as well, since we use them later on).

If we scan the event list from the back, we first have to treat event e_8. Since this is a departure, we may decide to assign this train to shunt track 1. The situation on the two shunt tracks after this decision is given in Figure 1.4(a). The next event e_7 is also a departure, and we may assign the train AA to shunt track 2 (see Figure 1.4(b)). For shunting the arriving train units of event e_6 we now have the nice option to match the whole train to the two train units of type A being assigned to shunt track 2. By this matching, i.e. shunting the two train units to shunt track 2, this shunt track gets empty and the resulting situation is as in Figure 1.4(c). Next, we may assign the train of departure event e_5 to shunt track 2 and the train of departure event e_4 to shunt track 1 resulting in the situation as in Figure 1.4(d). If we now treat the arriving event e_3, the train consisting of two type B units cannot be matched as a whole to a shunt track, but we have to split the train and match the two type B units to the two type B units in front of the two shunt tracks leading to the situation in Figure 1.4(e). Note that this matching leads to two separate shunt movements. Finally, the two arriving events e_2 and e_1 are processed by matching the corresponding train units to the units of the same type still being on the shunt tracks.

As can be seen from the above example, the presented algorithm decides for each arriving train unit to which departing unit it is coupled and via which shunt track this assignment takes place. In this way shunt movements are specified. For

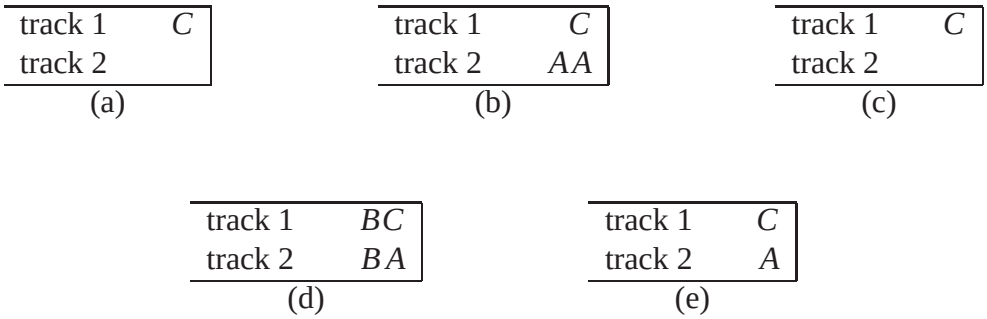


Figure 1.4: Situations on the shunt track for Example 1.

departing trains the shunt movement can be done with the train as a whole since we always assign to be 'ready for departure'. We implicitly assume that there are shunt tracks long enough to accommodate for departing train. For arriving trains more complex shunt movements may be necessary. In the above example, all the shunt movements were directly possible, but in general it may be necessary to rearrange the train units on the shunt tracks at certain moments to achieve a feasible solution. If, for instance, the arriving event e_3 would have been that of an AC train, first the two B units already being at the shunt tracks would have to be removed to place the A and C unit at the dead-end of the shunt tracks.

The advantage of the presented approach is that it always gives a feasible solution as long as the list of arrival and departing events is consistent, in the sense that there is never a negative stock of train units of some type, there are shunt tracks long enough to accommodate for the departing trains, and there is always some empty (part of) track to move a train unit around. Furthermore, the departing trains can always be handled efficiently. The price to achieve this is that we may create costly shunt movements for arriving events.

In the following we sketch some possible improvements of the greedy method and give some more detailed information on possible implementations of the assignment and matching in steps 2(a) and 2(b).

Leaving train units on platform tracks One of the goals of this research is to develop methods which allow the option of leaving train units on platform tracks or to move it from one platform to another platform without parking it in between at the shunt yard. A simple approach is to scan the solution achieved by the greedy heuristic and to search for 'shortcuts'. In the above example such a short cut is for example possible between the events e_6 and e_7. The AA train arriving on platform 2 (event e_6) may be passed directly to platform 1 from which it departs as event e_7. In this way, the AA train does not have to be moved to the shunt track 2, probably saving shunt movements. Another short cut is possible by leaving one of the arriving type B units of event e_3 on platform 2. In this way the departing train of event e_4 is already on the platform without any movement.

A more effective method than a scan after finishing the greedy approach may be to take such possibilities already into account during the greedy algorithm. If we have to assign a departing event in step 2(a) of the algorithm, we may scan the event list some positions further back in time to detect if there is an assignment of this train to a shunt track which allows using shortcuts. Such an assignment is preferable over other assignments.

Delaying the shunt movement If for an arriving event the shunt movements of possible matchings take a large effort (e.g. the corresponding units do not occur at a reachable end of a shunt track), we may scan the event list back in time to see if we can improve the situation by letting some other arriving trains wait on their platform. To clarify this possibility, let us assume that in the given example the train of event e_2 is a B train and that of e_3 an AB train. If we now deal with event e_3, no easy matching is possible since on shunt track 2 the train units are not in the correct order (see Figure 1.4(d)). But we may delay the movements belonging to event e_2, since this event is on a different platform. For the greedy approach this means that we consider event e_2 before e_3. By matching the B unit of that train to the B unit in front of shunt track 2, we achieve a situation where on shunt track 1 we have BC and on shunt track 2 we have A. Now we can match the two units in front of the two shunt tracks to form the AB train of event e_3.

Formally, in step 2(b) of the greedy approach we may search the event list backwards and consider for each platform the first occurring event. If this event is an arrival, we may treat this event before the current event. Note that it is not possible to delay departure events or two arriving events on the same platform.

Assignment rules in step 2(a) Up to now, we have not specified the way how we assign in step 2(a) the trains to shunt tracks. The most simple way is to assign them in some *round robin* way (meaning that the tracks are used in a given cyclic order) or to assign them based on some priorities of the tracks. Possible priorities may be: smallest number of shunt movements to reach the platform, largest free capacity, et cetera. However, it may be worthwhile to incorporate also planning rules of the planners of Dutch Railway into this step. One such rule is, for example: do not park more than two different unit types on the same shunt track. Furthermore, a backward scan in the event list by a few positions may help to overcome problems in the next iterations. Consider, for example, the event list in Table 1.2. Two possible shunt track assignments after treating the events e_6, e_5, e_4 are given in Figure 1.5. The first assignment is made using round robin, but has not taken into account the arriving B train. The second assignment does not have this problem.

Matching rules in step 2(b) As in step 2(a), also in step 2(b) there may be some freedom in matching the arriving trains to units already assigned to the shunt tracks. Again, this decision may be based on priority rules like the number of necessary

Events	e_1	e_2	e_3	e_4	e_5	e_6
Arrivals	A	A	B			
Departures				A	A	B

Table 1.2: Event list Example 2.

track 1	AB
track 2	A

track 1	AA
track 2	B

Figure 1.5: Situations on the shunt track for Example 2.

shunt movements but, as in the previous case, it may also be worthwhile to incorporate some backward scan to see which resulting remaining situation on the shunt tracks forms the better situation for the next events. To illustrate this consider Example 2.

Example 2 This example is the same as Example 1, with the difference that events e_2 and e_3 are interchanged and that after considering event e_4 we have the first shunt track assignment in Figure 1.5. If we now have to deal with event e_3, matching the type A unit of this event to the A unit in front of shunt track 1 allows a direct access to the B unit on that track in the next iteration. Having chosen for the A unit on shunt track 2 would not have given this possibility leading to a situation where units on the shunt tracks have to be rearranged.

Improvements Several of the suggested improvements contain some sort of partial backward scan of the event list to improve the decision for the current event. In principle this means that some sort of simultaneous treatment of several events is considered. Based on the outcome of this, a decision for the current event is fixed. This treatment of several events simultaneously, can be seen as a new optimization problem on its own. This problem gets harder the more events are taken into account. An interesting topic of further research is to try to find a good balance between the effort spent on this backward scan and the improvement in quality. Furthermore, concrete decision rules for the treatment of several events simultaneously have to be developed.

To sum things up: the greedy algorithm we have developed is able to create feasible schedules for the shunting problem quite fast. However, without additional improvements, the achieved solution may not be of much practical use. Above we have shown, that the basic structure of the method forms a good framework which easily can be extended by more sophisticated elements and even with rules used by planners.

1.3 Second Approach: A Dynamic Programming Algorithm

To solve the shunting problem, a response to each event, arrival or departure, has to be given. This response has some influence on the position of train units on the different tracks and platforms and has to guarantee, that the next event can take place. *Getting ready for a departure* means that the right train composition is on the departure platform, and *getting ready for an arrival* means that the arrival platform can accommodate for the arriving train.

To describe the given situation of train units on the different shunt tracks and platform tracks (called a configuration), we define a vector S. S is called the state of the system and is an ordered list of train type units on each of the tracks. For the example given in Section 1.1.2, the first element in S describes the train units on platform 1, the second on platform 2, the third on shunt track 1, et cetera. With (S, e_i) we indicate that the train units are in state S just before event e_i happens. The pair (S, e_i) is valid if and only if event e_i can take place with the given state S; i.e. in state S we are ready for event e_i.

With this notation we can describe the solution for the example of Section 1.1.2 as in Table 1.3.

$$\begin{pmatrix} - \\ - \\ - \\ - \\ - \end{pmatrix}, e_1 \overset{c=0}{\rightarrow} \quad \begin{pmatrix} AB \\ - \\ - \\ - \\ - \end{pmatrix}, e_2 \overset{c=1}{\rightarrow} \quad \begin{pmatrix} - \\ AA \\ - \\ AB \\ - \end{pmatrix}, e_3 \overset{c=0}{\rightarrow} \quad \begin{pmatrix} CCC \\ AA \\ - \\ AB \\ - \end{pmatrix}, e_4 \overset{c=0}{\rightarrow} \quad \begin{pmatrix} C \\ AA \\ - \\ AB \\ - \end{pmatrix}, e_5 \overset{c=1}{\rightarrow}$$

$$\begin{pmatrix} - \\ - \\ C \\ AB \\ - \end{pmatrix}, e_6 \overset{c=1}{\rightarrow} \quad \begin{pmatrix} CDC \\ - \\ - \\ AB \\ - \end{pmatrix}, e_7 \overset{c=0}{\rightarrow} \quad \begin{pmatrix} - \\ - \\ - \\ AB \\ - \end{pmatrix}, e_8 \overset{c=3}{\rightarrow} \quad \begin{pmatrix} AA \\ BB \\ - \\ - \\ - \end{pmatrix}, e_9 \overset{c=0}{\rightarrow} \quad \begin{pmatrix} AA \\ - \\ - \\ - \\ - \end{pmatrix}, e_{10}$$

Table 1.3: Solution of the example.

1.3.1 Network of (S, e_i)-pairs

The basic idea behind the dynamic programming algorithm is the following. From the initial state and the first event we can determine all possible responses which are compatible with the second event. In this way a set of new pairs (S, e_2) are created which are then treated recursively in the same way. For a formal description, let each pair (S, e_i) be a node and let each transition (set of shunt movements) leading to a following node be an arc. This way we get a network in which we can move from one pair (S, e_i) to an other pair (S', e_{i+1}). In this network we only allow valid pairs, and each transition has an associated cost equivalent to the number of

shunt moves required for carrying out the transition. It is not difficult to see that the shunting problem is equivalent to finding a shortest path in this network.

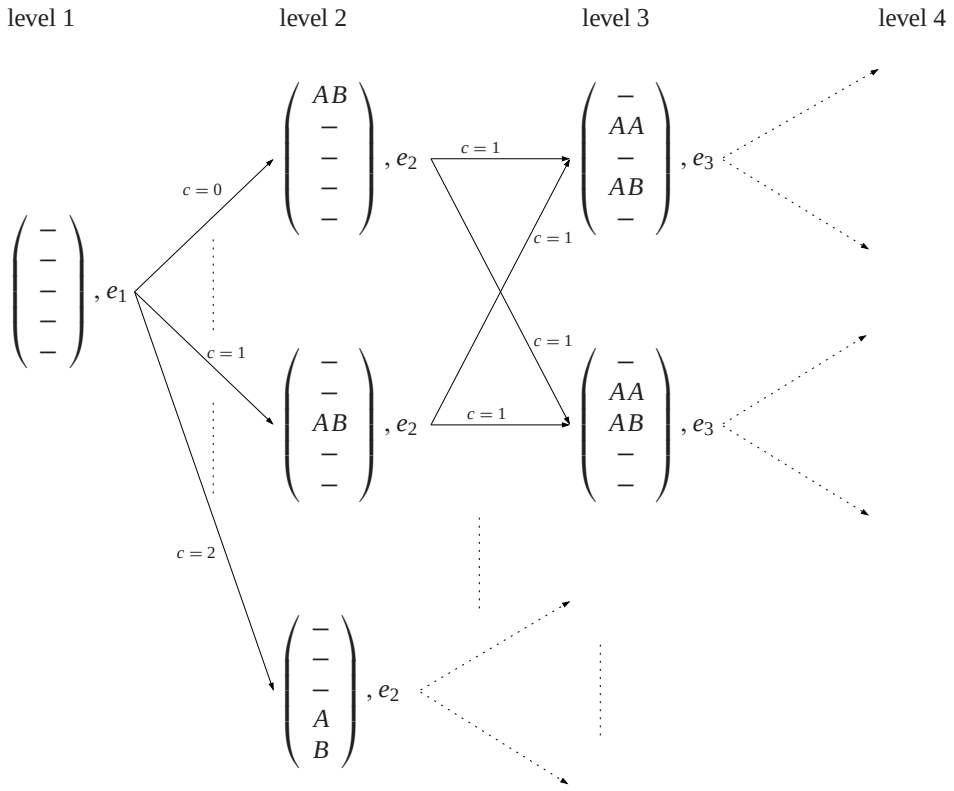


Figure 1.6: Dynamic programming network.

Although the network becomes very large, the network is highly structured. The network consists of a number of levels, where each level corresponds to one event, see Figure 1.6. Hence, there are only arcs going from level i to level $i + 1$. This means that the cost of getting to a particular state is given by the cost of the states in the previous level plus the cost associated with the arcs.

To obtain the optimal solution, we just have to construct the network level by level and calculate the cost of getting in each of its nodes. However, though this would work in theory, in practice the running time of this algorithm may explode as the instances get larger (remember that the problem is NP-hard).

1.3.2 Eliminating Nodes

To make the dynamic programming approach work in practice we need to bring down the size of the dynamic programming network. In this section we present several suggestions to speed up the dynamic program algorithm. However, by applying some of these suggestions we can no longer guarantee that the optimal solution is found.

Eliminate symmetry Whenever there are two tracks with the same characteristics (same capacity and reachable with the same number of shunt movements from the platforms), there are many nodes in the network that are basically the same. In the example given earlier we have not used shunt track 3. If all the units assigned to track 2 are assigned to shunt track 3, we have a different solution which is essentially the same. So, in the network we can delete many states which are symmetric without affecting the solution.

Disallow costly transitions Given a transition with a high number of shunt movements, one might not want to allow this transition from a practical point of view. We can incorporate this, by simply deleting the arcs corresponding to these costly transitions from the network. This may reduce the number of outgoing arcs from nodes and may even lead to nodes which are not reachable anymore, reducing the number of nodes in the network. Note, that disallowing costly transitions may exclude the optimal solution.

Upper bounding the solution For each node in the dynamic programming network we know the cost of getting to this node. If by some (heuristic) procedure we know that there exists a solution with cost c, we do not have to proceed with nodes in the network that have cost exceeding c, i.e. these nodes can be deleted from the network. Reducing the dynamic programming network in this way does not affect the optimal solution.

Detecting bad paths Suppose we have created the dynamic programming network up to level i. If we now compare the cost of all nodes in level i, we may expect that the costly ones have only a small chance to result in the overall optimal solution. Deciding not to continue from the nodes with high costs reduces the dynamic programming network. However, this may exclude the optimal solution.

Rolling horizon To make a decision for level 1, we may restrict ourselves to creating the dynamic programming network only up to level i. Based on the information up to level i we may decide which arc to take leaving level 1. Starting with the resulting node on level 2, we now may create the network up to level $i+1$ and use this network to decide upon the level 2, et cetera. This type of decision making is called *rolling horizon*. Each time we make a decision, only a small part of the network is considered. Again, we may exclude the optimal solution.

1.3.3 Computational Results

We have made a proof-of-concept implementation of the dynamic programming approach in C++, comprising about 1000 lines of code. The example of Section 1.1.2

is used to test both the implementation and some of the elimination rules. The results are summarized in Table 1.4.

In this table,

- **clm** indicates the maximum allowed cost between each level,
- **sym** indicates whether or not symmetry elimination is used,
- **ntp** indicates whether or not states, in which more than 2 types of train units are on the same shunt track, are forbidden,
- **tp** indicates whether or not states, in which unit of types A/B and C/D are mixed, are forbidden,
- **#states** gives the number of states on each level in the network. In most cases, only state counts up to level 4 are given, as the runtime increases dramatically after that,
- **runtime** gives the runtime for those computations that we ran to completion (the running times are after various optimizations of the code, on a 2.16GHz laptop),
- **cost** gives the resulting costs for those computations that we ran to completion.

The number of valid states does not tell the entire story, though. The number of *intermediate* states, i.e. those states that have to be computed and may or may not be valid, has a large impact on the runtime as well. In case I, each of the 128 states in level 2 generates about 25000 new states, of which in total only about 1500 are valid. This is quite a large number compared to e.g. case 4A, where the number 25000 is already reduced to about 3700.

The impact of limiting the costs between levels in the network is clear: If we do not enforce any limits, the network is simply too large to compute. If we limit to 4, we can complete the computation, but if we limit to 3 the speedup is almost a factor of 5 without losing the optimal solution. Limiting the costs of the arcs to 2 removes the optimal solution, but could provide a good heuristic for upper bounding the solution (see Section 1.3.2).

The other elimination rules also cut down the number of states significantly, although not as dramatically as limiting the costs of arcs.

One of the major advantages of this approach is that adding new rules (e.g. heuristics used by Dutch Railways planners) is extremely easy: in our implementation it is literally a matter of minutes. Furthermore, the chosen DP-approach is very suitable for parallelization.

id	Elimination rules clm	sym	ntp	tp	#states	run-time	cost
I.	∞	-	-	-	$28 \rightarrow 128 \overset{\sim 25000}{\rightarrow} \sim 1500 \rightarrow \ldots$		
4A.	4	-	-	-	$28 \rightarrow 128 \overset{\sim 3710}{\rightarrow} 1500 \rightarrow 180 \rightarrow \ldots \rightarrow 14 \rightarrow 1$	737s	6
4B.	4	y	-	-	$19 \rightarrow 72 \overset{\sim 2410}{\rightarrow} 780 \rightarrow 108 \rightarrow \ldots \rightarrow 10 \rightarrow 1$	280s	6
4C.	4	y	y	-	$19 \rightarrow 72 \overset{\sim 2410}{\rightarrow} 630 \rightarrow 90 \rightarrow \ldots \rightarrow 10 \rightarrow 1$	242s	6
4D.	4	y	y	y	$19 \rightarrow 72 \overset{\sim 2410}{\rightarrow} 178 \rightarrow 40 \rightarrow \ldots \rightarrow 10 \rightarrow 1$	153s	6
3A.	3	-	-	-	$28 \rightarrow 128 \overset{\sim 870}{\rightarrow} 1500 \rightarrow 180 \rightarrow \ldots \rightarrow 14 \rightarrow 1$	146s	6
3B.	3	y	-	-	$19 \rightarrow 71 \overset{\sim 630}{\rightarrow} 776 \rightarrow 108 \rightarrow \ldots \rightarrow 10 \rightarrow 1$	63s	6
3C.	3	y	y	-	$19 \rightarrow 71 \overset{\sim 630}{\rightarrow} 628 \rightarrow 90 \rightarrow \ldots \rightarrow 10 \rightarrow 1$	54s	6
3D.	3	y	y	y	$19 \rightarrow 71 \overset{\sim 630}{\rightarrow} 178 \rightarrow 40 \rightarrow \ldots \rightarrow 10 \rightarrow 1$	24s	6
2A.	2	-	-	-	$19 \rightarrow 121 \overset{\sim 140}{\rightarrow} 1196 \rightarrow 180 \rightarrow \ldots \rightarrow 12 \rightarrow 1$	18s	7
2D.	2	y	y	y	$13 \rightarrow 64 \overset{\sim 120}{\rightarrow} 166 \rightarrow 40 \rightarrow \ldots \rightarrow 8 \rightarrow 1$	3s	7

Table 1.4: Dynamic Programming Results.

1.4 Alternative Approaches

Besides the presented Greedy Algorithm and Dynamic Programming Algorithm, other solution approaches may be possible. In this section we give some comments on such approaches.

1.4.1 Local Search

One might expect that a local search approach is useful to obtain good solutions, since each solution is a list of shunt movements compatible with the event list. However, we feel that defining small local operations on this list which result in new compatible lists of shunt movements, is extremely difficult. When a small change is made in the movement list, many repair operations may be required to keep the list compatible with the event list. Consider the example given in Section 1.1.2, where between events e_2 and e_3 train AB is shunted to track 2. Suppose we modify this first shunt movement by moving AB to shunt track 1 instead of moving it to shunt track 2. This small change makes the remainder of the list incompatible with the events, i.e. the shunt movement between events e_8 and e_9 cannot be performed. This example shows that changing a single movement is not just a local change, it requires repair operations that can be much further down the list. Furthermore, it seems to be difficult to calculate the resulting change in the objective value in a simple way since we know nothing about the amount of repair operations. This convinces us that a local search approach may be not an easy way to go.

1.4.2 Integer linear programming

A possible approach is to extend the model from [7] by other shunt moves. For example, to include the possibility to wait at the platform and delay shunting, we need to include the 'shunting time' explicitly. The current model includes a variable z_{js} which equals 1 if train unit j is parked at or retrieved from tracks s. We could replace these variables by z_{jst} signaling if train unit j is parked at or retrieved from tracks s at time t. Another possibility is to add variables t_j representing the shunting time of train unit j. Although the number of reasonable shunting times for a train unit is limited, both options significantly complicate the model: the first by strongly increasing the number of variables and the second by the need for additional 'nasty' constraints. The computation time will probably increase accordingly.

A different LP-based approach is to apply *column generation*. In [4] a column generation algorithm for the planning of aircraft at gates or platform stands at Amsterdam Airport Schiphol is presented. Because of the similarity with the problem of planning train units on a shunt yard, i.e., shunt tracks correspond to gates at an airport, the idea seems useful to explore. The idea is that the problem is decomposed into two levels. At the highest 'master' level we have variables representing a complete shunting plan for one shunt track and the most important constraint is that the

retrieval of each departing train unit and the parking of each arriving train unit is included in exactly one shunting plan. At the detailed or subproblem level we determine feasible shunt plans for one track which are expected to be beneficial for the optimization at the master level. Column generation approaches have been successful to solve large optimization problems in many different applications. However, certain shunt moves such as rearrangements of trains between different shunt tracks seem to be quite complicated to include in the model and therefore we are not convinced that it is worth to investigate this approach further.

1.5 Further research

In this paper we have presented two approaches for shunting train units. The first one is a greedy algorithm that can find a feasible shunt plan quickly. This algorithm typically chooses one single possibility that looks best at the current moment in time. The second one is a dynamic programming algorithm that can find the optimal shunt plan and typically explores many possible states. We presented an outline and a basic version of the algorithms and developed a preliminary prototype of the dynamic programming algorithm.

Each of the algorithms can be improved by moving more towards the other approach. The greedy algorithm can be improved by including smart look-ahead rules and rules used by operational planners. The dynamic programming can be improved by rules to prune non-promising states and in this way make the set of states that have to be explored smaller. To have the best of both worlds, the two algorithms can also be combined. For example, a state within the dynamic program can be extended to a complete feasible solution by the greedy algorithm. This solution can then be used as an upper bound to prune non-promising states. Investigating these possible improvements is a topic of future research.

Bibliography

[1] U. Blasum, M.R. Bussieck, W. Hochstättler, C. Moll, H.-H. Scheel, and T. Winter. Scheduling trams in the morning. *Mathematical Methods of Operations Research*, 49(1):137–148, 1999.

[2] S. Cornelsen and G. Di Stefano. Track assignment. *Journal of Discrete Algorithms*, 5(2):250–261, 2007.

[3] E. Dahlhaus, P. Horak, M. Miller, and J. F. Ryan. The train marshalling problem. *Discrete Applied Mathematics*, 103(1-3):41–54, 2000.

[4] G. Diepen, J.M. van den Akker, J.A. Hoogeveen, and J.W. Smeltink. Using column generation for gate planning at Amsterdam Airport Schiphol. *Techni-*

cal report UU-CS-2007-018, Department of Information and Computing Sciences, Utrecht University, submitted to Transportation Science, 2007.

[5] A. Goldstein, P. Kolman, and J. Zheng. Minimum common string partition problem: Hardness and approximations. In *ISAAC*, pages 484–495, 2004.

[6] R. Jacob, P. Marton, J. Maue, and M. Nunkesser. Multistage methods for freight train classification. In *Proceedings of the 7th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems (ATMOS)*, 2007.

[7] L.G. Kroon, R.M. Lentink, and A. Schrijver. Shunting of passenger train units: an integrated approach. *Erasmus University Rotterdam, ERIM Report Series 2006-12-20*, 2006.

[8] R.M. Lentink. *Algorithmic Decision Support for Shunt Planning*. PhD thesis, Erasmus University Rotterdam, The Netherlands, 2006.

2 Neural spike sorting with spatio-temporal features

Claude Archer[1] Michiel Hochstenbach[2] Kees Hoede[3]
Gjerrit Meinsma[3*] Hil Meijer[3] Albert Ali Salah[4]
Chris Stolk[3,5] Tomasz Swist[6] Joanna Zyprych[7†]

Abstract

The paper analyses signals that have been measured by brain probes during surgery. First background noise is removed from the signals. The remaining signals are a superposition of spike trains which are subsequently assigned to different families. For this two techniques are used: classic PCA and code vectors. Both techniques confirm that amplitude is the distinguishing feature of spikes. Finally the presence of various types of periodicity in spike trains are examined using correlation and the interval shift histogram. The results allow the development of a visual aid for surgeons.

Keywords: spike sorting, deep brain stimulation, PCA, interspike interval histogram

2.1 Introduction

The problem addressed in this study involves helping a neurosurgeon get his or her bearings during deep brain surgery. A stereotactic frame is used to fix a patient's head during an operation, and simultaneously to provide a coordinate system for the surgeon to navigate. The region to be operated is determined by imaging techniques

[1]Ecole Royale Militaire (MECA), Brussels
[2]Eindhoven University of Technology
[3]University of Twente
[4]CWI, Amsterdam
[5]University of Amsterdam
[6]CERGE-EI, Prague
[7]Agricultural University of Poznan
*corresponding author, g.meinsma@utwente.nl
†Other participants: Marta Dworczynska (Wroclaw University of Technology), Marcel Lourens (University of Twente), Tomasz Olejniczak (Wroclaw University of Technology)

prior to the surgery. For some tasks, like taking out a tumor, the resolution of the image is good enough for the operation. For finer tasks, however, the structural anatomy of the brain is less relevant than the functional anatomy. An example of the latter is deep brain stimulation (DBS), which requires a high resolution to determine the location at which to stimulate.

One method to determine the functional anatomy is to insert fine needles into the brain to record neuron action potentials during the surgery. This can indicate whether the targeted area is reached or not. However, this task is very difficult, and requires a lot of expertise. The medical group we are working with uses the following approach. Several micro-needles (10 micron thick, multiple needles about 2 millimeters apart) are inserted into the operating region. The neural activity is recorded for periods of 10 seconds, converted to sound waves, and played to the surgeon, who then decides whether the needle is on target or not. If not, the surgeon moves the needle some 0.5mm and the procedure is repeated.

Our aim in this project is to determine which methods of analysis and information presentation would help the surgeon to classify the recorded neural activity in real time. Moreover we would like to incorporate the knowledge of the expert surgeon into the analysis in a way that helps inexperienced surgeons, particularly as expert knowledge is highly qualitative, depends on intuition honed by many surgeries and is very difficult to state as a procedural description.

Apart from the difficulty of modeling expert knowledge, there are several other challenges in this problem. When a needle is recording neural activity, it records a great deal of background noise too, which needs to be accounted for. Deep brain recordings have much higher noise levels than cortical recordings. Depending on the proximity of neurons in the area, several neural activities can be recorded with a single needle, and the fact that closely spaced neurons usually have highly correlated activities makes their separation difficult. A single neuron can have relatively regular interspike intervals, or it can alternate periods of low activity and high-frequency firing. Furthermore, neurons can go active or inactive during a single recording, and the number of neurons contributing to the signal may change. The recording time is typically short, which makes temporal classification via statistical methods difficult, if not impossible. On the other hand, classification via the spike shape is not trivial either.

2.1.1 The data and problem details

The basic object of study are voltage traces $x(t, L)$ with L the level of insertion and t the time. Possible levels are $L \in \{0, 50, 100, \ldots, 500\}\mu$ m and the time ranges over precisely 10 seconds, $t \in [0, 10]$. Available for analysis are sampled

$$x_k := x(kT_s, L)$$

at a sampling frequency of

$$f_s = 1/T_s = 20\text{kHz}.$$

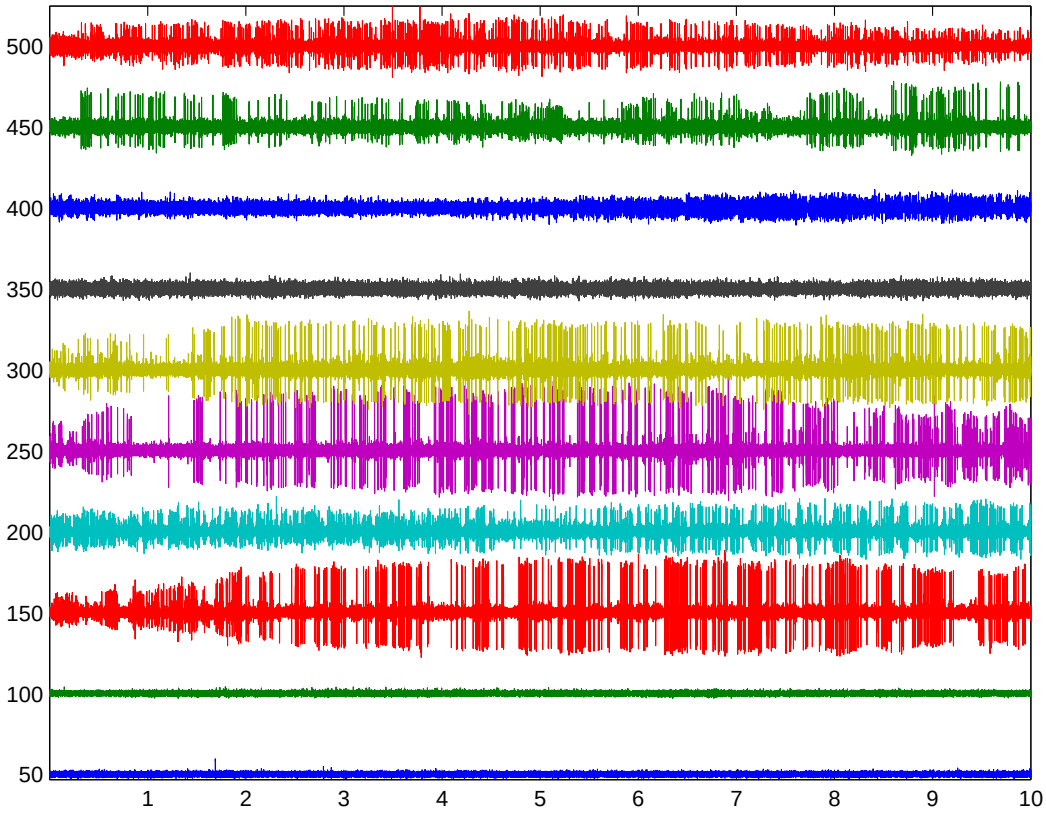
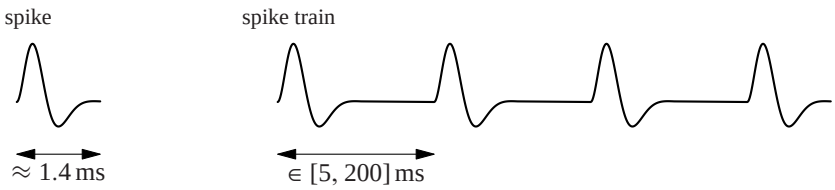


Figure 2.1: Traces $x(t, L)$ for levels $L = 50, 100, \ldots, 500\,\mu\text{m}$ and time $t \in [0, 10]\,\text{s}$.

This means that frequencies up to 10kHz can in principle be captured by the discrete measurements x_k. Note that from now on the level L is suppressed in our notation x_k. We will analyze voltage traces only for a given fixed level. Powerline artifacts and similar disturbances are assumed to have been removed from x_k. Figure 2.1 shows a typical set of traces for various levels L. Its behavior changes per level but also within each level the signal characteristics may change over time. We assume that signals are stationary within 1 second. At most levels in Figure 2.1 peaks are clearly visible, which suggests that significant signal power is attributed to these peaks. A quick scan however shows that the power due to the peaks is negligible and also in the frequency domain the power due to the peaks turns out to be not clearly separated from that of background noise, i.e. their respective spectra overlap significantly. Inspection of Figure 2.1 suggests that background noise can be removed in the time domain using a threshold. This is explained in Section 2.2, where we follow the approach given in [10].

The basic waveform, and repeated waveform, respectively known as *spike* and *spike train* can be depicted as follows:



Given the sampling frequency of 20kHz this means that a single spike covers at least 20 samples. Spikes with a large amplitude stand out in Figure 2.1. Surgeons distinguish three types of spike trains:

1. spike trains of *regular* firing rate. These originate from neurons that fire at a rate of 5Hz to 50Hz;

2. spike trains of *regular-HF* firing rate. These originate from neurons that fire at a rate of 50Hz to 150Hz;

3. spike train *bursts*. These originate from neurons with firing rates around 100Hz with the main feature that pockets of activity are interlaced with pockets of inactivity. The amplitude of spikes may vary within a burst.

This is a coarse classification and irregular firing patterns and many other types may be present as well. For instance a neuron can stop firing for some time or change its amplitude. There are many other sources of non-stationarity. One source is due to the movement of the neurons with respect to the needle. Another is the dynamics of the neuron itself. For example, when a needle advances, it can stun the nearby tissue, so that the neuron stops firing completely or at least temporarily alters its firing behavior, before turning back to normal behavior. Detecting time windows of near stationarity is crucial and this is why the analysis has to take place for every window of, about, 1 second.

The problem is to automate what the surgeon does and to do so in real time, with a delay of at most 5 seconds. In short, we want to:

1. pinpoint the location of spikes (i.e. remove background noise),
2. separate the set of spikes trains into various classes (corresponding to different neurons),
3. determine for each of the classes of spike trains to which of the three types they belong (if any),
4. visualize the findings.

Problems 2 and 3 combined are known as the problem of *spike sorting*. In the rest of the paper we describe a set of ideas that could be useful in solving these problems in real time. A color-coded visualization as exemplified in Figure 2.2 is a possible desired outcome of the project, as it would help the surgeon to decide on the nature of neuronal activity in the measured area.

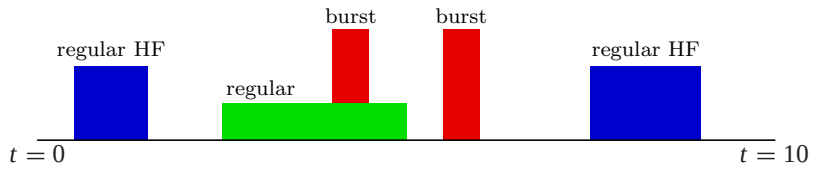


Figure 2.2: Visualizing the presence of regular spike trains (green), regular-HF spike trains (blue) and spike train bursts (red) as a function of time.

2.1.2 Literature survey

Spike sorting has been around since the 1960s. The earlier methods relied on template matching, and required heavy offline processing [14]. More recent methods combine feature extraction, probabilistic modeling, and clustering. The accuracy and efficiency of these methods are much greater than before, but most of them are still too computationally intensive to be used during the surgery, and they do not work well with deep brain recordings. An excellent recent review of the problem is the one by Lewicki [6].

The success of spike sorting methods is determined by simulations on artificial data (for which the correct classification is known) or by comparisons to human-annotated real recordings. Harris *et al.* studied the performance of a human operator when sorting spikes recorded from a tetrode (4-wire electrode) manually, and decided that human operators sort the spikes suboptimally [5]. Single-needle recordings (as we study in this work) were markedly more difficult to classify than tetrode recordings, where the presence of multiple sensors provides robustness in the decisions. Their conclusion was that “automatic spike-sorting algorithms have

the potential to significantly lower error rates." Similar observations were made in [17], which reports average error rates of 23% false positive and 30% false negative for humans sorting synthetic data. In artificially created data sets, this type of error is reduced. Consequently many researchers create artificial data sets by modifying a small set of annotated signals, adding noise and superposing them to make the problem more difficult [1, 2, 10, 18], or by resampling from the distribution that characterizes the data [17]. Generation of realistic data is another issue. In [8], a cortical network simulation based on GENESIS was used to generate artificial spike data. The authors note that the spike sorting algorithms tested on their simulated data failed. More recently, Smith and Mtetwa proposed a biophysical model for the transfer of electrical signals from neural spikes to an electrode to generate realistic spike trains for benchmarking purposes [15].

Assuming that the procedure to validate a proposed spike sorting method is adequate, the first phase is usually filtering to remove artifacts and noise. The recordings are influenced by the ambient signals, interference from nearby electronic devices, vibrations caused by movement and noise from other neurons firing in the vicinity. The amplitude of the signal is a good indicator of a neural spike, and is frequently used to determine spike occurrences. It is necessary to select static or adaptive thresholds for this purpose. Once a threshold is selected, activity below the threshold is considered to be noise. To eliminate noise on the selected spikes, a smoothing procedure can be applied. In [3] the signal is resampled with a cubic spline interpolation for a better alignment of the spike shape with its peak amplitude. (Section 2.2 of our paper describes an efficient alternative approach.) In [13] spikes are detected by looking at threshold crossings of a local energy measurement of the bandpass of a filtered signal, which is shown to be more reliable than the raw signal.

Once the spikes are extracted, they can be classified by their shape characteristics, temporal characteristics, or both. For temporal characteristics, the interspike interval distribution and its correlation-based analysis can reveal different spike firing patterns [11]. But these methods ignore the spike shape. For shape-based characterization, the spike shapes are normalized by their maximum amplitude, cropped, and treated like shape vectors. The two approaches that are frequently used are clustering to get the mean shapes for spikes, or matching against a pre-specified set of templates. The difficulty in the clustering approach lies in the fact that the number of clusters is usually unknown. One method proposes to start with a large number of clusters, and to combine clusters that are sufficiently close, until a stopping criterion is reached [3]. This resembles the method proposed by Figueiredo and Jain for determining the complexity of a Gaussian mixture model automatically [4]. In this approach, the number of clusters in the mixture is not specified prior to model learning, but determined on the fly. The algorithm is initialized with n clusters, and during each step of the algorithm the smallest cluster is combined with another cluster, and the expectation-maximization (EM) algorithm is run until convergence. Each step ends with one component less than the previous step, until only a single

component remains. Then, all the intermediate steps are evaluated by a minimum description length criterion to select one model as the final output of the system. In [3], instead of generating all possible models, a statistical test is employed to stop the combination procedure.

Both template matching and clustering methods face the potential problem that spikes do not have fixed amplitude and shapes. During the recording, movements of the electrode or a change in the membrane potential can cause a change in the spike amplitude and shape [6]. Similarly, Quiroga *et al.* remark that when the spike features deviate from normality, most unsupervised clustering methods will face difficulties [10].

In [16], several spike characteristics were contrasted to see which features lead to a better classification. The parameters of the waveform (i.e. amplitude, spike width, peak-to-zero-crossing time, peak-to-peak time) were found to be insufficient for effective discrimination. The authors also contrasted optimal filtering techniques [12], template matching (with root-mean squares error criterion), and principal components analysis (PCA)-based techniques. Their results show that even though it is possible to obtain good results with the costly template matching method, PCA-based approaches were much more robust against higher noise levels. The overlap of waveforms was found to be greatly impairing the accuracy of template-based methods. A possible solution to this problem was proposed in [18], where PCA and clustering techniques are combined to test incrementally whether a single source or multiple sources contribute to the signal. Recently, Pavlov *et al.* contrasted wavelet and PCA-based methods, and argued that wavelet-based methods could perform better than PCA, yet they need to be carefully tuned for this purpose [9].

For real-time applications, even the PCA-based methods may be too computationally intensive. In [19] a front-end hardware architecture is described for spike sorting, but the system is tested on a 'clean' sample for which PCA achieves 100% accuracy. Still, the proposed algorithm can achieve good results with much less computation steps.

2.2 Spike classification

In this section we formulate ways to separate dominant spikes from background noise and subsequently try to split the many spikes into classes that correspond to individual neurons, or at least to neurons with similar firing behavior.

2.2.1 Detection, double spike removal and time shifting

Consider a noisy trace x_k, such as in Figure 2.1. If the value x_k of the signal is above a certain threshold, it is assumed to belong to a spike. The paper [10] describes how to choose the threshold using the standard deviation σ_n of the noise. Under the assumption of being normally distributed (and the background noise indeed appears

to be so) the standard deviation equals

$$\sigma_{\mathrm{n}} = \frac{1}{0.6745}\,\mathrm{median}(|x_1|, \ldots, |x_N|).$$

The usual formula using an average of squares is not used, because then the extremes due to the spikes would affect σ_{n} too much. The threshold is given by a constant α_{thr} times σ_{n},

$$V_{\mathrm{thr}} = \alpha_{\mathrm{thr}}\,\sigma_{\mathrm{n}},$$

with $\alpha_{\mathrm{thr}} = 4$ or 5, or a number in between, the choice of which appears to be somewhat subjective as different values were found in the literature.

Each spike will lead to a small interval of values above the threshold. To have a simple criterion, we take *maxima in the signals whose value is above the threshold*, which define a set of points $t_{\mathrm{p},j}$ (p for 'peak'). This is our initial set of 'raw' spike times[8]. We crop a temporal window that contains the spike, starting 0.4 ms before the peak and ending after 1.2 ms, resulting in a 1.6 ms data window. These form our 'raw' set of spike traces. An example of such a raw set is displayed in Figure 2.3. In this example 674 spikes where found in 10 seconds of data.

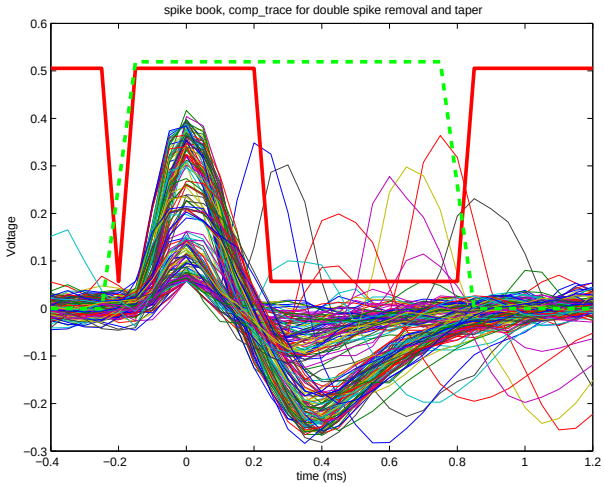


Figure 2.3: 'Raw' spikes, cropped and aligned by their peaks at time zero. Also displayed are the functions w_{d}, used for identifying double spikes (thick solid line), and the taper function (thick dashed line), which we use to select only the part of interest for each spike. (Every fourth spike plotted.)

The transformation from the no-activity state (signal within noise level) to the peaked activity is very fast, comprising about 0.15 ms, which means that with our

[8]The coding was done in MATLAB, and the experiments were conducted on a set of traces that were available from patient measurements

sampling rate, three samples can be acquired for the spike before it peaks. After the peak some of the spikes continue for up to about 25 samples (1.25 ms), although for the shape analysis the first 20 samples seem to be sufficient.

There are several potential problems at this stage:

- *Double detection*: A single spike could be mistaken for two individual spikes due to noise, say within two or three sample points. A possible strategy to deal with this is to consider the largest of two close peaks to be the real peak, and to ignore the other. For the limited set of sample traces that we worked with, this problem did not occur.

- *Overlapping spikes*: It is possible that a second spike occurs shortly after or before a spike. It can be seen in the figure that this happens in our data. These are outliers for the purpose of spike shape analysis, as a single neuron cannot fire again in such a small period, and we should therefore remove them.

  To remove double spikes, we use two threshold areas around the peak, one containing samples [-0.2ms, 0.2ms] around the peak (about 9 samples) and the second from [0.25ms, 0.8ms] after the peak (about 11 samples). Values above the threshold (depicted with a thick solid line in Figure 2.3) indicate the presence of a double spike. Obviously, it remains to be investigated whether the parameter settings we use are suitable for other measurements, i.e. on larger collections of recordings. But a visual inspection of Figure 2.3 and a plot of the rejected spikes can be used to assess reliability of the result. In our data set 24 of the 674 spikes were rejected as double spikes.

  We use a taper function to limit the interval around the peak, and the subsequent smoothing of the signal depends on the choice of the taper function. This can be important when interpolation is applied later in the process. The taper function we have used had a width of 0.1 ms to keep tapering to a minimum, and to prevent lossy smoothing. A scaled version of the taper function is plotted as the thick dashed line in Figure 2.3. The spikes that are thus excluded from the analysis and the remaining valid spikes are plotted in Figure 2.4.

- *Negative polarity spikes*: Spikes with negative polarity were ignored.

The next step would be to apply *time shift corrections* to the spike traces, to align them better. Spikes can have a time shift that is a fraction of the sampling period, so interpolation becomes necessary to apply such time shifts. In a Scholarpedia paper, it is proposed to interpolate the spikes at a finer resolution and then align them by their maxima. To keep keep the subsequent computational complexity low we developed an alternative approach. Each spike $f_j(t)$, $j = 1, \ldots, N$ is time shifted

[8] www.scholarpedia.org/article/Spike_sorting

over a time β_j. Now the vector $\beta = (\beta_1, \ldots, \beta_N)$ is chosen that *maximizes the total correlation* of the traces, given by

$$\int dt \Big|\sum_j f_j(t - \beta_j)\Big|^2, \quad \text{with constraint} \sum_j \beta_j = 0.$$

Fourier interpolation was used, so that the interpolation and optimization can both be done in the Fourier domain, using off-the-shelf interpolation algorithms. Computation time in MATLAB takes about 0.5 second for 640 spikes on a regular machine, which indicates that an optimized code will have acceptable temporal complexity.

A comparison of Figures 2.3 and 2.4 shows that time shifting leads to much higher similarity between the spikes. In the next section, we will show that time shifting is also beneficial for PCA-analysis. Optimal time shifting results in much better clustering behavior, with tighter clusters, and occasionally with better separation, resulting in more clusters.

To summarize, we have implemented the necessary codes for the following purposes:

1. Detection of maxima above the threshold.
2. Removing double spikes.
3. Tapering the remaining spikes.
4. Time shift corrections in order to maximize total correlation.

These steps give an adequate pre-processing for the subsequent shape analysis, see Figure 2.4(b), and our method of computation of time shift corrections makes the overall procedure efficient.

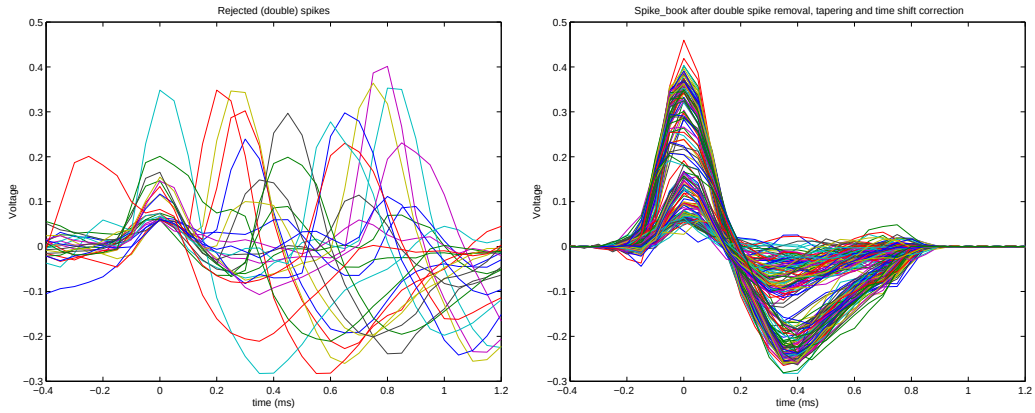
2.2.2 Principal Component Analysis (PCA)

The Principal Component Analysis (PCA) is a popular tool that is used in numerous scientific, medical, and engineering applications such as noise reduction in signal processing and face recognition. Here we will use the PCA to recognize and analyze the different types of spikes.

Let $\tilde{A} \in \mathbb{R}^{m \times n}$ be the wide matrix containing the spike data as columns,

$$\tilde{A}_{ij} = \text{sample } i \text{ of spike } j, \qquad i \in \{1, \ldots, m\},\ j \in \{1, \ldots, n\}.$$

Here n is the number of spikes found in the signal (for instance, $n \approx 650$ in the previous subsection), and m is the number of samples per spike, typically $m \approx 20$. Although it is no real restriction, for convenience we will assume in the following that $n > m$; in practice n may be much larger than m.



(a) (b)

Figure 2.4: (a) Double spikes removed from the set of spikes; (b) spikes after removal of double spikes, tapering and time shift correction (every fourth spike plotted in part (b)).

The PCA is based on the Singular Value Decomposition (SVD); often, the SVD and PCA are used as synonyms. However, in PCA the SVD is applied to the matrix A obtained from $\tilde{A}$ by subtracting from each trace (column) the mean of that trace

$$A_{ij} = \tilde{A}_{ij} - \frac{1}{m}\sum_{k=1}^{m} \tilde{A}_{kj}.$$

The SVD of a matrix is a decomposition of the form

$$A = U\Sigma V^T$$

with $U^TU = I$, $VV^T = I$ and Σ a diagnonal matrix with nonnegative, non-increasing entries, $\sigma_1 \geq \sigma_2 \geq \cdots$ on its diagonal (The ^{T} denotes transpose.) There are two forms of an SVD: a full and a reduced SVD. In the full SVD, both U and V are square matrices. For almost all applications the data contained in the full SVD are superfluous and it is much more efficient to use the reduced SVD, in which U is still square, size $m \times m$, with Σ now size $m \times m$ as well, and V has size $m \times n$.

The columns $u_1, u_2, \ldots, u_m$ of U are the *left singular vectors* or *principal components* and give information on the patterns that are present in the collection of spike data. Their corresponding singular values $\sigma_1, \sigma_2, \ldots, \sigma_m$ indicate how strong the respective patterns are. By construction the patterns $u_1, u_2, \ldots, u_m$ are orthogonal; they do not represent spikes except u_1.

We compute the PC's of the spike collection and show the main results in the figures below. In Figure 2.5 we plot the first two singular values against each other for all spikes in a single trace x_k. This kind of plot is useful to find clusterings of spike shapes in the trace, i.e. groups of spikes with similar shapes. In this case three clusters can be observed. This was exceptional, most of the traces had only

two clusters, one consisting of large spikes, and the other of the remaining spikes. Some had no clear clustering. In Figure 2.6 we plot the mean of the traces (the thick dashed line), and the first four principal components, the thickest being the first, and the thinnest the fourth.

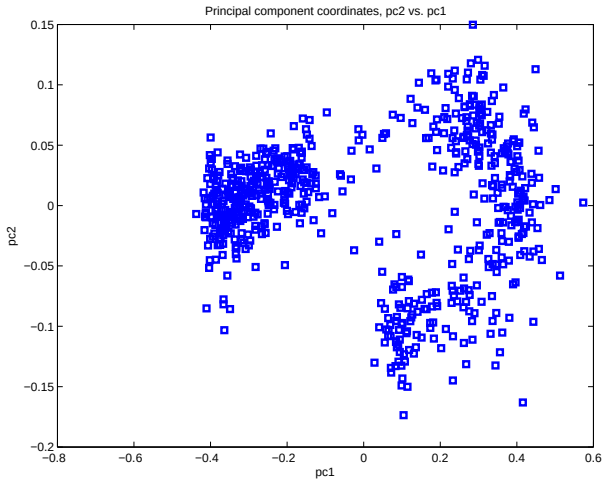


Figure 2.5: The first two singular values from PCA analysis plotted against each other. Three clusters can be observed.

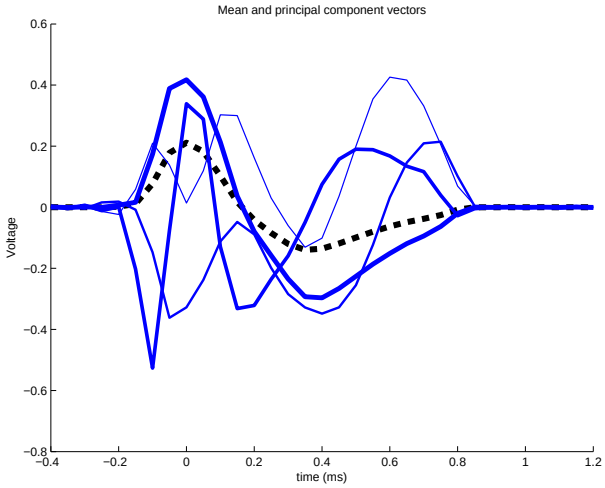


Figure 2.6: The mean (dashed black line), and first 4 principal component vectors, the first corresponding to the thickest solid line.

Since in Figure 2.5 σ_2 is much smaller than σ_1, this figure suggests that there is one quite dominant spike pattern. Indeed, the distinguishing feature is the size of the spikes. Of course, this outcome is influenced by the removal of the outliers (the second spike in a sequence of two consecutive spikes) in the previous subsection. In signals where many spikes with negative polarity are present, we expect a much

larger σ_2 corresponding to a pattern u_2. In Figure 2.7 we plot the largest singular value against time. This picture shows that the presence of several clusters is related to a change in observed spike shapes that occurs around $t = 8000$ms, and thus reveals even more structure in the data.

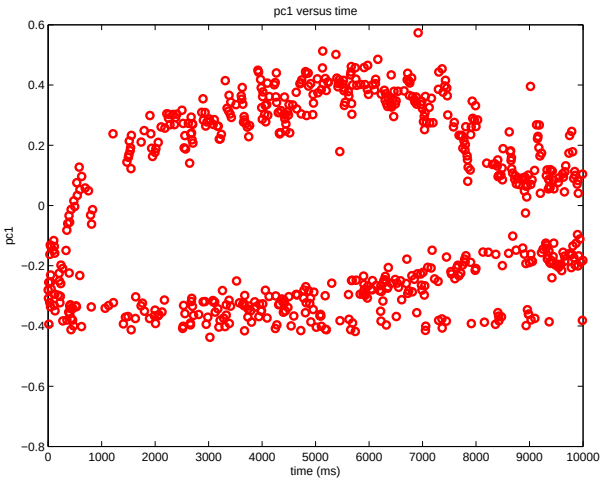


Figure 2.7: The largest singular value from PCA plotted against time (in milisecconds). The clustering can also be observed in this picture.

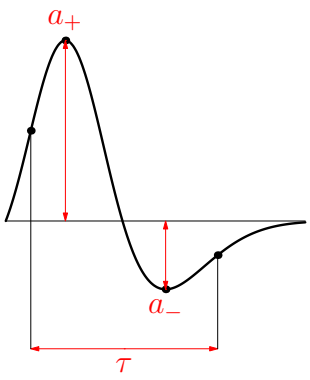


Figure 2.8: Coding spike features.

2.2.3 Coding

Another technique to classify spikes is to represent any distinguishing feature by a number on a scale and combine these numbers to create a *code vector*. There are several features that can be defined:

- A spike has a *top* value a^+. As the amplitude depends on how close the probe is to the neuron, it should be normed e.g. by considering $a_n = a/a_{max}$ where a_{max} is the maximum amplitude occurring during a measurement.

- A spike also has a *bottom* value a^- (taken positive). Now the total *amplitude* $b = a^+ - a^-$ can be considered as a feature, scaled as $b/\max(b)$.
- The *polarity* p depends on the temporal order of a^+ and a^-. It is positive, $p = 1$, if a_+ is attained before a_-, and negative $p = -1$ otherwise.
- The *width* w can be defined as the time difference between the time τ_1 when the signal reaches half peak value $a^+/2$ and the time τ_2 when it first exceeds $a^-/2$ after the occurrence of a^- for a spike with positive polarity. For a spike with negative polarity the width can be defined as the width of minus the signal.

These features are illustrated in Figure 2.8. The idea of coding is now as follows. After normalization, a_n takes a value in the interval $[0, 1]$. This value could be taken as the encoding of the amplitude, but the interval may also be divided into some, say three, equal parts that can be encoded by 0 (if $a_n \in [0, \frac{1}{3})$), 1 (if $a_n \in [\frac{1}{3}, \frac{2}{3})$), and 2 (if $a_n \in [\frac{2}{3}, 1]$). The amplitude is thus encoded on a 3-point scale: "low", "medium" and "high". In a similar way the width, polarity and amplitude of a spike can be encoded on either a 2-point or a 3-point scale. With these four features we have $3 \times 2 \times 2 \times 2 = 36$ different code vectors

$$(a_n, b, p, w) \in \{0, 1, 2\} \times \{0, 1\} \times \{-1, 1\} \times \{0, 1\}.$$

Some other features were also suggested:

- Similar to total amplitude, the *relative height* $h_{\text{rel}} = |\frac{a^+}{a^-}|$ can be defined and may be encoded by a 2-point scale, 0 if $h_{\text{rel}} \geq 1$ and 1 if $h_{\text{rel}} < 1$.
- The slope at the second halftime τ_2, as there are some neurons which can show an afterhyperpolarization, i.e. a prolonged negative phase.
- Different types of neurons may show spikes that differ in the *regeneration quotient* of the two time intervals between start and passage of zero respectively passage of zero and the end. So for "width" there are various ways to define "start" and "end".

As we have seen in the former subsection it seems doubtful that many essentially different types of spikes occur. This is confirmed by this alternative classification method. In fact encoding only amplitude, polarity and relative height, leads to only 12 different code vectors, from $(0, 0, -1)$ to $(2, 1, 1)$. Figure 2.9 shows four histogram of two traces, one at level $L = 200$ and one at level $L = 50$. First, we see that a_n and b within a single trace encode more or less the same feature. A fast majority of spikes have positive polarity, and manual inspection of spikes with negative polarity led to the conclusion that there was in fact another cause for an early negative peak to be present. The few spikes with negative polarity we did find

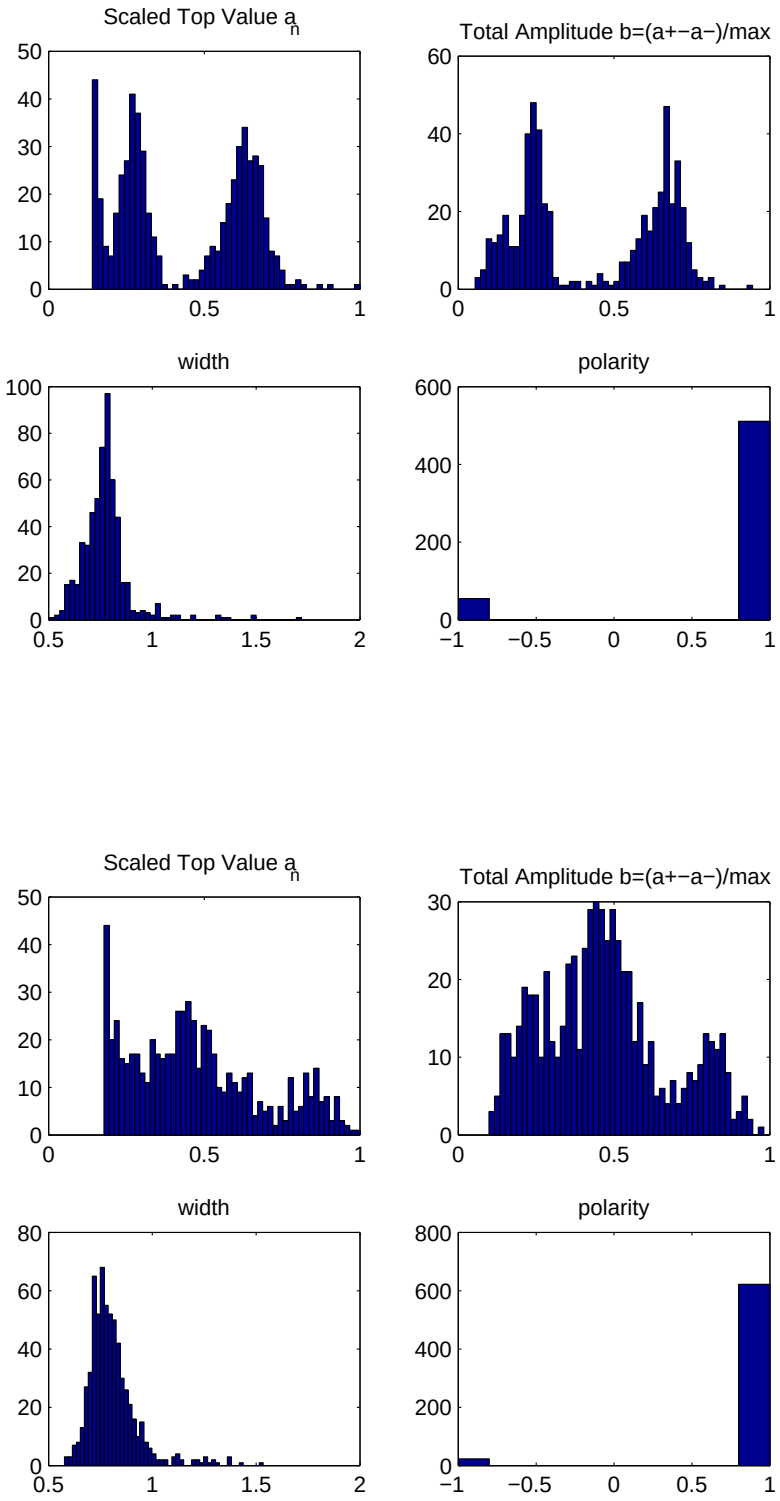


Figure 2.9: Histogram for four coding features for two traces x_k: (top four) trace at level $L = 50$; (bottom four) trace at level $L = 200$.

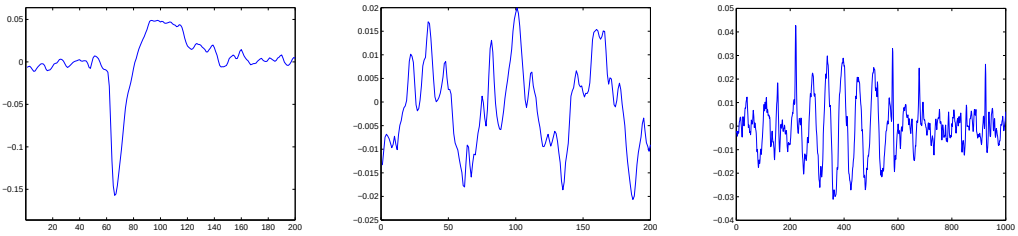


Figure 2.10: Data that cause problems when defining features. Top: negative polarity; middle: several spikes after each other; bottom: burst. The peaks are above threshold.

could be due to a dying cell. So polarity does not distinguish spikes. Neither does the width. Moreover, the "half"-times τ_1 and τ_2 did not always exist in case several spikes occurred shortly after each other or during a burst, see Figure 2.10.

These computations show that the neurons can be distinguished using just the maximum a_+. Only a few code vectors are relevant, i.e. correspond to occuring types of spikes. This is in agreement with the PCA results.

2.3 Regularity extraction

Now we assume that background noise in a trace has been removed and that the remaining spikes in x_k are classified (separated) into a collection of a few different spikes, each with its own characteristics. In this section we continue with the analysis of a *single* spike train. By definition then any spike in a spike train shares the same features, hence we need only specify the time instances at which the spikes occur (e.g. where the maximum of the spikes occur). We use s_k to denote such a spike train time series. That is, $s_k = 1$ if a spike occurs at discrete time index k, and $s_k = 0$ if no spike occurs at k. The repeating firing patterns of neurons induce periodicities in the spike train s_k and we should now try to pinpoint what type of firing pattern is present in s_k: a regular firing rate, a regular-HF firing rate or a burst, and possibly a superposition of the above.

2.3.1 Autocorrelation and Fourier Analysis

Classically periodicities are determined by correlation $r_k := \sum_i s_{i+k} s_i$ and the discrete Fourier transform (DFT). A distinct advantage of both correlation and DFT is that computation is very efficient: for a trace of n samples it takes $\mathcal{O}(n \log_2(n))$ opertions to compute correlations and the DFT. Fourier and equivalent autocorrelation analyses are fairly robust with respect to small variations in the periodicity of the spikes. A more severe problem occurs when the spike train is a *superposition* of periodic signals (and noise). Figure 2.11(a) demonstrates this problem: while the signal s_k clearly is a superposition of two purely periodic signals—with period

5 and 8—the autocorrelation analysis does not clearly pinpoint the periodicities of the involved signals, and does not help in separating them.

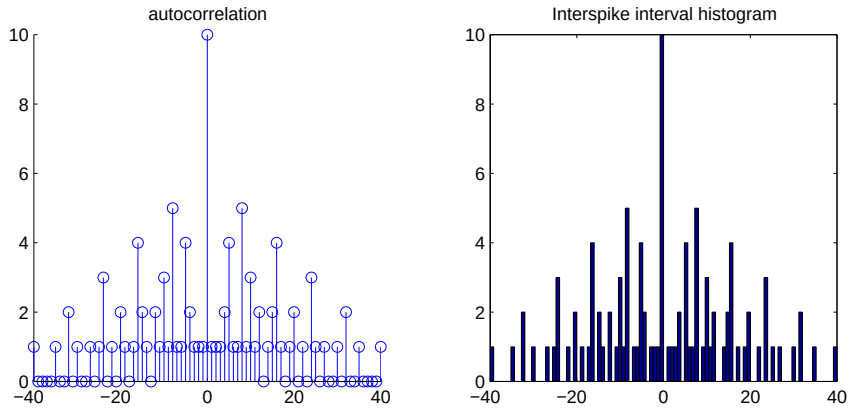


Figure 2.11: Autocorrelation (left) and interspike interval histogram (right) of spike train s_k with spikes at $t = (0, 5, 8, 10, 15, 16, 20, 24, 32, 40)$.

While autocorrelation and DFT consider a spike train as a function s_k of time k, it is more efficient for computational purposes to store spike trains as sequences $t = (t_1, t_2, \ldots)$ of time instances at which spikes occur. For instance the spike train

$$s \quad = \quad \underset{k=0}{\overset{}{\mid}} \cdot \cdot \cdot \underset{k=4}{\overset{}{\mid}} \cdot \cdot \cdot \cdot \cdot \underset{k=10}{\overset{}{\mid}} \cdot \cdot \underset{k=15}{\overset{}{\mid}} \cdot \cdot \; k \rightarrow$$

can be stored more efficiently as the sequence $t = (0, 4, 10, 15)$. The analysis of time sequences t is considered next.

2.3.2 Interspike interval histogram

Several mathematical techniques are known for discovering regularity in time sequences, with autocorrelation, discussed in the former subsection, being one of them. The method that we will describe in this subsection is related to autocorrelation, but turns out to be appropriate for determining the beginning and end point of periods of regular firing of neurons, even when there are pockets of inactivity between windows of regular activity. The idea will be introduced for strictly regular sequences. Let us consider a regular time sequence with *period* 5,

$$t = (0, 5, 10, 15, 20, 25, 30).$$

The regularity with period 5 is discovered simply by looking at the consecutive time differences, which indeed are all equal to 5. Now suppose the data is contaminated with time instances at 8, 16 and 18, so

$$t = (0, 5, \underline{8}, 10, 15, \underline{16}, \underline{18}, 20, 25, 30).$$

The period 5 is now masked. Considering consecutive differences now gives rise to new "periods" $8-5=3$, $10-8=2$ and $16-15=1$, $18-16=2$, $20-18=2$. The idea now is that by comparing not only neighboring time differences, but also other possible time differences, we can recover the dominant difference, which is 5 in this case. In fact, addition of the series of neighboring differences will produce, among others, in our case $3+2=5$ and adding up once again produces $1+2+2=5$. Considering *all* differences between pairs of time instances will result in a histogram in which the period 5, as well as multiples of 5 dominate. If there are m time instances, then $\binom{m}{2} = \frac{1}{2}m(m-1)$ differences are to be calculated.

The resulting histogram is called the *Interspike Interval Histogram*, or IIH for short [11]. The IIH procedure can be visualized as follows: for all $t_k \in t$ the sequence t is first shifted by $-t_k$ (effectively shifting its kth element to zero) and the resulting sets of shifted $t-t_k$ are then added up, see Figure 2.12. As we count the differences to obtain the histogram, it might also be called a *Difference Histogram* but we stick the literature standard of IIH.

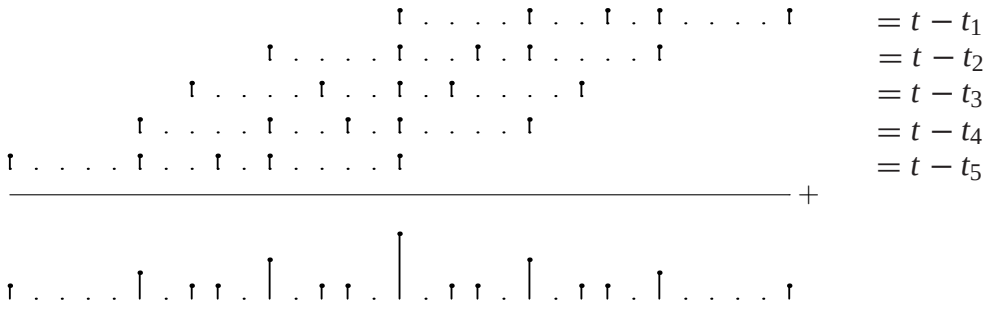


Figure 2.12: Visualization of the construction of the IIH.

To illustrate the procedure differently we superimpose a random set of times on our example sequence. Say we have

$$t = (0, 5, 8, 10, 14, 15, 16, 18, 20, 25, 27, 28, 30). \tag{2.1}$$

The consecutive differences form the sequence

$$(t_2 - t_1, t_3 - t_2, \ldots) = (5, 3, 2, 4, 1, 1, 2, 2, 5, 2, 1, 2).$$

In this sequence the difference 2 occurs five times while difference 5 occurs only twice. Adding two consecutive differences leads to the sequence

$$(8, 5, 6, 5, 2, 3, 4, 7, 7, 3, 3).$$

Adding three consecutive differences leads to the sequence

$$(10, 9, 7, 6, 4, 5, 9, 9, 8, 5).$$

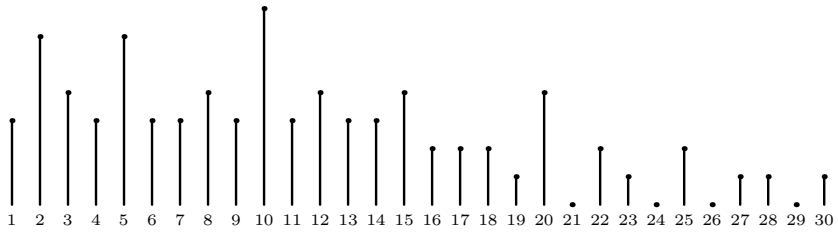


Figure 2.13: IIH of the t of Eqn. (2.1). By symmetry we need only specify the IIH for positive lags, as done here.

On the basis of these three sequences of differences we already see that "2" and "5" show up as likely periods of regular subsequences. The full IIH, for positive lag, is shown in Figure 2.13.

The six intervals in t of length 2 are [8, 10],[14, 16],[16, 18], [18, 20], [25, 27] and [28, 30], whereas the six intervals of length 5 are [0, 5], [5, 10], [10, 15], [15, 20], [20, 25] and [25, 30].

The first six intervals show regular sequences 8–10, 14–16–18–20, 25–27 and 28–30, while the second six intervals show one regular sequence 0–5–10–15–20–25–30. We thus find the regularity with period 5 and *duration* (total length) 30 but also a regularity with period 2 and duration 6. Just two times cannot be considered a real sequence. Looking upon intervals as train wagons that can be coupled by spikes which occur at common times (the ends of the wagons) we indeed can speak of *spike trains* as coming forward by this procedure.

Figures 2.14 and 2.15 show how IIH can be employed to determine the firing frequency of the dominant neuron in the recording. In Figure 2.14, a small portion of the raw spike data is shown on the left. Once the data is processed, and the spikes are localized, the IIH is constructed by pooling spike events after each spike. The peak of the IIH represents the dominant interspike interval time, i.e. 187 Hz. When we look at the rest of the IIH, the global wave pattern is indicative of long-term tremor. In Figure 2.15, the high-frequency signal from a dying neuron is depicted. The IIH reveals that the neuron bursts with 227 Hz frequency.

2.3.3 Connection between autocorrelation and IIH

The IIH procedure generates from a sequence of m time instances t a new sequence of $m-1$ positive time lags and it appears to require $\mathcal{O}(m^2)$ operations. Forming the autocorrelation $r_k = \sum_j s_j s_{k+j}$ of a signal $s \in \mathbb{R}^n$ on the other hand requires $\mathcal{O}(n \log(n))$ operations. In theory there is no relation between n and m (other than $n > m$ and some variations) so without further assumptions it is hard to compare the complexity of the two approaches. Oddly enough autocorrelation and IIH are equivalent for a single event type[9]:

[9] When different categorical events can be related to each other, the inter-event interval histogram can be employed to determine the regular patterns too, see [7]

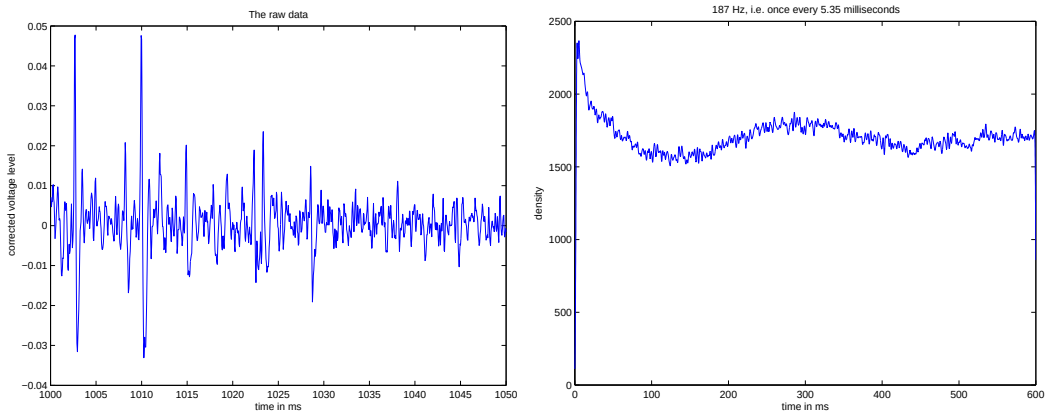


Figure 2.14: 187Hz period+long term tremor. Left: raw data x_k; right: IIH with a peak at $t = 5.35$ms corresponding to frequency of 186.9 Hz.

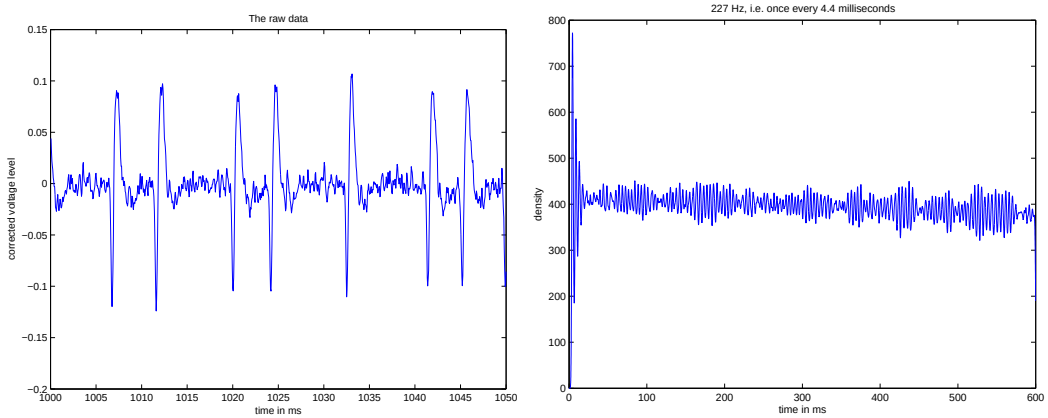


Figure 2.15: X-cell (RIP). Left: raw data x_k; right: IIH with peaks at $t = 4.4$ ms and multiples, indicating a frequency of 227.3 Hz

Lemma 2.3.1. *Let $t \in \mathbb{N}^m$ and $s \in \mathbb{N}^n$ form a pair of time sequence and corresponding time series. Then the autocorrelation of s equals the time series of the IIH of t.*

Proof. The IIH seen as an operation on s (rather than t) is a sum of shifted s and therefore is a discrete convolution $h * s$. It is easily seen that h is in fact the time reversed s, but then the convolution $h * s$ is the autocorrelation. □

Indeed the two plots in Fig. 2.11 are equivalent. The result remains valid if time instances appear more than once in t, in which case s_k should be defined to mean the number of times that k appears in t. The result also implies that IIH and spectral analysis (DFT of s or its autocorrelation) contain the same information. The difference is the way they are computed and stored. It is as yet an open problem which of the two approaches is more efficient computationally. The IIH appears more natural.

2.3.4 Approximate regularity

Neurons will fire at time intervals that are not completely equal in length, but sufficiently close to call it regular firing. We therefore consider *approximate regularity* for firing rates, demonstrated on a very simple but illustrative example. Let the time sequence for spike events be

$$s_t = (0, 30, 59, 87, 119, 150).$$

The consecutive intervals have lengths 30, 29, 28, 32, 31, which would correspond to quite "close" values in the IIH. A strictly regular sequence with period 30 would show five times 30, but now there are five intervals close to 30 and with average 30.

The question of determining the regularity of a sequence can be answered by considering intervals $[30-\Delta, 30+\Delta]$ around the average value. $\Delta = 0$ corresponds to the strictly regular sequence. We propose to use the following *measure* for the regularity sequences:

$$R = 1 - \frac{\Delta}{\text{average}} \approx 0.93.$$

where $R = 1$ corresponds to strict regularity. Δ is the maximum difference occuring betweeb interval lengths and the average for a set of close differences of times that is tested for regularity. We assume that no set should be considered for which Δ is larger than the average, so that R is a non-negative number in the interval $[0, 1]$.

It must be stressed that once a set of differences is chosen, one still has to check whether indeed one spike train has been found. A very simple example of two spike trains with period 5 that interfere, is given by the sequence

$$t = (0, 1, 5, 6, 10, 11, 15, 16).$$

The histogram shows peaks at “1” and at “5”. The four differences of 1 do not form a train at all, whereas the six differences 5 turn out to form two trains: (0, 5, 10, 15) and (1, 6, 11, 16).

An alternative approach to detect regularity using statistcal methods is indicated next. For a sequence of time instances $t = (t_1, t_2, \ldots)$ at which spikes occur, define the sequence of differences

$$\Delta t = (t_2 - t_1, t_3 - t_2, \ldots).$$

Assume that the differences $t_{k+1} - t_k$ are a realization of a single random variable T. Based on the emperical distribution and using an unparametrc test it is possible to find the distribuition of the random variable T. Under the assumption that T is normally distributed, $N(\mu, \sigma^2)$ and based on the available realization Δt it is possible to find estimators $\hat{\mu}$ and $\hat{\sigma}^2$ of the mean and the variance of the normal distribution. Then taking into consideration a confidence level of, say, 95% for all the realizations then $t_{k+1} - t_k \in (\hat{\mu} - 2\hat{\sigma}, \hat{\mu} + 2\hat{\sigma})$ can be considered indicating approximate regularity of the firing rates.

2.4 Concluding remarks

In this paper we mentioned four goals in Section 2.1.1.

The first goal mentioned was pinpointing the location of spikes. The main problem was the removal of background noise in combination with fractional time shift correction. This problem was dealt with in Section 2.2.1, with Figure 2.4(b) as description of the final result.

The second goal, classification of spikes, was treated in sections 2.2.2 and 2.2.3. We can view a spike as having several features (width, height, width and height of upward part, width and height of downward part, et cetera). Also combinations of features can be relevant. The PCA treated in Section 2.2.2 *automatically selects* features that distinguish spikes. In the coding approach of Section 2.2.3 these features are set *manually*. It turns out that the main feature is the amplitude. The PCA analysis revealed that occasionally other features are relevant, as shown by the presence of three clusters in Figure 2.5. To obtain this second feature from the PCA it is important that the alignment of the spikes in time is good. The three clusters were only observed after the fractional time shifts of Section 2.2.1 were done.

In Figure 2.5 values for the two dominant features from the PCA are displayed for a set of spikes. Clearly groups (clusters) can be distinguished. Although these groups are clearly visible, it is still a question how to select the groups. For this purpose automatic clustering algorithms exist. Of course in such simple examples manual grouping is also easily done. We feel that automatic clustering combined with visual inspection of the outcome and the possibility to change the cluster areas could be of interest for the application.

Both the manual and the PCA based feature selection were only applied to very few traces, so it is difficult to say whether the manual or PCA based method is better. Also, the main difference between spikes is in the amplitude, which is easy to measure. But overall our judgment at this moment is in favor of the PCA. It is a well established technique, which produces pictures suitable as input for cluster analysis. Results of the manual method are less clear.

The third goal was to distinguish spike trains according to three types. This was discussed in Section 2.3. The main problem was to determine spike trains with certain characteristic time spacings and determine their duration. The difficulty lies in the fact that different spike trains may overlap. In Section 2.3.1 classical autocorrelation was applied, whereas in Section 2.3.2 another approach, the so-called interspike interval histogram (IIH) was considered. In Section 2.3.3 the two techniques were connected. Since the two techniques are essentially equivalent they share the same advantages and disadvantages, except for their computational complexity which is yet unsettled. For overlap free spike trains and artificial data the two methods are transparent and appear to work well. The case of overlapping spike trains needs to examined further before conclusions can be drawn.

To deal with the fact that the intervals between two consequitive firings of a neuron will only be approximately the same in Section 2.3.4 the concept of approximate regularity was introduced.

Acknowledgment. We would like to thank Kevin Dolan and Lo Bour for their active presence during the SWI-week and for furnishing all kinds of information.

Bibliography

[1] A.F. Atiya. Recognition of multiunit neural signals. *IEEE Transactions on Biomedical Engineering*, 39(7):723–729, 1992.

[2] R. Chandra and LM Optican. Detection, classification, and superposition resolution of actionpotentials in multiunit single-channel recordings by an onlinereal-time neural network. *IEEE Transactions on Biomedical Engineering*, 44(5):403–412, 1997.

[3] M.S. Fee, P.P. Mitra, and D. Kleinfeld. Automatic sorting of multiple unit neuronal signals in the presence of anisotropic and non-Gaussian variability. *Journal of Neuroscience Methods*, 69(2):175–188, 1996.

[4] M.A.F. Figueiredo and A.K. Jain. Unsupervised learning of finite mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(3):381–396, 2002.

[5] K.D. Harris, D.A. Henze, J. Csicsvari, H. Hirase, and G. Buzsaki. Accuracy of Tetrode Spike Separation as Determined by Simultaneous Intracellular

and Extracellular Measurements. *Journal of Neurophysiology*, 84(1):401–414, 2000.

[6] M.S. Lewicki et al. A review of methods for spike sorting: the detection and classification of neural action potentials. *Network: Computation in Neural Systems*, 9(4):53–78, 1998.

[7] M.S. Magnusson. Discovering hidden time patterns in behavior: T-patterns and their detection. *Behavior Research Methods, Instruments & Computers*, 32(1):93–110, 2000.

[8] K.M.L. Menne, A. Folkers, T. Malina, R. Maex, and U.G. Hofmann. Test of spike-sorting algorithms on the basis of simulated network data. *Neurocomputing*, 44(46):1119–1126, 2002.

[9] A. Pavlov, V.A. Makarov, I. Makarova, and F. Panetsos. Sorting of neural spikes: When wavelet based methods outperform principal component analysis. *Natural Computing*, 6(3):269–281, 2007.

[10] R. Quian Quiroga, Z. Nadasdy, and Y. Ben-Shaul. Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering. *Neural. Comp.*, 16:1661–1687, 2004.

[11] F. Rieke, D. Warland, R. de Ruyter van Steveninck, and W. Bialek. *Spikes: Exploring the Neural Code*. 1999.

[12] W.M. Roberts and D.K. Hartline. Separation of multi-unit nerve impulse trains by a multi-channel linear filter algorithm. *Brain Res*, 94(1):141–9, 1975.

[13] U. Rutishauser, E.M. Schuman, and A.N. Mamelak. Online detection and sorting of extracellularly recorded action potentials in human medial temporal lobe recordings, in vivo. *Journal of Neuroscience Methods*, 154(1-2):204–224, 2006.

[14] E.M. Schmidt. Computer separation of multi-unit neuroelectric data: a review. *J Neurosci Methods*, 12(2):95–111, 1984.

[15] L.S. Smith and N. Mtetwa. A tool for synthesizing spike trains with realistic interference. *Journal of Neuroscience Methods*, 159(1):170–180, 2007.

[16] B.C. Wheeler and W.J. Heetderks. A Comparison of Techniques for Classification of Multiple Neural Signals. *IEEE Transactions on Biomedical Engineering*, pages 752–759, 1982.

[17] F. Wood, MJ Black, C. Vargas-Irwin, M. Fellows, and JP Donoghue. On the variability of manual spike sorting. *IEEE Transactions on Biomedical Engineering*, 51(6):912–918, 2004.

[18] P.M. Zhang, J.Y. Wu, Y. Zhou, P.J. Liang, and J.Q. Yuan. Spike sorting based on automatic template reconstruction with a partial solution to the overlapping problem. *Journal of Neuroscience Methods*, 135(1-2):55–65, 2004.

[19] A. Zviagintsev, Y. Perelman, and R. Ginosar. Algorithms and Architectures for Low Power Spike Detection and Alignment. *Journal of Neural Engineering*, 3(1):35–42, 2006.

3 Math Fights Flooding

Niels Besseling[1] Onno Bokhove[1] Alla Kolechkina[2]
Jaap Molenaar[3] Ronald van Nooyen[2] Vivi Rottschäfer[4]
Alfred Stein[1,5] Anton Stoorvogel[1*†]

Abstract

Due to climate changes that are expected in the coming years, the characteristics of the rainfall will change. This can potentially cause flooding or have negative influences on agriculture and nature. In this research, we study the effects of this change in rainfall and investigate what can be done to reduce the undesirable consequences of these changes.

Keywords: climate change, rainfall, drainage system

3.1 Introduction

At the 2008 Study Group Mathematics with Industry one of the problems concerned the impact of climate change on Dutch water management practices. More specifically, we were asked to study the effect of the increasing intensity of peaks of precipitation events on the water system managed by "het Waterschap Regge en Dinkel". Some explanation of the nature of this problem owner is in order. A Dutch "waterschap" is an institution run by a democratically elected board that is in charge of the management of the water quantity and quality of open water (streams, brooks, lakes, ditches and canals) in a given region. The board is elected by the local inhabitants and the institution is self financing: it determines the level of certain local taxes and collects those taxes for its own use. One of its main tasks is to protect the inhabitants against flooding and to manage the water levels such that agriculture, nature and shipping are supported. In the remainder of this paper we will use the term "water board" as a rough translation of "waterschap".

[1]University of Twente
[2]Delft University of Technology
[3]Wageningen University
[4]Leiden University
[5]Int. Institute for Geo-Information Science and Earth Observation, Enschede
*Other participants: Simon van Mourik (University of Twente)
†corresponding author, a.a.stoorvogel@utwente.nl

Figure 3.1: Twente (source: Waterschap Regge en Dinkel).

The Water board Regge and Dinkel is in charge of an area of approximately 40 by 40 kilometers containing the towns of Almelo, Enschede and Hengelo (Figure 3.1).

The problem statement limited the area of interest to the area that, due to terrain elevation and hydrology, discharges its precipitation into the stream the Regge. This area is called the *catchment* of the Regge. The Regge in its turn discharges into the river Vecht.

We examined the Sobek[6] model that was made available by the water board and found that the region below the Twente Kanaal discharges mostly into the Twente Kanaal despite the presence of culverts under the Twente Kanaal. This provided a clear southern border for the catchment. The total Regge catchment consists of a considerable number of subcatchments. A subcatchment is a subarea that discharges all its water via one point on its boundary into a small stream or canal.

In brief, the problem is to find a way to design and evaluate adaptations of the Regge catchment that will keep the discharge peak into the Vecht within a given envelope. Of course, this discharge peak varies in time. To establish general recom-

[6]Trademark of WL — Delft Hydraulics (part of Deltares)

mendations, the water board agreed on defining a typical precipitation event, which serves as a kind of benchmark for any system. This is simulated under the assumption of uniform rainfall over the catchment. This standard precipitation event is a 10-day period of rainfall data (preceded by a long period of almost 40 days with a constant minimal amount of rain to counter initialization effects in a model such as Sobek) as shown in Figure 3.2. For each subcatchment area this precipitation event will lead to a discharge curve that lags behind the precipitation curve and is longer than 10 days. Examples of such discharge curves are shown in Figure 3.3.

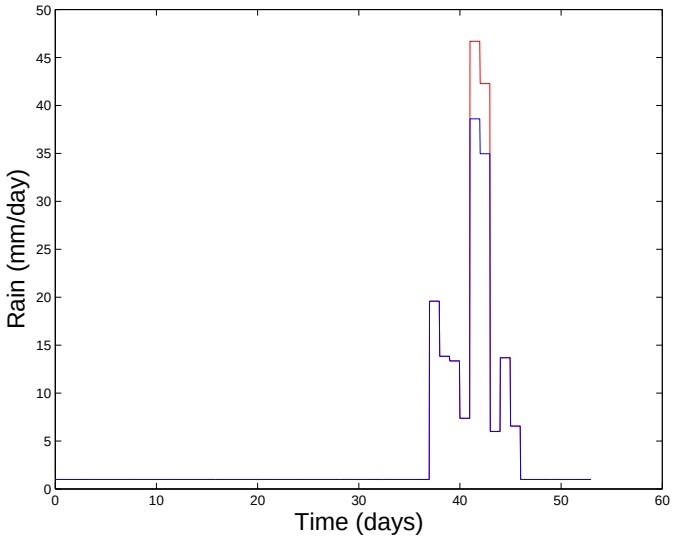


Figure 3.2: Benchmark precipitation event before (blue) and after (red) after climate change.

The discharges from different subcatchments flow together in the Regge. The water involved arrives at the Regge with a time delay that is mainly determined by the distance between the discharge point of the subcatchment under consideration and the Regge. The discharges from all the subcatchments sum up with the appropriate time delays in the Regge. In turn, the Regge discharges its water into the Vecht and a typical Regge discharge curve for the benchmark precipitation event in the current climate is the blue curve in Figure 3.4. This discharge has been computed using the Sobek model. In this figure, the red curve is the maximal discharge imposed to us by the Water Board Regge and Dinkel. The discharge curve is obtained when the standard precipitation event, which is a kind of worst-case rainfall in the current climate, is applied to the present situation in the Regge catchment. It is important to preserve the dip in the discharge after 46 days to allow for the discharge peak from another catchment that flows into the Vecht further upstream. This is an important boundary condition for the study of this project. After the climate change the response to the *new* standard precipitation event, which is a kind of worst-case rainfall in the future climate should respect the upper bound in the discharge curve indicated in red. However, as shown in Figure 3.5, without additional measures,

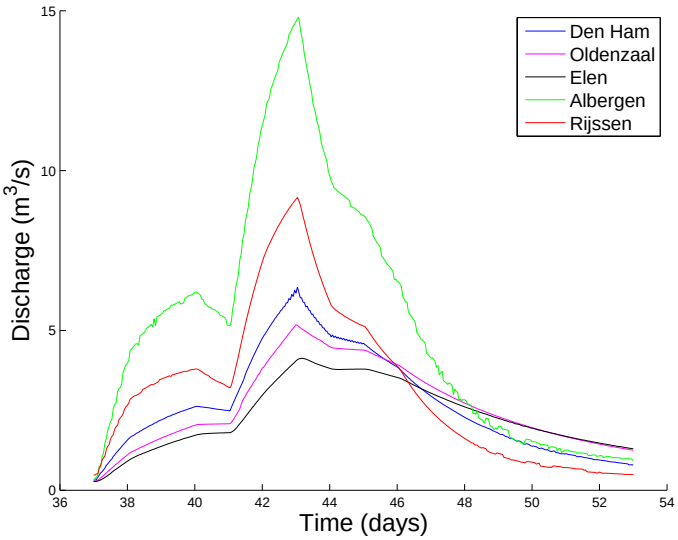


Figure 3.3: Discharge of selected catchments.

the expected discharge (indicated in blue) clearly violates this upper bound. The objective of this study is to look at measures that can be taken such that we get for instance a discharge as indicated in green which mostly respects the given upper bounds.

In other words, the aim of this project is to study what happens if the rainfall would intensify due to climate change. To show the effect we artificially increased the peak discharge in the standard precipitation event in such a way that the total volume in the event increased by ten percent (see Figure 3.2).

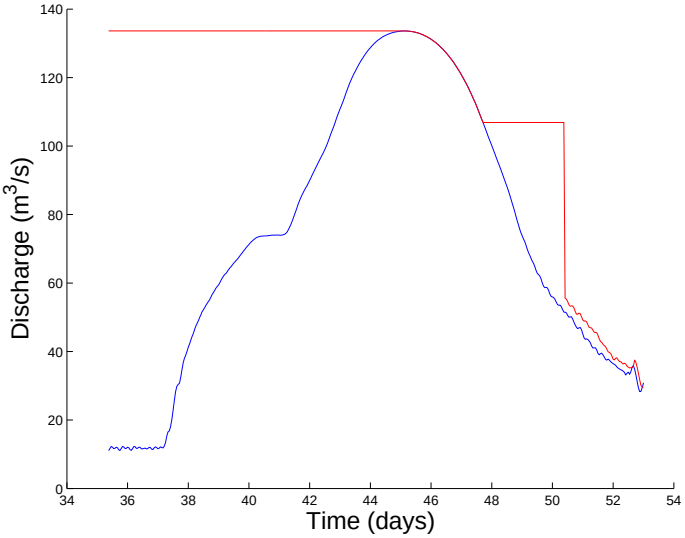


Figure 3.4: Discharge in current climate (blue) and maximal discharge (red).

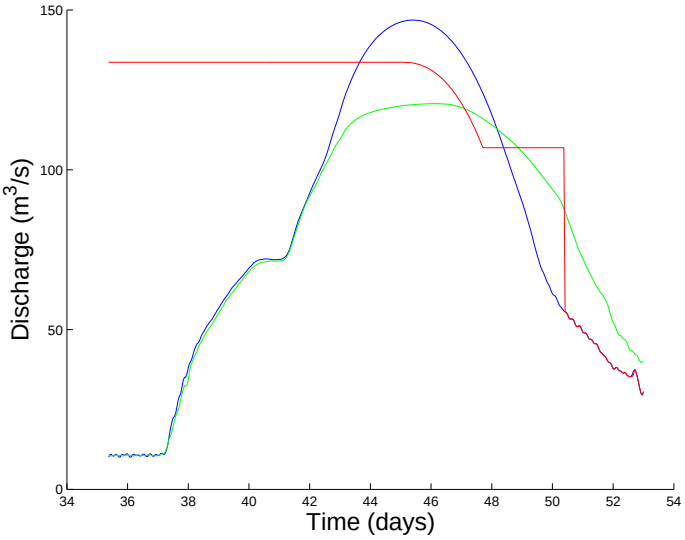


Figure 3.5: Discharge after climate change without and with additional measures together with maximal discharge.

To avoid undesirable discharge rates of the Regge due to increased rain fall, the water board suggested the following possible measures as viable components for a solution.

- *Improved drainage in a subcatchment.* This results in an earlier release, and in a narrower discharge peak from that subcatchment. The improvement could be achieved by additional drainage pipes and/or drainage ditches. However, this can also be realized, to a certain extent, by lowering of the overflow heights of the weirs. Earlier arrival of the run-off at the Vecht from a certain subcatchment could reduce the height of the peak by a better spreading over time of the discharge of the different catchments over time.

- *Slower drainage in a subcatchment.* This results in a later release and a flatter discharge peak from the subcatchment. Reduction of the drainage can be achieved by removal of drainage pipes and/or drainage ditches or by raising the water level in the drainage ditch network. This can also, to a certain extent, be realized by increasing the overflow heights of the weirs. This increases the available storage in the soil and the local collection canals. It flattens and delays the entry of the discharge peak from this subcatchment into the transport canals. Later arrival of a flattened discharge peak can reduce the height of the total discharge peak arriving at the Vecht directly by the flattened peak of the discharge of the subcatchments or, indirectly, by a better spreading over time of the discharge of the different catchments.

- *Storage.* Adding storage basins has effects that are similar to those of slowing the drainage of a subcatchment, but they are more flexible as they can also be

used to flatten and/or delay a discharge peak that has already left the soil and the collection canals of a subcatchment. It can affect peaks in the transport canals. Again, as argued before, later arrival of a flattened discharge peak can reduce the height of the peak arriving at the Vecht.

For the total discharge into the Vecht, we must take all contributions of the subcatchments into account. If we link subcatchments whose discharge peak reaches the Vecht at approximately the same time, we get a set of isochrones on the map. This illustrates two aspects of the problem. First, for narrow peaks the longest isochrone (the line connecting points from which water will take the same amount of time to reach the discharge point into the Vecht) will tend to dominate the discharge peak. Second, for wide peaks or rainfall-runoff curves with fat tails the later peaks will piggy back on top of the earlier ones and dominate the discharge peak. The second process will later be confirmed by a sensitivity analysis. The scale of the area (about $40 \times 40\,\mathrm{km}^2$), combined with the width of the peaks from separate subcatchments and the average transport velocity of $1\,\mathrm{m/s} = 86.4\,\mathrm{km/day}$ (according to Regge and Dinkel) implies that the first process does not play a role of much importance.

In Section 3.2 we will obtain a simple model for the discharge based on fitting the data provided to us by the very detailed Sobek model. In Section 3.3 we will model one meadow with adjacent ditches in detail. It will be shown that this model, after suitable fitting of the physical parameters, fits very closely to the earlier model even for an area of more than 1000 hectare which has a lot of detailed structure (small ditches; non-uniform soil characteristics, etc) which are not taken into account in the physical model. In Section 3.4, the sensitivity of the discharge curve in the Vecht to changes in the parameters of the model is analyzed for a specific subcatchment. This gives an idea what can be done to modify the discharge into the Vecht by taking specific actions in suitably chosen subcatchments. Resulting recommendations of our analysis are presented in Section 3.5.

3.2 A dynamical relation between precipitation and discharge

In this section we develop a dynamic model to relate a known discharge curve of a subcatchment to a known precipitation curve, see [1]. In the next section we shall outline how a physical model for the discharge curve of a subcatchment can be obtained which, for a given rainfall data, will result in a discharge curve. The latter curve clearly still depends on certain physical parameters used in the model. In contrast, in this section both the rainfall and discharge curves are given and then a dynamic relationship is fitted between the two curves.

In a subcatchment C, we have during day i an amount of rainfall r_i, which leads to a total discharge d_i in m^3 over that day into the release point of the subcatchment.

Here r_i is the amount of m^3 of rainfall which is the product of the rainfall in a particular day (indicated in Figure 3.2) times the area of the subcatchment (we assume uniform rainfall over the whole region).

A transition is introduced that quantifies on day i the fraction ρ of r_i that is discharged and a fraction $1-\rho$ of r_i that is kept within the catchment. The dynamics within one day are discarded. That means that in the first day, i.e. at the start of the rain event, a fraction ρ of the rainfall r_1 is discharged, and a fraction $1-\rho$ is kept in the catchment. To initialize the model we assume there is no water in the catchment at the beginning of this event. At the next day, the discharge d_2 is given by:

$$d_2 = \rho r_2 + \rho(1-\rho)r_1,$$

where a fraction ρ of the new rainfall is discharged but also a fraction ρ is discharged of the remaining water in the system due to rainfall of earlier days. For a specific subcatchment we have observations of rainfall and discharge over n days and we obtain:

$$d_{i+1} = (1-\rho)d_i + \rho r_{i+1}, \qquad d_0 = 0. \tag{3.1}$$

This can alternatively be presented using a matrix representation:

$$\begin{pmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{pmatrix} = \begin{pmatrix} \rho & 0 & \cdots & 0 \\ (1-\rho)\rho & \rho & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ (1-\rho)^{n-1}\rho & \cdots & (1-\rho)\rho & \rho \end{pmatrix} \begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ r_n \end{pmatrix} =: A \begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ r_n \end{pmatrix}. \tag{3.2}$$

Hence, the discharge has been approximated by a one parameter model. This parameter, however, is specific for each subcatchment. It is governed by the physical conditions of the catchment, like the lateral movement, the vertical changes in elevation, the carriage capacity of the soil and the physical soil unit composition. The parameter indicates in an averaged way how fast the rain is discharged into the canal system outside the area.

3.2.1 Estimation

Estimation of parameter ρ was carried out by a least squares method. Using (3.1), we first note that the matrix in (3.2) has an inverse with a nice structure and we obtain:

$$A^{-1} = \rho^{-1} \begin{pmatrix} 1 & 0 & \cdots & \cdots & 0 \\ \rho-1 & 1 & \ddots & & \vdots \\ 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \rho-1 & 1 \end{pmatrix}$$

For several subcatchment we compared the actual discharge from the Sobek model to the discharge predicted by our model. The expression

$$\left\| \rho A^{-1} \begin{pmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{pmatrix} - \rho \begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ r_n \end{pmatrix} \right\|^2,$$

which is is equivalent to:

$$\sum_{i=0}^{n} \|d_{i+1} - (1-\rho)d_i - \rho r_{i+1}\|^2$$

is then a quadratic function in ρ and the minimization of this error to find the optimal value for ρ is then easily achieved. The first 35 days of the rainfall in 3.2 are intended to reduce the effect of initialization. This is crucial in the Sobek model. In our case, the initialization is only related to setting $d_0 = 0$. However, our model needs to be more accurate in days where the discharge is substantial. We improved this process slightly by scaling the squared error by the actual discharge per day:

$$\sum_{i=0}^{n} \|d_{i+1}\| \|d_{i+1} - (1-\rho)d_i - \rho r_{i+1}\|^2$$

This weighting makes the model more accurate during days with a large discharge.

3.2.2 Results

Area	$\hat{\rho}$
Elen	0.15
Oldenzaal	0.16
Den Ham	0.27
Albergen	0.39
Rijssen	0.40

Table 3.1: Estimated ρ coefficients for 5 selected catchments.

We obtained the results listed in Table 3.1 for five selected catchments. Rijssen and Albergen have the largest values of ρ which corresponds to a high peak and a short tail, since most of the rain is discharged into the canal system within a few days. This is clearly consistent with the discharge curves in Figure 3.3. Elen and Oldenzaal have a low value of ρ and hence a low peak and a long tail. These areas keep the rain within the catchments and slowly discharge it into the canal system.

These results are as expected since the Rijssen catchment is located on sandy soil on a large elevation and, hence, the catchment will have a smaller carrying capacity than the other catchments. Consequently, the discharge occurs in a shorter period.

3.3 Rake model

The model proposed in the previous section uses a simple model ignoring for instance the faster dynamics during the day but also ignoring the spatial structure within a subcatchment. In this section, we propose a simplified one-dimensional ground water and hydraulic model to investigate an optimization strategy for designing catchment basins and ground water level management. It incorporates explicitly the weirs and the spatial structure and hence can be used to study the effects of raising or lowering the overflow heights of the weirs or the introduction of additional ditches. It is called the "rake model" because the river Regge is assumed to be connected to a series of ditches associated with two adjacent meadows. Rain will uniformly fall on the whole region, thus also on each meadow. A simple one-dimensional diffusion model is set-up to manage the transport of rain water into the ground to an adjacent ditch. Each (half) meadow is connected to a ditch. Each ditch runs into the Regge and is controlled by a weir at its exit point. And, finally, this exit point has a certain distance to the mouth of the Regge into the river Vecht. Each meadow is chosen to be rectangular and has a width W and length L, the latter also being the length of the ditch. See also Figure 3.7.

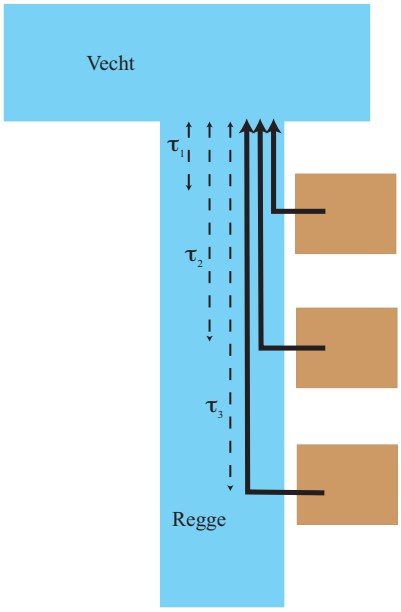


Figure 3.6: Sketch of subcatchments with a different distance to the Vecht, leading to a time lag τ_i in the time when the water reaches the Vecht.

We consider $m = 1, \ldots, M$ meadows and consider one meadow-ditch combination or catchment with index m, dropping the index m at first for ease of notation. Rain water seeps into the ditch from the meadow and the ground water level $h = h(x,t)$ in the pasture depends on the distance x from the ditch with $x \in [0, W/2]$, and time t. The ditch lies at $x = 0$ and the middle of the meadow at

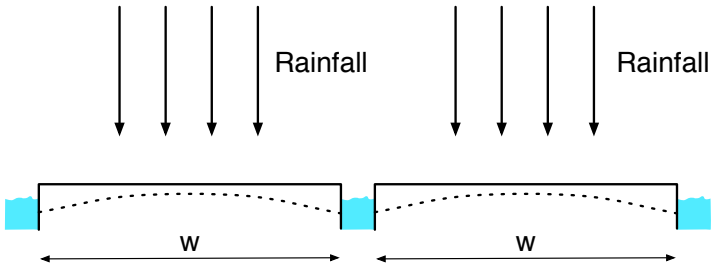


Figure 3.7: Cross section of a meadow with a ditch on each side. The water level is also indicated.

$x = W/2$. Diffusion with diffusion constant μ and soil permeability k governs the dynamics as well as the rain fall $R = R(t)$. The ground water level is assumed uniform in the direction along the ditch; hence we ignore end effects. The governing equation is

$$\frac{\partial h}{\partial t} = \mu \frac{\partial^2 h}{\partial x^2} + \frac{R}{\varphi} \tag{3.3}$$

with φ the porosity of the soil. The water level $h_0 = h_0(t)$ in the ditch is as follows

$$\frac{\mathrm{d}h_0}{\mathrm{d}t}(t) = \frac{\varphi \mu k}{b}[h(0,t) - h_0(t)] - \frac{\sqrt{2g}}{L} \max(h_0 - h_w, 0)^{3/2}, \tag{3.4}$$

in which k is a permeability coefficient, g is the acceleration of gravity, and the last term models a weir at the entrance of the ditch into the Regge. The last term consists of a standard hydraulic approximation for flow over weirs, see [4]. The height of the weir $h_w = h_w(t)$ is a specified function (of time); it can be used to control the outflow of water into the Regge hydraulic system. Catchment basins are modeled simply by specifying a different width $b = b(t)$ of the ditch; it is also a specified function of time. The boundary conditions involve symmetry at $x = W/2$, and consistency at $x = 0$:

$$\frac{\partial h}{\partial x}(0,t) = k[h(0,t) - h_0(t)] \qquad \text{and} \qquad \frac{\partial h}{\partial x}(W/2,t) = 0. \tag{3.5}$$

It is useful to consider the volume balances of water. The change in time of the volume $V = V(t)$ of water in the meadow, associated with one ditch, follows by integration of the diffusion equation (3.3) over the relevant area $W/2 \times L$ and multiplication by φ, while using the boundary conditions (3.5); we obtain

$$\frac{dV}{dt} = \varphi\, L\, \frac{d}{dt} \int_0^{W/2} h(x,t)\, dx = -\mu\, k\, \varphi\, L\, \big(h(0,t) - h_0(t)\big) + \tfrac{1}{2} RLW. \tag{3.6}$$

The change of volume V_0 of water in the ditch follows by multiplication of (3.4) with $L\,b$, to obtain

$$\begin{aligned}\frac{dV_0}{dt} &= b\,L\,\frac{d}{dt}h_0(t) \\ &= \mu\,k\,\varphi\,L\,\big(h(0,t)-h_0(t)\big) - \sqrt{2\,g}\,b\,\max(h_0(t)-h_w(t),0)^{3/2}.\end{aligned} \tag{3.7}$$

Hence, we observe that the discharge from the meadow into the ditch is consistently modeled as

$$\mu\,k\,\varphi\,L\,\big(h(0,t)-h_0(t)\big).$$

The total discharge $Q = Q(t)$ of the ditch over the weir and into the river Regge follows from (3.7) as

$$Q(t) = \sqrt{2\,g}\,b\,\max(h_0(t)-h_w(t),0)^{3/2}. \tag{3.8}$$

We use $Q_m(t)$, instead of $Q(t)$, to indicate the discharge of ditch-meadow combination number m into the Regge which lies at a distance D_m from the mouth of the Regge into the Vecht. It is assumed that water released into the Regge from a ditch flows with a constant velocity v to the Vecht. Hence, water released from ditches of meadows lying further away from the Vecht will travel longer. We immediately see an optimization strategy emerge: by delaying or accelerating fallen rain water to reach the Regge as a function of the location of the meadow from the Vecht we may be able to avoid flooding downstream at the Vecht. Hence, the maximum discharge of water into the Vecht may be managed.

3.3.1 Numerical discretization

To facilitate the numerical discretization, we used a non-dimensional form of the model (3.3)–(3.5). These non-dimensional equations have subsequently been discretized with a finite difference methods, second order in space and first order in time. An explicit forward Euler time discretization is used for the diffusion equation, and the water level equation (3.4) is discretized semi-implicitly by integrating $h_0 - h_w$ instead of h_0 and splitting the nonlinear term as $\sqrt{(h_0^n - h_w^n)}\,(h_0^{n+1} - h_w^{n+1})$ with current time level h_0^n and future time level h_0^{n+1}, and so forth. A time step restriction follows directly from a maximum principle. We refer to a standard text book on numerical methods, see [3].

3.3.2 Numerical results

For simplicity we took a square meadow, i.e. $L = W/2$ and let rainwater, fallen on a meadow of area L^2, seep diffusively into one ditch. Firstly, we gauged the parameters μ, k and φ based on a reference simulation of the Sobek model. The

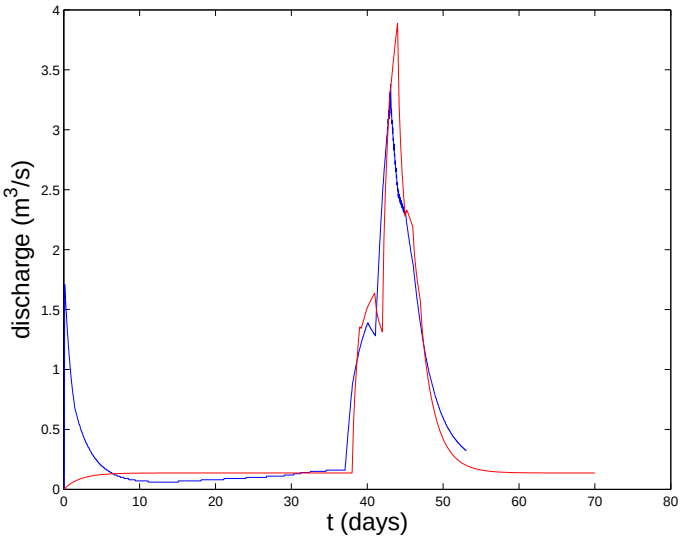


Figure 3.8: Comparison of the Sobek model (blue) and rake model (red) for an area near Den Ham.

Sobek model was run with the heavy ten-day rainfall distribution shown in Fig. 3.2. Subsequently, the discharge of a catchment area located between Den Ham and Vroomshoop concerning an area of $A_r = 118 \times 10^5\,\mathrm{m}^3$ was taken. We performed a run for one meadow of size $L^2 = 200\,\mathrm{m}^2$ scaled with factor s_f such that $L^2 s_f = A_r$, and compared the run-off curves. For the values $\mu = 4\,L_s^2/T$ and $k = 10/L_s$ and spatial scale $L_s = 50\,\mathrm{m}$ and time scale $T = 1\,\mathrm{day}$, the agreement between the Sobek model and one meadow in our rake model is surprisingly good, see Fig. 3.8. Other parameter values are $b = 2\,\mathrm{m}$, $h_w = 0.5\,\mathrm{m}$, and initially we filled the ditch to weir level, e.g., using initial conditions $h_0(0) = h_w$, and also $h(x,0) = h_0(0)$. Or, perhaps more appropriately, we note that the model is rainfall driven, and the sensitivities on μ and k appear to be relatively small.

Secondly, we considered the rake model with three meadows and ditches, at distances $D_m = m\,L_d$ with $m = 1,2,3$ away from the Vecht. We took $L_d = 20 \times 10^3\,\mathrm{m} = 20\,\mathrm{km}$ and the flow velocity v was taken to be $v = 1\mathrm{m/s}$. The (imaginary) water board for the Vecht has given us a maximum discharge rate of $8\,\mathrm{m}^3/\mathrm{s}$ of Regge water that is allowed to flow into the Vecht. In the base run the three ditches have the same parameter values as above, the only difference being their distance to the river Vecht. Our simulations for the same rainfall as in Figure 3.2 then show that the discharge peaks of each catchment arrives with a delay of about a quarter day ($20 \times 10^3/(3600 \times 24)$ day) into the Vecht, see the lines for the three shifted peaks of about discharge heights $4\,\mathrm{m}^3/\mathrm{s}$ in Figure 3.9. The accumulated discharge of these three catchment supersedes the allowed discharge maximum denoted by the fat horizontal line approximately between days 42 and 46. In our first attempt to optimize, we increased the weir height in the last catchment area to $0.65\,\mathrm{m}$, while starting the ditch level at $0.5\,\mathrm{m}$. Hence, the ditch of length A_r first needs to be filled

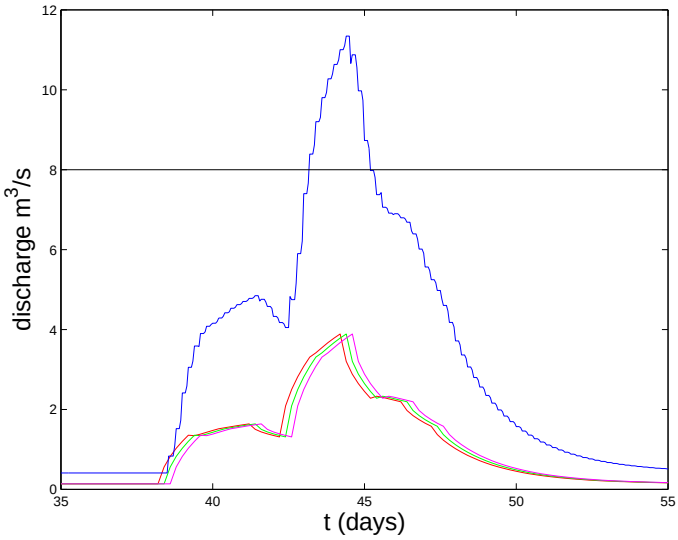


Figure 3.9: Discharge rates of the three catchment areas and their accumulated values (blue).

before rain water flows into the Regge. This constitutes a delay. In Figure 3.10, we see that the discharge peak of the third catchment area (indicated in magenta) starts later, at day 42 instead of day 38 as we saw in the first run, but the accumulated discharge denoted by the blue line, is still too high. Flooding thus still occurs. Our final strategy, in Figure 3.11, is to heighten the weir to 0.6 m and lower the water level in the ditch and the meadow to $h_0(0) = h(x, 0) = 0.25$ m, for example, by an early precautionary release of water. This mimics the use of an additional storage basin. As a consequence, the discharge peak (in magenta) in the lower right half of the plot, is greatly reduced, and assures that the accumulated water discharge of Regge water into the Vecht stays below the maximum discharge level. Clearly, these changes need to be optimized but this can only be done if other factors are taken into account. For instant, increasing or decreasing the overflow level of a weir has economic effects on agriculture in the region, has ecological effects, et cetera. Also zoning plans might not allow certain actions to be taken.

3.4 Sensitivity analysis

In this section we investigate the influence of measures taken in individual subcatchments on the discharge curve of the Regge $D(t)$. The latter is the sum of the discharge curves of individual subcatchments $D_m(t)$, $m = 1, \cdots, M$ in the following way

$$D(t; \rho_1, \cdots, M) = \sum_{m=1}^{M} D_m(t - \tau_i; \rho_i). \tag{3.9}$$

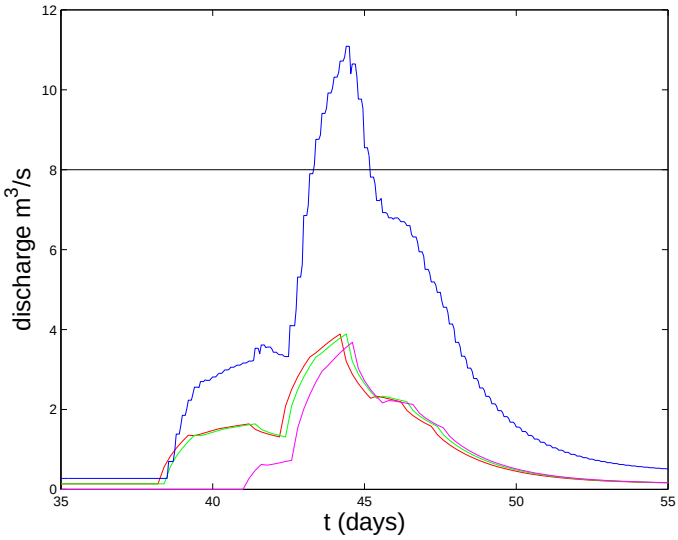


Figure 3.10: Discharge rates with weir height 0.65 m and initial ditch level 0.5 m for one subcatchment (red).

Here, τ_m is the time lag resulting from the fact that the water from a subcatchment needs to flow from the exit point of that subcatchment to the point where the Regge discharges into the Vecht, see Figure 3.6. As discussed in Section 3.2, each individual discharge curve D_m can quite accurately be characterized by only one parameter, ρ_m. From the simple structure of (3.9) it directly follows that

$$\frac{\partial D(t;\rho_1,\cdots,M)}{\partial \rho_m} = \frac{\partial D_m(t-\tau_m;\rho_m)}{\partial \rho_m}. \tag{3.10}$$

In Section 3.2 we introduced discretized versions d_i (indicating the discharge during day i) of $D_m(t;\rho_m)$ (indicating the discharge at time t). In that representation the derivative with respect to ρ_m can for any m be explicitly indicated as:

$$\frac{\partial}{\partial \rho_m}\begin{pmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{pmatrix} = \begin{pmatrix} 1 & 0 & \dots & 0 \\ (1-2\rho_m) & 1 & \cdots & 0 \\ \vdots & \dots & \dots & \vdots \\ (1-\rho_m)^{n-2}(1-n\rho_m) & \dots & (1-2\rho_m) & 1 \end{pmatrix}\begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ r_n \end{pmatrix}. \tag{3.11}$$

So, given the value ρ_m of a subcatchment and given a standard (or adjusted) precipitation curve, the derivative curve $\partial D_m/\partial \rho_m$ is easily approximated as a function of time. An example is given in Figure 3.12.

This curve gives an indication of the sensitivity of any discharge curve to changes in the corresponding ρ. From this figure it is clear that the effect of a ρ is largest about 6 days after the rainfall started. This strongly coincides with the peak positions in both the precipitation and discharge peaks. The conclusions from such

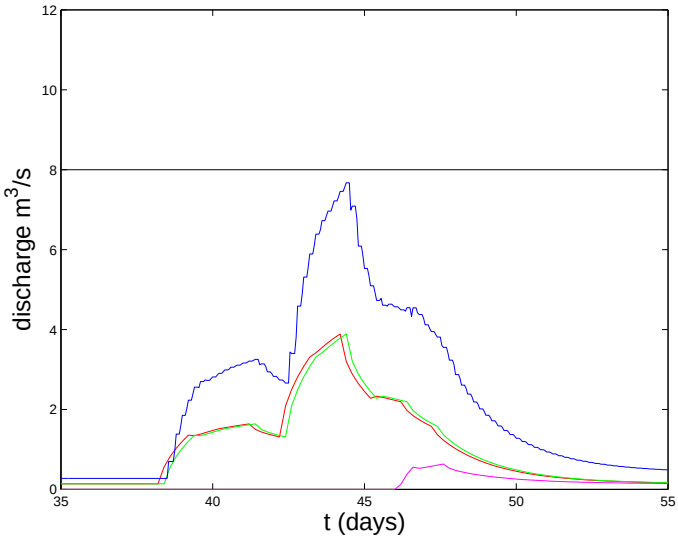


Figure 3.11: Discharge rates with weir height 0.6 m and initial ditch level 0.25 m for one subcatchment (red).

sensitivity analysis can be easily read in Figure 3.12: An increase in ρ_i strongly increases the height of the peak in the discharge curve D_i and flattens the tail. And reversely, if ρ is decreased the discharge curve will get a lower peak and a thicker tail. This agrees with the interpretation of ρ as the parameter measuring the fraction of water fallen on some day that is discharged that same day.

3.5 Recommendations

The discussions above yield the insight that changing the ρ_i parameter of a subcatchment influences the height of the discharge curve but does not influence the respective peak and tail positions in the discharge curve. Since the delay times τ_i are relatively small compared to the widths of the peaks in rainfall and discharge curves, the peaks in the discharge curves D_i all accumulate in the peak of the Regge discharge curve D and the same holds for the tails. This immediately leads to the following recommendation:

> *In case of intensified peaks in the rainfall due to climate change, the ρ value of a number of subcatchments should be decreased.*

The implementation of this recommendation requires some subtle considerations, which we summarize in the following remarks:

Remark a.: Reduction of the ρ value of a subcatchment implies that the drainage of the area should decrease. This could be achieved by closing some ditches or by

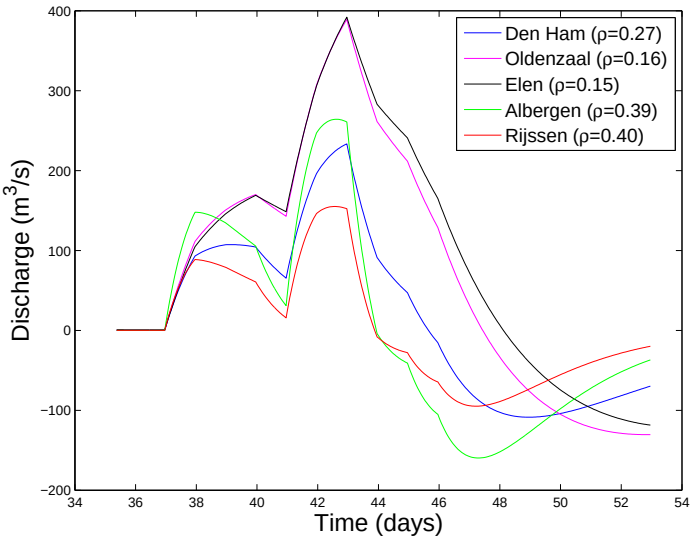


Figure 3.12: Time behavior of the derivative of a typical discharge curve with respect to the parameter ρ.

raising the water level in the drainage ditch network, by increasing the height of the weir, which at the same time increases the water storage capacity of the soil.

Remark b.: It does not matter whether ρ is reduced by a great amount in a relatively small number of subcatchments or if ρ is reduced by a small amount in many subcatchments. The total effect is in both cases nearly the same.

Remark c.: Reducing ρ in some subcatchments reduces the peak height in the Regge discharge curve, but enhances also its tail. So, the optimal choice must follow from a balance between these effects. The total effect of reducing values ρ_i should be such that the peak height in the Regge discharge curve remains under the critical value, dictated by the risk of flooding along the Vecht, and, at the same time, the tail in the Regge discharge curve should remain so low that no dangerous interference with the peak in the Vecht discharge curve occurs. This is a subtle balance. Since the choice of the subcatchments that are most suitable for a change in drainage capacity heavily depends on the local conditions and possibilities, we have not worked out this choice in detail.

Remark d.: The effect of the time delays τ_i is relatively small. If one would like to make use of the fact that the subcatchments differ in this aspect, one could best reduce the ρ parameter in the subcatchments with the largest time delays; the ones furthest away from the discharge point of the Regge into the Vecht. This is because their peaks would arrive latest at the discharge point and thus would interfere most with the peak in the Vecht discharge curve.

Bibliography

[1] A.D. Cliff and J.J. Ord. *Spatial processes*. Pion, London, 1981.

[2] Delft Hydraulics, the Netherlands. *User Manual SOBEK 2.11.002*, 2007.

[3] K.W. Morton and D.F. Mayers. *Numerical solution of partial differential equations*. Cambridge University Press, 1994.

[4] D.F. Young, B.R. Munson, and T.H. Okiishi. *A brief introduction to fluid mechanics*. Wiley, 1997.

4 Some studies on the deformation of the membrane in an RF MEMS switch

Vijaya Raghav Ambati[1] Andreas Asheim[2] Jan Bouwe van den Berg[3]
Yves van Gennip[4] Tymofiy Gerasimov[5] Andriy Hlod[4]
Bob Planqué[3] Martin van der Schans[6] Sjors van der Stelt[7]*
Michelangelo Vargas Rivera[3] Erwin Vondenhoff[4]

Abstract

Radio Frequency (RF) switches of Micro Electro Mechanical Systems (MEMS) are appealing to the mobile industry because of their energy efficiency and ability to accommodate more frequency bands. However, the electromechanical coupling of the electrical circuit to the mechanical components in RF MEMS switches is not fully understood.

In this paper, we consider the problem of mechanical deformation of electrodes in RF MEMS switch due to the electrostatic forces caused by the difference in voltage between the electrodes. It is known from previous studies of this problem, that the solution exhibits multiple deformation states for a given electrostatic force. Subsequently, the capacity of the switch that depends on the deformation of electrodes displays a hysteresis behaviour against the voltage in the switch.

We investigate the present problem along two lines of attack. First, we solve for the deformation states of electrodes using numerical methods such as finite difference and shooting methods. Subsequently, a relationship between capacity and voltage of the RF MEMS switch is constructed. The solutions obtained are exemplified using the continuation and bifurcation package AUTO. Second, we focus on the analytical methods for a simplified version of the

[1]University of Twente
[2]Norwegian Institute of Science and Technology, Norway
[3]VU University, Amsterdam
[4]Eindhoven University of Technology
[5]Delft University of Technology
[6]Leiden University
[7]University of Amsterdam
*corresponding author, s.vanderstelt@uva.nl

problem and on the stability analysis for the solutions of deformation states. The stability analysis shows that there exists a continuous path of equilibrium deformation states between the open and closed state.

4.1 Introduction

Radio Frequency switches (RF) of Micro Electro Mechanical Systems (MEMS) have achieved considerable attention in the mobile industry because of the need for an increase in frequency bands and energy efficiency. RF MEMS switches have several advantages over traditional semiconductors such as power consumption, lower insertion loss, higher isolation and good linearity. However, a thorough understanding of the electromechanical coupling between the electrical circuit and mechanical component of an RF MEMS switch is not fully established and this forms the subject of the present paper.

Problem description: RF MEMS switches typically consist of two electrodes which are thin membranes parallel to each other as shown in the Figure 4.1. In the schematic cross-section of the switch, Figure 4.1(a), the thick black lines indicate the bottom and top electrodes in which the bottom electrode is fixed and the top electrode is free to deform with its ends fixed. In the presence of equal and opposite electric charge Q in the electrodes, the top electrode deforms to balance the electrostatic force $F_{\text{electrostatic}}$ induced with its mechanical spring force F_{spring} for equilibrium. To avoid the contact between the two electrodes, a dielectric of thickness d_{diel} is provided on the top of the bottom electrode as indicated with dashed lines in Figure 4.1(a). Further, the thickness of the top electrode is h and it is separated by a distance g from the dielectric in the unforced state. The deformed shape of the top electrode at equilibrium is described by the displacement $u(x)$.

The equilibrium states are the critical points at which the the total energy is minimized. The total energy E_{tot} is given by the sum of the electrical energy E_{el} and the mechanical energy E_{mech}:

$$E_{\text{tot}} = E_{\text{el}} + E_{\text{mech}}.$$

The electrical energy E_{el} is given as

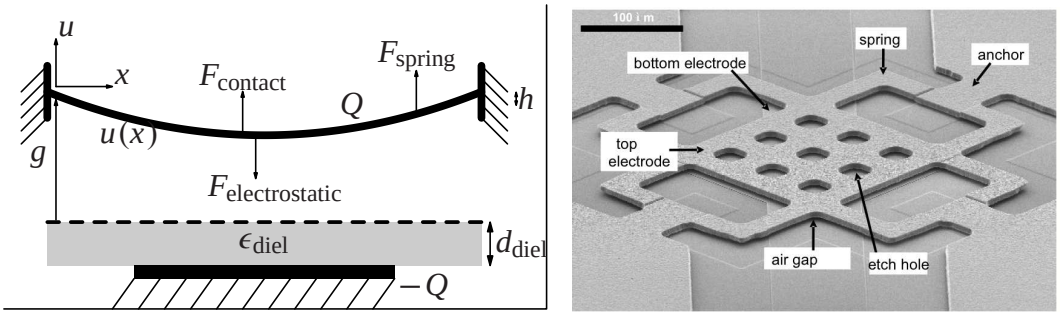
$$E_{\text{el}} = \frac{Q^2}{2C} \quad \text{with} \quad C(u(x,y)) := \int_{A_{\text{bot}}} \frac{\epsilon_0 \, \mathrm{d}x\mathrm{d}y}{g + u(x,y) + d_{\text{diel}}/\epsilon_{\text{diel}}},$$

where C is the capacitance, Q the electric charge, u(x,y) the displacement, ϵ_0 the vacuum permittivity coefficient, d_{diel} the thickness of dielectric, ϵ_{diel} the dielectric constant and A_{bot} the area of bottom electrode. In determining the capacitance C, the two electrodes are assumed to be parallel under no charge in the unforced state. Taking only the bending forces into account and assuming the thickness of

the electrode to be very small with zero initial stress, the mechanical energy is given by

$$E_{\text{mech}} = \int_{A_{\text{top}}} \frac{D}{2} |\Delta u|^2 \, \mathrm{d}x\mathrm{d}y, \quad \text{where} \quad D = \frac{2h^3 Y}{3(1-\nu^2)},$$

h is the thickness, A_{top} area of the top electrode, Y is Young's modulus and ν is the Poisson ratio of the top electrode.



(a) (b)

Figure 4.1: (a) Schematic cross-section of a capacitive RF MEMS switch. (b) Scanned electron microscope picture of a capacitive RF MEMS switch.

Problem formulation: The main problem is the following: find all the displacement states i of the top electrode $u_{\text{eq},Q,i}(x, y)$ for which the forces on the top electrode are in equilibrium at a fixed charge Q on the top electrode (or for a fixed voltage V between the electrodes). Several sub-problems are posed as follows:

- Is there always a continuous path of equilibrium states $u_{\text{eq},Q,i}(x, y)$ between the open state $u_{\text{eq},Q,i} = 0$ for all $x, y \in A_{\text{top}}$ and the closed state $u_{\text{eq},\infty,N} = -g$ for all $x, y \in A_{\text{bot}}$.
- Is there a function $f(u_{\text{eq},Q,i}(x, y), Q)$ that is monotonically increasing along this path?
- Can it be shown that along this continuous path $dE_{\text{mech}}dC > 0$ is always valid? Here E_{mech} is the mechanical energy and C is its capacitance.
- Is there a simple way to determine whether a state is stable or unstable at a fixed voltage or charge?
- For which geometries and boundary conditions is the problem analytically solvable? Most interesting is the situation in which the top electrode springs are clamped (zero displacement and zero slope) at some points of its boundary.

- The dynamics of the structure under the presence of gas damping is a related interesting problem.

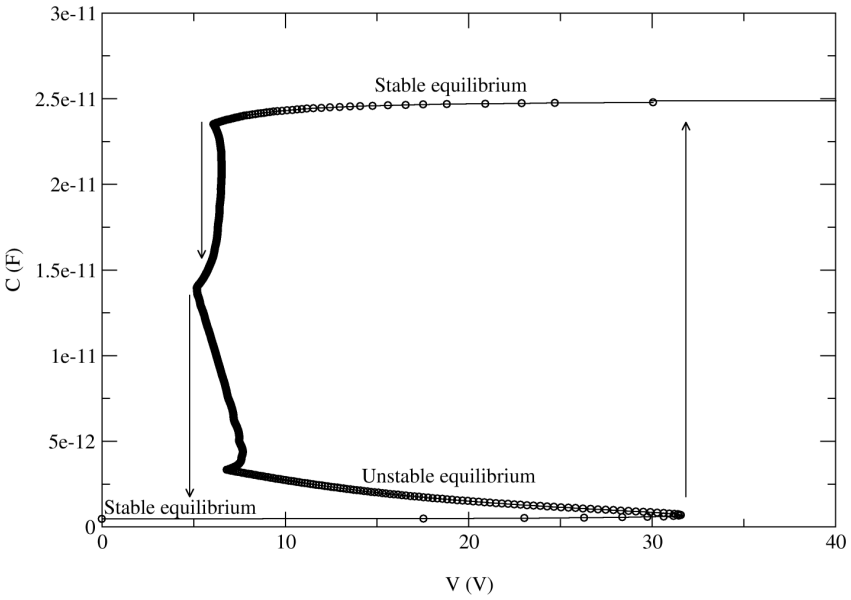
Finite element method: The deformation shapes at equilibrium are often solved using finite element packages. However, it is not straightforward to find multiple or all deformation shapes at equilibrium for a given voltage V as some of them are unstable. Given the deformation shape of the electrode $u(x, y)$, the capacitance of the RF MEMS switch is determined. Such a CV-curve is shown for two examples of RF MEMS switch in Figure 4.2(a) and (b). Multiple values of capacitance C for a given voltage V are clearly seen in Figure 4.2; a phenomenon called *hysteresis*.

Overview: The equilibrium problem of a RF MEMS switch is interesting both from a practical as well as a mathematical point of view. It should be stressed, however, that the entire problem is too general and difficult. Hence, in the present paper, we have considered a one dimensional version to obtain some interesting insights and solutions.

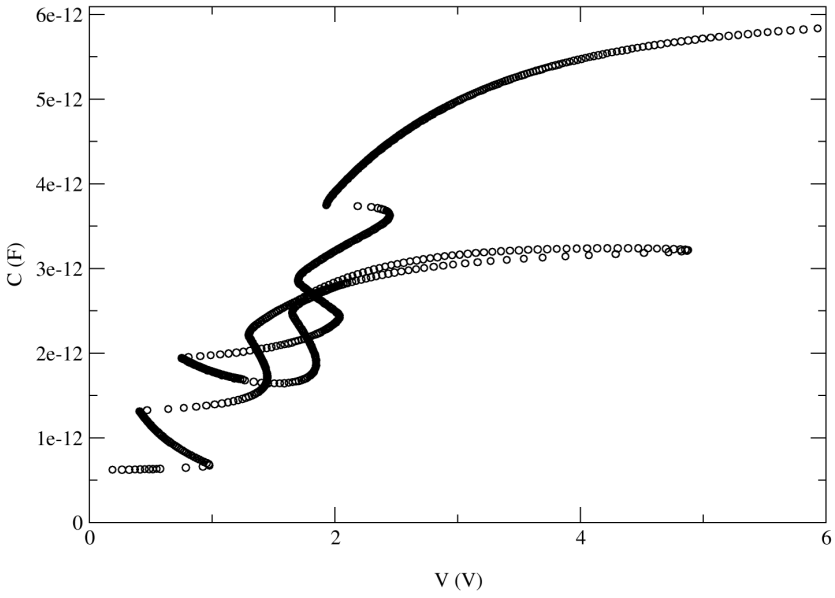
First, we prove that under certain conditions on the total energy of RF MEMS, the deformation states at equilibrium are stable. Second, we formulate an inequality from which the stability conditions are derived. Third, we prove that when the top electrode touches the dielectric, its deformation shape will have no gaps in the contact area with dielectric. Finally, we prove the existence of a continuous path of equilibrium states under some given mild conditions on the energy of the system.

Besides these theoretical results, we make use of numerical methods such as the finite difference and shooting methods to solve for the displacements of the deformation shape of the top electrode. To acquire insight into the nature of solutions, we generate several sets of deformation shapes using the continuation and bifurcation package AUTO. AUTO [3] typically generates sets of solutions to a given problem by continuation, i.e., it calculates a solution for any given parameter of the system. The main advantage of this approach as opposed to using finite element packages is that the non-unique or multiple solutions for a given problem are easily found. In addition, an article on modeling MEMS by using continuation is in preparation (see [14]).

The paper is divided into two parts. In the first part, we present the numerical methods to the present problem to gain some insight into the nature of solution. We then employ the continuation method AUTO and a shooting method to generate numerical solutions. In the second part, we discuss various analytical approaches to the problem. We derive full solutions to the linearized problem. Linear problems with any suitable boundary conditions have a unique solution and hence, no hysteresis is found. Finally, we present various other results for the nonlinear problem.



(a)



(b)

Figure 4.2: Calculated CV-curve (capacitance-voltage characteristic) of two different switches. (a) CV-curve of the switch of Figure 4.1. (b) CV-curve of the so-called “seesaw” RF MEMS switch.

4.2 Numerical Methods

4.2.1 Finite difference scheme

We consider a one dimensional model problem of the RF MEMS switch which exhibits the important qualitative aspects of the system and its non-dimensional form follows from the minimization of total energy:

$$\frac{\partial^4 u}{\partial x^4} = -\frac{\epsilon_0 V^2}{1-\eta+u} + \phi(u) \quad \text{on } x \in [0,1] \tag{4.1}$$
$$\text{with } u = \frac{\partial u}{\partial x} = 0 \text{ at } x = 0 \text{ and } x = 1,$$

where $u(x)$ is the displacement, η a small parameter, ϵ_0 the vacuum permittivity, V the voltage between the electrodes and $\phi(u)$ the contact force between the plate and the dielectric which is non zero for $u < -1$, i.e., when the scaled downward displacement is greater than the scaled gap $g = 1$ between the electrodes.

A simple finite difference scheme for the 1D model problem (4.1) is developed and implemented in MATLAB. The numerical solutions of this scheme are compared to the analytical approximations and they can serve as a basis for more advanced 2D simulations in the future. To obtain the finite difference scheme, we first divide the domain into $n-1$ grid cells with grid size Δx and n grid points. The displacement at each grid point x_i is denoted as $u(x_i) = u_i$. The biharmonic operator in (4.1) is discretized using a central difference scheme as follows:

$$\frac{\partial^4 u}{\partial x^4} \approx \frac{u_{i-2} - 4u_{i-1} + 6u_i - 4u_{i+1} + u_{i+2}}{\Delta x^4} + \mathcal{O}(\Delta x^2) \quad i = 2, \ldots, n-1. \tag{4.2}$$

Near the boundaries, we employ the boundary conditions $u_1 = u_n = 0, u_2 - u_0 = 0$ and $u_{n+1} - u_{n-1} = 0$ which are second order central difference approximations to the boundary conditions in order to get a consistent approximation. Substituting the approximation of biharmonic operator (4.2) in (4.1), the finite difference discretization takes the following form:

$$A\mathbf{u} = -\frac{\epsilon_0 V^2}{1-\eta+\mathbf{u}} + \phi(\mathbf{u}), \tag{4.3}$$

where A is a constant matrix and $\mathbf{u}$ is the displacement vector at the points $\mathbf{x} = x_i, i = 2, \ldots, n-1$. The discretized biharmonic operator A can be efficiently inverted using an iterative solver such as conjugate gradient method (CG). However, the right hand side of the equation is non-linear and hence, it is typically treated with a fixed-point iteration. The fixed-point iteration scheme is easily described by rewriting (4.3) as follows:

$$\mathbf{u}^{k+1} = A^{-1}\left(-\frac{\epsilon_0 V^2}{1-\eta+\mathbf{u}^k} + \phi(\mathbf{u}^k)\right). \tag{4.4}$$

Now given a guess $\mathbf{u}^k$ for displacement $\mathbf{u}$, we compute for displacement $\mathbf{u}^{k+1}$ using (4.4) per grid cell and iterate with respect to k until the solution converges. From a physical point of view, it is clear that the equation is not uniquely solvable for a certain range of voltages V. In fact, this is reflected in the fixed-point iteration scheme as it could converge to two different solutions for the displacement vector. Typically, the solution to which it converges depends on the starting point for the iteration. This suggests that a CV-curve with stable solutions of the system can be drawn. To draw the CV-curve, we start with a low voltage V for which the solution is unique and stable. Subsequently, we increment voltage V and use the previous solution as the starting for the fixed-point iteration scheme which resulted in a quick convergence to the nearby solution. Similarly, to obtain the remaining branch of solutions, we started with a high voltage V and repeated the previous procedure by decreasing the voltage V. This has lead us to construct a "continuous" branch of the CV-curve.

4.2.2 Shooting method

In this section, we consider a shooting method to solve the nonlinear one dimensional model problem of RF MEMS switch. The shooting method in some sense is the easiest method to find numerical solutions for a boundary value problem of a nonlinear ordinary differential equation. It relaxes the problem by ignoring one of the boundary conditions and replacing it by a "free" initial choice instead. This initial choice is adapted until the obtained solution satisfies the boundary condition that was ignored. We refer to [11] for a detailed description of the shooting method.

We distinguish three situations for the shooting method:

1. The top electrode touches the dielectric over some interval.

2. The top electrode touches the dielectric at one point.

3. The top electrode does not touch the dielectric.

Each of these cases contribute to different parts of the CV-curve. We describe the shooting method in detail for the first situation, i.e., when the plate touches the dielectric on some interval, and solve the shooting problem. The remaining two situations are solved analogously and hence, we omit the description. Finally we compute the CV-curve according to

$$C(v) = \frac{\Lambda\epsilon_0}{g}\int_{-1}^{1}\frac{dx}{1+u(x;v)-\eta}. \tag{4.5}$$

For all computations, we employ MATHEMATICA 6.

Electrode touches dielectric over some interval

Because of symmetry, we consider the electrode membrane in the half interval [0, 1] and take that the membrane touches the dielectric at $x = a$, where a is the distance measured from the fixed end $x = 0$ of the membrane and $0 \leq a < 1$. The nonlinear differential equation describing the shape of the membrane $u(x)$ between the fixed end and the contact with the dielectric is

$$\frac{\partial^4 u}{\partial x^4} = u'''' = -\frac{\epsilon_0 V^2}{1 - \eta + u}, \tag{4.6}$$

with boundary conditions

$$u(1) = 0, \quad u'(1) = 0, \quad u(a) = -1, \quad u'(a) = 0, \quad u''(a) = 0. \tag{4.7}$$

Here, an additional condition $u(a) = -1$ is required for the unknown contact point at $x = a$ on the dielectric.

It is convenient to make a change of variable x to $\tilde{x} = x - a$, $\tilde{u}(\tilde{x}) = u(x)$. Consequently, boundary conditions (4.7) now become as

$$\tilde{u}(1-a) = 0, \quad \tilde{u}'(1-a) = 0, \quad \tilde{u}(0) = -1, \quad \tilde{u}'(0) = 0, \quad \tilde{u}''(0) = 0, \tag{4.8}$$

and (4.6) remains the same as

$$\tilde{u}'''' = -\frac{\epsilon_0 V^2}{1 - \eta + \tilde{u}}. \tag{4.9}$$

In order to solve (4.6) and (4.7), we study the initial value problem for (4.9) with initial conditions

$$\tilde{u}(0) = -1, \quad \tilde{u}'(0) = 0, \quad \tilde{u}''(0) = 0, \quad \tilde{u}'''(0) = P, \tag{4.10}$$

which has a solution $\tilde{u}(\tilde{x}; P)$ with P an unknown parameter to be found later. Now, it remains to find a solution $P = P_s$ such that the solution of (4.9) and (4.10) satisfies the following condition at some point $b > 0$:

$$\tilde{u}(b; P_s) = 0, \quad \tilde{u}'(b; P_s) = 0. \tag{4.11}$$

Setting $a = 1 - b$, we obtain the solution $u(x) = \tilde{u}(\tilde{x}; P_s)$ satisfying (4.6) and (4.7). Note that, for the case $b > 1$ a solution of (4.6) and (4.7) does not exist.

The function $\tilde{u}(\tilde{x}; P)$ increases as function of P, see Figure 4.3(a). For small P, $\tilde{u}(\tilde{x}; P)$, as a function of $\tilde{x}$, increases, reaches a negative maximum and then decreases, see curves below the red one in Figure 4.3(a). For larger P, $\tilde{u}(\tilde{x}; P)$ increases and has positive first derivative where it crosses the line $\tilde{u} = 0$ for the first time, see curves above the red one in Figure 4.3(a). For $P = P_s$ the function $\tilde{u}(\tilde{x}; P)$ has a local maximum $\tilde{u} = 0$ (the red curve in Figure 4.3(a). This function satisfies the conditions (4.11) and b is the value of $\tilde{x}$ at which $\tilde{u}$ has the local maximum.

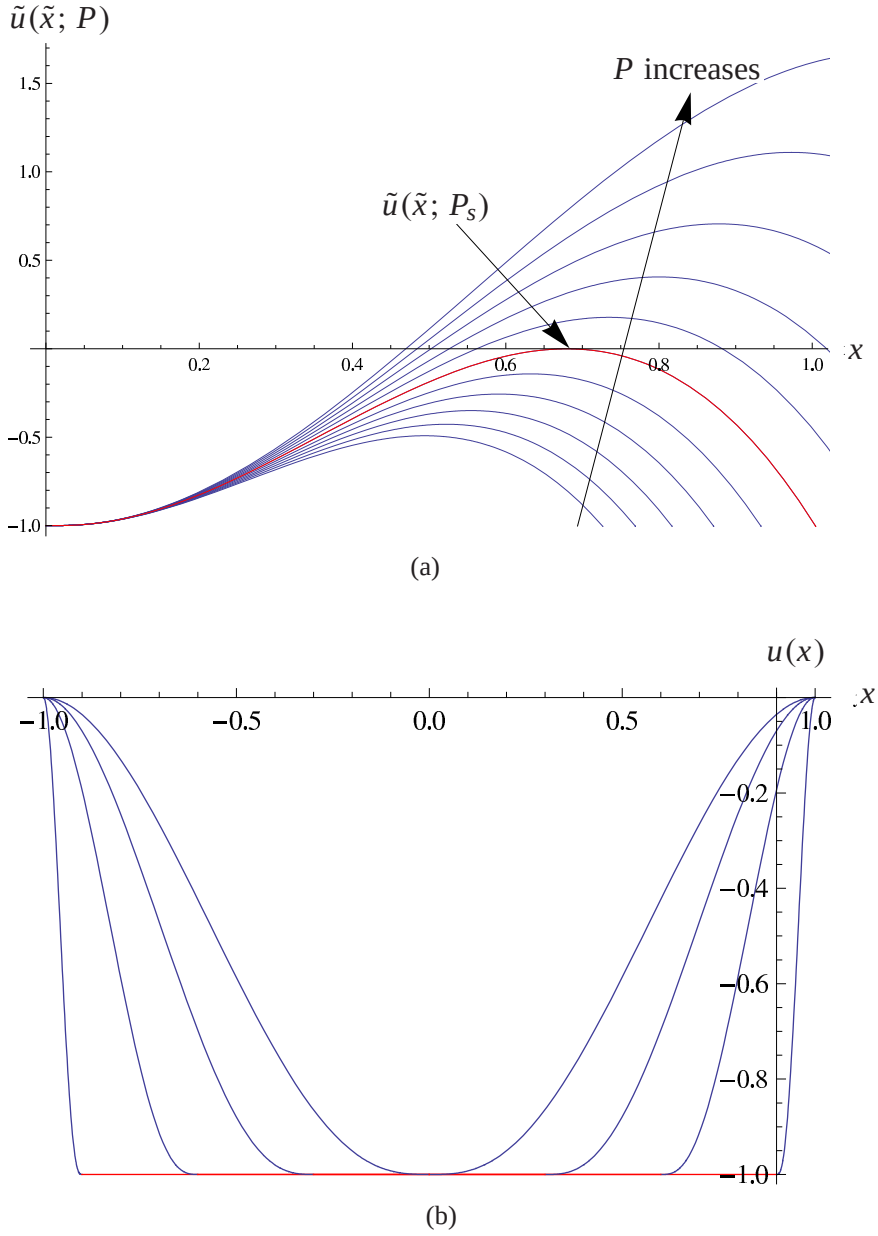


Figure 4.3: (a) The function $\tilde{u}(\tilde{x};\, P)$ for different values of the shooting parameter P and $V = 890$. Here, $\tilde{u}(\tilde{x};\, P)$ increases as P increases. The red curve corresponds to a solution $\tilde{u}(\tilde{x};\, P_s)$ which satisfies (4.11) and solves (4.6) and (4.7). (b) The membrane shape for different values of V. The red line depicts a part of membrane in contact with dielectric. The blue curve is the shape of the membrane between the support and the dielectric.

Next, we present an alternative method for solving (4.6) and (4.7). This method is convenient for fast construction of a CV-curve because it requires solving a boundary value problem only once. Then using a scaling argument we get an easily calculable expression for C.

First we rescale $\tilde{x}$ according to $\hat{x} = \tilde{x}/(1-a)$ and $\hat{u}(\hat{x}) = \tilde{u}(\tilde{x})$. Then the boundary value problem (4.6) and (4.8) becomes

$$\hat{u}'''' = -\frac{\epsilon_0 V^2 (1-a)^4}{1-\eta+\hat{u}}, \tag{4.12}$$

$$\hat{u}(1) = 0, \quad \hat{u}'(1) = 0, \quad \hat{u}(0) = -1, \quad \hat{u}'(0) = 0, \quad \hat{u}''(0) = 0. \tag{4.13}$$

To solve (4.12) and (4.13) using the shooting method routine implemented in MATHEMATICA 6 we rewrite (4.12) as follows

$$\hat{u}''''(\hat{x}) = -\frac{\epsilon_0 \hat{V}(\hat{x})^2}{1-\eta+\hat{u}(\hat{x})}, \quad \hat{V}'(\hat{x}) = 0. \tag{4.14}$$

Here the unknown $V^2(1-a)^4$ is described as an unknown constant function $\hat{V}(\hat{x})$. A solution $\hat{u}(\hat{x})$ and $\hat{V}(\hat{x}) = V_s$ of (4.14) describes the shape of the membrane $u(x) = \hat{u}(x)$ for $a = 0$, and V_s is the minimum value of V for which (4.6) and (4.7) has a solution. A solution $u(x)$ for arbitrary $V > V_s$ is written as

$$u(x) = \hat{u}((x-a)/(1-a)), \quad a = 1 - \sqrt{\frac{V_s}{V}}.$$

The shape of the membrane is

$$u(x) = \begin{cases} \hat{u}((|x|-a)/(1-a)), & \text{for } a < |x| \le 1, \\ -1, & \text{for } |x| \le a, \end{cases}$$

see Figure 4.3(b).

With increasing V the contact with the dielectric increases and the membrane shape between the support and the dielectric becomes steeper.

The value of C is computed from (4.5) as

$$C(V) = \frac{2\Lambda\epsilon_0}{g}\left(\frac{1-\sqrt{V_s/V}}{\eta} + \sqrt{\frac{V_s}{V}} I_1\right), \quad \text{where} \quad I_1 = \int_0^1 \frac{d\hat{x}}{1+\hat{u}(\hat{x})+\eta},$$

from which follows that $C(V)$ has a horizontal asymptotic

$$\lim_{v\to\infty} C(V) = \frac{2\Lambda\epsilon_0}{g\eta}. \tag{4.15}$$

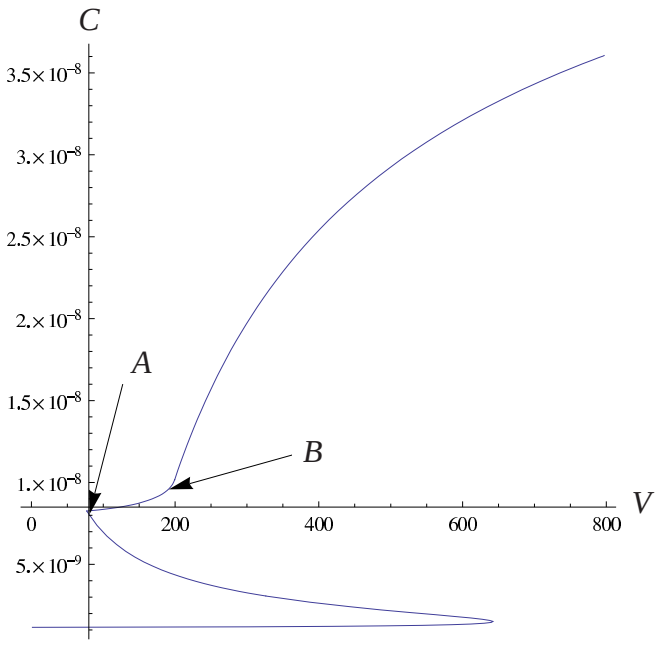


Figure 4.4: CV-curve for all three cases. The membrane first touches the dielectric at the point A. The change between the situations when the membrane touches the dielectric at one point, and on some interval is indicated by the point B.

CV-curve and influence of model parameters

Summarizing the results of the CV-curves for all three situations, we construct the CV-curve for all V, see Figure 4.4. The complete CV-curve has discontinuous derivative at the transition point when the membrane touches the dielectric for the first time (point A in Figure 4.4). At the transition between the situations when the membrane touches the dielectric at one point and on some interval (point B in Figure 4.4), the CV-curve is C^1. For some interval of V three values of C are possible (see Figure 4.4). This is a consequence of the non-uniqueness of the solution to the original problem for u.

4.3 The continuation problem

AUTO is a software package that is used for finding and displaying solutions, and tracking bifurcations of solutions of ordinary differential equations (ODEs) by continuation of some system parameter.[8] A bifurcation is, loosely formulated, a sudden change in the qualitative behaviour of ODEs when some system parameter (or *bifurcation parameter*) crosses a certain threshold (the *critical value*). For example,

[8] The package has been developed initially by E. Doedel and subsequently expanded by a range of authors, see [3]

an equilibrium solution may loose stability when the bifurcation parameter crosses a critical value. For more on the notion of bifurcation, see [6].

By continuation we mean the process of changing this system parameter and calculating the deformation of a solution when this parameter is changed. A typical continuation starts out with some (acquired) solution for the system with a certain value for the system parameter. Then the parameter is changed, and the solution is calculated for each value of the parameter. AUTO also detects bifurcations when they take place. So, in order to do a continuation, one has to find one solution for a specific value of the bifurcation parameter (often zero is a smart choice). By changing a parameter (i.e. by a *continuation* in one of the parameters) the solution generically changes as well. This solution can be found by AUTO, for each value of the bifurcation parameter.

Most continuation software, and especially AUTO, allows for continuation in two or more parameters as well. AUTO is not only able to perform continuation of equilibria to ODEs, but also the continuation of periodic solutions of ODEs, fixed points of discrete dynamical systems, and even solutions to partial differential equations (PDEs) that can in some sense be transformed to ODEs, like spatially uniform solutions (i.e. solutions that do not depend on any spatial variable) of a system of parabolic[9] partial differential equations (parabolic PDEs), travelling wave solutions to a system of parabolic PDEs, and even more.

It is presently not of our interest *how* AUTO finds this solution. For convenience, we only note here that all continuation methods basically rely upon some version of Newton's method (and therefore the Implicit Function Theorem).

We want to stress that continuation always leads to a (discretized) *continuum* of solutions. This is an advantage with respect to the other numerical methods we described so far. Moreover, a continuation and bifurcation package such as AUTO is able to detect bifurcations of the system as well. This is the second main advantage.

We show the method of continuation applied to our equilibrium problem which consists of a nonlinear ordinary differential equation which is difficult to solve analytically. The nonlinear differential equation for which the voltage V and capacitance C are calculated, reads

$$\frac{\partial^4 u}{\partial x^4} = -\frac{V^2}{2}\frac{\epsilon_0}{(u+d/\epsilon)^2} + \alpha k_1 \mathrm{e}^{-k_2 u} \tag{4.16}$$

with $u'(0) = u'(1) = u(0) = u(1) = 0$ and α a dummy parameter to switch between nonlinear $\alpha = 1$ and linear problem $\alpha = 0$. Setting $\alpha = 0$, the associated linear problem is obtained as

$$\frac{\partial^4 u}{\partial x^4} = -\frac{V^2}{2}\frac{\epsilon_0 \epsilon}{d} \tag{4.17}$$

with $u''(0) = u''(1) = u(0) = u(1) = 0$.

[9] We do not explain the notion of a *parabolic* PDE here; it is of no importance to us. But see any introductory text on partial differential equations

By solving the above linear equation, AUTO knows a solution of the "nonlinear" problem for $\alpha = 0$. By continuation in α, it subsequently finds solutions for the nonlinear problem with $\alpha \neq 0$. For each of these solutions the capacitance C and voltage V are calculated and a CV-curve is plotted in Figure 4.5. The CV-curve in Figure 4.5 exhibits a hysteresis behaviour.

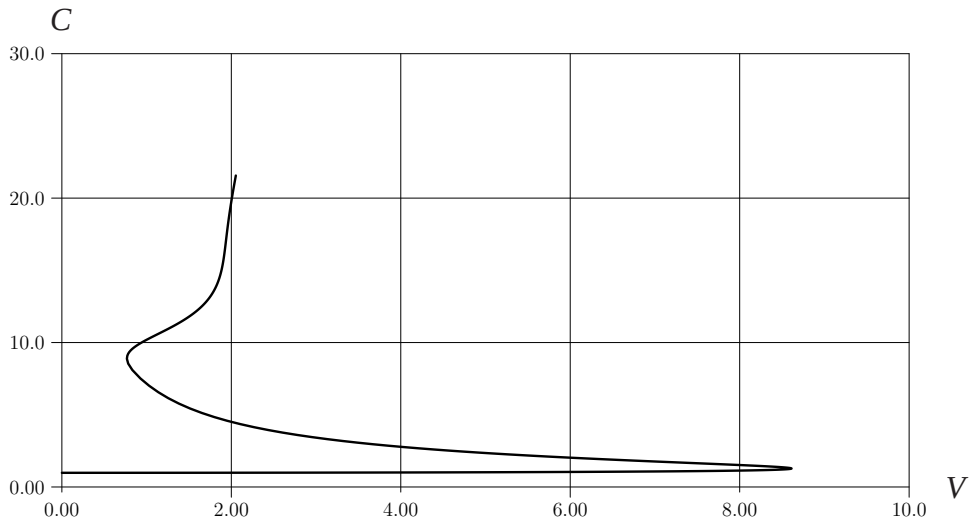


Figure 4.5: CV-curve generated by AUTO.

4.4 Analytical results

4.4.1 The linearized problem

It is possible to fully solve the linearized problem for three different cases: (i) the case in which the membrane does not touch the dielectric at all (ii) the case in which the membrane touches the dielectric in one point only and (iii) the case in which the membrane touches the dielectric on an interval. Since most linear problems have unique solutions, it is clear from the outset that the typical hysteresis behaviour does not show up in the linearized model. Some of those calculations may nevertheless be of interest, we have placed a summary of the linearized problem in the appendix

4.4.2 Collected analytical results

We prove some results for a functional E that may be interpreted as the total energy. The functional can be written as an integral over some domain Ω in $\mathbb{R}^2$. To read this section, it might be necessary to consult a text on variational methods, see for example [4] or [5].

First, it is proved that the solution for the membrane cannot touch the dielectric "with holes", i.e. in one dimension, the membrane is stuck to the dielectric between

every two points where the membrane touches it. Second, it is derived that every critical point u for which $u = 0$ on some open set $\Omega_1 \subset \Omega$, has $\Delta u = 0$ on $\partial\Omega_1$. Third, we prove that stationary solutions for the energy E for which it holds that $dC/dV < 0$ are necessarily unstable. The final result is argued but still not completely proved. It states that if for both large V and small V a unique critical point exists, then under some conditions on the energy functional, a continuous family of solutions connects the two solutions.

Just for notation's sake: the main functional we consider is

$$E = \frac{D}{2}\int_\Omega u''^2 - \frac{V^2}{2}\int_\Omega \frac{\epsilon_0}{(u + d/\epsilon)} + \int_\Omega k_1 \mathrm{e}^{-k_2 u}, \tag{4.18}$$

where Ω is a domain (e.g. a rectangle, or a circle) in $\mathbb{R}^2$ or an interval in $\mathbb{R}$, depending on the question considered. The second integral is the *capacity*

$$C = \int_\Omega \frac{\epsilon_0}{(u + d/\epsilon)}.$$

The boundary conditions are $u = g$ and $\partial u/\partial n = 0$ on $\partial\Omega$. Unless stated otherwise, all integrals are over Ω.

Short list of results

1. For any minimizer (or general critical point) u of the *infinitely-hard bottom* problem

$$\min\left\{\frac{D}{2}\int \Delta^2 u - \frac{V^2}{2}C \;\middle|\; u \geq 0\right\} \tag{4.19}$$

   there exists *no* nonempty open sets $\Omega_1 \subset \Omega$ satisfying $u|_{\Omega_1} > 0$ and $u|_{\partial\Omega_1} = 0$. In particular, in dimension $n = 1$, the contact set $\{x \in \Omega \mid u(x) = 0\}$ is a (possibly empty) interval; in two-dimensions it means that the contact set has only simply connected components (no rings).

2. If u is minimizer of (4.19), or more generally a critical point, then if $u = 0$ on an open set $\Omega_1 \subset \Omega$, then $\Delta u = 0$ on $\partial\Omega_1$ (also of course on the interior of Ω_1).

3. Stationary points of E lying on a branch for which $dC/dV < 0$, are necessarily *unstable*. That is, there exists a perturbation w such that

$$E''(u) \cdot w \cdot w < 0.$$

   More generally, consider energies of the form

$$F(u, C, V) = \int f(x, u, \nabla u, \Delta u, \dots)dx + G(V, C),$$

where $C = \int c(u(x))dx$ and V is a parameter (the Voltage for example). Then if $\partial^2 G/\partial C \partial V < 0$, any solution lying on a branch for which $dC/dV < 0$, is unstable.

4. The last result is more tentative; it should be true, but requires additional work to prove: if there exists a unique critical point of E both for small V and for large V, and provided that some type of coercivity holds for the energy functional (4.18), then there exists a continuous family of solutions connecting these two.

Sketches of the proofs

Ad 1 In 1D: let u be a stationary point, satisfying, where $u > 0$,

$$-Du'''' = \frac{\epsilon_0}{(u + d/\epsilon)^2}. \tag{4.20}$$

Note that the right hand side of (4.20) is strictly positive. Suppose u has two contact points $x_1 < x_2$. Since $u(x_i) = u'(x_i) = 0$ and $u \geq 0$, we must have $u''(x_i) \geq 0$. Furthermore, $-(u'')'' > 0$ and it follows from the maximum principle that $u''(x) \geq \min\{u''(x_1), u''(x_2)\} \geq 0$ for $x \in (x_1, x_2)$. This implies, again by the maximum principle, that $u \leq 0$ on (x_1, x_2). We thus conclude that $u \equiv 0$ on $[x_1, x_2]$.

In more dimensions exactly the same (pair of maximum principle) arguments prove that the contact region can have no holes, as asserted.

Ad 2 We do not give a full proof, but illustrate the main idea. On a one-dimensional domain $\Omega = [-L, L]$, let $u_R(x)$ be a smooth family of symmetric solutions with "forced" contact region $[-R, R]$, with $R < L$. By symmetry, we only need to consider the left half of the solution:

$$\begin{cases} -Du_R'''' = \frac{\epsilon_0}{(u_R + d/\epsilon)^2} & \text{for } -L < x < -R, \\ u_R(-L) = g,\ u_R'(-L) = 0, \\ u_R(-R) = 0,\ u_R'(-R) = 0. \end{cases}$$

Now, u_R is a critical point of E if and only if $dE(u_R)/dR = 0$.

Writing $E(u) = \int_\Omega \frac{D}{2}u''^2 + g(u)dx$, where $g(u) = -\frac{V^2}{2}\frac{\epsilon_0}{(u+d/\epsilon)} + k_1 e^{-k_2 u}$, we obtain

$$E(u_R) = 2\int_{-L}^{-R} \frac{D}{2} u_R''^2 dx + 2\int_{-L}^{-R} g(u_R)dx + 2Rg(0).$$

Calculating this derivative with respect to R we infer that

$$\frac{dE(u_R)}{dR} = E_u(u_R)\frac{\partial u_R}{\partial R} - Du_R''^2(-R) - 2g(u_R(-R)) + 2g(0) = -Du_R''^2(-R),$$

since $u_R(-R) = 0$ and $E_u(u_R) = 0$, because u_R is a critical point when keeping R *fixed*. It follows that $u''(-R) = 0$ if u is a critical point of E.

This argument can be extended to higher dimensions quite easily, under the *assumption* that the solution for fixed contact region varies smoothly with the geometry of the contact region. Without this assumption, more complicated arguments are needed.

Remark 4.4.1. We note that if the contact set is a single point, then the second derivative in this point need *not* be zero. Indeed, in one dimension for example, there is a branch of solutions with contact only in the midpoint of the domain (interval) and with varying second derivative.

Ad 3 Let us first give the argument for the specific energy E in the one-dimensional case. Consider

$$\frac{D}{2}\int u''^2 - \frac{V^2}{2}\int \frac{\epsilon_0}{(u+d/\epsilon)} + k_1 \mathrm{e}^{-k_2 u}.$$

Let us look at stationary points, i.e., solutions of

$$Du'''' = -\frac{V^2}{2}\frac{\epsilon_0}{(u+d/\epsilon)^2} + k_1 k_2 \mathrm{e}^{-k_2 u}, \tag{4.21}$$

which are, on the branch under consideration, parametrized by V. Let us write $\overline{u}$ for the derivative of the solutions u with respect to V along the branch. Taking the derivative of (4.21) along the branch, we obtain

$$D\overline{u}'''' = V^2\frac{\epsilon_0\overline{u}}{(u+d/\epsilon)^3} - V\frac{\epsilon_0}{(u+d/\epsilon)^2} - k_1 k_2^2 \mathrm{e}^{-k_2 u}\overline{u}. \tag{4.22}$$

The second variation of the energy in the direction $\overline{u}$ gives

$$E''(u)\cdot\overline{u}\cdot\overline{u} = D\int \overline{u}''^2 - V^2\int \frac{\epsilon_0\overline{u}}{(u+d/\epsilon)^3} + k_1 k_2^2\int \mathrm{e}^{-k_2 u}\overline{u}^2.$$

After performing partial integration twice on the first term, we can substitute (4.22) and, with most terms cancelling, we obtain

$$E''(u)\cdot\overline{u}\cdot\overline{u} = -V\int \frac{\epsilon_0\overline{u}}{(u+d/\epsilon)^2}.$$

This simplifies as

$$E''(u)\cdot\overline{u}\cdot\overline{u} = -V\int \frac{\epsilon_0\overline{u}}{(u+d/\epsilon)^2} = VC'(u)\overline{u} = V\frac{dC}{dV}.$$

Hence $\frac{dC}{dV} < 0$ implies that u is unstable.

For the general case, critical points $u = u(V)$ satisfy, subscripts denoting partial derivatives,

$$F_u(u(V))\cdot w + G_C(V, C(u(V)))\, C_u(u(V))\cdot w = 0 \qquad \text{for any } w.$$

Taking the derivative with respect to V gives (always evaluating at $u = u(V)$),

$$F_{uu} \cdot w \cdot u_V + G_{CV}\, C_u \cdot w + G_{CC}\, C_u \cdot w\, C_u \cdot u_V + G_C\, C_{uu} \cdot w \cdot u_V = 0 \quad \text{for any } w. \tag{4.23}$$

On the other hand, the second variation (for fixed V) gives

$$F''(u) \cdot w \cdot w' = F_{uu} \cdot w \cdot w' + G_{CC}\, C_u \cdot w\, C_u \cdot w' + G_C\, C_{uu} \cdot w \cdot w',$$

hence, using (4.23), we obtain for $w = w' = u_V$

$$F''(u) \cdot u_V \cdot u_V = -G_{CV}\, C_u \cdot u_V.$$

When we write $\overline{C}(V) = C(u(V))$, this reduces to

$$F''(u) \cdot u_V \cdot u_V = -G_{CV} \frac{d\overline{C}}{dV}.$$

Hence, if $\partial^2 G/\partial C \partial V < 0$ then solutions on branches where $d\overline{C}/dV < 0$ are always unstable.

Remark 4.4.2. One can also consider the problem where we put a charge $Q = VC$ on the switch. In that case the physically relevant energy does *not* include the energy stored in the battery, which is given by $-V^2 C$. The energy E_Q thus becomes

$$E_Q = \frac{D}{2} \int u''^2 + \frac{Q^2}{2C} + \int k_1 \mathrm{e}^{-k_2 u},$$

and the arguments above show that solutions on curves with $dC/dQ < 0$ are always unstable.

Ad 4 Such a result follows from degree theory, see e.g. [8]. However, it still needs to be checked rigorously that there indeed does exist a unique critical point for very large and very small V. For $V = 0$ this is obvious, the energy being convex in that case, but the situation for large V is less straightforward, since the energy contains both convex and concave parts, although in numerical experiments uniqueness is observed.

4.4.3 Functional estimates

In this section, two estimates for the first and second variation of the total energy are derived.

The energy functional modeling the deformation of a clamped plate Ω under influence of an electrical field due to a potential difference with a fixed plate reads

$$\begin{aligned} E[u] &= E_{\text{mech}} + E_{\text{el}} \\ &= \int_\Omega \left[\frac{1}{2} D(\Delta u)^2 - \frac{1}{2} V^2 \frac{\varepsilon_0}{u + g + \frac{d}{\varepsilon_0}} \right] \mathrm{d}x\,\mathrm{d}y, \end{aligned} \tag{4.24}$$

where $u \in H_0^2(\Omega)$, implying $u = \frac{\partial u}{\partial n} = 0$ on $\partial\Omega$. Equilibria for the system are the zeroes of the first variation,

$$\delta E[\tilde{u}, h] = 0, \quad \forall h \in H_0^2(\Omega).$$

These can be stable and unstable equilibria (e.g. saddle points). A stable equilibrium is a minimum of the functional. Such states are characterized by the fact that the second variation at the equilibrium state is strictly positive,

$$\delta^2 E[\tilde{u}, h] > 0, \quad \forall h \in H_0^2(\Omega)$$

We here wish to give a sufficient condition for an equilibrium to be stable.

The first variation is found by putting $u = \tilde{u} + \varepsilon h$, where $h \in H_0^2(\Omega)$ is a test function, and taking the derivative with respect to. ε at $\varepsilon = 0$. We then obtain

$$\delta E[\tilde{u}, h] = \int_\Omega \left[D\Delta^2 \tilde{u} + \frac{1}{2} V^2 \frac{\varepsilon_0}{(u + g + \frac{d}{\varepsilon_0})^2} \right] h \, \mathrm{d}x\mathrm{d}y.$$

The variation lemma yields the boundary value problem for the system from this functional. Let us assume we have a solution for the system. Now the question is whether the solution is stable or not. The second variation in the direction $h \in H_0^2(\Omega)$ is found to be

$$\delta^2 E[u, h] = \int_\Omega \left[D(\Delta h)^2 - V^2 \frac{h^2 \varepsilon_0}{(u + g + \frac{d}{\varepsilon_0})^3} \right] \mathrm{d}x\mathrm{d}y. \tag{4.25}$$

In this form it is difficult to check positivity. However, we can prove a Cauchy-type inequality for the test functions in the space $H_0^2(\Omega)$ when Ω has a simple shape. For the case of a rectangle with sides L_1 and L_2 we have

$$\int_\Omega (\Delta h)^2 \mathrm{d}x\mathrm{d}y \geq \frac{4}{\max[L_1^4, 2L_1^2 L_2^2, L_2^4]} \int_\Omega h^2 \mathrm{d}x\mathrm{d}y$$

Using this inequality together with equation (4.25) we have the following estimate for the second variation,

$$\delta^2 E[u, h] \geq \int_\Omega \left[\frac{4D}{\max[L_1^4, 2L_1^2 L_2^2, L_2^4]} - V^2 \frac{\varepsilon_0}{(u + g + \frac{d}{\varepsilon_0})^3} \right] h^2 \mathrm{d}x\mathrm{d}y$$

Necessary conditions for the stability of the functional can be obtained from this estimate. For example, take $u^* = \min(u)$, then

$$\delta^2 E[u, h] \geq \left[\frac{4D}{\max[L_1^4, 2L_1^2 L_2^2, L_2^4]} - V^2 \frac{\varepsilon_0}{(u^* + g + \frac{d}{\varepsilon_0})^3} \right] \int_\Omega h^2 \mathrm{d}x\mathrm{d}y,$$

and it is sufficient to check the positivity of the constant.

Appendix

As said in section 4.4.1, the linearized problem can be fully elaborated for three different cases: (i) the case in which the membrane does not touch the dielectric at all (ii) the case in which the membrane touches the dielectric at $x = 0$ only and (iii) the case in which the membrane touches the dielectric on an disc $\bar{B}_a$, for $0 < a < 1$. We will make a few remarks on how to do this, in the case of a radially symmetric MEMS switch. We consider case (iii). It will be clear from the result that the typical hysteresis behaviour does not show up in the linearized model.

To focus on the right parameter combinations in the problem, we rescale it. For example, in the 2-D radially symmetric version of the problem one obtains for the capacitance:

$$C(w) = \frac{2\pi\epsilon_0\Lambda^2}{g}\int_0^1 \frac{r}{1+\eta+w(r)}\,dr.$$

and, by computing the Euler-Lagrange equation corresponding to this energy we find

$$\Delta_r^2 w = -\frac{\delta v^2}{(1+\eta+w)^2}. \tag{4.26}$$

where $\Delta_r = \frac{1}{r}\frac{d}{dr}$, δ some algebraic expression in terms of the other parameters, v a non-dimensionalized voltage and w a scaled version of the distance u. This problem can subsequently be linearized around $w = 0$:

$$\Delta^2 w = \omega^4\left(-\frac{1+\eta}{2}+w\right),$$

where $\omega = \sqrt[4]{\frac{2\epsilon v^2}{(1+\eta)^3}}$ is just a scaling. Regarding w as a radially symmetric function depending on r only we get

$$w(r) = AJ_0(\omega r) + BY_0(\omega r) + CI_0(\omega r) + DK_0(\omega r) + \frac{1+\eta}{2}, \tag{4.27}$$

where J_0 and Y_0 are Bessel functions of the first and second kind respectively and I_0 and K_0 are modified Bessel functions of the first and second kind. We add the following boundary conditions:

$$w(1) = w'(1) = w'(a) = w''(a) = 0, \quad w(a) = -1.$$

By rewriting this system as a four-dimensional first-order system, one obtains the constants A, B, C and D.

Bibliography

[1] J. Bielen and J. Stulemeijer. *Proc. Eurosime*, 2007.

[2] F. Bin and Y, Yang. *Proc. R. Soc. A*, 463, p.1323, 2007.

[3] E. Doedel, R.C. Paffenroth, A.R. Champneys, T.F. Fairgrieve, Y.A. Kuznetsov, B.E. Oldeman, B. Sandstede and X. Wang. AUTO2000: Continuation and bifurcation software for ordinary differential equations. *Technical Report, Concordia University*, 1997.

[4] Lawrence C. Evans. Partial differential equations. *American Mathematical Society*, 1998.

[5] I.M. Gelfand and S.V. Fomin. Calculus of variations. *Dover publications*, 2000.

[6] J. Hale and H. Koçak. Dynamics and Bifurcations. *Springer-Verlag, New York, Inc.*, 1996.

[7] J.A. Pelesko and D.H. Bernstein. Modeling MEMS and NEMS. *Chapman & Hall/CRC*, 2003.

[8] Paul H. Rabinowitz. Some global results for nonlinear eigenvalue problems. *J. Functional Analysis*, 7:487-513, 1971.

[9] G.M. Rebeiz. RF MEMS: Theory, Design and Tenchnology. *John Wiley and Sons*, 2004.

[10] S.D. Senturia. Microsystem Design. *Springer*, 2001.

[11] Josef Stoer and Roland Bulirsch. Introduction to Numerical Analysis. *New York: Springer-Verlag*, 1980.

[12] P.G. Steeneken, Th.G.S.M. Rijks, J.T.M. van Beek, M.J.E. Ulenaers, J. de Coster, R. Puers. Dynamics and squeeze film gas damping of a capacitive RF MEMS switch. *J. Micromech. Mircoeng.* 15, 176, 2005.

[13] Peter Steeneken, Hilco Suy, Rodolf Herfst, Martijn Goossens, Joost van Beek, Jurriaan Schmitz. Micro-elektromechanische schakelaars voor mobiele telefoons. *Nederlands Tijdschrift voor Natuurkunde*, p. 314-317, September 2007.

[14] Jiri Stulemeijer, Jan Bouwe van den Berg, Jeroen Bielen and Peter G. Steeneken. Numerical path following as method for modeling electrostatic MEMS. *Preprint NXP*, 2008.

5 Increasing Detection Performance of Surveillance Sensor Networks

Nelly Litvak[1*] Muhammad Umer Altaf[2] Alina Barbu[2]
Sudhir Jain[3] Denis Miretskiy[1] Leila Mohammadi[4]
Ertan Onur[2] Jos in 't panhuis[5] Julius Harry Sumihar[2]
Michel Vellekoop[1] Sandra van Wijk[5] Rob Bisseling[6†]

Abstract

We study a surveillance wireless sensor network (SWSN) comprised of small and low-cost sensors deployed in a region in order to detect objects crossing the field of interest. In the present paper, we address two problems concerning the design and performance of an SWSN: optimal sensor placement and algorithms for object detection in the presence of false alarms. For both problems, we propose explicit decision rules and efficient algorithmic solutions. Further, we provide several numerical examples and present a simulation model that combines our placement and detection methods.

Keywords: sensor deployment, detection probability, overlap, hypothesis testing, Bayesian approach, hidden Markov models, Viterbi algorithm, simulations.

5.1 Introduction

An important class of wireless sensor networks (WSN) is the WSNs comprised of small and low-cost sensors with limited computational and communication power [1]. Sensors are deployed in a region, they wake up, organize themselves as a network,

[1]University of Twente
[2]Delft University of Technology
[3]Aston University, Birmingham, UK
[4]Leiden University Medical Center
[5]Eindhoven University of Technology
[6]Utrecht University
[*]corresponding author, n.litvak@ewi.utwente.nl

[†]Thanks to other participants who helped during the week. We would like to thank Y. Boers, H. Driessen and P. Verveld from THALES Nederland B.V., Hengelo, for useful discussions during the week and for the help with preparation of the manuscript.

and start sensing the area. The objective of the sensors is sensing the environment and communicating the information to a data collection center. Many types of employment are envisaged for WSNs ranging from the monitoring of endangered animal populations to military surveillance or the surveillance of critical infrastructures [12], archaeological sites [2], perimeters, or country borders [10]. The tasks of a surveillance wireless sensor network (SWSN) is to detect objects crossing the field of interest. The sensors monitor the environment and send reports to a central control unit. The major requirement of a surveillance application is that the SWSN is to monitor the environment with a certain quality for a specific period of time. Important issues in designing an SWSN are the deployment decisions such as the sensing range of sensor nodes and density of the SWSN, and deployment strategy (random, regular, planned, et cetera.) to be applied [10].

Different types of sensors may have to be utilized in a WSN to address the problem at hand. For outdoor surveillance systems, radar, microwave, ultrasonic and/or infrared sensors are typical. To analyze the detection performance of the sensors or the surveillance systems, a common measure such as the single-sensor detection probability p may be utilized since it allows to abstract the different working principles of the sensors. The factors that affect p are the object-to-sensor distance, environmental characteristics, the size and the motion pattern of the object, et cetera. Moreover, He et al. [7, 8] showed that sensors produce a non-negligible amount of false alarms. The false alarms are defined as positive reports of a sensor when no object exists. Each sensor may produce a false alarm with a certain probability q. If data/decision fusion [5] is allowed, then the false alarm probability q negatively affects the detection performance of the network. The cost of false alarms varies depending on the application. For example, it is lower in a home surveillance system when compared to the cost of false alarms in a surveillance application of mission-critical infrastructure such as a nuclear reactor. Hence, the objective of an upstanding SWSN design is to maximize the detection probability of the system while minimizing or bounding the false alarm rate of the system. To this end, in the present paper, we study two problems concerning the design and performance of an SWSN: optimal sensor placement and algorithms for intruder detection in the presence of false alarms. Our main performance characteristics of the SWSN are the system's intruder detection probability and false alarm probability, for given input parameters p and q representing single-sensor probabilities. The problem of correctly communicating the reports of the sensors to the central control unit (with possibly additional failure probabilities) is beyond the scope of the present study. It has been studied elsewhere, among others in a previous study group Mathematics with Industry [9]. Therefore, we will assume perfect communication of the reports.

The sensor placement problem addressed in this work is formulated as follows: given a limited number of homogeneous sensors with an effective sensing range r and a field of interest modelled as a one- or two-dimensional area, determine the optimal location of the sensors that maximizes the detection performance of the SWSN. In Section 5.2.1, we study the trade-off involved in overlapping sensor

ranges. If the number of sensors is limited then, clearly, overlaps decrease the total sensing part of the area but increase the detection performance in the overlap of two or more sensor ranges. We give an explicit condition when overlap in sensor ranges leads to better detection performance of the system. Next, in Section 5.2.2 we propose an algorithm for efficient coverage of a 2-D area, based on a priori knowledge on the probability distribution of the intruder position. When the distribution of an object's location in the area is uniform, our algorithm performs closely to the optimal hexagonal placement.

Given a particular layout of the sensors, the probability of intruder detection and the false alarm probability of the network depend on the decision rule that prescribes in which situation an intrusion alarm has to be reported, based on observations from all deployed sensors. For instance, if we have two completely overlapping sensors and report an intrusion alarm only if both sensors signal an intruder, then the intruder detection probability of the SWSN is p^2 and the false alarm probability is q^2. The problem is to determine a decision rule for reporting an intrusion alarm such that the detection performance of the network is maximized. In Section 5.3 we attempt to resolve this problem by statistical methods. Our main conclusion is that several observations of the same object are absolutely necessary to report an intrusion alarm with a reasonable confidence. However, multiple observations will result in a huge variety of observed patterns. Which patterns signal the intruder and which are caused by false alarms only? This question is tackled in Section 5.4 where we design a procedure for intruder detection, based on hidden Markov models and the Viterbi algorithm.

Finally, in Section 5.5 we present a simulation model that combines our placement and detection methods. Using this model, we characterize the detection performance in several configurations of a detection area.

Throughout the paper, we use the following notations:

- p – single-sensor detection probability, the probability that a sensor signals an object given that the object is present in the sensing range (assumed to be a circle, or sphere);
- q – the single-sensor false alarm probability, the probability that a sensor signals an intruder given that there is no intruder in the sensing range;
- r – sensing radius of a sensor;

Further, a random variable $X \in \{0, 1\}$ is an indicator of the event that an object is present in the sensing range of a sensor, and a random variable $Y \in \{0, 1\}$ is an indicator of the event that a sensor gives an alarm. We will also assume that the alarm events of individual sensors are mutually independent when conditioned on the absence or presence of the object.

5.2 Optimal coverage of the area

In this section we study the problem of optimal sensor placement, or sensor deployment, formulated as follows. Consider an area where a number of sensors are to be deployed, and assume that there is an object in the area. We define as $p_{\text{detection}}$ the probability that at least one sensor correctly detects the object. The goal is to find a sensor deployment maximizing $p_{\text{detection}}$. In order to compute $p_{\text{detection}}$, throughout the section we assume an a priori statistical knowledge on the object position.

One natural solution to this problem is to maximize the coverage of the observed area for a given number of sensors, or, equivalently, minimize the number of sensors employed while covering the complete area. If each sensor has a range with radius r, then we model the sensing area as a circle of radius r with a center at the sensor location. Thus, the question of minimizing the number of sensors while covering the complete area is equivalent to the so-called *covering problem* in two dimensions: cover a given area completely with the least amount of circles with a given fixed radius. This problem (and many others like the packing and kissing problems) is solved by using the hexagonal lattice, defined as the set of points $\lambda v + \mu w$, $\lambda, \mu \in \mathbb{Z}$, where $v = (1, 0)$ and $w = (1/2, \sqrt{3}/2)$ are the vectors spanning the lattice. To cover an area with circles of radius r, the vectors v, w must be scaled by a factor $r\sqrt{3}$. In the asymptotic limit, with a large area covered by sensors and with negligible boundary effects, the sum of the sensor ranges is 1.209 times the covered area, meaning that about 20.9% of the area is covered by two sensors and the remainder by one sensor. For further details, see [4]. An example of 7 sensors placed by using the hexagonal lattice and completely covering a rectangular area is given in Figure 5.1. An example of hexagonal placement of 105 sensors with non-covered gaps in between is given in Figure 5.3.

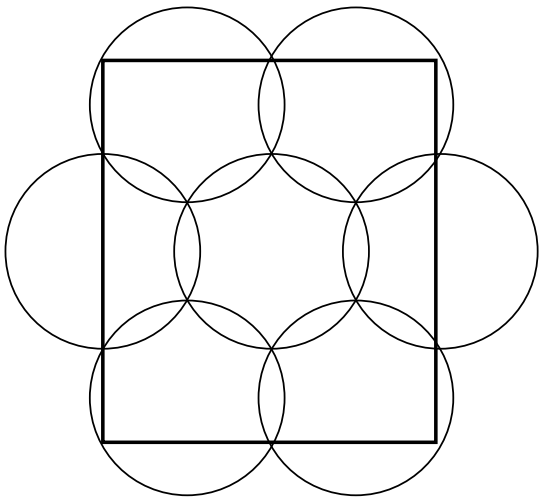


Figure 5.1: Rectangular area covered by seven sensors placed by using a hexagonal lattice.

Intuitively, a sensor placement with minimal overlapping or without overlapping must be optimal if the distribution of the object's position is uniform. Below in Section 5.2.1 we show that this is often the case also for non-uniform distributions, and in Section 5.2.2 we propose a procedure for close-to-optimal sensor placement.

5.2.1 Optimal allocation of two sensors

Does it make sense to let two sensors overlap? Having some overlap might be reasonable if we want a better detection in most vulnerable regions. However, if the number of sensors is limited then overlaps reduce the total coverage. In order to resolve this trade-off, we consider the following simple model. We restrict ourselves to a one-dimensional area, which constitutes an interval of length two, and a pair of sensors with $r = 1/2$. For each of the two sensors, the detection probability is p and the probability of a false alarm is q. The question is how to place these sensors so that the detection probability $p_{\text{detection}}$ is maximized.

Formally, let $S = [0, 2]$ be the area under surveillance. Denote by x_1 the leftmost point of the first sensor's coverage and by x_2 the leftmost point of the second sensor's coverage. Thus, the first sensor covers the segment $S_1 = [x_1, x_1 + 1]$ and the second one covers the segment $S_2 = [x_2, x_2+1]$, where $x_1 \in [0, 1]$ and $x_2 \in [x_1, 1]$, as shown in Figure 5.2.

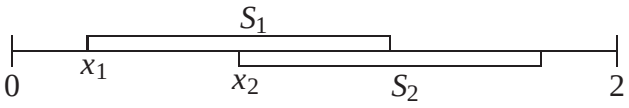


Figure 5.2: Partial overlapping of two sensors.

Now assume that the intruder location L has a distribution $P(L \leq x) = F(x)$, $x \in [0, 2]$. Then in the doubly covered segment $S_1 \cap S_2$ the detection probability by the two-sensor system is $p^2 + 2p(1 - p)$, and the object is in this segment with probability $F(x_1 + 1) - F(x_2)$. In the singly covered segment $(S_1 \cup S_2) \setminus (S_1 \cap S_2)$ detection probability is p, and the object is there with probability $F(x_2 + 1) - F(x_1 + 1) + F(x_2) - F(x_1)$. Finally, in the remaining uncovered part $S \setminus (S_1 \cup S_2)$ the detection probability is 0.

Rearranging the terms, we can formulate the problem of maximizing the detection probability $p_{\text{detection}}$ as follows:

$$\begin{aligned} &\max_{x_1,x_2}\{p_{\text{detection}}(x_1, x_2)\} \\ &\quad = \max_{x_1,x_2}\{p\,(F(x_2 + 1) - F(x_1)) + p(1 - p)\,(F(x_1 + 1) - F(x_2))\}. \end{aligned} \tag{5.1}$$

In general, in order to find an optimal pair (x_1, x_2) we need exact knowledge of $F(x)$. However, as a direct consequence of (5.1), we can provide the following particular decision rule.

Lemma 5.2.1 (No-overlap principle). *It is optimal to allocate sensors without overlapping, if*

$$1 - p \leq \frac{f(x)}{f(x+1)} \leq \frac{1}{1-p}$$

for every $x \in [0,1]$, *where* $f(x) = \frac{dF(x)}{dx}$ *is the probability density function of the object location.*

Proof. By differentiating the expression to be maximized in (5.1), we show that it decreases in x_1, if $f(x_1)/f(x_1+1) \geq 1-p$, for every $x_1 \in [0,1]$. In this case, 0 is the optimal value for x_1. Similarly, this expression increases in x_2 if $f(x_2)/f(x_2+1) \leq 1/(1-p)$ for $x_2 \in [0,1]$, which sets 1 as the optimal value for x_2. □

The no-overlap principle indicates that it is optimal to maximize the coverage if the distribution of the intruder's position is sufficiently close to uniform. We illustrate the no-overlap principle by means of two examples, namely one example where the principle is applicable, and another where it is not.

Example 5.2.2. Assume that the intruder's entering position has uniform distribution, i.e., $f(x) = 1/2$, for every $x \in [0,2]$. In this case our decision rule says that it is optimal to avoid any overlapping.

Example 5.2.3. Assume that the intruder's position has a linear density function, e.g., $f(x) = x/2$, for every $x \in [0,2]$. The no-overlap rule cannot give us an unambiguous answer in this case. By solving (5.1), we obtain a more sophisticated joint sensor's allocation:

$$x_1 = \min\left\{\frac{1-p}{p}, 1\right\} \text{ and } x_2 = 1.$$

5.2.2 Sensor deployment in a 2-D area

Let $N \in \mathbb{N}$, and let $\mathcal{X} \subseteq \{1,\ldots,N\} \times \{1,\ldots,N\}$ be a two-dimensional discrete grid. Further, for all $x \in \mathcal{X}$, let $f(x)$ be the probability that an object is at position x, provided that there is an object in the area. As before, r is an effective sensing range of a sensor, and p is the detection probability of one sensor. Our objective is to provide an algorithm which finds the 'optimal' deployment of sensors in $\mathcal{X}$, so that the probability to miss the object is decreased as much as possible. Note that the problem now is discretized by allowing only placements on some pre-specified points.

We say that a sensor is *deployed at position* $y \in \mathcal{X}$ if y is the center of the sensor's sensing range. Further, a tuple $\vec{y} = (y_1,\ldots,y_n) \in \mathcal{X}^n$ ($n \in \mathbb{N} \cup \{0\}$) is called a *deployment of size* n, if n sensors are deployed at positions $y_1,\ldots,y_n$. We use $\emptyset$ for the empty deployment.

Now, for $x \in \mathcal{X}$, $n \in \mathbb{N}$ and $\vec{y} = (y_1, \ldots, y_n) \in \mathcal{X}^n$, define $g(\vec{y} \mid x)$ to be the probability that an intruder is *not* detected by any of the sensors deployed at positions $y_1, \ldots, y_n$ provided that the intruder's position is x. Further, denote by $p_{\text{missed}}(\vec{y})$ the probability that *none* of the sensors of the deployment $\vec{y}$ detects the intruder. Then, given that there is an intruder in the area, we obtain:

$$p_{\text{missed}}(\vec{y}) = \sum_{x \in \mathcal{X}} f(x) g(\vec{y} \mid x), \quad \text{for } \vec{y} = (y_1, \ldots, y_n) \in \mathcal{X}^n,\ n \in \mathbb{N}. \tag{5.2}$$

One can compute $p_{\text{missed}}((y_1, \ldots, y_m))$ for all $m \in \{1, \ldots, n\}$ iteratively as follows. First, note that, naturally, $g(\emptyset \mid x) = 1$ for all $x \in \mathcal{X}$, and thus

$$p_{\text{missed}}(\emptyset) = \sum_{x \in \mathcal{X}} f(x) g(\emptyset \mid x) = \sum_{x \in \mathcal{X}} f(x) = 1.$$

Next, let $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be the Euclidean distance function. Take $m \in \{1, \ldots, n\}$, $x \in \mathcal{X}$ and consider a deployment $(y_1, \ldots, y_m)$ of size m. Since the sensors are independent, we get

$$\begin{aligned} g((y_1, \ldots, y_m) \mid x) &= g((y_m) \mid x) g((y_1, \ldots, y_{m-1}) \mid x) \\ &= \begin{cases} g((y_1, \ldots, y_{m-1}) \mid x) & \text{if } d(x, y_m) > r \\ (1-p) g((y_1, \ldots, y_{m-1}) \mid x) & \text{if } d(x, y_m) \le r. \end{cases} \end{aligned} \tag{5.3}$$

Now, given the deployment $(y_1, \ldots, y_{m-1})$, the probability

$$p_{\text{missed}}((y_1, \ldots, y_{m-1}, y_m))$$

can be computed using (5.2) and (5.3).

Using the described iterative approach, we can now address two (closely related) optimization problems: Minimum Size Deployment (MSD) and Minimum Probability Deployment (MPD).

- MSD: Given $\beta \in [0, 1]$, find a deployment $\vec{y}$ of minimal size such that $p_{\text{missed}}(\vec{y}) \le \beta$.
- MPD: Given $n \in \mathbb{N}$, find a deployment $\vec{y}$ of size n such that $p_{\text{missed}}(\vec{y})$ is minimal.

We provide a heuristic algorithm described below, which can be used for both problems. The only difference is in the stopping criterion. In the main iterative step of the algorithm, a sensor is added to the deployment in such a way that the non-detection probability $p_{\text{missed}}(\cdot)$ is minimized (in case of a tie, the algorithm sticks to the candidate deployment that has been found first). This implies that the algorithm will find a 'locally optimal' solution, not necessarily the globally optimal one.

The heuristic algorithm

Input:

- MSD: $\beta \in [0,1]$;
- MPD: $n \in \mathbb{N}$.

Initialization: $m := 0$.

Iterative Step:

$$\begin{aligned} y_{m+1} &:= \arg\min_{y\in\mathcal{X}} p_{\text{missed}}((y_1,\ldots,y_m,y)) \\ &= \arg\min_{y\in\mathcal{X}} \sum_{x\in\mathcal{X}} f(x)g((y_1,\ldots,y_m,y)\mid x), \end{aligned}$$

where $g((y_1,\ldots,y_m,y)\mid x)$ is computed by (5.3) for all $x \in \mathcal{X}$;

$m := m+1$.

Termination:

- MSD: $p_{\text{missed}}((y_1,\ldots,y_m)) \le \beta$, then STOP;
- MPD: $m = n$, then STOP.

Output: $\vec{y} := (y_1,\ldots,y_m)$.

Note that there is a strong connection between the proposed algorithm and the no-overlap principle (see Lemma 5.2.1). Indeed, (3) says that the deployment of a new sensor at a position y reduces the non-detection probability $f(x)g(\vec{y}|x)$ by a factor $1-p$ for all x such that $d(x,y) \le r$. Since, ideally, we would like to reduce the highest values of $f(x)g(\vec{y}|x)$, the equivalent formulation of the iterative step is as follows:

$$y_{m+1} := \arg\max_{y\in\mathcal{X}} \sum_{x:d(x,y)\le r} f(x)g((y_1,\ldots,y_m)\mid x). \tag{5.4}$$

Now assume that we have deployed two sensors, and our algorithm allowed an overlap. Denote the sensing range of sensor $i = 1,2$ by S_i. Then, since (5.4) holds for the deployment of sensor 2, it follows that

$$\sum_{x\in S_1\cap S_2} (1-p)\,f(x) + \sum_{x\in S_2\setminus S_1} f(x) \ge \sum_{x\in S} f(x)$$

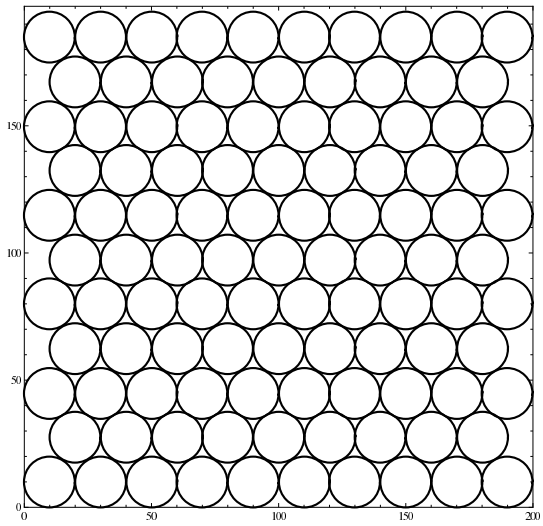


Figure 5.3: The hexagonal deployment of 105 sensors.

for any possible sensor range S in the area that does not overlap with S_1. (Otherwise, S could have been chosen instead of S_2.) Since the density in the first term of the left-hand side is taken with the factor $1-p$ we see that for the inequality to hold, the values of $f(x)$ in $S_1 \cap S_2$ and/or in $S_2 \setminus S_1$ should be considerably larger than in their neighborhoods. This can be seen as the condition of the no-overlap principle, applied in two dimensions: overlap is possible only if there exist positions x such that the density $f(\cdot)$ varies considerably (by a factor of $1-p$) within a sensor range of a sensor deployed in x.

In case two positions y would reduce the maximum non-detection probability by the same amount, we can break the tie arbitrarily, e.g. by using the first such position encountered, or by doing this randomly. The actual tie-breaking procedure does not matter too much on a global scale, because in the next iteration it is most likely that the other position will be chosen, except if the two positions are close (within a distance $2r$). Locally, there may occur significant effects of tie-breaking. We did not study this, but this topic warrants further investigation.

We have implemented the proposed algorithm in *Mathematica*. Below we present two examples of the deployment which is the output of our algorithm. Another example will be given in Section 5.5.

Example 5.2.4. Suppose $\mathcal{X} = \{1,\ldots,200\} \times \{1,\ldots,195\}$, $p = 0.9$ and $r = 10$. Moreover, suppose that f is the uniform distribution. We can construct a hexagonal deployment of 105 sensors in $\mathcal{X}$ such that an intruder cannot be within the range of two different sensors (see Figure 5.3). It is easy to see that this deployment is optimal for the given number of sensors, and a simple calculation shows that the non-detection probability of this deployment is 0.255. Deploying the 105 sensors according to our algorithm leads to the deployment shown in Figure 5.4. The non-detection probability of this deployment is 0.267 which is close to the non-detection

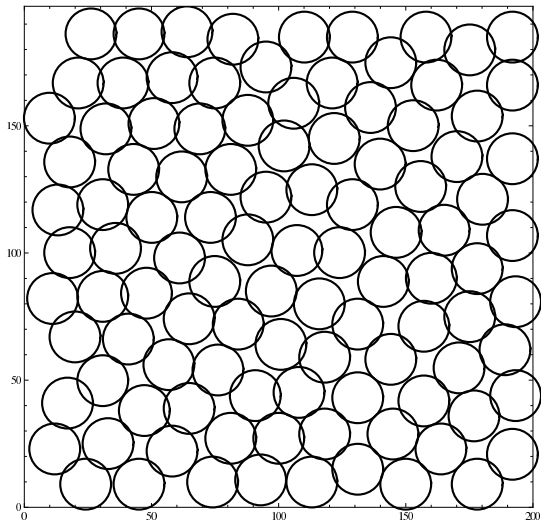


Figure 5.4: Deployment of 105 sensors according to the MPD algorithm: uniform distribution of the object location.

probability of the optimal hexagonal deployment.

Example 5.2.5. Suppose $\mathcal{X} = \{0, \ldots, 100\} \times \{0, \ldots, 100\}$, $p = 0.8$ and $r = 10$. Moreover, define $\sigma = 25$ and define r_x as the distance of x to the north-east axis (the line $y = x$) of $\mathcal{X}$ for each $x \in \mathcal{X}$. Now suppose that $f(x) = c\mathrm{e}^{-\frac{1}{2}(\frac{r_x}{\sigma})^2}$ for all $x \in \mathcal{X}$, where c is the normalization constant making f a probability distribution on $\mathcal{X}$. In other words, the signed distance between the intruder's position and the north-east axis of $\mathcal{X}$ follows a discrete version of the normal distribution with mean 0 and standard deviation $\sigma = 25$. Here, the sign is positive for positions above the line, and negative for those below.

Having 200 sensors at our disposal, applying our algorithm leads to the deployment in Figure 5.5. As one would expect, the density of the sensor deployment increases when approaching the north-east axis. Moreover, a simple calculation shows that the non-detection probability of this deployment is 0.066.

We conclude that our heuristic algorithm can be used to find deployments which result in a good detection probability and are in line with the analytical results from Section 5.2.1. In particular, in the case of a uniform a priori probability distribution of the intruder position we found a nearly optimal solution.

5.3 Statistical methods for intruder detection

Optimal sensor deployment studied in the previous section is important for increasing the overall detection probability, that is, the number of true alarms produced by the system. However, since the false alarm probability q can be high in practice (e.g. q can be about 2%, which already has a considerable impact), sensor networks

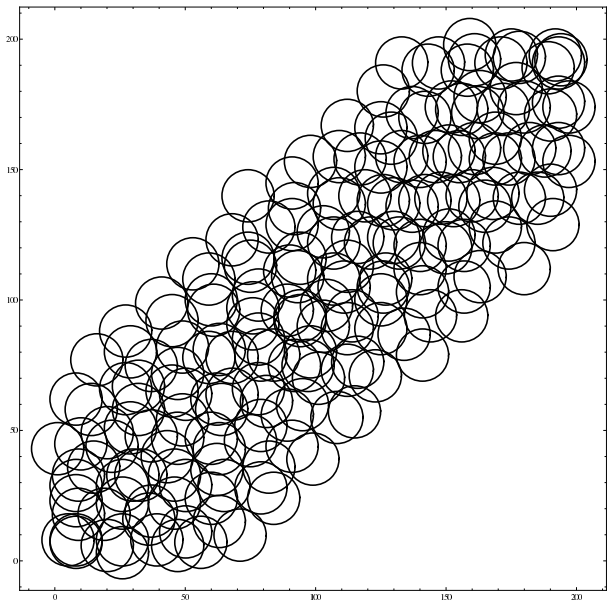


Figure 5.5: Deployment of 200 sensors according to the MPD algorithm: normal distribution of the signed distance between the object location and the north-east axis.

may even produce multiple false alarms at each moment in time. Still, the presence of an intruder increases the number of alarms, and after several observations one should be able to recognize an intrusion and report an alarm. To this end, we present in this section two statistical methods for intruder detection: one is based on classical hypothesis testing, and the other employs a Bayesian approach.

The hypothesis testing approach in Section 5.3.1 provides a decision making tool for reporting an intrusion alarm after a single observation of n identical sensors. In practice, false alarm reports are highly undesirable. Therefore, we bound the probability of a false report by choosing a high confidence level of the test. This sometimes leads to a poor performance of the test in a sense that with high probability, after one observation of n sensors, an object will stay undetected. In practice, however, this is not a big problem because there is usually enough time to produce several observations, not necessarily by the same sensor. Then the probability of the intruder's presence can be updated after each observation, for instance, using the Bayesian approach described in Section 5.3.2.

The Bayesian approach allows for great flexibility, because, along with the total number of alarms, it also takes into account the locations of the alarms. Therefore, in Section 5.3.2 we analyze a more general model than in Section 5.3.1. Specifically, we consider several non-overlapping parts of the coverage area, each deploying a number of completely overlapping sensors. Furthermore, we let the intrusion probabilities, as well as the detection and false alarm probabilities, depend on the sensor location. The motivation for this model is that although identical sensors will usually cover parts of the intrusion area with roughly equal sizes, the terrain

in which the sensors are placed may vary, e.g. in altitude, which can influence the local intrusion probabilities and the performance of the sensors.

5.3.1 Hypothesis testing for intruder detection

In the following, we consider the two extreme cases:

- Case I: n sensors, all at the same position; i.e. with identical sensing range;
- Case II: n non-overlapping sensors.

If there is an object in the area, then Case I is the case in which all sensors follow the same Bernoulli distribution with parameter p and Case II is the case that one of the sensors detects the object with probability p and each of the remaining sensors detect the object with probability q.

Assume that there can be at most one object in the area. Within the hypothesis testing formulation, we test the null-hypothesis that there is no intruder in the area against the alternative that the area is penetrated. If a critical number of alarms is observed then we reject the null-hypothesis and report an intrusion alarm. For $i = 1, \ldots, n$ let $[Y_i = 1]$ be the event that sensor i detects an object and $[Y_i = 0]$ be the complementary event. Assuming that there is an intruder in the range of sensor i, we have $P(Y_i = 1) = p$.

Consider Case I: n sensors deployed at the same position with 100% overlap. Thus, our hypothesis testing formulation is as follows:

$$\text{Case I:} \quad \begin{cases} H_0: & P(Y_i = 1) = q \text{ for all } i = 1, \ldots, n, \\ H_1: & P(Y_i = 1) = p \text{ for all } i = 1, \ldots, n. \end{cases}$$

In Case II, the sensors are not overlapping. Thus, the object can penetrate the range of at most one sensor. This leads to the following formalization:

$$\text{Case II:} \quad \begin{cases} H_0: & P(Y_i = 1) = q \text{ for all } i = 1, \ldots, n, \\ H_1: & P(Y_j = 1) = p \text{ for exactly one } j = 1, \ldots, n; \\ & P(Y_i = 1) = q \text{ for } i = 1, \ldots, n,\ i \neq j. \end{cases}$$

In both cases, as a statistic, we use the stochastic variable $T = Y_1 + \cdots + Y_n$, the number of alarms produced by the system. We reject H_0 if and only if $T \geq c$, for some critical $c > 0$. Clearly, under H_0, T has a Binomial(n, q) distribution. Denote the Binomial density function with parameters n and p at k by $B_{n,p}(k)$:

$$B_{n,p}(k) = \binom{n}{k} p^k (1-p)^{n-k}. \tag{5.5}$$

In our test, two types of errors can be made: false positives and false negatives (in statistical terms, type-one and type-two error, respectively). A *false positive* means

q \ n	3	4	5	6	7	8	9	10
0.02	1	1	1	1	1	1	1	1
0.04	1	1	1	1	1	1	1	2
0.06	1	1	1	1	2	2	2	2
0.08	1	1	2	2	2	2	2	2
0.10	1	2	2	2	2	2	3	3
0.12	1	2	2	2	2	3	3	3
0.14	2	2	2	2	3	3	3	3
0.16	2	2	2	3	3	3	3	4
0.18	2	2	2	3	3	3	4	4
0.20	2	2	3	3	3	4	4	4

Table 5.1: Critical number of alarms c for Cases I and II.

a false report, i.e., an intruder alarm is reported while there is no object in the area. In both Cases I and II, one has

$$p_{\text{false}} = P(\text{false positive}) = P_{H_0}(T \geq c) = \sum_{k=c}^{n} B_{n,q}(k).$$

We choose c in such a way that the above probability does not exceed an acceptable frequency of false alarm reports. A *false negative* means that an intruder is missed by the system, i.e., the intrusion alarm will not be reported while there was an object in the area. For Case I, we get

$$p^{I}_{\text{missed}} = P(\text{false negative}) = P_{H_1}(T < c) = \sum_{k=0}^{c-1} B_{n,p}(k),$$

and for Case II, we obtain

$$p^{II}_{\text{missed}} = P(\text{false negative}) = p \sum_{k=0}^{c-2} B_{n-1,q}(k) + (1-p) \sum_{k=0}^{c-1} B_{n-1,q}(k).$$

In this setting, the detection probability $p_{\text{detection}}$ of the system is equal to the power of the statistical test, i.e.,

$$p_{\text{detection}} = 1 - P(\text{false negative}).$$

We select some values for p and q and calculate corresponding values of c and $p_{\text{detection}}$ so that $p_{\text{false}} \leq 0.05$. In Tables 5.1 and 5.2 we present the values of c for Cases I and II. Table 5.3 gives the values of $p_{\text{detection}}$ for Case I, whereas Tables 5.4 and 5.5 give the values for Case II. In all the tables, the single-sensor detection probability is fixed at $p = 0.9$. The values of c used in Tables 5.3–5.5 are chosen according to the results of Tables 5.1 and 5.2.

As we see in Case I, $p_{\text{detection}}$ is very high. This is not surprising because in fact, in this case we have to distinguish between Binomial(n, p) and Binomial(n, q)

q \ n	15	30	45	60	75	90	105	120	135	150
0.02	1	2	3	3	4	4	5	5	6	6
0.04	2	3	4	5	6	7	8	9	9	10
0.06	3	4	6	7	8	9	11	12	13	14
0.08	3	5	7	8	10	12	13	15	16	18
0.10	4	6	8	10	12	14	16	18	19	21
0.12	4	7	9	12	14	16	18	20	23	25
0.14	4	7	10	13	16	18	21	23	26	28
0.16	5	8	11	14	17	20	23	26	29	32
0.18	5	9	12	16	19	22	26	29	32	35
0.20	6	10	14	17	21	24	28	31	35	38

Table 5.2: Critical number of alarms c for Cases I and II.

q \ n	3	4	5	6	7	8	9	10
0.02	0.9990	0.9999	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
0.04	0.9990	0.9999	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
0.06	0.9990	0.9999	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
0.08	0.9990	0.9999	0.9995	0.9999	1.0000	1.0000	1.0000	1.0000
0.10	0.9990	0.9963	0.9995	0.9999	1.0000	1.0000	1.0000	1.0000
0.12	0.9990	0.9963	0.9995	0.9999	1.0000	1.0000	1.0000	1.0000
0.14	0.9720	0.9963	0.9995	0.9999	0.9998	1.0000	1.0000	1.0000
0.16	0.9720	0.9963	0.9995	0.9987	0.9998	1.0000	1.0000	1.0000
0.18	0.9720	0.9963	0.9995	0.9987	0.9998	1.0000	0.9999	1.0000
0.20	0.9720	0.9963	0.9914	0.9987	0.9998	0.9996	0.9999	1.0000

Table 5.3: Values of $p_{\text{detection}}$ for Case I; $p = 0.9$.

q \ n	3	4	5	6	7	8	9	10
0.02	0.9000	0.9001	0.9002	0.9004	0.9006	0.9008	0.9010	0.9013
0.04	0.9002	0.9005	0.9009	0.9015	0.9022	0.9029	0.9038	0.2772
0.06	0.9004	0.9010	0.9020	0.9032	0.2795	0.3170	0.3524	0.3857
0.08	0.9006	0.9018	0.2554	0.3073	0.3551	0.3993	0.4402	0.4780
0.10	0.9010	0.2440	0.3099	0.3694	0.4233	0.4721	0.1687	0.2035
0.12	0.9014	0.2868	0.3609	0.4265	0.4846	0.1816	0.2242	0.2672
0.14	0.2344	0.3278	0.4087	0.4788	0.1807	0.2310	0.2817	0.3318
0.16	0.2650	0.3670	0.4534	0.1654	0.2232	0.2818	0.3396	0.1473
0.18	0.2948	0.4044	0.4951	0.2006	0.2672	0.3332	0.1454	0.1907
0.20	0.3240	0.4400	0.1629	0.2371	0.3119	0.1337	0.1838	0.2376

Table 5.4: Values of $p_{\text{detection}}$ for Case II; $p = 0.9$.

q \ n	15	30	45	60	75	90	105	120	135	150
0.02	0.9031	0.4010	0.1989	0.3008	0.1680	0.2404	0.1421	0.1975	0.1208	0.1648
0.04	0.3935	0.2944	0.2346	0.1922	0.1599	0.1344	0.1138	0.0968	0.1569	0.1337
0.06	0.1841	0.2288	0.1114	0.1298	0.1429	0.1526	0.0876	0.0945	0.1004	0.1053
0.08	0.2811	0.1813	0.1252	0.1759	0.1255	0.0909	0.1183	0.0875	0.1094	0.0822
0.10	0.1434	0.1448	0.1339	0.1213	0.1091	0.0980	0.0879	0.0789	0.1135	0.1009
0.12	0.2103	0.1157	0.1393	0.0836	0.0943	0.1020	0.1077	0.1117	0.0741	0.0772
0.14	0.2836	0.1952	0.1424	0.1065	0.0810	0.1040	0.0798	0.0971	0.0754	0.0891
0.16	0.1583	0.1567	0.1438	0.1297	0.1164	0.1044	0.0935	0.0839	0.0753	0.0677
0.18	0.2143	0.1253	0.1440	0.0908	0.0987	0.1036	0.0691	0.0722	0.0743	0.0756
0.20	0.1180	0.0995	0.0792	0.1084	0.0833	0.1020	0.0790	0.0925	0.0725	0.0828

Table 5.5: Values of $p_{\text{detection}}$ for Case II; $p = 0.9$.

distributions. This can be done with a good precision because of a large difference between p and q. For instance, for 10 sensors, $p = 0.9$, $q = 0.02$, and $c = 5$, the probability $p_{\text{detection}}$ is 0.9999 while p_{false} is as small as 7.4×10^{-7}.

In Case II, $p_{\text{detection}}$ is low except for the cases when $c = 1$, that is, a detection signal of one sensor already triggers an intrusion alarm. The value $c > 1$ is obtained when the probability of just one alarm is reasonably high even if there is no intruder in the area. Effectively, $c > 1$ means that at least $c - 1$ false alarms are needed to detect the intruder. This is an undesirable result, which explains, in particular, the low power of the test. We conclude that in Case II one observation is simply not enough for efficient intruder detection, because in this case the observations with and without the intruder differ by at most one signal, which is difficult to reveal by classical hypothesis testing. One either has to make sensors overlap (as in Case I) or use several observations in a row. The latter can be done in several ways, for instance, one can use the Viterbi algorithm as in Section 5.4.

5.3.2 Bayesian approach for intruder detection

Consider Case II from the previous section, where $n \in \mathbb{N}$ different sensors are placed in such a way that the sensing ranges of different sensors do not overlap. Let $X \in \{0, 1\}$ denote the number of intruders present, with $P(X = 1) = \alpha$ an a priori probability of the intruder being present in the area. As before, let T be the stochastic variable denoting the total number of single-sensor alarms given at a particular time instant, so $T \in \{0, 1, \ldots, n\}$. We have

$$P(X = 0 \mid T = k) = \frac{P(T = k \mid X = 0)P(X = 0)}{P(T = k \mid X = 0)P(X = 0) + P(T = k \mid X = 1)P(X = 1)}.$$

Let F be the (unobservable) number of *false* alarms among the T. Then for all $k \geq 0$ we obtain

$$\begin{aligned} P(T = k \mid X = 1) &= P(T = k, F = k - 1 \mid X = 1) \\ &\quad + P(T = k, F = k \mid X = 1) \\ &= pB_{n-1,q}(k - 1) + (1 - p)B_{n-1,q}(k) \\ &= B_{n,q}(k)\left[\tfrac{kp}{nq} + \tfrac{(n-k)(1-p)}{n(1-q)}\right], \\ P(T = k \mid X = 0) &= B_{n,q}(k). \end{aligned}$$

Hence, the a posteriori probability of the presence of an object is

$$\begin{aligned}
&P(X=1 \mid T=k) \\
&\quad = 1 - \frac{P(T=k \mid X=0)P(X=0)}{P(T=k \mid X=0)P(X=0) + P(T=k \mid X=1)P(X=1)} \\
&\quad = 1 - \frac{1}{1 + \frac{P(T=k|X=1)P(X=1)}{P(T=k|X=0)P(X=0)}} \\
&\quad = 1 - \left(1 + \tfrac{\alpha}{1-\alpha}\left[\tfrac{kp}{nq} + \tfrac{(n-k)(1-p)}{n(1-q)}\right]\right)^{-1}. \qquad (5.6)
\end{aligned}$$

This formula can be generalized to the case combining Cases I and II from Section 5.3.1 as follows. Assume n non-overlapping ranges. Range $i = 1, \ldots, n$ contains $m_i \in \mathbb{N}$ completely overlapping sensors. Let $T_i \in \{0, \ldots, m_i\}$ be the number of alarms for range i and denote $\vec{T} = (T_1, \ldots, T_n)$.

The stochastic variables $X_i \in \{0, 1\}$, $i = 1, \ldots, n$, indicating the presence of an object in range i, have a priori probabilities $P(X_i = 1) = \alpha_i$ i.e., we allow certain parts of the area to have a higher a priori probability for intrusion than others. Also, we allow the detection and false alarm probabilities to depend on the sensor range; we use p_i and q_i to denote these respectively.

Since we assume that there can be at most one intruder at any given time instant, the vector $\vec{X} = (X_1, \ldots, X_n)$ can attain values in the set $\{e_j : j = 0, \ldots, n\}$ where e_j is the jth unit vector in $\mathbb{R}^n$ and e_0 the zero vector in that space. We will use the notation $\mathcal{N} = \{0, 1, \ldots, n\}$. We then calculate

$$\begin{aligned}
&P(\vec{X} = e_j \mid \vec{T} = \vec{k}) \\
&= \frac{P(\vec{T} = \vec{k} \mid \vec{X} = e_j)P(\vec{X} = e_j)}{P(\vec{T} = \vec{k} \mid \vec{X} = e_j)P(\vec{X} = e_j) + P(\vec{T} = \vec{k} \mid \vec{X} \neq e_j)P(\vec{X} \neq e_j)} \\
&= \frac{P(\vec{T} = \vec{k} \mid \vec{X} = e_j)P(\vec{X} = e_j)}{P(\vec{T} = \vec{k} \mid \vec{X} = e_j)P(\vec{X} = e_j) + \sum_{s \in \mathcal{N}\setminus\{j\}} P(\vec{T} = \vec{k} \mid \vec{X} = e_s)P(\vec{X} = e_s)}.
\end{aligned}$$

Further, we immediately have for $j > 0$ that

$$P(\vec{T} = \vec{k} \mid \vec{X} = e_j) = B_{m_j, p_j}(k_j) \prod_{i \in \mathcal{N}\setminus\{j\}} B_{m_i, q_i}(k_i). \qquad (5.7)$$

If we define $m_0 = k_0 = 0$, this formula also holds for $j = 0$. Furthermore, if we

define $\alpha_0 = 1 - \sum_{i\in\mathcal{N}\setminus\{0\}} \alpha_i$, we can state

$$
\begin{aligned}
&P(\vec{X} = e_j \mid \vec{T} = \vec{k}) \\
&\quad = \frac{P(\vec{T}=\vec{k}|\vec{X}=e_j)P(\vec{X}=e_j)}{P(\vec{T}=\vec{k}|\vec{X}=e_j)P(\vec{X}=e_j)+\sum_{s\in\mathcal{N}\setminus\{j\}} P(\vec{T}=\vec{k}|\vec{X}=e_s)P(\vec{X}=e_s)} \\
&\quad = \left(1 + \frac{\sum_{s\in\mathcal{N}\setminus\{j\}} P(\vec{T}=\vec{k}|\vec{X}=e_s)P(\vec{X}=e_s)}{P(\vec{T}=\vec{k}|\vec{X}=e_j)P(\vec{X}=e_j)}\right)^{-1} \\
&\quad = \left(1 + \frac{\sum_{s\in\mathcal{N}\setminus\{j\}} \alpha_s B_{m_s,p_s}(k_s)\prod_{i\in\mathcal{N}\setminus\{s\}} B_{m_i,q_i}(k_i)}{\alpha_j B_{m_j,p_j}(k_j)\prod_{v\in\mathcal{N}\setminus\{j\}} B_{m_v,q_v}(k_v)}\right)^{-1} \\
&\quad = \left(1 + \sum_{s\in\mathcal{N}\setminus\{j\}} \frac{\alpha_s}{\alpha_j}\left(\frac{p_s}{q_s}\right)^{k_s}\left(\frac{1-p_s}{1-q_s}\right)^{m_s-k_s}\left(\frac{p_j}{q_j}\right)^{-k_j}\left(\frac{1-p_j}{1-q_j}\right)^{-(m_j-k_j)}\right)^{-1} \\
&\quad = \left(\sum_{s\in\mathcal{N}} \frac{\alpha_s}{\alpha_j}\left(\frac{p_s}{q_s}\right)^{k_s}\left(\frac{1-p_s}{1-q_s}\right)^{m_s-k_s}\left(\frac{p_j}{q_j}\right)^{-k_j}\left(\frac{1-p_j}{1-q_j}\right)^{-(m_j-k_j)}\right)^{-1} \\
&\quad = \frac{\alpha_j\left(\frac{p_j}{q_j}\right)^{k_j}\left(\frac{1-p_j}{1-q_j}\right)^{m_j-k_j}}{\sum_{s\in\mathcal{N}} \alpha_s\left(\frac{p_s}{q_s}\right)^{k_s}\left(\frac{1-p_s}{1-q_s}\right)^{m_s-k_s}}. \qquad (5.8)
\end{aligned}
$$

For $j = 0$, we thus find

$$
P(\vec{X} = e_0 \mid \vec{T} = \vec{k}) = \frac{1 - \sum_{i\in\mathcal{N}\setminus\{0\}} \alpha_i}{1 - \sum_{i\in\mathcal{N}\setminus\{0\}} \alpha_i + \sum_{s\in\mathcal{N}\setminus\{0\}} \alpha_s\left(\frac{p_s}{q_s}\right)^{k_s}\left(\frac{1-p_s}{1-q_s}\right)^{m_s-k_s}},
$$

so the conditional probability of an intruder given the observed area alarms vector $\vec{T}$ equals

$$
\begin{aligned}
P(\vec{X} \neq e_0 \mid \vec{T} = \vec{k}) &= 1 - P(\vec{X} = e_0 \mid \vec{T} = \vec{k}) \\
&= 1 - \left(1 + \frac{\sum_{s\in\mathcal{N}\setminus\{0\}} \alpha_s\left(\frac{p_s}{q_s}\right)^{k_s}\left(\frac{1-p_s}{1-q_s}\right)^{m_s-k_s}}{1 - \sum_{i\in\mathcal{N}\setminus\{0\}} \alpha_i}\right)^{-1}.
\end{aligned}
$$

Notice that the Case II treated in Section 5.3.1 corresponds to

$$
p_i = p, \quad q_i = q, \quad m_i = 1, \quad \alpha_i = \alpha/n,
$$

for all $i \in \mathcal{N}$ and we then find back our earlier formula (5.6) for the conditional probability of an intrusion.

In the case where we use only one time instant to observe the alarms, it seems natural to conclude that an intruder is present whenever

$$
A(\vec{k}) = P(\vec{X} \neq e_0 \mid \vec{T} = \vec{k}) = 1 - \left(1 + \frac{\sum_{s\in\mathcal{N}\setminus\{0\}} \alpha_s\left(\frac{p_s}{q_s}\right)^{k_s}\left(\frac{1-p_s}{1-q_s}\right)^{m_s-k_s}}{1 - \sum_{i\in\mathcal{N}\setminus\{0\}} \alpha_i}\right)^{-1} \qquad (5.9)
$$

satisfies $A(\vec{k}) \geq \gamma$ for some critical threshold γ, where e.g. we can choose $\gamma \in (0.5, 1)$, which means that we raise an alarm whenever the conditional probability of an intruder is sufficiently larger than the conditional probability that there is no intruder. The probability of a false intrusion alarm then becomes

$$\begin{aligned} p_{\text{false}} &= P(\,(A(\vec{T}) \geq \gamma\,) \wedge (\vec{X} = e_0)\,) \\ &= \sum_{\vec{0} \leq \vec{k} \leq \vec{m}} \mathbb{1}_{\{A(\vec{k}) \geq \gamma\}} P(\vec{T} = \vec{k} \mid \vec{X} = e_0) P(\vec{X} = e_0), \end{aligned}$$

where we use the shorthand notation $\vec{0} \leq \vec{k} \leq \vec{m}$ for $\{\vec{k} : \ 0 \leq k_i \leq m_i,\ i \in \mathcal{N}\}$. The probability of a missed intrusion is

$$\begin{aligned} p_{\text{missed}} &= P(\,(A(\vec{T}) < \gamma\,) \wedge (\vec{X} \neq e_0)\,) \\ &= \sum_{\vec{0} \leq \vec{k} \leq \vec{m}} \mathbb{1}_{\{A(\vec{k}) < \gamma\}} \sum_{j \in \mathcal{N} \setminus \{0\}} P(\vec{T} = \vec{k} \mid \vec{X} = e_j) P(\vec{X} = e_j). \end{aligned}$$

By substituting (5.7) and (5.9) in these expressions, we can now calculate explicitly what the probabilities of a false intrusion alarm or missed intrusion are (based on a single observation in time) for the given a priori probabilities in $\vec{p}$ and $\vec{q}$ and a given sensor configuration vector $\vec{m}$.

We note that the Bayesian approach can be also extended to a sequence of observations. For instance, the a posteriori probabilities obtained by using (5.8) after the first observation, can be substituted back into (5.8) instead of α_j's to recompute the probabilities of the intruder's presence after the second observation, and so on.

5.4 Viterbi algorithm for intruder detection

In this section, we present a novel method of using sequential observations for intruder detection. We model the signals from the sensors as a so-called hidden Markov model. This is a stochastic process, based on a Markov chain to which noise is added. Using this representation we can distinguish between the signals that should have been given off by the sensors, i.e. the 'true' state of the system, and the signals that are actually given off, including the false alarms and missed detections.

Given a sequence of signals we determine the most likely sequence of true states, using the so-called Viterbi algorithm. In this way, we decide whether the signals indicate indeed an intruder, or are only false alarms. From this we derive a decision rule for when to report an intrusion alarm, thus reducing the number of false reports. All calculations needed to obtain this rule can be pre-computed.

We outline the proposed method for the case of one sensor. In particular, we explain the hidden Markov model, and illustrate how, based on a few signals from the sensor, we decide if an intrusion alarm should be given. We indicate how the method can be extended to networks of sensors. As the state space, and so the

number of calculations, increases exponentially with the number of sensors, we show how to truncate it in a clever way.

5.4.1 A one-sensor model

Consider the case of one sensor, where an object possibly passes by. Assuming a low speed of the object, the object is in the range of the sensor for multiple time steps. Let the stochastic process $\{X_t\}_{t\in\mathbb{N}}$ denote if an object is in the range of the sensor, where

$$X_t = \begin{cases} 1 & \text{if an object is in the range of the sensor at time } t, \\ 0 & \text{otherwise.} \end{cases}$$

So X_t gives the 'true' state of the system at time t.

We assume that the process $\{X_t\}$ is a Markov chain, so the probability law for X_{t+1} only depends on X_t. Denote $p_{ij} = P(X_{t+1} = j \mid X_t = i)$. The speed of the object and its path through the range of the sensor are modelled in the transition probabilities. The number of consecutive ones in a Markov chain follows a geometric distribution, with $E(\#\text{ of steps in sensor range}) = 1/p_{10}$. We want the stationary distribution of $\{X_t\}$, say X_∞, to be such that $P(X_\infty = 1) = 1 - P(X_\infty = 0) = \alpha$, the a priori probability that there is an object in the system. This gives the following transition probability matrix A:

$$A = \begin{pmatrix} 1 - \frac{\alpha}{1-\alpha} p_{10} & \frac{\alpha}{1-\alpha} p_{10} \\ p_{10} & 1 - p_{10} \end{pmatrix}.$$

We take the initial distribution for X_0 to be equal to the stationary distribution.

To the process $\{X_t\}$ we add noise, which consists of false alarms and missed detections. This gives the process of signals given off by the sensor, say $\{Y_t\}_{t\in\mathbb{N}}$. Let

$$Y_t = \begin{cases} 1 & \text{if the sensor gives an alarm at time } t, \\ 0 & \text{otherwise.} \end{cases}$$

So Y_t is the observed state at time t. The noise is such that Y_t only depends on X_t, in an independent and identically distributed (i.i.d.) way. A false alarm occurs when $[Y_t = 1]$ given $[X_t = 0]$, and this happens with probability q. A missed detection occurs if $[Y_t = 0]$ given $[X_t = 1]$, and this happens with probability $1 - p$.

We now have that the process $\{Y_t\}$ is a *hidden Markov model* [11]. We can interpret $\{Y_t\}$ as observing $\{X_t\}$ via a noisy channel. Only the process $\{Y_t\}$ is observed, while the states of the process $\{X_t\}$ are not known, i.e. hidden, which explains the name of this model. The process $\{X_t\}$ is often referred to as the underlying or hidden process. Whereas for a Markov chain it holds that the next state of the process depends only on the previous state, or a fixed number of previous states, for a hidden Markov model the transition probabilities depend on the entire history of the process.

5.4.2 The Viterbi algorithm for the one-sensor model

Given a sequence of observed states, say $O = \{O_1, O_2, \ldots, O_L\}$, the question now rises, what is the most likely sequence of underlying ('true') states, $Q = \{Q_1, Q_2, \ldots, Q_L\}$. There is an efficient algorithm for solving this problem, called the *Viterbi algorithm* [6]. This algorithm, based on dynamic programming, calculates

$$\max_Q P(Q \mid O).$$

Applying this algorithm we are able to correct false alarms and missed detections for a given sequence of observations. For example, a single one in between many zeros is likely to be a false alarm, while a zero in between many ones is probably a missed detection. If we, for instance, observe the sequence 000111011000 then it is not surprising that the most likely underlying state sequence is 000111111000, i.e., a missed detection is corrected. More important are the corrections of false alarms. The observed sequence 0001000 will most likely have an underlying sequence of all zeros, so a false alarm is corrected. In this way, we prevent reporting a false intrusion alarm. While for these two examples the most likely underlying states are straightforward to see, the algorithm also helps with cases like 00010100. Here, it is not immediately clear whether the ones are two false alarms, or the zero in between represents a missed detection.

Based on the results of this algorithm, we give a decision rule whether or not to report an intrusion alarm for a given sequence of observations. We illustrate this for two and for three consecutive observed states, but it can be done for every desired number of observations. We give an intrusion alarm if the most likely underlying state sequence contains at least one 1 in it, signifying that in the most likely scenario, an intrusion took place in at least one moment in time. We also calculate the probability that the underlying state sequence consists of only zeros, given the observation. One minus this quantity equals the probability that there was an intruder. The latter is equal to the probability p_{missed} that the intruder will pass undetected in case the sequence of all zeros happens to be most likely. All calculations can be done off-line, resulting in a list of observed states for which an intrusion alarm should be given.

For the values $p = 0.9$, $q = 0.02$, $\alpha = 0.01$ and $E(\#\text{ of steps in sensor range}) = 10$, the probabilities for all possible combinations of states are given in Table 5.6 for two and three consecutive observations. For two observations, we only give an intrusion alarm in case both observations are a 1. With probability 0.9441 this is indeed the underlying sequence, and the probability that there was no intruder is about 0.05. Giving no intrusion alarm when the observed sequence contained two or one zeros turns out to be correct with probabilities 0.9997 and 0.92, respectively. For three observations, there are four cases for which we give an intrusion alarm. To improve the probability of correct decisions further, one could make use of more consecutive observations.

O	Q	alarm?	$P(Q \mid O)$	$P(Q = \vec{0} \mid O)$
0 0	0 0	No	0.9997	0.9997
0 1	0 0	No	0.9196	0.9196
1 0	0 0	No	0.9196	0.9196
1 1	1 1	Yes	0.9441	0.0512
0 0 0	0 0 0	No	0.9998	0.9998
0 0 1	0 0 0	No	0.9491	0.9491
0 1 0	0 0 0	No	0.9833	0.9833
0 1 1	0 1 1	Yes	0.4016	0.2177
1 0 0	0 0 0	No	0.9491	0.9491
1 0 1	1 1 1	Yes	0.6060	0.3577
1 1 0	1 1 0	Yes	0.4016	0.2177
1 1 1	1 1 1	Yes	0.9936	0.0013

Table 5.6: Hidden Markov Model for the case of one sensor. For each observed state O the most likely underlying state Q is given.

For this model we have assumed that $\{X_t\}$ is a Markov chain. The number of steps in the range of the sensor is geometrically distributed, which models a variable speed and direction of the object. We can improve this by letting $\{X_t\}$ be a Markov chain of order k, where the probability law of X_{t+1} depends on the last k states: $X_{t-k+1}, \ldots, X_t$. This allows us to vary the distribution of the number of steps in the sensor range. For instance, in this way one can model a deterministic number of steps. The state space then increases to 2^k states, but the problem remains numerically tractable since the calculations for the decision rule need to be done only once.

5.4.3 A sensor-network model

We can extend this method to networks of several sensors. Consider for instance the following example with $n = 4$ non-overlapping sensors as given in Figure 5.6. Let $\vec{X}_t^T = \left(X_{1,t}, X_{2,t}, X_{3,t}, X_{4,t}\right)$, where $X_{i,t} = 1$ if there is an intrusion in the range of sensor i at time t, and $X_{i,t} = 0$ otherwise, $i = 1, 2, 3, 4$; $t \geq 1$. Assume that there is at most one object in the area at any moment in time, so that the state space of $\{\vec{X}_t\}$ consists of $n + 1 = 5$ states: the all-zero state and the states where the object is in the range of one of the n sensors. We assume the process $\{\vec{X}_t\}$ again to be Markov. The path and the speed of the object are modelled in the transition probabilities. This can be based on historical data, or on other knowledge about the system. If the object can remain in the range of one sensor for several time steps, p_{ii} is positive. Here, we assume that the object always enters via sensor 1, and then continues its path through sensor 2, 3 or 4, or outside the range of any of these

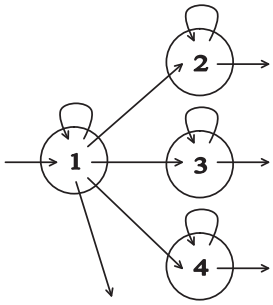


Figure 5.6: A network with four sensors. Indicated are possible transitions.

sensors. The transition probabilities and the corresponding states are given by

$$A = \begin{pmatrix} 1-\alpha & \alpha & 0 & 0 & 0 \\ 1-p_{1\bullet} & p_{11} & p_{12} & p_{13} & p_{14} \\ 1-p_{22} & 0 & p_{22} & 0 & 0 \\ 1-p_{33} & 0 & 0 & p_{33} & 0 \\ 1-p_{44} & 0 & 0 & 0 & p_{44} \end{pmatrix}, \quad \begin{matrix} (0,0,0,0) \\ (1,0,0,0) \\ (0,1,0,0) \\ (0,0,1,0) \\ (0,0,0,1) \end{matrix}$$

with the necessary conditions on the p_{ij} imposed to let A be a stochastic matrix. Here, $p_{1\bullet} = \sum_{j=1}^{4} p_{1j}$.

The probability law of observing $\vec{Y}_t$ given $\vec{X}_t$ follows a multinomial distribution. As before, there are four possibilities for the pair $(X_{i,t}, Y_{i,t})$, specifically, $P(Y_{i,t} = 1|X_{i,t} = 0) = q$ and $P(Y_{i,t} = 0|X_{i,t} = 1) = 1 - p$.

The state space of $\{\vec{Y}_t\}$ now consists of 2^n states: each sensor can give an alarm or not. As the size of the state space grows exponentially with n, already for a moderately large number of sensors n the problem becomes huge. Because of this, but moreover because many of these states are very unlikely to occur, we truncate the state space of $\{\vec{Y}_t\}$. For this, we calculate the number of false alarms, say c, that has a probability of occurring less than say 0.001:

$$P(\#\text{ false alarms} > c) < 0.001.$$

Now we allow only the vectors $\vec{Y}_t$ in the state space of $\{\vec{Y}_t\}$ that are at Hamming distance $\leq c$ away from any of the states of $\{\vec{X}_t\}$, where the *Hamming distance* between two zero-one vectors is the number of indices in which they are different. In this way, we drastically reduce the state space of $\{\vec{Y}_t\}$, making the calculations more tractable.

We now again have a hidden Markov model, for which we can derive a decision rule when to give an intrusion alarm in the same way as for the case of one sensor. We can list all possible sequences of a number of observations of the process $\{\vec{Y}_t\}$. By the Viterbi algorithm, we calculate the most likely underlying state sequences of the process $\{\vec{X}_t\}$. If it contains at least one 1, for such a sequence an intrusion alarm should be given. By calculating the probability that the underlying states are only zeros, the probability of making an error is found.

The hidden Markov model method of the present section can be used in combination with the heuristic algorithm for placement of sensors presented in Section 2. One way of doing this to use the Viterbi method to combine the results of multiple single-sensor readings into one result, giving improved values for p and q that can be used in the placement algorithm. This is done in some of the numerical experiments of the next section.

5.5 Numerical results

We verify and combine the proposed methods for sensor deployment and intruder detection using a simulation model of a network consisting of a number of individual sensors, which perform under uncertainty. The performance of each individual sensor is characterized by the probability of true detection p and the probability of false alarm q. As before, we use similar performance measures for characterizing the performance of the sensor network. Thus, our performance measures are the probability of true detection of the network $p_{\text{detection}}$ and the probability of a false intrusion alarm p_{false}.

The objective of a surveillance wireless sensor network (SWSN) design is to get a value $p_{\text{detection}}$ that is as high as possible and a value of p_{false} that is as small as possible. In this study, we explore numerically the possibility of affecting the values $p_{\text{detection}}$ and p_{false} of the sensors by arranging their locations as well as by exploiting multiple readings. In the numerical experiments, we estimate $p_{\text{detection}}$ and p_{false} for an SWSN. Numerically, these measures are defined as follows:

$$p_{\text{detection}} = \frac{N_{\text{detection}}}{N}, \tag{5.10}$$

$$p_{\text{false}} = \frac{N_{\text{false}}}{N}, \tag{5.11}$$

where $N_{\text{detection}}$ and N_{false} are the number of true and false detections respectively, while N is the total number of experiments, with or without the object in the area, respectively.

The experimental setup is as follows. The presence of an object in the SWSN is simulated N times, and the intrusion alarm is reported based on the readings of n individual sensors, according to the criteria of detection, e.g. as in Sections 5.3 and 5.4. Then $p_{\text{detection}}$ is computed by formula (5.10). In this study, N is set to 1000. To account for the variability of the simulation results, we have repeated all experiments 100 times. The estimate of $p_{\text{detection}}$ is represented by the average of the results as well as by the standard deviation. The results are also presented as a histogram, where the x-axis gives the values of the estimates obtained and the y-axis represents the relative frequency of occurrence of the estimates. The same experimental setup is used for computing the p_{false} of the SWSN by setting the object to reside outside of the SWSN coverage area for N consecutive times and

then using (5.11). In all the experiments presented here, the individual sensors are identical with $p = 0.9$ and $q = 0.02$. The other parameters are varied in the different examples to obtain the most demonstrative results.

To verify that our simulation gives correct estimates of $p_{\text{detection}}$ and p_{false}, we first perform an experiment using a simple sensor network of one sensor but with two consecutive readings. In this case, as suggested by the Viterbi algorithm from Section 5.4, the criterion of an intrusion alarm is that the sensor raises an alarm in two consecutive readings. Since the two readings are independent of each other, we have $p_{\text{detection}} = p^2 = 0.81$ and $p_{\text{false}} = q^2 = 0.0004$. The numerical results shown in Figure 5.7 demonstrate that the numerical method gives accurate estimates.

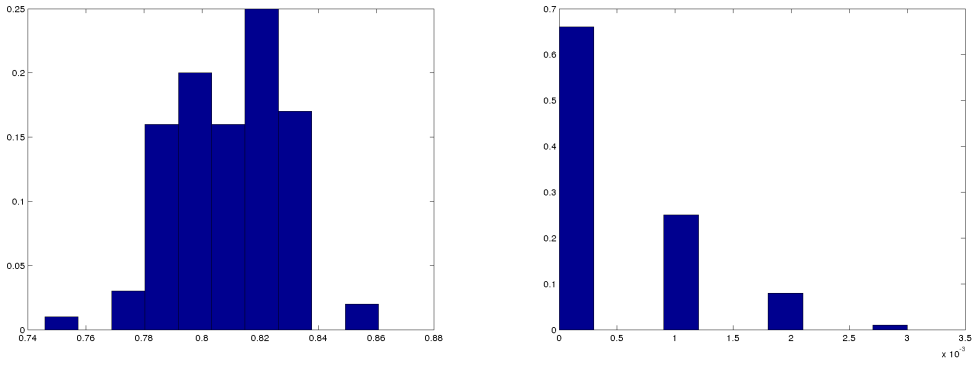


Figure 5.7: (Left) Estimate of $p_{\text{detection}}$ for one sensor with two consecutive readings. The mean is 0.8093, the standard deviation is 0.0181. (Right) Estimate of p_{false}: the mean of the estimate is 4.4×10^{-4} and the standard deviation is 6.9×10^{-4}.

In the example above, we have verified that our simulation program gives correct estimates of $p_{\text{detection}}$ and p_{false}. As a next step, in our simulation model we will combine the results on sensor deployment and intruder detection from the previous sections to detect a moving target. The area of interest is assumed to be the unit square, defined by $x \in [0, 1]$ and $y \in [0, 1]$, where (x, y) represents the location of a point. We describe the motion of an object using the white noise acceleration model described e.g. in [3, p. 263]:

$$x_o(t_{k+1}) = x_o(t_k) + v_x dt + \sqrt{dt} a_x \eta_x(t_k), \tag{5.12}$$

$$y_o(t_{k+1}) = y_o(t_k) + v_y dt + \sqrt{dt} a_y \eta_y(t_k), \tag{5.13}$$

where $(x_o(t_k), y_o(t_k))$ represents the object coordinate at time t_k, dt the time step, v_x and v_y the velocity in the x and y direction, respectively, a_x and a_y the acceleration terms, and η_x and η_y the noise terms, which are independent standard-normally distributed at each time step. The values of v_x, v_y, a_x and a_y are all set to 0.01 and dt is equal to 0.1. For illustration, we presented two realizations of the object's motion in Figure 5.8.

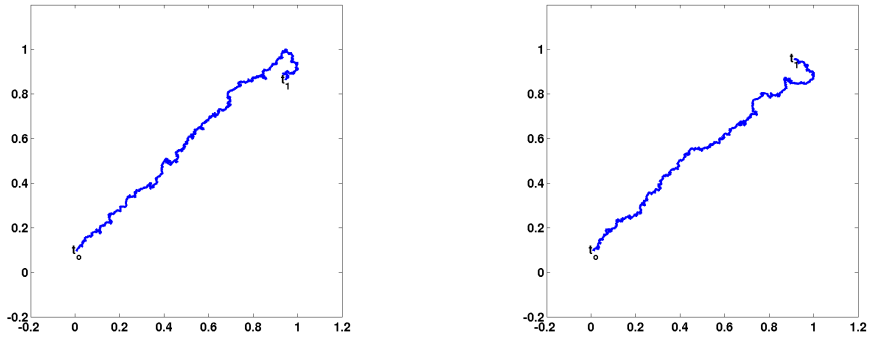


Figure 5.8: Some realizations of the object path used in the experiments; t_0 is the starting point and t_1 is the end point.

Now we would like to investigate the impact of sensor deployment. To this end, consider an SWSN consisting of eight individual sensors. Three different sensor arrangements are studied. In arrangements A and B, the locations of the sensors are determined randomly. In arrangement C, the sensors are located according to the MPD deployment algorithm from Section 5.2.2. Therefore, the sensors in arrangement C are located along a diagonal of the area of interest since these are the most likely locations of the object. The SWSN arrangements are depicted in Figure 5.9. The position of the object is depicted by an asterisk and the sensor that gives an intrusion alarm by a highlighted circle.

In this study, we have computed the $p_{\text{detection}}$ and p_{false} of the three sensor networks by exploiting the multiple readings by each sensor. Since the sensing ranges practically do not overlap, we are in the situation of Case II of Section 5.3.1. However, since each sensor raises an alarm based on the results of k readings according to the decision rule from Table 5.6, we have to adjust the probabilities p and q to the detection probability $p(k)$ and the false alarm probability $q(k)$ for $k = 1, 2, 3$. Simple calculations give:

$$\begin{aligned} p(1) &= p, & q(1) &= q; \\ p(2) &= p^2, & q(2) &= q^2; \\ p(3) &= p^3 + 3p^2(1-p), & q(3) &= q^3 + 3q^2(1-q). \end{aligned}$$

According to Table 5.1, the critical value for $q = 0.02$ is 1, that is the SWSN should give an intrusion alarm if the alarm is coming from at least one of the sensors. Since $q(2)$ and $q(3)$ are smaller than $q = 0.02$ the critical value remains the same if we use multiple readings from each sensor. Thus, if k readings of each sensor are used at each time point, for our three SWSN arrangements we have

$$p_{\text{detection}} = p_{\text{coverage}} \cdot p(k), \tag{5.14}$$

$$p_{\text{false}} = 1 - (1 - q(k))^8, \tag{5.15}$$

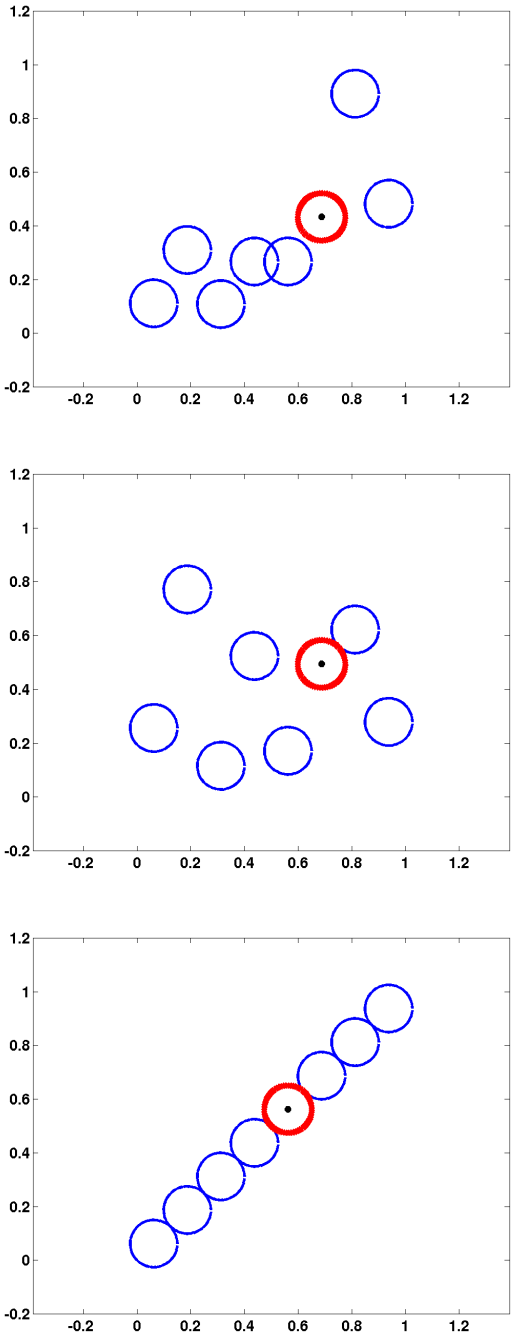


Figure 5.9: Example of sensor networks: (top) network A; (middle) network B; (bottom) network C.

where p_{coverage} is the probability that the object is within the network coverage, i.e., within one of the sensor ranges. Clearly, $p(k)$ in (5.14) and p_{false} in (5.15) are identical for the layouts A, B and C if the same number of readings is used. In this case, the performance of the SWSN is determined by how likely the object will pass through the network coverage, allowing the network to detect the existence of the object. This relative frequency is the estimate of the probability p_{coverage} that affects $p_{\text{detection}}$ in (5.14).

To estimate p_{coverage}, we simulate the object's motion into the area for each sensor network and compute the relative frequency of the object passing through the sensor network coverage. As in the previous experiments, the object is allowed to move inside the area of interest for 1000 time steps. Moreover, the experiments are repeated 100 times to account for the variability in the estimates. The results are presented in Figure 5.10. The estimates of p_{coverage} are 0.2884, 0.1420, and 0.6367 for sensor network A, B, and C, respectively. The conclusion is that the SWSN C is more likely to detect the object than the others.

Now, consider an SWSN of 50 sensors deployed by means of the MPD algorithm from Section 5.2.2 (see Figure 5.11). As before, the advancing of the object in the area is described by (5.12) and (5.13), where we choose $v_x = 0.02$, $v_y = 0.02$, $a_x = 0.001$, $a_y = 0.01$. Again, we report an intrusion alarm if a sensor signals an intruder in two consecutive readings, as suggested in Table 5.6 in case of two observations. In Figure 5.11, we show one time instant of a simulation run. An asterisk denotes the object position. The two overlapping highlighted circles depict the two sensors that give a correct intrusion alarm. The highlighted circle that does not contain the object, gives a false alarm.

For this network, the rate of false intrusion alarms is 0.0004. Furthermore, since the SWSN consists of an ample amount of sensors, our deployment strategy ensures that p_{coverage} (almost) equals one. The histogram for the detection probability at each time point is given in Figure 5.12. The high values of $p_{\text{detection}}$ are due to a considerable overlap of sensor ranges for the most likely positions of the object.

5.6 Conclusions

In this paper, we addressed two problems concerning design and performance of an SWSN: sensor placement and object detection. For the first problem, we suggest to use a hexagonal placement for optimal coverage. Further, we recommend to cover most vulnerable locations first, but avoid an overlap in sensor ranges unless the distribution of the object position is highly irregular. As a rule of thumb, one may call a distribution highly irregular if there exist pairs of points such that the distance between two points in such a pair is $\leq 2r$ while the value of the density differs by a factor $1 - p$.

For the detection problem, we state that several observations of the same object are absolutely necessary to report an alarm with reasonable certainty. A classical

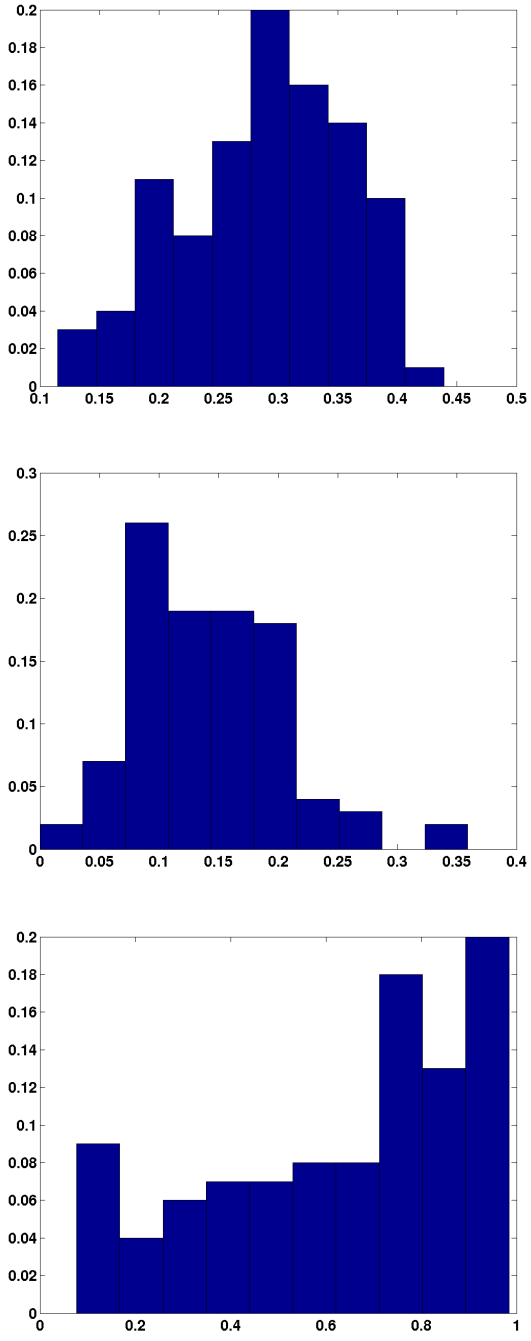


Figure 5.10: Estimate of p_{coverage} of SWSN. (Top) Network A. The mean of the estimate is 0.2884 and the standard deviation is 0.0698. (Middle) Network B. The mean of the estimate is 0.1420 and the standard deviation is 0.0631. (Bottom) Network C. The mean of the estimate is 0.6367 and the standard deviation is 0.2658.

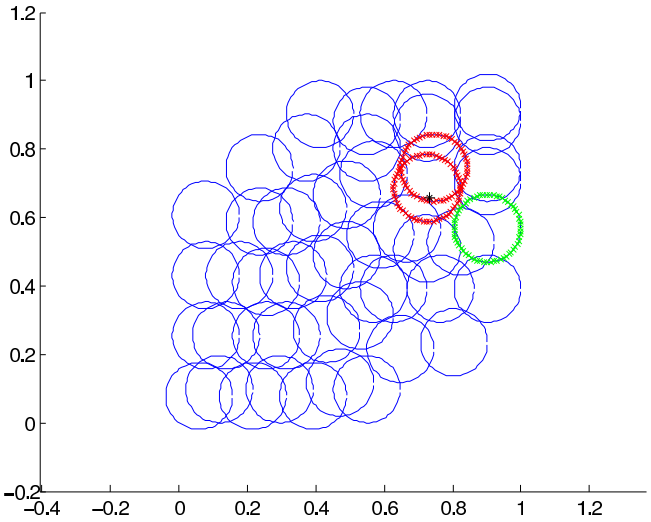


Figure 5.11: One time instant of a simulation run of the SWSN of 50 sensors containing a moving object (∗). Highlighted circles: two correct intrusion alarms and one false alarm.

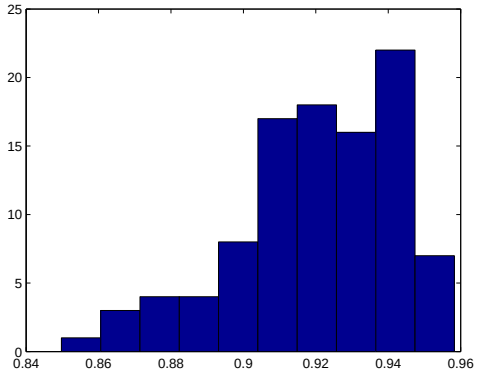


Figure 5.12: Estimate of $p_{\text{detection}}$ of the SWSN in Figure 5.11. The mean of the estimate is 0.9205 and the standard deviation is 0.0224.

hypothesis testing works well only if multiple sensors overlap in the same location. Otherwise, one must use information from consecutive readings of the SWSN. In the latter case, either a Bayesian approach or a hidden Markov model (HMM) approach can be used for object detection. To the best of our knowledge, the HMM approach involving the Viterbi algorithm to filter out the noise of non-detections and false alarms, has never been used in an SWSN before. The advantage of this approach is that it allows to pre-compute off-line all observation patterns that signal an intruder. Then the decision rule is very simple: report an intruder if one of the alarming patterns is observed. The HMM techniques in the SWSN context definitely deserve further study.

In this research, one could clearly see that the two problems under consideration are closely related. Although each of the proposed methods may be useful in its own right, it is essential to develop an integral approach to sensor deployment and intruder detection, in order to enhance the SWSN performance. In the last numerical example (see Section 5.5), we demonstrated that our techniques can be successfully combined, thus considerably increasing the efficiency of the network.

We would like to add that, potentially, our methods can be also used for tracking a target advancing through the area. For instance, by observing a simulation run of a moving object in the last numerical example, one could see that in spite of occasional false alarms, the correct intrusion alarms indicate a clear path that can be easily deciphered from multiple sensor readings.

Bibliography

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *IEEE Communications Magazine*, 40(8):102–114, August 2002.

[2] E. Ardizzone, M. La Cascia, G. L. Re, and M. Ortolani. An integrated architecture for surveillance and monitoring in an archaeological site. In *Proceedings of the third ACM international workshop on Video surveillance & sensor networks*, pages 79–86, Hilton, Singapore, 2005. ACM, New York, NY, USA.

[3] Y. Bar-Shalom and X. R. Li. *Estimation and Tracking: Principles, Techniques and Software*. Artech House, 1993.

[4] J. H. Conway and N. J. A. Sloane. *Sphere Packings, Lattices and Groups*, volume 290 of *Grundlehren der mathematischen Wissenshaften*. Springer-Verlag, New York, third edition, 1998.

[5] H. Delic and P. Papantoni-Kazakos. Robust decentralized detection by asymptotically many sensors. *Signal Processing*, 33(2):223–233, 1993.

[6] G.D. Forney Jr. The Viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, 1973.

[7] T. He, S. Krishnamurthy, L. Luo, T. Yan, L. Gu, R. Stoleru, G. Zhou, Q. Cao, P. Vicaire, J. A. Stankovic, T. F. Abdelzaher, J. Hui, and B. Krogh. Vigilnet: An integrated sensor network system for energy-efficient surveillance. *ACM Transactions on Sensor Networks*, 2(1):1–38, 2006.

[8] T. He, S. Krishnamurthy, J. A. Stankovic, T. Abdelzaher, L. Luo, R. Stoleru, T. Yan, L. Gu, J. Hui, and B. Krogh. Energy-efficient surveillance system using wireless sensor networks. In *Proceedings of the 2nd international conference on Mobile systems, applications, and services*, pages 270–283, New York, NY, USA, 2004. ACM.

[9] P. Korteweg, M. Nuyens, R. Bisseling, T. Coenen, H. van den Esker, B. Frenk, R. de Haan, B. Heydenreich, R. van der Hofstad, J. in 't panhuis, L. Spanjers, and M. van Wieren. Math saves the forest: analysis and optimization of message delivery in wireless sensor networks. In Erik Fledderus, Remco van der Hofstad, Ellen Jochemsz, Jaap Molenaar, Tim Mussche, Mark Peletier, and Georg Prokert, editors, *Proceedings 55th European Study Group Mathematics with Industry Eindhoven 2006*, pages 117–140. Technische Universiteit Eindhoven, 2006.

[10] E. Onur, C. Ersoy, H. Delic, and L. Akarun. Surveillance wireless sensor networks: Deployment quality analysis. *IEEE Network*, 21(6):48–53, November – December 2007.

[11] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[12] R. Roman, C. Alcaraz, and J. Lopez. The role of wireless sensor networks in the area of critical information infrastructure protection. *Information Security Tech. Rep.*, 12(1):24–31, 2007.

6 Modeling and simulation of phase-transitions in multicomponent aluminum alloy casting

Andreas ten Cate[1] Bernard J. Geurts[2*] Michael Muskulus[3]
Daniel Köster[2] Adrian Muntean[4] Joost van Opheusden[5]
Alex Peschansky[6] Bert Vreman[7] Paul Zegeling[8]

Abstract

The casting process of aluminum products involves the spatial distribution of alloying elements. It is essential that these elements are uniformly distributed in order to guarantee reliable and consistent products. This requires a good understanding of the main physical mechanisms that affect the solidification, in particular the thermodynamic description and its coupling to the transport processes of heat and mass that take place. The continuum modeling is reviewed and methods for handling the thermodynamics component of multi-element alloys are proposed. Savings in data-storage and computing costs on the order of 100 or more appear possible, when a combination of data-reduction and data-representation methods is used. To test the new approach a simplified model was proposed and shown to qualitatively capture the evolving solidification front.

Keywords: aluminum casting, alloys, phase-transition, mushy zone, moving solidification front

[1]Molten Aluminum Processing, Corus RD&T, IJmuiden
[2]University of Twente
[3]Leiden University
[4]Eindhoven University of Technology
[5]Wageningen University
[6]Delft University of Technology
[7]Vreman Research, Hengelo, bert@vremanresearch.nl
[8]Utrecht University
[*]corresponding author, b.j.geurts@utwente.nl

6.1 Introduction

In aluminum half products such as direct-chill (DC), cast ingots (aluminum blocks of $0.5 \times 1.5 \times 6\,\mathrm{m}^3$), and billets (aluminum poles of $0.2 - 0.5\,\mathrm{m}$ diameter and $6\,\mathrm{m}$ length) the spatial distribution of alloying elements is very important. In advanced aluminum products a considerable number of elements (typically elements such as Cu, Cr, Fe, Mg, Mn, Si, Zn) is involved at small to intermediate concentrations. These elements are very important as they determine the specific properties of the final alloy such as strength, fracture toughness, hardness, brittleness, dent resistance, surface quality et cetera. The aluminum research in industries such as Corus aims at developing new products for demanding applications such as the aerospace and automotive markets. It is the objective of this research to optimize the specific properties of the alloy for the particular applications by modifying the alloy composition in a generally narrow composition window. The consistency and homogeneity of the cast product in the solid phase is a prime aspect of casting technology. However, due to the casting process the homogeneity of the cast products may be compromised. Understanding and controlling the mechanisms that contribute to formation of spatial heterogeneity, also called macrosegregation, is therefore crucial.

In the casting process initially all elements in the mixture are in the liquid phase and spatially well-mixed. In semi-continuous casting of aluminum alloys the liquid metal is poured into a cooled mould. The molten metal is chilled by contact with the mould and application of cooling water. As the temperature decreases solidification sets in and a front between the already solidified and the still liquid part develops. It is exactly this transition band between solid and liquid, also known as the 'mushy zone', that plays a crucial role in the uniformity and hence the quality of the final cast product. Upon solidification the elements tend to redistribute between the solid and the liquid phases. Each element does this in its own manner, which is controlled by the thermodynamic equilibrium. Consequently, the liquid phase can become enriched and the solid phase can become depleted in elements. Local transport of the liquid phase due to shrinkage induced straining of the solid phase and due to buoyancy driven flow effects in the liquid part of the domain thus will cause redistribution of the elements on the scale of the ingot or billet cross-section. For a comprehensive overview of macrosegregation literature see [1].

This partial segregation is detrimental to the quality of the resulting cast and generally the resulting cast is beyond repair. As a consequence the resulting product is off-spec and has a reduced economic value or becomes rejected, which results in recycling of the entire cast product and obvious economic loss. These additional production costs can potentially be reduced if a more precise understanding of the origin of these cast defects can be obtained. In this paper we describe mathematical models that aim to simulate the details of solidification and transport induced segregation that take into account a large number of different species. We specifically present efficient methods for including in a computationally efficient manner the complex thermodynamics that characterize the solidification of many-species

mixtures used in modern aluminum products.

In casting technology research over the last decennia many numerical models have been developed for the prediction of the many different physical phenomena involved in the casting process (for example [2, 3, 4]). Such computational models generally predict the fluid flow in the liquid part, compute the solidification (the transition from liquid to solid) and calculate how the metal deforms when cooling down. These models often assume a constant composition throughout the domain. This assumption ignores the effect of spatial segregation, which is at the heart of current aluminum casting problems. The crucial step for simulations of industrially relevant alloys is that a large number of elements (about five or more) should be included in the simulations to achieve a proper modeling of the processes and phase-transitions. This leads to a strong increase in the simulation times. The challenge is to propose computational strategies to establish this crucial step in an efficient way.

Currently solidification models are under development that include the variation of composition during solidification (e.g. [5, 6, 7]). This requires that the relation between the local composition and temperature is computed. To a good approximation, this relationship is determined by considerations of thermodynamic equilibrium. A key element is the phase diagram, which gives the relation between phases, composition and temperature. For a binary mixture this already results in a complex parameter-space with widely different transitions in different regions. In case of a realistic multi-element mixture the complexity of the thermodynamic representation rapidly increases. Direct coupling of a thermodynamic database to a solidification simulation may impose limitations to the practical applicability.

In simulations of the casting process that include the effect of composition, the thermodynamic equilibrium needs to be determined each time step and in each grid cell. Commercial software is available to compute the thermodynamic equilibrium via a minimization of the Gibbs free energy (examples are [8], factsage[9], jmatpro[10]), but this is a computationally time consuming step. A direct coupling between the database and the casting simulation will result in infeasible simulation times. The challenge is to propose efficient coupling methods between the solidification simulation and the thermodynamic database. The question is how the solidification path in the computations can be constructed in a computationally efficient manner, considering that thermodynamic equilibrium data contains highly irregular features such as discrete transition points (e.g., an eutectic point) and large variations in the regions in which phase equilibria appear (e.g., some phases appear over a range of 5 Kelvin, others are present over several hundred Kelvin). One approach applied and presented in this work is to adopt local polynomial fits to thermodynamic data. This resulted in a significant reduction of the computational expense with full recovery of the physical properties of the casting process within the required numerical accuracy. The problem posed by CORUS to the 63rd *European Study Group Mathematics With Industry* was twofold: (1) Propose a simple PDE model for the simulation of the aluminum casting process and methods to establish an efficient coupling between the thermodynamic database and the involved PDEs.

(2) Assure that the model can simulate efficiently an industrially relevant number of alloying elements.

In this paper we review the continuum modeling in Section 6.2 and present an efficient method for simplifying the complex and computationally intensive thermodynamics that occur in Section 6.3. A one-dimensional numerical model will be adopted in Section 6.4 to illustrate the basic physical processes arising in the casting process, emphasizing the treatment of the solid-liquid mushy zone front. Finally, concluding remarks will be collected in Section 6.5.

6.2 Modeling transport and phase-transitions in multi-component aluminum casting

In this section, we present a complete model for transport and phase transitions that occur during the aluminum casting process. Our aim here is not to redo more involved mathematical models describing aluminum casting (e.g., [2, 12]), but to find a simple, yet realistic description of fluid flow and solidification of an aluminum alloy which allows to develop and test techniques for handling the multi-element thermodynamics during solidification. The formulations will result in the definition of a one-dimensional model that will be used in Section 6.4 for testing the thermodynamics evolution and to assess whether the main characteristics of the casting process can be recovered.

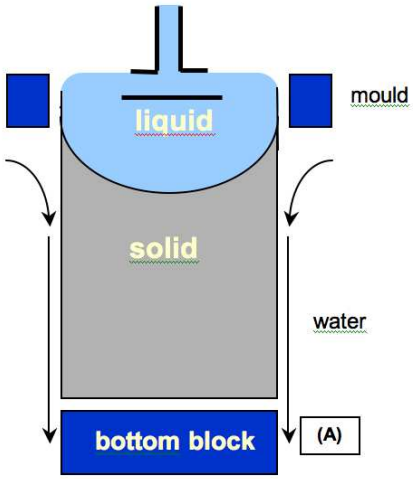


Figure 6.1: Sketch of the basic geometry in the aluminum casting process. The bottom block is continuously lowered as liquid aluminum is added on the top. Throughout water is applied for cooling the boundary of the aluminum block.

We consider a spatial domain split into a solid and a liquid region, see Fig. 6.1. The two regions are separated by a mushy zone, whose exact position has to be cal-

culated along with the flow and temperature fields. To describe the fluid mechanics and solidification physics a number of unknowns needs to be introduced. We refer to Table 6.1 for the unknowns of the problem as well as Table 6.2 for the list of the necessary 'parameters'. We refer to these as parameters, although strictly speaking their values are functions of the primary unknowns (e.g., the latent heat Δh is a function of the molar concentrations of various alloy elements, thermal properties of the hosting material and, of course, of the local temperature). For simplicity, we assume no pouring of liquid material into the solid domain and neither changes nor motion of the physical domain.

Notation	Dimension	Description
ϵ_l	1	local volume fraction occupied by liquid
$\epsilon_s := 1 - \epsilon_l$	1	local volume fraction occupied by solid
c_l^X	$\mathrm{mol/m^3}$	molar concentration of material X in liquid
c_s^X	$\mathrm{mol/m^3}$	molar concentration of material X in solid
$\mathbf{v}$	m/s	fluid velocity
p	$\mathrm{kg/(ms^2)}$	fluid pressure in liquid and mushy region
T	K	temperature

Table 6.1: Unknowns of the model.

Notation	Dimension	Description
m^X	kg/mol	molar mass of species X
ν, ζ	$\mathrm{m^2/s}$	kinematic standard/bulk viscosity of liquid
$\mathbf{g}$	$\mathrm{m/s^2}$	gravitational acceleration
K	$\mathrm{m^2}$	permeability tensor in the mushy zone
κ	$\mathrm{kg\,m/(K\,s^3)}$	heat conductivity
Δh	$\mathrm{kg/(m\,s^2)}$	latent heat of phase transition
C_p	$\mathrm{kg\,m^2/(K\,s^2)}$	heat capacity at constant pressure

Table 6.2: Parameters of the model.

Our model consists of conservation laws for the liquid and solid mass of all alloy elements $X_1, \ldots, X_N$, the averaged momentum of the fluid flow, and the total internal energy. Since the formation of micro-structure (dendrites, see Fig. 6.2) creates a mushy environment with a definite porous structure of the material, the momentum equation is formally replaced by the conceptually simpler Darcy law; see, e.g., [11]. The unknown ϵ_l serves to distinguish between those parts of the domain that are currently liquid, mushy, or solid. Note that, e.g., the "liquid" region could be defined as that part of the domain with $\epsilon_l \in (0.9, 1]$.

As a first step toward the mathematical model we present the equations describing conservation of mass of each individual element X participating in the solidification process. We express the balance of mass of the liquid and solid species separately.

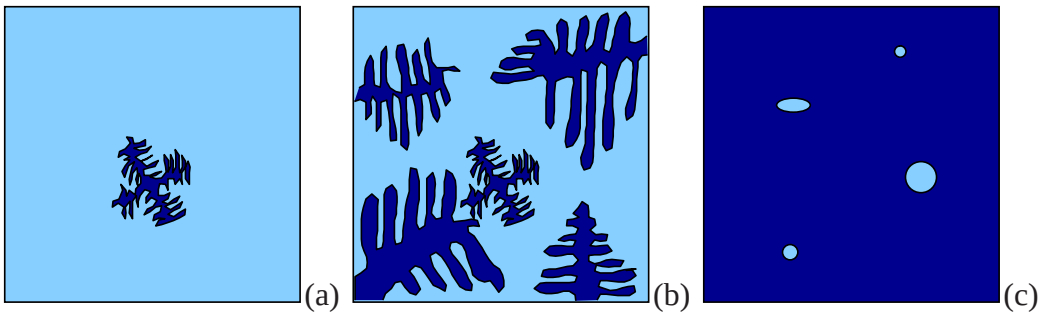


Figure 6.2: Local conditions in liquid (a), mushy (b), and solid regions (c) of the domain.

Assuming that diffusion due to concentration gradients is negligible at the characteristic flow time scale in the solidification process, the conservation of mass of species X in the liquid state is

$$\frac{\partial \epsilon_l c_l^X}{\partial t} + \nabla \cdot \left(\mathbf{v} \epsilon_l c_l^X\right) = 0. \tag{6.1}$$

This evolution equation assures that the integral of $\epsilon_l c_l^X$ over any volume Ω in the flow domain can change only due to fluxes through the boundary of Ω. Similarly, the conservation of mass of species X in the solid state reads

$$\frac{\partial \epsilon_s c_s^X}{\partial t} + \nabla \cdot \left(\mathbf{v} \epsilon_s c_s^X\right) = 0. \tag{6.2}$$

To characterize the flow in this scenario, we distinguish between liquid, mushy and solid zones. The balance equation for the linear momentum, which applies in the liquid zone, is given by

$$\frac{\partial m_i}{\partial t} + \nabla \cdot (m_i \mathbf{v}) = g_i \rho + \frac{\partial \sigma_{ij}}{\partial x_j}, \tag{6.3}$$

for $i, j = 1, \ldots, 3$. The liquid is considered incompressible with ρ is constant. Throughout, we adopt the Einstein convention on summation over repeated indices. Here, we have used the total momentum density m_i in the x_i direction

$$\begin{aligned} m_i &= v_i \rho, \\ \rho &= \rho_l + \rho_s = \epsilon_l c_l^{X_k} m^{X_k} + \epsilon_s c_s^{X_k} m^{X_k}, \end{aligned}$$

with $k = 1, \ldots, N$. The two terms on the right-hand side of (6.3) represent gravity and viscous drag, modeled as Newtonian fluid for simplicity:

$$\sigma_{ij} = -p\delta_{ij} + \nu \left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i} - \frac{2}{3}\eta \frac{\partial v_l}{\partial x_l}\delta_{ij} \right) + \zeta \frac{\partial v_l}{\partial x_l}\delta_{ij}.$$

In the mushy zone, solidified alloy dendrites form a dense porous medium. Inspired by [12], we use Darcy's expression to relate velocity and pressure:

$$v_i = -\frac{1}{\epsilon_l \nu \rho_l} K_{ij} \frac{\partial p}{\partial x_j}.$$

This is only an Ansatz. A rigorous derivation via homogenization-type arguments is still needed (see, e.g., [13]). Finally, the solid zone (the aluminum) is described as a state of rest:

$$\mathbf{v} = 0.$$

In the liquid region ($\epsilon_l \approx 1$) we define the velocity through solving the momentum equation, in the solid zone ($\epsilon_s \approx 1$) we use the state of rest and in the remaining mushy zone the Darcy formulation is chosen. Temperature and energy dynamics is sketched next. We express the total internal energy density as

$$e^* = C_p T + \frac{1}{2} v_j^2 + \text{const.}$$

The conservation of total internal energy is given by

$$\frac{\partial}{\partial t}(\rho e^*) + \nabla \cdot (\rho e^* \mathbf{v}) = Q + \nabla \cdot (p\mathbf{v}), \tag{6.4}$$

with the heat source rate expressed as

$$Q = \nabla \cdot (k \nabla T) + \Delta h \frac{\partial \epsilon_l}{\partial t}.$$

Heat is thus added to the system by the liquid-solid phase transitions taking place in the mushy region, expressed by the latent heat Δh, as well as by heat conduction with coefficient k (Fourier's law). Viscous heating due to friction is neglected.

Besides the calculation of the model parameters (which typically depend on the unknowns of the problem), we need to close our model by additional constitutive relations. Here we suggest two such relationships. In principle, (local) thermodynamic properties could be used to determine the pressure as a function of temperature and species concentrations:

$$p = F_1(T, c_l^{X_1}, \ldots, c_l^{X_k}). \tag{6.5}$$

Alternatively, we could use information from thermodynamic phase diagrams to calculate the liquid fraction

$$\epsilon_l = F_2(T, c_l^{X_1}, \ldots, c_l^{X_k}, p). \tag{6.6}$$

The evaluation of (6.5) (or (6.6)) can be based on information available from thermodynamic databases. Only one of these two expressions needs to be selected -

which particular one is chosen may depend on the application. Furthermore, to solve the pressure a more involved analysis is required in which (6.5,6.6) do not play an important role. The closure poses the problem of efficiently accessing the thermodynamic information, especially in the case when many species are present. Alternatively, this may be obtained via variational principles (by minimizing the corresponding Gibbs functional), which poses the problem of simultaneously solving a PDE system and finding local minimizers to a non-linear non-convex functional. Both these approaches increase the computational effort. Considerable care in the reduction of the mathematical model and algorithm development is needed to achieve realistic costs of simulation. We present an approach based on local polynomial fitting in Section 6.3 and estimate theoretically the computational saving compared to a full gridding of the thermodynamic state-space.

In Section 6.4, some example calculations are given for a simplified one-dimensional model for a slow solidification process of a single species. This model can be readily appreciated as a special case of the general formulation given above. The purpose of this reduced model is to isolate the main characteristics of the solidification process and to test the efficiency of the evaluation with which thermodynamic properties such as Δh are being processed. The 1D model that is proposed can be written as

$$\begin{aligned} \frac{\partial T}{\partial t} - \frac{\partial^2 T}{\partial x^2} - L\frac{\partial \epsilon_l}{\partial t} &= 0, \\ \frac{\partial \epsilon_l}{\partial t} - M\frac{\partial^2 \epsilon_l}{\partial x^2} &= 0, \end{aligned} \tag{6.7}$$

where L is a coefficient related to the latent heat used to produce the phase transitions, while M is a constant effective diffusivity of the liquid. The rationale behind this model is that we neglect all fluid flow, thus $\mathbf{v} = 0$, i.e., both in the liquid and in the solid. Correspondingly, only diffusive transport for ϵ_l remains in this very crude model. In the absence of gravity and at constant pressure, the momentum equations are trivially fulfilled. It remains to discuss the energy conservation equation (6.4). Under the additional assumption that the parameters ρ, k, and C_p are constant, equation (6.4) yields $e^* = C_p T$ in which temperature is governed by

$$C_p \rho \frac{\partial T}{\partial t} = \nabla \cdot (k \nabla T) + \Delta h \frac{\partial \epsilon_l}{\partial t}.$$

If in addition $k = C_p \rho$, then the last equation reduces to

$$\frac{\partial T}{\partial t} - \frac{\partial^2 T}{\partial x^2} - L\frac{\partial \epsilon}{\partial t} = 0,$$

where $L = \Delta h/(C_p \rho)$ and we dropped the subscript l. The second equation of the simplified model (6.7) is then obtained by assuming that the liquid fraction is proportional to the temperature T, within some reasonable range of T. In this case,

the second equation of (6.7) is recovered. This model has an effect of latent heat being released, while the solidification front progresses pushed by diffusion. This is a particularly appealing model for numerical analysis and illustration of the main physics of solidification. We return to this in Section 6.4.

6.3 Thermodynamic representations and data reduction

Any CFD simulation of solidification of an alloy requires thermodynamic input in each fluid cell and at each time-step. This input may be the latent heat, the heat capacity, and the local predictions of phase concentrations and compositions that occur for a given temperature under certain thermodynamic assumptions. Minimization of the Gibbs functional 'on the fly', i.e., everywhere and anytime, is too time-consuming in this context. One way to circumvent this problem is to employ a thermodynamic database, which is also called a mapping file in the literature [18]. This database can be pre-computed by performing Gibbs minimizations for a large number of specific combinations of temperature and phase concentrations. The database is a discrete numerical representation of the information contained in the physical phase diagram. In general, the local temperature and phase concentrations in a fluid cell in the CFD simulation are not precisely equal to the available discrete values of the entries in the database. Interpolation is thus necessary, which is much less time-consuming than the Gibbs minimization computation itself. In this section we will pursue this method and incorporate polynomial fitting to reduce the storage requirements for the database. Theoretical estimates of the efficiency are also provided.

6.3.1 Polynomial fit

The problem with precomputed databases is that they easily become much larger than the present memory of computers. Consider for example an alloy solidified from the four materials Al, Cu, Fe and Mg. Then a thermodynamic quantity, such as the heat-capacity C_p, is dependent on temperature T and on three independent species concentrations c_1, c_2 and c_3, while the remaining one c_4 is given by $c_4 := 1-c_1-c_2-c_3$ in a non-dimensionalized situation. The function C_p then depends on 4 variables. If we would use a uniform grid for each of the four arguments, covered each by 600 points for sake of argument, we would need a memory of $2\times4\times600^4 =$ 4TB to store two thermodynamic quantities with single precision. Such a database approach has been considered in [18], where it was noted that calculations of up to four elements can thus be realized, but calculations involving five or more elements seem to be beyond reach at present. The aim of the present section is to investigate whether it is possible to reduce the size of the database strongly, without unduly affecting the accuracy of the thermodynamic input delivered to the CFD-simulation.

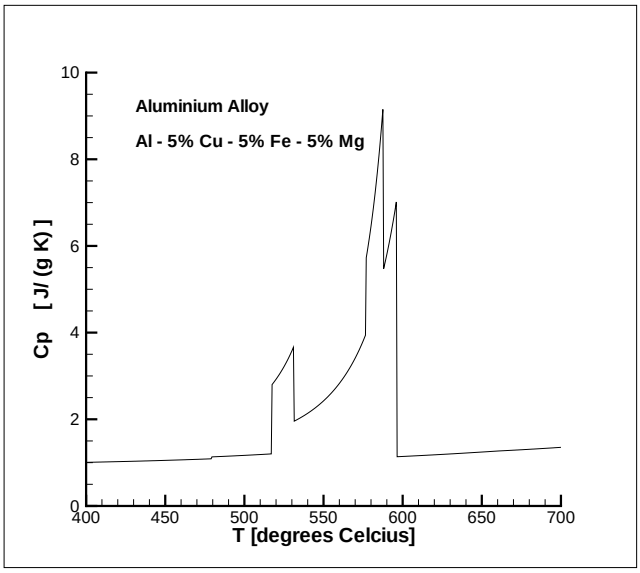


Figure 6.3: Example of the dependency of the heat capacity on temperature for fixed composition at 5% of Cu, Fe and Mg in an Al alloy.

Since the thermodynamic quantities in phase diagrams typically display strong jumps, a large number of grid points is necessary in each direction if a uniform grid is used for the entries of the database. Unstructured nonuniform meshing of the table automatically adapted to the shape of the phase diagram is expected to reduce the size of the grid needed to represent the table. That such a strategy leads to much smaller databases is illustrated in the remainder of this section by considering a simple example of homogeneous solidification.

The temperature in a process of homogeneous solidification of the alloy Al-Cu-Fe-Mg can be described by the following equation:

$$C_P(T, c_1, c_2, c_3)\frac{dT}{dt} = -Q < 0, \tag{6.8}$$

where T is the temperature, assumed to be spatially independent in this case, and Q the heat extracted from the system. The heat-capacity C_P is the so-called effective heat-capacity, in which the latent heat is included. Three concentrations c_1, c_2 and c_3 are needed to describe the concentration distributions, i.e., the relative amount of molecules of Al, Cu, Fe and Mg. For the present example we assume that the three concentrations of Cu, Fe and Mg are equal, $c_1 = c_2 = c_3 = 5\%$ (mass concentrations). Since the solidification process considered in the present example is homogeneous, the concentrations are constant in space, but also constant in time, because of mass conservation. Therefore, to solve (6.8) the thermodynamic database (the phase diagram) can essentially be reduced to the representation of C_P as a function of temperature.

We computed the temperature dependence of C_P under these concentration conditions for the Al-Cu-Fe-Mg system by minimizing the Gibbs free energy. The

result is shown in Fig. 6.3, clearly illustrating a central feature of the phase diagrams: strong jumps appear, but in between these 'discontinuities', the function is relatively smooth. Fig. 6.3 has been obtained with a uniform temperature grid containing 600 points. Obviously, C_P can be accurately captured with much less points if one would only store the locations at which the 'discontinuities' appear, while the smooth parts in between would be approximated by suitable polynomials. This basic observation will be worked out in more detail next, to show the principle.

To reduce the thermodynamic database storage we define a jump location by a threshold of $0.5\,\mathrm{J/(gK)}$, fixed a priori for simplicity. We consider two options for the smooth pieces between jumps: first-order (straight lines) and second-order Lagrange polynomials (parabolae). The coefficients of the polynomials can simply be computed from the values at and between two jumps. The two end-points of a smooth region are collocation points for the first-order but also for the second-order polynomial. For the second-order polynomial a third collocation point needs to be added. For this we take the point half way in the interval under consideration. Thus instead of 600 floating point numbers (uniform grid) we need to store much less floating point numbers to represent the behavior in Fig. 6.3 with piecewise continuous polynomials. In particular, we require only 17 numbers in case of linear polynomials, and 23 in case of second-order polynomials.

To assess the quality of the reduced data representations we solve (8) for the three different numerical representations of C_P. We compare (a) the fine-grid representation consisting on 600 uniformly distributed points, (b) a linear polynomial and (c) a second-order polynomial fit. In each case a four-stage Runge-Kutta method with a sufficiently small time-step is used to integrate the equation. The right-hand side is assumed to be constant and equal to $Q = -1\,\mathrm{J/(gs)}$. The results of the computations are shown in Fig. 6.4. The second order polynomial fit provides a very accurate approximation of the fully resolved case – there is no discernible difference between the curves based on method (a) and (b). It is concluded that in this example the size of the database can be reduced by a factor of around 30 without significant loss of accuracy (in this example a reduction from 600 data points to 17 or 23 in case linear or quadratic interpolation is used).

The homogeneous case above is very simple; C_P is reduced to a function of temperature alone because the concentrations remained constant. In practical CFD-calculations the concentrations change. Nevertheless, the above method can in principle also be applied to more practical cases: the temperature dimension can be treated as in the example above, using piecewise discontinuous polynomials, while the concentration dimensions are still treated with linear interpolation on uniform grids. If we would use a structured nonuniform meshing of the concentrations (clustering in the most important regions) for the Al-Cu-Fe-Mg alloy we might be able to obtain a reduction of a factor of 3 in each concentration reduction. Thus the total storage reduction would be a factor $30 \times 3^3 \approx 800$, such that the original database of 4TB would reduce to 5GB and thus fit well into the memory of any modern personal computer.

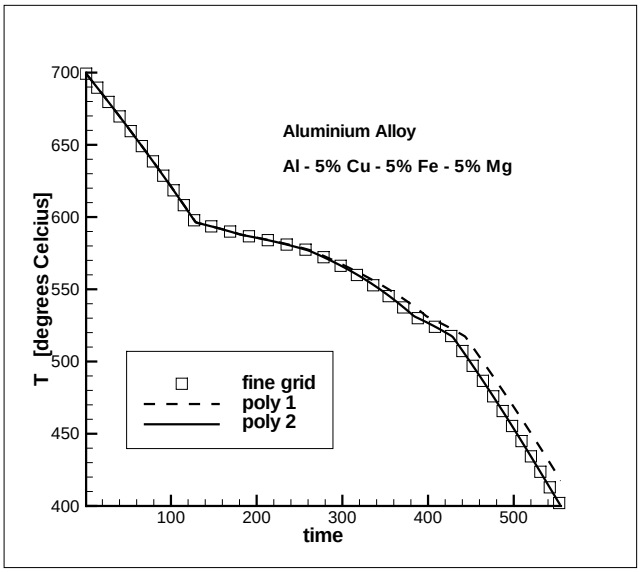


Figure 6.4: Simulation results for 1D solidification with constant composition.

The above basic approach to reducing the storage required for the thermodynamics database can be extended to a more complete computational data-representation scheme for demanding casting problems. In the next sections, we describe the main elements of this methodology.

6.3.2 Alternative approaches

In the following sections we consider an alternative approach based on a non-uniform mesh representation and discuss its merits and disadvantages. The development of this method has been guided by the following principles:

1. The thermodynamic quantities of interest fall into two different categories:

    a) Quantities that are smooth and change slowly with respect to changes in composition and temperature, for example: *enthalpies* and *phase composition* (what elements are present in a certain phase).

    b) Quantities that change abruptly and discontinuously, for example: *phase information* (what phases are present and in what relative amounts) and effective *heat capacities*.

2. Some regions of the phase diagram are more important and should be represented with higher accuracy than other regions of less interest. This is partly due to the occurrence of phase changes, but also since some of the elements are only present in rather small concentrations in the system, such that large parts of the phase diagram are (probably) never needed in a simulation.

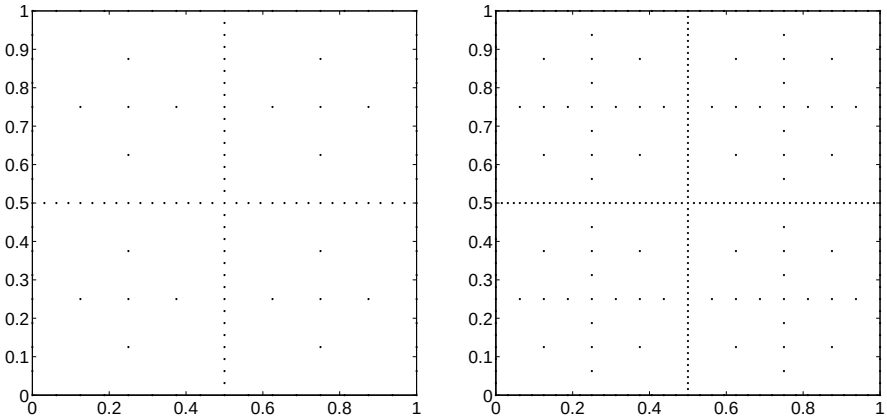


Figure 6.5: Example hierarchical sparse grids (Curtis-Clenshaw type) containing five levels of successive approximations (left panel) and six levels (right panel), respectively.

3. The evaluation of phase diagram data needs to be very efficient, such that complicated interpolation schemes are out of the question.

With regards to the last point, the optimal solution would be a database approach and multi-linear interpolation of the values of query points, which is very fast to implement, running in $O(n)$ time when the database is represented on a regular grid with n grid points per dimension. As has been noted in the previous section, however, such an approach is ultimately infeasible due to the large number of grid cells needed to represent the phase diagram accurately, the space complexity being of order $O(n^d)$, where d is the dimension.

It should be noted, though, that all thermodynamic quantities of interest, for example the heat capacities, can be derived from two ingredients alone: smoothly varying enthalpies and *phase information*. Were this phase information discrete, we could proceed with two different strategies:

1. Model the continuous enthalpies by some simple interpolation scheme.

2. Model the discontinuous phase boundaries separately.

The first point can be realized, for example, by a hierarchical representation on sparse grids [16], for which an efficient implementation in MATLAB is available [21]. The mean of a quantity of interest over the phase diagram is represented as a single number in the first node of the hierarchy, and more localized changes are represented by a number of sparsely distributed points at lower levels of the hierarchy. Fig. 6.5 shows an example of the sparse grids typically used at different levels of detail.

If the thermodynamic data were represented in *thermodynamic variables*[9] then the phase diagram would consist of a so-called *cell structure* [17], such that each cell represents one unique phase. Unfortunately, it is nontrivial to transform element concentrations into thermodynamic variables and vice versa. But when concentrations are used as variables, then *mixtures* of phases occur, where two or more phases are coexisting in the system in varying amounts. For binary systems, these phase mixtures can be described by analytical formulae, for example the *lever* rule [17], which is simply linear interpolation of two phases with respect to concentration, but already for ternary systems no such simple rule is available. This means that in certain regions of the phase diagram unfortunately not only the phase boundaries have to be represented, but also the complete phase information. On the other hand, this information is usually also smooth inside a given region in the phase diagram.

To conclude: In principle the thermodynamic information needed in actual simulations of solidification processes concerns either (1) smoothly varying data, or (2) discrete information about the phase boundaries. This distinction was already apparent in the example discussed in Subsection 6.3.1.

6.3.3 Tracing the phase boundaries

From the above it is clear that the biggest problem in the efficient calculation of thermodynamic properties is the accurate representation of the boundaries of the phase diagram. These boundaries form a $n-1$ dimensional hyper-surface if the system is n dimensional, i.e., is described by the relative concentrations of n distinct elements and temperature. Note that concentrations have to sum up to one, so in fact there are only $n-1$ independent concentration variables to consider. In a binary system, the phase boundaries are one-dimensional, for example.

In general, one can distinguish two basic approaches for the representation of hyper-surfaces such as occur in the thermodynamic closure describing the phase transitions. An *explicit* surface is represented by some parametric surface, given by a multidimensional spline, for example, or a representation as an unstructured grid by simplices. In two dimensions the latter is often realized by a Delaunay triangulation [15]. On the other hand, an *implicit* surface is represented by a number of smooth, local basis functions and the surface is defined as an iso-contour of a scalar function. This method is attractive, since it allows to trace surfaces elegantly and accurately by level set methods [25], but unfortunately the computational costs can be very high.

Since we need phase boundary information for the approach outlined in the following section, we describe here a simple method to trace the boundaries. The information obtained consists of a number of points lying very close to the actual phase boundaries (within a user-specified numerical tolerance) and can be used as input

[9] Thermodynamic variables form a complete set that uniquely describes a thermodynamical system. For the solidification process, these are usually taken to be the temperature, pressure and chemical potentials associated to the involved species

for the more advanced level set methods mentioned. The approach is demonstrated on a binary system consisting of the two elements Pb and Sn. Thermodynamic data for this system is available through calls to the CHEMAPP library[10] which is the calculational back end of the commercial CHEMSAGE software [19, 20]. The independent variables are temperature T and composition x. The latter measures the relative amount of Pb, such that $0 \leq x \leq 1$. The region of the phase diagram we considered was a temperature range of $320 \leq T \leq 620$, measured in Kelvin.

The boundaries of the phase diagram have been traced by a bracketing method. For simplicity, we have distributed a number of points (320) regularly along the T axis and then bracketed all points where a phase change occurs, varying x, by an iterative bisection method [24]. The algorithm stores two different concentration values $x_1 < x_2$ and evaluates the discrete phase information at both points. If a difference is found, the phase information at the middle point $x_{12} = \frac{x_1+x_2}{2}$ is evaluated. If the phase at x_{12} is the same as the one at x_1, then x_1 gets updated to x_{12}, otherwise x_2 gets updated. If the phase at the middle point is different from both phases at x_1 and x_2, respectively, both subintervals are (recursively) bisected. The algorithm continues until $|x_1 - x_2| < \epsilon$; here we used $\epsilon = 10^{-4}$.

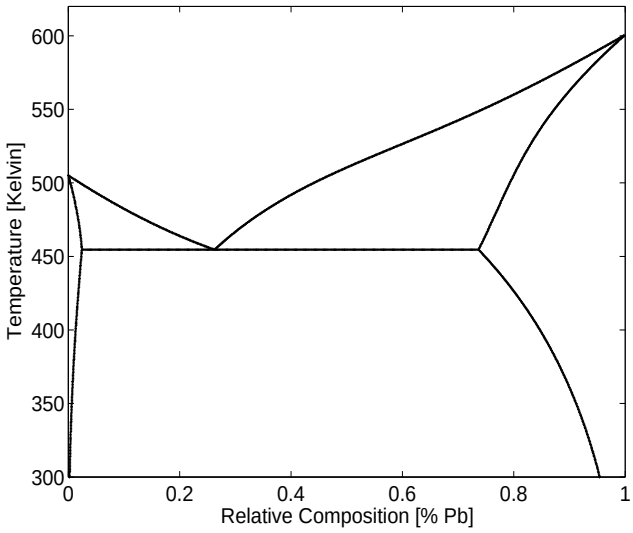


Figure 6.6: Traced phase diagram of Pb-Sn binary example system.

Complementing this "vertical" tracing, we have analogously distributed points along the x axis and bracketed all phase changes, varying T. For this horizontal tracing we have used 500 points. The resulting phase boundaries are shown in Fig. 6.6. In each of the six areas in the figure a physically different equilibrium state is found.

[10] A restricted version called CHEMAPP LITE is available for private, non-commercial use.
URL: `http://gttserv.lth.rwth-aachen.de/~cg/Software/ChemApp/`

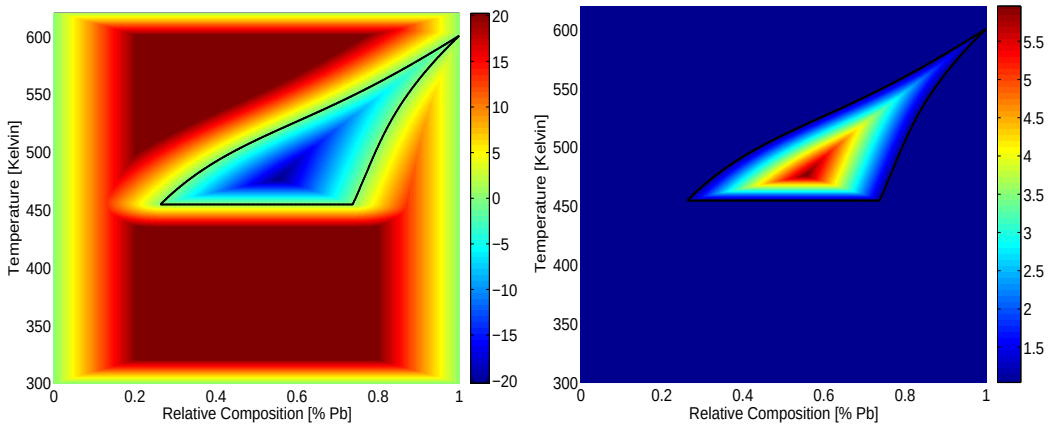


Figure 6.7: Distance (left panel) and size function (right panel) for the liquid-Pb mixture phase in the Pb-Sn binary example system.

6.3.4 Triangulation of phase regions

The next step is the triangulation of the different phase regions. These have been performed with the simple mesh generator developed in [22]. The input needed for this code is a signed distance function $d(T, x)$ that returns the distance to the nearest phase boundary, and a size function $h(T, x)$ that returns the desired edge length of the triangulation at each point, thereby allowing non-uniform adaptive meshing.

Fig. 6.7 shows the distance function for a certain phase region of the liquid-Pb mixture phase. We use the Euclidean distance

$$d(T, x) = \left((T - T^*)^2 + k\,(x - x^*)^2\right)^{1/2},$$

where (T^*, x^*) denotes the point on the phase boundary closest to (T, x) and $k = 200$ was used to weigh the contribution of concentration changes with respect to temperature changes. The distance function we used is interpolated on a regular grid, where the distance to the closest phase boundary point has been approximated by the minimum of the distances to the previously traced boundary points.

From this distance function, a size function has been computed. For simplicity, we used

$$h(T, x) = 1 + 10\ \exp\left(|d(T, x)/2d_0|\right),$$

where $d_0 = \min_{T,x} d(T, x)$ is the characteristic width of the phase region. Results of such an adaptive meshing are shown in Fig. 6.8.

In a practical application of this method, one needs to mesh the phase diagram separately in each region and then join the triangulations at the internal interfaces, i.e., the phase boundaries. A discussion of these issues can be found in [23]. Also, the size function should depend on the local accuracy level that is required. In fact, one can also consider a data-driven approach, where an actual simulation is

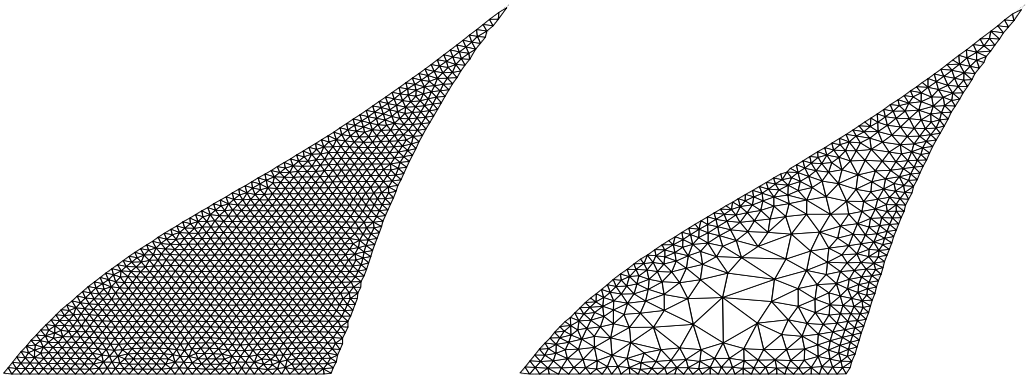


Figure 6.8: Example triangulation of the liquid-Pb mixture phase. Left panel shows results by uniform size function, right panel shows results by distance-dependent size function.

performed in which the locations in the phase diagram that are needed are recorded. From these data one can construct a density function $h(T, x)$, where regions of the phase diagram that are needed often in a simulation would be represented in more detail than regions that are needed rarely. Of course, one can also combine all these considerations into one common size function.

This method generalizes to n dimensions by replacing triangles (2-simplices) by n-simplices. Each simplex is then represented by $n + 1$ points and consists of $\binom{n+1}{2}$ edges. The storage requirements are therefore of order $O(n^2)$ in the number of simplices used. More importantly, when a CFD simulation needs to evaluate phase diagram information, first the corresponding simplex needs to be found, and then the values stored at its edges are linearly interpolated. The location of the simplex containing the query point is an example of a point location problem with a typical time complexity[11] of order $O(\log n)$ [15] in the number of stored simplices n, whereas the interpolation is linear.

6.3.5 Localized caching

From the above it should be clear that the problem of efficiently representing phase diagram information is quite difficult, and the familiar tradeoff between storage and time complexity is encountered. Probably the biggest savings in computing time can therefore be expected to be achieved on quite a different level. Recall that thermodynamic data is needed for each grid cell and at each time step, but (1) the local state in each cell (temperature, concentrations) usually changes slowly in between time steps, and (2) in most cases the local state changes slowly between spatially neighboring cells. An efficient implementation should therefore try to also make use of these two properties, recycling already computed thermodynamic data

[11] In MATLAB this is implemented in the function tsearch, which is based on the QHULL code [14] freely available from http://www.qhull.org/

as much as possible and only recomputing these data if absolutely necessary.

The basic idea is to store a pointer in each grid cell that points to the thermodynamic data used in the last time step. If the state of the cell does not change in a certain range, the thermodynamic data is reused without computation (in a more advanced implementation, linear changes could be taken into account and interpolated locally at each time step). Of course, the tolerance used would usually depend on the location in the phase diagram: close to a phase boundary the thermodynamic data of each cell should be updated more often than in the middle of a phase region. Furthermore, if the local state of a grid cell changes too much such that re-computation of thermodynamic data is necessary, the local structure of the grid could be used advantageously. Quite often a neighboring grid cell could have used the necessary data in the previous time step[12]. Only if no neighbor has the necessary data cached, a re-computation/lookup should be started. Even then, also the representation of the phase diagram could use local structure advantageously. Instead of $O(\log n)$ a constant time complexity (on the average) seems possible.

6.4 Computational modeling of solidification fronts

In this section we consider the PDE system (6.7) to illustrate some basic mechanisms that characterize a progressing solidification front. Emphasis is given in this model to the effects of latent heat release in the absence of flow. The model describes the phenomena in one spatial dimension only, roughly mimicking the behavior along the central axis of the ingot. It will be shown that a simple spatial discretization suffices to capture the physics of the problem and that the qualitative features of the solidification front are well captured. This implies that (6.7) can be used as an efficient vehicle for testing improvements in the thermodynamics treatment without leading to lengthy simulations. This can be beneficial in development stages of reduced thermodynamics representations, while retaining a clear view at the accuracy penalty incurred. In the future, it would be helpful to extend this simple model with a realistic thermodynamic description of the latent heat, to illustrate the computational gain that may be achieved with one of the approaches outlined above. Currently, this model is only used to illustrate the occurrence of solidification fronts in case the latent heat is only roughly parameterized.

We consider the coupled system of equations (6.7) on the unit interval $]0,1[$. The initial temperature is taken constant and larger than the melting temperature of the mixture, denoted by T_m. Moreover, we consider the initial state to be liquid, implying that at $t = 0$ we have $\epsilon_l = 1$ throughout the system. For convenience, we drop the index l and implicitly assume that $\epsilon := \epsilon_l$ refers to the volume fraction in the liquid phase. Fully solidified material corresponds then to $\epsilon = 0$. To complete the basic description, we impose Neumann conditions at $x = 0$, i.e., put $\partial\epsilon/\partial x(0,t) =$

[12] It even seems possible to use a grid cell's spatial neighbors to interpolate the thermodynamic values at that cell, sufficiently far away from phase boundaries at least

$\partial T/\partial x(0,t) = 0$ and Dirichlet conditions at $x = 1$, i.e., $\epsilon(1,t) = 0$ and $T(1,t) = T_0$ where, with proper non-dimensionalization $T_0 = 1 < T_m$, indicating that at $x = 1$ the solidification front starts:

$$\begin{aligned} \frac{\partial T}{\partial t} - \frac{\partial^2 T}{\partial x^2} - L\frac{\partial \epsilon}{\partial t} &= 0 \\ \frac{\partial \epsilon}{\partial t} - \frac{\partial^2 \epsilon}{\partial x^2} &= 0 \\ T(x,0) = 1 < T_m, \quad \epsilon(x,0) &= 1 \\ \frac{\partial T}{\partial x}(0,t) = \frac{\partial \epsilon}{\partial x}(0,t) &= 0 \\ T(1,t) = 1,\ \epsilon(1,t) &= 0 \end{aligned} \tag{6.9}$$

where, for convenience, we use a unit diffusion coefficient $M = 1$.

This problem can be readily discretized using standard finite differences and an explicit time-stepping method. For convenience, we formulate the discrete model on a uniform grid $x_j = jh$ where $h = 1/N$ denotes the mesh spacing. Likewise, we choose a constant time-step Δt and approximate the solution at times $t_n = n\Delta t$. Following the usual steps, we arrive at

$$\begin{aligned} \epsilon_j^{n+1} &= \epsilon_j^n + \nu(\epsilon_{j+1}^n - 2\epsilon_j^n + \epsilon_{j-1}^n) \\ T_j^{n+1} &= T_j^n + \nu(T_{j+1}^n - 2T_j^n + T_{j-1}^n) + \Delta t L_j^n \Big(\frac{\partial \epsilon}{\partial t}\Big)_j^n \end{aligned} \tag{6.10}$$

for $1 \le j \le n-1$. Here, $\epsilon_j^n \approx \epsilon(x_j, t_n)$ and $T_j^n \approx T(x_j, t_n)$. The term $(\frac{\partial \epsilon}{\partial t})_j^n$ is approximated backward in time. At the boundaries we put $T_N^n = 1$ and $\epsilon_N^n = 0$ and use the simple approximation for the Neumann boundary at $x = 0$ as: $T_0^n = T_1^n$ and $\epsilon_0^n = \epsilon_1^n$. In this formulation $\nu = \Delta t/h^2$ which has to be kept sufficiently small in order to maintain stability of the simulation.

The effect of heat released during solidification is represented by the function L. Purely intuitively, one may expect L to be large in case the temperature is close to the melting temperature and considerably smaller at temperatures away from the melting temperature. Suitably normalized, the simplest possible discrete model for L is

$$L_j^n = \begin{cases} \beta & \alpha T_m < T_j^n < T_m \\ 1 & \text{otherwise} \end{cases} \tag{6.11}$$

where for illustration purposes we assume $\beta \gg 1$. More involved models for L can be obtained analogously to that presented in Section 3. However, at this level of detail it is sufficient to indicate the effect of heat release in this crude modeling.

Simulating the solution to the simple model can be done with a straightforward MATLAB implementation. For this purpose we adopted $T_m = 2$, $\beta = 100$ and

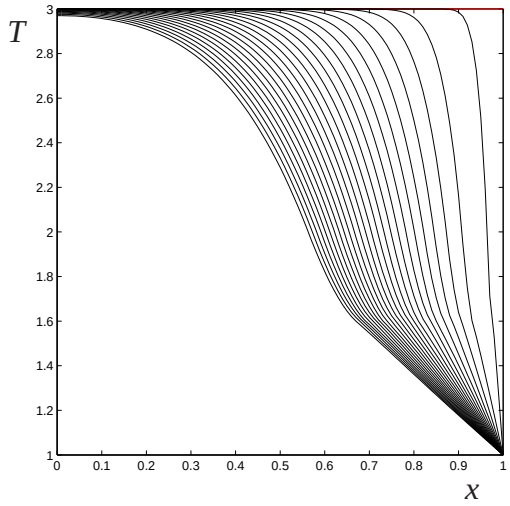


Figure 6.9: Developing temperature profile characterizing the solidification front. The solid develops from the right - subsequent curves correspond to snapshots at different times.

$\alpha = 0.8$. The moving solidification front that is obtained in this way is shown in terms of the temperature profiles in Fig. 6.9. We clearly recognize the progressing solidification. Particular to the adopted model for L is the slight jump in the derivative near the front. In Fig. 6.10 we display the effect of heat release on the location of the mushy zone. We notice that an increased heat release yields a more rapid solidification. This problem was also treated independently with an implicit time-stepping method in combination with an adaptive mesh. This allows to capture the phenomena in more detail at lower computational cost. The final results of the two codes compared very closely, thereby providing an independent check.

6.5 Concluding remarks

In this paper we described the modeling of solidification processes in aluminum casting. We emphasized the central role that the thermodynamics of solidification has. Particularly at realistic numbers of alloying elements the proper description of the thermodynamic components is a strong limiting factor. The obvious brute force approach based on minimization of the Gibbs free energy does not provide a realistic option. Rather, database approaches, not unlike those used in combustion research, need to be developed to bring the computational effort down to a more manageable level. It was argued that simply using a pre-computed database to represent the thermodynamics is insufficient and further data-reduction is mandatory. In Section 3 a simple approach based on piecewise polynomial fitting was described and shown to bring the data-handling down to a realistic level. However, the method cannot be easily extended to spatially dependent situations. For that purpose more

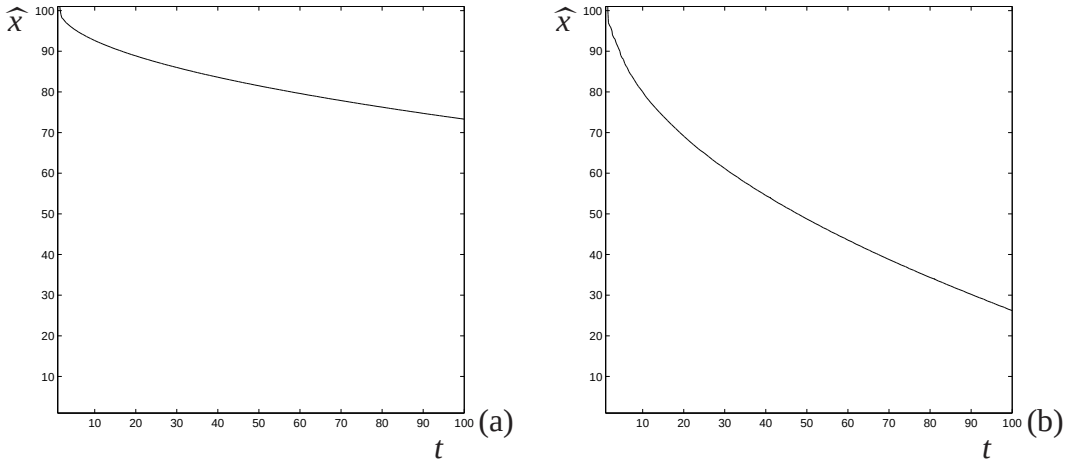


Figure 6.10: Effect of heat release on the solidification front defined at $\widehat{x}$ where $T(\widehat{x}, t) = T_m$. In (a) we use $\beta = 1$ and in (b) the value $\beta = 100$ is adopted.

involved data representations and methods for efficient processing were suggested as well. The confrontation of these methods with realistic solidification simulations as are adopted in industry is still an open challenge. Based on the experience with the simplified approach, savings on the order of 100 or more appear possible without affecting the accuracy of predictions too much. While developing the improved data-base handling for solidification processes, use could be made of the simplified one-dimensional simulation model that appears to capture the main physics of a progressing solidification front at modest computational costs. This could be a helpful testing ground for the incorporation of several of the proposed data-reduction techniques and measures to speed-up the computations. Research in that direction is much needed and constitutes a challenge for the future.

Bibliography

[1] Nadella, R. D.G. Eskin, Q. Du, L. Katgerman: 2008. Macrosegregation in direct-chill casting of aluminum alloys. *Prog. Mat. Sci.*, **53** 421–480

[2] Mortensen, D.: 1999. A Mathematical Model of the Heat and Fluid Flows in Direct-Chill Casting of aluminum Sheet Ingots and Billets. *Metallurgical and Materials Transactions A*, **30B** 119–133

[3] Fjaer, H.G., and A. Mo: 1990. ALSPEN – A Mathematical Model for Thermal Stresses in Direct Chill Casting of aluminum Billets. *Metallurgical and Materials Transactions A*, **21B** 1049–1061

[4] Pequet, C., M. Gremaud, M. Rappaz: 2002. Modeling of microporosity, macroporosity, and pipe-shrinkage formation during the solidification of al-

loys using a mushy-zone refinement method: Applications to aluminum alloys. *Metallurgical and Materials Transactions A*, **33** 2095

[5] Rerko, R.S., H.c. de Groh, III, C. Beckermann:2003. Effect of melt convection and solid transport on macrosegregation and grain structure in equiaxed Al-Cu alloys. *Mat.Sci. Eng.* **A347** 186–197

[6] Ahmad, N., H. Combeau, J-L. Desbiolles, T. Jalanti, G. Lesoult, J. Rappaz, M. Rappaz, and C. Stomp: 1998. Numerical Simulation of Macrosegregation: a Comparison between Finite Volume Method and Finite Element Method Predictions and a Confrontation with Experiments.*Metallurgical and Materials Transactions A*, **29A** 617–630

[7] Du, Q., D.G. Eskin and L. Katgerman: 2007. Modeling Macosegregation during Direct-Chill Casting of Multicomponent Aluminum Alloys. *Metallurgical and materials transactions A*, **38A** 180–189

[8] Thermocalc Thermodynamic Database Software, `www.thermocalc.com`.

[9] Factsage and Chemapp Integrated Thermodynamic Database System, `www.factsage.com`.

[10] JmatPro, Thermo-physical and thermodynamic properties software, `www.thermotech.co.uk`.

[11] Bear, J.: 1972. *Dynamics of Fluids in Porous Media.* Dover Publications Inc., New York

[12] Schneider, M.C., Beckermann, C.: 1995. Formation of Macrosegregation by Multicomponent Thermosolutal Convection during the Solidification of Steel, *Metallurgical and Materials Transactions A*, **26A** 1995–2373

[13] Goyeau, B. Bousquet-Melou, P., Gobin, D., Quintard, M., Fichot, F.: 2004. Macroscopic modeling of columnar dendritic solidification, *Computational and Applied Mathematics* **23: 2-3** 381-400

[14] Barber, C.B., Dobkin, D.P., Huhdanpaa, H.: 1996. The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software* **22** 469–483

[15] Berg, M. de, Kreveld, M. van, Overmars, M., Schwarzkopf, O.: 2000. *Computational Geometry. Algorithms and Applications.* Springer-Verlag

[16] Bungartz, H-J., Griebel, M.: 2004. Sparse grids. *Acta Numerica* **13** 147–269

[17] DeHoff, R.: 2006. *Thermodynamics in Materials Science.* CRC Press

[18] Doré, X., Combeau, H., Rappaz, M.: 2000. Modelling of Microsegregation in Ternary Alloys: Application to the Solidification of Al-Mg-Si. *Acta materialia* **48** 3951–3962

[19] Eriksson, G., Hack, K.: 1990. ChemSage — A Computer Program for the Calculation of Complex Chemical Equilibria. *Metallurgical Transactions B* **21** 1013–1023

[20] Eriksson, G., Spencer, P.J., Sippola, H.: 1995. A General Thermodynamic Software Interface. In: A. Jokilaakso (ed.), Proceedings of the 2nd Colloquium on Process Simulation, pg. 113, Espoo, Finland, Helsinki University of Technology, Report TKK-V-B104

[21] Klimke, A., Wohlmuth, B.: 2005. Algorithm 847: spinterp: Piecewise Multilinear Hierarchical Sparse Grid Interpolation in MATLAB. *ACM Transactions on Mathematical Software* **31** 561–579
URL: `http://www.ians.uni-stuttgart.de/spinterp/index.html`

[22] Persson, P.-O., Strang, G.: 2004. A Simple Mesh Generator in MATLAB. *SIAM Review* **46** 329–345

[23] Persson, P.-O.: 2005. Mesh Generation for Implicit Geometries. PhD Thesis, Department of Mathematics, MIT

[24] Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P.: 2002. *Numerical Recipies in C++*. Cambridge University Press

[25] Zhao, H.-K, Osher, S., Fedkiw, R.: 2001. Fast Surface Reconstruction Using the Level Set Method, In: VLSM Proceedings of the IEEE Workshop on Variational and Level Set Methods (VLSM'01), pg. 194