Proceedings of the 148^{th} European Study Group Mathematics with Industry

Wageningen, The Netherlands, January 28 - February 1, 2019

Editors:
Jaap Molenaar
Hans Stigter

ISBN: 978-94-6395-192-0
DOI: 10.18174/505266

The Study Group Mathematics with Industry 2019 was financially supported by:

Preface

In 2019, the "Studiegroep Wiskunde met de Industrie (SWI 2019)" took place in Wageningen, The Netherlands, in the week of 28 January – 1 February, organized by a team from Wageningen University and Research (WUR). When organizing an SWI, one stands in a long tradition that started at Oxford University in 1968 and has been copied all over the world. For an overview of this tradition we refer to the website `http://miis.maths.ox.ac.uk`. In Europe the numbering is strictly kept and the present SWI 2019 was also the 148th "European Study Group with Industry (ESGI)". Since 1998, nearly every year an SWI has been organized in The Netherlands, by different teams in different places.

During an SWI, a group of 50 to 80 mathematicians from anywhere in the Netherlands and abroad come together to tackle industrial problems. Six companies present their problems on Monday. This time the problems came from Bruil, Hendrix Genetics, Synopsys, KWR Water, Sweco, and Marin. The group members then devote the entire week aiming for solutions of the problems. By Friday the groups present their solutions. The main purpose of such an SWI is to show that mathematical modeling and mathematical techniques are powerful tools to solve real life problems. The motivation of most participants is also greatly driven by the mere pleasure to apply mathematics to open problems that still need a lot of modeling and to do this in a team. The enthusiasm and commitment to get optimal results given the limitations of such an event is overwhelming and also this time the industrial partners were astonished by the results obtained in a short period of intense, cooperative research.

In these proceedings the scientific reports of the six groups of SWI 2019 are presented. These scientific proceedings are accompanied by a separate issue in Dutch, written by Anouck Vrouwe, containing descriptions of the projects and their results in a form that is readable for a broad, non-scientific audience.

It is a pleasure to thank the sponsors of SWI 2019. Since the very beginning of the Dutch series of Study Groups, NWO has been sponsor, and this financial contribution is much appreciated. We also thank 4TU.AMI and Wageningen UR in this respect.

The organizing team of SWI 2009,

Guus ten Broeke, Jos Hageman, Lia Hemerik, Karel Keesman, Jaap Molenaar,
Hans Stigter, Dinie Verbeek, George van Voorn, Gerard van Willigenburg.

Contents

Chapter 1

Prediction of print success for concrete 3D printing

Pieter Collins[1], Simcha van Helvoort[2], Giorgi Khimshiasvili[3], Antonio Marsella[1], Jaap Molenaar[4], Lense Swaenen[5]

[1]Maastricht University, The Netherlands
[2]Fontys Hogeschool, The Netherlands
[3]Ilia State University, Georgia
[4]Wageningen University, The Netherlands
[5]Sioux Lime, The Netherlands

Abstract:
Bruil beton & mix is specialized in the production of concrete for all kinds of applications. In the past years, Bruil has developed a new, exciting technique that has revolutionized this sector: production of prefab elements using concrete 3D printing. This development offers architects a completely new scala of design possibilities in form, colour, and structure. 3D printing starts with a digital 3d model of the object. From this 3D model a print path is created, based on the required layer width and height. Code based on the created print path, will steer the movements of the concrete printer. Using these movements to deposit concrete at the right locations, the 3D printing process thus results in a printed concrete object. There are several difficulties with 3d printing of concrete. One of them is that concrete is a material of which the properties change during the drying process. Furthermore, these properties quite strongly depend on external conditions. Therefore, in practice there are many variables that influence the printing result and thus determine whether a print will be successful: material properties, environmental conditions, rheology, the shape of the growing object, print speed etc. Second, not all structures dreamt up on the drawing table can be easily produced with 3D printing. During the printing process the emerging structure could start to bend, collapse and/or deform, depending on geometry and material properties. So, the drying process and the printing process should be fine-tuned, to allow for efficient, fast, and reliable production of 3D objects.
In this report we first present some checks on printability that are easily implemented in practice. Then, we dealt with the question whether the stress profile during the printing process fulfils the condition that the developing structure does not collapse under its own weight. To answer that question, we study a variety of different approaches to calculate the stress profile. These approaches range from several approximating analytical methods to numerical simulations. The conclusion is that for simple geometries, such as a tilted wall, analytical, explicit formulae can be used to check stress conditions, but that for general geometries a numerical approach, based on, e.g., a finite element method, is indispensable.

KEYWORDS: *3D-printing, concrete, stress conditions, perturbation method, finite element method*

1.1 Introduction

3D printing of concrete is a recent breakthrough in the construction world. In the traditional approach of building with concrete, the fresh material is poured in a form work, that has been designed according to the required final shape. After a drying period in which the concrete hardens, the form work is taken away and a solid structure results. In contrast, 3D printing is a completely different concept. The fresh material is extruded via a nozzle in a continuous way. The extruded material forms a flexible thread of concrete that is laid upon or against earlier layers. By steering the spray head along a well chosen, predefined path, all kinds of shapes can be formed. The 3D printing technique has revolutionized this sector, since it allows for many new applications of concrete. It offers architects a completely new scala of design possibilities in form, colour, and structure. See for example figure section 1.1.

Concrete companies like Bruil apply 3D printing already in practice, but are faced with several challenges. One of the possible problems that may arise stems from the fact that concrete changes its properties while hardening. For example, a layer that is being extruded has other properties than the layer on top of which it is deposited. If the differences are too large, it may happen that the neighboring layers do not neatly stick to each other, with as result that the structure of the final object is not completely uniform. Another complication may be that the bottom layers are not hardening fast enough to bear the layers above them. In that case the bottom layers will start to flow side wards and the whole construction will collapse. Still another source of problems may be that, while under construction, the centre of gravity of the construction shifts every time a new layer is added. This may cause the object to topple.

All these aspects force the constructor to carefully design the printing process: the printing process should be fine-tuned such that it allows for efficient, fast, and still reliable production. In this report we deal with a number of sensitive aspects of the printing process. In section 2 we discuss the minimum requirements to be satisfied while printing. The speed of the printing head when hovering over the object under construction and the rate of deposition should be chosen not too low on the one hand, since otherwise the concrete is already hardening too much in the printing head, but on the other hand not too high since then the preceding layers are not yet hardened enough. In building up a tilted object layer by layer, its centre of mass should remain positioned such that the object will not topple. In section 3 we pay attention to structural analysis checks: does the hardening process remain under control so that the construction does not collapse under its own weight. We analyse the stress equations in several ways. First, they are implemented in a Finite Element Package and evaluated on the computer. The

advantage of this approach is that this allows the analysis of all kinds of shapes. On the other hand, computer simulations may be costly and do not always converge. Second, in a complementary endeavour, we follow a number of analytical approaches to solve the stress model equations for special shapes in an approximating way. The results give rise to new insights, that would not have been obtained if we had stuck to computer simulations only. This is especially advantageous for the printing daily practice, since in the procedure of designing the printing process one is often particularly interested in rules of thumb.

We conclude this report with formulating a set of recommendations. These are partly very practically oriented, but we also advocate to involve computer simulations of the stress and strain profiles of the material into the preparation of the printing process, since only such an analysis can yield definite answers if non standard shapes have to be printed.

Figure 1.1: Example of 3D printed concrete object (copyright Bruil).

1.2 Direct printability checks

In this section we formulate some printability checks that can easily be performed once the path of the printing head and the concrete flow rates have been chosen for a specific geometry.

1.2.1 Flows and rates

Fundamental printing path parameters, to be set by the user:

- ρ: Flow rate,
- h_l: Layer height,
- w_l: Layer width,
- l_l: Layer path length.

In addition, we have

- l_l: Layer path length,

which follows from the geometry under consideration.
We consider 4 checks:

1. The flow rate should be such high that the concrete does not dry while still in the hose. This can be simply expressed as

$$\rho \geq \rho_{\min}. \tag{1.1}$$

2. The hose has a maximum flow capacity. This can be simply expressed as

$$\rho \leq \rho_{\max}. \tag{1.2}$$

3. The concrete should form a homogeneous mass and should not exhibit stratification (layering) which compromises structural integrity. The most natural expression of this constraint is that the time between layers should be below a certain value $t_{\max}$, typically 2 minutes. During the printing, it is not possible to wait between layers, as this produces geometrical artifacts not acceptable for design purposes. The time per layer is therefore

$$\text{time per layer} = \frac{\text{volume per layer}}{\text{flow rate}}. \tag{1.3}$$

This leads to the condition

$$\frac{h_l w_l l_l}{\rho} \leq t_{\max}. \tag{1.4}$$

4. The concrete should be solid enough to support the layers on top of it. This questions can be considered very generally as a structural analysis problem, and is treated as such in section 1.3. However, for simple geometries like a straight column, a check can be formulated in terms of the 4 fundamental user settings used above; see Perrot [2]. He proposes as criterion that the height rate at which the structure grows, should be below a certain value H, typically 1.5 meters per hour. The height rate can be related to the layer height and time per layer. This condition can be written as:
$$\frac{\rho}{w_l l_l} \leq H, \tag{1.5}$$
but we emphasize that this criterion will certainly not hold in general.

1.2.2 Centre of mass

Since the printed structure is not attached to the floor in any way, the centre of mass must lie above the convex hull of the base. The centre of mass can be computed cheaply, for every desired time (e.g., per layer) as shown below.

Assume a possibly time-dependent flow rate $\rho(t)$, with the concrete being deposited at a point $(x(t), y(t), z(t))$. Then the total volume $V(t)$ is given by

$$V(t) = \int_0^t \rho(\tau) d\tau \,, \tag{1.6}$$

and the x- and y-coordinates of the centre of mass are

$$\bar{x}(t) = \frac{1}{V(t)} \int_0^t \rho(\tau) x(\tau) d\tau \,, \quad \bar{y}(t) = \frac{1}{V(t)} \int_0^t \rho(\tau) y(\tau) d\tau \,. \tag{1.7}$$

For a small time increment δt,

$$V(t + \delta t) \approx V(t) + \rho(t)\delta t$$

and

$$\bar{x}(t + \delta t) = \frac{1}{V(t + \delta t)} \int_0^{t+\delta t} \rho(\tau) x(\tau) d\tau \quad \approx \frac{1}{V(t) + \rho(t)\delta t} \big(V(t)\bar{x}(t) + \rho(t)x(t)\delta t \big).$$

Taking measuring time points $t_0, t_1, \ldots$, and writing $\delta t_n = t_{n+1} - t_n$ we may thus simply follow the centre of mass coordinates in time by evaluating the update formulae

$$\begin{gathered} V(t_{n+1}) \approx V(t_n) + \rho(t_n)\delta t_n \,, \\ \bar{x}(t_{n+1}) \approx \frac{V(t_n)\bar{x}(t_n) + \rho(t_n)x(t_n)\delta t_n}{V(t_{n+1})} \,, \quad \bar{y}(t_{n+1}) \approx \frac{V(t_n)\bar{y}(t_n) + \rho(t_n)y(t_n)\delta t_n}{V(t_{n+1})} \,. \end{gathered} \tag{1.8}$$

It would also be possible to obtain more accurate estimates using higher-order integration methods such as Simpson's rule, but taking δt small is probably enough to obtain good accuracy.

As mentioned in the beginning of this section, the center-of-mass should be in the convex hull of the support not to topple. However, when on the boundary of the convex hull, one can expect all of the weight of the structure to be concentrated on a tiny portion of the support, which makes a structural failure more likely. One could shrink the support a certain amount to improve stability. A relationship between the height/mass of the structure and how much to shrink the support can probably be derived.

The center-of-mass makes for a nice bridge to more fundamental structural analysis, as center-of-mass is already a matter of balances of forces. We finally remark that one could use the lateral force required to topple the structure and calculated in a stress analysis, as another way to get some margin on the center-of-mass check.

1.3 Stress checks

In this section we discuss the printability restrictions that follow from the stresses in the drying concrete. To calculate the stresses at equilibrium we have to solve the so-called stress equations for the stress tensor $\boldsymbol{\sigma}$ with components:

$$\boldsymbol{\sigma} = \begin{pmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{xz} \\ \sigma_{xy} & \sigma_{yy} & \sigma_{yz} \\ \sigma_{xz} & \sigma_{yz} & \sigma_{zz} \end{pmatrix} . \tag{1.9}$$

At equilibrium, the stress satisfies

$$\boldsymbol{\nabla} \cdot \boldsymbol{\sigma} = -\boldsymbol{f} \,, \tag{1.10}$$

where $\boldsymbol{f}$ is the force density in the body. Under gravity, $\boldsymbol{f} = \rho\, g\, \boldsymbol{e}_z$. The acceleration of gravity g has the value $g \approx 9.81 m\, s^{-2}$. The boundary conditions for a free surface with normal $\boldsymbol{n}$ are $\boldsymbol{n} \cdot \boldsymbol{\sigma} = \boldsymbol{0}$.
The units of stress are $N m^{-2} \equiv kg\, m^{-1} s^{-2}$.

The most important insight to be taken into account is that drying concrete has a yield stress that depends on the drying history of the material and thus on time. Note that in a pile of several concrete layers, each of which deposited at a different time, the

yield stress varies per layer and in time. As long as the local stress remains below the local yield stress, the concrete will stay at rest, not flow and thus not deform. When one concrete layer has been deposited, the material will have a certain stress distribution. In general the stress will be highest at the bottom and lowest (maybe vanishing) at the top of the layer. In the next round of printing a new layer will be deposited on the first one. This will change the stress distribution in the first layer. At that moment the stress distributions in the first and the new layer have to be calculated and it should be checked whether the maximum stress remains below the yield stress everywhere. If a third layer is deposited, the stress check has to be repeated but now for three layers. And so on for an arbitrary number of layers. If at some moment in time at some point in the material the stress condition is not fulfilled, the printing procedure will not lead to a robust build up of the desired geometry. These stress checks can be applied before printing starts, through simulation of the stress distributions in space and time on the computer, together with keeping track of the time and position dependent yield stress distribution. In the subsections below we show how the required stress calculations can be done for a variety of geometries. We first show results from numerical (computer) simulations. Thereafter we also present results from analytical approaches.

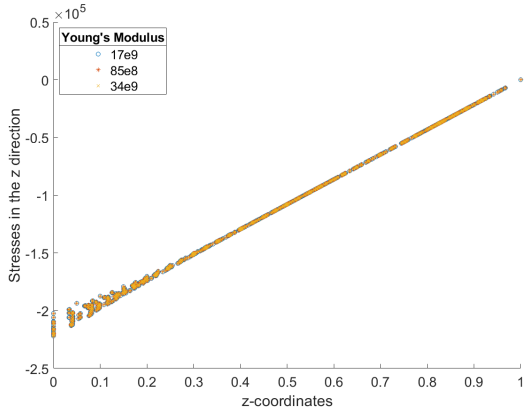
1.3.1 Numerical approach

The holy grail for this problem would be a simulation that takes the 3D printing model as input and a True value as output if the structure is printable and a False value as output if the structure is not printable. This section is proposing a way to simulate the printing part with the 3D printing model as input.
For large scale production, with structures that are not the same, every structure needs to undergo these checks. Importantly, a calculation time longer than ten minutes is not desirable.
For the FEM simulations, we used the MATLAB PDE toolbox.

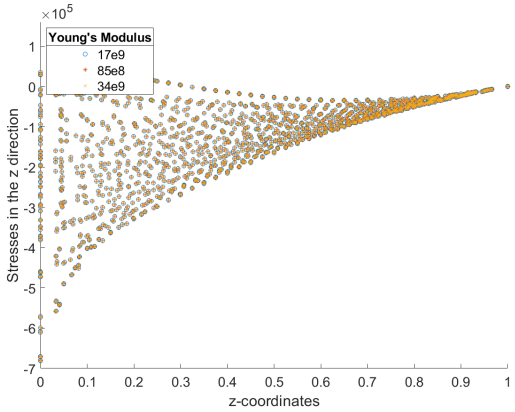
Modeling approach

When concrete is stiffening, a change in the elasticity modulus is expected. The concrete comes out of the nozzle already stiffened up, but still wet enough to merge with the neighbouring layers as described in section 1.2.1.

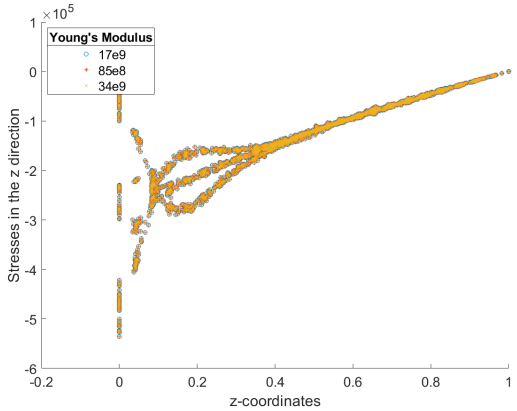
We do not know what the elasticity modulus is over time, so a couple of checks have been made. For three types of structures the simulations have been run with different Young's Modulus. As seen in fig. 1.2, the Young's modulus has no effect for these struc-



(a) Z-Stresses for a straight beam.



(b) Z-Stresses for a tilted beam.



(c) Z-Stresses for a hollow cone.

Figure 1.2: Comparison of the Z-stresses for different Young's Moduli for various structures.

tures. Whether the Young's modulus is down or up a factor of two, the results are the same.

This makes the problem time invariant and only height variant. To simulate layering, we varied the z-coordinates before meshing. This required making a new mesh every iteration and this is very computationally expensive since calculation time per iterations will increase exponentially with mesh size. A possible solution will be proposed in section 1.4, but for the present simulations we didn't apply this idea yet.

The boundary condition applied at the bottom of the object deserves special attention. The concrete object does not stick to the table. A realistic boundary condition for the contact between the table and the concrete would be hard contact, which allows pushing (positive normal stress) but not pulling (negative normal stress). Such a boundary condition is very challenging computationally, as the problem is no longer linear.

Moreover, the MATLAB PDEtoolbox does not have this feature. More specialized FEM packages like ANSYS allow for such boundary conditions by modelling both the support and the contact. If such a boundary condition is used, toppling of a structure (like in the center-of-mass checks) can also be derived from the structural analysis. In the ANSYS best practices manual, this is referred to as 'lift-off'. When the center-of-mass is outside of its support, there is no feasible distribution of stresses that does not have a normal stress at the table/concrete interface.

Another aspect is the shear stress at the bottom. Since there is friction between table and object, few shear stress will not cause the object to shift. Much shear stress, however, will lead to a shifting object. It is not clear what the critical shear stress is for drying concrete, so we could not specify it. In our calculations we assumed the object to be rigidly fixed to the table:

$$u(x, y, 0) = 0\,, \tag{1.11}$$

where $u(x, y, z)$ is the displacement of the object. This boundary condition was easy to implement within the MATLAB PDE toolbox.

Numerical solution via the finite element method(FEM)

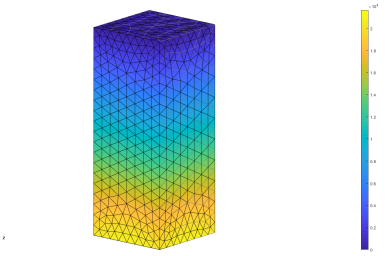
The results of the FEM simulation are given in fig. 1.3, where the final, steady state stress distributions are shown. The computational times are given in table 1.1. These results can be checked via comparison with results from analytical approaches in the subsections below.

Structure	Computing time
Straight Beam	31 seconds
Tilted Beam	32 seconds
Hollow Cone	No layering simulation has been done

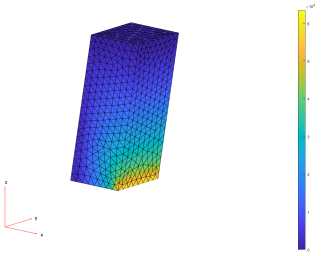
Table 1.1: Computing time for different structures. The computing time is the time it took to simulate the layering using MATLAB.

1.3.2 Analytical approaches

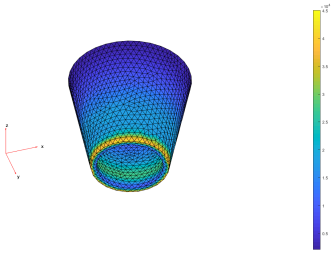
As solving stresses from a full Finite Element model is computationally rather expensive, in this section we explore analytical approaches to approximate stresses throughout the



(a) FEM result for a straight beam.



(b) FEM result for a tilted beam.



(c) FEM result for a hollow cone.

Figure 1.3: FEM steady state results for different kinds of structures.

geometry. Here, we derive analytical formulae for simple geometries. We emphasize that the present lines of thought are by far not yet complete, in view of the limited time that spanned the SWI 2019, but are meant to inspire further research. The results can be used to check the numerical results, but the insights may also be used to find rules of thumb for general geometries. We start with the simplest geometry, a straight wall, and incrementally move to more general geometries.

Buckling check

Three basic cases of straight wall structures were considered in a recent paper of A. Suiker [5]. We elaborate on the results of this paper concerned with elastic buckling by relating them to some basic results of catastrophe theory and outlining hypothetical analogous results on elastic buckling of wall structures of a slightly more general type described below. To this end we rely on the rigorous analysis of buckling of thin rods (Euler buckling) and thin elastic membranes given in the fundamental monograph of T.Poston and I.Stuart [6].

Consider the model of 3D printing of a rectangular wall used in the paper of A. Suiker. The wall is given by a rectangular, heterogeneous plate of length L, width b and thickness h subjected to in-plane forces acting in the mid-plane of the plate. It is assumed that

this wall is produced by a 3D printing process, with linear curing function, applied in the direction of length L (i.e., L should be considered as the height of wall) which is characterized by the following parameters:

- constant wall growth velocity in vertical direction $l^* = Q/(hv_hT_l)$,
- the initial bending stiffness $D = (Eh^3)/(12(1-\gamma^2))$,
- dimensionless parameter $\mu = \rho g(h/D)(l^*/\phi)$,

where Q is the material volume delivered by the nozzle per unit time, v_h is the horizontal velocity of the nozzle, T_l is the time needed for printing one layer, ρ is the volumetric mass density, $g = 9.81m/s^2$ is the gravitational acceleration, γ is the Poisson modulus of material considered, E is the initial stiffness modulus, ϕ is the curing speed of the linear curing process. More detailed descriptions of the above quantities are given in [5].

As was shown in [5], both elastic buckling and plastic collapse can happen in this process and a criterion of the possible failure mechanism can be expressed in terms of geometrical, material and printing process data. Using the aforementioned data one can algorithmically calculate dimensionless quantities L_c, L_p and Λ and formulate the following criteria:

- elastic buckling happens if $L_c/L_p < \Lambda$,
- plastic collapse happens if $L_c/L_p > \Lambda$.

Here $\Lambda = (h/D)^{1/3}|\sigma|(\rho g)^{-2/3}$, where σ is the yield strength of material. The explicit formulae and computational algorithms for L_c and L_p are given in Eqs. (75), (91), (92), (94) of [5] .

Our first observation is that these criteria agree with the classical results on buckling of thin rods and elastic membranes given in Chapter 13 of the monograph [6]. To this end notice that, for small values of thickness h, the plate (wall) under consideration can be approximated by an elastic rectangular membrane of length L and width b so that membrane can be considered as a limit as $h \to 0$. Analogously, if both h and b tend to zero in a commensurable way, then the limiting object can be identified with an elastic thin rod of length (height) L. Then it is easy to verify that the above criteria agree with the classical criterion of Euler buckling described in [6].

Our second observation is aimed at obtaining similar results for vertical walls over more complicated horizontal bases. To this end consider a vertical wall $W(X, L, h)$ of height L and thickness h obtained as a tube of radius $h/2$ around a vertical plate of height L over a circular arc X of length b and curvature K in the horizontal plane. Such an object gives a natural generalization of the rectangular wall discussed above.
For such a circular wall, repeating the analysis given in [5] is easy. It yields that the condition for plastic collapse remains the same as for the rectangular wall considered above. The situation with elastic collapse is more interesting. For some models of 3D printing process, there exists numerical evidence that the curvature influences the critical height for elastic buckling by making it bigger than in the flat case. We are unable to give a rigorous proof of this fact and to give a hypothetical formula for the increase of critical height in terms of the curvature K. Elaborating on the influence of curvature of wall profile is an interesting mathematical challenge and, moreover, may appear useful for analysis of mechanical performance of 3D printing processes for walls over more general planar profiles.

Analytical approach: tilted wall with linear vertical normal stress variation

We consider a tilted wall of width w, angle θ with the vertical, and total height h, and try to approximate the stresses at any intermediate height. We make the following two assumptions:

1. Only σ_{zz} is non-zero. All other stresses are zero.

2. σ_{zz} varies linearly across the width at a certain height: $\sigma_{zz}(x, z) = a(z)x + b(z)$

Note that both assumptions hold for the non-tilted wall. A rationale for the second approximation comes from the fact that the thickness of the printed wall will be small compared to the other dimensions (length and height of the wall). Inspiration was drawn from somehttps://www.overleaf.com/project/5c63d9969a586b5c02a038ad figures in [1].

To derive approximate values for the vertical normal stress throughout the tilted wall, we consider a cross-section at a certain height, as in figure 1.4. h is the height of the part of the wall above this cross-section. According to the assumptions, the vertical normal stress varies linearly. The entire profile across the cross-section is therefore described completely by the two variables f_L and f_R. We now require that the block of material above the intersection should be in static equilibrium: both the balance of forces and the balance of moments should be zero. This leads to two equations that are linear in f_L and f_R. This system with 2 unknowns and 2 equations can be easily solved for f_L and f_R.

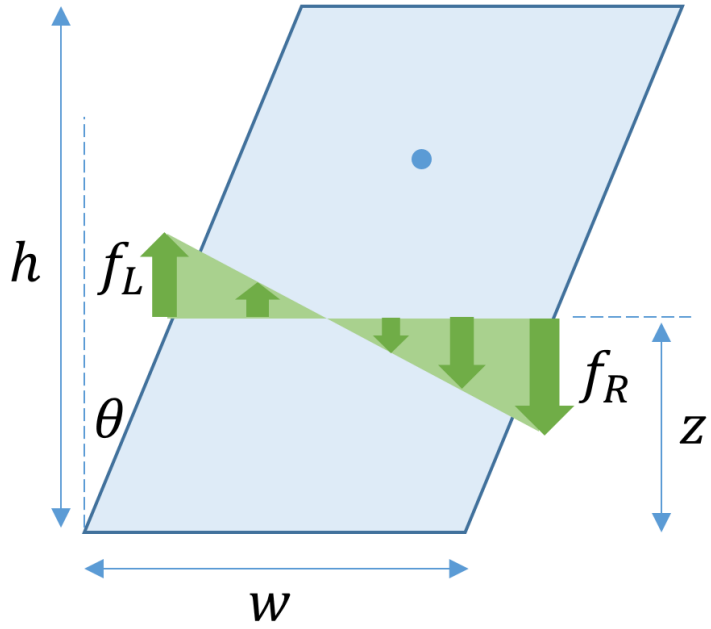


Figure 1.4: Approximate model diagram for the tilted wall.

The balance of forces reads

$$\int_{x_L}^{x_R} f(x)\,dx = mg\,, \tag{1.12}$$

with $x_R - x_L = w$. The forces at the interface need to counteract the gravity on the mass above the interface. Note that this mass is z-dependent: $m = \rho w(h - z)$. $f(x) \equiv \sigma_{zz}$ is the force at any point across the thickness:

$$f(x) = f_L + \frac{f_R - f_L}{w}(x - x_L). \tag{1.13}$$

The balance of forces can be formulated about any point. We choose x_L:

$$\int_{x_L}^{x_R} f(x)x\,dx = mg \cdot x_{CoM}, \tag{1.14}$$

with x_{CoM} the x-coordinate of the center-of-mass of the upper section. This leads to

$$x_{CoM} = \frac{w}{2} + (h - z) \cdot \tan(\theta). \tag{1.15}$$

Rewriting the integrals in equations (1.12) and (1.14) in terms of f_L and f_R yields the linear 2×2 system:

$$f_L + f_R = 2\frac{mg}{w} = 2\rho g(h - z), \tag{1.16}$$

$$f_L + 2f_R = 6\,\frac{mg \cdot x_{CoM}}{w^2}. \tag{1.17}$$

Solving for f_L and f_R yields:

$$f_L = \rho g(h-z)\left(1 - 3\frac{h-z}{w}\cdot\tan(\theta),\right) \tag{1.18}$$

$$f_R = \rho g(h-z)\left(1 + 3\frac{h-z}{w}\cdot\tan(\theta).\right) \tag{1.19}$$

These equations correctly reduce to the straight wall conditions for which $\theta = 0$. With $\theta > 0$, f_R now increases non-linearly with $h - z$, which has been also observed in the numerical FEM results. While f_R will be strictly positive for $\theta > 0$, we see how f_L can switch sign, indicating tensile stresses (inward/upward normal stress) in the material. Due to the tilt, one side of the wall experiences a downward 'pushing' force, whereas the other side experiences an upward 'pulling' force to compensate. Moreover, we observe that the non-linear term also has a dependency of $\frac{1}{w}$, which can also be understood intuitively: thin walls have a stronger push-pull action.

We compare the analytical results from this approximate model with the numerical results obtained earlier in figure 1.5. The stresses of the FEM mesh are plotted as a scatter plot. The stresses at the left and right faces are extreme (for a given height), so the modeled f_R and f_L should match the upper and lower envelopes. We observe the correct qualitative results, but quantitatively there is a definite mismatch. Through experimentation, we have observed that

$$f_{L,corr} = f_L - \rho g\frac{(h-z)^2}{w}\tan(\theta) = \rho g(h-z)\left(1 - 4\frac{h-z}{w}\cdot\tan(\theta)\right), \tag{1.20}$$

$$f_{R,corr} = f_R - \rho g\frac{(h-z)^2}{w}\tan(\theta) = \rho g(h-z)\left(1 + 2\frac{h-z}{w}\cdot\tan(\theta)\right). \tag{1.21}$$

match the numerical FEM results (for the upper section of the wall) almost perfectly. This correction can be seen as changing the multiplier of the non-linear terms in both equations; this correction is taken the same for both stresses. This means that in terms of the 2×2 balance of forces and moments equations, both the weight carried and the moments would need to be modified. It is not yet fully understood how and why this correction works. A possible explanation could be the model assumption that all stresses except σ_{zz} are 0. The FEM results show that this is not exactly true. If we compare the von Mises stress, which are given by

$$\sigma_{vm} = \frac{1}{\sqrt{2}}\sqrt{(\sigma_{xx}-\sigma_{yy})^2 + (\sigma_{yy}-\sigma_{zz})^2 + (\sigma_{zz}-\sigma_{xx})^2 + 6\sigma_{xy}^2 + 6\sigma_{yz}^2 + 6\sigma_{zx}^2}, \tag{1.22}$$

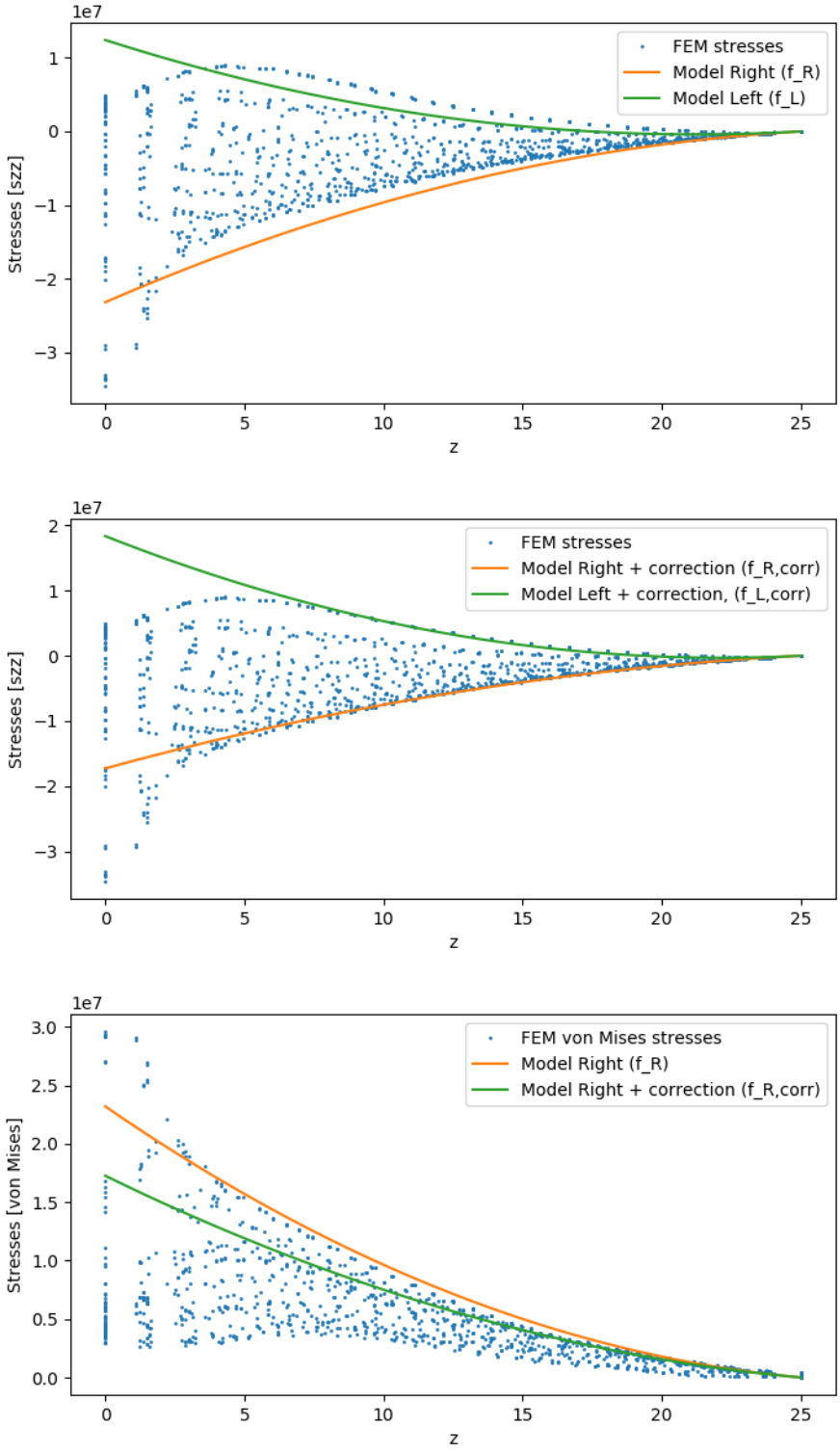


Figure 1.5: Comparison of FEM model with approximate model. Top: Comparing σ_{zz}. Middle: With correction added to the approximate model. Bottom: Comparing von Mises stresses.

then the original model already performs better.
Finally, we remark that the predicted quadratic behavior with $(h-z)^2$ is a good approximation in the upper section of the geometry, but near the bottom this description is clearly poor. Unfortunately, the bottom part is just the section we are most interested in, as stresses are highest and failure is most likely in that region. We observed earlier that the stresses at the bottom are quite sensitive to the boundary conditions chosen there. The present result again emphasizes that this boundary condition needs extra investigation.

Disregarding the accuracy of the approximation, this model has enabled us to derive local criteria on stability which look like $f_R(h-z,\theta,w) < \sigma_{max}$ which captures physics more realistically than the currently used $\theta < \theta_{max}$ criterion.

1.3.3 Analytical approach: separation of variables

We now consider an alternative approach to solve the stress equations for a number of geometries.

Vertical wall

Consider a vertical a wall of width w and height h. Omit the y-coordinate. Then the equilibrium equations become

$$\begin{aligned} \partial_x \sigma_{xx} + \partial_z \sigma_{xz} &= 0, \\ \partial_x \sigma_{xz} + \partial_z \sigma_{zz} &= -\rho g. \end{aligned} \tag{1.23}$$

The boundary conditions at the sides are

$$\sigma_{xx} = \sigma_{xz} = 0 \text{ when } x = 0 \text{ and } x = w. \tag{1.24}$$

The boundary conditions at the top are

$$\sigma_{xz} = \sigma_{zz} = 0 \text{ when } z = h. \tag{1.25}$$

An obvious solution of these equations is

$$\sigma_{xx} = \sigma_{xz} = 0; \quad \sigma_{zz}(x,z) = \rho g(h-z).$$

However, there are other solutions. Since the stress equations and boundary conditions are linear, any solution is the sum of a solution of the homogeneous equations (with r.h.s.

equal to zero) and a particular solution of the full, nonlinear equations. We show that the solutions of the homogeneous equations are not unique:

$$\begin{aligned} \partial_x \sigma^h_{xx} + \partial_z \sigma^h_{xz} &= 0, \\ \partial_x \sigma^h_{xz} + \partial_z \sigma^h_{zz} &= 0. \end{aligned} \tag{1.26}$$

The homogeneous equations have d a class of solutions of the following, separable form:

$$\sigma^h_{xx} = f_x(x) f''_z(z), \quad \sigma^h_{xz} = -f'_x(x) f'_z(z), \quad \sigma^h_{zz} = f''_x(x) f_z(z). \tag{1.27}$$

Suppose f_x satisfies $f_x(0) = f_x(w) = f'_x(0) = f'_x(w)$. Then any f_z satisfies the side boundary conditions. Similarly, if $f_z(h) = f'_z(h) = 0$, then any f_x satisfies the top boundary conditions. We can also consider homogeneous boundary conditions for the lower surface, $\sigma_{xz}(x,0) = \sigma_{zz}(x,0) = 0$.

We therefore have nontrivial solutions of the homogeneous equations, the simplest of which read as

$$f_x(x) = x^2(h-x)^2, \quad f_z(z) = z^2(h-z)^2,$$

yielding

$$\begin{aligned} \sigma^h_{xx} &= x^2(w-x)^2 \times 2(h^2 - 6hz + 6z^2), \\ \sigma^h_{xz} &= -2x(w-x)(w-2x) \times 2z(h-z)(h-2z), \\ \sigma^h_{zz} &= 2(w^2 - 6wx + 6x^2) \times z^2(h-z)^2, \end{aligned}$$

and

$$\sigma_{zz}(x,z) = 2(w^2 - 6wx + 6x^2) \times z^2(h-z)^2 + \rho g(h-z).$$

We can even set the horizontal stress at the bottom surface to zeros, $\sigma_{xx}(x,0) = 0$, requiring $f''_z(0) = 0$, and take $f_z(z) = z^3(h-z)^2$. This makes clear that there is an infinite-dimensional space of solutions to the homogeneous equations.

Remark 1.1. This shows that the equilibrium equations are underdetermined; in principle we need information about the stress-strain relationship to obtain a unique solution for the stress tensor. This likely means that to solve for the stresses, we in principle need to consider the dynamic processes involved as the concrete flows and sets while forming the object. In the following we take the simplest solution of the stress equations as the preferred one.

Tilted wall

In section 1.3.3 we sketch a vertical cross section of a tilted wall, of width w, height h, and inclination angle θ to the vertical. Let $\alpha = \tan\theta$, and $\xi = x - \alpha z$. Omit the y-coordinate.

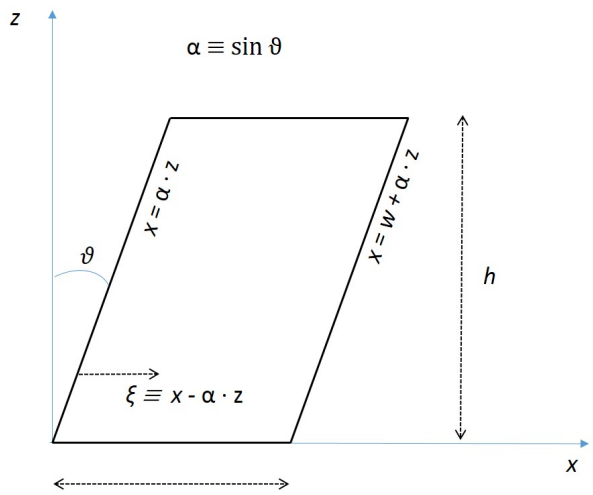


Figure 1.6: Vertical cross section along the x - axis of a tilted wall, of width w, height h, and angle of inclination θ. The wall is tilted in the (x, z) plane, but becomes vertical after application of the coordinate transformation $(x, z) \to (\xi, z)$.

Then the equilibrium equations become

$$\begin{aligned} \partial_x \sigma_{xx} + \partial_z \sigma_{xz} &= 0, \\ \partial_x \sigma_{xz} + \partial_z \sigma_{zz} &= -\rho g. \end{aligned} \tag{1.28}$$

The boundary conditions at the sides are

$$\left.\begin{aligned} \sigma_{xx} - \alpha \sigma_{xz} &= 0 \\ \sigma_{xz} - \alpha \sigma_{zz} &= 0 \end{aligned}\right\} \text{ when } x = \alpha z, \text{and} x = \alpha z + w. \tag{1.29}$$

The boundary conditions at the top are

$$\sigma_{xz} = \sigma_{zz} = 0 \text{ when } z = h. \tag{1.30}$$

For a frictionless supporting bottom surface the boundary condition would read as

$$\sigma_{xz} = 0 \text{ when } z = 0. \tag{1.31}$$

A simple particular solution σ^p of the full equations can be borrowed from the vertical wall, dealt with in the previous section:

$$\sigma^p_{xx} = \sigma^p_{xz} = 0; \quad \sigma^p_{zz}(x, z) = \rho g(h - z). \tag{1.32}$$

The full solution can be written as $\sigma = \sigma^h + \sigma^p$, where σ^p solves the homogeneous interior equations

$$\begin{aligned} \partial_x \sigma^h_{xx} + \partial_z \sigma^h_{xz} &= 0, \\ \partial_x \sigma^h_{xz} + \partial_z \sigma^h_{zz} &= 0. \end{aligned} \tag{1.33}$$

Then the boundary equations for σ^h become

$$\left.\begin{aligned}\sigma^h_{xx} - \alpha\sigma^h_{xz} &= 0\\ \sigma^h_{xz} - \alpha\sigma^h_{zz} &= \alpha\rho g(h-z)\end{aligned}\right\} \text{ when } x = \alpha z, \text{and} x = \alpha z + w \tag{1.34}$$

$$\sigma^h_{xz} = \sigma^h_{zz} = 0 \text{ when } z = h. \tag{1.35}$$

As above, we may try homogeneous solutions in separable form, and find a basis of solutions which satisfy:

$$\sigma^h_{xx} = f_x(x)f''_z(z), \quad \sigma^h_{xz} = -f'_x(x)f'_z(z), \quad \sigma^h_{zz} = f''_x(x)f_z(z). \tag{1.36}$$

We can also attempt to find homogeneous solutions of involving a function $f_\xi(\xi)$. The interior equations yield

$$\begin{aligned}\sigma^h_{xx} &= f_\xi(x-\alpha z)f''_z(z) - 2\alpha f'_\xi(x-\alpha z)f'_z(z) + \alpha^2 f''_\xi(x-\alpha z)f_z(z),\\ \sigma^h_{xz} &= -f'_\xi(x-\alpha z)f'_z(z) + \alpha f''_\xi(x-\alpha z)f_z(z), \quad \sigma^h_{zz} = f''_\xi(x-\alpha z)f_z(z).\end{aligned} \tag{1.37}$$

The side boundary conditions at $\xi_* = 0, w$ then become

$$\begin{aligned}f_\xi(\xi_*)f''_z(z) - 2\alpha f'_\xi(\xi_*)f'_z(z) + \alpha^2 f''_\xi(\xi_*)f_z(z) - \alpha\big(-f'_\xi(\xi_*)f'_z(z) + \alpha f''_\xi(\xi_*)f_z(z)\big) &= 0,\\ -f'_\xi(\xi_*)f'_z(z) + \alpha f''_\xi(\xi_*)f_z(z) - \alpha\big(f''_\xi(\xi_*)f_z(z)\big) &= \alpha\rho g(h-z).\end{aligned}$$

which simplifies to

$$\begin{aligned}f_\xi(0/w)f''_z(z) - \alpha f'_\xi(0/w)f'_z(z) &= 0\\ -f'_\xi(0/w)f'_z(z) &= \alpha\rho g(h-z).\end{aligned} \tag{1.38}$$

The top boundary conditions are then

$$-f'_\xi(\xi)f'_z(h) + \alpha f''_\xi(\xi)f_z(h) = 0, \quad f''_\xi(\xi)f_z(h) = 0. \tag{1.39}$$

Alternatively, we can formulate equations in terms of ξ and z. Taking $\tau_{\xi\xi}(\xi,t) = \sigma_{xx}(x-\alpha z, z) - \alpha\sigma_{xz}(x-\alpha z, z)$, $\tau_{\xi z}(\xi,t) = \sigma_{xz}(x-\alpha z, z) - \alpha\sigma_{zz}(x-\alpha z, z)$, $\tau_{zz}(\xi, z) = \sigma_{zz}(x-\alpha z, z)$ yields

$$\begin{aligned}\partial_\xi\tau_{\xi\xi} + \partial_z\tau_{\xi z} + \alpha\partial_z\tau_{zz} &= 0\\ \partial_\xi\tau_{\xi z} + \partial_z\tau_{zz} &= \alpha\rho g(h-z)\\ \left.\begin{aligned}\tau^h_{\xi\xi} &= 0\\ \tau^h_{\xi z} &= \alpha\rho g(h-z)\end{aligned}\right\} &\text{ when } \xi = 0, w.\end{aligned} \tag{1.40}$$

We then aim to find solutions of the homogeneous equations satisfying the boundary conditions

$$\left.\begin{aligned}\sigma^h_{xx} - \alpha\sigma^h_{xz} &= 0\\ \sigma^h_{xz} - \alpha\sigma^h_{zz} &= \alpha\phi(z)\end{aligned}\right\} \text{ when } x = \alpha z,\ x = \alpha z + w. \tag{1.41}$$

As in the case of the vertical wall, the equilibrium stress equations are underdetermined, and a full solution requires knowledge of the stress-strain relationship.

Decorative wall

Consider a decorative wall where the x-coordinate depends on the height: $\phi(z) \leq x \leq \phi(z) + w$. Let $\xi = x - \phi(z)$. We look for solutions of the homogeneous equations with

$$\sigma_{zz} = f''_\xi(x - \phi(z))f_z(z). \tag{1.42}$$

Then

$$\partial_x \sigma_{xz} - \phi'(z)f'''_\xi(x - \phi(z))f_z(z) + f''_\xi(x - \phi(z))f'_z(z) = 0,$$

so we can take

$$\sigma_{xz} = \phi'(z)f''_\xi(x - \phi(z))f_z(z) - f'_\xi(x - \phi(z))f'_z(z). \tag{1.43}$$

Similarly

$$\begin{aligned}\partial_x \sigma_{xx} - \phi'(z)^2 f'''_\xi(x - \phi(z))f_z(z) + \phi''(z)f''_\xi(x - \phi(z))f_z(z) \\ + 2\phi'(z)f''_\xi(x - \phi(z))f'(z) - f'_\xi(x - \phi(z))f''_z(z) = 0,\end{aligned}$$

so we can take

$$\begin{aligned}\sigma_{xx} = \phi'(z)^2 f''_\xi(x - \phi(z))f_z(z) - \phi''(z)f'_\xi(x - \phi(z))f_z(z) \\ - 2\phi'(z)f'_\xi(x - \phi(z))f'(z) + f_\xi(x - \phi(z))f''_z(z).\end{aligned} \tag{1.44}$$

Substituting $\xi = x - \phi(z)$ yields solutions of the form

$$\begin{aligned}\sigma_{zz} &= f''_\xi(\xi)f_z(z), \\ \sigma_{xz} &= \phi'(z)f''_\xi(\xi)f_z(z) - f'_\xi(\xi)f'_z(z), \\ \sigma_{xx} &= \phi'(z)^2 f''_\xi(\xi)f_z(z) - \phi''(z)f'_\xi(\xi)f_z(z) - 2\phi'(z)f'_\xi(\xi)f'(z) + f_\xi(\xi)f''_z(z).\end{aligned} \tag{1.45}$$

The inward normal to the boundary at the point $(\phi(z), z)$ is $(1, -\phi'(z))$, so the boundary conditions are

$$\begin{aligned}\sigma_{xx}(\phi(z), z) - \phi'(z)\sigma_{xz}(\phi(z), z) = 0, \\ \sigma_{xz}(\phi(z), z) - \phi'(z)\sigma_{zz}(\phi(z), z) = 0.\end{aligned} \tag{1.46}$$

with identical conditions holding at $(\phi(z) + w, z)$.

We look for separable solutions to the homogeneous equations with $f_\xi(\xi) = e^{i\kappa\xi} = \exp(i\kappa\xi)$. Further, we want κ to yield a wavelength dividing w, so $\kappa = \kappa_n = 2\pi n/w$. Then

$$\begin{aligned}f_\xi(\xi + w) &= \exp(i\kappa\xi + i\kappa w) = \exp(i\kappa\xi + 2\pi i n/w \times w) \\ &= \exp(i\kappa\xi + 2\pi i n) = \exp(i\kappa\xi) = f_\xi(\xi).\end{aligned}$$

Solving the homogeneous interior equations yields solutions in the following form:

$$\begin{aligned}
\sigma_{zz} &= -\kappa^2 f_\xi(\xi) f_z(z), \\
\sigma_{xz} &= -\kappa^2 f_\xi(\xi)\phi'(z) f_z(z) - i\kappa f_\xi(\xi) f_z'(z), \\
\sigma_{xx} &= -\kappa^2 f_\xi(\xi)\phi'(z)^2 f_z(z) - i\kappa f_\xi(\xi)\phi''(z) f_z(z) - 2i\kappa f_\xi(\xi)\phi'(z) f'(z) + f_\xi(\xi) f_z''(z).
\end{aligned} \tag{1.47}$$

At the left boundary value, $\xi = \phi(z)$. We aim to solve the boundary condition involving σ_{xx} exactly, yielding the differential equations

$$-\kappa^2\phi'(z)^2 f_z(z) - i\kappa\phi''(z) f_z(z) - 2i\kappa\phi'(z) f'(z) + f_z''(z) - \phi'(z)\big(-\kappa^2\phi'(z) f_z(z) - i\kappa f_z'(z)\big) = 0.$$

This simplifies to

$$-i\kappa\phi''(z) f_z(z) - i\kappa\phi'(z) f'(z) + f_z''(z) = 0.$$

The boundary term involving $\sigma_z z$ becomes

$$-\kappa^2 f_\xi(\xi)\phi'(z) f_z(z) - i\kappa f_\xi(\xi) f_z'(z) - \phi'(z)\big(-\kappa^2 f_\xi(\xi) f_z(z)\big),$$

which simplifies to

$$\sigma_{xz}(\phi(z), z) - \phi'(z)\,\sigma_{zz}(\phi(z), z) = -i\kappa\,\exp(i\kappa\phi(z))\,f_z'(z).$$

Summing over such terms gives

$$\sigma_{xz}(\phi(z), z) - \phi'(z)\,\sigma_{zz}(\phi(z), z) = \sum_{n=0}^{\infty} -ic_n\kappa_n\,\exp(i\kappa_n\phi(z))\,f_{z,n}'(z).$$

Conic wall

A natural next step in moving from simple to more complex geometries would be to generalize the tilted wall to conic walls, as visualized in figure 1.2. While a tilted wall is mainly parameterized by the height h, the width w and the slop θ, the conic wall has an additional radius r. One approach we would like to propose is that criteria such as the approximate ones derived for the tilted wall, like

$$f_{L,R}(h - z, \theta, w) < \sigma_{max} \tag{1.48}$$

should be generalized to criteria dependent on radius r, like

$$f_{L,R}(h - z, \theta, w, r) < \sigma_{max}. \tag{1.49}$$

One property one might expect/desire would be that

$$\lim_{r\to\infty} f_{L,R}(h - z, \theta, w, r) = f_{L,R}(h - z, \theta, w). \tag{1.50}$$

This expresses that the limit of the conic wall with a larger and larger radius is the straight wall. A second property one would like to require is that this limit is approached from below, as for a diverging cone, the curvature is expected to increase the stability of the structure. Mathematically one could formulate this as $f_{L,R}(h-z,\theta,w,r)$ being strictly increasing in r (though a rigorous formulation would require some more attention.

1.3.4 Analytical model: perturbation approach

Here, we show still another approach to obtain analytical approximations for the stresses in a tilted wall, sketched in section 1.3.3. The present method is based on expansion of the stress expressions in the parameter $\alpha = \tan(\theta)$, under the assumption $\alpha << 1$. In the (x,z) plane the stress equations read as

$$\begin{aligned} \partial_x \sigma_{xx} + \partial_z \sigma_{xz} &= 0, \\ \partial_x \sigma_{xz} + \partial_z \sigma_{zz} &= -\rho g. \end{aligned} \tag{1.51}$$

As already argued above, it is advantageous to apply the transformation $(x,z) \rightarrow (\xi,z)$ with ξ defined as $\xi = x - \alpha z$, since then the boundary conditions become simpler. After transformation the stress equations read as

$$\begin{aligned} \partial_\xi \sigma_{\xi\xi} - \alpha \partial_\xi \sigma_{\xi z} + \partial_z \sigma_{\xi z} &= 0, \\ \partial_\xi \sigma_{\xi z} - \alpha \partial_\xi \sigma_{zz} + \partial_z \sigma_{zz} &= -\rho g. \end{aligned} \tag{1.52}$$

The boundary conditions at the sides are

$$\left.\begin{aligned} \sigma_{\xi\xi} - \alpha \sigma_{\xi z} &= 0 \\ \sigma_{\xi z} - \alpha \sigma_{zz} &= 0 \end{aligned}\right\} \text{ when } \xi = 0 \,\text{and}\, \xi = w. \tag{1.53}$$

The boundary conditions at the top are

$$\sigma_{\xi z} = \sigma_{zz} = 0 \text{ when } z = h. \tag{1.54}$$

The boundary conditions at the bottom, where $z = 0$, should be such that the shear stress $\sigma_{\xi z}$ may not exceed the frictional force between object and table. However, this frictional force is proportional to the weight of the object with an unknown constant of proportionality, so we cannot specify this condition in detail.

For the vertical wall, i.e., when $\alpha = 0$, we have $\sigma_{\xi\xi} = \sigma_{\xi z} = 0$ and $\sigma_{zz} = \rho g(h-z)$. With respect to this zero-order solution, We write the first order expansions of the stresses in terms of α as

$$\begin{aligned} \sigma_{\xi,\xi} &= \alpha f_1 \xi, z, \\ \sigma_{\xi,z} &= \alpha f_2(\xi, z), \\ \sigma_{z,z} &= \rho g(h-z) + \alpha f_3(\xi, z). \end{aligned} \tag{1.55}$$

Substituting these expressions in the stress equations, we obtain

$$\begin{aligned} \partial_\xi f_1 + \partial_z f_2 &= 0, \\ \partial_\xi f_2 + \partial_z f_3 &= 0. \end{aligned} \tag{1.56}$$

with boundary conditions at the sides:

$$\left.\begin{aligned} f_1 &= 0 \\ f_2 &= \rho g(h-z) \end{aligned}\right\} \text{ when } \xi = 0, w. \tag{1.57}$$

and boundary conditions at the top:

$$f_2 = f_3 = 0, \text{ when } z = h. \tag{1.58}$$

Further progress requires specification of the stress-strain properties of concrete. But also without these details we may find a good approximation to the stress equations in the following way. We assume that f_2 is homogeneous in z and write $f_2 = \rho g(h-z) f_4(\xi)$ for some function $f_4(\xi)$. In view of the boundary conditions we must require $f_4(0) = f_4(w) = 1$. Substituting this in the stress equations we find

$$\partial_\xi f_1 = \rho g f_4(\xi), \tag{1.59}$$

with boundary conditions

$$f_1 = 0, \text{ when } \xi = 0, \text{and } \xi = w. \tag{1.60}$$

Its solution reads as

$$f_1 = f_1(\xi) = \rho g \int_0^\xi f_4(\xi') d\xi'. \tag{1.61}$$

This expression fits the boundary condition $f_1(\xi = 0) = 0$. To also fit the boundary condition $f_1(\xi = w) = 0$, we take for f_4 the simplest possible form, i.e., the parabola given by

$$f_4(\xi) = 6(\xi - (w/2)^2)/w^2 - 1/2. \tag{1.62}$$

and sketched in figure *section* 1.3.4. From the stress equations above we find that

$$\partial_z f_3 = \partial_\xi f_2 = -\rho g(h-z) f_4(\xi), \tag{1.63}$$

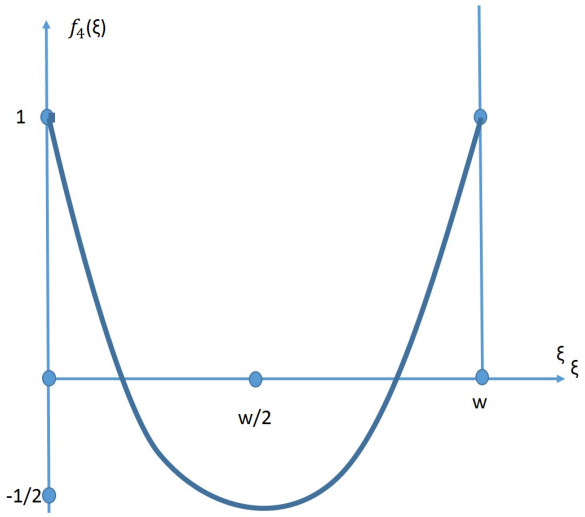


Figure 1.7: Sketch of the function $f_4(\xi) = 6(\xi - (w/2)^2)/w^2 - 1/2$.

with boundary condition

$$f_3(z = h) = 0. \tag{1.64}$$

Its solution reads as

$$f_3(\xi, z) = 6\rho g(z - h)^2(\xi - (w/2))/w^2. \tag{1.65}$$

Finally, we thus obtain for σ_{zz} the approximation

$$\sigma_{zz}(\xi, z) = \rho g(h - z) + \alpha f_3(\xi, z) = \rho g(h - z)[1 + 6\alpha(h - z)(\xi - (w/2))/w^2]. \tag{1.66}$$

This expression provides us with a useful approximation for the stress at the bottom of the tilted wall. Setting $z = 0$, we find that this stress is given by

$$\sigma_{zz}(\xi, 0) = \sigma_{zz}(x, 0) = \rho g h[1 + 6\alpha h(x - (w/2))/w^2]. \tag{1.67}$$

We observe that the extra stress due to the tilt of the wall varies linearly with the horizontal direction x. This was an assumption made in section 3.2.2. The present analysis confirms the correctness of this assumption for relatively small tilt of the wall. The maximum of the extra stress due to tilt is found in the right lower corner with coordinates $(w, 0)$. The vertical stress in that point is given by

$$\sigma_{zz}(w, 0) = \rho g h[1 + 3\alpha h/w]. \tag{1.68}$$

Since this is also the maximum stress in the whole wall, each stress check should focus at this value and make sure that this maximum stress does not exceed the yield stress in that corner point.

1.4 Recommendations

Based on the analysis presented above, we summarize the following recommendations:

- Flow rate checks
  - Concrete should not dry while in hose: Flow rate $\geq F_{min}$.
  - Maximum hose capacity: Flow rate $\leq F_{max}$.
- Stratification checks
  - Time per layer $<$ 2 L / min.
  - Time per layer = Volume per layer / flow rate.
  - Volume per layer = layer height $\cdot$ layer width $\cdot$ path length of layer.
- Center-of-mass check
  - Calculate in advance the position of the centre of mass, using the formulae in *section* 1.2.2 for all times of the printing process and check whether the object under construction runs the risk to topple.
- Stress Checks
  - Every round a new layer has been deposited the stress distribution in the object under construction changes. If locally the internal stress exceeds the local yield stress that part of the object will start to flow or collapse. So, the stress distribution has to be continuously calculated or estimated. With this information it should be checked whether the local stress is everywhere below the local yield stress. Note that the local yield stress depends on the local history thus on the time elapsed since that part of the object was deposited.
  - In the stress calculations the following issues deserve extra attention:
    * The stress equations as such have no unique solution. They should be coupled to the stress-strain relation of concrete.
    * The boundary condition to be applied at the bottom, so where it is in contact with the table, deserves extra attention. There is friction between object and table, but this friction will depend on the local normal stress.
    * To calculate the stress distribution, it is recommended to make use of standard software based on the Finite Element Method (FEM).

* Analytical approaches also provide useful insights that can be used to check numerical outcomes and for deriving rules of thumb. E.g., for a wall tilted to the right the maximum stress will be attained in the right, lower corner and its value is given by equation (1.68)

Acknowledgements

We kindly acknowledge the contributions of Charlotte Philips and Elise Buiter from Bruil company to accomplish this project. They were very helpful in answering all kind of questions during the modelling week and even organized a guided tour for us through the Bruil facilities. They also critically reviewed earlier drafts of this report and came up with useful suggestions.

Bibliography

[1] N. Umetani, and R. Schmidt, Cross-sectional Structural Analysis for 3D Printing Optimization, in: SIGGRAPH Asia 2013 Technical Briefs, pp. 5:1-5:4, ISBN 978-1-4503-2629-2, http://doi.acm.org/10.1145/2542355.2542361, 2013.

[2] A. Perrot, D. Rangeard, A.F. Pierre, Structural built-up of cement-based materials used for 3D-printing extrusion techniques, Materials and Structures, Vol 49, pp 1213 - 1220, DOI 10.1617/s11527-015-0571-0, 2016.

[3] L. Reiter, T. Wanler,N. Roussel,R.J. Flatt, The role of early age structural build-up in digital fabrication with concrete, Cement and Concrete Research, Vol 112, pp 86 - 95, DOI 10.1016/j.cemconres.2018.05.011, 2018.

[4] N. Roussel, Rheological requirements for printables concretes, Cement and Concrete Research, Vol 112, pp 76 - 85, DOI 10.1016/j.cemconres.2018.04.005, 2018.

[5] A.S.J Suiker, Mechanical performance of wall structures in 3D printing processes: theory. design tools and experiments, International Journal of Mechanical Sciences, Vol 137, pp 145 - 170, DOI 10.1016/j.ijmecsci.2018.01.010, 2018.

[6] Poston, T. and Stewart, I., Catastrophe Theory and Its Applications, Dover Publications, 1996

Chapter 2
Body Weight Prediction of Turkeys: From Walk to Mass

Alexandery Mey[1], Behrouz Raftari[1], Elizabeth Zúñiga[2], Javier Fernández[3], Jos Hageman[4], Martin Stefanov[4], Oliver Sheridan-Methven[5], Yang Zhou[6]

Abstract:
Force plates are useful for examining kinetic movements of turkeys. They provide information about the external forces when a turkey stands or walks on them. Such collected time series data can be used to calculate the weight force and body mass of a turkey. Obtaining data of a high quality and minimum error requires understanding a force plate, good control of data collection, as well as accurate process of transferring data and analysis.
The aim of this article is to investigate how some mathematical and statistical models, combined with machine learning procedures, can help to predict body weight of turkeys based on electrical signals from a force plate. The authors also provide a discussion about techniques for arranging force plates and a scientific method of data collection to assist with a few practical examples of how force plates and mathematical models can be used to reduce manual workload. This work was done jointly with Hendrix Genetics, who in particular provided the data.

Key words: *Time series; Signal filtering; Force plate; Machine learning; Bayesian machine learning, Bayesian hypothesis testing.*

[1]Delft University of Technology, The Netherlands
[2]Université d'Évry Val D'Essonne, France
[3]University of Mondragon, Spain
[4]Wageningen University, The Netherlands
[5]University of Oxford, UK
[6]University of Bath, UK

2.1 Introduction

The problem of predicting the body weight of turkeys by using a force plate was raised by Hendrix Genetics. This introduction section will briefly explain the business interest of the company and why solving the problem could be valuable for the sector and the commercial values of Hendrix Genetics. The main focus of our report is to demonstrate the potential use of statistical models and machine learning techniques to be applied to the problem. However, we aim to present the report in a way that it is also accessible for non-experts in the field.

2.1.1 Background

Hendrix Genetics is a Dutch multispecies animal breeding, genetics and technology company and one of the world's leading breeders and distributors of turkeys and laying hens (fig. 2.1).

Figure 2.1: Hendrix Genetics has breeding programs in turkeys, layers, traditional poultry, swine, salmon, trout and shrimp. They look for innovative, sustainable solutions, together with the entire animal protein chain. At the start of the chain, through better breeding, they have a big influence on the outcome at the consumer level.

The core business consists of the following four steps:

- data collection is carried out for one generation of one breeding species;

- genetic evaluation methods are used for understanding the physical features of that generation;

- animals' selection strategies are outlined based on the genetic evaluation of the generation;

- strategies are considered for breeding the future offspring of the considered generation. Selected individuals will become the new generation to be studied and evaluated through the whole process again.

An important feature of turkeys is their body weight as it can be used to monitor their development. The current practice for weighing these animals is by picking them up and standing with them on a scale. As these animals can weigh up to 25 kilos, this is a slow and labour intensive task which can potentially be stressful for the animals. As a consequence, the animals are not weighed as often as people would want to.

One way of weighing the animals more often and at the same time improving animal welfare is the use of an automated setup for weighing the turkeys. One way of automating weight readings is to measure the turkeys when they walk across a force plate. This way, measurements can be taken each time a turkey crosses the force plate while the animals remain unhindered. A force plate gives of a set of varying electrical signals as long as the animal walks on the force plate. The idea is that the signals are linked in some way to the weight of the turkey. In this study, we will investigate if and how we can use the electrical signals from the force plate to predict an animal's body weight. For this purpose we have a set of roughly 200 animals that have walked across a force plate and also have their body weight manually determined. We will use different mathematical, statistical and machine learning approaches to this end.

2.1.2 Outline

In our report, we will examine the problem in four main sections:

- In Section 2.2 we approach the problem by explaining how the problem at hand can be linked and translated into mathematical language. We discuss in particular how physical values of forces can be calculated from the output signals of force plates.

- In Section 2.3 we present the mathematical and machine learning models we used to approach the problem. We introduce our four models: Bayesian hypothesis testing, sparse Bayesian generalized linear model, Linear Regression and Random Forests.

- The Results section presents and compares the results from all four models. Each model has its advantages and disadvantages in dealing with different types of input. At the same time, the outputs of the models presents insights of our problem of determining the weight of turkeys from different perspectives.
- In the last section, limitations and applications of the models, practical advice on improving data collection and modifying models with new data, are discussed.

2.2 Theoretical Background

2.2.1 Problem Description

We want to develop a model that can accurately predict turkey body weights using a force plate, removing the need to manually weigh the birds on a scale. The challenge of this project is to develop an innovative model to predict body weight and maybe other features from the plate measurements. A model is considered accurate enough when the mean average error is 50 grams.

A force plate is simply a metal plate with one or more sensors to produce an electrical output proportional to the force on the plate. One of the applications of the force plate is to measure the reaction force of the ground on each foot while walking. Since here we are studying an animal, not a human being, and the fact that we can not tell an animal to walk properly, it is better to replace "walking" with "motion". So apart from walking, the movement can include standing, running, jumping and whirling. Accordingly, the time-series records that we have for the turkeys probably are related to a combination of all types of movements. As a result, although this information shows some interesting details of the motion process, it cannot be fully understood and analyzed. Now, in order to estimate the body mass through the dynamic force measurements, it will definitely be useful to study each type of movement separately.

Let's start with the easiest one which is "standing". If a turkey stands on the force plate, a force roughly equal to Mg is recorded. Here, because we are interested in body weight, when we say "force", it means the vertical component of force (F_z). An accurate Mg can be recorded, when the turkey is completely stable and maybe with the center of mass on the vertical line passing through the turkey legs. Any imbalance and consequently a change in the center of mass eliminates this accuracy and causes the force to be greater/less than Mg. During walking, F_z initially rises from zero to the maximum Mg, then goes below Mg and repeats a similar process and eventually arrives at zero again at the end. This is definitely related to the 3D motion of the center of the mass. Considering a 2D picture in the vertical axis and the direction of movement,

the center of the mass follows a swinging curve path. The same analysis is expected to be at least qualitatively valid for "running", since running can be seen as walking with high speed. Let us stop here and assume that motion only involves standing, walking and running, so it can be argued that the maximum F_z/g can be an acceptable, though rough, estimate for mass. This conclusion is supported by our result shown in Fig. 2.2, which compares actual weights with the max and mean values of F_z/g. However, this argument is challenged if we add jumping to move. This is due to fact that by jumping a force much greater than Mg will be recorded. The force generated by jumping is pretty much the same as that of a bouncing ball. The last type of motion considered here, whirling, is perhaps the most complex one. With that in mind, the only thing we can easily understand is that the force will be lower than that of jumping. Further details on the force plate, along with a comprehensive analysis of various types of motion, can be found in the well-cited study of [2]. The complications mentioned are part of the reason we prefer machine learning techniques to a model based on physics.

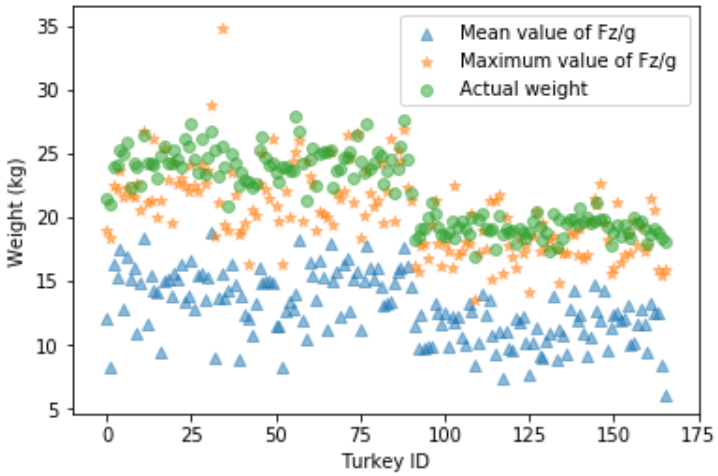


Figure 2.2: Actual weights of the turkeys, max and mean values of F_z/g. Individual turkeys are indexed on the x-axis.

Kistler force plates are useful tools in many areas as clinical research, performance diagnostics and motion analysis. Hendrix Genectics used this force plate to perform a gait analysis in 200 turkeys. The force plate output voltages (eight channels) are used to compute ground reaction forces, moments, center of pressure and coefficients of friction as seen in Appendix 2.A.

We are interested to find a model that can accurately predict turkey body weights using the force plate outputs. The turkeys walks over the force plate approximately for 15 seconds while voltages are measured per centisecond. This information results in time

series of 1500 measurements for each turkey in each channel. This data together with the computed forces, the blood line (identified as 1 or 2) and the actual weight of the turkey constitutes the database we work with.

2.2.2 Theories

The authors of [4] measure the body weight with smart insoles. Their methodology is based on Newton's second law, which states that the net force is equated to the product of the mass times the acceleration. When integrating all forces in the vertical direction (plane-z) over time, it leads to the next equation:

$$\int_{HSL}^{HSR} (F_{ZL} + F_{ZR} + W)dt = m \int_{HSL}^{HSR} a_z dt, \tag{2.1}$$

where F_{ZL} and F_{ZR} are the vertical forces from the left and right foot respectively. Furthermore a_z is the gravitational acceleration and W is the body weight. In their experiment, participants start with their left foot and finish with the right one, therefore the measurement starts from the first left heel strike HSL until the last right heel strike HSR. They stated that mean vertical velocity is zero in level walking, thus $\int_{HSL}^{HSR} a_z dt = 0$. Assuming the body weight does not change over the short period of the test, it leads to the following equation:

$$-W = \frac{\int_{HSL}^{HSR}(F_{ZL} + F_{ZR})dt}{t_{HSR} - t_{HSL}} \tag{2.2}$$

The integral (2.2) will be used as a variable in some of our models, computed taking the total force F_z exerted by the two foot. Figure 2.15 from [1] shows the typical form of the curve of the vertical ground reaction force F_z over time when a person is in the stance phase (foot in contact with the ground) of the gait cycle. The curve has two peaks that surpass the body weight and in the middle of these two peaks, a minimum is attained. The curve starts once the foot contacts the ground and finishes once the heel lifts away from the ground.

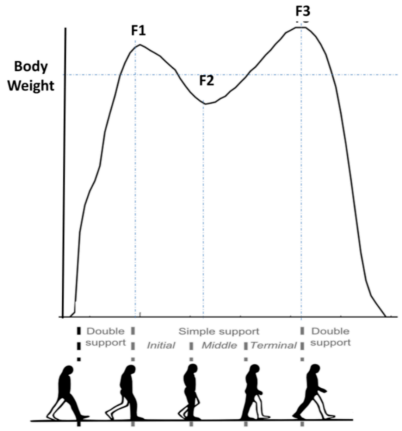


Figure 2.3: Vertical ground reaction force F_z.

2.3 Methodology

2.3.1 Bayesian Hypothesis Testing

One approach which suggests itself for tackling this problem would be Bayesian hypothesis testing. To see this we note that before any turkey walked across the force plate, the farmer/breeder of the turkeys will likely have an initial guess at what the weight should be. After a measurement from the force plate has been recorded, the farmer would have a better estimate of the turkey's weight. However, given the measurements are noisy, the farmer has an estimate for the turkey's weight after the measurement, but with a degree of uncertainty. Assuming the farmer could force the turkey to repeatedly walk over the force plate, with more measurements, the farmer would hopefully become more certain about the birds weight if the readings were consistent, and less certain if the readings were very noisy or contradictory. The process of updating some initial estimate of the birds weight as data becomes available is naturally modelled using a Bayesian updating scheme and can be posed as a hypothesis testing problem. Furthermore, if a farmer wished to know a turkey's weight to a given confidence, this approach would predict when the farmer can remove a turkey from the force plate, and when he requires more readings and longer measurement periods. For a more detailed treatment of the mathematics underlying this section we recommend the reader to Grindrod [3, Chapter 5].

To begin by establishing the mathematical framework, let us denote $\mathbb{P}(A)$ as the probability of an event A happening, $\mathbb{P}(A \cap B)$ as the probability of both A and B happening, and $\mathbb{P}(A \mid B)$ as the probability of A happening given that event B has happened (called a *conditional* probability). We can relate these three quantities using

Bayes' theorem [3, pages 219–223]

$$\mathbb{P}(A \mid B) = \frac{\mathbb{P}(A \cap B)}{\mathbb{P}(B)}. \tag{2.3}$$

We can denote the set of previously recorded/known observations as D, and a new observation d. Before observing the data we may have a hypothesis H for what the turkey's weight is, and the alternative hypothesis H^{c}. Based on the newly observed data we would like to update our belief that H is true or false. This can be achieved by trivially manipulating eq. (2.3) into the Bayesian updating scheme

$$\underbrace{\mathbb{P}(H \mid D, d)}_{\text{Posterior}} = \underbrace{\frac{1}{\mathbb{P}(d)}}_{\text{Normalisation}} \underbrace{\mathbb{P}(d \mid H, D)}_{\text{Model}} \underbrace{\mathbb{P}(H \mid D)}_{\text{Prior}}. \tag{2.4}$$

The question is then, how can we apply Equation (2.4) to the problem of weighing a turkey? This is easily done, where the first step is to realise that the weight of the turkeys is a continuous property, and hence we must discretise the turkey's weight into certain levels. These could be very coarse discretisations, or arbitrarily fine discretisations, where for mathematical and computational purposes the latter will be more convenient. Our hypothesis then is that the weight w of a turkey is in a weight class W_i. In this case we can write Equation (2.4) in the form

$$\mathbb{P}(w \in W_i \mid D, d) \propto \mathbb{P}(d \mid w \in W_i, D)\mathbb{P}(w \in W_i \mid D) \tag{2.5}$$

where we have ignored the normalisation factor, (which is trivially implemented).

The next question is, what are the data which we observe? The observations ultimately come from the force plate in the form of a force profile as in Figure 2.4. From this we can extract various possible features to act as the new observation d and we show some possibilities in Figure 2.5. For the sake of ease and simplicity to demonstrate the method we will present the following results using the value of the maximal force difference as our observed feature d, which is depicted in the bottom left of Figure 2.5, which we call the variation in F_z.

The first and easiest part of implementing this scheme is coming up with a prior estimate for a turkey's weight. In the presence of prior knowledge, a good choice is to use the empirically observed distribution of weights for the turkeys. This prior distribution is shown in Figure 2.6. (An even better prior would be the last posterior distribution for an individual turkey on a previous occasion, provided it is available and the turkey can be identified).

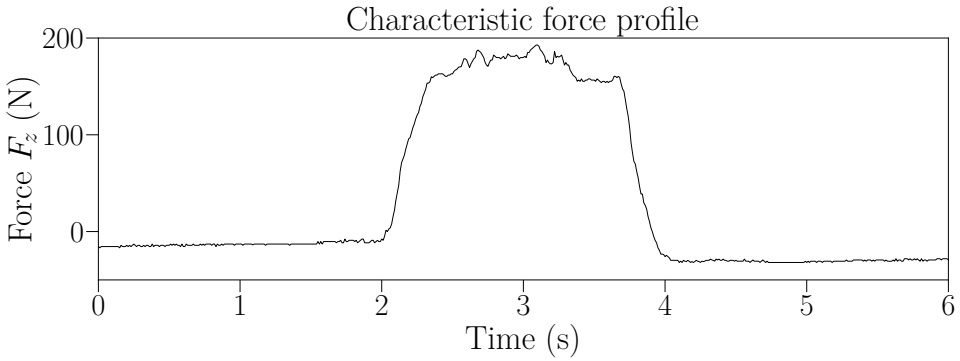


Figure 2.4: Characteristic force profile as a turkey walks over a force plate.

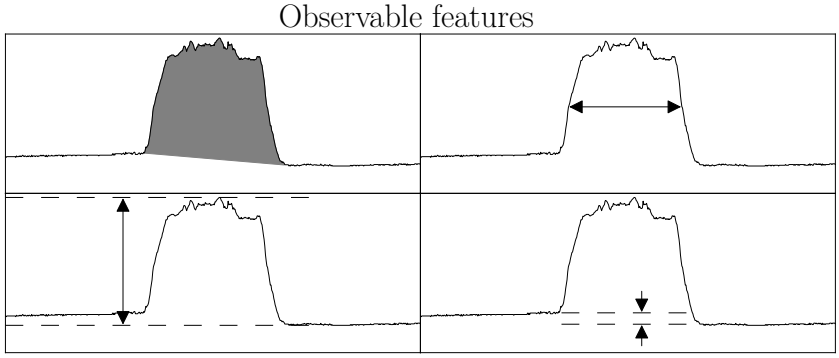


Figure 2.5: Some possible observable features which can be extracted from Figure 2.4. For much of our discussion we use the maximal force difference, which is the bottom left feature.

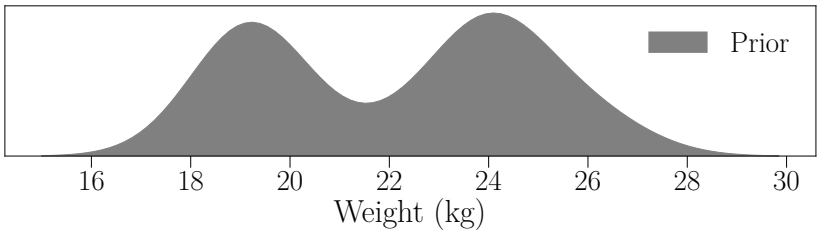


Figure 2.6: A prior model for the turkey's weight.

Having a prior, we can construct a model. The easiest way to do this is to empirically see if a feature is non-uniform across weight classes. This can be readily done by seeing the empirical distribution of a feature and the corresponding weights. The model can be constructed from this by a simple *kernel density estimate* (KDE), where for our purposes we use a Gaussian KDE. An example of such a Gaussian KDE is shown in Figure 2.7. Having constructed a model, we can evaluate the model update in Equation (2.5) for a given weight class by using the value of the normalised margin for the given weight

class (depicted by the horizontal region in Figure 2.7). For a different weight class we update similarly, ensuring the marginals are normalised. After this update we only have to normalise the posterior distribution is normalised.

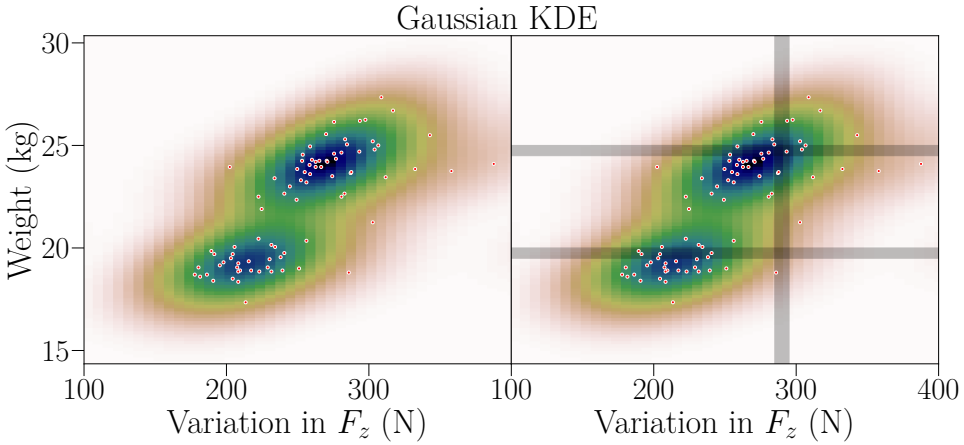


Figure 2.7: The Gaussian KDE formed by considering the variation in F_z. On the right we show two marginal cross sections for two different weight classes and a value for a new observation.

Having outlined how we construct the Bayesian model and apply Equation (2.5), it is informative to see how the prior changes for a given observation. A demonstration showing the difference between the prior and posterior is shown in Figure 2.8.

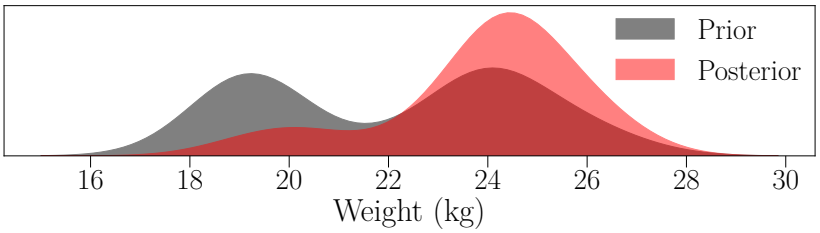


Figure 2.8: A posterior model of the turkey's weight.

It is then a matter of discretion about how to measure the accuracy of such a scheme. Should the predicted weight be the most likely weight, the expected weight, the median, etc. Similarly, how should the error be measured? As we have a probability distribution we could now measure the classification performance using either the RMSE, or a weighed RMSE, where we use the posteriors inverse variance as a weighting. There are several possibilities that can be explored here, and this is left as an avenue for further

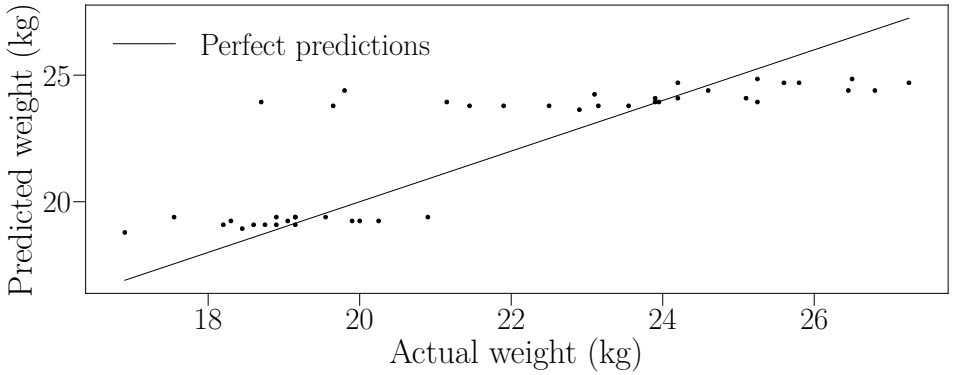


Figure 2.9: The points represent for each considered individual its predicted weight from a simple Bayesian hypothesis testing scheme.

investigation. If we take the most likely weight as our predicted weight, then we can compare the effectiveness of such a classification scheme, as shown in Figure 2.9.

To see the improvement achieved from only using a single feature (recall that we suggested several in Figure 2.5) we can compare the RMSE from using only the prior to the posterior. Using just the maximal value of the prior the RMSE was 4.3 kg, and using the mean of the prior gives a RMSE of 2.8 kg. Using just the maximal value of the posterior gives an RMSE of 2.1 kg, and using the mean gave 2.0 kg, although this latter improvement in going from the maximum likelihood estimate to the mean is likely not statistically significant.

2.3.2 Sparse Bayesian Generalized Linear Model

The first step in this work was to clean up and visualize the data in aid of the analysis. First, we removed duplicate data. Second, we inspected the time series of the 3D forces per turkey. We classified the time series manually in terms of their shape and baseline offset. Furthermore, we eliminated incomplete or dubious time series.

We extracted the main peaks from the time series to filter out the signals before and after a turkey walked over it. We extracted the beginning and end-points from the (vertical) z-force time series as they are the more consistent in shape than the x-, and y- forces. We performed this selection with the help of two lines defined by the initial, final, maximum and mean observations (Figure 2.10). The horizontal coordinate of the maximum observation and the mean of the time series formed one point of both lines. The initial and final points defined the second points of both lines. The beginning and end of the measurements were defined by the maximum distance between these liness and the time series. The reasonable location of the starting and end points were manually

inspected for all turkeys. The baseline was defined as the line between the extracted starting and end points and was corrected for. The baseline of the forces in the x-, and y- directions were corrected for in a similar fashion, but the signals themselves were transformed on absolute scale.

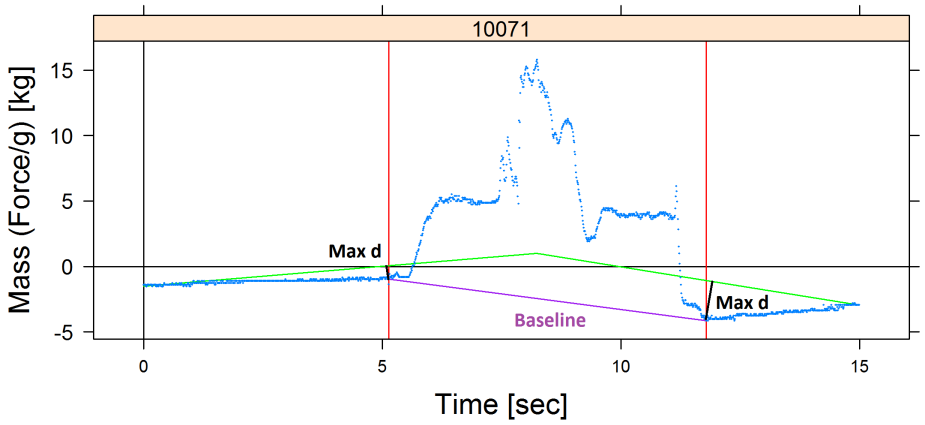


*Figure 2.10: Extracting the measurement portion of the z-force (vertical) time series for a turkey with Wing band number 10071. The vertical axis reflects the measured mass, and the horizontal axis the time. The blue points are the measurements. The green lines connect at the mean force during the entire measurement process at the horizontal location of maximally measured force. They start at the beginning and end of the time series. The maximum distance (**d**) below the light green lines and above the measurement curve determines the start and end of the turkey measurement process as visualized by the horizontal red lines. The purple line connects the start and end point of the measurement process and marks the adjusted linear baseline.*

The bodyweights Y of turkeys i were modelled according to eqs. Equation (2.6)-Equation (2.10), where *Normal* and *Cauchy+* indicate normal and positive Cauchy distributions; μ_{i} is the expected weight of turkey i; σ is the scale (standard deviation) of the measurement noise; α is the average weight of turkeys from line 1; $Line2_i$ is a logical variable that indicates if turkey i belongs to line 2; β is the difference between the average bodyweights of turkeys from lines 2 and 1; $\gamma_{j,k}$ indicates the effect of a force curve quantile k for the j force (mx, my, or mz); $X_{j,k}$ indicates the quantiles k for the force j at 1% intervals (i.e. 0% , 1% , 2% , ..., 100%) for each turkey i; τ is the global scale parameter of a horseshoe prior, whereas λs are the local scale parameters; p_0 are the expected number of non-zero parameters in the model, out of all D parameters; n is the total number of observations (turkeys); and τ_0 is the scale of the parameters. The normal distribution is used to capture the measurement noise of turkey weight estimation. We

estimated the average bodyweight per genetic line of turkeys and refined the estimation per turkey with the help of the x-,y-, and z-force quantiles. It should be noted that the quantiles of the x-, and y- forces were taken from their absolute measurements. In addition, the quantile predictors were standardized by subtracting the mean and dividing by the standard deviation. The effects of the quantiles were modelled with a horseshoe prior. The horseshoe prior induces sparsity (i.e. near zero effects) if the effects of the predictors were too small to be distinguished from the measurement noise. It models the effects as normal distributions with Cauchy distributed scales. The global scale parameter τ is related to the scale of the measurement noise σ and determines the effects (parameters) that are large enough to be explained by the measurement noise alone.

$$Y_i = normal\left(\mu_i, \sigma\right) \tag{2.6}$$

$$\mu_i = \alpha + \beta \cdot Line2_i + \sum_{j=1}^{n_{forces}} \sum_{k=1}^{n_{quantiles}} \gamma_{j,k} \cdot X_{j,k} \tag{2.7}$$

$$\gamma_{j,k} \sim Normal\left(0, \tau \cdot \lambda_{j,k}\right) \tag{2.8}$$

$$\lambda_{j,k} \sim Cauchy\left(0, 1\right)^{+} \tag{2.9}$$

$$\tau = \frac{p_0}{D - p_0} \cdot \frac{\sigma}{\sqrt{n}} \cdot \tau_0 \tag{2.10}$$

In the spirit of Bayesian statistics, we provided prior information for the parameters. We assigned a standard normal prior to the measurement error since the force plates are expected to provide accurate measurements within 1 kg accuracy. Furthermore, we put a normal prior with mean of 20 kg and standard deviation of 2.5 kg since the average weight of adult turkeys is expected to be in the range of 15-25 kg on the basis of the information provided in the presentation from Hendriks Genetics. The scale of the difference between lines should be the same as the scale of the average turkey weight. Also, the standardized quantiles are expected to improve the accuracy of turkey estimates beyond the turkey line average by no more than the scale of the average turkey weight. Thus, the global scale of the parameters was set to 2.5 kg.

Lastly, we estimated the leave-one-out root-mean-square-error (LOO-RMSE) by pareto smoothed importance sampling (PSIS). This is a fast approximation to the LOO-RMSE obtained from leave-one-out cross validation.

Table 2.1: Prior distributions for the parameters in the model.

Parameter	***Designation***	***Prior***
σ	Scale of measurement error	Normal(0,1)
α	Average weight of turkeys from line 1	Normal(20,2.5)
p_o	Expected non-zero parameters in the model	10.5 out of 21 (i.e. 50%)
τ_o	Scale of parameters	Cauchy+(0,2.5)

All analyses were carried out with the program *R*, and the packages *gdata*, *lattice*, *ggplot*, *rstan*, rstanarm, *loo*, *brms*, *plyr*, *dplyr*, and q*uantmod*.

2.3.3 Appliance of Machine Learning algorithms

In this approach, the aim was to discover the capacity of the regressor models to predict the target value, based on the attributes that complement it. For this, tests have been carried out with Machine Learning regression algorithms: *Random Forests* (RF) and *Multiple Linear Regression* (MLR). In the same way, we wanted to know its performance using all the attributes (a.k.a. variables), or using only those that were more correlated with the target variable.

MLR has been used because it is the most basic regression algorithm. With this, it was intended to demonstrate the usefulness of the attributes and their selection to solve the problem. On the other hand, RF has been used to check if with a more sophisticated regression algorithm the results can vary.

First the data has been pre-processed. As can be seen in Figure 9, there are entries in which the peaks are not clearly defined. In some cases there is more than one peak for the same entry (5), and in others the capture of the signal has not been completed (7). Therefore, we decided to eliminate those data that contain incomplete or duplicate entries. For the rest of the cases, the signal has been cut to work only with the part in which the peak is represented. Once the signal relative to the peak has been cut, the equivalent cut-off signal is obtained for the rest of the signals.

All these analyses were carried out with the programming language *Python*, and the libraries *matplotlib, pandas, scikit-learn* and *numpy*.

2.4 Results

2.4.1 Sparse Bayesian Generalized Linear Model

Characteristic measurement time series can be seen in Fig. 2. The typical time series (Figure 2.11.1) constitutes of two peaks in the vertical z-force that likely reflect the two steps of a turkey. However, the second peak is always lower than the first. What is more, the baseline does not return to 0 after a bird steps of the force plate but goes to negative values. The direction, number of peaks, and shape of the x- and y- force time series are highly variable and unpredictable. Many turkeys have been measured before the z-force measurement returned to the null baseline (Figure 2.11.2). In some time series it looks like that if a turkey walked fast and that the two steps appear as 1 (Figure 2.11.3), whereas others seem to show a turkey making many slow steps (Figure 2.11.4). Unfortunately, there are time series with two distinct signals (Figure 2.11.5), though sometimes it is possible to filter out the likely walk of the turkey from the (smaller) sensor disturbance (Figure 2.11.6). Finally, there are incomplete time series that suggest that the measurement process started too late (Figure 2.11.7). Turkey measurements of the later type were removed from further analysis.

We extracted the portion of the time series of all forces when the turkey walked by filtering the time series of the z- force (Figure 2.12.1). We then adjusted for the dynamically sinking or raising baseline of all forces (Figure 2.12.2-3). This step did not seem to fix the consistently higher z-force(vertical) peak of the first turkey step, but it did centre the time series around the null baseline. The algorithm that we employed seemed to work on manual inspection of the time series of all turkeys (not shown). It should be noted that the estimated baseline for the x-, and y- forces can be positive because the signal can be positive or negative depending on which leg the turkeys make the first step with.

Fitting the model on the full dataset yielded a measurement error with a scale (standard deviation) of 0.75 kg with a standard error of 0.05 kg. This implies that for 95 (%) power, the same turkey needs to be measured ca. 5848 times to estimate its weight with the desired within 0.05 kg accuracy. This method is comparable to the others as the correlation between the fitted values and the data is 0.93 (Figure 2.13.1), and cross validation of the model resulted in an LOO-RMSE of 0.85 (Figure 2.13.2). The residuals appear homogeneous, although there are a number of outliers.

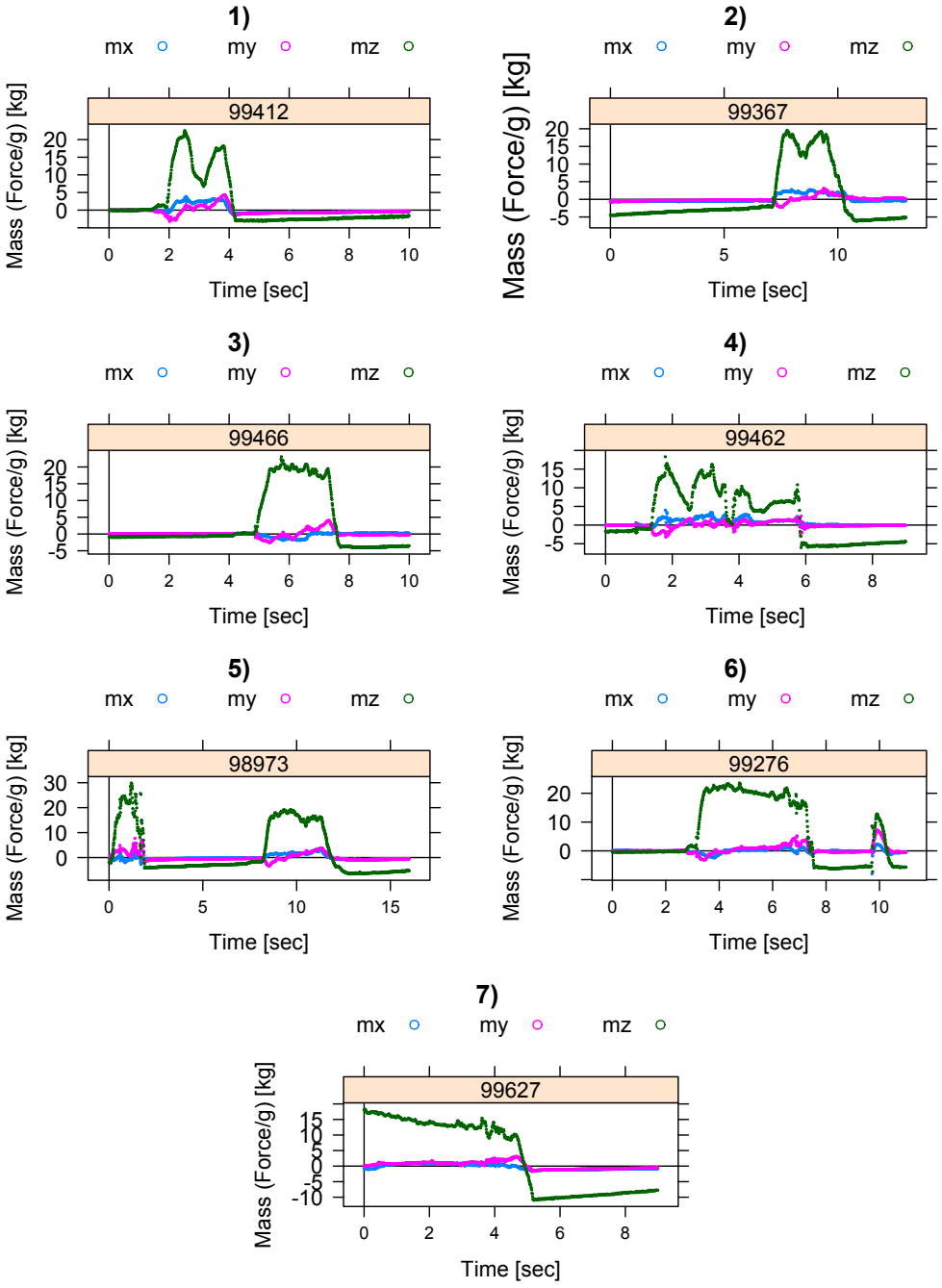


Figure 2.11: Typical time series of the 3D forces in the x-, y, and z- directions, or noted as mx (blue), my (red), and mz (green). The vertical axis reflects the measured mass, and the horizontal axis the time. 1) Typical time series with peaks for 2 steps, 2) Time series with offset, 3) Time series with unified 2 steps, 4) Time series with multiple steps, 5) Time series with 2 possible measurements, 6) time series with disturbances after the measurement, 7) incomplete time series.

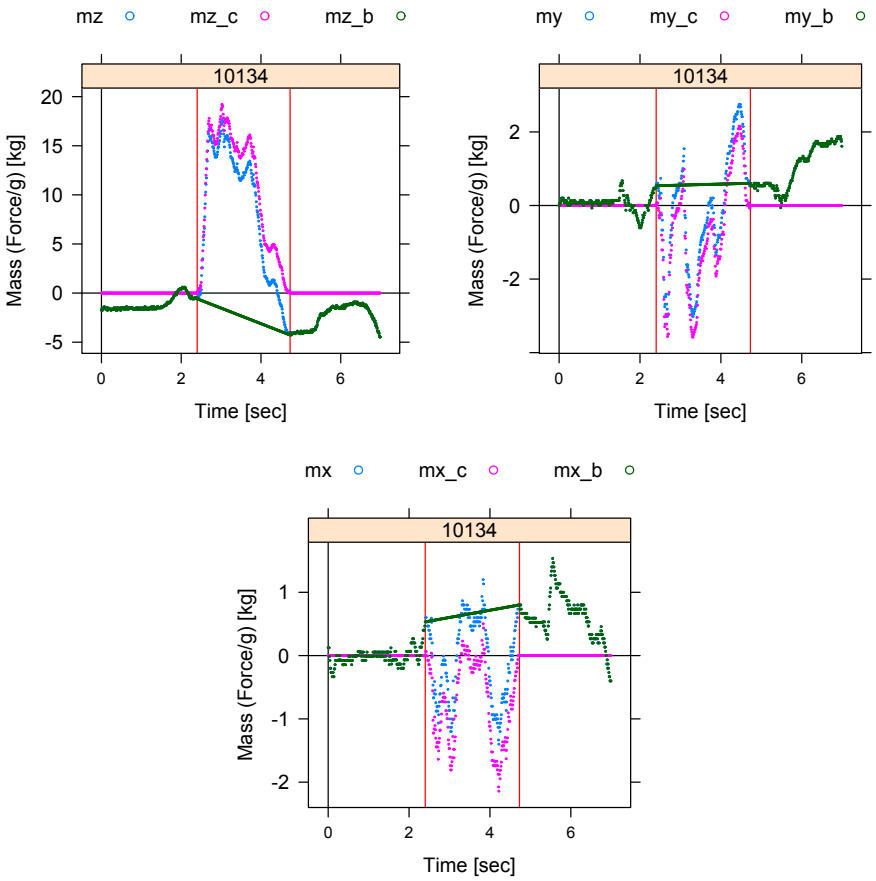


Figure 2.12: A sample force time series (blue) with corrected baseline (green), and adjusted measurements (red). The subplots indicate the 3D forces (mz, my, and mx) with suffices indicating the baseline corrected (_c) time series, and the baseline (-b).

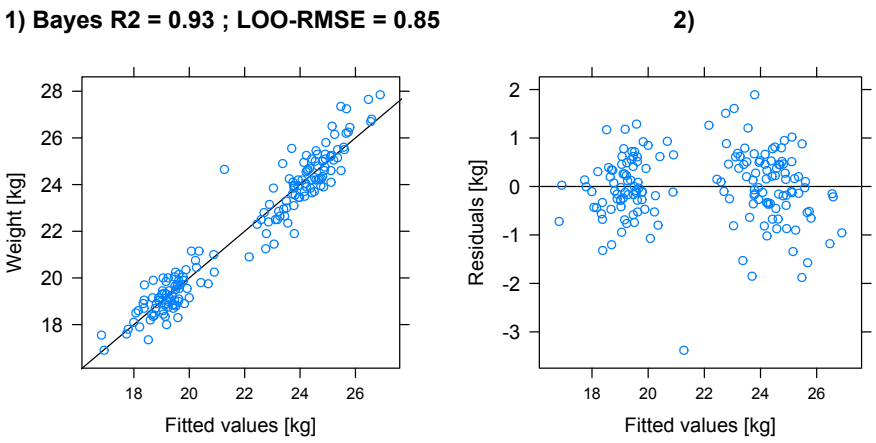


Figure 2.13: Quality of the predictions. 1) Fitted vs measured values, 2) residuals.

2.4.2 Learning Curves with Different Features

In this subsection we demonstrate the influence of the selected features for the prediction. We essentially demonstrate that most information is already captured by the integral of F_z (see also other sections) and the bloodline. First we describe the different types of features we test.

Bloodline As there are two bloodlines of turkeys, the bloodline takes either the value 1 or 2. More information on that is in earlier sections.

Integral F_z This is indeed the integral of the force F_z, so the force in the z-direction, from the cropped signal. So essentially the value calculated in Equation (2.1).

Summary With summary we captured a summary of the raw cropped signal from each of the 8 channels. For that we divided each cropped channel in 5 intervals and stored the mean value of each interval. This leads to 40 features per Turkey.

To compare the different features we show corresponding learning curves in Figure 2.14 as well as the leave-one-out error on the full dataset as shown in Table 2.2. For all curves we used a simple linear ridge regression scheme with regularization set to 0.1. The bloodline alone gives already a very good baseline with a leave-one-out root-mean-sqaured-error (LOO-RMSE) of 1.25, while the integral alone only achieves a LOO-RMSE of 2. Nevertheless, there is still valuable information in the integral, as when we add it to the bloodline we achieve a LOO-RMSE of 1.14. Although it does not seem much of an improvement, it is still significant with a p-value of 0.0082.[7] Through the learning curves we see that more data will not help when using those simple features, at least when using a simple linear regression scheme.

The summary features are able to outperform the integral feature alone, which indicates that there is more to learn from the channels than the force in the z-direction alone. Unfortunately though, the summary features together with the bloodline cannot outperform the simple integral and bloodline features.

For the summary feature we also observe that there is still a gap between the test and train error in the learning curve, so more data could help in this case. We note, however, that even in this case we expect a LOO-RMSE around 1 as this is the value that the

[7]The significance test was done with a paired two-tailed t-test.

training error approaches. So in case we have more data and we would use the summary feature we need a model that is more flexible than a simple linear regressor.

Features	Summary	Summary+Bloodline	Bloodline	Integral	Integral+Bloodline
RMSE	1.59	1.45	1.25	2	1.14

Table 2.2: The RMSE based on leave-one-out on the complete dataset.

2.4.3 Application of Machine Learning algorithms

There are three different tests to compare the performance of the regressors: (i) All: using all variables (the raw signals offered by the company and the calculated weights); (ii) Only Weights: using the weights calculated from the original variables; (iii) Correlated: using only the initial and transformed variables that were most correlated with the target label.

The results are reflected in table 2.3. The data are relative to the Root Mean Squared Error (RMSE):

Table 2.3: Results obtained by the application of Machine Learning algorithms.

Algorithm	**Only Weights**	**All**	**Correlated**
Multiple Linear Regression	1.167	1.125	1.034
Random Forests	1.142	1.058	0.706

Two conclusions can be obtained from these results: (i) applying the *Random Forests* algorithm with a configuration of 10 trees for the training, a lower RMSE is obtained than in the case of the *Multiple Linear Regression* in any case. (ii) Significantly better results are obtained using only the correlated features than all at the same time.

This leads to the consideration that the weight of the turkey is actually correlated with specific attributes, and that there are signals that are not relevant when generating the prediction. On the other hand, another of the main conclusions is that there have not been large amounts of data. This is due to the fact that a quantity of the captured data was incomplete or noisy, and it has been decided to eliminate from the historical data for the learning process. Possibly, if more clean and valid data were available, the prediction would be better through Machine Learning algorithms like those shown.

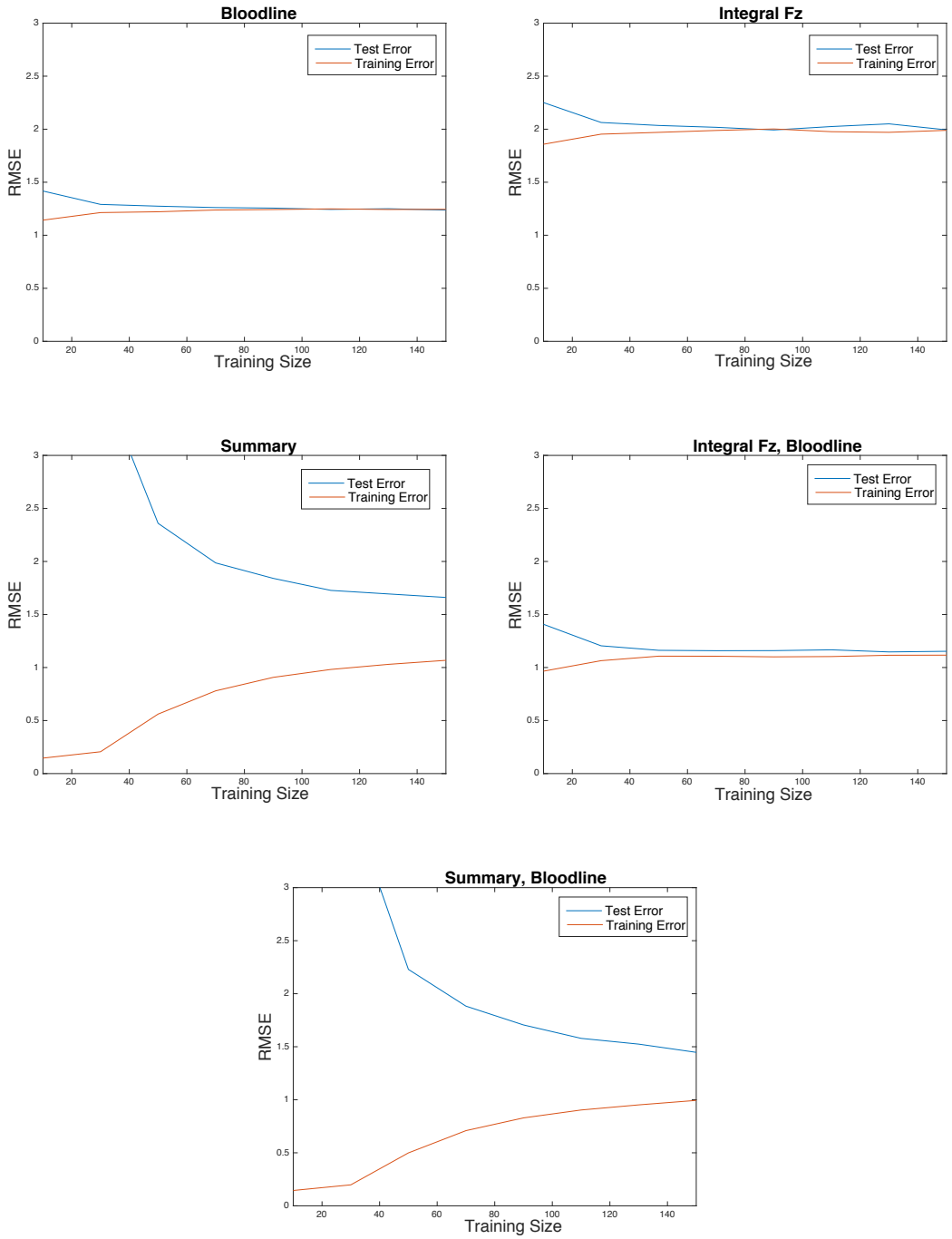


Figure 2.14: The five different learning curves when using different sets of features. We always use ridge regression with a linear Kernel and the regularization parameter set to 0.1*. On the top of each plot we indicate which features we used, while we plot the root mean squared error (RMSE) vs the training set size. We show the RMSE as measured on the training set as well on the test set.*

2.5 Conclusion, Discussion and Further Research

2.5.1 Conclusion

Our Bayesian hypothesis testing model requires more than one step from the turkeys. With every step from the same bird, the model can calculate and update the prior knowledge and produce much accurate results. The current model first give a guess that the body mass of a turkey is equal to the sample mean value.This guess produces initial RMSE of more than 4 kg. By taking one update of one step from the turkey, the posterior reduces the RMSE to around 2 kg. Therefore, the advantage of the first model is accuracy when there are more walking steps from the turkeys. From theoretical calculation, the accuracy could be good with information of 5 to 10 steps. Whereas, with the current limited data, this model gives the highest error.

Our second model - Sparse Bayesian generalized linear model - take the offset of signals into account. It applied Bayesian theory on learning the information from the time series. This method produces the best result by far and the RMSE is 1.14 kg. This is relatively good, but the the speed of increasing the accuracy is relatively slow compare to our first model. However, when more data are available, the model can be updated and produce more accurate results.

Our last two models uses basic machine learning: linear regression and random forest. These two are the simplest models but produces the best results regardless of the simplicity. The RMSE from Linear Regression is 1.034 kg and the RMSE from Random Forest is 0.706 kg. The input training data used the original noisy data without any baseline offset nor correction. The advantage of these two models are that they are really easy to build and gives really good results even with strong influence of noise. Disadvantage is that the improvement of these models heavily relies on the cleaning of noise. When we have clean data, the potential of increasing of accuracy will be quite limited.

Regarding the feature selection process we come to the following conclusion. Section 2.14 shows the integral feature (as defined in that section) together with the bloodline captures the most interesting features from the dataset. The summary feature suggests, however, that if we have more data available we could extract more information from the channels, possibly with more flexible regression schemes than linear regression.

2.5.2 Discussion

A few things need to be discussed before arguing how good our models are. The goal of this project is to explore the possibility of making prediction of turkeys' body mass within the error of 0.05 kg. However the trouble to verify the accuracy of our models

is caused by the missing knowledge of the force plate. For example, if the equipment itself could cause errors of more than 0.5 kg, then such measurement mistakes can only damage the accuracy of our models which directly relays on the accuracy of measured data. Such systematical errors could be only reduced from the measurement side not from the modeling.

Another question is the reliability of 0.05 kg as an accuracy indicator, because body weight of either human or fowls has a range of values to move within a day. Especially after meals or drinking water, body mass measured could easily have a increase or decrease of roughly 0.5 kg. Therefore, in order to scientifically justify the comparison, the measurements should all be collected within a specially chosen time period, e.g. before meal time.

A third thing that caused our attention is the noise in the given data. Most of the noise could be avoided if certain methods being considered when recording those data. Here we suggest a few practical methods for helping to collect data:

1) One way to take the most of our Bayesian Hypothesis model is by collecting more steps. This can be achieved by placing three more plates along the current one. So each plate can collect one to two steps and four together can give a really good number of step information for our first model to get well updated.

2) To improve the overall accuracy, we need to reduce the noise of collected data, especially the problem of negative baselines. This can be avoided by allowing a short break between turkeys walking on the plate. This will allow the voltage signal to come back to the zero value.

3) We need to carry out a few simple tests to help to understand the force plate. Staff can step on or walk over the plate with different break between them. This could show the accuracy and performance of the equipment between measurements. People who will develop further models based on our study should have some knowledge of those force plates. A discussion with the company who build the plates can be helpful.

4) In general research experiment, force plates are mainly used in labs. The surface of plates are clean and directly touched or connected to the experimented subjects. However in order to make sure the turkeys are behaving in a natural way, the force plate used here is covered with mud and feathers. How this coverage is influencing the plate should be understood as well.

2.5.3 Further Research

Our models presented in this report are relatively basic due to time limitations. We have achieved good results but more sophisticated models should be developed based on our conclusions and results.

When clean data are available, we strongly suggest to run the models on the new data to see the accuracy level. Then, based on the improvement suggestions, new data should be collected and models can use those information to update themselves.

The time limit during the project also prevented us develop other models to make the whole process automatic. We went through the plots of time series of each turkey manually, which is not the most ideal and efficient method. Advanced machine learning models can be developed for pattern recognition - automatic working to speed up the data process for the company. Another direction to improve our machine learning model is to develop deep learning algorithms to understand the data and situation further.

Bibliography

[1] A. Bonnefoy and S. Armand. Normal gait. *Orthopedic Management of Children With Cerebral Palsy: A Comprehensive Approach*, (3):567, 2015.

[2] Rod Cross. Standing, walking, running, and jumping on a force plate. *American Journal of Physics*, 67(4):304–309, 1999.

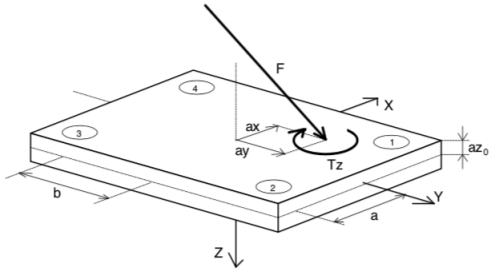
[3] P. Grindrod. *Mathematical Underpinnings of Analytics: Theory and Applications (1 edition)*. Oxford University Press, Oxford, January 27, 2015.

[4] Ming Tan, Yehuda Weizman, and Konstantin Fuss. Measurement accuracy of the body weight with smart insoles. *Proceedings*, 2(6), 2018. ISSN 2504-3900. doi: 10.3390/proceedings2060274. URL `http://www.mdpi.com/2504-3900/2/6/274`.

Appendix

2.A Force plate formulae

Kistler Force Plate Formulae



Force plate output signals

Output signal	Channel	Description
fx12	1	Force in X-direction measured by sensor 1 + sensor 2
fx34	2	Force in X-direction measured by sensor 3 + sensor 4
fy14	3	Force in Y-direction measured by sensor 1 + sensor 4
fy23	4	Force in Y-direction measured by sensor 2 + sensor 3
fz1 ... fz4	5 ... 8	Force in Z direction measured by sensor 1 ... 4

Calculated parameters

Parameter	Calculation	Description
Fx	= fx12 + fx34	Medio-lateral force 1)
Fy	= fy14 + fy23	Anterior-posterior force 1)
Fz	= fz1 + fz2 + fz3 + fz4	Vertical force
Mx	= b * (fz1 + fz2 - fz3 - fz4)	Plate moment about X-axis 3)
My	= a * (-fz1 + fz2 + fz3 - fz4)	Plate moment about Y-axis 3)
Mz	= b * (-fx12 + fx34) + a * (fy14 - fy23)	Plate moment about Z-axis 3)
Mx'	= Mx + Fy*az0	Plate moment about top plate surface 2)
My'	= My - Fx*az0	Plate moment about top plate surface 2)
ax	= -My' / Fz	X-Coordinate of force application point (COP) 2)
ay	= Mx' / Fz	Y-Coordinate of force application point (COP) 2)
Tz	= Mz - Fy * ax + Fx * ay	Free moment, Vertical torque, „Frictional" torque
COFx	= Fx/Fz	Coefficient of Friction x-component
COFy	= Fy/Fz	Coefficient of Friction y-component
COFxy	= sqrt (COFx^2 + COFy^2)	Coefficient of Friction absolute

All formulae are in Kistler coordinate system

1) Walking direction is positive Y-axis

2) az0 = top plane offset (negative value)

3) a, b = sensor offset (positive values)

Figure 2.15: Force plate formulae.

Chapter 3
Predicting the Removal Performance of Activated Carbon Filters in Water Treatments

Garnet Akeyr[1], Karel Keesman[2], Mara Smeele[1], Mia Jukic[1]

Abstract:
In this paper we investigate a model for the removal of pollutants in surface water via adsorption onto the surface of activated carbon (AC). Both micropollutants and organic matter are present in surface water due to a variety of sources such as industrial waste, agricultural runoff and household pharmaceuticals. The presence of these contaminants makes surface water unsuitable for drinking and so they must be filtered out. One of the final steps in the filtration process involves using AC as an adsorbate. Models used to estimate the removal efficiency of activated carbon for different contaminants have value in that optimal conditions for the process of water filtration can be found, and the removal efficiency of new contaminants can be estimated. We examined multiple approaches to improve the computational efficiency of the model used by KWR Water and conclude by offering recommendations based on our results.

Keywords: *adsorption filtration*

[1]Leiden University, The Netherlands
[2]Wageningen University, The Netherlands

3.1 Introduction

We consider a problem posed by KWR Water at SWI 2019. KWR Water is a water research institute focused on the entirety of the water cycle, providing the water industry with solutions and advice on their operations. One of the key steps in the water cycle is the treatment of surface water, which comprises around 40% of the drinking water in the Netherlands. In one of the latter stages of the purification process, micropollutants and natural organic matter (NOM) are filtered out of the water via adsorption onto activated carbon (AC).

Carbon filtering is a common type of water purification method, since activated carbon is considered an ideal material for filtration due to its very high porosity. During the process the contaminants that go inside the filter are adsorbed onto each carbon particle's surface. Due to the porous structure of the carbon particles, the contaminants then diffuse inside the pores. The process is illustrated in Figure 3.1. Once all pore space is covered, the carbon gets worn–out and should be changed. Not changing the filter on time will lead to the release of certain contaminants with the outflow from the carbon filter. Therefore simulating the process until the carbon filter becomes saturated is very useful in operation of the process. Frutehrmore, the efficiency of this process is generally unknown for newly detected micropollutants, which leads KWR to developing a model to simulate the filtration process.

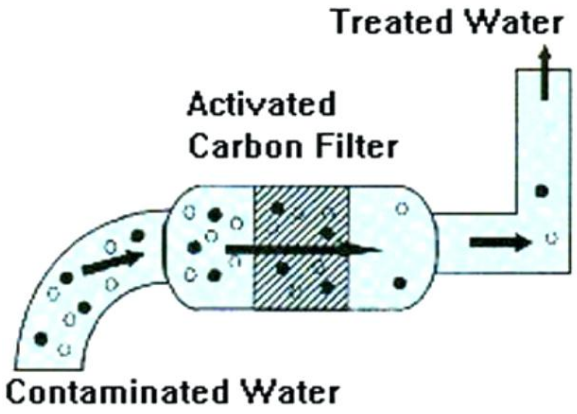


Figure 3.1: Illustration of water filtration via an activated carbon filter.

The model used by KWR is based on ideal adsorbed solution theory (IAST) using principles from thermodynamics. A key feature of IAST is that the rate of adsorption of contaminants is dependent on the other contaminants present, if any. For a more

in-depth treatment of IAST and the thermodynamics relevant to it, the interested reader may consult reference [2].
In this report we will reformulate the mathematical model used by KWR water and will further propose numerical methods, which can be used to solve two of the main problems of the company—robustness of the omputational methods (at present, KWR water solver does not always converge to a solution), large run-times for some numerical experiments.

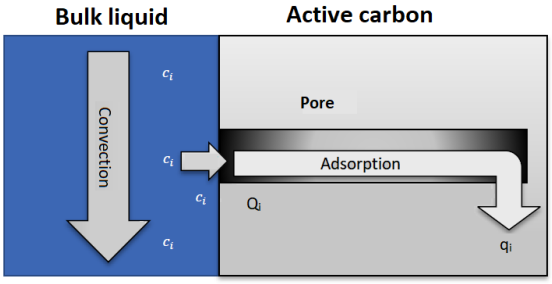


Figure 3.2: Schematic of the adsorption process in water filtration from [1].

The organisation of the report is as follows. In section 3.2, the IAST model used by KWR Water is examined and a reformulation in dimensionless quantities is presented. Section 3.3 describes the approach of KWR water for finding the solution of the problem. Section 3.4 presents the approaches we used to address the issues KWR Water had in their approach and Section 3.5 contains results from numerical experiments for troublesome values of the model parameters. We conclude in Section 3.6 by offering recommendations on how to improve the simulations based on our results.

3.2 Mathematical model

The process of carbon filtration is modelled in [1] by a system of partial differential equations. For each compound the concentration of micropollutant or natural organic matter is described by a one-dimensional convection-adsorption equation. Diffusion is assumed to be negligible in comparison to convection and the water flow rate is assumed to be constant.

The equation for the concentration is coupled to an equation describing the load on the carbon.

3.2.1 Original form of the model

Let $c_i(x,t)$ and $q_i(x,t)$ be the concentration and load of compound i inside the carbon at time t at the point x.

Let $t \in [0, T]$ and $x \in [0, X]$, where $x = X$ corresponds to the inlet of the carbon filter.

Then carbon filtration is modelled by the following system of PDEs, following [1]:

$$\begin{aligned} \frac{\partial c_i}{\partial t} &= \frac{\nu}{\epsilon}\frac{\partial c_i}{\partial x} - \rho\frac{1-\epsilon}{\epsilon}\gamma\left(Q_i - q_i\right), \\ \frac{\partial q_i}{\partial t} &= \gamma\left(Q_i - q_i\right). \end{aligned} \tag{3.1}$$

We close the PDE system by imposing the following initial and boundary conditions:

$$c_i(x, 0) = 0,\ q_i(x, 0) = 0, \quad c_i(X, t) = c_{in}.$$

Let us consider the physical meaning of the terms in each equation:

- In the equation for the concentration c_i, the term $\frac{\nu}{\epsilon}\frac{\partial c_i}{\partial x}$ is a convection term, where ν is the constant velocity of the water flow and ϵ is the filter bed porosity. The term $\rho\frac{1-\epsilon}{\epsilon}\gamma\left(Q_i - q_i\right)$ describes the adsorption, where Q_i is the load of compound i on the surface of the carbon and $\gamma = \frac{6\cdot 10\cdot D_s}{d_p^2}$ is the rate of transfer of a compound into the pore [1]. Here, D_s is the intraparticle diffusion constant and d_p is the particle diameter. The parameter ρ is the density of the carbon, so that $\rho\frac{1-\epsilon}{\epsilon}$ is the mass of carbon per volume unity.

- The equation for the load describes the change of the load due to surface diffusion.

The load on the surface of the carbon, Q_i, can be computed, based on empirical laws. If we assume there is only one compound in the water then Q_i can be determined by the Freundlich isotherm

$$Q_i = K_{F,i} c_i^{\frac{1}{n_i}}, \tag{3.2}$$

where $K_{F,i}$ and n_i are the so called Freundlich parameters of compound i. If there are multiple compounds in the water, there is competition between the different compounds and adsorption is limited. This competition can be modeled using Ideal Adsorbed Solution Theory (IAST) [1]. In this report, we will not go in further detail about IAST, since for our computations we assumed for simplicity that there is only one compound in the water.

Table 3.1: Model parameters

parameter	description	order of magnitude
$\gamma[\frac{1}{s}]$	rate of transfer of a compound into a pore	$\sim 10^{-6}$
$\nu[\frac{m}{s}]$	velocity of the water flow	$\sim 10^{-1}$
$c_{in}[\frac{mol}{m^3}]$	initial concentration	$\sim 10^{-2}$
$\epsilon[-]$	filter bed porosity	$\sim 10^{0}$
$\rho[\frac{g}{m^3}]$	density of the carbon	$\sim 10^{6}$
$K_F[\frac{mol}{g}\left(\frac{m^3}{mol}\right)^{1/n}]$	mass based Freundlich constants	$\sim 10^{-2}$
$1/n[-]$	Freundlich exponent of compound	~ 0

3.3 KWR's current approach—basic idea and issues

KWR water uses the original form of the model (3.1) to model carbon filtration. To solve the partial differential equations, KWR water discretises space to obtain a system of ODEs. To solve the system of ODEs, KWR water uses a build-in Python solver.To take the competition betweem micro-pollutants and natural organic matter into account a Python built-in package for IAST is used. Figure 3.3 shows a sketch of the approach.

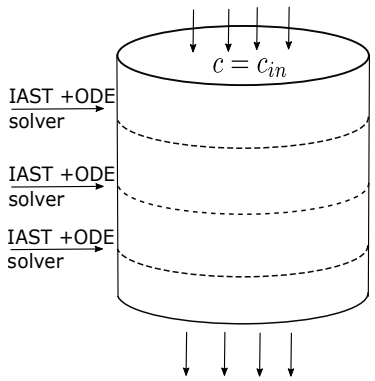


Figure 3.3: Sketch of the approach of KWR water for simulating carbon filtration.

As mentioned in Section 3.1, two main problems occur when using the Python solver—for some parameter values the solution does not converge or takes too much time to compute. Our goal was to localise the problem with the current approach and suggest a numerical scheme that is more robust and more efficient. For simplicity we assumed there is only one compound in the water. So we assume there is no competition, therefore we can ignore IAST. The reason for this assumption is that we think the problem of the current approach is due to using the Python built-in ODE solver and that the IAST method does not cause any problems.

In this respect, we note the following. To solve the system of ODEs the built-in Python solver uses adaptive numerical methods with a high order of convergence. However, the spatial discretization introduces a first-order approximation error and it is impractical to use high order methods for integration over time.

Thus, in the following sections we construct several different numerical methods and implement them in Mathematica and MATLAB for numerical experiments.

3.3.1 Dimensionless model formulation

Before setting up numerical schemes, we rewrite the system of equations (3.1) in dimensionless quantiities in order to decrease the number of parameters in (3.1). Also, this might reduce the problems in numerical schemes caused by different orders of magnitude of parameters, see Table 3.1.

We perform the change of variables

$$\bar{t} = \gamma t,\ \bar{x} = \frac{\gamma}{\nu} x,\ \bar{c}_i = \frac{c_i}{c_{in}},\ \bar{q}_i = \frac{\rho q_i}{c_{in}}. \tag{3.3}$$

Hence, the system of equations (3.1)-(3.2) now becomes

$$\frac{\partial \bar{c}_i}{\partial \bar{t}} = \frac{1}{\epsilon}\frac{\partial \bar{c}_i}{\partial \bar{x}} - \frac{1-\epsilon}{\epsilon}\left(\overline{Q}_i - \bar{q}_i\right) \tag{3.4}$$

$$\frac{\partial \bar{q}_i}{\partial \bar{t}} = \overline{Q}_i - \bar{q}_i, \tag{3.5}$$

$$\overline{Q}_i = \frac{\rho}{c_{in}} K_{F,i} M W_i^{\frac{1}{n_i - 1}} \left(c_{in}\bar{c}_i\right)^{\frac{1}{n_i}}, \tag{3.6}$$

where MW is the molar weight of the compound and $\overline{Q}_i$ is a function of $\bar{c}_i$. Very important effect due to rescaling of the variables is the change of the boundary conditions. Namely,

$$\bar{c}(x,0) = 0,\ \bar{c}(X,t) = 1,\ \bar{q}(x,0) = 0. \tag{3.7}$$

Therefore, from now on, we always solve this reformulated model. Also, we omit the bars above the c and q.

3.4 Numerical methods

The temporal domain is $[0,T]$ and the spatial domain is $[0,X]$, for some parameters T and X. We discretize the domain with the mesh (x_k, t_l), where x_k and t_l are defined as follows

$$\begin{aligned} x_k &= k\Delta x, \quad k = 0,\dots n, \quad n = X/(\Delta x), \\ t_l &= l\Delta t, \quad l = 0,\dots m, \quad m = T/(\Delta t). \end{aligned} \tag{3.8}$$

We denote by c_k^l and q_k^l approximations of $c(x_k, t_l)$ and $q(x_k, t_l)$. Taking initial conditions into account, we define $c_k^0 = 0$, $q_k^0 = 0$ for $k = 0, \ldots, n$.

3.4.1 Explicit scheme

First we set up an explicit scheme for solving the problem (3.4)-(3.7). The main idea behind any explicit scheme is to calculate the value of an approximation at a later time from the value of an approximation at the current time. Hence, replacing the derivatives by corresponding finite-difference representations, we obtain following system of algebraic equations

$$\frac{c_k^{l+1} - c_k^l}{\Delta t} = \frac{1}{\epsilon}\frac{c_{k+1}^l - c_k^l}{\Delta x} - \frac{1-\epsilon}{\epsilon}\left(Q(c_k^l) - q_k^l\right) \tag{3.9}$$

$$\frac{q_k^{l+1} - q_k^l}{\Delta t} = Q(c_k^l) - q_k^l, \tag{3.10}$$

for $l = 0, \ldots m-1$ and $k = 0, \ldots n-1$. For $k = n$ we have boundary condition $c_n^l = 1$ from which we compute q_n^{l+1} using formula (3.10). This scheme is straightforward to implement due to the fact that we calculate the solution in the time step $l+1$ explicitly using the information we obtained in the step l. Hence, we get the following algorithm.

For $l = 0, \ldots, m-1$:

For $k = 0, \ldots, n-1$:

$$c_k^{l+1} = \frac{1}{\epsilon}\frac{\Delta t}{\Delta x}c_{k+1}^l + \left(1 - \frac{1}{\epsilon}\frac{\Delta t}{\Delta x}\right)c_k^l - \frac{1-\epsilon}{\epsilon}\Delta t(Q(c_k^l) - q_k^l),$$
$$q_k^{l+1} = (1-\Delta t)q_k^l + \Delta t Q(c_k^l),$$

For $k = n$:

$$c_n^{l+1} = 1, \quad q_n^{l+1} = (1-\Delta t)q_n^l + \Delta t Q(c_n^l).$$

The function Q is defined as $Q(c) = \frac{\rho}{c_{in}} K_F M W_i^{\frac{1}{n-1}} (c_{in} c)^{\frac{1}{n}}$. Now we describe disadvantages of this method. Recall that we want $c_k^{l+1} \geq 0$, since c represents concentration. In order to achieve this, we rewrite the equation (3.9) as

$$c_k^{l+1} = \frac{1}{\epsilon}\frac{\Delta t}{\Delta x}c_{k+1}^l + c_k^l\left(1 - \frac{1}{\epsilon}\frac{\Delta t}{\Delta x} - \frac{(1-\epsilon)}{\epsilon}\Delta t K_F M W^{\frac{1}{n-1}} {c_k^l}^{\frac{1}{n}-1}\right) + \frac{1-\epsilon}{\epsilon}q_k^l. \tag{3.11}$$

Therefore, in order to ensure that c_k^{l+1} stays non-negative, we derive condition on Δt:

$$\Delta t \leq \frac{1}{\frac{1}{\Delta x \epsilon} + \frac{1-\epsilon}{\epsilon} K_F M W^{\frac{1}{n-1}} (c_k^l)^{1/n-1}} \tag{3.12}$$

Notice that $c_i^k \ll 1$ and $1/n - 1 < 0$. Therefore, Δt must be very small in order to ensure $c_k^{l+1} \geq 0$, which represents a numerical obstacle that is hard to overcome. Therefore, we turn our attention to the implementation of implicit and semi-implicit methods which do not have such a strict requirement on the smallness of Δt.

3.4.2 Implicit scheme

In contrast to the explicit scheme, an implicit method finds a solution by solving an equation involving both the current state of the system and the next one. Replacing the right-hand-sides in the system (3.1) with the corresponding finite-difference approximations, we get the following scheme:

$$\frac{c_k^{l+1} - c_k^l}{\Delta t} = \frac{1}{\epsilon} \frac{c_{k+1}^{l+1} - c_k^{l+1}}{\Delta x} - \frac{1-\epsilon}{\epsilon} \left(Q(c_k^{l+1}) - q_k^{l+1} \right) \tag{3.13}$$

$$\frac{q_k^{l+1} - q_k^l}{\Delta t} = Q(c_k^{l+1}) - q_k^{l+1}, \tag{3.14}$$

for $l = 0, \dots m-1$ and $k = 0, \dots n-1$. If we denote with $\mathbf{c}^l$ and $\mathbf{q}^l$ the vectors $\mathbf{c}^l = (c_1^l, \dots, c_{n-1}^l, 1)$ and $\mathbf{q}^l = (q_1^l, \dots, q_n^l)$, we get the following algorithm.

For $l = 0, \dots m-1$:

$$\mathbf{c}^{l+1} - \frac{\Delta t}{\epsilon} F(\mathbf{c}^{l+1}) + \Delta t \frac{1-\epsilon}{\epsilon} G(\mathbf{c}^{l+1}, \mathbf{q}^{l+1}) = \mathbf{c}^l, \tag{3.15}$$

$$\mathbf{q}^{l+1} - \Delta t G(\mathbf{c}^{l+1}) = \mathbf{q}^l \tag{3.16}$$

$$\tag{3.17}$$

where vector functions F and G are defined as

$$F(\mathbf{c}^{l+1}) = \begin{bmatrix} \frac{c_{k+1}^{l+1} - c_k^{l+1}}{\Delta x} \end{bmatrix} \tag{3.18}$$

$$G(\mathbf{c}^{l+1}, \mathbf{q}^{l+1}) = \begin{bmatrix} Q(c_k^{l+1}) - q_k^{l+1} \end{bmatrix} \tag{3.19}$$

Notice that $\mathbf{c}^{l+1}$ and $\mathbf{q}^{l+1}$ are vectors of dimension n. Therefore, in each time step the numerical solver has to find roots of the system of $2n-1$ coupled equations.

After implementing the method in MATLAB, the task of finding roots $(\mathbf{c}^{l+1}, \mathbf{q}^{l+1})$ of nonlinear equations (3.15) and (3.16) seemed to be too difficult, especially for values of $1/n$ close to 0. MATLAB function *fsolve* often resulted with the message that step size became too small and it could make no more progress. Hence, we turned our attention to developing and implementing a semi-implicit scheme.

3.4.3 Semi-implicit scheme

Semi-implicit schemes are a compromise between explicit and implicit numerical methods, which both suffer from its standard drawbacks. The explicit method requires very small time step in order to converge while an implicit method brings difficulties in the form of finding roots of highly nonlinear multivariable functions. One possible approach to tackle both problems is to treat some terms explicitly and the others implicitly. Since the requirement in explicit scheme on smallness of Δt comes from the nonlinear adsorption term Q, we have decided to treat that term implicitly in the calculations of the concentrations c_k. Hence, we get the following finite-difference approximation of (3.1).

$$\begin{aligned} \frac{q_k^{l+1} - q_k^l}{\Delta t} &= \frac{Q(c_k^l) - q_k^l}{\Delta x} \\ \frac{c_k^{l+1} - c_k^l}{\Delta t} &= \frac{1}{\epsilon}\frac{c_{k+1}^l - c_k^l}{\Delta x} - \frac{1-\epsilon}{\epsilon}\left(Q(c_k^{l+1}) - q_k^{l+1}\right), \end{aligned} \tag{3.20}$$

for $l = 0, \dots m-1$ and $k = 1, \dots n-1$. Therefore, we propose the following algorithm.

For $l = 0, \dots, m-1$:

For $k = 0, \dots, n-1$:

$$q_k^{l+1} = (1 - \Delta t)q_k^l + \Delta t Q(c_k^l), \tag{3.21}$$

$$c_k^{l+1} + \Delta t \frac{1-\epsilon}{\epsilon} Q(c_k^{l+1}) = \left(1 - \frac{\Delta t}{\epsilon \Delta x}\right) c_k^l + \frac{1-\epsilon}{\epsilon} q_k^{l+1} \tag{3.22}$$

For $k = n$:

$$c_n^{l+1} = 1, \quad q_n^{l+1} = (1 - \Delta t)q_n^l + \Delta t Q(c_n^l).$$

Notice that we have reversed the order of equations. First we calculate q_k^{l+1} using an explicit forward finite-difference formula. Then we find the value c_k^{l+1} as a root of a nonlinear equation (3.22).

3.5 Numerical experiments

In this section, we shall present results from numerical simulations, corresponding to parameter values shown in table 3.1, that led to difficulties in the current version of KWR's software. The numerical results are based on an implementation of the semi-implicit scheme (3.20) that we derived in the previous section. The numerical domain in non-dimensional units is defined by $x \in [0, 1]$, $t \in [0, 150000]$. The latter corresponds to a simulation of more than one year in the physical time domain. The space-discretization step is chosen to be $h = 0.1$ and the time-discretization step is $\Delta t = 1$. For all of the simulations we consider the following values of the model parameters, proposed by KWR water: $\gamma = 2.45 \times 10^{-12}$, $\epsilon = 0.4$, $\rho = 440000$ and $\nu = 0.144$.
We shall present results for two different compounds—natural organic matter (NOM) and a micropollutant, namely Furosemide. We believe that the difficulties in the numerical solution, if they exist, should be clearly visible even in the case of a single compound. Thus, we present results for this simple case. It can be further generalized by implementing IAST for multiple compounds.

Example 1. We choose parameter values $K_f = 0.018$ and $1/n = 0.9$, corresponding to NOM. Results for $c(t)$ and $q(t)$ are depicted in Fig.3.4 and in Fig.3.5, respectively. The rightmost edge (i.e., $x = 1$) corresponds to the top of the filter, where the inlet boundary condition is imposed, and $x = 0$ corresponds to the bottom.

As can be seen from the pictures, the qualitative behaviour of the numerical solution seems to be physically plausible, given the time the filter gets "exhausted", starting from the top and progressing to the bottom. Following this, the concentration of the contaminant in the water gets also gradually increased, until at the bottom the concentration gets equal to 1, thus, the filter is completely exhausted.

Example 2. In this second example, we use parameter values $K_f = 0.34608$ and $1/n = 0.0574$, corresponding to the micropollutant Furosemide. The corresponding results are presented in Fig.3.6 and Fig.3.7.

As can be seen, for the micropollutant it takes more time for the filter to get exhausted, which can also be expected.

Further experiments were conducted with various model parameters, provided by KWR water, corresponding to compounds that KWR water's software was not able to simulate. All the experiments ran successfully and showed similar qualitative behaviour. Thus, we omit them here in order not to make the presentation unnecessarily complicated.

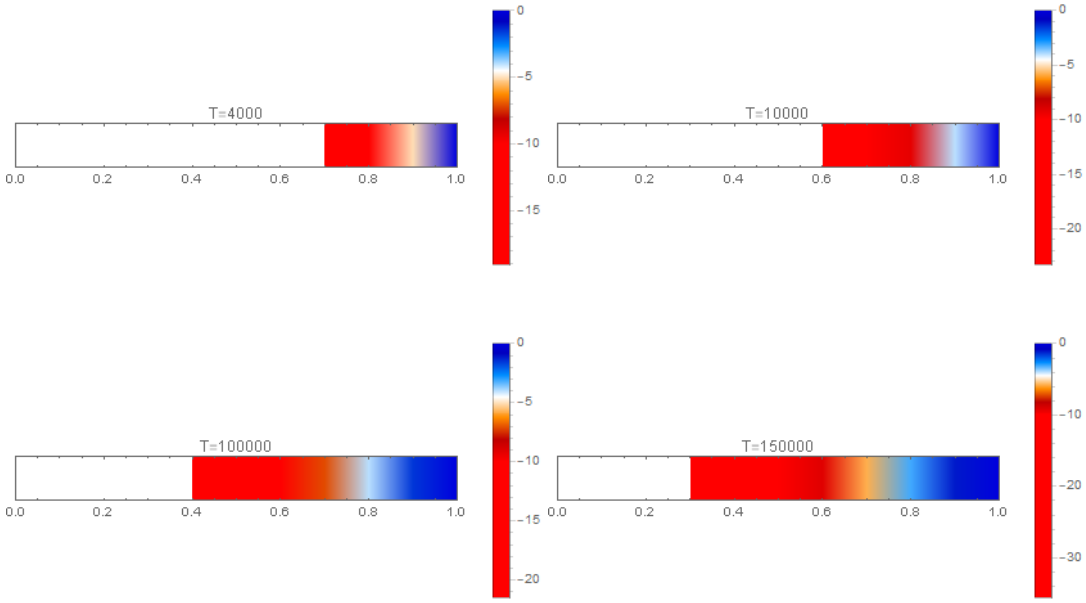


Figure 3.4: Results for the concentration c in log–scale, natural organic matter, for $T = 4000$, $T = 10000$, $T = 100000$ and $T = 150000$.

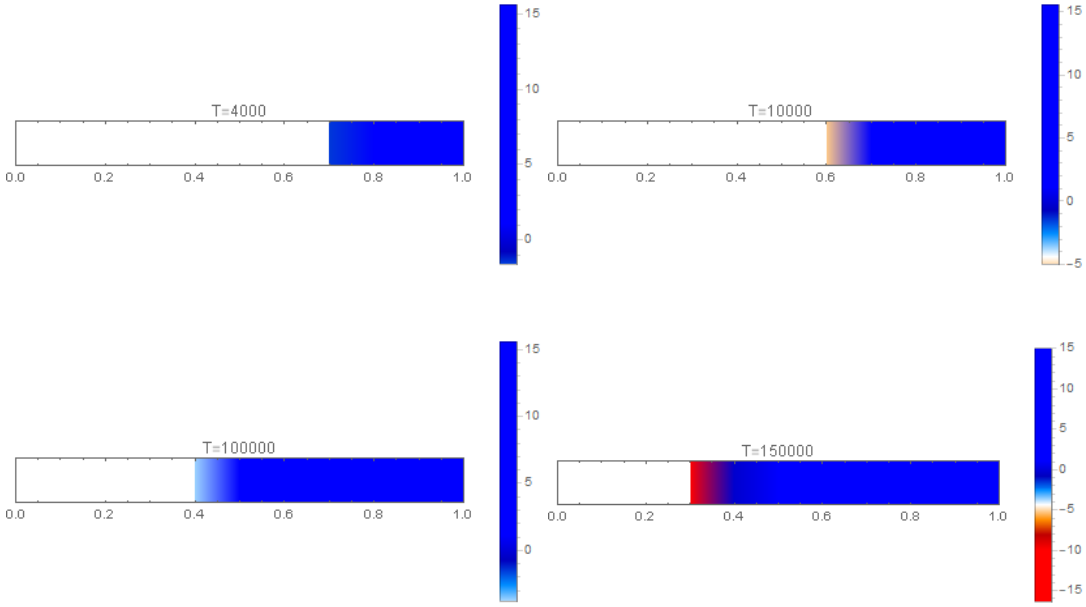


Figure 3.5: Results for the load q in log–scale, natural organic matter, for $T = 4000$, $T = 10000$, $T = 100000$ and $T = 150000$.

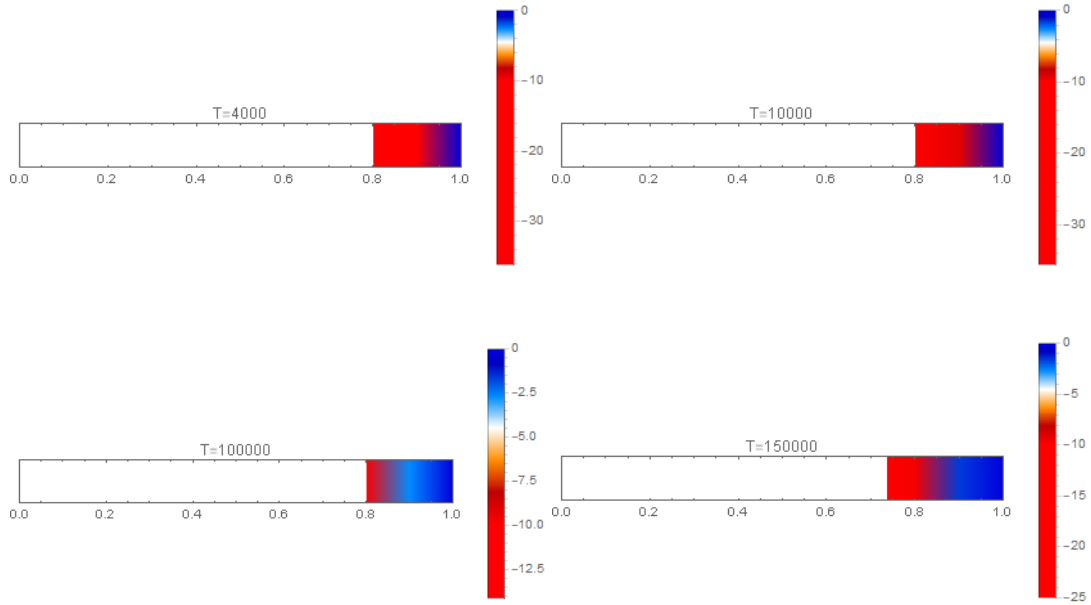


Figure 3.6: Results for the concentration c in log–scale, micropollutant, for $T = 4000$, $T = 10000$, $T = 100000$ and $T = 150000$.

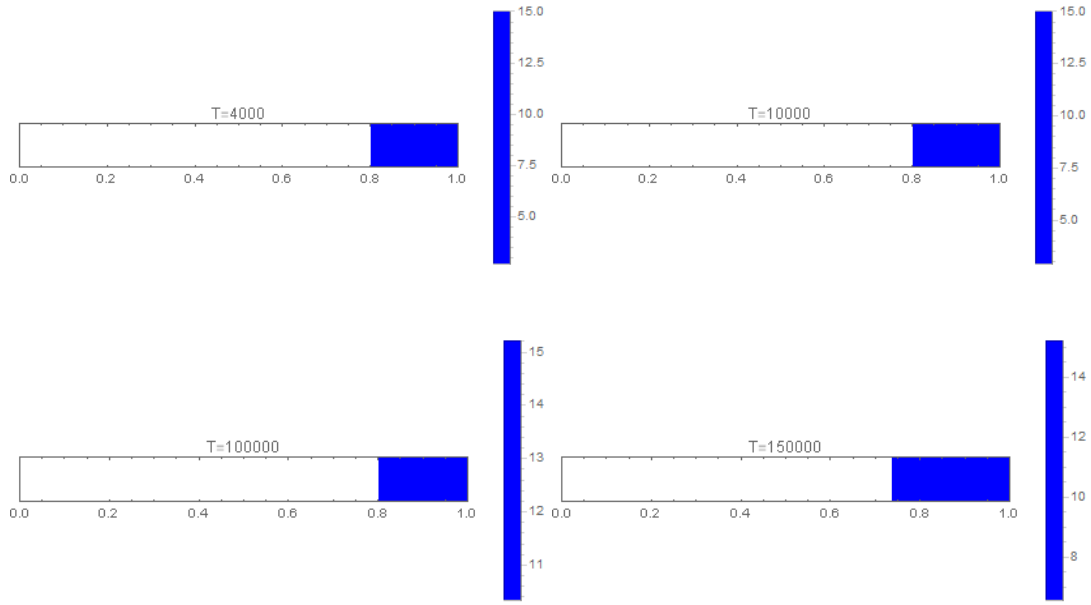


Figure 3.7: Results for the load q in log–scale, micropollutant, , for $T = 4000$, $T = 10000$, $T = 100000$ and $T = 150000$.

Concerning computational times, when ran on a PC in Wolfram Mathematica, the

simulations take about 15 min. Of course, the computations are expected to be much quicker, when the algorithm is implemented in a compiled programming language.

3.6 Conclusion and recommendations

In the present work, we have considered a well-known mathematical model, describing the process of filtering water by carbon filters. We have pointed out numerical difficulties that the company KWR water was facing, when trying to solve the corresponding system of PDEs. Thus, we have conducted the following steps that seem to give very good results towards solving the problem:

- We have rewritten the mathematical model in dimensionless quantities and hence, reduced the number of parameters;
- We have constructed a semi-implicit finite-difference scheme that seems to be solving the numerical stability problems in the present algorithms. Furthermore, we believe that the correct way to attack this problem is by implementing such an semi-implicit scheme. Many built-in ODE solvers, especially those using adaptive time steps, do not seem to be appropriate for this problem.

Due to the lack of time, we have carried out experiments that include only one compound in water. We suggest that the next step should be to include multiple compounds, using IAST. The generalization should be realtively easy. Further numerical experiments and comparison to empirical observations are needed, in order to validate the applicability of the proposed approach.

Bibliography

[1] Dirk Vries, Bas Wols, Martin Korevaar, Erwin Vonk, Aquapriori: a priori het verwijderingsrendement bepalen, 2017

[2] Jerry Perrich, Activated Carbon Adsorption for Wastewater Treatment, CRC Press, 1981

Chapter 4
Boosting Ship Simulations at Marin

David Kok[1], Vivi Rottschäfer[2], David van Keulen[3], George A.K. van Voorn[4], Manu Kalia[5], Bas van't Hof[2], L. Gerard van Willigenburg[4]

Abstract:
Two approaches to significantly reduce ship model simulation times at Marin *are presented. Combining them is shown to provide a most interesting and general perspective for improvement. One approach makes use of a highly simplified ship model that can be partly solved analytically. The other applies proper orthogonal decomposition (POD) to reduce the ship model currently used at* Marin*. POD is demonstrated using a model that is more convenient for presentation and implementation than the* Marin *ship model. Arguments are provided why POD is expected to also work for* Marin *ship simulations.*

Keywords: *Singular Value Decomposition, Proper Orthogonal Decomposition, Model Order Reduction, Method of Averaging, Asymptotic approximation*

[1]Mathematical Institute Leiden, The Netherlands
[2]Leiden University, The Netherlands
[3]Fontys Hogeschool, The Netherlands
[4]Wageningen University, The Netherlands
[5]Twente University, The Netherlands

4.1 Introduction

Certain ship simulations performed at MARIN solve a detailed model containing Navier-Stokes type dynamics using computational fluid dynamics (CFD). Currently, these are computationally too expensive. This problem relates to the fact that ship speed is varying very slowly compared to the rotation of the propeller and surrounding water, leading to very different time-scales in the simulation. The challenge is to improve computational efficiency. This will enable valuable ship simulations in regards to improving the design of ships as well as the energy efficiency of steering ships. In studying this challenge we took two approaches that in the end can be combined. For one approach, we developed a highly simplified ship model that reveals the origin of the computational inefficiency. In the other approach, we started from the current computational fluid dynamics computations and searched for ways to improve their efficiency. One method that can be used is proper orthogonal decomposition (POD) [6]. This will provide a way to reduce the model thereby improving computational efficiency. In this paper we first present the highly simplified model revealing the essence of the inefficiency of the ship simulations in Section 4.2. In Section 4.5, POD is presented and applied, not directly to the ship models of MARIN, but to a model that is more suitable for presentation and implementation. Arguments are given why we expect this method to be applicable to the ship models of MARIN as well. In the conclusions we argue that combining the outcomes of both approaches results in recommendations for improving the computational efficiency that are most promising and practical.

4.2 Highly simplified ship model

In this section we present a highly simplified model describing the speed of a ship, driven by a high-frequency propeller in water, near cruising speed. This model provides a computationally very cheap alternative to a full numerical solution of partial differential equations (PDE's) describing water, propeller and ship, while still preserving some of the observed features of the long-term behaviour of the solution.

We model the dynamics of the ship by using Newton's second law, which states that the time derivative of the speed v is given by

$$\frac{dv}{dt} = \frac{1}{m} F_{\text{total}} = \frac{1}{m}(F_{\text{prop}} - F_{\text{water}}) \tag{4.1}$$

with F_{total} the total force acting on the ship. This force consists of two components, a

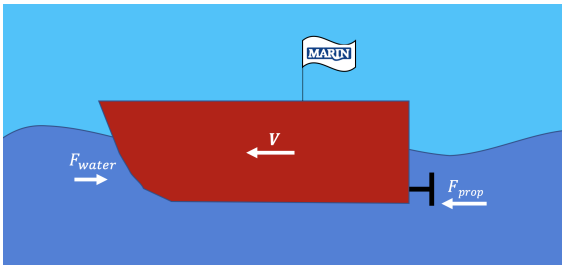


Figure 4.1: Highly simplified model of ship movement.

frictional force F_{water} exerted on the ship by the surrounding water, and the driving force F_{prop} delivered by the propeller. We rescale and thereby set $m = 1$. For the friction force we assume a simple expression consisting of a linear and a quadratic component

$$F_{\text{water}}(v) = p_5 v + p_6 v^2, \tag{4.2}$$

where p_5 and p_6 are positive constants. We note that the propeller force depends on the pressure of the water near the bow of the ship (averaged over the surface of the propeller). When the ship speeds up, this pressure increases, allowing the propeller to deliver more force. In general this means we model the net effect of the water pressure on the propeller as $p = p(v, t)$. Here it should be noted that we need the explicit time dependence of this pressure term. It represents the fact that the ship generates water currents near the bow which the propeller slices through, changing the total pressure across the blades as a function of the angle of the blade.

We expect the behaviour of the ship to be sensitive to the precise interplay between the water currents near the propeller and the movement of the ship itself. To account for this interplay, we model the flow of the water with two components - one rotating along with the propeller (the component generated by the propeller itself) and one stationary (the time-averaged flow generated by the ship moving at cruising speed). In general we expect the water flow profile, and the full long-term solution, to be periodic with a period equal to that of the propeller. Therefore we will use a first-harmonic approximation of the full interplay between water flow and propeller force. This is described by

$$F_{\text{prop}}(v, t) = \left(1 + \frac{1}{2} p_1 \cos\left(\omega t/\epsilon\right)\right)(p_2 + p_3 v). \tag{4.3}$$

Here ϵ is a small parameter introduced to explicitly indicate the occurrence of two timescales. The rotation frequency ω/ϵ is much higher than the typical timescale on which we expect the speed of the ship to change. Furthermore p_2 represents the average water pressure and p_3 the rate of change of water pressure with respect to velocity.

Finally p_1 determines the relative contribution of the two types of water flow near the propeller to the propulsion. All these constants are positive. The full model is then given by

$$\frac{dv}{dt} = \left(1 + \frac{1}{2}p_1 \cos(\omega t/\epsilon)\right)(p_2 + p_3 v) - p_5 v - p_6 v^2. \tag{4.4}$$

4.3 Approximate analytical solutions

4.3.1 Ansatz

Based on numerical simulations of equation (4.4) we expect the solution to consist of a quick oscillation around a slowly varying average, where the amplitude of the oscillation also varies slowly. Furthermore, the amplitude of this oscillation is small.
The standard argument of perturbation theory suggests that, for sufficiently small ϵ, the solution $v(t;\epsilon)$ of equation (4.4) is analytic in ϵ and can therefore be expressed as

$$v(t;\epsilon) = \sum_{n=0}^{\infty} \epsilon^n v_n(t). \tag{4.5}$$

We also expect the solutions of 4.4 to converge to periodic solutions with the same periodicity as the driving term. Restricting to periodic solutions implies that $v(t+T;\epsilon) = v(t;\epsilon)$ for all t, ϵ where $T = \frac{2\pi}{\omega/\epsilon} = \frac{2\pi\epsilon}{\omega}$ is the period of the driving term. However, if this holds for all ϵ, then for fixed t the power series expressions for $v(t;\epsilon)$ and $v(t+T;\epsilon)$ must be identical. Therefore all functions $v_n(t)$ in equation (4.5) above are also periodic with period T. Introduction of a Fourier expansion allows us to write

$$v(t;\epsilon) = \sum_{n=0}^{\infty} \sum_{k=-\infty}^{k=\infty} \epsilon^n v_{n,k} \exp(ki\omega t/\epsilon). \tag{4.6}$$

Numerical simulations show that at order ϵ^0, i.e. leading order, the solution is not oscillating rapidly, so $v_{0,k} = 0$ for $k \neq 0$. This leads to the following Ansatz/approximation

$$v(t) = a(t) + \epsilon b(t)E + \epsilon\overline{b(t)}\bar{E} + O(\epsilon^2, \epsilon E^2) + \text{c.c.} \tag{4.7}$$

where we used the big-O notation and introduced $E = \exp(i\omega t/\epsilon)$ while c.c. denotes complex conjugation. Substituting (4.7) in equation (4.4) gives

$$\begin{aligned}\frac{dv}{dt} &= \frac{da}{dt}(t) + \left(\epsilon\frac{db}{dt}(t)E + i\omega b(t)E\right) + O(\epsilon^2, \epsilon E^2) + c.c. \\ &\quad - p_6(a(t) + O(\epsilon))^2 + c.c. \\ &= \left(p_2 + p_3 a(t) - p_5 a(t) - p_6 a(t)^2\right) + \left(\frac{1}{2}p_1(p_2 + p_3 a(t))\right) E \\ &\quad + O(\epsilon) + c.c.\end{aligned}$$

Collecting terms in orders of ϵ and per frequency we find

$$\mathcal{O}(\epsilon^0 E^0): \; p_2 + (p_3 - p_5)a - p_6 a^2 = \frac{da}{dt} \tag{4.8}$$

$$\mathcal{O}(\epsilon^0 E^1): \; \frac{1}{2}p_1(p_2 + p_3 a) = i\omega b \tag{4.9}$$

where we have suppressed the explicit time dependence from our notation.

4.3.2 Asymptotic behaviour

The first-order ODE (4.8), has an attracting fixed point

$$a_{\text{lim}} = \frac{p_3 - p_5 + \sqrt{(p_3 - p_5)^2 + 4p_2 p_6}}{2p_6}. \tag{4.10}$$

There is also a second fixed point at a negative value of a, which is not only unphysical but also repelling. The solution of (4.8) converges to the positive fixed point. In the $t \to \infty$ limit we find from (4.9) that

$$b_{\text{lim}} = -i\frac{\frac{1}{2}p_1(p_2 + p_3 a_{\text{lim}})}{\omega} \tag{4.11}$$

Some additional remarks are in order:

- The limiting amplitude b_{lim} is purely imaginary, which represents that the movement of the ship is a quarter period out of phase with the driving force. For the actual amplitude of the oscillations around the average we can simply take the modulus. Note that since our Ansatz adds the complex conjugate of the oscillating terms the full solution is still real-valued.

- We have ignored all higher-order terms. So $\overline{v}(t) \equiv a(t) + \epsilon b(t)E + c.c.$ is only an approximation of the real solution. We expect this to be accurate only if ϵ is small, i.e. if $T = \frac{2\pi\epsilon}{\omega}$ is small.

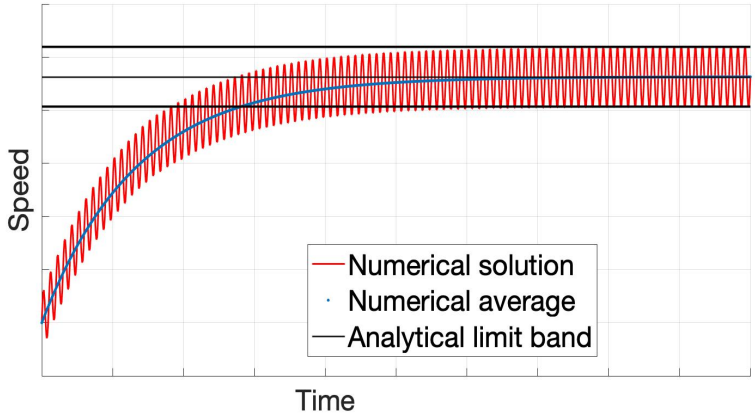


Figure 4.2: A simulation of equation (4.4) *in red with a short-time average in blue. The upper and lower black lines denote* $a_{lim} \pm b_{lim}$ *respectively, with* a_{lim} *plotted in between. The simulation parameters are* $p_1 = 1, p_2 = 1, p_3 = 0.5, T = 0.1, p_5 = 1$ *and* $p_6 = 0.1$.

4.3.3 Comparing numerical results and analytical approximations

In this section we compare the approximations given above with direct numerical integration of equation (4.4). To that end we use the MATLAB ode23s-solver (see the *Appendix* for the code) for a duration of $100T$. In Fig. 4.2 we compare the numerical results with the constants a_{lim} and $a_{\text{lim}} \pm b_{\text{lim}}$ computed with the expressions given above

We also compute the analytical limit parameters and the numerical asymptotic behaviour for a range of values of p_1 fixing the values of $p_2, \ldots, p_6$ and find that the two are in good agreement (see Table 4.1). In Table 4.2 similar results are given but now for various values of p_2.

The tables confirm that Ansatz is quite good. This suggests that the method of averaging might be an effective approach to simplify the full PDE model. This suggestion is explored in the next section.

p_1	a_{lim}	numerical	b_{lim}	numerical
0.1	1.53	1.54	0.0028	0.0075
0.2	1.53	1.55	0.0056	0.0069
0.3	1.53	1.55	0.0084	0.0095
0.4	1.53	1.54	0.011	0.012
0.5	1.53	1.55	0.014	0.004
0.6	1.53	1.54	0.017	0.018
0.7	1.53	1.55	0.020	0.020
0.8	1.53	1.53	0.022	0.024
0.9	1.53	1.54	0.025	0.026
1.0	1.53	1.54	0.028	0.028
1.1	1.53	1.54	0.031	0.031
1.2	1.53	1.56	0.034	0.034
1.3	1.53	1.54	0.037	0.036
1.4	1.53	1.56	0.039	0.040
1.5	1.53	1.55	0.042	0.042

Table 4.1: Comparison of simulated and analytical limit (after $100T$) behaviour of equation (4.4)*. The simulation parameters are $p_2 = 1, p_3 = 0.5, T = 0.1, p_5 = 1$ and $p_6 = 0.1$ where p_1 varies.*

p_2	a_{lim}	numerical	b_{lim}	numerical
0.1	0.19	0.20	0.003	0.003
0.2	0.37	0.38	0.006	0.006
0.3	0.54	0.54	0.009	0.009
0.4	0.70	0.70	0.012	0.012
0.5	0.85	0.86	0.015	0.015
0.6	1.00	1.01	0.018	0.018
0.7	1.14	1.15	0.020	0.020
0.8	1.27	1.29	0.023	0.023
0.9	1.41	1.42	0.026	0.026
1.0	1.53	1.54	0.028	0.028

Table 4.2: Comparison of simulated and analytical limit (after $100T$) behaviour of equation (4.4)*. The simulation parameters are $p_1 = 1, p_3 = 0.5, T = 0.1, p_5 = 1$ and $p_6 = 0.1$ where p_2 varies.*

4.4 The 'Method of averaging'

Here we show that the results obtained above are identical to those that would be obtained using the method of averaging [4]. This method can be applied to systems of the form

$$\frac{dx}{dt} = \epsilon f(x,t), \tag{4.12}$$

where $x \in D \subset \mathbb{R}$ for some domain D and f and $\frac{\partial f}{\partial x}$ are continuous and bounded in $D \times [0,\infty)$. The function f must be T-periodic in t and T independent of ϵ. Then, upon introducing the averaged system

$$\frac{dy}{dt} = \epsilon \frac{1}{T} \int_0^T f(y,s)ds, \tag{4.13}$$

the Averaging Theorem states that

$$x(t) - y(t) = \mathcal{O}(\epsilon), \tag{4.14}$$

on the time-scale $\frac{1}{\epsilon}$ This averaging method can be applied to the toy model of Section 4.2 by introducing the timescale $\tau \equiv \frac{t}{\epsilon}$. Then, equation (4.4) transforms to

$$\frac{dv}{d\tau} = \epsilon \left(\left(1 + \frac{1}{2} p_1 \cos(\omega\tau) \right) (p_2 + p_3 v) - p_5 v - p_6 v^2 \right) \tag{4.15}$$

Now taking the average over one period of the right-hand-side with v fixed, we remove the oscillating exponentials and this reduces the equation to

$$\frac{d\tilde{v}}{d\tau} = \epsilon \left(p_2 + (p_3 - p_5)\tilde{v} - p_6 \tilde{v}^2 \right). \tag{4.16}$$

Note that, to leading order, this is equal to (4.8). Since the method of averaging doesn't heavily rely on specifics of the dynamical system, and our analysis of the previous and upcoming sections suggest that the results obtained from this method are accurate, it is likely that we can apply this strategy to the full PDE model to obtain a simplified time-averaged model.

4.4.1 Method of averaging for Marin's ships

This section provides a rough sketch of how the method of averaging could be applied to the Marin models.

The Marin model of ship and water consists of a model of the flow around the propeller, coupled to a model of the flow around the ship, as shown in Figure 4.3. The propeller-model is a cylindrical, rotating domain, and the ship-flow model moves with the ship. Apart from their moving domains both models have constant geometry.

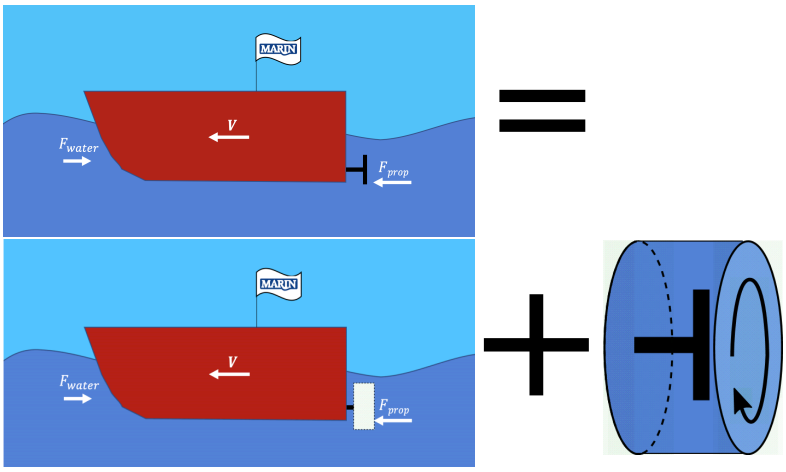


Figure 4.3: Decomposition of the Marin ship model into two coupled models having very different time-scales: one of the ship and water and one of the propeller and its surrounding water.

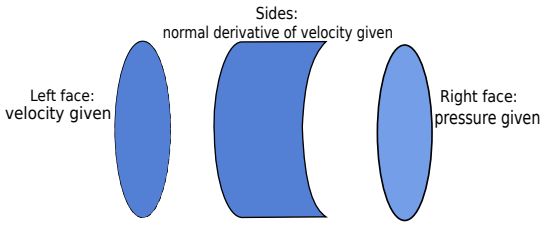


Figure 4.4: The propeller model can be solved for given boundary conditions along its cylindrical domain.

Before applying the averaging-method, we replace the propeller-model by a forcing term. To do this, a set of boundary conditions is chosen, as shown in Figure 4.4. Each of the chosen boundary conditions is constant in time in the ship-flow model, and is therefore periodic in the (rotating) propeller model. Next, the flow equations are solved in the propeller model until the solution is (almost) periodic. When the solution is (almost) periodic, the solution is recorded at the boundary and time-averaged over a period. After the chosen boundary conditions have been processed this way, and the results stored in a table, interpolation is used to approximate the solution on the boundary of the propeller-model for any given boundary condition. Now, the propeller model can be replaced by the interpolation tables thus obtained, which is a time-averaged forcing without the small time scale inherent to the full propeller model.

The interpolation tables obtained this way are very similar to the *actuator disc* used

at Marin to obtain initial estimates of the water flow around a ship. Like the interpolation tables, the actuator disc is a forcing term without a short time scale, that provides the thrust necessary to propel the ship. However, the approach roughly sketched in this section produces a forcing term directly based on the model of the propeller used in reality (including all the details of cavitions etcetera), and is expected to be much more accurate than the actuator disc.

4.4.2 Introducing multiple time-steps in ship simulation

During full-scale simulations, MARIN noticed that the solution changes rapidly in only approximately 30% of all the grid-points.

This observation suggests to not update all model states at the same rate. Instead, the computational domain should be split in a 'fast changing' part and a 'slowly varying' part, so that a larger time step could be applied in the 'slowly varying' part of the domain.

4.5 Model order reduction using POD

Since computational fluid dynamics (CFD), used by MARIN for ship simulation, is too computationally expensive, in the last decade or so there is an increased interest in producing *Reduced Order Models* (ROMs). These reduced models are intended to keep the dominant flow dynamics while reducing the size and hence the computational costs of fluid dynamics models, including RANS models and large Eddy simulations.

In this Section we discuss model order reduction based on the method of POD (Proper Orthogonal Decomposition; [6]). Let the dynamics of the system be described by

$$\dot{x} = f(x) \ , \ x \in \mathbb{R}^N \ , \tag{4.17}$$

where x is the vector with continuous state variables, $f(\cdot)$ are smooth functions, and $N \approx 10^8$ in the case of the MARIN model. Numerical simulation of the full model is costly, and therefore the simulations done so far only cover the small time period of transient behaviour from initial conditions. From the transient behaviour stage of the simulation snapshot data can be taken, put into the matrix

$$W = (w_1 \cdots w_M) \ , \tag{4.18}$$

of dimensions $N \times M$, where w_i are the vectors containing the snapshot data, and M is the number of snapshots. Keep in mind that each snapshot vector has the same length as x (namely N) and hence represents several GB of data, while $M \lll N$.

The matrix W will now be used for singular value decomposition (SVD) as a basis for linearisation. The most accurate approximation would be to use the SVD

$$W = U\Sigma V^* \,, \tag{4.19}$$

where U is a unitary $N \times N$ matrix, V^* is the conjugate transpose of a unitary $M \times M$ matrix, and Σ is a diagonal $N \times M$ matrix containing non-negative real numbers, the so-called singular values. In practice there will be, say, 600 snapshots, conveniently chosen at equidistant time intervals to cover a time span with information-rich model dynamics. This leads to a matrix of about 4000 GB. As a result, the matrices U and Σ would be large also. The singular values can also be obtained as the eigenvalues of the matrix

$$R = W^T W \,, \tag{4.20}$$

which has dimensions $M \times M$, and hence is much less costly to handle, at the price of losing some accuracy. There are standard packages available for extracting the eigenvalues λ_i of matrix R, where $i = 0, 1, \ldots, M$. In SVD the singular values are ordered from large to small. For POD we select only the top-ranking singular values, indicated by index j, where $j = 0, 1, \ldots, P$, with $P \ll M$. The cut-off will be determined by the orders of magnitude the eigenvalues differ. Below we will demonstrate for a test case that the cut-off can potentially be very low, resulting in a reduction where $P \lll M \lll N$. For instance, [5] reduced a Navier-Stokes model for driven cavity flows with high Reynolds number using POD, and found reasonable approximations using the first 20 singular values. For the MARIN model, it remains to be seen what an acceptable P could be.

The Galerkin projection of the full model does not yet include the nonlinear terms, and the approximation of the nonlinear terms still depends on N though. To complete the model order reduction, the nonlinear terms need to be approximated as well. Several techniques are discussed in the literature to estimate the nonlinear terms. Among them, precomputing techniques and empirical interpolation methods. See for instance [2] for a modern treatment of POD.

In general, the scheme for model order reduction is as follows:

- Convert the non-linear PDE to a full, discretized system based on ODEs with dimension N, and solve this system to produce snapshots (this step has already been performed by MARIN);
- Take snapshots from the simulation data;
- Determine the POD basis using SVD;

- Use Galerkin projection to produce the reduced discretized system consisting of ODEs, with linear terms of dimension $M \ll N$ but non-linear terms still of dimension N. This saves memory and enhances the accuracy;
- Use a method like pre-computing to approximate the non-linearities and produce a reduced discretized system of ODEs with linear terms of dimension $M \ll N$ and non-linear terms of dimension $Q \ll N$. This improves efficiency, i.e., it saves time.

4.6 Numerical experiment using the Lorenz 96 model

We use the Lorenz 96[6] model (see [3]) as a test model to evaluate the plausibility and feasibility of the approach. The model is given as

$$\dot{X}_i = -X_{i-2}X_{i-1} + X_{i-1}X_{i+1} - X_i + F \,, \tag{4.21}$$

where $i = \{1, 2, 3 \dots N\}$ for arbitrary N and F is some forcing. For the testing, we assume $N = 999$. We know that for $F = 1.2$, the system exhibits a stable periodic orbit, which we plot in Figure 4.5 **A**. Using POD, we construct a reduced order model (ROM) of dimension $k = 10 \ll N$ to reconstruct this periodic solution. In Figure 4.5 **B**, we show the periodic solution exhibited by the ROM, which is qualitatively very similar to that of the original system. In Figure 4.5 **C**, we plot the 2-norm of the state variables for every time step. We see that the reconstructed solution stabilizes quickly to a periodic regime with constant 2-norm, as exhibited by the original Lorenz-96 system.

This experiment demonstrates that the POD approach significantly reduces model order when we reconstruct periodic solutions. As general polynomial systems can be reduced to quadratic systems using lifting transformations (see [1]), we expect similar behaviour in other smooth non-linear systems as well.

[6]Note that the model was never published by Lorenz in 1996, but was finally presented at the ECMWF 2006 meeting on predictability.

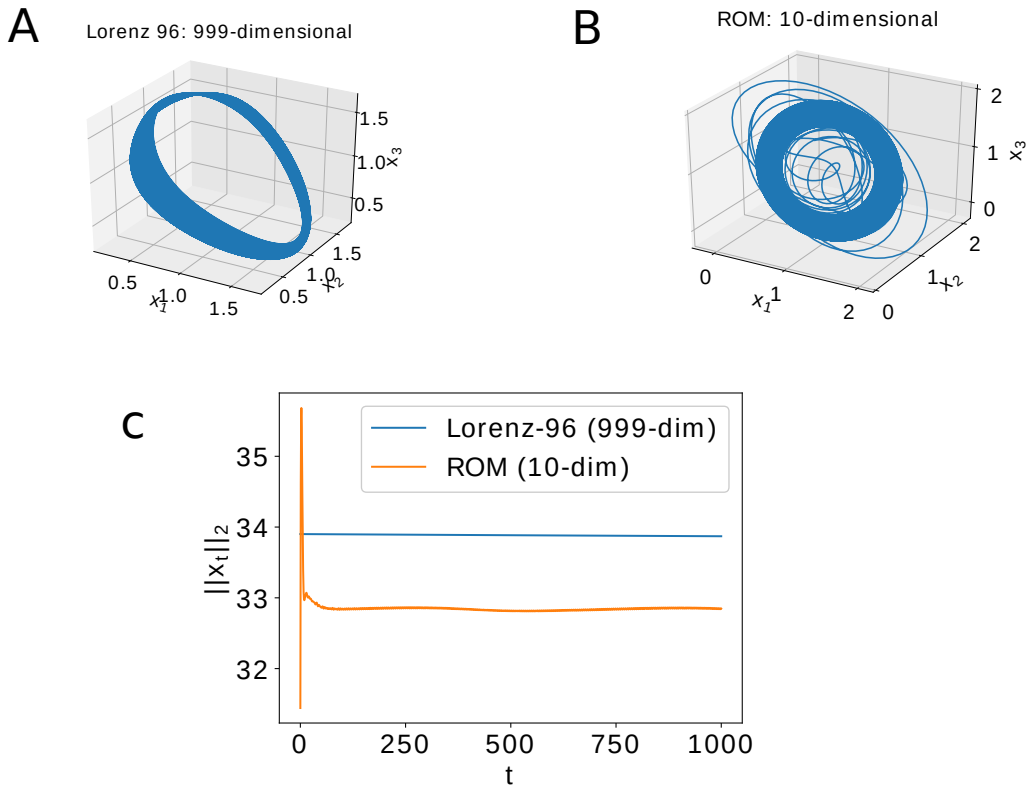


Figure 4.5: Demonstration of POD-based model order reduction applied to the Lorenz-96 model. The parameter $F = 1.2$ for these simulations. In ***A****, we see the stable periodic solution corresponding to the Lorenz-96 system with $N = 999$. In* ***B****, we plot the projection of the 10-dimensional ROM on the original coordinates. Thus we see a reconstruction of the periodic solution via a reduced order model. In* ***C****, we plot the 2-norm of the state variables at each time step.*

4.7 Conclusions and recommendations

A highly simplified one-state model describing ship velocity has been developed. This model still captures the major features, being fast rotation of the ship propeller and surrounding water on the one hand, versus slow changes of ship speed on the other. Approximate analytic solutions of this model were verified numerically and show that 'the method of averaging' may be applied. When applied to computational fluid dynamics this method recommends to not update all model states at the same rate. Instead, fast states associated with the propeller and surrounding water should be updated at a high rate, whereas updating the other states should be performed at a much slower rate. In this manner computational effort is seriously reduced since the fast states make up approximately 30% of all the states according to MARIN. Alternatively a 'time av-

eraged propeller model' might be developed, that is updated at the same slow rate as the remaining part of the model. Model reduction of the full-state model by means of proper orthogonal decomposition (POD) was also investigated and applied to the Lorenz 96 model that is quite suitable for presentation and implementation. POD is expected to work well since it essentially reduces the model by also averaging fast states appearing in the model. Successful application of this method does require an empirical interpolation method to approximate non-linear terms in the model. As to POD we therefore recommend to further investigate this promising approach.

Bibliography

[1] Kramer and Willcox, Nonlinear Model Order Reduction via Lifting Transformation and Proper Orthogonal Decomposition, arXiv:1808.0208v2[cs.NA], 2019

[2] Brenner, P., Cohen, A., Ohlberger, M., and Willcox, K., Model reduction and approximation – Theory and algorithms, SIAM Computational Science and Engineering, 2017

[3] Lorenz, E.N., Predictability – A problem partly solved, Editors: Palmer, T., Hagedorn, R., Cambridge University Press, 2006

[4] Verhulst, F., Methods and applications of singular perturbations: Boundary layers and multiple timescale dynamics, Springer Science & Business Media, 2005

[5] Cazemier, W., Verstappen, R.W.C.P., Veldman, A., Proper orthogonal decomposition and low-dimensional models for driven cavity flows, Physics of Fluids, 1998

[6] Lumley, J.L., The structure of inhomogeneous turbulence. Atmospheric turbulence and wave propagation. Editors: Yaglom, A.M., Tatarski, V.I., 1967

Appendix

4.A Matlab code to simulate highly simplified Marin ship model

```
% simship: simulations highly simplified MARIN ship model
%
% Programmer: GvW @ SWI2019-MARIN
clear; close all; clc; tb=[];
% Generate table from ship simulations with different p2 values
for p2=0.1:0.1:1
  T=0.1; p=zeros(6,1); p=[1;p2;0.5;1/T;1;0.1];
  if isinf(T); T=0.1; end
  ta=0:0.1*T:100*T; % Time axis
  x0=[0.15*p(1)/p(3);0.15*p(1)/p(3)]; % Initial state
  % Numerical integration
  [t,x]=ode23s(@(t,x)dnship(t,x,p),ta,x0);
  lx=size(x,1); lxm=round(lx/10); % 10% of response
  % Terminal & limiting values
  mblim=0.5*(max(x(end-lxm:end,1))-min(x(end-lxm:end,1)));
  malim=sum(x(end-lxm:end,1))/lxm;
  alim=(p(3)-p(5)+sqrt((p(3)-p(5))^2+4*p(2)*p(6)))/(2*p(6));
  blim=p(1)*(p(2)+p(3)*alim)*T/(2*pi);
  dmblim=mblim-blim;
  % Plot two velocity responses
  figure(1); plot(t,x(:,1),'-',t,x(:,2),'.');
  hold on;
  % Plot model terminal value a(tf)
  line([ta(1),ta(end)],[alim,alim]);
   %Plot model max(v)
  line([ta(1),ta(end)],[x(end,1)+blim,x(end,1)+blim]);
   %Plot model min(v)
  line([ta(1),ta(end)],[x(end,1)-blim,x(end,1)-blim]);
  hold off;
  % Display alim,model alim,blim,model blim
  disp('     alim        model alim      blim        model mblim')
  disp([alim,malim,blim,mblim])
  % Extend table
  tb=[tb; p2,alim,malim,100*(malim-alim)/alim,blim,mblim,...
  100*(mblim-blim)/blim];
  pause
end
tb % Show table

```

```
function [f]=dnship(t,x,p)
% State-space representation highly simplified MARIN ship model
% [f]=dnship(t,x,p)
%
% t,x,p : time, state & parameter vector
% f     : state derivatives
%
% Programmer: GvW @ SWI2019-MARIN
Fp1=(1+p(1)*cos(2*pi*p(4)*t))*(p(2)+p(3)*x(2)); % Propeller force
Fp2=(p(2)+p(3)*x(2)); % Average propeller force
Fw=p(5)*x(2)+p(6)*x(2)*x(2); % Water force
f=[Fp1-Fw; Fp2-Fw]; % State & averaged state derivative
```

Chapter 5
Synopsys: Latency Prediction for On-Chip Communication

Xingang Cao[1], Kristof Cools[2], Nhung Dang[1], Yingjun Deng[1], Tieme Goedendorp[3], Anastasiia Hraivoronska[1], Amir Parsa Sadr[1], Mikola Schlottke[1], Oliver Sheridan-Methven[4], Niels van der Wekken[5], Harshit Bansal[1]

Abstract:
Chip design is a complicated and multi-step process. After the functional description is translated in a concrete semi-conductor network, the components comprising this network need to be placed on the actual chip. The challenge not only lies in fitting a very large amount of components in a relatively small area, but also in ensuring that the electrical latency of components and links does not jeopardise the target clock frequency. In this report, techniques for latency estimation are described that will allow the placement algorithm to make well informed decisions as to where to place components, even before the final links and routes have been decided on. The methods described here are combination of classification of components based on their type and connectivity, and a data driven approach for the identification of underlying patterns. The methods will be tested on real life data and compared against the current state-of-the-art. The estimation accuracy in the root-mean-square error shows promising results.

[1]Eindhoven University of Technology, The Netherlands
[2]Delft University of Technology, The Netherlands
[3]Fontys Hogeschool, The Netherlands
[4]Oxford University, UK
[5]VOR Tech, The Netherlands

5.1 Introduction

5.1.1 Outline

The structure of this report is as follows:

- Description of background and problem statement: this summarises the problem at hand. This part is the result of a shallow literature study, our internal discussions, and most importantly the input and feedback we received from SynopSys.
- The Challenge: stripped down version of what we actually should achieve, and a short outline of the various approaches we explored.
- Methodology: provides short descriptions for each strategy implemented. For each strategy, the premise and assumptions are detailed, the implementation presented, as the success (or lack thereof) reported on.
- Conclusion: summarise the advantages and disadvantages of each method, compare to the state-of-the-art. Recommend for the future directions.

5.2 The Challenge

Synopsys is one of the main suppliers of Electronic Design Automation tools. It uses complex software to design digital *integrated circuits* (ICs), also known as *chips*. There are multiple steps involved to makes sure that these can operate at the desired clock frequency.

Currently a multiple step simulation protocol is in-place, starting from a functional description of the chip, and resulting in a concrete physical design, ready to be submitted for fabrication.

Chips within a single generation are designed and built according to a certain micro-architecture. The architecture prescribes the semi-conductors used, the manufacturing capabilities and tolerances, etc.

A physical layout implementing the target functionality needs to be designed within the restrictions imposed by the micro-architecture. The design process is subdivided in a number of steps, each lowering the level of abstraction. These are in order of execution:

- Floorplanning
- Synthesis
- Placement

- Electrical Optimisation
- Routing
- Mask preparation

The most optimal outcome to each of these steps depends on the other steps. This, however, would lead to an untractable problem. In addition, the above workflow is the part of the company and sector standard workflow and is shaped by the availability of tools, the training of experts, etc.

5.2.1 The Placement Algorithm

In this project we will focus on the placement step. In this step, the semi-conductor implementation of the logic gates, clocks, and other components are assigned placement on the board. A number of restrictions apply:

- Routability: because routes can only be created on a finite number of levels, it is possible that certain placements prohibit making the connections that are required. This is a hard constraint and should be avoided at all cost
- Electrical proximity: because routes between elements have a certain capacity, they can only respond so fast to any inputs applied to them. The longer the route, the slower the response time. This leads to de facto delays in transmission and in turn may affect the maximum clock frequency at which the device can run.

The placement algorithm can be further broken down into: *(i)* ensuring that the components do not overlap in the 2D plane, *(ii)* making sure the routing does not exhaust the number of levels in the substrate and the capacity of the via holes, *(iii)* minimizing the total length of the wires between the components, and *(iv)* meeting the targeted clock frequency. The placement step is an instance of a pretty large scale problem as the chips are generally composed of more than 10 million components and connections.

The work addressed in this report will only focus on meeting (or minimising the violation of) the targeted clock frequency. In order to achieve a specified clock frequency, the on-chip delays need to be estimated as accurate and as fast as possible by effectively utilizing the computational resources. On-chip delays could arise due to several reasons. Few of them are: *(i)* there could be a delay between a change at an input and the corresponding change at the output, *(ii)* there could be a delay due to the time taken by the signal to propagate across the wire.

The placement algorithm ensures routability, compatibility of placement with e.g., the size of components etc. In order to minimise delays, the algorithm needs more information. In fact, it would need to know the outcome of the actual routing algorithm. This in practice, is not a viable approach. Instead, a heuristic is required to predict the final delay caused by routing the elements at their respective positions. Using this information, a placement is arrived at that is compatible with the desired clock frequency.

The target clock frequency is only jeopardised if one of the routes creates a delay which is larger than the clock period. It is not necessarily a problem if the average delay is quite high, as long as it is under the clock period. In fact, it is desirable to allow this, as it provides the placement algorithm with more leeway. This will allow it to focus on the other target specifications. In terms of the heuristic, it means that in first instance it is more important to correctly guess the delay caused by connections that will likely be very close to the critical delay that is the clock period. Sinks at the end of such a connection are called *critical sinks*.

Because components are connected using routes that follow either vertical or horizontal lanes on the chip's substrate, it stands to reason that if a single distance measure should be chosen as the starting point, it will be the $l^1(R^2)$ norm $|(x, y)|_1 = |x| + |y|$. Because of it's central role in what follows, it is given the more imaginitive name *Manhattan distance*.

In pseudo-code, the placement algorithm looks like:

Algorithm 1:

```
decrease smoothness ;
adjust cost weighting factors ;
set the delay function (revisit) ;
for each conjugate gradient restart (∼ 5×): do
    for each conjugate gradient iteration (∼ 50×): do
        calculate gradients using the delay function ;
        perform one CG step ;
    endfor
endfor
```

We were given data sets from two different chip design technologies. Each data set constituted a collection of electronic networks. Each network comprises a single driver component and the sink components it is connected to.

Each component is specified as (x, y, r, f, s) signifying the position (x, y) of the component, the rise/fall time (relative to a global clock) and the slack, s, which is the excess

time a signal can take to reach a specified point in the flow path. The XML files provided by Synopsis as training and verification data also contained a field called LIBCELL for each component, which refers to its type, i.e. components with a different LIBCELL are of a different nature. It is not unreasonable that such different types are treated differently by the optimisation algorithms. This provided one of the starting points for the various classification strategies in what follows.

This available information can be used to compute the delay for every connection between a driver and a sink in a given network. It should be noted that the driver sink connection can be part of a longer chain. This implies that not in all cases the delay equals the clock period minus the difference between sink rise and driver rise.

The connection between a driver and a sink is dubbed critical if the slack is negative. In this report, we aim to arrive at an accurate heuristic to predict delays along critical paths. In practice this means that in our data-driven approaches non-critical connections will simply be filtered out.

The information on rise/fall time, and the recorded slacks should be considered training data only. This leaves the collection of (x, y) coordinates of all components at a given step of the placement algorithm as the input for our delay function generator. The specifics of the placement algorithm require us to come up with a function:

$$t_D(\Delta x, \Delta y) = t_D(|x_d - x_s|, |y_d - y_s|), \tag{5.1}$$

that gives an estimate for the delay d, given the difference along the x-axis and y-axis of the positions of the driver-sink pair under consideration. This function is required to be continuously differentiable in (Δx, Δy) i.e., (belong to differentiability class, $\mathcal{C}^1$) and needs to be strictly increasing. It is, however, allowed to use a new delay estimator in each new iteration of the outer algorithm loop. This allows e.g., to account for component density etc. (see density based approach in Section 5.6).

5.2.2 Methodology

Subsequent chapter will visit each methodoly we considered on the study group days:

- M1: A classification and regression approach: this approach is based on filtering out non-critical components and then finding reasonable linear interpolants for each components type (this info together with (x, y) coordinates is available at the outset of the placement step. See Section 5.3.

- M2: Classification and quadratic regression. A quadratic regress could be more successful based on the underlying circuit dynamics. The underlying physics of

delay and responsiveness is briefly revisited. A cost functional is introduced and the results are presented. See Section 5.4.

- M3: A statistical approach based on identifying the distribution of delays in the training data. A distribution is fitted for each value of the Manhattan distance. The resulting sequence of means is then interpolated. See Section 5.5.

- M4: It can be meaningful to include global information such as the local density of components. A strategy for this has been included in this document. Because of time constraints this approach has not been field tested. See Section 5.6.

- M5: A pure data-driven predictor based on Gradient-boosting regression. This methodology achieves the best testing performance on Data-set-1 and Data-set-2 among our trials. See Section 5.7.

5.3 M1: Classification

The patterns and trends in a Manhattan-dist vs recorded delay graph are qualitative and quantitatively different for different classes of driver-sink pairs. This provides the starting point for the classification scheme introduced here. We will considering classification according to LIBCELL, NUMSINKS, and PARITY (inclusion of invertors).

5.3.1 LIBCELL

Each driver has a LIBCELL, a field hinting towards the type of component, given in the form L*number* (e.g., $L0$ $L1$, $L2$,...). The meaning of these labels is that every driver with the same LIBCELL is the same type of component. We were hoping that different components show different behaviours. Among the critical sinks this leads to five classes of components. The trends for those five classes do not seem to be very different and indeed a classification and regression approach based on LIBCELL did not improve upon the benchmark error for delay estimation. (Figure 5.1).

5.3.2 isink vs sink: Invertor parity

Every driver has a number ($n \geq 0$) of sinks and isinks. An isink is a sink that is preceeded by an odd number of inverters. Distinction between a sink and an isink is important for deciding whether we have to compare rise with rise time and fall with fall time, or rise with fall time and fall with rise time. Now if we look at the delay time plotted against

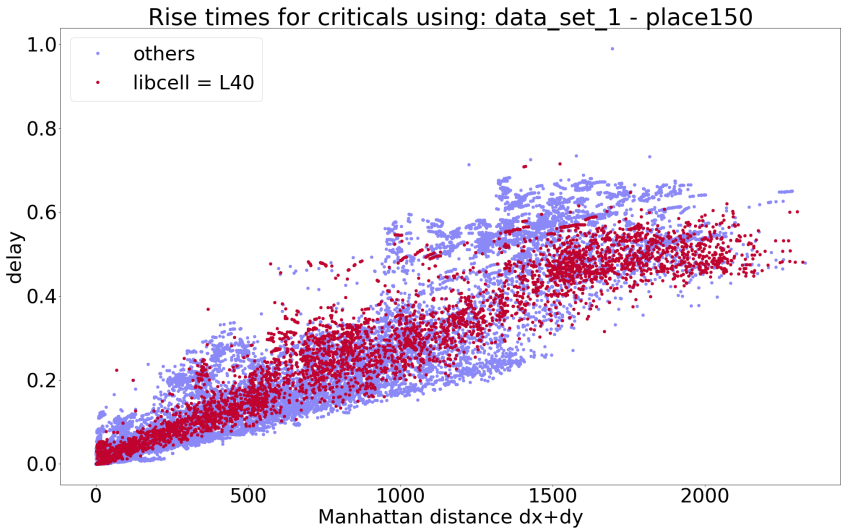


Figure 5.1: Highlighting the libcell = L40 data.

the Manhattan distance for the sinks and isinks we can see that the isinks tend to have an above average delay. See Figure 5.2.

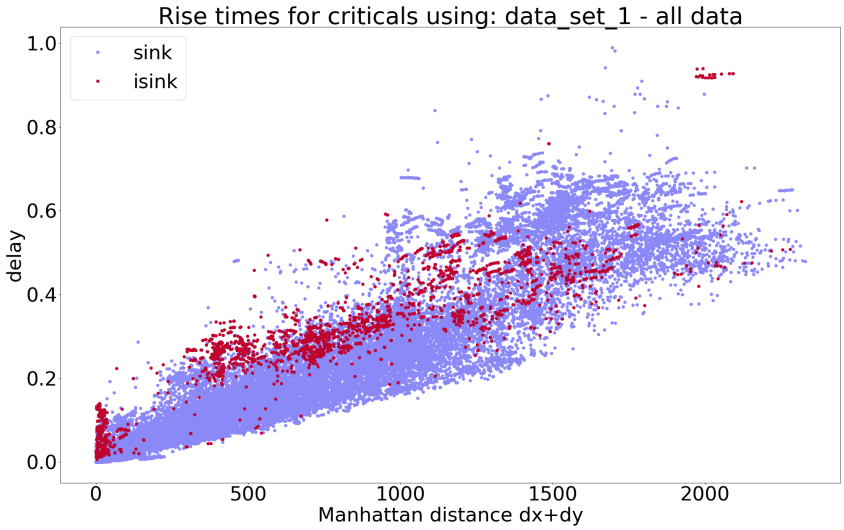


Figure 5.2: Highlighting the isink drivers.

5.3.3 Number of sinks per driver

Each component is connect to one or more other components. This network layout comes from the translation of a functional description of the chip into a set of interconnected electronic components. The graph properties of this network can be considered fixed for the problem at hand.

Some drivers have to drive only a single sink. On the other hand, some drivers have to drive ten sinks or more. Driving lots of sinks is a heavy load, and will often demand the intervention of buffers. When looking at the critical (driver, sink) pairs where the driver only has one sink, on average the delay is lower than for most (driver, sink) pairs. For (driver, sink) pairs where the driver is driving 10 or more sinks the delay is higher than on average. Improvement might be possible by further fine-tuning the exact groups, but we settled for: $n = 1$, $n = 2$, $n = 3$, $4 \leq n \leq 6$, $7 \leq n \leq 9$, $10 \leq n \leq 99$, and $100 \leq n \leq 999$.

5.3.4 Clustering of points

In the centre of the chip there is a very dense cluster of sinks. These sinks appear in the (driver, sink) pairs that have a short Manhattan distance but a relatively (very) high delay time. Filtering out this specific group of (driver, sink) pairs might result in an accurate estimate for this group, and less of an offset due to this group on the bulk data.

5.3.5 Results

Making a delay predictor function f using linear regression per category of (driver, sink) pairs has been performed. The classes of pairs used are a combination of the classification by sink vs isink and number of sinks per driver. So a total of 14 classes are defined, see Table 5.1. See Figures 5.3 and 5.4 for a selection of regression lines through 4 different categories of (driver, sink) pairs.

An interesting difference between dataset 1 and dataset 2 is that in dataset 1 both the even and odd numbered categories show a somewhat single band of datapoints. This suggests that selecting on either sink or isink neatly splits the dataset. For dataset 2 however the even numbered (sink) categories show two bands of data, where the gap in the middle is somewhat filled by the next category (isink), this is seen better for the case with a few sinks per driver. This suggests that where selecting on isink does give a single group of (driver, sink) pairs that are similar, selecting the sink (driver, sink) pairs leaves us with two distinct groups of datapoints. This could be because the lower band contains (driver, sink) pairs with 0 inverters between them, while the upper band contains (driver, sink) pairs with 2 inverters between them, resulting in an even number of inverters, and thus the 'sink' label.

Table 5.1: Classification used for the different linear regression categories.

Category 0	sink & 1 sink/driver
Category 1	isink & 1 sink/driver
Category 2	sink & 2 sinks/driver
Category 3	isink & 2 sinks/driver
Category 4	sink & 3 sinks/driver
Category 5	isink & 3 sinks/driver
Category 6	sink & 4 . . . 6 sinks/driver
Category 7	isink & 4 . . . 6 sinks/driver
Category 8	sink & 7 . . . 9 sinks/driver
Category 9	isink & 7 . . . 9 sinks/driver
Category 10	sink & 10 . . . 99 sinks/driver
Category 11	isink & 10 . . . 99 sinks/driver
Category 12	sink & 100 . . . 999 sinks/driver
Category 13	isink & 100 . . . 999 sinks/driver

5.3.5.1 RMS errors

The final result from using this method gives for the data sets, with training data the `place_150` set, and testing data the `place_100` set a root mean square error of:

Input	Error
Dataset-1	0.03881
Dataset-2	0.00844

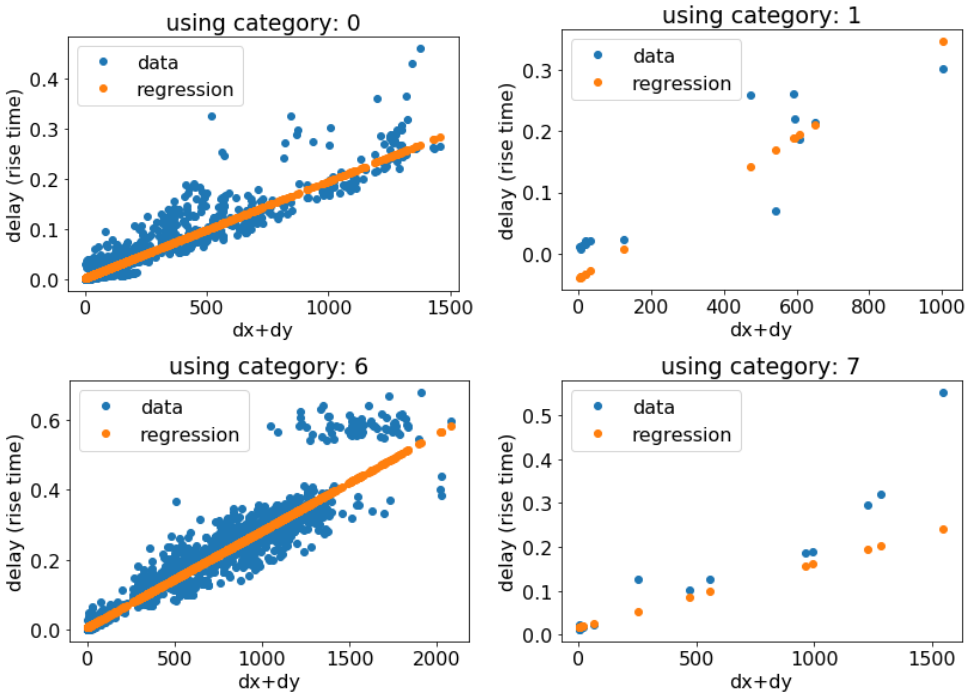


Figure 5.3: Linear regression trough subsets of dataset 1.

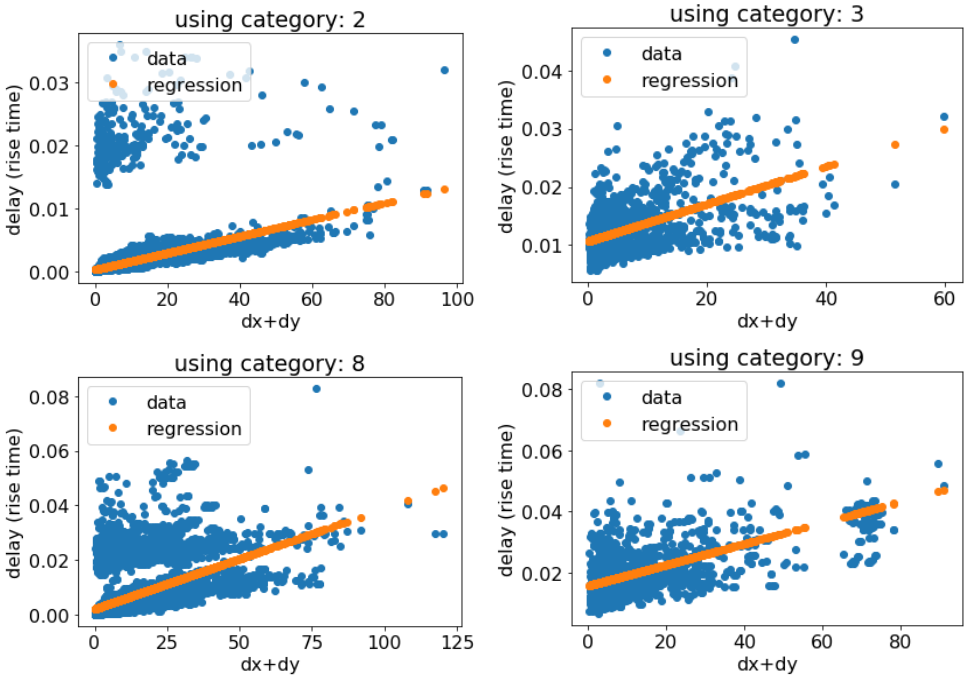


Figure 5.4: Linear regression trough subsets of dataset 2.

5.4 M2: Regression based on underlying physics, classification, and quadratic fitting

5.4.1 Physical insights of the delay between source and sinks

The latency of a signal is determined by the resistances and capacitances of the connections (or wires) and components (or cells). Both delay in cell and delay in wire usually depend on placement, routing topology, etc. In this section, we are interested in the delay in wire, which also depends on metal layer choices, neighbouring, etc. In fact, the approximate delay in wire $t_{D,wire}$ is

$$t_{D,wire} \approx R_w \times \left(\frac{C_w}{2} + C_{in}\right) = \frac{R_u C_u}{2} \times L_w^2 + R_u C_{in} \times L_w, \tag{5.2}$$

with $R_w \approx L_w \times R_u, C_w = L_w \times C_u$, where L_w [meter], R_u [Ohm/meter] and C_u [Farad/meter] represent the length of the connection from diver to sink/isink, conductance and capacitance belonging to the connection, respectively, see Figure 5.5. Moreover, when the connections are long (see Figure 5.5) or connections are from one to many points (see Figure 5.6), repeaters (or buffers) are placed to boost the signal. In this case, $t_{D,wire}$ is approximated by

$$t_{D,wire} \approx t_{D,buf} + 2 \times \frac{R_w}{2} \times \left(\frac{C_w}{4} + C_{in}\right) = t_{D,buf} + \frac{R_u C_u}{4} \times L_w^2 + R_u C_{in} \times L_w, \tag{5.3}$$

where $t_{D,buf}$ is the delay of a repeater.

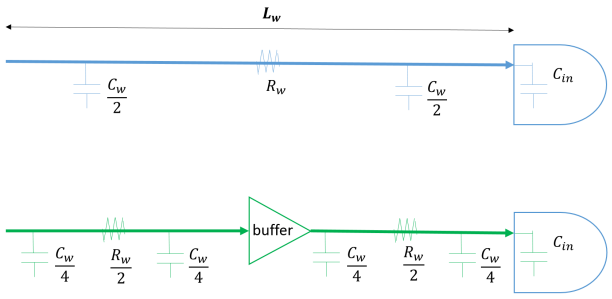


Figure 5.5: Connection from driver to sink (top) and long connection buffered by repeaters (bottom). For each case, the delay $t_{D,wire}$ is determined by (5.2) *and* (5.3) *respectively.*

From (5.2) and (5.3) of $t_{D,wire}$, we see that physically, the delay function should have quadratic behaviour, and is of the form

$$f(x_d, y_d, x_s, y_s) = \alpha_2 d_M^2 + \alpha_1 d_M + \alpha_0, \tag{5.4}$$

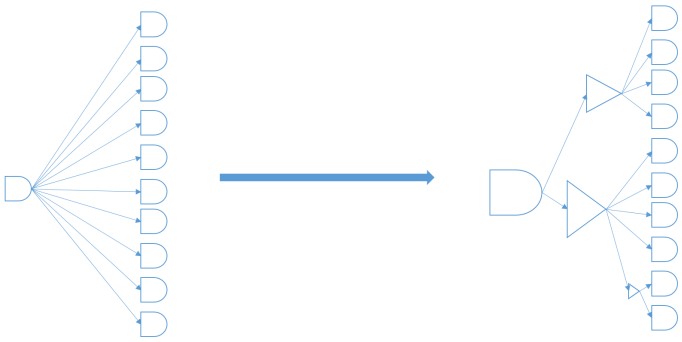


Figure 5.6: Connections are not one to one but one driver to many sinks/isinks. In this case components/cells may be sized and/or buffer may be needed.

where d_M represents the Manhattan distance between two points, with $d_M = |x_d - x_s| + |y_d - y_s|$. When the distance is long and/or the connection is not point to point, a repeater with delay α_0 is needed. $\alpha_i, i = \{1, 2\}$ are coefficients corresponding to resistances and/or capacitances from (5.2) and (5.3). We need to determine $\alpha_i, i = \{0, 1, 2\}$ to have the general formula of delay function. To aim this, a regression model is built.

Note: we realise that the electrical optimisation has as a consequency that the piecewise quadratic form described above globally resembles a linear function. This implies the approach pursued here cannot in general be successful and requires some information about the circumstances under which electrical optimisation takes place.

5.4.2 Regression model

For simplicity let $p = (\alpha_2, \alpha_1, \alpha_0)$. The piecewise function J minimizes the differences between our predicted delay function $f_p(x_d, y_d, x_s, y_s)$ and delay computed after placement (or reference delay) t_D from the data. Denote N as the number of delays computed for each pair of driver and sink/isink. A cost function can be defined as

$$J(p) = \frac{1}{N} \sum_{i=1}^{N} \left(f_p(x_d^i, y_d^i, x_s^i, y_s^i) - t_D^i \right)^2 \tag{5.5}$$

To minimize the $J(p)$, we use the data extracted from file `place_150` of data set 1. The parameters $p = (\alpha_2, \alpha_1, \alpha_0)$ of $f_p(x_d, y_d, x_s, y_s)$ are found for different types of extracted data. Precisely, we find p for

- 1 driver, multiple sinks/isinks case
- 1 driver, critical sinks/isinks case

Case	$p = (\alpha_2, \alpha_1, \alpha_0)$	Error E
1	$p = (10^{-7}, 4.8 \times 10^{-4}, 3.5 \times 10^{-3})$	0.18
2	$p = (-2.87 \times 10^{-8}, 3.54 \times 10^{-4}, -5.21 \times 10^{-4})$	0.19
3	$p = (-3.26 \times 10^{-8}, 3.81 \times 10^{-4}, 2 \times 10^{-2})$	0.36

Table 5.2: Computed values of p and error E

- 1 driver, critical sinks case

The value of computed p and E for each case are in Table 5.2. The total relative error for each case is computed by

$$E = \sum_{i=1}^{N} \frac{|t_D^i - f_p(x_d^i, y_d^i, x_s^i, y_s^i)|}{t_D^i}$$

.

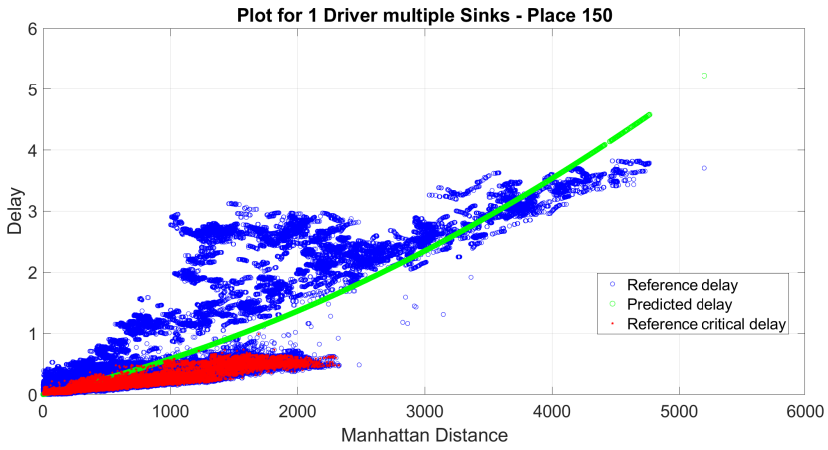


Figure 5.7: Estimated delay for case 1: 1 driver, multiple sinks/isinks

For the first case, our delay function follows the dominated shape of the reference delay t_D in Figure 5.7 which is close to what we expected. The second and third cases give results not as good as the first one since we deal with critical points (which is more challenging). In fact, latency of signal also depends on topology of the circuit, physical effects, etc., thus looking at only coordinates of driver and sink/isink would not give enough information. Extra efforts would need in the future to overcome this challenge.

5.5 M3: A Statistical Approach on Data Set 2

In data set 1, the relation between the Manhattan distance and the delay shows a clear trend. However, the relation of these two physical quantities is not very clear for the data set 2. This can be seen from Figure 5.8. Whatever the cause, the qualitative difference suggests additional data is needed in order to develop an algorithm that is efficient on multiple data sets, corresponding to different chips design and fabrication technologies. Parameters that we expect are particularly relevant are number of layers in the substrate, etching resolution, and choice of semi-conductor materials.

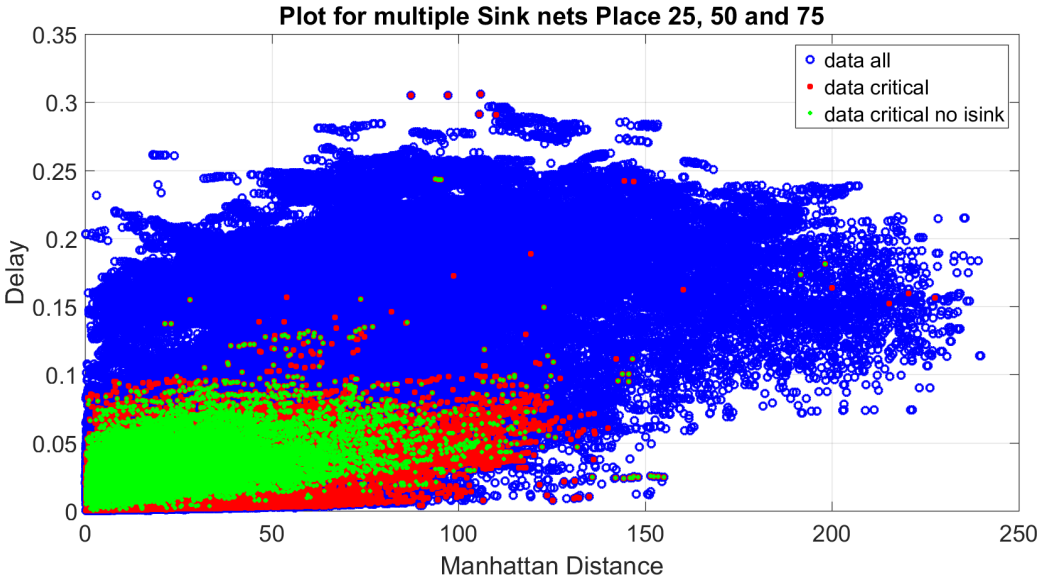


Figure 5.8: Scatter plot of `place_25`*,* `place_50` *and* `place_75` *of data set 2.*

To investigate the relation between the Manhattan distance and the delay, we attempt to first find the distribution of the data and then use the statistics behind it to model the relation between the Manhattan distance and the delay.

5.5.1 Distribution Fitting

To perform statistic analysis of the relation between the Manhattan distance and the delay, we first divide the data into small subsets. Denote the Manhattan distance as d_M and the delay as t_D. When Manhattan distance is less than 40, we divide the whole data set into 8 subsets. Namely, in subset $\mathcal{S}_i, i = 1, 2, \ldots, 8$, the Manhattan distance satisfies $d_M \in [(i-1)\cdot 5, i\cdot 5)$ and the delay t_D are the corresponding delay data at those

Manhattan distances. Then $\mathcal{S}_i$ can be written as

$$S_i := \{(d_M, t_D) | d_M \in [(i-1)\cdot 5, i\cdot 5)\}, \text{ if } d_M < 40,\ i = 1, 2, \ldots, 8.$$

When $d_M \in [40, 80)$, to make sure there is enough data in each subset, this part of data is divided into two subsets. Namely,

$$S_9 := \{(d_M, t_D) | d_M \in [40, 60)\},$$
$$S_{10} := \{(d_M, t_D) | d_M \in [60, 80)\}.$$

The last data set contains the delay and the Manhattan distance data corresponding to $d_M \in [80, 120)$, i.e.,

$$S_{11} := \{(d_M, t_D) | d_M \in [80, 120)\}.$$

Investigating the histogram of the delay t_D, one can see that the distribution to the delay in each subset can be approximated by a Beta distribution, which has the probability density function defined as

$$Beta(\alpha, \beta, x) := \frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha, \beta)}, \tag{5.6}$$

where

$$B(\alpha, \beta) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)},$$

and $\Gamma(z)$ is the Gamma function. Many definitions of the Gamma function exist. For example, if $z \in \mathbb{C}^+$, i.e., a complex number with positive real part, then the Gamma function on z is defined via the following improper integral

$$\Gamma(z) = \int_0^\infty x^{z-1} e^{-x} dx.$$

If $z \in \mathbb{Z}+$, i.e., a positive integer, the Gamma function is simply the factorial of $z-1$,

$$\Gamma(z) = (z-1)!.$$

For a more detailed explanation, we refer to the Wikipedia page of the Gamma function ([1]).

The parameters α and β in Beta distribution are used to control the shape of the function. Figure 5.9 shows how the shape of the probability density function varies according to different values of α and β. Hence, suitable choices of the parameter α and β may lead to a satisfactory fitting of the histogram of the delay data t_D in the distribution sense. Furthermore, we only consider the data points which are critical,

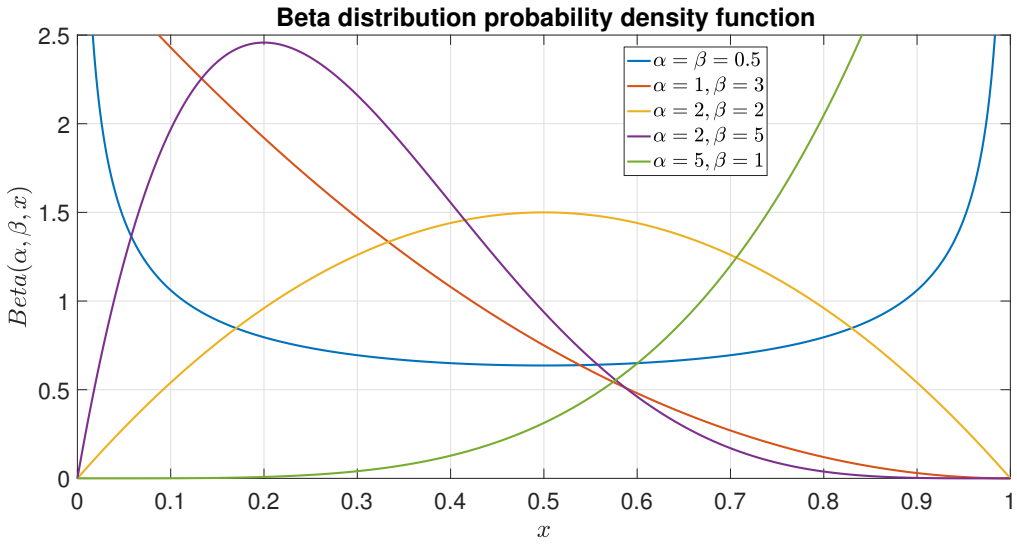


Figure 5.9: Probability density function of Beta distribution for different parameters.

the connections **without** "isink" and the connections which have less than 1000 sinks. Namely, the green dots in Figure 5.8. Some fits seem to be not very accurate. This is an indication that there might be a more appropriate family of distributions to describe the statistics of the data. To make sure the data set is rich enough, we combine data sets `place_25`, `place_50` and `place_75` of the data set 2. We call this data set training data. The histogram and the resulted fitting curves are depicted in Figure 5.10. The parameter values of α and β are shown in Table 5.3.

Table 5.3: Values of α and β for the training data.

Manhattan distance	$d_M \in [0, 5)$	$d_M \in [5, 10)$	$d_M \in [10, 15)$	$d_M \in [15, 20)$
α	6.2233	5.0182	4.9405	5.2435
β	397.9271	228.9191	188.9371	176.3081
Manhattan distance	$d_M \in [20, 25)$	$d_M \in [25, 30)$	$d_M \in [30, 35)$	$d_M \in [35, 40)$
α	5.3851	5.2903	5.7696	5.5687
β	164.3603	148.3772	147.8672	134.2422
Manhattan distance	$d_M \in [40, 60)$	$d_M \in [60, 8)$	$d_M \in [80, 120)$	
α	5.9534	5.9899	5.9899	
β	130.6464	112.7353	112.7353	

To test whether the obtained probability density function can be used to approximate the distribution of the remaining data in data set 2, i.e., the data in `place_100` and `place_150`, we plot the histogram of the data in `place_100` and `place_150` and the

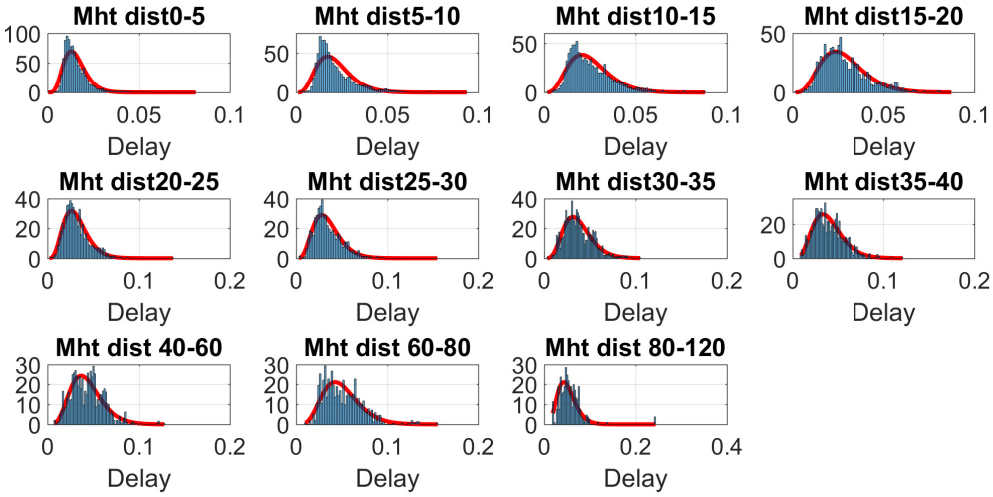


Figure 5.10: Histogram and fitted probability density function curves for `place_25`*,* `place_50` *and* `place_75` *in data set 2.*

probability density function curves. The results are shown in Figure 5.11. It can be

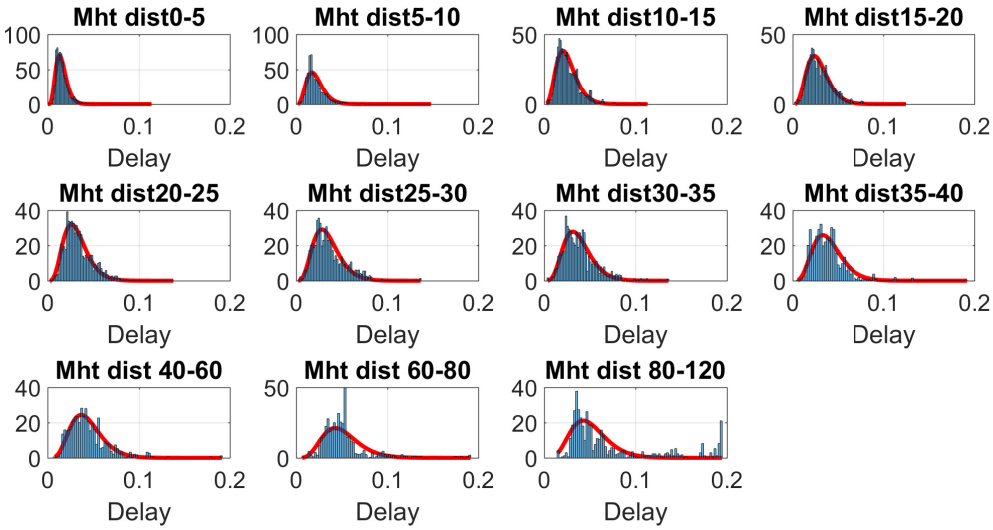


Figure 5.11: Histogram of the delay data in place 100 and 150 and the probability density function curves derived from the training set.

seen that the probability density function curves can still fit the histogram of the data in `place_100` and `place_150` very well when the Manhattan distance is less than 60. When the Manhattan distance is between 60 and 120, the differences are much larger.

This is most probably caused by the lack of data point when $d_M \in [60, 120)$. Hence, the statistical analysis approach requires to have enough data.

When the probability density functions are approximated, we still need to find a relation between the Manhattan distance and the delay. One possibility is to simply use the mean and variance calculated from the Beta distribution and then connect the mean to make a piece-wise linear function to approximate this relation. However, due to the time constraints, we could not further pursue this direction. In the following, we investigate the mean-bound method (introduced in the next subsection), which has the same reasoning.

For a more complicated model, which may provide a better approximation of the relation between the Manhattan distance and the delay, Gaussian process regression (see, e.g., [3]) can be a potential future research direction.

5.5.2 The AM-method

In this section, we explain how to obtain a candidate for the delay function by taking either averages or medians of the data points. The abbreviation "AM" refers to average and median. First, we illustrate the method by starting from the data set as represented in Figure 5.8, where as explained in Section 5.5.1, we only consider the connections without 'isink'. Secondly, we motivate the reasoning behind the AM-method.

The algorithm basically follows three steps:

1. First, divide the domain of Manhattan distances of Figure 5.8 into subintervals I_i, $i = 1, \ldots, N$. The length of each subinterval I_i and the number of subintervals N are parameters that have to be chosen.

2. Secondly, for each subinterval I_i, determine the average delay time t^i_{av} from the data points contained in $I_i \times [0, \infty)$. Alternatively, one can take the medians.

3. Thirdly, denoting by x_i the left-point of the subinterval I_i, connect for each $i = 1, \ldots, N-1$ the data points (x_i, t^i_{av}) and $(x_{i+1}, t^{i+1}_{\text{av}})$ by a linear function.

Figure 5.12 shows the resulting delay function when following the above procedure. When comparing the delay-functions from Figure 5.12 to the test data, we obtain a root-mean-square error of 0.01215 for the "average"-delay-function, and a root-mean-square error of 0.01208 for the "median"-delay-function. **Both of them outperform the state-of-the-art method used by Synopsys.**

The motivation behind the AM-method is similar to the one given in Section 5.5.1. The basic assumption is that the distribution of delay depending on the Manhattan

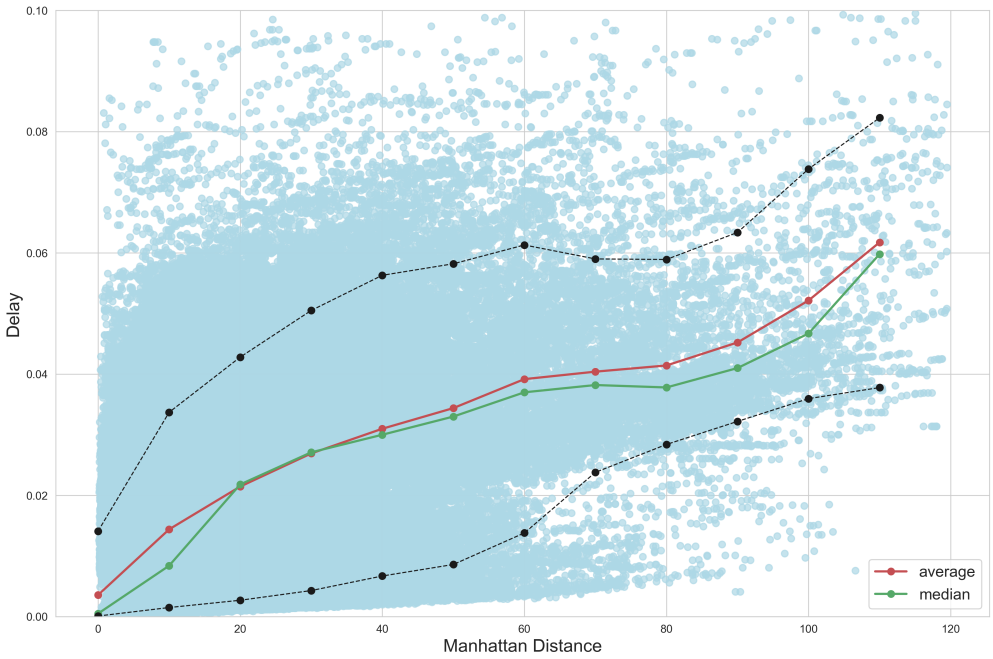


Figure 5.12: The delay functions obtained by the AM-method. The delay function in red is obtained when taking averages, the one in green when taking medians. The black curves give an idea of the standard deviations of the distributions in the corresponding category.

distance is meaningful. If that is the case, then a statistical analysis of these distributions might improve the estimate of the delay function. The difference of the AM-method to the distribution-fitting method of Section 5.5.1 is that the averages or medians are computed directly from the data, rather than being derived from distributions.

5.6 M4: Incorporate the connection structure

It seems that in several of the approaches proposed we are not utilising the network/graph structure. We know that there are several connections, regions of the IC are dense, and other areas are sparse, and this is not being incorporated in the predictions.

5.6.1 Speed of signal propagation

For convenience let's assume that the signal delay is approximately proportional to the distance between the driver and the sink. (This is in contrast to the theoretical/physical

analysis based on a wire's capacitance which predicts quadratic dependence, but the data seems to show approximately linear behaviour). In a manner similar to most introductions to variational calculus, we can consider there as being some speed profile (field) for signals propagating across the IC. If we suppose that the speed of signal propagation is homogeneous and isotropic, then the quickest path is a straight line between any two points. (There is the constraint that wires are confined to Manhattan paths along the circuit, but if most connections are locally small then a straight line path is likely not an awful approximation). We are interested in the related problem, which appears to be the reverse of this, namely, that if we are forced to travel in an approximately straight line, can we predict the signal's travel time. The key ingredient here is then the construction of the signal's speed profile, which is an area where we suspect the network structure of the connections can be exploited (hopefully intelligently).

5.6.1.1 Incorporating the network structure

Consider the following example dynamics, which is that when several drivers are closely packed together in a dense region, that the area appears quite congested. Perhaps such a scenario would suggest that there is a large amount of signal traffic in this area, and hence that small perturbations of a driver/sink in this region will drastically influence the signal delay.

Perhaps the converse might be true, and that a high density of drivers/sinks in a region is indicative that the area is a popular and fast flowing region, and that signals travel fast in this area. Hence perturbations here only produce small changes in the delay time.

Another possibility is that if we were to draw in all the paths the signals travel along, there are regions where these overlap extensively, and then sparser regions. In this case it is the degree of overlap that a connecting path would need to travel through which would determine the signal delay.

5.6.1.2 Constructing a speed profile

For the sake of convenience, let's suppose that the density/clustering of drivers in a region determines the signal's travel time. There are a few means by which we can define some form of density profile. One which comes to mind is to construct a *kernel density estimate* (KDE), where the neighbourhood on the IC which a driver congests is reflected in the choice of bandwidth. An example of one such KDE is shown in Figure 5.13.

One of the pitfalls of a KDE is that it is expensive to evaluate. However, if speed is

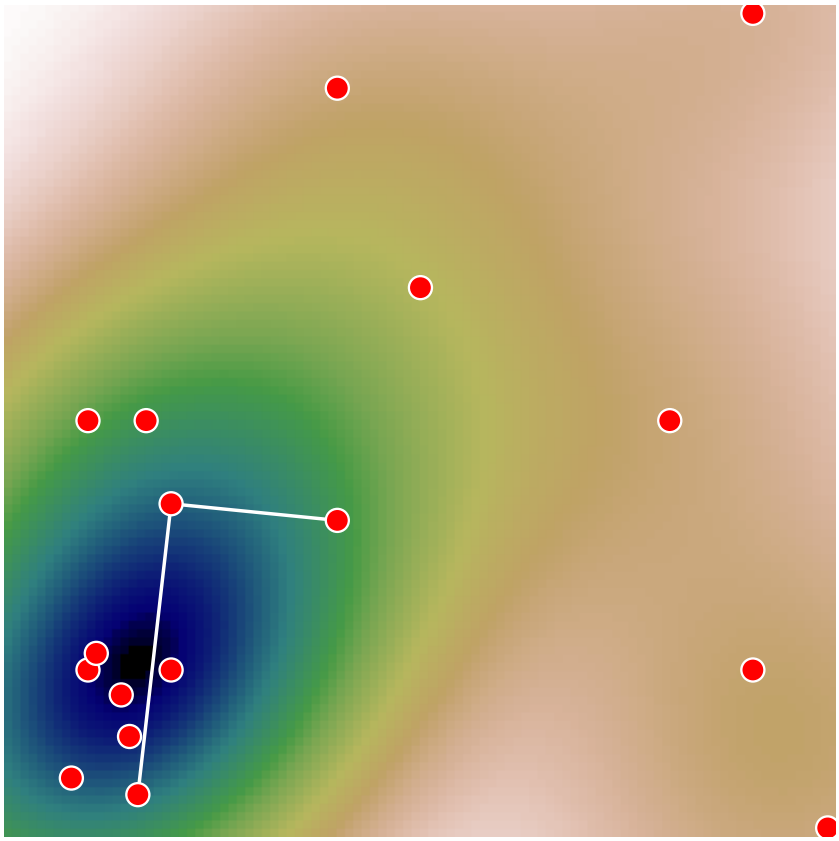


Figure 5.13: An example of a kernel density estimate constructed for a selection of drivers. We highlight two example connections, where one passes through the most dense region, and another along a less dense region.

very critical then perhaps this can be approximated by some sufficient order 2-dimensional polynomial. The advantage of this is that then the integrated travel time is trivially evaluated computationally (requiring no computationally intensive approximate integration techniques). A second pitfall is that the KDE's range is $[0, \infty)$, where in reality we would want some baseline speed. This can be achieved by some transformation of the form

$$\text{speed} = \alpha + \beta \times \text{density}, \tag{5.7}$$

where α is strictly positive. The exact form of these coefficients, and also the mapping could be explored during a calibration/training stage.

Overall, once some speed profile $v(\boldsymbol{r})$ has been constructed, and two points A and B have been found, then we approximate the delay t_D as

$$t_D \approx \int_{\boldsymbol{r}_A}^{\boldsymbol{r}_B} v(\boldsymbol{r})\, d\boldsymbol{r}\,, \tag{5.8}$$

where the integral is trivially evaluated if polynomial approximations (or splines, Cheby-

shev functions, etc.) are used. Furthermore, this is trivially differentiated or evaluated under small perturbations.

5.6.2 Other network/graph features

So far this has used a very geometric interpretation of the network. However, it is not too complicated to also pull out prediction features, such as degree of connectedness, the number of vertices forming a cliques, etc., then these can also be used as features (possibly). Other simpler degrees of how crowded a region is would be its nearest neighbour distance, or the number of neighbours within some region ε around a node. Other ideas from graph theory might be to try and classify regions that might form a *small world network*, and see if the delays of connections within these regions show a structure/correlation different to large world networks.

5.7 M5: Data-based Prediction via Gradient-boosting regression

5.7.1 Problem Setup

In this section, a pure data-based delay predictor will be presented based on gradient-boosting regression (GBR). Generally the available data for prediction are associated to the driver and the sink, respectively. The name and label of each data is not clearly related to the delay, which can be ignored in later discussions. Other data regarding the driver and sink (isink) are described as follows.

- driver: position (X_d, Y_d), rise r_d, fall f_d, sink driving number N_d.
- sink/isink: position (X_s, Y_s), rise r_s, fall f_s, slack S_k.

From explanations from the company engineers, the delay can be described as the rise difference $r_s - r_d$ between driver and sink.

From the current experience in Synopsis, the delay is influenced by the distance between drivers and sinks, and not related to respective positions, leading to the initial need to find a mapping from the distance to the delay (which is described in the competition description).

However during our discussions and from the analysis in Section 5.3, the sink driving number N_d is found to have a great influence for the delay. This leads to our predictor design to accept input of distance and the driving number simultaneously in the following.

More specifically, the delay prediction task is set to find a mapping from three inputs (X-distance,Y-distance,driving number) to the delay estimate of $r_s - r_d$, say P_d,

$$\varphi : (|X_d - X_s|, |Y_d - Y_s|, N_d) \to P_d. \tag{5.9}$$

We have tried different prediction models to fit current historical data where the gradient-boosting regression (GBR) model outperforms in our limited tests. Hence the following discussion contributes to introduce GBR models and its empirical performance on the training data provided during the SWI workshop.

5.7.2 Gradient-boosting Regression

Gradient-boosting (see [2]) is a powerful algorithm widely used in data-driven machine learning tasks like regression, classification and ranking. It belongs to a general ensemble learning algorithm called boosting methods, where base estimators are built sequentially such that combined estimators' bias can be reduced. The target of GBR is to learn a strong model from the combination of several weak models.

The basic idea of boosting starts from following setup:

- A loss function, say $L(y, f(x))$ for the true y and the prediction $f(x)$.
- Weak learners, say $h_i, i = 1, ...$
- An additive model to add weak learners to minimise the loss function, say $F(x) = \sum_{i=1}^{M} \gamma_i h_i(x)$ for pending coefficients γ_i.

Specifically in the Scikit-learn package, the weak learners are chosen as decision trees. From the official help document of GBR model, "Gradient Tree Boosting uses decision trees of fixed size as weak learners. Decision trees have a number of abilities that make them valuable for boosting, namely the ability to handle data of mixed type and the ability to model complex functions."

Moreover with the tree learners, the additive model itself can be formed as follows to fit the sequential learning,

$$F_i(x) = F_{i-1}(x) + \gamma_i h_i(x), \tag{5.10}$$

where the weak learner h_i may come from the following optimisation problem:

$$h_i = \arg\min_{h} \sum_{j=1}^{n} L(y_j, F_{i-1}(x_j) + h(x_j)). \tag{5.11}$$

Here $\{x_j\}_{j=1}^{n}$ denote the train data samples.

To solve the above optimisation problem is not facile, and an insightful understanding of gradient-boosting is to introduce the gradient-descent algorithm in the above boosting setup. Hence a GBR model is finally described as an updating algorithm as follows,

$$F_i(x) = F_{i-1}(x) - \gamma_i \sum_{j=1}^{n} \nabla_F L(y_j, F_{i-1}(x_j)), \tag{5.12}$$

where γ_i can be chosen by linear search with a constant $\gamma > 0$,

$$\gamma_i = \arg\min_{\gamma} \sum_{j=1}^{n} L(y_j, F_{i-1}(x_j) - \gamma \frac{\partial L(y_j, F_{i-1}(x_j))}{\partial F_{i-1}(x_j)}) \tag{5.13}$$

5.7.3 Empirical Results

To design the predictor, we will use the gradient-boosting regression models in Scikit-learn package directly and our experiments are done in a laptop workstation with i7-7700HQ CPU.

GBR model is setup with 100 estimators and maximal search depth of 7, which is trained and tested in data-set-1 and data-set-2 respectively. Following the general setting in the group discussion, we use the place-150 data as train data, and the place-100 data as test data in both data-sets. The test results measured by RMSE is illustrated in Table 5.4, which achieves the best test results in all of our current trials (M1-M5). Also the testing performance is illustrated in Fig. 5.14 and 5.15 respectively.

The accuracy of GBR has a higher time-cost. From our experiments, the GBR model is about 70 times slower than the linear regression model (0.02961s v.s. 0.000475s) based on the testing time over 12081 samples for data-set-1; about 200 times slower than the linear regression model (0.5184s v.s. 0.002507s) based on the testing time over 191440 samples for data-set-2.

Metric	Data-set-1:place-100(sink)	Data-set-2:place-100(sink)
RMSE	**0.03845777**	**0.00663324**

Table 5.4: Test results on chosen data-sets. Models are trained from the place-150 data in data-set-1 and data-set-2 respectively.

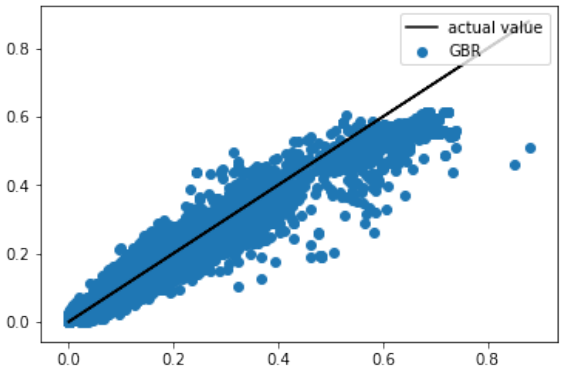


Figure 5.14: Test on Data-set-1:place-100

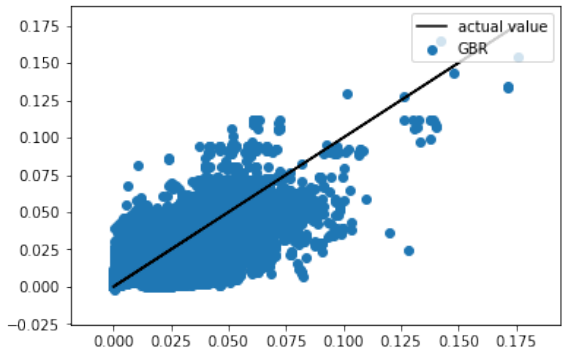


Figure 5.15: Test on Data-set-2:place-100

5.8 Recommendations

From our experiments the most performing and robust method turned out to be **M3: Statistical method**. At this point we recommend the implementation of this method. It is possible that upon further fine-tuning of the optimiser and classifier other methodologies turn out to be superior. Regardless the method chosen, benefits can be gained from a classification before the data is fed to the optimiser. In this regards, we can formulate the following *recommendations*:

- do **not** use the libcell label for classification. This does not lead to a meaningful subdivision of the data in the sense that each resulting subset exhibits trends not already present in the complete set.
- provide more detailed information about the component type. It is possible that this will lead to a meaningful classification with predictive power as pertains delay estimation. Examples of parameters we imagine could provide more insight are: components size, component power, and number of ports.
- **do** use sink vs isink for classification. From our analysis it is clear that components at the end of lines containing odd/even numbers of invertors exhibit significantly different trends and statistics. We expect this not just to be a property of the parity, but of the exact number of invertors. Having access to this datum likely results in more accurate predictions.
- **do** use the number of sinks a driver powers for classification. This information is partially available from the data files provided.
- **do** use the location of sinks, combined with the local sink density for classification. *In our tests this was not considered due to time restraints.*

5.9 Conclusion

An improvement over the state-of-the-art has been observed. It is however likely that a stronger correlation, and thus a better delay estimator, can be created by first pre-processing the data given. It is possible e.g., that distance to the boundary of the chip, crowding and etc., significantly affect the final delay after routing.

A good place to start in our opinion for further improvement, would be to study the routing algorithm and try to extract heuristics and patters from its outcome and inner workings. This can help to direct the search for strong correlations in the placement vs delay function.

It is unclear at this stage what the optimal accuracy of any delay estimator is. Obviously a perfect delay estimator requires the consolidation of placement and routing; this is prohibitively costly and should not be aimed at.

Another question is how robust any solution is in transitioning from one generation of micro-architecture to the next. Can the delay function be copied over? This is unlikely, but it seems feasible that the type of considerations that lead to the delay function remain valid across generations.

Finally we would like to thank SynopSys and its representatives for this opportunity. It is interesting to see what wealth of technology and know-how lies behind a truly omnipresent technology. This introduction has at the same time an effect of demystification and deep appreciation of the designers and design methodologies!

Acknowledgements

We would like to take the opportunity to explicitly thank the NWO for their support of the SWI initiative and the delegates from Synopsys for their time and help during the week. It was an enriching experience and a good showcase of how mathematics can play an important role in the industrial design process. Also thanks to the reviewer to take the time to go through this document and to help improve it.

Bibliography

[1] Gamma function Wiki Page

[2] Gradient Boosting Wiki Page

[3] Rasmussen, C.E. and Williams, C.K.I., Gaussian processes for machine learning, MIT Press, 2006

Chapter 6

Smart Traffic: Intelligent Traffic Light Control

Joseph Field[1], Rogier Brussee[2], Evert-Jan Bakker[3], Jeremy Budd[4], Rémi de Verclos[5], Steven Fleuren[6], Stephanie Gonzalez Riedel[6], Emma Keizer[3], Hans Stigter[3], Guus ten Broeke[3], Mark van den Bergh[7]

[1]Oxford University, UK
[2]University of Applied Sciences Utrecht, The Netherlands
[3]Wageningen University, The Netherlands
[4]University of Nottingham, UK
[5]Radboud University, The Netherlands
[6]Utrecht University, The Netherlands
[7]Leiden University, The Netherlands

Abstract:
The Study group participants for the Intelligent Traffic Light Control assignment were tasked with investigating the use of autonomous traffic light controls for a network of road intersections such as the road map of a city. Smart Traffic, the software used and designed by our client Sweco, is designed to control a single intersection. Our task involved optimising the traffic flow without the need for global (e.g. city wide) governance of all traffic lights. This would both fit with the existing approach in Smart Traffic, avoids bad experiences in the past, and mathematically avoids a huge explosion of the state space. Unfortunately, the Smart Traffic software was not available for use and study, and this somewhat limited the approaches we considered. Instead, the software was treated as a black-box, for which we only knew an idealised version of the cost function that the software tried to optimise. We therefore focused on finding a method to have multiple instances of this control coupled to each other in a loose and comparatively simple way. The methodology of Smart Traffic meant that each intersection would act in a 'greedy' fashion, working solely with local information and without any regard for neighbouring intersections. We were advised to devise simple ways in which the software could be adapted, such that each intersection would use only limited information about its neighbours. In particular we should avoid direct communication of optimisation strategies between neighbouring intersections. Indeed, this would be akin to using a global optimisation scheme for the entire network. The approach that we settled upon, was that each intersection implements a (time-dependent) cost function like for the existing Smart Traffic software, but modified to take into account the expected wait-time of downstream traffic. This balances local wait-times with future expected wait-times at neighbouring intersections as communicated by their nearest neighbours. It should result in intersections being less eager to release traffic in a direction where cars would, in the near future, be waiting a long time, and would be sending cars at a reduced rate in the direction of an already congested intersection. At the same time, each instance remains in control of the traffic at a single intersection.

Keywords: *Smart Traffic, Optimisation, Networks*

6.1 Introduction

6.1.1 Company Background

Sweco is a European engineering consultancy company. One of the products that it is further developing is *Smart Traffic*: a real time traffic data driven software that can be used for the efficient control of traffic lights. Using this information, Smart Traffic is able to create detailed predictions for upcoming traffic conditions, and is able to control traffic lights in order to optimise future traffic flow. The main sources of the traffic data used by Smart Traffic are loop detectors, which are embedded in most roads of urban areas, and data such as Floating Car Data, which details the location of moving vehicles in a road network using GPS and cell based location-data from mobile phones carried by the drivers.

A particular feature of Smart Traffic is that it balances the movement of individual road users with the overall mean traffic flow, and avoids long waiting times of individual cars, i.e. it weights individual car "latency" with intersection "throughput", such that mean square vehicle wait-times at four-way intersections can be reduced by up to 40%.

6.1.2 Problem Description

The implementation of Smart Traffic for a single intersection was introduced by Sweco with great success. At the scale of a single intersection, traffic predictions can be made such that total wait-time of cars can be greatly reduced. It is driven by minimising a cost function that is essentially the sum of squares of the waiting times of all cars *at the intersection* (see Section 6.2.2). In particular, the optimisation problem being solved is purely local to the intersection. Unfortunately, local optimisations may result in catastrophic consequences for traffic flow on the whole road network. In fact, the actual deployed software needs to take ad hoc measures to avoid such (rare) situations. Indeed, the group was told that a 2×2 grid of four-way intersections, each controlled by Smart Traffic, would generally tend to a state of gridlock. This was shown by van Hout in a master student's thesis at Eindhoven University of Technology that Sweco worked with prior to SWI 2019 [3]. He investigated the general utility of Smart traffic, as well as the use of Smart Traffic to control coupled intersections. His thesis was the primary source for the group in understanding how traffic models were used in Smart Traffic, and what type of cost function was optimised to provide efficient traffic flow.

Our problem was to find a method to couple multiple instances of the existing Smart Traffic software, such that each intersection was optimised using *full* local information but only using *limited* neighbouring information, without compromising the global behaviour

of the network.

6.2 Mathematical Model

We first review the work presented in [3], outlining the main simplifying assumptions before discussing the model itself.

6.2.1 Assumptions

The following simplifying assumptions are made for the single-intersection model. They will be made for the multi-intersection model as well, unless specifically indicated, in order to focus on the problem of coupling different intersections in a simpler idealised context.

Assumption 1: *Traffic is comprised of a single vehicle type.*

We assume that all vehicles are behaving indistinguishably (or are statistically governed by a single simple law such as Poissonian behaviour), i.e., we assume that all vehicles are of the same size, travel at the same speed, and have the same reaction times. For simplicity and ease of reading we will often call a vehicle a car.

Assumption 2: *Vehicles either travel at their desired speed, or are stationary.*

Rather than model car acceleration, the time needed for cars to reach their desired speed is instead attributed to a delay time, τ. Thus, velocities are treated as step functions.

Assumption 3: *Driver reaction time is zero.*

Similar to the effect of acceleration, any effects due to the reaction of the car driver are accounted for by the delay time, τ.

Assumption 4: *Traffic outflow occurs at a maximum flow rate μ.*

Cars can only leave an intersection, once the corresponding light is green, and the delay time, τ, has passed. After this time, cars leave each open lane of an intersection at a constant rate of μ cars per unit time.

Assumption 5: *Traffic lights are either green or red, we ignore yellow.*

When faced with a yellow light, drivers will either continue to drive through the intersection (effectively extending the green light interval), or will stop early (effectively extending the red light interval). Thus yellow could be treated as a (time dependent) probabilistic mix of red and green but for simplicity we simply ignore it.

Assumption 6: *Vehicles arriving at a red light 'stack' on already-present vehicles.*

This assumption allows us to forgo the need to keep track of specific positions of cars in the model. Instead, any vehicles that are waiting together are treated as a 'platoon', that will travel together to the next intersection. In larger networks this needs some modification as cars divide themselves over the network.

Most of these assumptions are used to reduce the amount of book-keeping that would be needed for a more involved model, i.e. a model that allowed for different vehicle characteristics, or specific modelling of delay effects. However, from the results presented in [3], the reduced model captures much of the behaviour of a complex one.

6.2.2 Model

The traffic model that we discuss is found in [3], with notation being taken from [5]. To fix notation wedefine

λ_p	Mean arrival rate for cars approaching lane i
$n_p(t)$	Number of stationary cars in lane i at time t
ℓ_p	Distance to lane i from the previous intersection/boundary
v	Desired vehicle speed (assumed equal for all vehicles)
$w_p(t^*, \Delta t)$	Total waiting time for vehicles in lane i during the interval $[t^*, t^* + \Delta t]$
$Q_p^{\text{arr}}(t)$	Number of cars arriving at lane i
$Q_p^{\text{dep}}(t)$	Number of cars departing from lane i

Let us consider the arrival times of cars on lane i, assuming that the cars are entering the network from the boundary. They are determined by function $Q_p(t)$, that can be modelled by a Poisson process with rate λ_p. We assume that we are able to observe incoming traffic well ahead of its arrival, by sensors that are placed at a distance ℓ_p ahead of the intersection. With the assumption that vehicles travel at a constant speed v, it is clear that the approaching vehicle will reach the intersection $T_p := \ell_p/v$ seconds after it is observed. Thus, we define the intersection arrival function

$$Q_p^{\text{arr}}(t) := Q_p(t - T_p). \tag{6.1}$$

We note that in [3] it is assumed that traffic light schedules are fixed at least 30 seconds ahead of time. This means that for each intersection $T_p \geq 30$, otherwise it would not be possible to incorporate vehicle arrival information into future schedules. The total number of vehicles to have arrived at lane i by time t is given by

$$N_p^{\text{arr}}(t) = \int_{-\infty}^{t} Q_p^{\text{arr}}(s)\mathrm{d}s. \tag{6.2}$$

Similarly, the total number of vehicles to have departed lane i by time t is

$$N_p^{\text{dep}}(t) = \int_{-\infty}^{t} Q_p^{\text{dep}}(s)\mathrm{d}s, \tag{6.3}$$

where $Q_p^{\text{dep}}(t)$ is determined by the controls of Smart Traffic. The number of stationary vehicles in lane i at time t is therefore $n_p(t) = N_p^{\text{arr}}(t) - N_p^{\text{dep}}(t)$ (see Fig. 6.1).

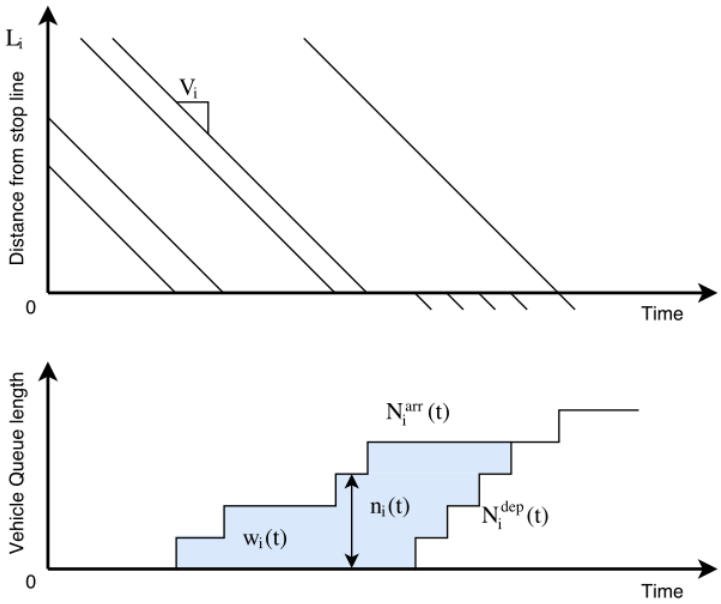


Figure 6.1: (Top) Process of vehicles arriving at an intersection. (Bottom) Cumulative wait-time for vehicles in an intersection. [Image taken from [3]].

6.2.2.1 Cost function

Smart Traffic is built on the concept of minimising the wait-time of traffic until the last time the lights changed. The total wait-time for traffic in lane p over the time interval $[t^*, t^* + \Delta t]$ is given by

$$\langle w_p \rangle (t^*, \Delta t) = \int_{t^*}^{t^*+\Delta t} n_p(t)\mathrm{d}t. \tag{6.4}$$

However, Smart Traffic insists on not only minimising waiting times for the whole intersection but for individual vehicles, such that we want to minimise the *combined* cost of the wait-time for all vehicles over *all* lanes of an intersection. More specifically, the goal is to avoid having one lane waiting for a long time even if it would decrease the total waiting time for the whole intersection. This can be implemented by minimising a *strictly* convex cost function function that depends on the waiting times of individual cars, or less accurate but with reduced book keeping, on the individual lanes, that penalises excessively

long waiting times of a car or of a lane[8]. One can argue that the precise cost function could take into account experimental psychology measuring the stress of drivers, but the simplest approach is to minimise the square of the waiting times of cars in the lanes. This still ensures that, as an extreme example, if one lane has a high traffic throughput, while a crossing lane has only one car waiting, eventually the *squared* cost of the wait-time of the single-car will outweigh the combined effect of multiple cars in the other lane. Hence the algorithm will eventually allow the cars to travel through the intersection.

We consider a fixed lane p_0 as a FIFO (First In, First Out) queue, and compute the squared waiting time in terms of the occupation number $n(t)$ (here as below we will temporarily suppress the lane p_0 from the notation). Assume we start at $t = t_0$ with an empty lane and that cars arrive at $t_0 < t_1^{(\mathrm{a})} < t_2^{(\mathrm{a})} < \ldots < t_{N^{(\mathrm{a})}}^{(\mathrm{a})}$ and leave at $t_1^{(\mathrm{d})} < t_2^{(\mathrm{d})}, \ldots t_{N^{(\mathrm{d})}}^{(\mathrm{d})} < t^*$. Car number i can not leave before it arrives so $t_i^{(\mathrm{a})} \leq t_i^{(\mathrm{d})}$ (i.e. a lane is a FIFO queue). Moreover we cannot have more cars leaving then arriving so $N^{(\mathrm{a})} \geq N^{(\mathrm{d})}$. The squared waiting time from $t = t_0$ when there were no cars waiting, up to time t^*, is given by

$$\langle w_i^2 \rangle(t_0, t^*) = \sum_{i=1}^{N^{(\mathrm{d})}} (t_i^{(\mathrm{d})} - t_i^{(\mathrm{a})})^2 + \sum_{j=N^{(\mathrm{d})}+1}^{N^{(\mathrm{a})}} (t^* - t_j^{(\mathrm{a})})^2 \tag{6.5}$$

$$= \langle w^2 \rangle(t_0, t_{N^{(\mathrm{d})}}^{(\mathrm{a})}) + \int_{t_{N^{(\mathrm{d})}}^{(\mathrm{a})}+\epsilon}^{t^*} (t^* - t)^2 \mathrm{d}n(t) \tag{6.6}$$

$$= \langle w^2 \rangle(t_0, t_{N^{(\mathrm{d})}}^{(\mathrm{a})}) + \int_{t=t_{N^{(\mathrm{d})}}^{(\mathrm{a})}+\epsilon}^{t^*} (t^* - t)^2 \mathrm{d}(n(t) - n(t_{N^{(\mathrm{d})}}^{(\mathrm{a})} + \epsilon)), \tag{6.7}$$

where $\epsilon > 0$ is so small, that $t_{N^{(\mathrm{d})}}^{(\mathrm{d})} + \epsilon < t_{N^{(\mathrm{d})}+1}^{(\mathrm{a})} < t^*$ (assuming such an arrival time $< t^*$ after the last departure time $< t^*$ exists: the integral is zero for any ϵ such that $t_{N^{(\mathrm{d})}}^{(\mathrm{d})} + \epsilon < t^*$ if $N^{(\mathrm{a})} = N^{(\mathrm{d})}$ as it should) Technically, the integral is a Stieltjes integral, but no harm is done pretending it is the limit of smooth approximations of the step function $n(t)$.

We can now do partial integration with the boundary terms vanishing because of the

[8]In computer science this is called balancing throughput and latency. It ensures e.g. that a video stream which requires that IP packets get through with low latency does not stop when someone using the same WIFI access point downloads a long file for which it is optimal that as many IP packets get through at once as possible.

rewrite in the last line. This gives

$$\langle w^2\rangle(t_0,t^*) - \langle w^2\rangle(t_0,t^{(\mathrm{a})}_{N^{(\mathrm{d})}}) = -\int_{t^{(\mathrm{a})}_{N^{(\mathrm{d})}}+\epsilon}^{t^*} (n(t) - n(t^{(\mathrm{a})}_{N^{(d}}+\epsilon)\mathrm{d}(t^*-t)^2 \tag{6.8}$$

$$= -\int_{t^{(\mathrm{a})}_{N^{(\mathrm{d})}}}^{t^*} (n(t) - n(t^{(\mathrm{a})}_{N^{(\mathrm{d})}}+)\mathrm{d}(t^*-t)^2 \tag{6.9}$$

$$= 2\int_{t^{(\mathrm{a})}_{N^{(\mathrm{d})}}}^{t^*} (t^*-t)(n(t) - n(t^{(\mathrm{a})}_{N^{(\mathrm{d})}}+))\mathrm{d}t, \tag{6.10}$$

where $n(t^{(\mathrm{a})}_{N^{(\mathrm{d})}}+)$ is a shorthand for $n(t^{(\mathrm{a})}_{N^{(\mathrm{d})}}+\epsilon)$ with ϵ as above (i.e. the number of cars in the lane right after the last car that left before t^*). Thus, the additional square waiting time absorbed in the interval $[t^*, t^*+\Delta t)$ is

$$\langle w^2\rangle(t^*+\Delta t,t_0) - w^2\rangle(t^*,t_0) = 2\int_{t^*}^{t^*+\Delta t} (t^*-t)n(t)\mathrm{d}t \tag{6.11}$$

$$+ 2\int_{t_{N^{(\mathrm{d})}}}^{t^*+\Delta t} (\Delta t)(n(t) - n(t^{(\mathrm{a})}_{N^{(\mathrm{d})}})\mathrm{d}t. \tag{6.12}$$

Under the integral in the first term, $-\Delta t < (t-t^*) < 0$, so is negligible (and negative) for $\Delta t \ll (t^* - t^{(\mathrm{a})}_{N^{(\mathrm{d})}})$. Hence

$$\langle w^2\rangle(t^*+\Delta t,t_0) - \langle w^2\rangle(t^*,t_0) = 2(t^*-t_)(n(t^*) - n(t_{N^{(\mathrm{d})}})\Delta t + O((\Delta t)^2)$$

and the per unit time increase of square time is

$$\frac{\mathrm{d}}{\mathrm{d}t^*}\langle w^2\rangle(t^*,t_0) = 2(t^* - t_{N^{(\mathrm{d})}})\left(n(t^*) - n(t_{N^{(\mathrm{d})}})\right).$$

In [3], the optimisation performed by Smart Traffic is not explicitly given. We therefore take as the cost function of lane p at time t^*, the additional squared waiting time in the linear approximation in Δt derived above[9] , up to the irrelevant factor 2 (we keep the factor Δt for clarity) .

$$C_p(t^*,t^*+\Delta t) = \left(t^* - t^{(\mathrm{a})}_{p,N^{(\mathrm{d})}_p}\right)\left(n(t^*) - n(t^{(\mathrm{a})}_{p,N^{(\mathrm{d})}_p})\right)\Delta t. \tag{6.13}$$

The total cost for a particular light setting of all lanes of an intersection X with lanes $p \xrightarrow{\text{☹}} X$ that get red is then

$$C^X_{\text{☹}}(t^*,t^*+\Delta t) := \sum_{p\xrightarrow{\text{☹}}X} C_p(t^*) = \sum_{p\xrightarrow{\text{☹}}X} \left(t^* - t^{(\mathrm{a})}_{p,N^{(\mathrm{d})}_p}\right)\left(n(t^*) - n(t^{(\mathrm{a})}_{p,N^{(\mathrm{d})}_p})\right)\Delta t. \tag{6.14}$$

[9]This is equivalent to considering a cost per unit time

For a traffic light schedule to be updated at time t, Smart Traffic will compute the configuration with minimal cost C for the interval $[t^*, t^* + \Delta t]$, where $t^* = t + 30$, and $\Delta t \ll 30$. Such that Smart Traffic appends an interval of Δt onto the existing schedule, and updates it predictions on the number of cars arriving and leaving in the interval $[t, t*]$ based on the new information available at the sensor loops. Under our simplifying assumptions, however, cars behave predictable given a choice of traffic light settings.

6.2.3 Model Extension

To extend the utility of Smart Traffic to a network, we take the situation at other intersections into account. This necessarily implies some communication between neighbouring intersections. It was discussed with Sweco what type of information could be shared between intersections. They strongly advised against a global control mechanism for all lights in the city network, i.e. that all intersections are controlled by a single big server that takes into account the situation in the whole city, or alternatively, that each smaller server for an intersection has to take into account the algorithmic "decisions" of the smaller servers for all the other intersections. It was decided, however, that intersections, could, share information regarding the expected wait-times for cars that it will receive, to upstream neighbours. The simplest kind of information passed to a neighbouring intersection would be the running average of observed wait-times for a car approaching from the intersection's direction, or only slightly more difficult some Karman filter type extrapolation and noise filtering.

For example, if an intersection has only one car waiting then uncoupled Smart Traffic would generally let it leave immediately. However, if the downstream intersection is able to communicate that wait-times for traffic will be high by the time the car arrives, it may be more beneficial for the first intersection to keep the car waiting for a little longer until downstream traffic is alleviated.

Note that there is a subtlety here: when a car leaves an intersection it is unknown which lanes a car will choose at the next intersection, i.e. in which direction a car will continue its trip. Moreover, the street network of the city may allow cars to turn up at different intersections. One can, however, estimate the probability for a car departing at time t_1 at intersection X on lane p, to arrive at intersection Y at lane q on time t_2 from previous behaviour[10].

Thus, if, at time t^*, each intersection X has expected wait-times for downstream traffic, we want to balance the cost between keeping cars waiting *now* against them perhaps waiting longer further downstream. To implement this we want define a new

[10] Aggregated gps data from car drivers would be particularly useful here

cost function C^X of a particular lights setting of the intersection X that also takes into account some future waiting time cost for the lanes $p \xrightarrow{\text{☺}} X$ that get a green.

$$C^X(t^*, t^* + \Delta t) = C^X_{\text{☺}}(t^*, \Delta t) + C^X_{\text{☺}}(t^*, t^* + \Delta t). \tag{6.15}$$

The cost function for the green lanes should take into account the chance to arrive at a lane of a different intersection, the additional squared waiting times at that downstream lane, some discounting for events the farther away they are in the future, and for good measure some fudge factors that allows for tuning the importance of waiting at different intersections allows for a bit of tuning. This gives

$$C^X_{\text{☺}}(t^*, t^* + \Delta t) = (\Delta t) \sum_{p \xrightarrow{\text{☺}} X} n_p^X(t^*) \sum_{Y \neq X} a^Y_{X,p} \int_{t^*}^{\infty} \mathrm{d}t D(t, t^*) \sum_{q \to Y} p(Y, q; t | X, p; t^*) W_q^Y(t), \tag{6.16}$$

where the sum is over all lanes of the intersection X and each intersection $Y \neq X$ in the network, each given a weight a^X_Y, over all lanes q of that intersection. The integral $\int_{t_0}^{t_1} p(Y, q; t | X, p; t^*)\mathrm{d}t$ is the chance that a car leaving from lane p of intersection X at time t^* will arrive at lane q of intersection Y, between t_0 and t_1, and $W_q^Y(t)$ is the estimated waiting time for lane q. This sum is very general but a little unwieldy, so we can make some simplifying assumptions.

- We assume that all lanes of an intersection have equal weight as far as costs are concerned and is non zero only if a lane directs to a neighbouring intersection Y[11] i.e.

$$a^Y_{X,p} = \begin{cases} a & \text{if } p \to X \to Y \\ 0 & \text{otherwise} \end{cases} \tag{6.17}$$

- We assume that the discount function has a finite time window and/or a fairly steep descent with respect to planning period (30 seconds) so waiting time estimates are not too critical far in the future, e.g. $D(t, t*) = \exp((t - t^*)/\theta)$ with θ 1min.

- the probability to arrive at a lane only depends on the time difference $t - t^*$ and the distance $d(X, Y)$ between two (directly neighbouring) intersections provided lane p on X directs to lane q on Y. More over each lane q of an intersection Y reachable from lane p (through X) is equally probable. Hence

$$p(Y, q; t | X, p; t^*)\mathrm{d}t = \begin{cases} \frac{1}{(\#p \to q \to Y)} p_{d(X,Y)}(t - t^*)\mathrm{d}t & \text{if } p \to X \to q \to Y, \\ 0, \end{cases} \tag{6.18}$$

[11] These time independent parameters may be good candidates for using machine learning for the most satisfying results

where $p_{d(X,Y)}(t-t^*)\mathrm{d}t$ is some 1 parameter family of probability distributions over time. In fact simplifying even further we can make the assumption of cars moving in platoon at speed v after a delay τ and use the degenerate delta function probability distribution

$$p_{d(X,Y)}(t-t^*)\mathrm{d}t = \delta(t-t^*-d(X,Y)/v-\tau)\mathrm{d}t. \tag{6.19}$$

With these assumptions the cost function simplifies to

$$\begin{aligned} &C^X_{\text{☺}}(t^*, t^*+\Delta t) \\ &\quad = a\Delta t \sum_{p \xrightarrow{\text{☺}} X} n_p^X(t^*) \sum_{p\to q\to Y} \int_{t^*}^{\infty} \mathrm{d}t e^{-(t-t^*)/\theta} W_q^Y(t)\, \frac{p_{d(X,Y)}(t-t^*)}{(\#p\to q\to Y)}, \end{aligned} \tag{6.20}$$

or simplifying even further using the degenerate distribution over time

$$C^X_{\text{☺}}(t^*, t^*+\Delta t) = a\Delta t \sum_{p \xrightarrow{\text{☺}} X} n_p^X(t^*) \sum_{p\to q\to Y} e^{-(d(X,Y)/v+\tau)/\theta} \frac{W_q^Y(t^*+d(X,Y)/v+\tau)}{(\#p\to q\to Y)} \tag{6.21}$$

Finally to make a decision on a plan for traffic lights for the next T = 30 seconds and take into account account the rules for changing the lights. Toy rules would be

- lights stay green for at least 1 second,
- lights stay red for at least 3 seconds,
- an intersection must block an opposite lane for at least 10 seconds before a crossng lane can get green

Under these rules we have to minimise total squared waiting time[12]

$$C^X(t,T) = \sum_{n=0}^{T/(\Delta t)-1} C(t+n\Delta t, t+(n+1)\Delta t). \tag{6.22}$$

Where $\Delta t = 0.1$ second should be amply good enough resolution.

6.2.3.1 MATLAB Implementation

For a four-way intersection, with each road having three lanes (see Fig. 6.2) there are many light configurations that conflict with one another; for example, in the figure shown Lanes 2 and 7 cannot both be relieved at the same time.

[12]This is essentially a poor numerical integration and using Simpson's or the trapezoidal rule and subtracting half the boundary values might be appropriate

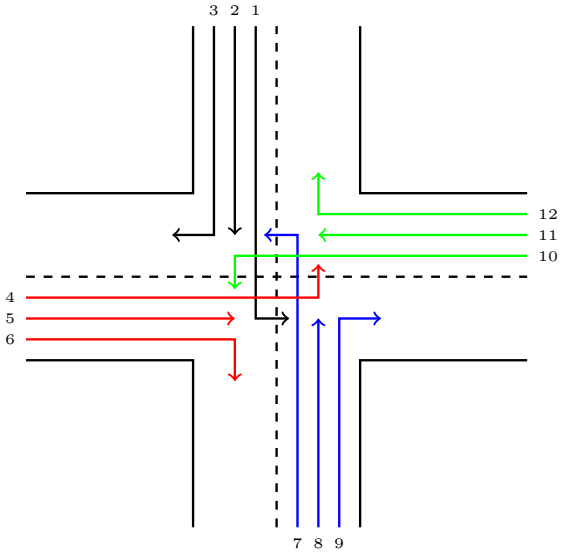


Figure 6.2: Four-way intersection with twelve lanes.

There are roughly 100 light configurations that do not cause conflict, which can be described by a binary vector of length 12, with green lights corresponding to a 1, and red lights corresponding to a 0. For a single intersection, we define the optimisation space to be

$$\mathcal{S} := \{\mathbf{v} \in \mathbb{B}^{12} \ : \ \mathbf{v} \text{ causes no conflicts}\}, \tag{6.23}$$

where $\mathbb{B}$ is shorthand for the Booleans $\mathbb{Z}_2$. Given that each intersection will be governed by separate instances of Smart Traffic, the optimisation space for the entire network will therefore be a product of $\mathcal{S}$, which, for a network of M intersections, we will define as

$$\mathcal{S}^M := \{\mathbf{v} \in \mathbb{B}^{12M} \ : \ \mathbf{v} \text{ causes no conflicts}\}. \tag{6.24}$$

Describing the action space of the coupled network in this way allows us to build a model using simple linear algebra. We define $\mathbf{A}$ to be the adjacency graph of the network, $\mathbf{I}$ to be the identity matrix (the same size as A), and $\mathbf{n}(t)$ to be the vector holding the number vehicles in each lane at time t. Note that the multiplication $\mathbf{An}$ describes the instantaneous travelling of all vehicles to their desired destination, while $\mathbf{In}$ describes the event that all vehicles remain stationary.

For a particular action $\mathbf{v} \in \mathcal{S}^M$, we create the diagonal matrix $\mathbf{V} := \text{diag}(\mathbf{v})$, such that $\mathbf{AV}$ defines an augmented adjacency matrix corresponding *only* to those lanes which allow traffic through. Similarly, the matrix $\widetilde{\mathbf{V}} := \mathbf{V} - \mathbf{I}$ defines all lanes which are not opened. Therefore, defining the matrix $\mathbf{B} = \mathbf{AV} + \widetilde{\mathbf{V}}$, the multiplication $\mathbf{Bn}$ describes the instantaneous state transition for all vehicles in the model.

For implementation purposes, as we have assumed that vehicle data is perfectly accurate, we only need to look at discrete events that occur in the model, i.e. new vehicles entering the model from the boundary, and cars arriving at or leaving an intersection. Given that we assume that all vehicles travel at a fixed speed, and all road lengths are known, we can therefore predict all future events.

By using an event-based model, we therefore need to keep track of vehicles that are stationary at any point, and vehicles that are moving between intersections. This is done in such a way that a car released from an intersection can be described by a Heaviside function for arrival at the downstream intersection. This means that rather than using $\mathbf{A}$ to describe the *instantaneous* travel of a vehicle to its destination, we can instead make it a time-dependent Heaviside matrix $\widetilde{\mathbf{A}}(t)$ which keeps track of when vehicles will be arriving downstream, such that we can compute wait-time effects from cars arriving between successive schedules.

Let us assume that the model has some configuration of stationary and moving vehicles, as well as knowledge of vehicles approaching from the network boundary. Every Δt seconds, Smart Traffic will amend the network schedule by optimising traffic flow over the interval $[t^*, t^* + \Delta t]$. As we have previously described, the total wait-time for traffic at a single lane during this interval is merely the integral given by Eq. 6.4. In vector form, we have

$$\mathbf{w}(t^*, \Delta t) := \int_{t^*}^{t^*+\Delta t} \mathbf{n}(t)\mathrm{d}t, \tag{6.25}$$

$$= \int_{t^*}^{t^*+\Delta t} \widetilde{\mathbf{V}}\mathbf{n}(t^*) + \widetilde{\mathbf{A}}(t)\mathbf{1}\mathrm{d}t \tag{6.26}$$

$$= \Delta t\widetilde{\mathbf{V}}\mathbf{n}(t^*) + \int_{t^*}^{t^*+\Delta t} \widetilde{\mathbf{A}}(t)\mathbf{1}\mathrm{d}t. \tag{6.27}$$

Therefore, for each entry in the vector $\mathbf{w}$, the wait-time is comprised of all cars that are not released at time t^*, plus all cars that arrive before $t^* + \Delta t$. Given that we want to minimise the squared wait-times, this is equivalent to minimising the 2-norm of $\mathbf{w}$.

In the extended model, we also want to balance the cost of sending cars downstream, assuming that we have knowledge of expected future wait-times, which we hold in a diagonal expectation matrix $\mathbf{E}(t^*)$. Therefore, the expected cost of releasing vehicles due to an action $\mathbf{v}$ is merely

$$\mathbf{e}(t^*) := \mathbf{E}(t^*)\mathbf{A}\mathbf{V}\mathbf{n}(t^*). \tag{6.28}$$

Thus, every Δt seconds we must solve the following optimisation problem:

$$\min_{\mathbf{v}\in\mathcal{S}^M} \{\|\mathbf{w}(t^*, \Delta t)\|_2 + \|\mathbf{e}(t^*)\|_1\}. \tag{6.29}$$

Due to the fact that we do not explicitly couple the intersections together, the above problem is equal to n instances of the smaller problem

$$\min_{\mathbf{v}_j \in \mathcal{S}} \{\|\mathbf{w}_j(t^*, \Delta t)\|_2 + \|\mathbf{e}_j(t^*)\|_1\}, \tag{6.30}$$

where the subscript $j = 1, \ldots, M$ corresponds to each particular intersection.

The benefit of framing the problem in this way is that we are able to explicitly model *all* events in the coupled model, yet the optimisation scheme is completely separable. Thus, the computational complexity grows linearly with M. While the implementation seems very simple, it was not possible to get a working version of the model in MATLAB in only three days. While MATLAB is built for matrix operations, the utility of a Heaviside matrix was problematic, and made computation times extremely high for even simple integrals like those in Eq. 6.27. Therefore, we were not able to implement an optimisation scheme that had any chance of terminating in a reasonable time. Given that we need to optimise every Δt seconds, this meant that no useful results could be obtained, as the model that was implemented was not efficient enough to replicate the workings of Smart Traffic.

6.3 Other Approaches

6.3.1 Markov Decision Process

In some interpretation of the Smart Traffic model of Sweco, it seems that the mathematical framework of Markov decision processes fits the problem well. In general, a Markov decision process, or MDP for short, consists of the following. We start with a countable state space S. Time is discrete, indexed, say, by N. In time step t, the process is in some state $s \in S$. In every state, there is a set of actions $A(s)$ to choose from. If some action $a \in A(s)$ is chosen, there is an immediate reward of $r_s^t(a)$ and the process transitions to state j with probability $p_{sj}^t(a)$. The reward may be negative, indicating a cost instead. If the reward and transition probabilities are independent of the time t, the model is called stationary. For a more detailed description of the model, as well as an explanation of some optimization methods in this framework, see the lecture notes on MDPs by Kallenberg [4] or the book on reinforcement learning by Barto and Sutton [6].

For the traffic model, one could choose as time steps the intervals between the times where the calculation of an optimal policy is started, e.g., Δt. The state space, unfortunately, is hard to define. Ideally, one would only have to keep track of the amount of cars waiting at every traffic light, that is, $n_p(t)$ for every lane p. However, cars that are currently travelling between intersections also need to be taken care of. Theoretically, it

is possible for every car to keep track of its exact location and speed. However, this would cause the state space to explode in size. Therefore, one could propose to only count the number of cars present in each lane of each intersection, and to count the number of cars travelling between intersections, disregarding their exact whereabouts.

As action set in every state, we can take the set of all different configurations of green lights for the intersection, or a choice between continuing the current green scheme or changing it. The reward for every action is somehow dependent on the expected waiting times of the cars in the system, being a summation of the $w_i(t, \Delta t)$ over the lanes i, given the action chosen. One could choose, for example, the cost function described in Section 6.2.2. Finally, the probability of transitioning from state s to state j given some action a is largely deterministic, apart from cars that might enter the system with some probability at the borders. Moreover, having discarded the exact location of the cars travelling between intersections, their movement may also taken to be stochastic.

As we assume the computations always being done for a finite horizon, e.g., 30 seconds, one might use recursive methods to solve for the optimal policy, such as dynamic programming. Starting at the horizon, we move backwards in time, trying to find in time step $t-1$ an action which optimizes $r_s(a) + \sum_j p_{sj}(a)x_j^t$, where x_j^t is the reward we obtain when moving optimally from time step t up until the horizon. However, the size of the state space may in practice be a complicating factor for computing an optimal policy exactly. One might therefore resort to approximate methods, some of which are also described in the referenced works.

6.3.2 Model Predictive Control

As an alternative to Markov Decision support we mention the framework of model predictive control (MPC) that utilizes Pontryagin's Minimimum Principle as a guide to optimality. Formulations are available for both discrete and continuous time models [2]. Given a traffic model in (continuous-time) state space format, i.e.

$$\frac{dx}{dt} = f(x(t), u(t)), \quad x(t_0) = x_0 \tag{6.31}$$

with $x(t)$ the state vector (e.g. traffic volumes) and $u(t)$ the manipulated variables (e.g. traffic light settings). Since traffic lights are switched on/off, one should include this in the underlying equations and choose for a hybrid type of system description where both continuous and discrete time variables are possible [1]. If the cost is now formulated as in 6.4 for a time horizon Δt, we can formulate the following Hamiltonian

$$\mathcal{H}(x(t), u(t)) = n_p(t) + \lambda^T(t) \cdot f(x(t), u(t)) \tag{6.32}$$

with $n_p(t)$ the number of stationary cars at time t, and $\lambda(t)$ the co-state or *influence* function. Pontryagin's minimum principle states that amongst all choices for $u(t)$, the one that minimizes the Hamiltonian $\mathcal{H}(x, u)$ is the optimal choice. It is only in rare cases that this optimal strategy can be stated in terms of $x(t)$ as a feed-back law. More common is to solve the optimization problem as a two-point-boundary-value problem (TPBV) with the state $x(t)$ satisfying 6.31 at $t = t_0$, and the co-state $\lambda(t)$ satisfying

$$\frac{d\lambda}{dt} = -\frac{\partial \mathcal{H}}{\partial x}, \quad \lambda(t_f) = \phi_x(t_f) \tag{6.33}$$

where ϕ reflects the terminal cost at time $t = t_f$, e.g. the number of cars that are stationary at $t = t_f$. Efficient software is available to solve the TPBV problem. Typically, the first run of the software takes some time but once an optimal strategy has been found it is computationally efficient to *adapt* such a strategy every 30 seconds or so, when new data of the current traffic situation becomes available. Unfortunately, software of the current traffic system applied by SWECO was not available and therefore the optimization problem could not be encoded en solved in Matlab under the time-constraint of several days (the duration of the SWI workshop)..

6.3.3 Predicting behaviour of neighbouring intersections

In order to accurately predict the future state of the network, it is useful to predict the behaviour of neighbouring intersections. This will give us a better idea of how much traffic will be arriving at our intersection, and of how long cars we send to one direction will have to wait down the road. Let S be the state space as in Section 6.3.1, and let A_i denote the set of all possible actions of intersection I_i (here an action is a configuration of traffic lights at intersection I_i, so A_i does not depend on the current state). For simplicity, we restrict ourselves to two intersections, I_1 and I_2, that are connected by a road with no intersection in between I_1 and I_2. For larger networks it is possible to apply the described method on each neighbour separately. We will take the point of view where we want to predict the future actions of I_2 to plan the actions of I_1. For each intersection, Smart Traffic keeps track of and updates a planning P_i of actions that the intersections will perform. This planning will be updated at times $t_0, t_1, \ldots$, where $t_i = t_0 + i\Delta t$. At time t_i, the actions that will happen at times $t_i, \ldots, t_{i+k}$ have already been determined for some fixed k, and the method has to decide which action to take at time t_{i+k+1}. We assume that the traffic light configuration will change at most once during each time step. Different intersections do not have access to each other's planning. This means that at time t_i we would like to predict all the actions the neighbouring intersections perform between t_i and t_{i+k+1}. A general procedure to do this is given in Algorithm 2. One way

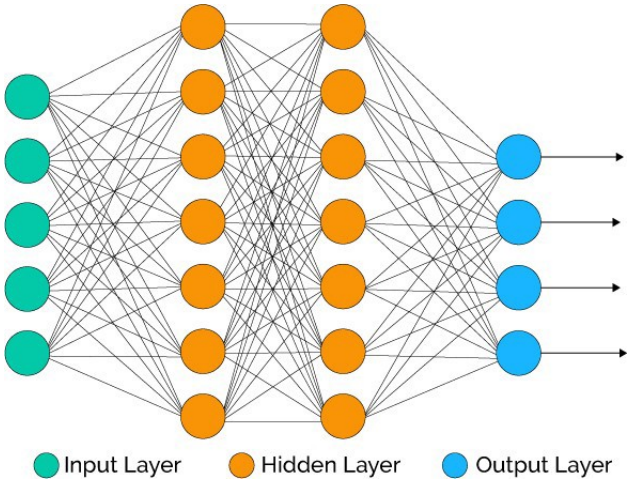


Figure 6.3: A small neural network.

to predict each distribution of actions is by using a neural network, we will discuss this method in the next section.

Algorithm 2:

input : State s_{t_i}
output: Distributions of actions $\tilde{a}_i, \ldots, \tilde{a}_{i+k+1}$

$\tilde{s}_{t_i} \leftarrow s_{t_i}$;
for $j \in \{i, i+1, \ldots, i+k+1\}$ **do**
    Predict $\tilde{a}_j$ given $\tilde{s}_{t_j}$;
    Compute expected state $\tilde{s}_{t_{j+1}}$;
endfor

6.3.3.1 Neural networks

In this section we give the basic ideas behind neural networks, and how we can use them for our problem. We will not go in-depth and often omit explicit formulas, so it will probably be wise to do some more research on the subject before attempting to write an implementation. Luckily, many resources are available, see for instance `https://www.edx.org/course/deep-learning-explained-3`. A neural network consists of a large number of nodes, divided into different layers. Each node in one layer is connected to every node in the neighbouring layers. See Figure 6.3 for an illustration. Each connection has an associated weight. When we give each node in the input layer a value, we can use the weights to compute values for the nodes in the next layer, and repeat this process until

the values of the nodes in the output layer are computed. To apply a neural network to our problem, we let the input nodes correspond to the network state s (e.g. have a node for each lane that corresponds to the current waiting time in that node), and each output node corresponds to an action $a \in A_2$. The idea is to set the weights of the connections in such a way that when the input is set according to some state s, the values of likely action will be high and the values of unlikely actions will be low. The process of adjusting the weights to achieve this goal is referred to as training the network. The usual technique for doing this is called back-propagation. This method works as follows: given some state s, we use the current neural network to get a value for each action. We then observe which action a actually takes place. The output value corresponding to a should have been high, and all the others should have been low. So the network can be improved by increasing the weights of connections to the node of a that helped increase the value corresponding the a, and lowering the weights of connections to the other nodes that helped increase their values. Then we can compute what the values of the nodes in the second to last layer should have been, and adjust the weights of the connections between the third to last and the second to last layer based on that. Repeat this process until every connection is updated. Note that this is a data driven approach, since in order to apply backpropagation we need to know which action will be taken given the state we use as input. It will be necessary to collect training data, both before the neural network is implemented to train a initial version, and while the network is active such that it can adapt to changing circumstances (such as changes to the systems that control the neighbours).

6.4 Conclusions and Recommendations

We have suggested several routes to a solution to the complex problem of controlling a network of intersections, given the strategy currently employed by Sweco. Much to our regret we have not been able to thoroughly test our hypotheses and models because of time-constraints and the limited availability of currently used software. Yet, some good suggestions were made on how to tackle this problem from various angles. Most important is to have the current strategy become less greedy in finding a solution for the current situation by including an *expected waiting time* for the downstream intersection into the problem. This could be implemented via, for example, model predictive control in which the goal function is an essential part of the control strategy that is found on the basis of Pontryagin's Minimum Principle. For numerical solution a receding horizon optimal control problem needs to be solved on-line with the expected waiting time of the down-stream traffic included in the goal function. This is certainly within reach since

advanced software packages are available (in Matlab) that are capable of solving these kind of problems. Other solutions, such as Markov decision processes and neural networks were also suggested as a means to find an optimal strategy.

Bibliography

[1] Branicky, M.S., Introduction to Hybrid Systems, Chapter in the handbook of networked and embedded control systems, Birkhäuser Boston, 2005

[2] Bryson, A.E., Dynamic Optimization, Addison Wesley, 1999

[3] van Hout, T.C.G., Smart Traffic: Anticipative flexible traffic light control and network stability, Master's thesis, Eindhoven University of Technology, 2017.

[4] Kallenberg, L.C.M., Markov Decision Processes (Manual), Leiden University, 2009

[5] Lämmer, S. and Helbing, D., Self-control of traffic lights and vehicle flows in urban road networks, Journal of Statistical Mechanics: Theory and Experiment, 2008, IOP Publishing

[6] Barto, A.G. and Sutton, R.S., Reinforcement Learning: An Introduction, MIT Press, Cambridge (MA), second edition, 2018