Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Engineering
Department of Mechanical and Industrial Engineering

Karsten Standnes

Numerical Simulation Comparing the Burridge-Knopoff and Burridge-Knopoff-Pad Model

Master's thesis in Engineering and ICT
Supervisor: Astrid de Wijn and Bjørn Haugen
June 2019

NTNU
Kunnskap for en bedre verden

Karsten Standnes

Numerical Simulation Comparing the Burridge-Knopoff and Burridge-Knopoff-Pad Model

Master's thesis in Engineering and ICT
Supervisor: Astrid de Wijn and Bjørn Haugen
June 2019

Norwegian University of Science and Technology
Faculty of Engineering
Department of Mechanical and Industrial Engineering

NTNU
Norwegian University of
Science and Technology

Preface

This Thesis concludes my five-year master's degree at the study Engineering and ICT with specialisation in Product Development and Materials. The code and name of the course is TMM4935 Industrial ICT, Master's Thesis. The Thesis was carried out during the spring semester of 2019 at the Department of Mechanical and Industrial Engineering, NTNU.

I would like to thank my supervisors, Astrid de Wijn and Bjørn Haugen with my deepest gratitude. They have not only freely and engagingly shared and supported with their scientific knowledge but also been highly motivating during the Project and Masters' Thesis.

I would also like to thank my parents, Esther Karin and Tor, as well as my siblings Tobias, Emma and Morten for all the love, laughs and support throughout the project.

Karsten Standnes
Trondheim, 11.06.2019

Summary

This Master's Thesis studies and analyses the output from batch simulations of the Burridge-Knopoff(BK) and Burridge-Knopoff-Pad(BKP) model. The latter is newly developed and largely unstudied, and the goal was to compare it to the well-established BK model further. To achieve a better understanding, the simulation was enabled for batch simulation through the use of separated, human-readable parameters files and simple scripts. These files and scripts control the simulation of the BK and BKP model. Each simulation producing output that was analysed and visualised. These visualisations showed individual behaviour in and caparisons within and between the models. An animation tool was developed for the models to increase the understanding of the models' behaviour while running. The animation, together with the other visualisation tools helped to highlight some of the similarities and difference between the two models. It also helped to discover ideas for further investigation of the models.

Sammendrag

Denne masteroppgaven undersøker og analyserer data produsert fra gruppesimulering av Burridge-Knopoff og Burridge-Knopoff-Pad modellene. Sistnevnte, BKP modellen, er nylig utviklet og lite undersøkt, derfor er målet å fortsette sammenligningen med den veletablerte BK modellen. For å oppnå bedre forståelse av modellene har simuleringen blitt implementert for gruppersimuleringer ved bruk av separerter parameter- og konfigurasjonsfiler som er lesbare for mennesker og enkle dataskript. Disse skriptene og filene kontrollere hvordan BK- og BKP modellene blir simulerte. Simuleringen som er blitt kjørt har produsert datam som videre har blitt analysert og visualisert. Disse visualiseringene viste den individuelle oppførselen av kjøringen, samt forskjeller innad i modellene og iforhold til den andre. Det ble under arbeidet med masteren uviklet et animeringsverktøy av forfatteren, som skulle øke forståelsen av modellene under kjøringen. Animasjon som verktøy hjalp, i likhet med de andre visualiseringsverktøyene med å fremheve likeheter og forskjeller mellom de to modellene. Dette verktøyet bidro også til oppdagelsen av ny idéer for hvordan modellene kunne undersøkes videre.

Table of Contents

List of Figures

List of Tables

Abbreviations

BK	=	Burridge-Knopoff
BKP	=	Burridge-Knopoff-Pad
CI	=	confidence interval
DFT	=	discrete Fourier Transform
GUI	=	graphical user interface
ODOF	=	one degree-of-freedom
PRNG	=	pseudorandom number generator
YAML	=	YAML Ain't Markup Language

Chapter 1

Introduction

1.1 Problem and goal

In this Thesis, the Burridge-Knopoff-Pad(BKP) model has been implemented dynamically, investigated through efficient batch simulations and equipped with suitable analysis and visualisation tools. The implementation is laying the ground for discovering this models application for studying the interplay between at the sliding interface, and the generation of noise in a resonator such as a brake system. To properly study a model, testing and interpretation are important. In this case, looking at how different parameters and configurations affect the model is useful. During the project Thesis by the author, the model was implemented to be prepared for efficient batch simulation. Enabling this put focus on the implementation and tools supporting it. The implementation strives to make a model implementation that is quick to change and review after the fact, as well as to be compatible with modern software development tools. This focus helps to figure out whether the BKP is a valid model for this field of study. The main goal of the Master's Thesis is to perform simulations and investigate the BKP model. Moreover, through this acquire a better understanding of how the pad makes it differ it from the well-established BK model.

1.2 Background and motivation

This Thesis is a continuation of the project report written by the author during the autumn of 2018. The project report was continuing the work in the Master's Thesis of prior NTNU student H. Nylund Ferre[1]. The Thesis looked at the model coined Burridge-Knopoff-Pad, which is a combination of the Burridge-Knopoff(BK) and One Degree-of-Freedom(ODOF) models.

With the Thesis, he wanted to quote " 1. implement and simulate the new combined Burridge–Knopoff–Pad model, and, 2. investigate the results to see if this brings anything new into the study of brake squeal or friction-induced vibrations."

Both the BK and ODOF models are studied before, but before H. Ferre's Thesis, the combination of them was not studied to the authors' knowledge. As concluded in the Thesis, the combined BKP is a step in the right direction regarding realism when studying brake systems — continuing by stating that further study is needed.

1.3 Approach

In order to make a simulation that is efficient to run and enables batch simulation, the implementation and structure are of high priority. The project is constructed with a clear and concise structure to ease the navigation and lower the learning curve. The simulation code is written in C++ with a basis in the code developed by H. Ferre [1]. This implementation is built upon, increasing the flow in the simulation process. To take advantage of running simulations on remote machines, tools such as CMake, command-line scripts and git version-control system are utilised. The command-line scripts enable quick setup for batch runs and synergies well with parameter and configuration files written in YAML. For each

run, both the code, run-specific YAML-files and compiled files are stored. Storing achieves backtracking and repeatability at low disk-space cost, neglectable compared to the storage used by the results. The output of the simulations is stored and then investigated, taking advantage of the parameter file to get information on each run. Appropriate tools developed in Python are then used to visualise and analyse the output, such as animating the model for visual investigation.

First, the results and conclusions of the earlier Thesis were replicated or built upon in order to establish what should be investigated further. Such results also build a ground for comparison and input on what analysis and visualisation tools are best suited. Further, these results are used to reflect on and determine parameters and configurations for new simulations.

1.4 Thesis structure

In Chapter 2, background literature related to the project is presented. Further, Chapters 3 and 4 present the models, methods and the tools used in the Thesis. Chapter 3 is focused on the mathematical representation and methods used, while Chapter 4 covers the programming languages, development and other technological tools. Next, Chapter 5 present the results and what they tell, with some brief discussion. These results are discussed in a bigger context and concluded upon in Chapter 6. Lastly, Chapters A, C and C are appendices providing parameter and configuration files for different simulations, additional figures and link to web-hosted animations.

Chapter 2

Background

2.1 What is friction?

Friction is everywhere and has been a matter of study for a long time. As mentioned in the introduction of *Sliding friction: physical principles and applications*[2] friction is often introduced in early classical mechanic and physics classes. This even if the subject is far from well understood. On a macroscopic level, it enables an object to remain still relative to its resting surface with a slight incline. It is what makes Newton's first law which states "A body continues in a state of uniform rest or motion unless acted upon by an external force." valid[3]. If there were no friction force, the object would have an accelerating motion due to gravity. Leonardo Da Vinci gave the first quantitative theory of friction[4]. From his observations, he gathered that the friction force was proportional to the normal force on the object, but also suggested it was independent of the surface area[4]. This suggestion was later debunked with the knowledge of how surfaces gliding against each other looks on a microscopic level. Even if a surface looks completely flat and smooth to the human eye, it might be a landscape of mountains and valleys at the micrometre-scale[4]. When one of the surfaces starts to move some of these *mountains* from the surfaces collide. Such a model makes friction depending on the normal force and the area of contact. An area which is much smaller than what is apparent.

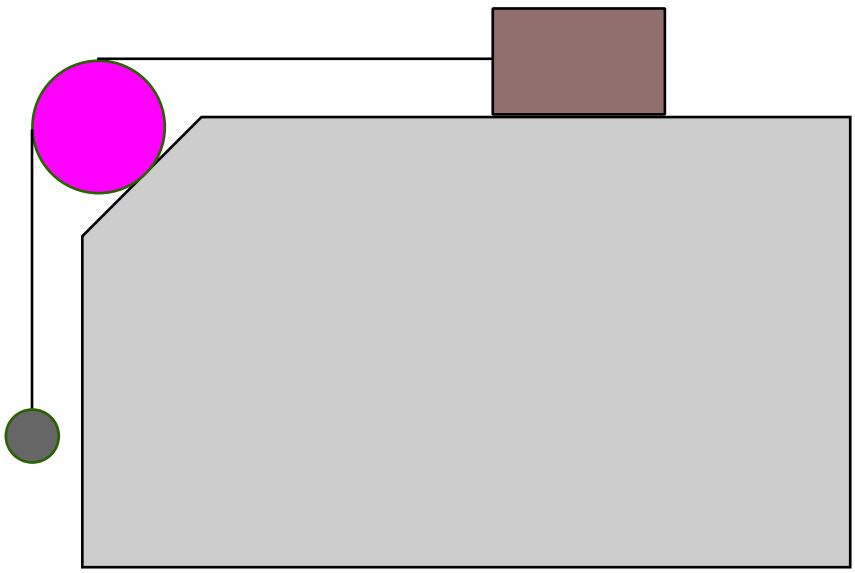


Figure 2.1: Figure showing simple experiment where a constant force is acting on a block resting on a surface. An external force from the weight on the lefthand side pulls on the block through a line. The block keeps still due to friction. Experiment originally performed by Leonardo Da Vinci[4].

As is consensus, friction is split into sliding friction and static friction in this thesis. The names speak

for them self, but two quick rules are that static means the surfaces in contact are still. While sliding refer that they are in relative movement. Some empirical relations have been discovered between them, namely 1. the static friction force is proportional to the force perpendicular on the surface 2. the static friction force is greater than the dynamic 3. the dynamic friction is largely independent. This assumption works reasonably well for very different sized systems of dry friction[5]. In his PhD, B. Huisman states "The mechanisms underlying the friction forces are very different for these sliding systems, and the study of the origins of friction defines the research field of tribology." when commenting how well these assumptions apply in different cases of dry friction[5].

2.2 Burridge-Knopoff Model

The Burridge-Knopoff model was made by L. Knopoff and R. Burridge to study friction[6, 7]. The BK model is a one-dimensional model used to describe two surfaces in contact with the upper gliding over the lower surface with a non-zero relative speed. The upper surface is represented with a row of block masses where each block connects to a plate and its neighbouring blocks with springs as seen in Figure 2.2. Improvement suggestion: Friction from the lower surface works on the blocks gliding over it. The idea is to get better representation than just having one giant block and better reflect the irregularities of real-life surfaces.

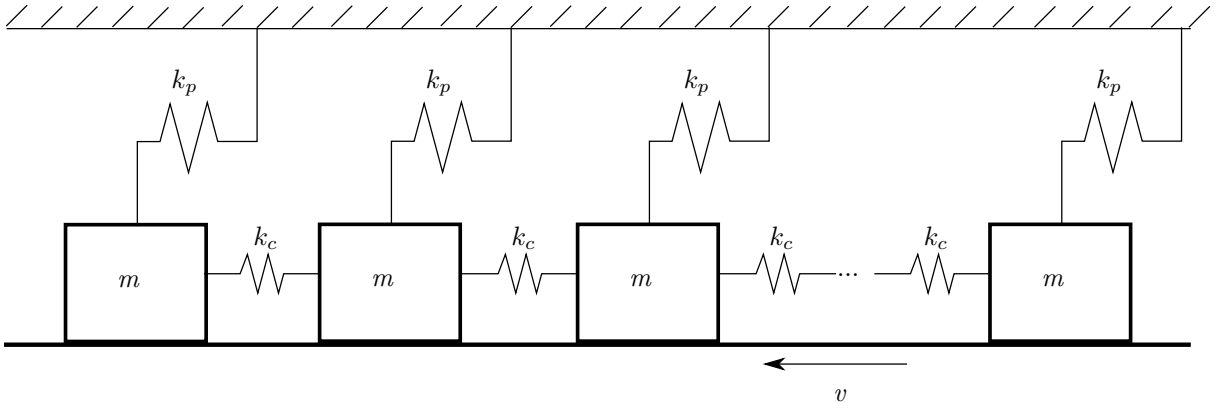


Figure 2.2: Illustration of the Burridge-Knopoff model describing two surfaces where one consists of a row with interconnected blocks, each connected to a upper surface through elastic springs. Made in Inkscape

The model can be represented by Equation 2.1 seen below

$$m\ddot{X}_j = k_c(X_{j-1} - 2X_j + X_{j+1}) - k_p X_j - F(v + \dot{X}_j), \tag{2.1}$$

where X_j denotes the position of block j relative to its equilibrium position. Further, c and p means for *coupling* and *pulling* respectively and is used to differentiate the different spring constants k_c and k_p. F is a friction scheme affected by the velocity if the upper surface v and the relative velocity of the block $\dot{X}_j$. The upper surface movement creates a force on the pulling spring which when exceeding a maximum static friction force F_0 leads to a slipping event. When a braking system uses friction to slow down a surface by transforming kinetic energy to heat it generates noises, the BK model has among other applications been used to study this[1]. Even though the model has been around for a long time, it is still subject for further studies, like Bastian Huisman's PhD Thesis[5]. The Thesis showcase that the BK model has been used for systems of vastly different length scales ranging from tectonic plates to surfaces at the atom level. This is made possible by changing the ratio of the coupling and pulling springs stiffness $\frac{k_c}{k_p}$ and friction scheme F. It can then be applied to systems ranging tectonic plates to atomically small surfaces[5].

2.3 One Degree-of-Freedom Model

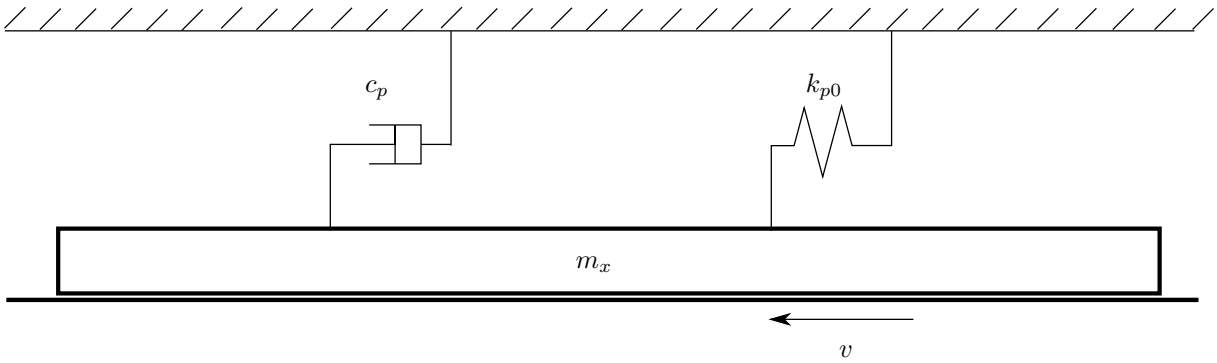


Figure 2.3: A ODOF model consisting of a pad mass, a single spring and a damper.

A one-degree-of-freedom model(ODOF) is a simple way of representing a dry friction oscillator[8]. The model is shown in Figure 2.3, fasten with a damper as well as a spring. The ODOF illustrated is combined with the BK model to constitute the BKP model presented in H. Ferre's Thesis[1]. It has been linked to that ODOF is to simple a model to study brake vibrations and noise, and its use has often might have been motivated to due limited computational power[9]. The notion of the ODOF probably not being suitable for these types of problems is supported by H. Ferre's Thesis.

An equation that is governing an ODOF

$$m_x\ddot{x} = -c_p\dot{x} - F(\dot{x} + v_0) - k_{p0}x \tag{2.2}$$

$$\vec{x} = \begin{bmatrix} x \\ \dot{x} \end{bmatrix} \quad \dot{\vec{x}} = f(\vec{x}) = \begin{bmatrix} \dot{x} \\ \frac{1}{m_x}[-c_p\dot{x} - F(\dot{x} - v_0) - k_{p0}x] \end{bmatrix} \tag{2.3}$$

2.4 Related literature

During the specialisation project, more related literature surfaced. Two papers on snaking bifurcation written by A. Papangelo et al. published in 2017 and 2018[10][11] was of interest. The fact that the Burridge-Knopoff model is not mentioned is interesting, even though there are apparent similarities. These similarities might be due to different branches in science looking at similar problems without the knowledge of each other's efforts.

2.4.1 Snaking bifurcation in a self-excited oscillator chain with cyclic symmetry

The first paper [10] by A. Papangelo et at. studies snaking bifurcations in a chain of mechanical oscillators. The model studied consist of a cyclic system with N oscillators. These are coupled together with springs of constant stiffness k_Δ. Each oscillator has the same mass m and connects to the ground with a linear spring k and a non-linear damper c. The oscillators are non-linear due to non-linear damping terms, which depends on the velocities in the system. These are introduced, quote "bringing into our purely structural model the corresponding non-linear forcing and dissipation terms from surrounding flow, or an involved friction interface."[10]. The model is shown Figure 2.4.

The model is used in the paper to look at the formation of snaking patterns using a bifurcation diagram to obtain a deeper understanding of it. They discuss how this can be used to predict localised non-linear vibration states which are known to be the source issues in engineering and technology; among them, they mention noise. The work is regarded as a first step and starting point for further studies.

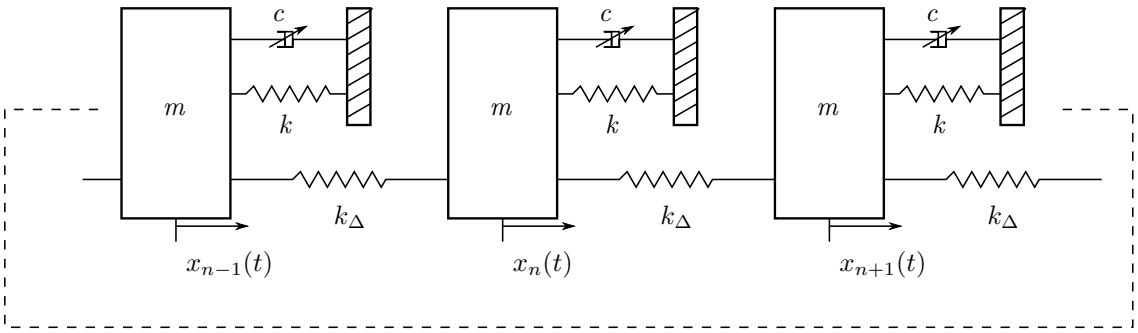


Figure 2.4: Model of the mechanical system studied in the paper[10]. It consist of a system with N non-linear oscillators. Made in Inkscape.

2.4.2 Multiple spatially localised dynamic states in friction-excited oscillator chains

Further, A. Papangelo et al. continues to look at *snaking-like* bifurcation patterns in the second paper[11]. The model used here (Figure 2.5) is similar to the one shown in Figure 2.4, but the oscillating blocks are gliding on a moving belt and pressed by a constant normal force P. The system is a chain of non-linear oscillators. Each oscillator has a mass m, stiffness k, a viscous damping coefficient c and is connected to the neighbouring oscillators by a weak linear spring. The paper concludes by saying the results obtained are relevant for systems experiencing friction-induced vibrations. Moreover, stating that "further work is required to understand how the non-linear mechanism of localisation caused by friction can interact with linear localisation phenomena occurring when the underlying linear system is not homogeneous"[11].

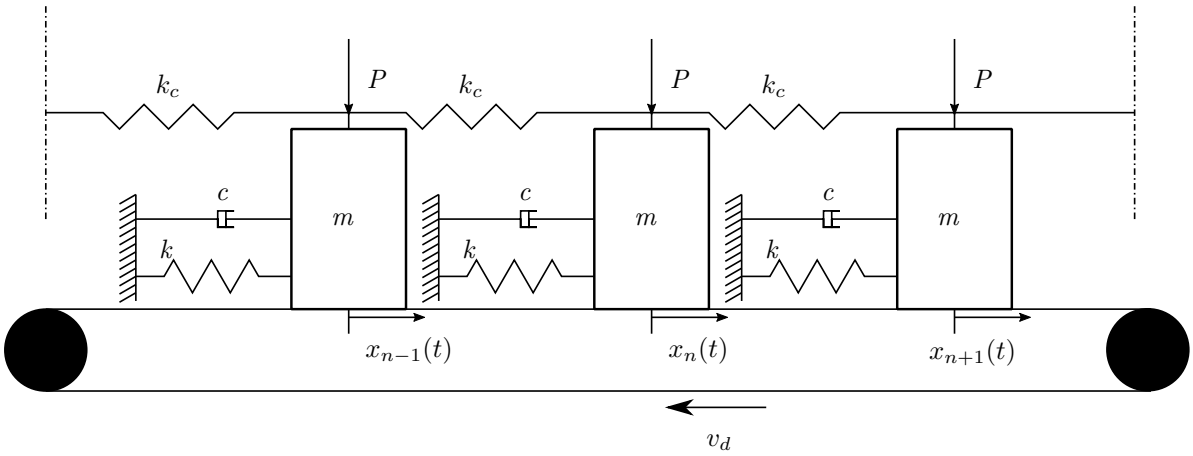


Figure 2.5: Model of the mechanical system studied in the paper[11]. It consist of a system with N non-linear oscillators on a moving belt. Made in Inkscape.

Chapter 3

Implementation

3.1 Burridge-Knopoff-Pad model

The main method studied in this thesis is the Burridge-Knopoff-Pad model. H. Ferre developed this model in his Master's Thesis, which to his knowledge has not been studied before[1]. The BKP model is a combination of the Burridge-Knopoff and one Degree-of-Freedom mode, each explained in Sections 2.2 and 2.3, respectively. The BKP model is illustrated in Figure 3.1. In this illustration, it is clear the upper pad is inherited from ODOF model seen in Figure 2.3 and the blocks from the BK model seen in Figure 2.2. The equations govern the BKP model in Equation (3.1)

$$\begin{aligned} m_u\ddot{u}_j &= k_c(u_{j-1} - 2u_j + u_{j+1}) - k_p(u_j - x) - m_u\phi(v + \dot{u}_j) \\ m_x\ddot{x} &= -c_p\dot{x} + k_p\sum_j(u_j - x) - k_{p0}x \end{aligned} \tag{3.1}$$

consisting of two 2nd ordered differential equations. The pad position is expressed by x, and the blocks position relative to the pad expressed as u_j, where $j = 1, \ldots, N$. N begin the total number of blocks. ϕ given by Equation (3.6) represent the friction law, shown in Figure 3.2. Due to it being a substantial amount of parameters in Equation (3.1), the remaining ones a described in Section 3.2. For a brief overview see Table 3.1.

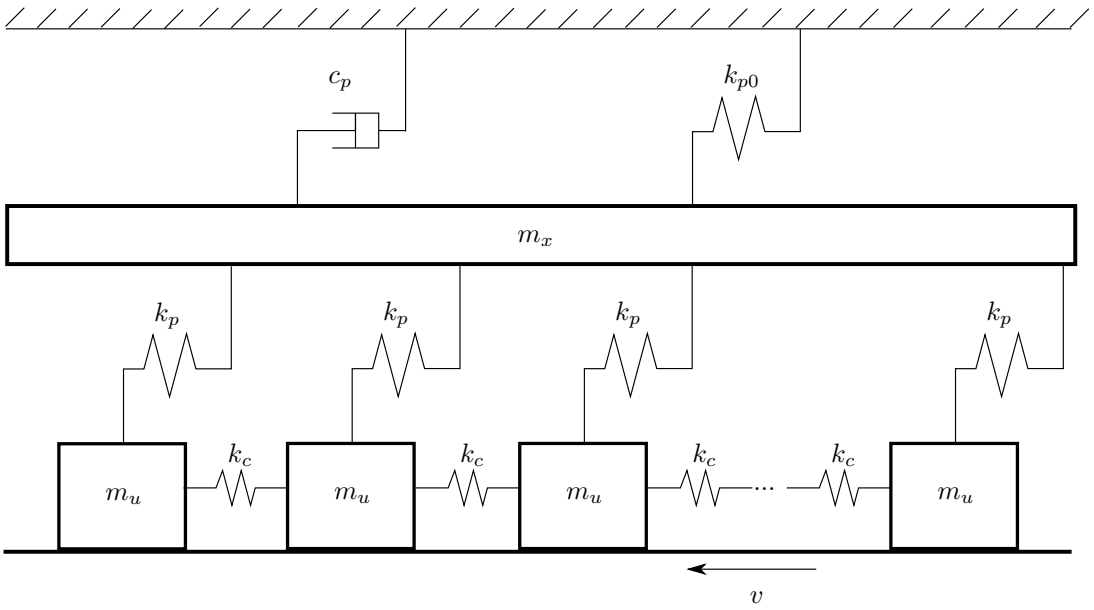


Figure 3.1: Illustration of the combined Burridge-Knopoff-pad model made in Inkscape. Parameters shown are given in Table 3.1.

3.2 Parameters

The BKP model contains of user-specified parameters/constants set in advance of running a simulation. These parameters establish the behaviour of the model, and a full list is presented in Table 3.1. As visible in the second part of the table, some parameters are derived from the user-specified shown in Equations (3.2) to (3.4).

$$m_u = \frac{m_x}{N} \tag{3.2}$$

$$k_p = \frac{k_{p0}}{N} \tag{3.3}$$

$$\begin{aligned} k &= \frac{EA}{L}, \quad L = \frac{L_{\text{tot}}}{N} \\ &= \frac{EA}{L_{tot}} N \end{aligned} \tag{3.4}$$

$$\zeta = \frac{c_p}{c_{crit}}, \quad c_{crit} = 2\sqrt{m_x K}, \quad K = k_{p0} + N k_p \tag{3.5}$$

Equation (3.2) shows that both the block mass m_u and the pulling spring constant k_p are proportional to the pads mass m_x and own pulling spring constant k_{p0} by $\frac{1}{N}$. Setting these parameters in this manner ensures that the total friction force from the disc through the blocks and onto the pad to be independent on the number of blocks N.

	User-specified		
Parameter	Symbol	Value	
Number of blocks	N	100	
Pad mass	m_x	100	
Caliper-pad spring constant	k_{p0}	100	
Damping ratio	ζ	$\frac{1}{12}$	
Maximum static friction force	F_0	1	
Friction law scaling parameter	σ	0.01	
Disc speed	ν	$Varying$	
Block mass scaling factor	s_m	1	
Pulling spring constant scaling factor	s_c	0.01	
Neighboring spring scaling factor	s_c	0.01	
		Derived	
Parameter	Symbol	Value	Given by
Mass of block	m_u	1	$s_m \frac{m_x}{N}$
Pulling spring constant	k_p	1	$s_p \frac{k_{p0}}{N}$
Neighboring spring constant	k_c	100	$s_c k_{p0} N$
Critical damping coefficient	c_{crit}	$2\sqrt{20000} \approx 282.84$	$2\sqrt{(k_{p0} + N k_p) m_x}$
Damping coefficient	c_p	$\frac{1}{6}\sqrt{20000} \approx 23.57$	ζc_{crit}

Table 3.1: Overview of parameters for the BKP model.

3.3 Friction

$$\phi(y) = \begin{cases} F_0[-1, 1], & y = 0 \\ F_0 \dfrac{1-\sigma}{1 + \frac{|y|}{1-\sigma}} \operatorname{sign}(y), & y \neq 0 \end{cases} \tag{3.6}$$

The reader should note that all dimensions used in the Thesis are dimensionless. This is because the system is not adapted to a specific application.

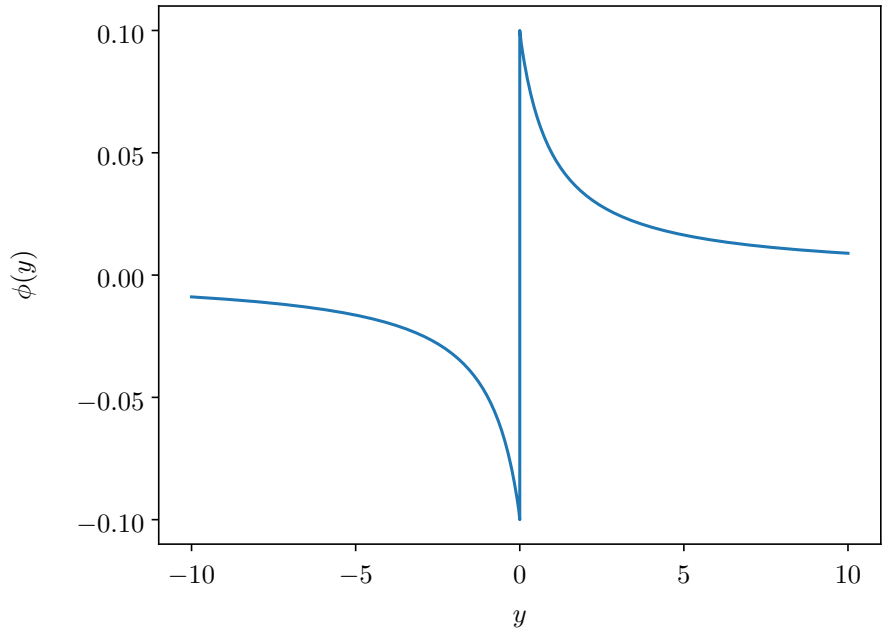


Figure 3.2: Friction scheme used in Equation 3.1

3.4 Numerical Scheme

To solve the differential equations governing the Burridge-Knopoff 2.2 and Burridge-Knopoff-Pad 3.1 model, a numerical scheme is used. As in the specialisation project, the 2nd Order Runge-Kutta - also known as the Midpoint Method - was used. The scheme is the same as the one used by H. Ferre [1] and suggested by R. Burridge and L. Knopoff in their 1967 Paper [7]. The 2nd Order Runge-Kutta gets its name from it being a two-stepped method and is defined for any function y as

$$\begin{aligned} y^{\mathrm{h}} &= y^n + \frac{\Delta t}{2}\frac{d}{dt}(y^n), \\ y^{n+1} &= y^n + \Delta t \frac{d}{dt}(y^{\mathrm{h}}). \end{aligned} \tag{3.7}$$

Here is the function y evaluated in between the current and next step. y^{h} is the half way evaluation of y, while y^n and y^{n+1} is the current and next step. Equation (3.7) needs to be adapted to solve governing Equation (3.1) of the BKP model. The adaptations requires the two 2nd order differential equations in Equation (3.1) written as y in vector form $\vec{y}$. $\vec{y}$ can be written with the general expression

$$\vec{y} = \begin{bmatrix} y \\ \dot{y} \end{bmatrix} \quad \dot{\vec{y}} = \frac{d}{dt}(\vec{y}) = \begin{bmatrix} \dot{y} \\ \ddot{y}, \end{bmatrix} \tag{3.8}$$

where the first element in $\dot{\vec{y}}$ is second element in $\vec{y}$ and the 2nd order derivative of y is contained in $\dot{\vec{y}}$. Writing the BKP model on the form of Equation (3.8) generates Equations (3.9) and (3.10).

$$\vec{u_j} = \begin{bmatrix} u_j \\ \dot{u_j} \end{bmatrix} \quad \dot{\vec{u_j}} = f(\vec{u_j}) = \begin{bmatrix} \dot{u_j} \\ \frac{1}{m_u}[k_c(u_{j-1} - 2u_j + u_{j+1}) - k_p(u_j - x) - m_u\phi(v + \dot{u_j})] \end{bmatrix} \tag{3.9}$$

With the boundary condition for the blocks set to zero.

$$\vec{x} = \begin{bmatrix} x \\ \dot{x} \end{bmatrix} \quad \dot{\vec{x}} = f(\vec{x}) = \begin{bmatrix} \dot{x} \\ \frac{1}{m_x}[-c_p\dot{x} + k_p\sum_j(u_j - x) - k_{p0}x] \end{bmatrix} \tag{3.10}$$

Where function f is the derivative of either $\vec{u_j}$ or $\vec{x}$, making Equations (3.9) and (3.10) applicable for the numerical scheme goverened in Equation (3.8).

The 2nd Order Runge-Kutta has a truncation error of order $\mathcal{O}(\Delta t^3)$ [12, p. 328]. This error can be confirmed by measuring calculating the total energy in the system. In Section 3.8.1 the error is investigated through calculating the total energy in the system.

The step sized used is based on the calculation by H. Ferre and has been set to $dt = 0.005$ for the batch runs.

3.5 Energy test

The amount of energy can be measured to test if the model behaves as expected. The law of *conservation of energy* states that the amount of energy does not change[13]. Moreover, the law states build the foundation for the first law of thermodynamics, that the total energy in an isolated system is constant. From this, removing damping and friction from the system means the total energy should be constant in the system and is expressed as

$$E = \frac{1}{2}m\dot{x}^2 + V(x) \tag{3.11}$$

where E is the total energy, m the system mass and V denote the potential energy[p. 159][14]. The system here being one of the two described below

1. pad is stationary, with only pulling springs connecting the pad and blocks activated. The total energy expression E_{ps} is shown in Equation (3.12).
2. pad is stationary, with only neighbouring springs in between neighbouring blocks activated. The total energy expression E_{ns} is shown in Equation (3.13).

$$\sum_{j=1}^{N} E_{ps} = \frac{1}{2}m_u \sum_{j=1}^{N} {\dot{u}_j}^2 + \frac{1}{2}k_p \sum_{j=1}^{N} u_j^2 \tag{3.12}$$

$$\sum_{j=1}^{N} E_{ns} = \frac{1}{2}m_u \sum_{j=1}^{N} {\dot{u}_j}^2 + \frac{1}{2}k_c \sum_{j=2}^{N} (u_j - u_{j-1})^2 \tag{3.13}$$

. Both of these are modified versions of the BKP model in Section 3.1. In Equations (3.12) and (3.13) the energy of each individual block is summed up.

3.6 Friction Amplitude

As performed by H.Ferre in his thesis, the amplitude of the friction is calculated in the same manner[1]. Letting the slider have a constant speed for a set amount of time, the total friction in the system is calculated and logged. The time interval is then split in two, and the friction amplitude is calculated by using the mean squared error of the mean of the friction force. In this thesis, the amplitude is calculated by looking at am interval I of the friction data. This interval is assumed to have constant slider velocity. In practice, this is done by step-wise increasing the slider velocity. It step makes an interval, where the second half is used as I. Further, the mean $\bar{I}$ of the interval is calculated. From $\bar{I}$, the amplitude is obtained as the root of the variance, as derived in Equations (3.14) and (3.15). The amplitude is multiplied by the square root of two to obtain the actual amplitude [1].

$$\bar{I} = \text{avg}(I) \tag{3.14}$$

$$A = \sqrt{2}\sqrt{\text{avg}(I - \bar{I})} \tag{3.15}$$

In order to indicate the accuracy, interval I is further split into J segments of size S_{size} to calculate the error. This is done in order to calculate the error of the amplitude. A formula for splitting the interval is shown in Equation (3.16).

$$I_i = I[i * S_{size}, (i+1) * S_{size}] \tag{3.16}$$

$$\bar{I}_i = \text{avg}(I_i) \tag{3.17}$$

$$\sigma = \sqrt{\text{avg}((\bar{I}_i - \bar{I})^2)} \tag{3.18}$$

$$\epsilon = \frac{\sigma}{\sqrt{J}} \tag{3.19}$$

As in Equation (3.15), the variance for the splits is calculated and from this the error as shown in Equation (3.19). The amplitude A can be seen in Figure 3.3 where the 16 different runs of the Burridge-Knopoff(BK) model is plotted. Here the the error ϵ is shown as red bars for velocities plotted. Note that the amplitude is plotted step-wise and the lines between these steps are there to show the development. The reader should also note that Figures C.1, 3.3 and 3.4 are plotted with log-scale on the ν-axis. Also note that the mean value of the friction is included in Figure 3.5, represented by the lines in the upper segment.

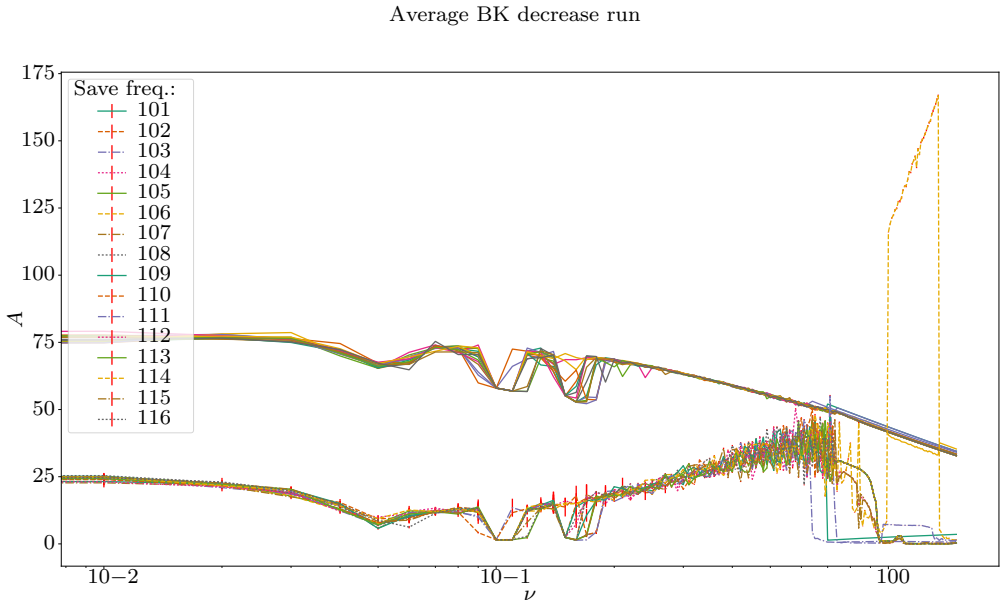


Figure 3.3: 16 runs of the BK model, showing each friction amplitude with corresponding error as an vertical bar. The mean of the friction force is included and are the top lines in the plot. The averaged line and bars of the runs is shown in Figure 3.4.

In the attempt to find the general, but at same time individual behaviour of the runs, unique runs are measured against the mean. This measure is done in the consecutive figures.

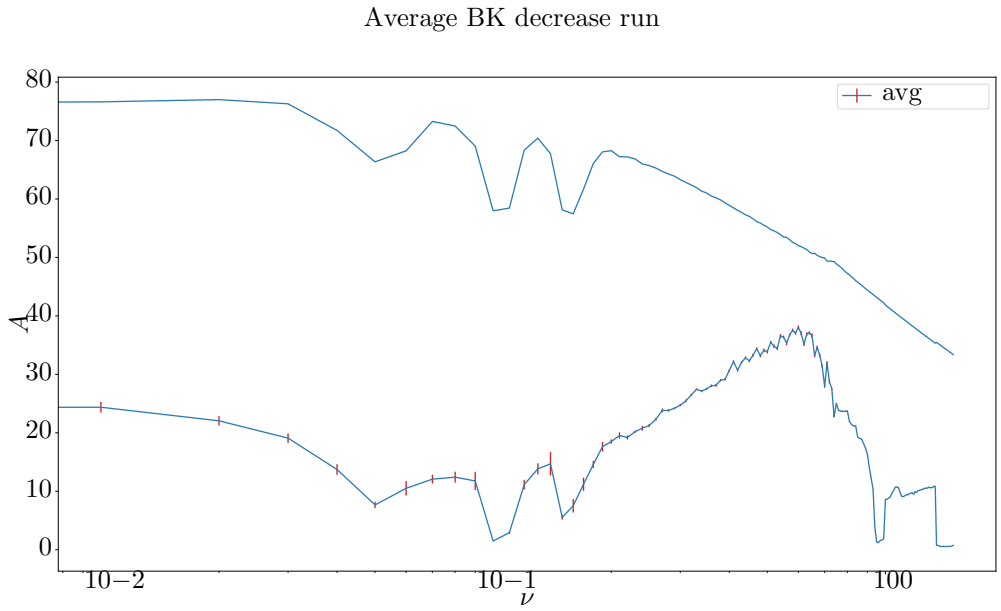


Figure 3.4: Average of the friction amplitude runs in in Figure 3.3. The top lines show represent the mean value of the friction and the bottom lines the friction amplitude.

In Figure 3.4, the mean of the multiple runs in Figure 3.3 is potted. As is done for the multiple runs, both the friction amplitudes with associated error bars and the mean friction is plotted. The mean friction amplitude is further used to view how much the individual runs differ from it. Figures 3.5 and 3.6 illustrate the per velocity and cumulative difference for each run to the mean.

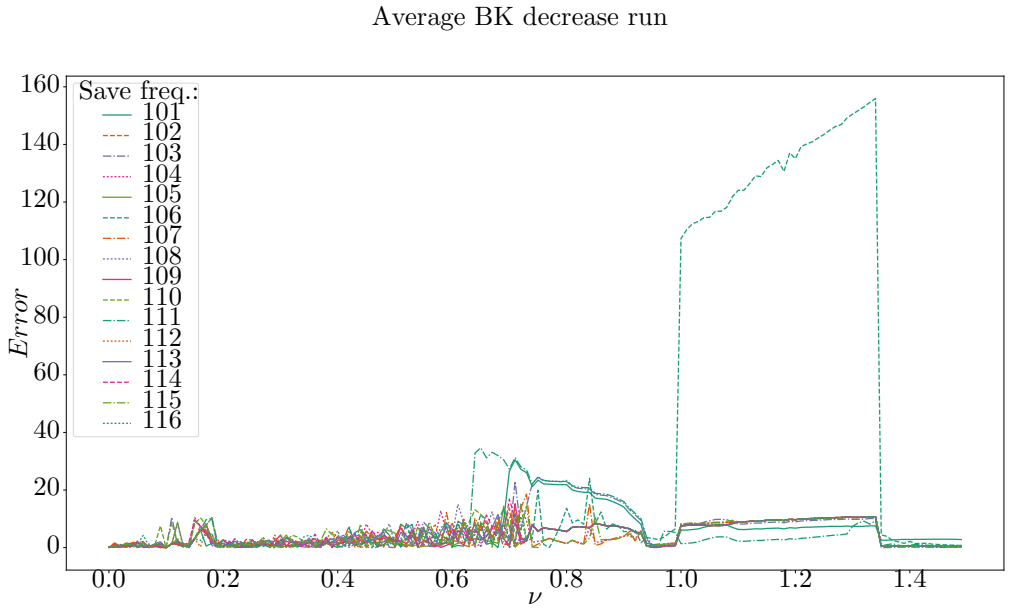


Figure 3.5: Difference path for each run in Figure 3.3 compared to the mean in Figure 3.4.

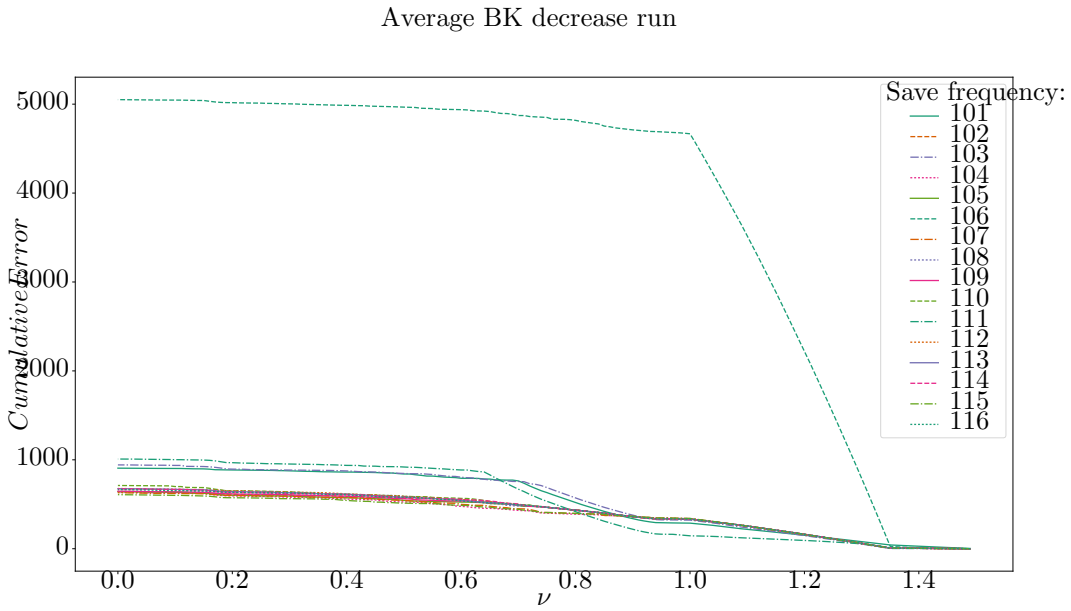


Figure 3.6: Cumulative difference for each run based on Figure 3.5.

As is seen in Figures 3.5 and 3.6, the general nature of a particular model and setup is visible. These figures also shows where the individual runs differ. In Figure 3.5 there are a few runs that varies more than the average and one that really sticks out. This particular event is detailed in Chapter 5, and shown here to establish the methods used to interpret the results of the simulations. Further, the total cumulative difference is plotted as a bar chart in order to clearly see such outliers, as seen in Figure 3.7.

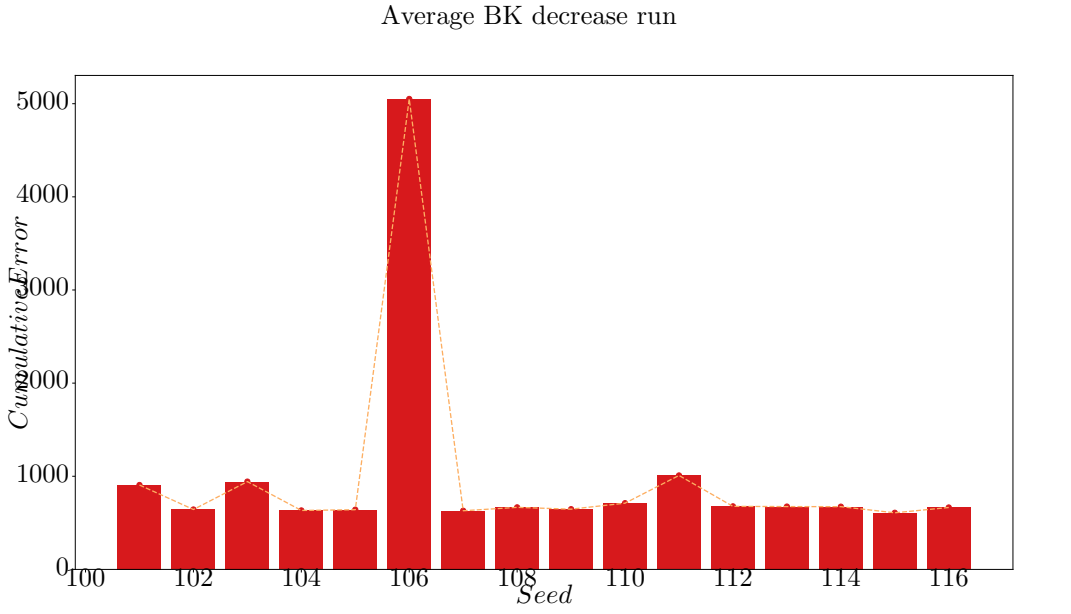


Figure 3.7: Total difference for each run in Figure 3.4.

To give another measurement for how much the value friction amplitude varies, the confidence interval(CI) was calculated. A CI is an interval where there is $100(1-\alpha)\%$ chance a random value or run will be in. If $\alpha = 0.05$, there is a 95% CI[15]. The CI utilised on Figure 3.5 is shown in Figure C.1.

3.7 Output from model

Several values are possible to choose as the output of the model. This list contains; the velocity and position of both the pad and blocks and combined friction force in the system. The most interesting is the latter, together with the velocity and position of the pad. Due to the model being evaluated numerically with a step size dt for a specified number of time steps, saving all the information in the system can quickly build up the required disk space, memory and run time. To combat these two measures are taken, only saving blocks specified in the parameters file and only saving a fraction of the steps. It shows in Figures 3.8 and 3.9 that the last one can be done to maintain the information of the systems behaviour. The position, velocity of the pad together with the combined friction, provides information that helps to determine which runs it is worth looking into more. These values make saving only a subset of the output of the block and outputting all when it is suiting a viable option.

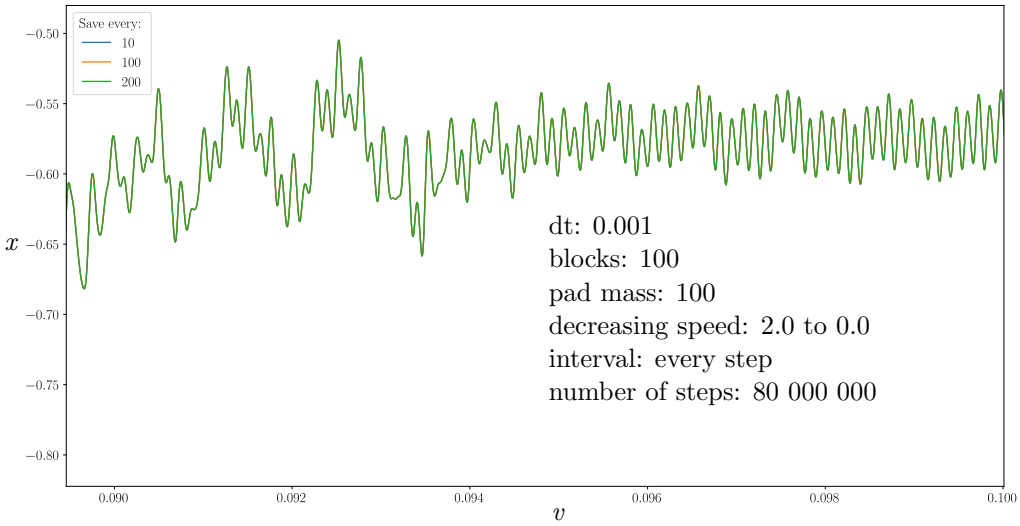
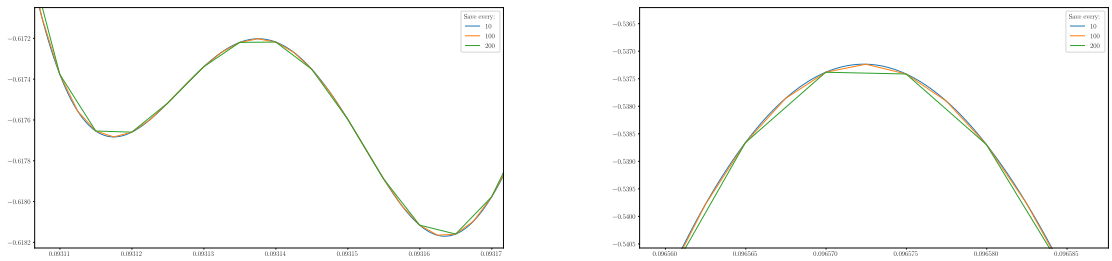


Figure 3.8: Pad position x plotted for different velocities ν. The velocity in the run is decreasing continuously, where the end is shown here. The same run is done with different frequency in how often the output is saved.



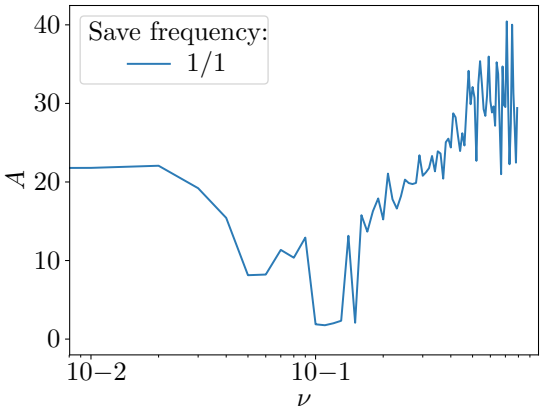
(a) Zoom of Figure 3.8

(b) Zoom of Figure 3.8

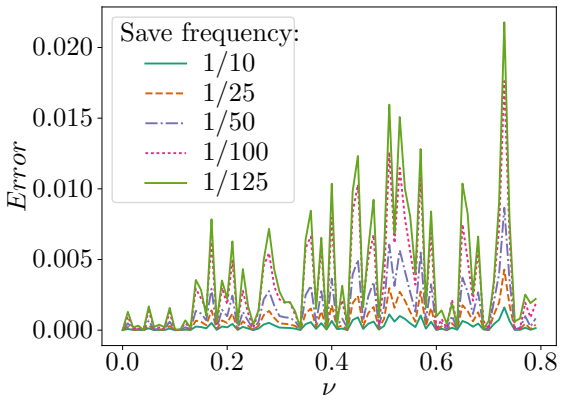
Figure 3.9: Smaller intervals of ν in the outputFigure 3.8 to show the effect of lowering the save frequency.

As is visible in Figures 3.9a and 3.9b, lower saving frequency leads to some loss of information and more so in the regions with more oscillation. The loss in Figures 3.10 and 3.11 shows that while its reasonable to not save every data point, but there is a limit. As is drawn from the sampling theorem referenced in Section 3.10, too low sampling frequency will lead to loss inability to pick up higher frequencies in the

signal. To investigate a suiting saving frequency, decreasing frequencies were compared to saving every data point.



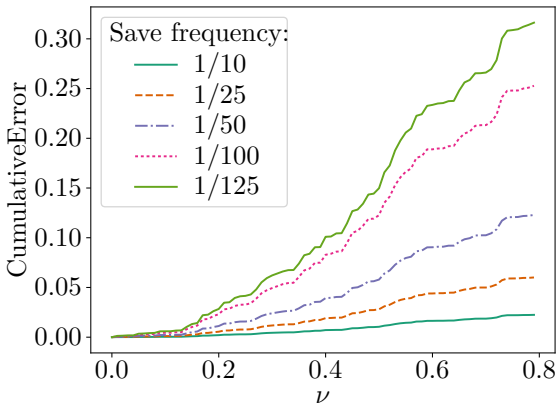
(a) Friction amplitude for a single run of the BKP model with increasing slider velocity. Here every computed data point is used to produce the plot.



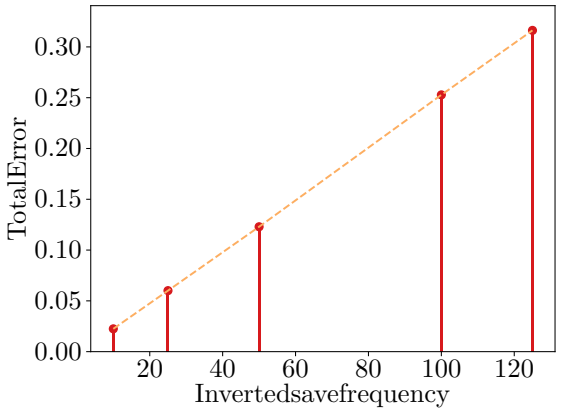
(b) Different save frequencies used in the same run as in Figure 3.10a. Here the difference from the 1/1 save frequency is shown for for each friction amplitude calculated with lines in between the values.

Figure 3.10: Comparing save frequencies. Note that ν in Figure 3.10a is plotted with log-scale.

The friction amplitudes - as in Figure 3.5 - for a single and randomly chosen run is shown in Figure 3.10a above. In Figures 3.10b, 3.11a and 3.11b saving every data point is compared to lower saving frequencies.



(a) Cumulative difference of the lines shown in Figure 3.10b.



(b) Total difference in friction amplitude for different save frequencies compared to saving every data point as in Figure 3.10a.

Figure 3.11: Comparing save frequencies.

3.8 Testing the implementation

In the project Thesis by the author, the BKP model was tested to review if it was behaving as expected and to demonstrate its behaviour. In this section, a summary of this is given, as well as some additional showcases of the model. This section is structured to effectively reference which part of the model is focused and how it is achieved. The latter done with references to the parameter which reads from a YAML file explained in Section 4.2.1.

3.8.1 Blocks

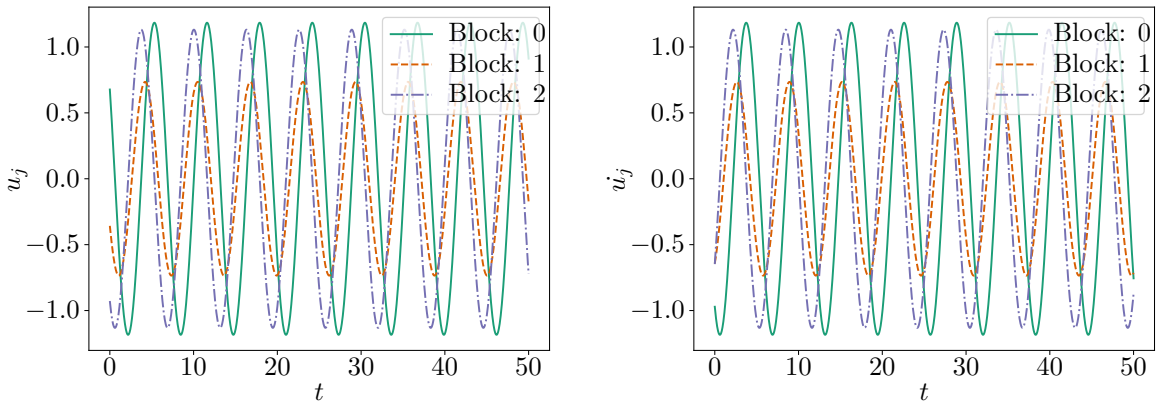
To understand how the blocks affect the model, the different block components will be deactivated to isolate the individual components of the BKP model. The isolation follows by deactivating different parts of Equation (3.1).

Stationary springs

In Figure 3.12 the is set to be stationary like the upper surface in the BK model shown in Section 2.2. The neighbouring springs and friction from the lower surface are deactivated. As visible in Figures 3.12a and 3.12b this leads to the three blocks oscillating with a constant amplitude and frequency. The oscillation is a result of the initial position of each block. Each block has an initial position u_j and velocity $\dot{u_j}$ drawn from a uniform distribution. The uniform distribution notes as $U(a, b)$ where a and b are the minimum and maximum value and each value in the interval has an equal chance of being selected. In this thesis the distribution $U(-1, 1)$ is used, but the implementation also supports $U(-1, 0)$ and $U0, 1)$ for generating the initial block position(s).

	Parameters	
Parameter	Symbol	Value
N	N	3
slider_speed	ν	0.0
	Debug	
Parameter	Symbol	Value
debug_no_friction		true
debug_no_neighbor_springs		true
debug_no_pad		true
debug_negative_initial_values		true

Table 3.2: Table over important parameters used to only test the stationary springs from the pad to the blocks. The different parameters reference those shown in Figures 4.4 and 4.5.

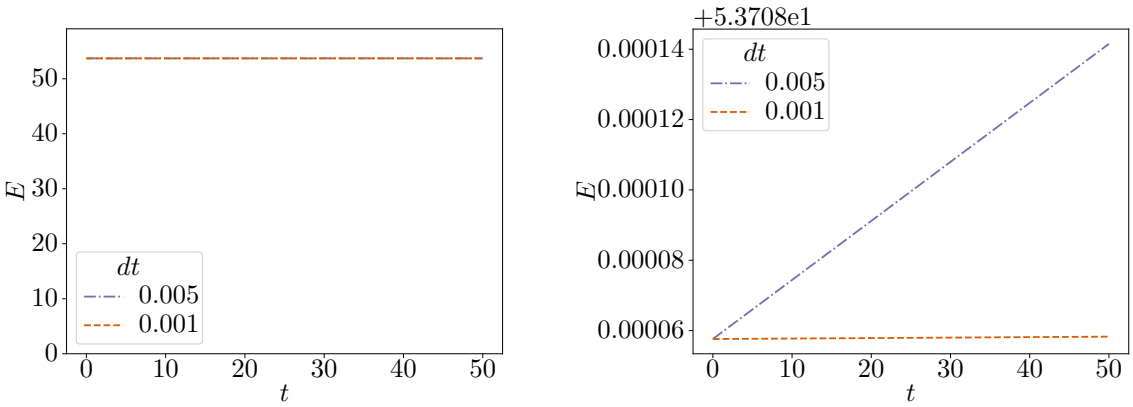


(a) Position of the blocks.

(b) Velocity of the blocks.

Figure 3.12: Output from model when only the stationary springs from the pad to the blocks is active. Based on parameters and configurations in Table 3.2.

In Figures 3.12 and 3.14 u_j and $\dot{u_j}$ are the relative position and velocity of blocks, where $j \in [1, 3]$.



(a) Total energy is constant.

(b) Numerical error from the numerical scheme is visible.

Figure 3.13: Total energy in system given by parameters and configurations in Table 3.2. Parameters file in Appendix Section A.1.

dt	Error E	Ratio dt	Ratio E
0.001	6.7133638737e-07	1	1
0.005	8.3910513645e-05	5	124.990266018

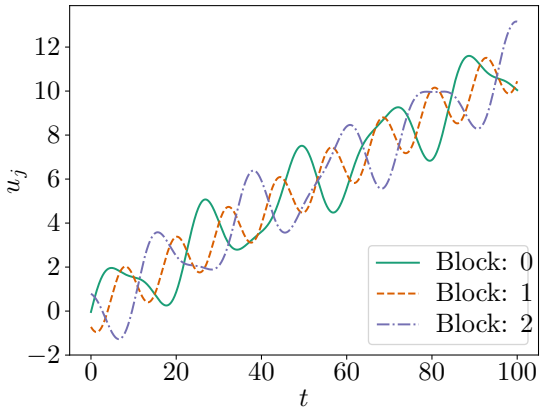
Table 3.3: Numerical error and ration of Figure 3.13 listed.

Neighbouring springs

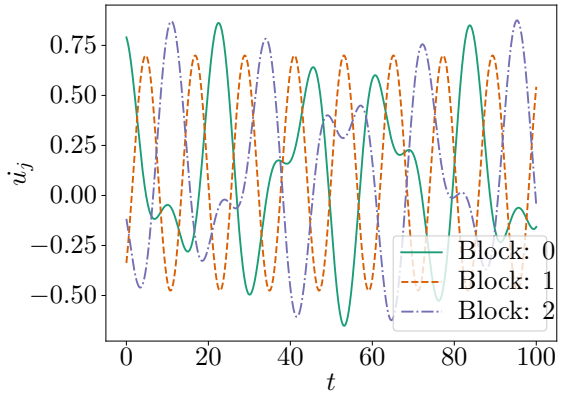
Further, the neighbouring springs of the blocks in Equation (3.1) are isolated. As in Figure 3.12, the position u_j and velocity $\dot{u_j}$ are shown in Figure 3.14, but with neighbouring instead of stationary springs activated.

	Parameters	
Parameter	Symbol	Value
N	N	3
slider_speed	ν	0.0
	Debug	
Parameter	Symbol	Value
debug_no_friction		true
debug_no_stationary_springs		true
debug_no_pad		true
debug_negative_initial_values		true

Table 3.4: Table over important parameters used to only test the neighbouring springs from between the blocks. The different parameters reference those shown in Figures 4.4 and 4.5

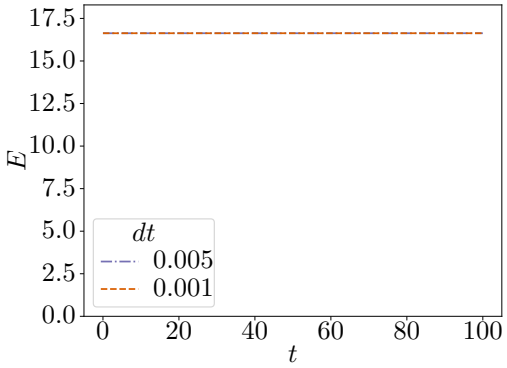


(a) Position of the blocks.

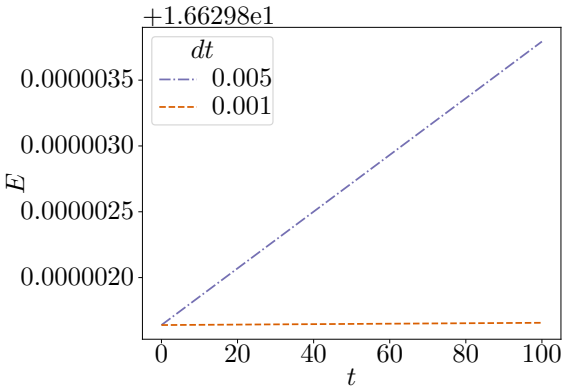


(b) Velocity of the blocks.

Figure 3.14: Output from model when only the neighbouring springs from the pad to the blocks is active. Based on parameters and configurations in Table 3.4.



(a) Total energy E plotted over time.



(b) Numerical error from the numerical scheme is visible.

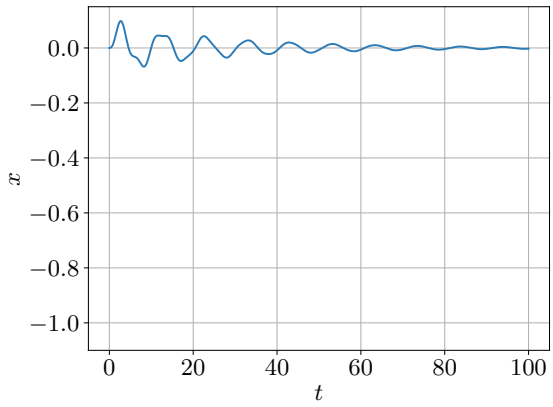
Figure 3.15: Showcasing the numeric error present in Figure 3.15a.

dt	Error E	Ratio dt	Ratio E
0.001	1.7230874505e-08	1	1
0.005	2.1537441377e-06	5	124.993315753

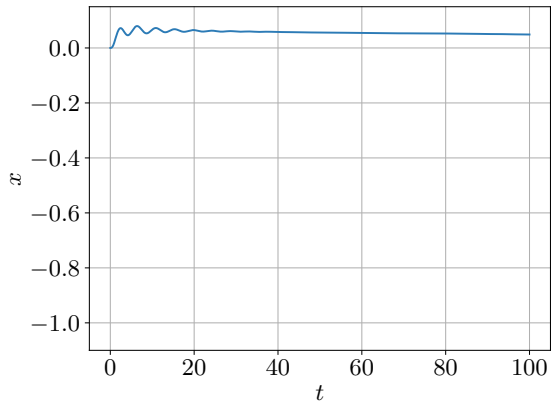
Table 3.5: Numerical error and ration of Figure 3.13 listed.

3.8.2 Pad

The pad is a big component of the BKP model, combining the ODOF and BK model. The pad is connected to the upper surface with a spring and a damper. It is also affected by the lower surface through the blocks connected to it by springs. Some of the pads behaviour is demonstrated in the figures below. In Figure 3.16 the slider has a zero velocity, meaning there system will eventually stop if friction or damping is activated. In both Figures 3.16 and 3.17 there are three blocks in the system, where their initial position and velocity is the same as in Figure 3.14.



(a) Pad position plot where the friction force is deactivated, the pad position goes gradually towards zero due to the damper seen in Figure 3.1.



(b) Pad position plot where the damper is deactivated, the pad slows down due to the blocks being slowed by friction from the blocks.

Figure 3.16: Two setup showing how different parts of the BKP model affect the pad. In both figures the slider velocity is zero. As seen in Figure 3.16b the the pads positions is positive. This is due to the blocks velocity and position being mainly positive.

In Figure 3.17 the slider speed is constant. The pad is dragged back to around $t \approx 25$ where the mean value of the position averages out. The behaviour of the pad at his time is described in Section 3.8.3.

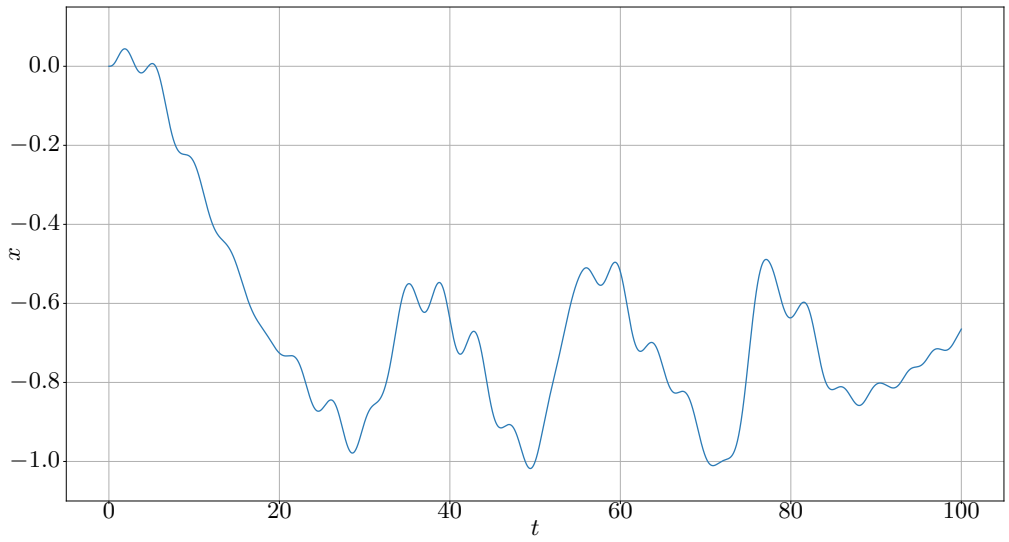


Figure 3.17: This setup shows how the pad is first dragged back from its origin position while oscillating due to the three blocks connected to it. The slider speed is set to $\nu = 0.1$.

3.8.3 Slipping

As described by in Ferres thesis, the system shows a stick-slip behaviour[1]. This can be seen clear in Figure 3.18. The run shown follows $\nu = 0.000 + 0.005 \cdot t \mathrm{mod} 2000$, meaning that the slider is not moving the first 2000 time steps. This is clearly visible when the pad is first being dragged back before stagnating at a steady state. After the slider velocity increases and there is a slipping event of the pad. The pad starts to get dragged back again before a new slipping event occurs. This behaviour continuous at a

steady rate. When the slider speed increases again its possible to observe that the slipping happens at a higher rate. The slipping occurs when $n \in [1, N]$ blocks slips and the pad back.

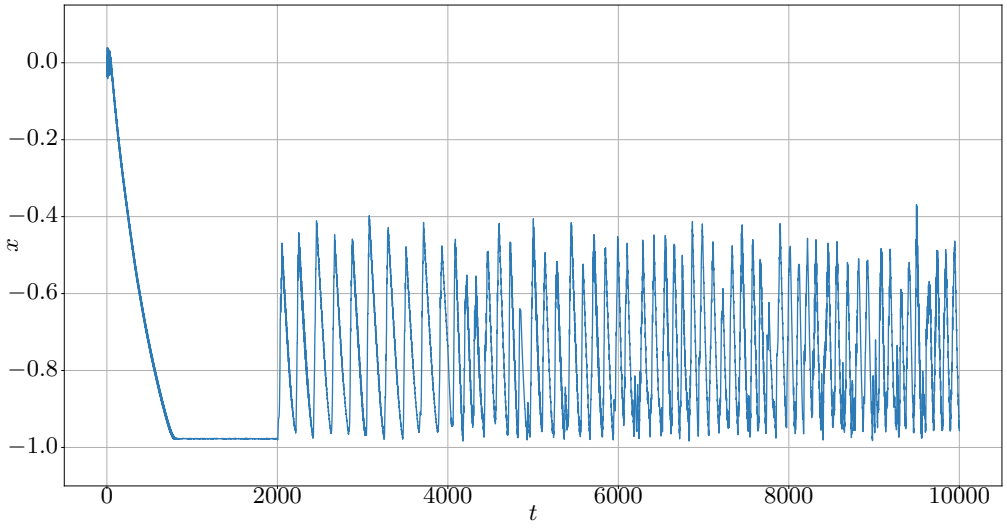


Figure 3.18: Pad position plot showing how the pad slips regularly at non-zero slider velocity ν. ν is increased by 0.005 every 2000 time step t.

3.8.4 Phase plot

As shown in the papers by A. Papangelo et al., plotting the blocks position versus their velocity in a phase plot can show when they are sticking and slipping[10, 11]. In Figure 3.19 a simple BK system is simulated with constant slider velocity $\nu = 0.08$.

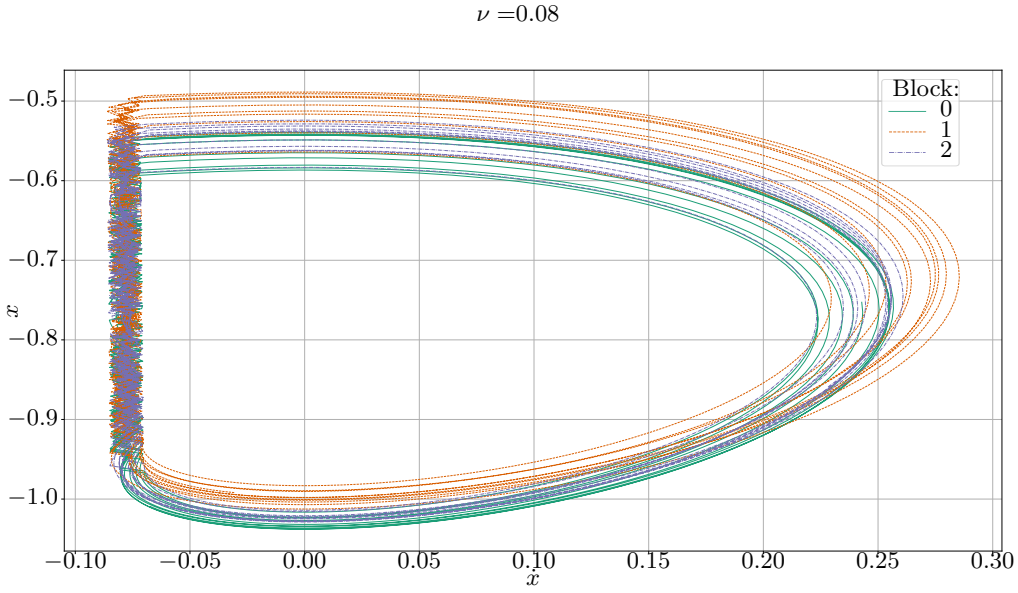


Figure 3.19: Phase plot for BK system with three blocks. The paths show their behaviour during stick-slipp events.

Figure 3.19 shows how the blocks stick a negative oscillating velocity before slipping. In the slip they obtain a positive velocity and moves in the opposite direction of the slider. This behaviour repeats multiple times in the time interval shown.

3.9 Animation

In order to capture more of the behaviour of the system, an animation tool was developed. This tool allows inspecting more closely what is already seen in the still plots and reveal possible new features. The animation implementation enables both the BK and BKP model to be investigated in 1. inspecting the interaction between the blocks themselves and with pad in the BKP model, 2. inspecting the transition during different events in the model, like change in velocity, 3. compare the behaviour of different runs and models in the manner described in the two previous points. Figures 3.21 and 3.22 each show a single frame from an animation of the BK and BKP model, in that order. Both animations follow the same layout in Figure 3.20,

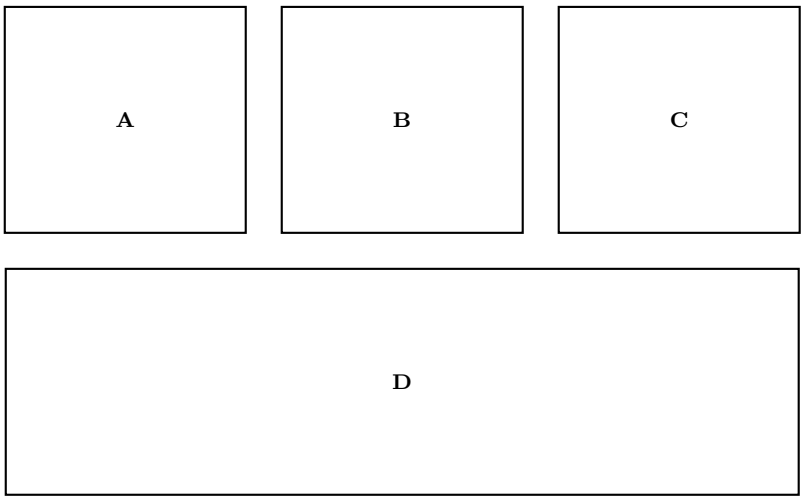


Figure 3.20: Diagram showcasing different parts of the animation. **A**: Block position(s) over time. **B**: Block position(s) as histogram over block number. **C**: Blocks avg. position or pad position over time. **D**: Figure of the BK or BKP model with stationary and neighbouring

where **A**, **B** and **C** can be used to show phase diagrams, block and pad velocities as well as still frames.

In both Figures 3.21 and 3.22 box **A** and **B** (as shown in Figure 3.20) visualise the position of the blocks. **A** shows all the $N = 100$ block's individual positions for the last - in this case - 100 logged data points. Logged refers to that not all the steps of the simulation is necessarily saved as demonstrated in Section 3.7. **B** Shows the relative position/offset of each block to the pad using a bar plot, also known as a histogram. The number of each block is given on the x-axis where the leftmost is the first and the rightmost the last block in the system. For Figure 3.21 box **C** shows the average position of all the blocks shown in **A**. This average is given due to the lack of a pad in the BK model. In Figure 3.22 box **C** shows the pads position. The figure in **D** is a representation of the BK or BKP model. The representation contains lines that are displaying the pulling and coupling springs in Equations (2.1) and (3.1). The blocks are shown as small squares and the pad as an elongated rectangle. Both blocks and pad moves corresponding to the values in box **A**, **B** and **C**.

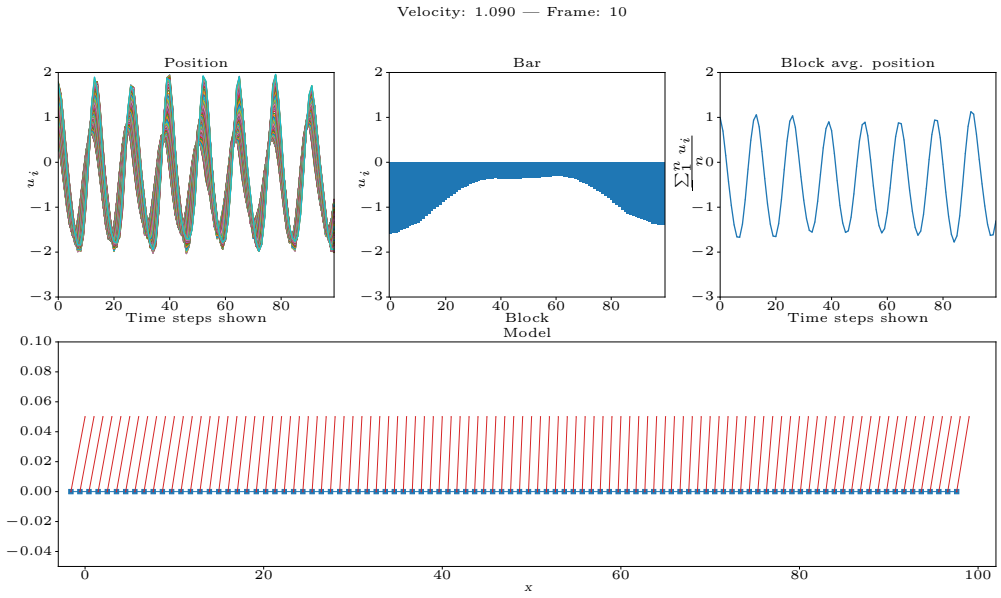


Figure 3.21: Example of BK animation interface. Showing frame number 10 of an animation from a decreasing velocity run if the BK model.

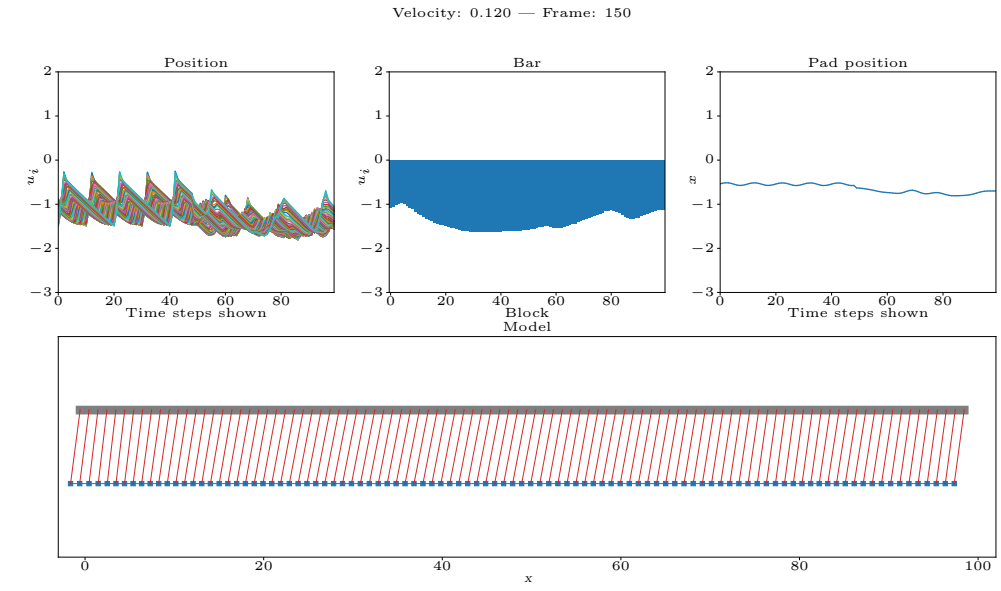


Figure 3.22: Example of BKP animation interface. Showing frame number 150 of an animation from a decreasing velocity run if the BKP model.

3.10 Fourier spectrum

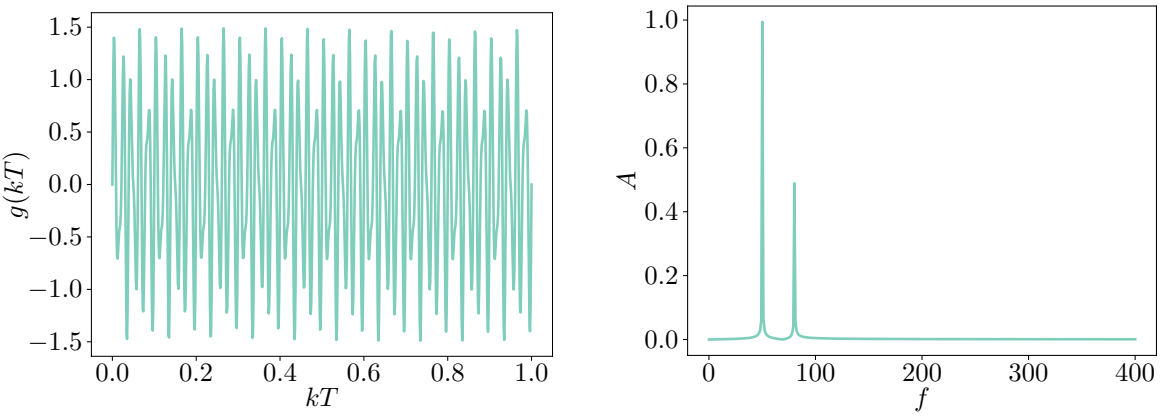
In order to capture the most predominant frequency and amplitude of the blocks and pad, discrete Fourier transform(DFT) was used. The Fourier transform is used to perform spectrum analysis on a signal which gives clear-to-read frequency and amplitude information[16, p. 365]. A signal in this thesis is refering to the position of a block or the pad over time. Referring here to the blocks in the BK model and pad and blocks in the BKP model. A signal is assumed to have a periodic waveform and is contained in N samples in the sample interval T. The DFT is governed by the equation

$$G(\frac{n}{NT}) = \sum_{k=0}^{N-1} g(kT) \cdot e^{j \cdot 2\pi \cdot n \cdot k/N} \qquad n = 0, 1, \ldots, N-1, \quad j = \sqrt{-1} \tag{3.20}$$

, where $g(kT)$ is the sampled periodic function [16, p. 97].

The Fourier spectrum for the combined sine function below is shown in Figure 3.23b

$$g(kT) = \sin(50 \cdot kT) + 0.5 \cdot \sin(80 \cdot kT)$$



(a) Value of $g(kT)$ for $kT \in [0.0, 1.0$.

(b) Different freqencies with respective amplitude in $g(kT)$ shown in Figure 3.23a.

Figure 3.23: $g(kT)$ with corresponding Fourier spectrum.

, where $kT \in [0, 1.0]$ is an arbitrary value, f is the frequency and A is the amplitude. The peaks in frequency corresponds well with the combined sine function $g(kT)$ given above.

An important thing to have in mid when using the Fourier transform for a function the frequency $\frac{1}{T} = 2 \cdot f_c$ known as the *Nyquist sample rate*. It comes from the sampling theorem which states the Fourier transform is zero for all frequencies grater than f_c[16, p. 83-84]. This means that if the frequency of the sampling is too low, the calculated frequencies in the DFT will miss the higher frequencies. The sample rate should be set to at least twice the size of the highest frequency in the signal.

In the implementation, the DFT is calculated in Python using the fast Fourier transform(FFT) function provided by SciPy. SciPy builds on top of NumPy to provide tools for optimisation, special functions and image processing [17].

Chapter 4

Implementation tools

4.1 Programming languages

The simulation runs by solving the differential equations in Equation (3.1) using the 2nd order Runge Kutta numerical scheme governed by Equation (3.7). The scheme is implemented in C++ and described more thoroughly in the following Section 4.1.1. When the simulation is complete, the results are placed in files using the *csv* format. *csv* is a simple file format storing data as text using a comma to separate values and a line break to separate data records[18]. The saved output can then be loaded and examined multiple times retrospect the simulation. It also opens the possibility - that has taken in this thesis - of using a second programming language to the analysis. Matlab, R and Python are good choices, with the latter picked. The use of Python is described in greater detail in Section 4.1.2. As mentioned in the introduction, the implementation strives to have good flow and erase the need to change the code between runs.

4.1.1 C++

The BKP simulation is implemented in C++, consisting of three -*.cpp* and two -*.hpp* These are compiled to one executable using CMake and Make. CMake uses a configuration file *CMakeLists.txt* which describe the dependencies needed and where it should look for these, as well as where to put the compiled executable. This building process is sketched out in Figure 4.1 which showcase the dependencies by connecting an arrow from the dependant to the dependency. CMake combines the files shown into one executable that takes **one** input, the path to a YAML file containing the parameters and configurations for a specific run. YAML is described in more details in Section 4.2.

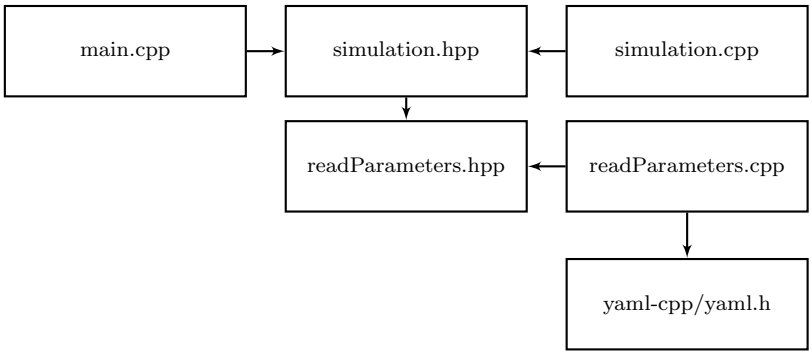


Figure 4.1: Flowchart showing the connection between the scripts. Source: Project thesis

When starting a new simulation, *main.cpp* is called taking in one argument intended to be the path to a parameter file. If the path is valid *readParameters.cpp* will read in and store the parameters and configuration settings for the run, otherwise return an error. The newly created *readParameters* object is used in *simulation.cpp*, calling it whenever a user-specified parameter is used or configuration checked. When the simulation completes, the output is saved, ready for inspection and visualisation.

Unique runs

The pads initial position and velocity are initially zero for every new run for the BKP model. As described in Section 3.8.1, the blocks position and velocity is drawn from a uniform distribution U. This initialisation allows obtaining two properties in the implementation; running unique runs with initial values from a pseudorandom number generator(PRNG), using *seeds* to recreate specific runs. A *seed* is an integer used to initialise the PRNG[19].

4.1.2 Python

Python, which refers to Python 3 and not Python 2 in this thesis, is a programming language that enables efficiently writing small scripts while still enabling making bigger straightforward implementations. As python developer J. Noller [20, p. xiii] stated, "It is useful for someone who wants only to do some math or write a simple script. And it is equally useful for programmers who want to create large-scale systems, web frameworks, and multimillion dollar video-sharing sites.". Noller also states that he use Python because it is quote "clean, simple and powerfull". In this thesis, Python is used to make simple scripts like reading a parameter from a YAML file as described in Section 4.2. It is also utilised to do more powerful tasks, like animating the output data shown in Sections 3.9 and 4.1.2. .

Visualisation

"Matplotlib is a 2D graphics package used for Python for application development, interactive scripting, and publication-quality image generation across user interfaces and operating systems.", Quote by from Matplotlib creator J.D. Hunter[21]. The Matplotlib package is the main addition to the Python core for visualising the data. It has been a tremendous and versatile contribution that support plotting with graphical user interface (GUI), multifigure plots, animation, bar charts, LaTex text and Scalable Vector Graphics to mention some[21]. This support, combined with an extensive amount of customisation in, e.g. colour, line style and labelling makes it fit the project very well. The packages SciPy and NumPy has also been significant in the implementation. These two packages are built by scientist, engineers and researchers to increase Python's usefulness in scientific computing[22].

Animation

As described by the creator of Matplotlib, J.D. Hunter, the package has three basic class at the highest level[21]. These classes can be imagined as a painters canvas, brush and the artist itself. Following this metaphor, an animation canvas is created, and then different graphics are drawn on it. Here it starts to differ from the analogy due to the ability to repaint the strokes in the painting, not needing to get a new canvas. The animation takes advantage of this, redrawing each component instead of producing all of the body, ticks and labels every frame.

4.1.3 Bash script

As the simulation code is implemented in Section 4.1.1, it synergies well with making a run folder for each new run. Each of these folders contains a copy of the code which is compiled and run when doing a new simulation. Simulating in this fashion gains the advantage of knowing the precise code used for a run and the output produced by it. Small changes were frequently done both to improve the code and extract new features during the project. Having the code for each run makes comparing changes simpler and acts as insurance if odd behaviour is discovered, with the cost being a few megabytes extra in multiple copies of the code. Each run has a describing name in the parent folder and a specified structure for smooth navigation. Such folders can be generated manually, but it quickly gets tedious to copy code and folder structure, then compile the code and call the executable with the path of the parameters. By using simple UNIX commands, a bash-script automates this process. In short, it takes in a path for a new run folder - relative to the current working directory -, copies code and parameter files, compiles and run the code as shown in Figure 4.2.

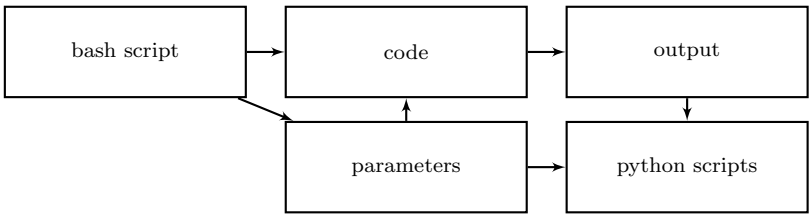


Figure 4.2: Flowchart showing the flow of the project. Source: Project thesis

In Figure 4.2 **python scripts** are included to help show how the output and parameters file is used when visualising and interpreting a run. Running the Python scripts combines very well with storing each run in a folder, see Subsection 4.1.2 for more detail on the use of Python.

Batch simulations

The term *batch simulations* is referencing running multiple simulations in sequence to compare or find averages of different setups. In the project bash script where used to enable this feature. Often the scripts utilised python code to change parameters in a predefined fashion, with altering the *seed* being between runs being an example.

4.2 YAML

YAML, abbreviated from "YAML Ain't Markup Language" is a data serialisation language. It is chosen for the implementation due to its qualities of being "designed to be human-friendly and work well with modern programming languages for common everyday tasks"[23]. The main advantage is that YAML is easy to read for humans, meaning there is little previous experience required when working with it. The rules of YAML are straightforward and results in a format that is similar to the ones used in everyday writing. Due to this and its compatibility with C++, Python and other modern programming languages YAML is well-suited for the project. It opens up the beneficial property of reading and altering the input parameters without changing or keeping track of any code in the project. Figure 4.3 shows a simplified version of the structure used for the input files for the simulation.

```yaml
---
map:
  key_11 : value_11
  key_12 : value_12
    ⋮
  key_1n : value_1n
list:
  - value_21
    ⋮
  - value_2n
...
```

(a) Simple example of the YAML syntax.

```yaml
---
map_1:
  key_11 : value_11
  key_12 : value_12
    ⋮
  key_1n : nested_map:
    key_1n1 : value_1n1
    key_1n2 : value_1n2
      ⋮
    key_1nm : value_1nm
...
```

(b) Simple example of nested mapping.

Figure 4.3: Examples showcasing some of the YAML syntax utilised in the project. n and m are used to demonstrate that an arbitrary amount of map and list elements can be chosen.

A common alternative to YAML is JSON, both of which has its pros and cons. JSON is to modern programming environments more trivial to generate and parse, while YAML is more complex in that regard with the advantage of increased human readability. It should be mentioned that a JSON file with unique keys is a valid YAML file. This property simplifies the process of migrating from one of the formats to the another. XML is also worth mentioning, being another popular format to store and transport data. While also being created with the object of being human readable, XML gave this a lower priority compared to YAML[24]. Quote from the YAML documentation state; "XML is a pioneer in many

domains, YAML is the result of lessons learned from XML and other technologies.". YAML works very well in the project since it enables the parameters and settings to be altered and read effectively. The language follows some simple rules of indentation and separation signs to construct *key : value* maps. Maps are used to create new parameters file is simple, and editing one even more so. An example of how it looks can be seen in Figure 4.4 and 4.5.

yaml-cpp

To parse the parameters to editable C++ objects, the YAML parser and emitter *yaml-cpp* is utilised[25]. It is used together with the 1.2 version of YAML[23]. In this Thesis, only the parser is used and has shown to be sufficient at parsing the data. The parser in implemented in program that after loading a YAML file, sets each of the known parameters and configuration variables setting the ones it finds. If a variable is not available, it will be set to a default value. There is also an emitter available, which is not used in the C++ implementation. Instead, there is a python-script to change values without directly editing the YAML-files.

PyYaml

The information contained in the YAML-files is useful when visualising the data. PyYaml is a python package[26] that enables parsing and emtting to the YAML format. Hence all the parameters set for a given run can be loaded into a Python script. Loading the parameters reduce the amount of tuning and file tracking needed when visualising runs with different parameters.

4.2.1 Parameters in YAML

The parameters file written in YAML is separated into two sections, one containing the parameters and the other boolean values that change the behaviour of the simulation. These are shown in Figure 4.4 and 4.5 respectively.

```yaml
---
Parameters:
  dt : 0.005
  seed : 101
  num_events : 1
  N : 100
  max_time : 16000
  slider_speed : 0.0
  increment : 0.1
  interval : 2000
  file_name : test_solutions
  progress_indicator : true
  m_F0 : 1
  m_alpha : 0.5
  m_sigma : 0.01
  m_mass_x : 100
  m_scale_mass : 1
  m_zeta : 0.0833
  m_k_P0 : 100
  m_scale_P : 1
  m_scale_C : 0.01
  m_t : 0.0
  m_v0 : 0.0001
  m_u_min : 0
  blocks : []
  start_speed_continuous: 0.0
  end_speed_continuous: 1.0
  save_interval_dt: 10
  threshold_speed: 0.1
```

Figure 4.4: Parameters in YAML.

```yaml
Debug:
  debug_no_friction :                false
  debug_no_neighbor_springs :         false
  debug_no_stationary_springs :      false
  debug_no_damper :                  false
  debug_no_min_speed :                false
  debug_no_pad :                     false
  debug_negative_initial_values :   true
  debug_only_negative_initial :      false
  debug_no_random_displacements :   false
  debug_special_phi :               false
  debug_pad_as_block :              false
  debug_stop_slider :               false
  debug_stick_blocks :              false
  debug_write_blocks :              false
  debug_only_write_friction :       false
  debug_continuous_slider_speed : false
  debug_one_degree_freedom_mode :   false
...
```

Figure 4.5: Debug and configuration booleans in YAML.

4.3 Technologies

As described in Sections 4.1 and 4.2, C++, Python and YAML are the leading technologies used in the implementation of the model and processing of the results. This section seeks to give a brief description of the main dependencies, supporting technologies and developing platform.

4.3.1 Key components

The whole project was developed on a Ubuntu 18.04 system using the code editor Visual Studio Code. This editor provides a powerful tool when developing, but does not make the implementation dependent on it. The C++ implementation has, in addition to Ubuntu 18.04, been successfully run on Ubuntu 16.04 and Debian 9.6 through an SSH connection.

		User-specified
Package	OS availability	Version
CMake	Unix & Windows	3.10.2
GNU Make	Unix & Windows	4.1
GNU Bash	Unix & Windows	4.4.19(1)
Armadillo	-	8.400.0
yaml-cpp	-	0.5.2
C++	Unix & Windows	
C++14 python3	Unix & Windows	3.6.5

Table 4.1: Overview of key packages and dependencies.

4.3.2 Supporting technologies

Below is a list of technological tools and software with a brief description which has been useful during the project.

- Kdenlive - free, open-source video editing software[27]. Used for combining animations videos described in Section 3.9.
- InkScape - free, open-source vector graphics editor[28]. Used to draw models, produce vector graphics and correct labels after the fact in an efficient manner.
- Git - free, open-source distributed version control system[29]. Used to maintain and track changes in the code during development.
- Doxygen - free documentation generator[30]. Used to combine code documentation with source files.

Chapter 5

Results

This section presents the results of the simulations for the BK and BKP model. It starts by looking at the pad position. Moreover, with the position establishing and justifying the use of step-wise changing slider velocity ν to compare the behaviour of the two models. Through this, the difference between increasing and decreasing ν is shown, both for continuous and step-wise changing ν. Further, the output from the different models is compared and interpreted through phase plot, Fourier spectrum and animation of the simulations.

All the models of the BK and BKP model shown in Chapter 5 are simulated with $N = 100$ blocks. More details on the parameters and configurations can be seen through the YAML files in Chapter A.

5.1 Pad behaviour

As described in Section 3.1, the BKP model consist of a damped pad and N blocks connected with springs. This section looks at output and results for the pad specifically.

5.1.1 Continuously changing slider velocity

To get an overview of pads nature, the pad position for the BKP model is studied for increasing and decreasing continuous slider velocity. Both increasing and decreasing versions can be viewed individually in Figures 5.1 and 5.2 and together in Figure 5.3. In all of these figures, x is the positional value of the pad and ν is the velocity of the slider, as governed by Equation (3.1). In all runs, the pad starts with a position, and velocity equal to zero.

In Figures 5.1 and 5.4 one can see that the pad starts by oscillating a short amount of time, before being drawn back(*back* here referring to decreasing values of x). Further, the pad continues to oscillate in the range $x \in [-0.5, -1.0]$ until it reaches a region $\nu \in [0.11, 0.14]$. In this region, the pad positions position amplitude is significantly lower than the immediately preceding and following regions. This type of region is from this point occasionally addressed as a *bridge* due to its shape. Following this bridge, the pads positional amplitude follows a trend that continuously rises with higher slider velocity of ν. An increasing positional amplitude is visible until $\nu \approx 0.9$. After $\nu \approx 0.9$, the pad amplitude rather abruptly decreases toward a stable pad amplitude.

As in Figure 5.1, Figure 5.2 shows the pad position versus the slider velocity ν, but with ν decreasing from 2.0 to 0.0. The difference in velocity direction is immediately visible with the pad being drawn back at the opposite side of the velocity range. The pads position keeps a steady decreasing position and amplitude till $\nu \approx 0.68$. At this slider velocity the pad starts to oscillate with an higher amplitude similar to what is seen with the increasing slider velocity in Figure 5.1. From this point, the decreasing run appears to follow the same behaviour as the increasing velocity run. This similarity is shown in Figures 5.3 and 5.4.

When inspecting the increasing and decreasing velocity runs collectively, the domain where they differ the most is $\nu \in [0.68, 1.11]$. In order to make both lines visible when overlapping, 20% transparency is added to the decreasing run.

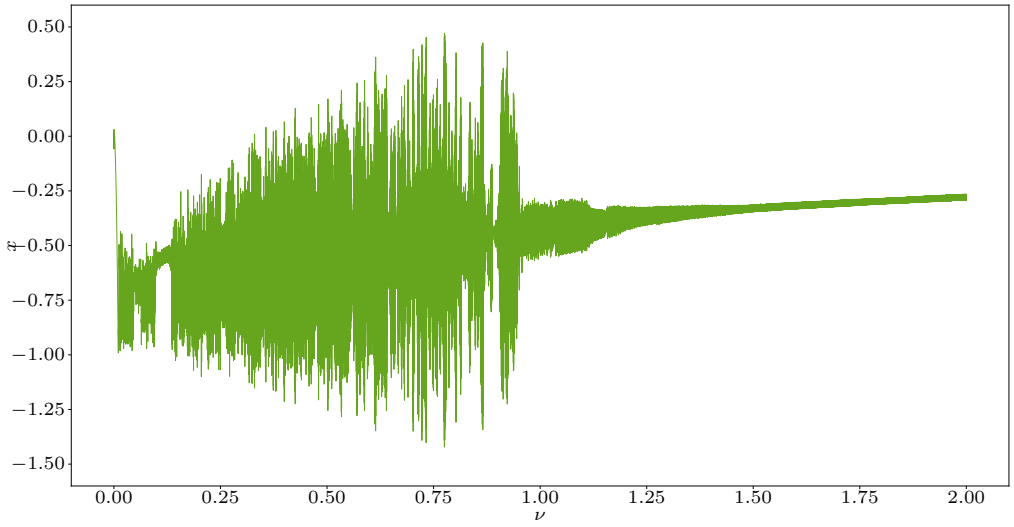


Figure 5.1: Continuously increasing the slider velocity from 0.0 to 2.0 over 80 000 time steps. The plot shows how the pad position x change as ν increases.

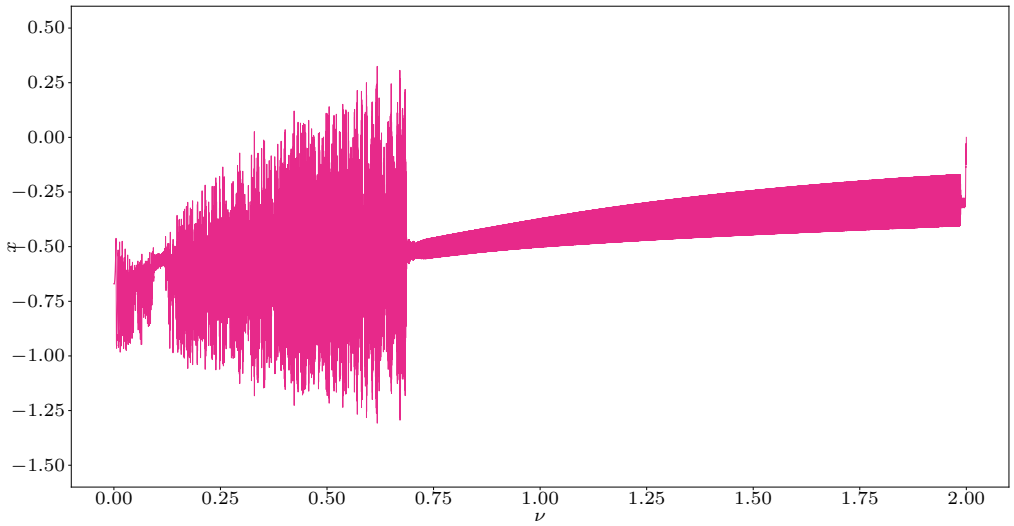


Figure 5.2: Continuously decreasing the slider velocity from 2.0 to 0.0 over 80 000 time steps. As in Figure 5.1, the plot shows how the pad position x changes as the ν decreases.

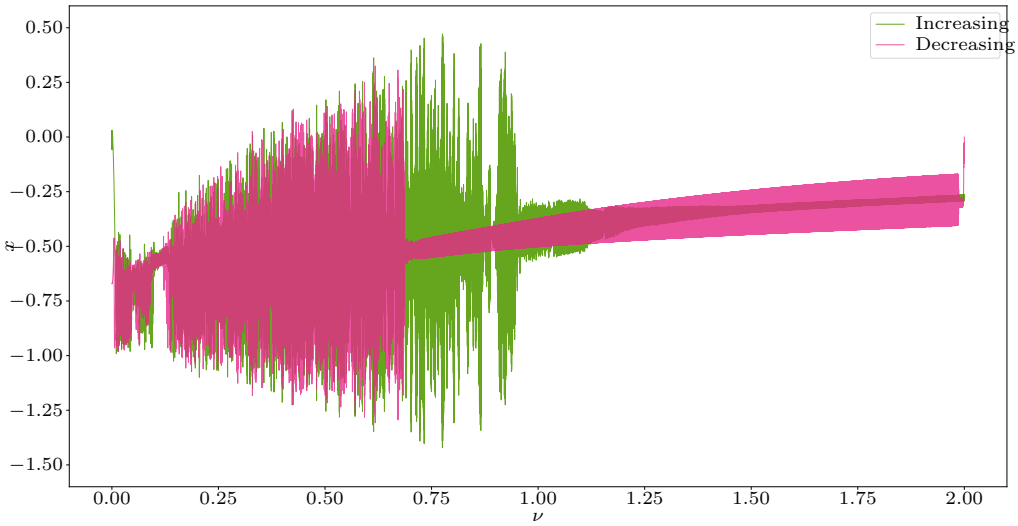


Figure 5.3: Comparison plot combining the pad position in Figures 5.1 and 5.2 to showcase where the increasing and decreasing slider velocity runs differ and matches. The figure shows clear the difference for the two runs in the domain $\nu \in [0.68, 1.11]$.

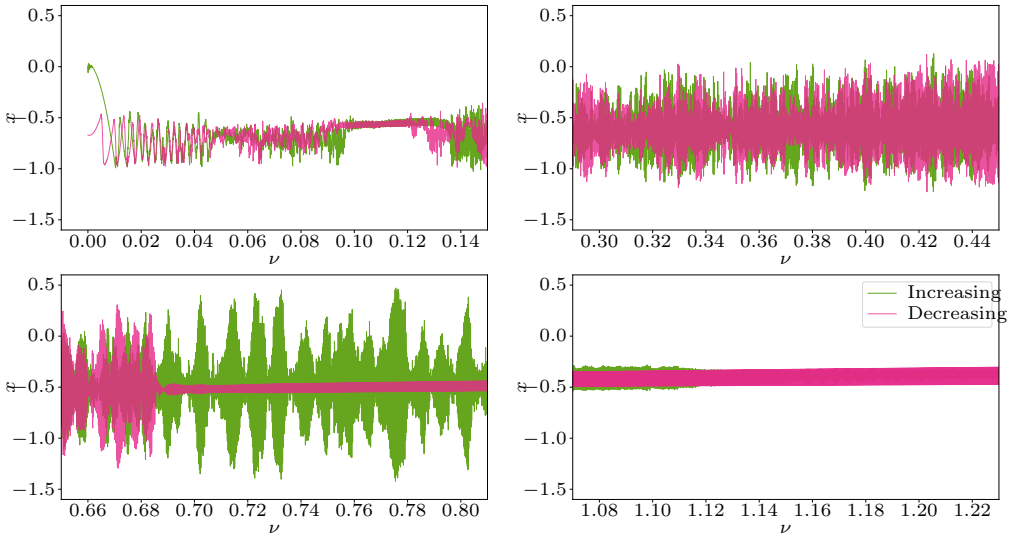


Figure 5.4: Comparison plot combining the pad position in Figures 5.1 and 5.2 highlighting some of regions of ν where the increasing and decreasing run either clearly differ from or matches the other.

5.1.2 Step-wise increasing slider velocity

In order to investigate specific slider velocities more closely, step-wise velocity acceleration was introduced. The step-wise acceleration is performed in the same manner as shown in Figure 3.18 when demonstrating slipping. As in this demonstration, the velocity is held constant for a set amount of time. Further, the slider accelerates a predefined amount at the end of each interval. In Figure 5.5 the plot with continuous slider velocity Figure 5.4 is repeated with step-wise increasing or decreasing slider velocity.

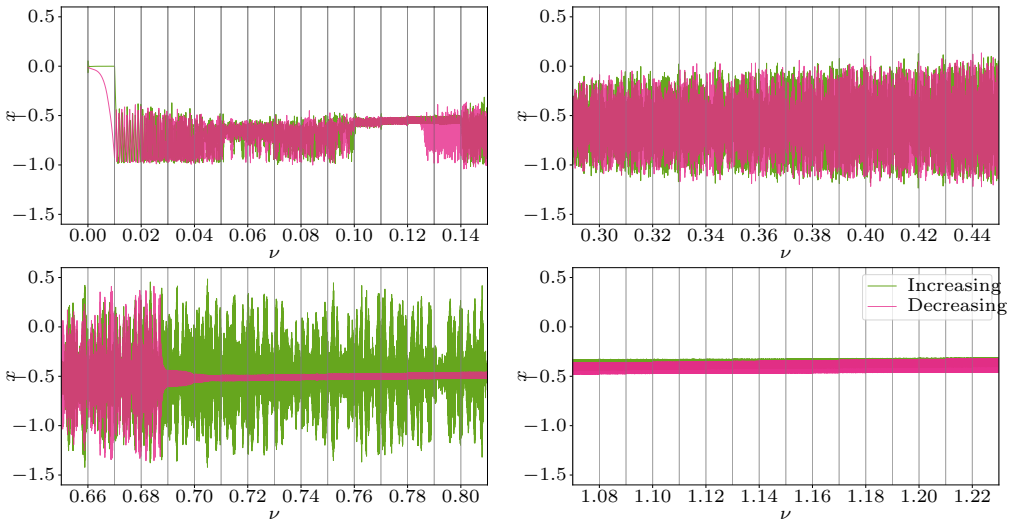


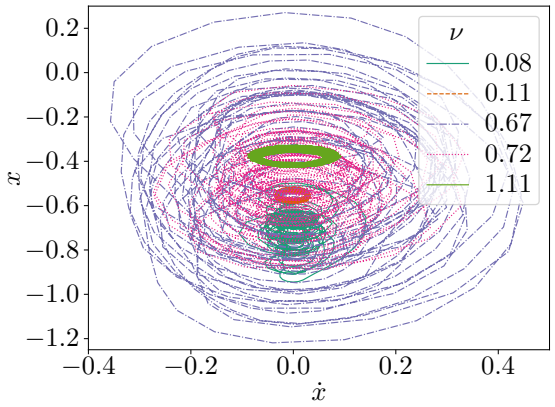
Figure 5.5: Step-wise changing the slider velocity ν every vertical line. The velocity increase with 0.01 every 2000 time step in the simulations. The plots indicates the difference and similarities of running continuous or step-wise increasing slider velocities.

The runs shown in Figure 5.5 uses seed 112 from Figure 5.15 and seed 101 from Figure 5.10. An alternative method to step-wise acceleration was also tested. This method used continuous slider acceleration until a threshold velocity is reached. From that point, the simulation ran for a set amount of time. This method was implemented and tested but was not used to calculate the friction amplitude and frequencies for different slider velocities. The choice to not use it was due to the higher computational cost this would take without seeing any major improvement. It is possible that this method can reveal aspects of the models when tested more and is written up as Section 6.5.

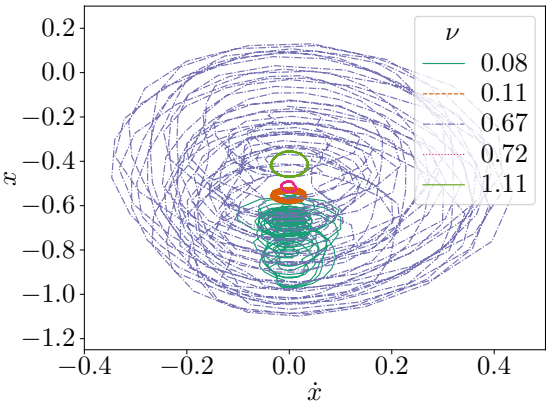
5.1.3 Phase plot

In order to get information about the pad at a specific slider velocity, the velocity of the pad is plotted against the position. This plotting produces a phase plot that can give a picture of how regular the model is behaving and if there are any predominant fluctuations. In Figure 5.6, phase plot from both runs in Figure 5.5 for five slider velocities are plotted. These velocities are chosen to inspect regions that behave differently when looking at the pad position.

The phase plots also visualise some of the similarities and difference of decreasing and increasing slider velocity. For $\nu = 0.72$, the difference in amplitude observed in Figure 5.5 is very clear. This difference is also present for the pad velocity $\dot{x}$, with Figure 5.6a showing a more chaotic path than in Figure 5.6b. The other phases visualised show more similarities with the decreasing runs showing a more spherical shape. The shape here refers to the visual shape with the set axis limits. Figure 5.6 is meant to showcase how the pad behaves quite untidy at some values of ν compared to others. Combined with the three-dimensional visualisation in Figure 5.7, it shows that the pad gets into a stable phase at certain slider velocities.

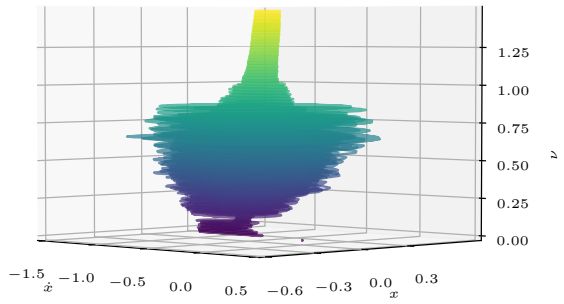


(a) BKP model with increasing slider velocity. $\nu = 0.67$ or $= 0.72$ show a much more irregular pattern than $\nu = 0.11$ and $= 1.11$, which oscillates with a stable position and velocity. As in Figure 5.6b, $\nu = 0.08$ show an irregular path, but with lower minimum and maximum values for position and velocity.

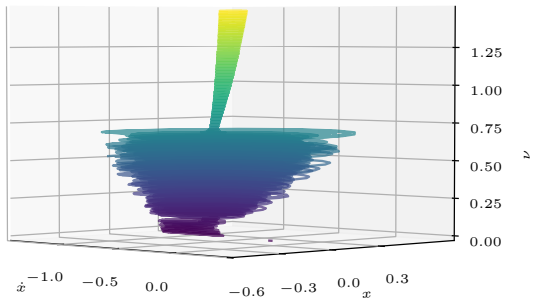


(b) BKP model with decreasing slider velocity. $\nu = 0.67$ show more dense, but otherwise similar path as in for the increasing slider velocity run in Figure 5.6a. At the higher $\nu = 0.72$, this similarity is gone, now oscillating with a stable velocity as with $\nu = 0.11$ and $= 1.11$.

Figure 5.6: Phase plot of the pad position x versus velocity $\dot{x}$. The phase is plotted for different slider velocities ν. Comparing the two one note that some velocities shows a higher $x/\dot{x}$-range ratio Note that the axis range is different for x and $\dot{x}$ which affect the shape of the phases.



(a) Increasing slider velocity for plotted against the pads position and velocity. The Phase has clear decrease at $\nu \approx 0.84$, followed by a linear decrease before stabilising at $\nu \approx 1.0$. Individual phase lines is shown in Figure 5.6a.



(b) Decreasing slider velocity for plotted against the pads position and velocity. A clear change in the phase is present at $\nu \approx 0.68$, as is shown though the $\nu = 0.72$ and $= 0.67$ phases in Figure 5.6b.

Figure 5.7: Both figures are originally 3D plots made of 2D phase plot for velocities $\nu \in [0.0, 1.49]$ stacked upon another. Such 2D phase plots is shown in Figure 5.6. The colour of each phase is directly dependent on the slider velocity ν, going from dark blue to yellow. Going bottom to top, Figures 5.7a and 5.7b, the general behaviour for the decreasing and increasing runs are the same till $\nu \approx 0.68$ as seen in Figure 5.5. From here the decreasing run has drastic contraction in positional and velocity range relative to the region prior. This contraction is also present in the increasing run, here happening at a higher slider velocity and less to a lesser extent.

5.2 Comparing BK and BKP

In order to compare the BK to the BKP model, the friction amplitude as described in Section 3.6 was compared for both step-wise increasing and decreasing slider velocity. In order to get a general behaviour of the models, an average of multiple runs of each model was done for decreasing and increasing slider velocity. The mean friction and friction amplitude from the models is shown together in Figure 5.12. Note that the number of intervals used to calculate the error bars for the friction amplitude as in Equation (3.19) is 4 for all runs. These error bars show the error when calculating the friction amplitude from a signal, which here is the friction. To support this, the mean friction amplitude is visualised with a 95% CI in Figure 5.8. While the error bars for the friction amplitude present the error when calculating the amplitude, the CI shows how much the calculated amplitude for each run differs from the mean of all the runs. These tools enable saying something about the accuracy of the amplitudes and if the models show more constant or varying behaviour at different slider velocities. To support the figures in this section, additional, but not essential figures containing visualisations of how each run used to calculate mean differ from it are available. These are figures are found in Chapter C, Section C.4 in the appendices. All of the runs compared in Section 5.2 have the same step size, number of time steps and range of velocities. The general YAML file can be found in Chapter A containing all the parameters, while the key once are listed in Table 5.1. The runs has a constant slider velocity for an interval which is given in Table 5.1.

	Parameters		
Parameter	Symbol	Increasing	Decreasing
N	N	100	-
slider_speed	ν	0.00	1.49
increment		0.01	-0.01
interval		2000	-
max_time		300000	-
	Debug		
Parameter	Symbol	BK	BKP
debug_no_pad		true	false

Table 5.1: Key parameters used when comparing the increasing and decreasing slider velocity simulations of the BK and BKP model. - mean the value is the same as the column to the left. The combination of slider_speed, increment, interval and max_time give both the increasing and decreasing runs 150 velocity steps in $\nu \in [0.0, 1.49]$.

Visualised in Figure 5.8 is the mean friction amplitude with corresponding 95% confidence interval. This figure enables comparison of the BK and BKP model, the simulations within each category, and how the direction of the velocity of the slider affect the systems. From the figure, it is clear that in the first few slider velocities, the BKP has a higher friction amplitude than the BK model. This difference is only for a few velocities, keeping in mind that the ν-axis is plotted on log-scale. All of the simulation types behave very similarly in the beginning. The similar behaviour is especially clear in the sudden *dip* experience around $\nu = 10-1$, shown in Figure 5.8. After this simultaneous drop, the different simulations increase in amplitude at various paces, then experiencing another dip of varying magnitude and width. Succeeding these drops in friction amplitude, the BK and BKP mode seems to stabilise at an increasing amplitude value independent of the slider speed increasing or decreasing. It is clear to draw from the figure that the BKP has a lower mean friction amplitude in this region, $\nu \in [0.25, 0.68]$. Around this point, all of the curves start to decrease with wider CI than the previous region. Eventually, all of the curves stabilise with the BK model with increasing slider velocity having a wider CI than the other. It is also quite notable how wide the CI interval is from $\nu = 0.99$ to 1.34. This distinctive width is due to an outlier seen in Figure 5.9 and animated in Figures 5.21 to 5.23.

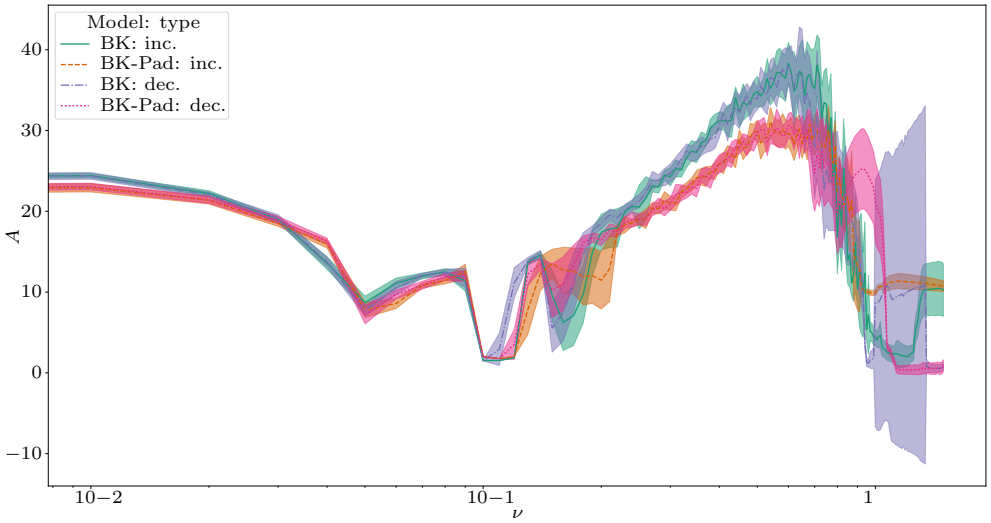


Figure 5.8: 95% CI of the friction amplitude for increasing and decreasing simulations of BK and BKP model. Each mean and CI is produced from 16 runs with different initial position and velocity for the blocks. The mean lines and CI intervals show where each run type has consistent behaviour and how it differs from the types. $\nu \in [0.0, 1.49]$ plotted with logarithmic axis

5.2.1 Decreasing slider velocity

As the run shown with the position of the pad in Figure 5.2, runs with decreasing slider velocity for the BK and BKP model was performed. As is clear in Figure 2.2, the BK model has no pad and therefore friction force is inspected. This measure is also used in Section 5.2.2.

In Figures 5.9 and 5.10, the dips seen in Figure 5.8 starting at $\nu = 0.1$ are both visible. One can also see that not all the runs experience the second dip. In the runs where the friction dip two times, the paths are similar. Further, the BK model enters the second dip more often than the BKP model, comparing 16 runs of each. This trend is detectable in Figure 5.12, where the second dip in the BKP model is tighter and shallower. Zooming in on the dips - see Figures 5.11 and 5.13 -, the second dip varies more than the first for both models with decreasing slider velocity. A faster increase in friction amplitude is also visible in the BK compared to the BKP model. Note the higher amplitude limit in Figure 5.9, which is due to the outlier.

Furthermore, it should be noted that the error bar, in, e.g. Figure 5.11 sometimes overlap, making the error seem bigger than it is. Further, the ν-axis is not shown with log-scale due to the smaller interval.

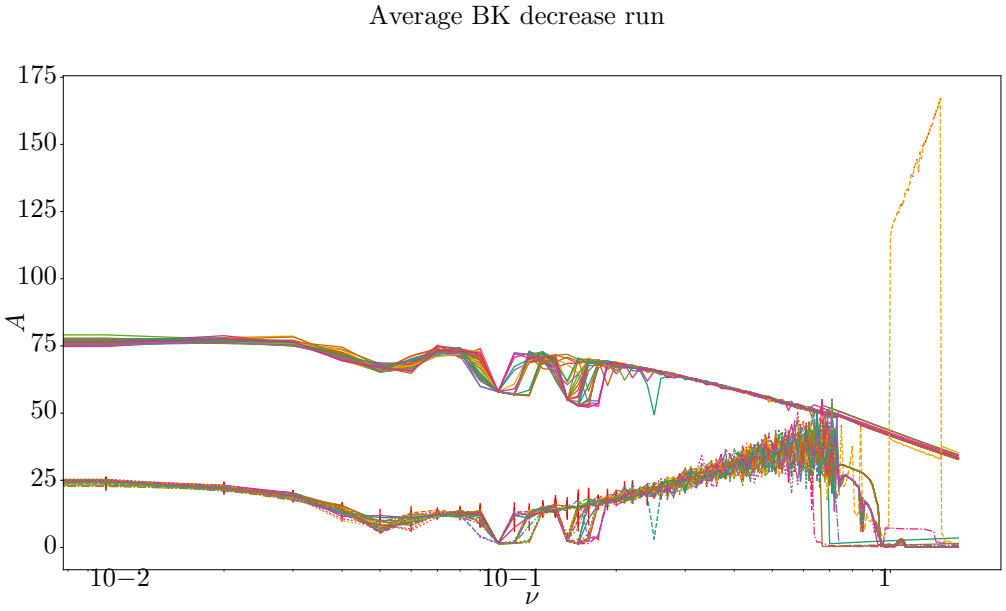


Figure 5.9: 32 unique runs of the BK model with step-wise decreasing velocity. The top lines shown represent the mean value of the friction and the bottom lines the friction amplitude. $\nu \in [0.0, 1.49]$ plotted with logarithmic axis.

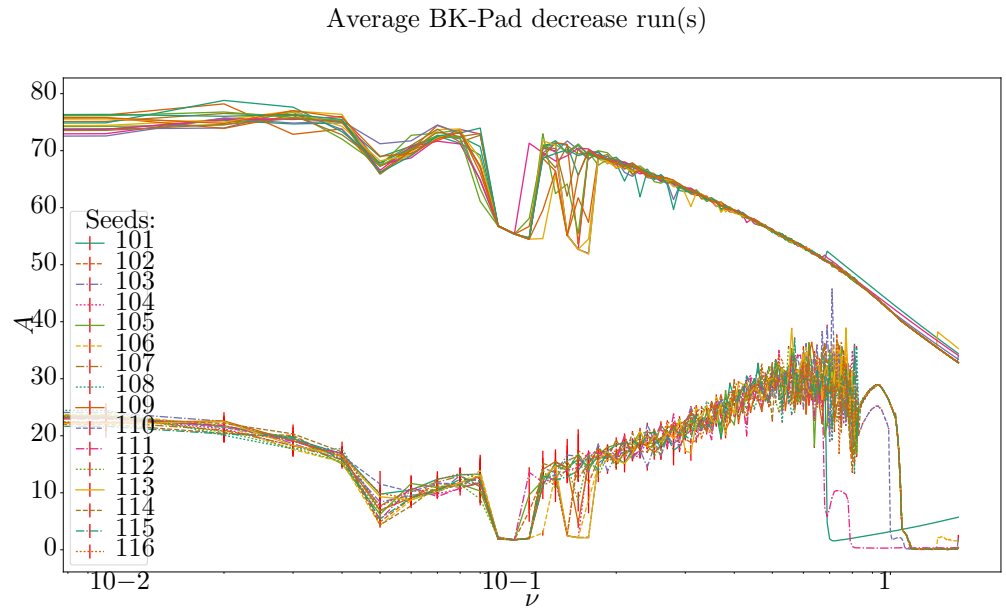


Figure 5.10: 16 unique runs of the BKP model with decreasing velocity. The top lines show represent the mean value of the friction and the bottom lines the friction amplitude. One of the runs clearly differ from the other in friction amplitude with a slightly lower mean friction. This run represented by a yellow dashed line differs to such a degree it is clearly widening the CI in Figure 5.8. $\nu \in [0.0, 1.49]$ plotted with logarithmic axis.

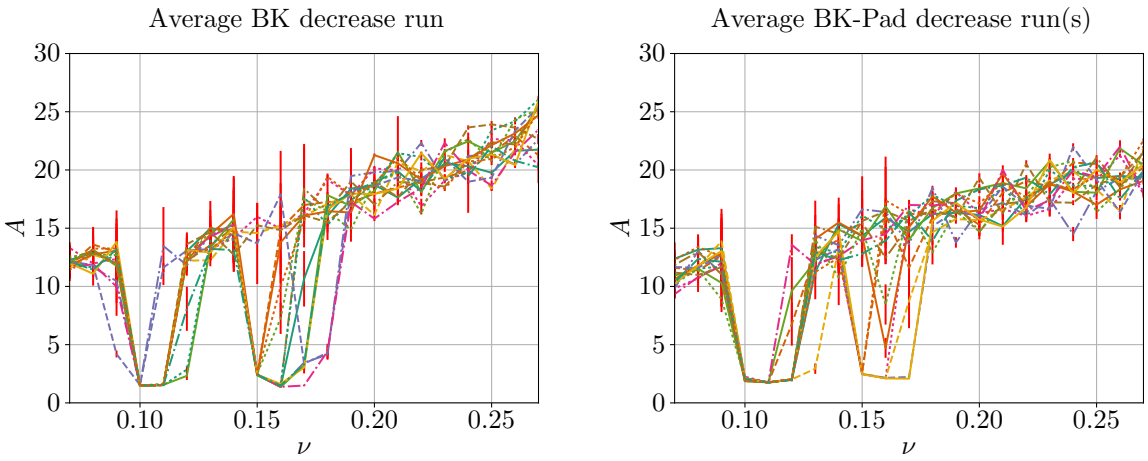


Figure 5.11: Dip region, comparing 16 runs of the BK and BKP model with decreasing slider velocity $\nu \in [0.08, 0.27]$. The two figures show how 16 different runs of each model behave in the region of the two *dips* apparent in the mean in Figure 5.8. The error bars appear bigger for some runs than is the reality due to bars from multiple runs overlapping. Grid is added to make the values easier to read.

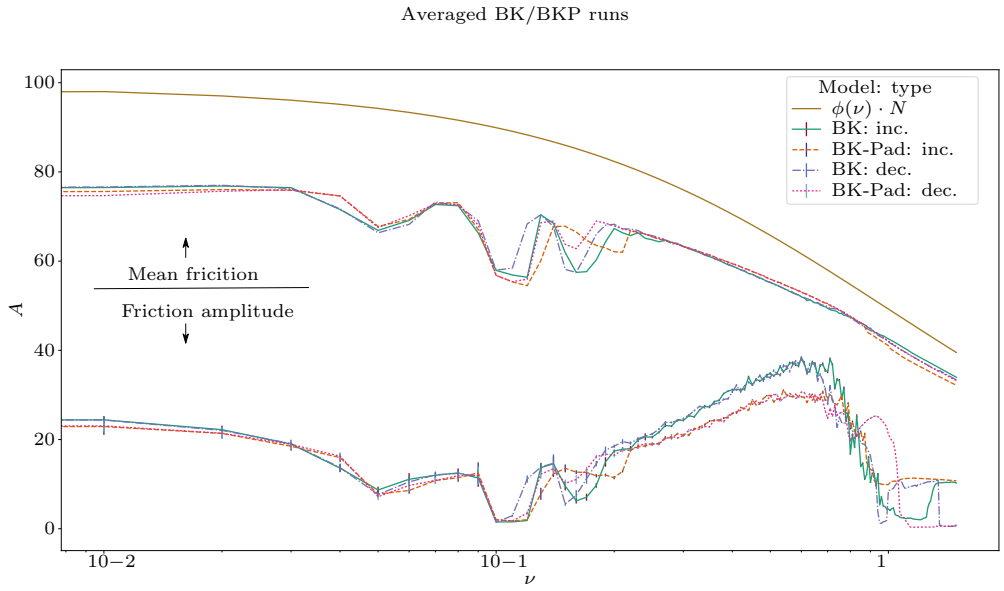


Figure 5.12: Mean and error for the friction amplitude of increasing and decreasing simulations of BK and BKP model. Each mean and error is produced from 16 runs with different initial position and velocity for the blocks and is represented by the lower lines. Each mean shown in the lower line has a corresponding mean of mean friction line among the upper lines with the same colour and line style. The topmost line is the friction law shown in Figure 3.2 scaled with the number of blocks for further comparison. $\nu \in [0.0, 1.49]$ plotted with logarithmic axis. Non-logarithmic version in Figure C.15

5.2.2 Increasing slider velocity

Continuing the comparison of decreasing and increasing slider velocity, the increasing part is inspected more closely. As done with the decreasing segment in Figure 5.11, the two dips present were focused. There is some immediate difference in the dips when changing the direction of the sliders acceleration. One change the fact that the dips are in some runs wider. From Figure 5.13 the wider dips are true for both the BK and BKP model regarding the second dip and the BKP model in the first.

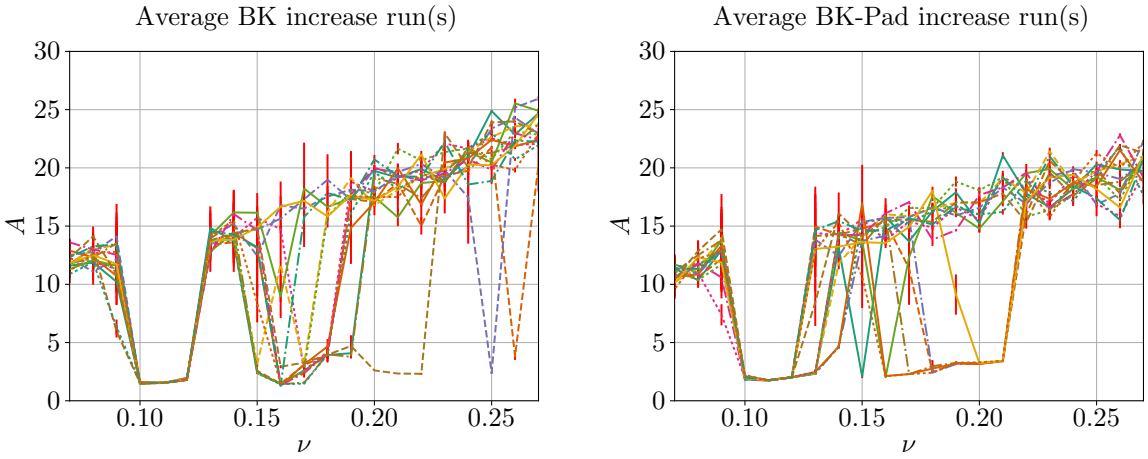


Figure 5.13: Dip region, comparing 16 runs of the BK and BKP model with increasing slider velocity $\nu \in [0.07, 0.27]$. The two figures show how 16 different runs of each model behave in the region of the two *dips* apparent in the mean in Figure 5.8. The error bars appear bigger for some runs than is the reality due to bars from multiple runs overlapping. Grid is added to make the values easier to read.

Further inspecting the mean in Figure 5.12 of the runs plotted in Figures 5.14 and 5.15, the second dip shallower is in the BKP model. This is consistent with the findings in Section 5.2.1 in regard to the second dip. It differs though in it not being wider than its BK counterpart.

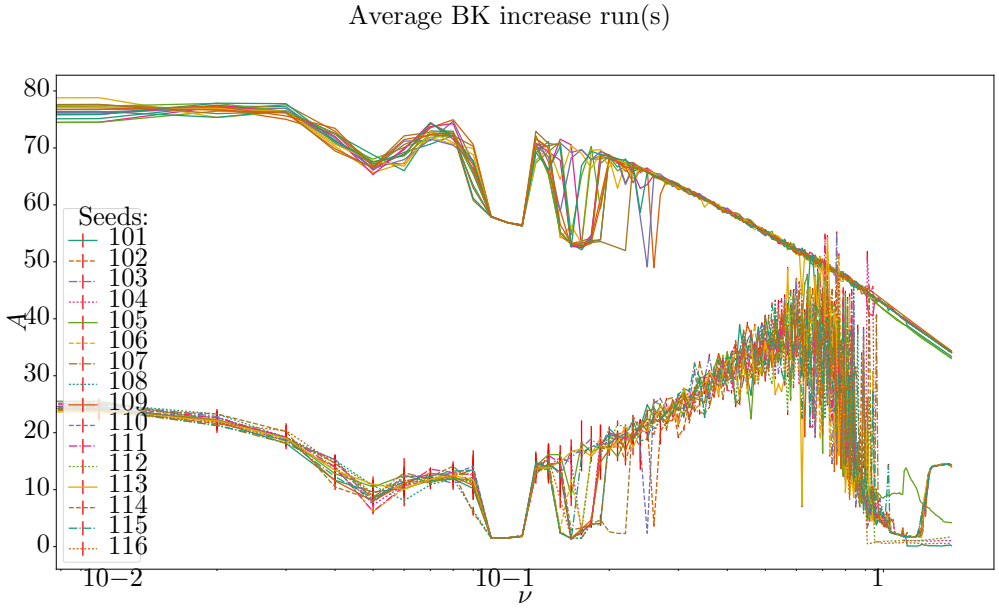


Figure 5.14: 16 unique runs of the BK model with increasing slider velocity. The top lines show represent the mean value of the friction and the bottom lines the friction amplitude.

It is worth noting that the runs stabilise on average at different friction amplitude when the slider is decreasing rather than increasing in velocity.

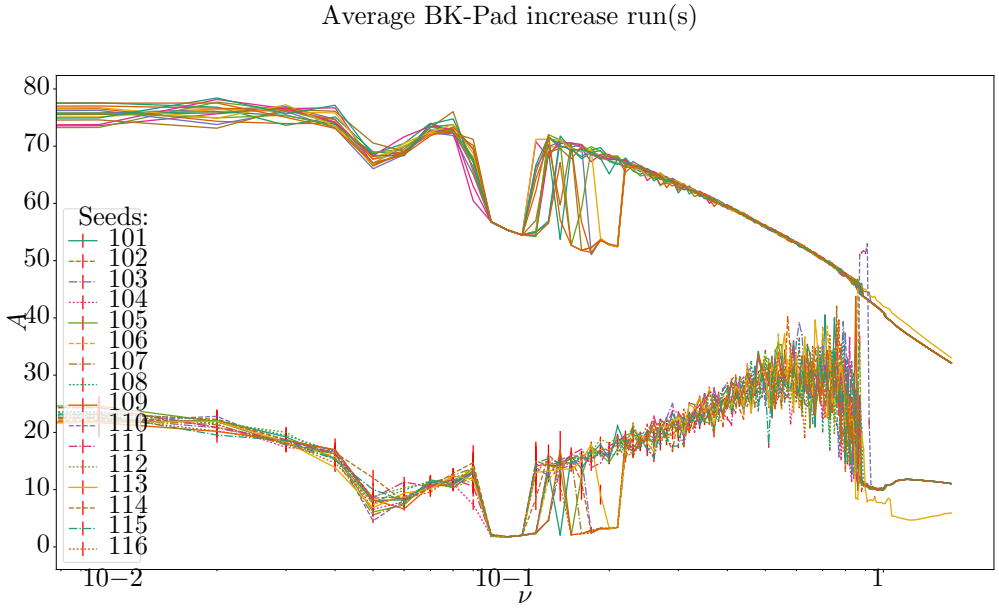


Figure 5.15: 16 unique runs of the BKP model with increasing slider velocity. The top lines show represent the mean value of the friction and the bottom lines the friction amplitude.

5.2.3 Pad vs. no pad

To further investigate the BK and BKP, this section contains a comparison of the use of pad and no pad in the models. This investigation is done by comparing the phase plots and Fourier spectrum of specific blocks for the BK and the same for pad and blocks in the BKP model. In all runs used in Figures 5.16 to 5.18 and Section C.3 showcasing blocks, the save interval is set to ten. Setting the save interval to this means every tenth output from the simulation is saved. Using the spectrum theorem referenced in Section 3.10 this enables the DFT to detect the highest frequencies in the block record using a save interval of one. It also gives a sufficient margin to the blocks natural frequency calculated by H. Ferre in his thesis[1] to 2.256.

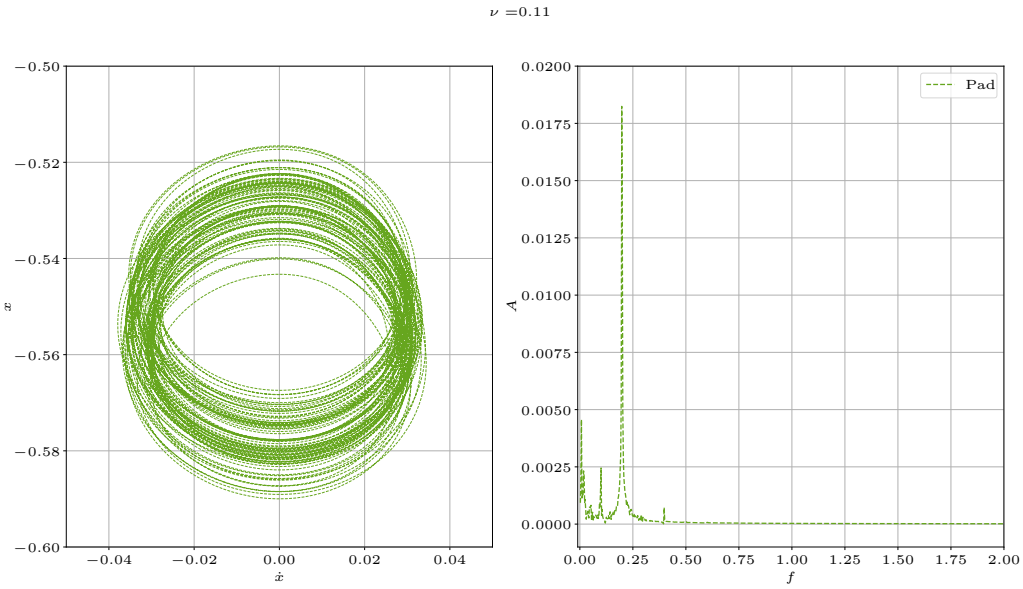


Figure 5.16: Showing the phase plot to the left and Fourier spectrum to the right of the pad for the BKP model with increasing slider velocity $\nu = 0.11$. Note the axis are different that those of Figures 5.16 and 5.18 in order to see the values of the two plots. Seed value is 112.

The phase plot and Fourier spectrum in Figures 5.16 and 5.17 are plotted from the same instance of the BKP model. As with the BK model in Figure 5.18 output for six different blocks are visualised. The number of each block corresponds to its respective position with 0 being the leftmost and 99 the rightmost block in a 100 block system.

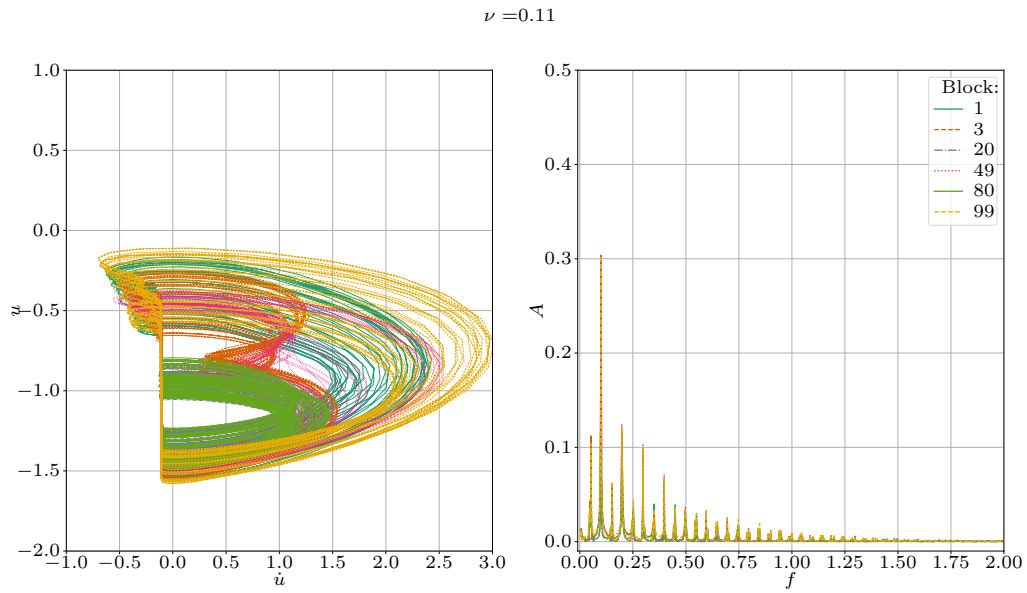


Figure 5.17: Showing the phase plot to the left and Fourier spectrum to the right of six different blocks for the BKP model with increasing slider velocity $\nu = 0.11$. Seed value is 112.

In Figures 5.16 to 5.18 a single simulation of the BK and BKP model is visualised. In the plots, there are similarities in the shape of the phases and peaks in the Fourier spectrum for the blocks. Both models have the same clear peaks with the BKP model having some lower once in between. Another difference is seen in the phase plot, where the BK model has a more condense pattern. It should also be noted

that the BK model has a higher maximum and lower minimum position u and approximately the same minimum and a lower maximum velocity $\dot{u}$ than the BKP model. Further, it is observed that the same blocks have the same general path in both phase plots.

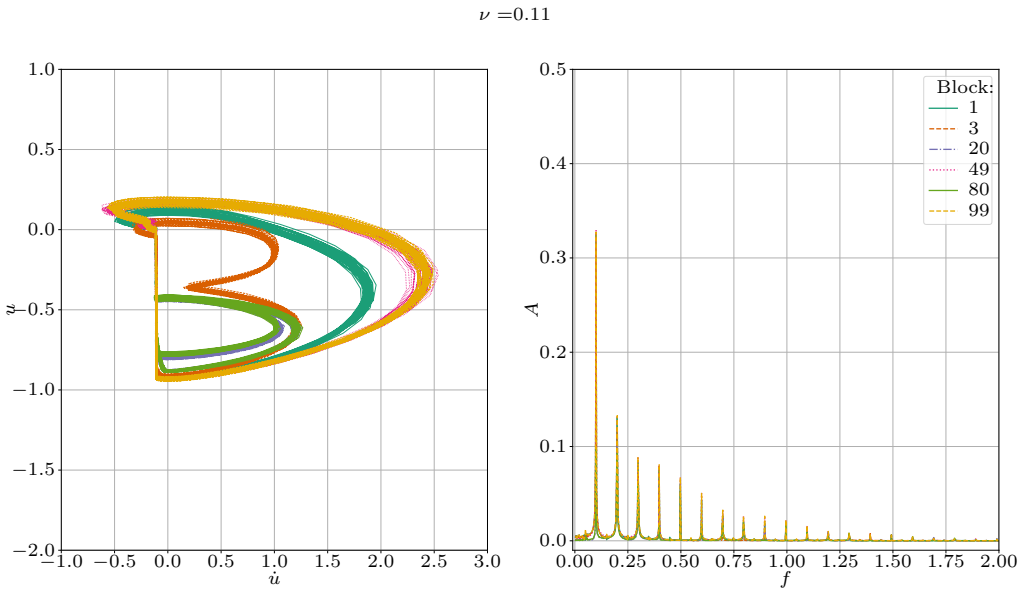


Figure 5.18: Showing the phase plot to the left and Fourier spectrum to the right of six different blocks for the BK model with increasing slider velocity $\nu = 0.11$. Seed value is 113.

The more condense pattern for the BK model is also present simulating with negative slider acceleration, see Figure C.6.

Another observation is that the similarity in shape, frequency and amplitude for the models changes for both increasing and decreasing at higher values of ν. Further, increasing and decreasing simulations show a difference in the mentioned measurement at these velocities. Furthermore, these behaviours of each simulation type are consistent with further increase in velocity. For visualisations, see Section C.3.

5.3 Fourier spectrum

The Fourier spectrum can as explained in Section 3.10 can be used to calculate the amplitude and frequency of a periodic wave function. In Figures 5.19 and 5.20, two different disc velocities are shown through the phase and amplitude frequency plot of a increasing BKP run. The Fourier spectrum is calculated on the pad position. From the figures, all the axis except the f-axis are quite different. Taken the difference in the axis, the amplitude of the pads position is clearly lower at $\nu = 0.12$ than $\nu = 0.70$. Since both are from the run with seed 112 shown in Figure 5.5 this is consistent with earlier results.

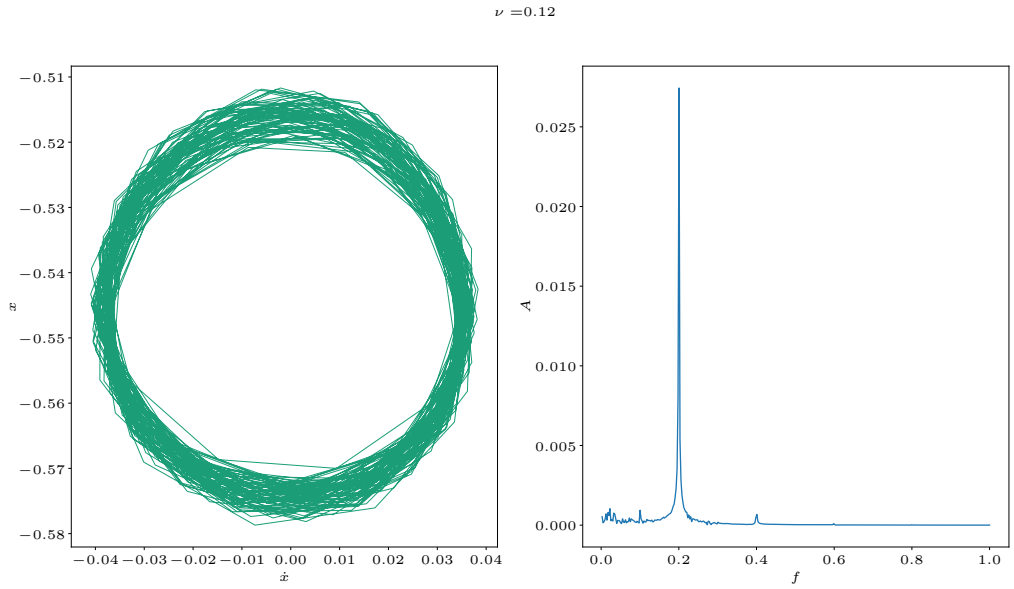


Figure 5.19: Phase plot and Fourier spectrum for the pads position with $\nu = 0.12$ in a BKP run with increasing slider velocity.

Another thing to point out is that there is a clear peak in the Fourier spectrum for Figure 5.20 and that value of f is lower. The frequency f decrease at the higher velocity as seen in Figure 5.20 and show a bigger range of motion.

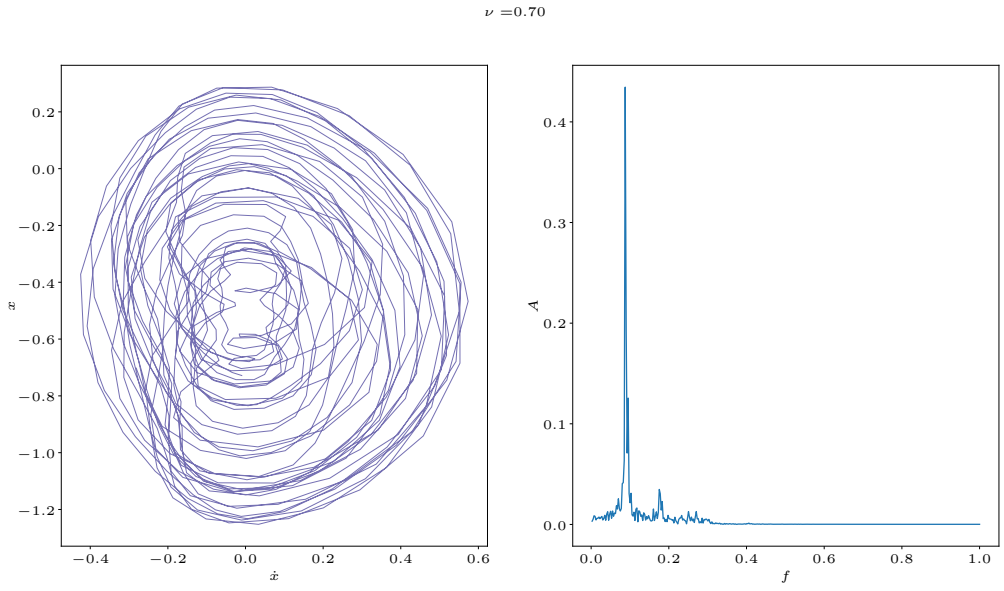


Figure 5.20: Phase plot and Fourier spectrum for the pads position with $\nu = 0.70$ in a BKP run with increasing slider velocity.

5.4 Animation

The clear outlier shown in the interval $\nu \in [1.34, 0.99]$ of Figure 5.9 raise a question for what happened for this particular run of the BK model with step-wise decreasing slider velocity. To identify this, the animation tool comes in handy. In order to fully utilise the developed animation tool, the outlier run was reproduced. The rerun was done so the output contained data of all the blocks. Due to the required disk space, this is not the default as is explained in Section 3.7. Here the use of seed comes in useful and makes it possible to rerun the same simulation with all the blocks logged.

In Figures 5.21 to 5.23 snapshots from the animation of the outlier is shown. These snapshots show the behaviour of the model before, during and after this interval. The animation is implemented so that when the first data point with a different slider velocity reaches the left side of the *boxes* in **A** and **C**, the velocity at the top of the animation changes. This velocity refers to the velocity of the slider ν. In Figure 5.21, the first velocity ν that has bigger positional, as well as friction amplitude than the mean is visualised. The data with slider velocity $\nu = 1.34$ starts at *Time steps shown* 40. At this time step, the blocks show a clear and synchronous waveform in the left window. This synchronous oscillation is apparent in the averaged block position in the right window. The average also shows the same underlying period in the blocks is kept from the previous velocity of $\nu = 1.35$, with a clear increase in amplitude. The increased amplitude is present in the friction plot until $\nu = 0.99$. In order to see if the behaviour in Figure 5.21 continues, the animation is also started in the interval and towards the end of it.

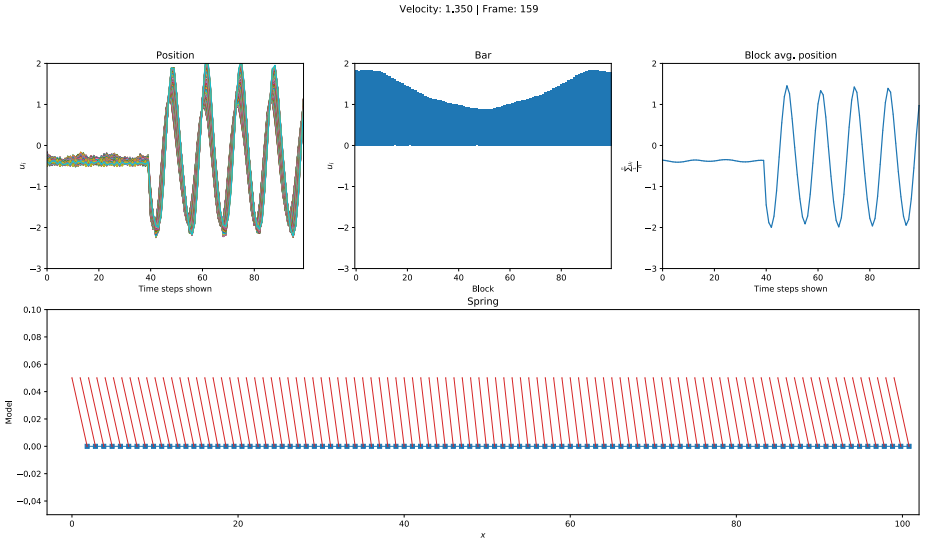


Figure 5.21: Animation interface showing frame number 159 in a BK model run with decreasing slider velocity. The figure shows when the run has a sudden spike in both the amplitude of the friction as well as the block positions.

Figure 5.22 shows the outlier run 50000 time steps later, since each velocity is run for 2000 time steps each. One can see that the behaviour of the blocks still is a periodic wave as in Figure 5.21. Figure 5.9 shows how friction amplitude decreases over time in the region considered. This decrease is also present in the averaged block position at $\nu = 1.09$. Furthermore, one can note that the maximum and minimum value for individual blocks in the left window has changed far less than the average. The decrease in the blocks average position comes from the blocks being less synchronous, which can be seen in the group of position lines being thicker in the left window.

At $\nu = 0.99$, the outlier in Figure 5.9 drops in fiction amplitude showing values closer to the mean in Figure 5.12. This drop is inspected by animating the end of the outlier region. At the point the amplitude drops, a clear change in the behaviour of the blocks is seen. This is clearly visible at *Time steps shown* 50 for the individual and averaged block positions animated in Figure 5.23. In the actual video, one sees how the blocks are no longer oscillating synchronously. Rather, the blocks seem to have multiple wave patterns moving through them. These patterns result in a clear decrease in the average block position. This decrease indicates that the outlier occurred when the blocks started to act in a synchronous, close

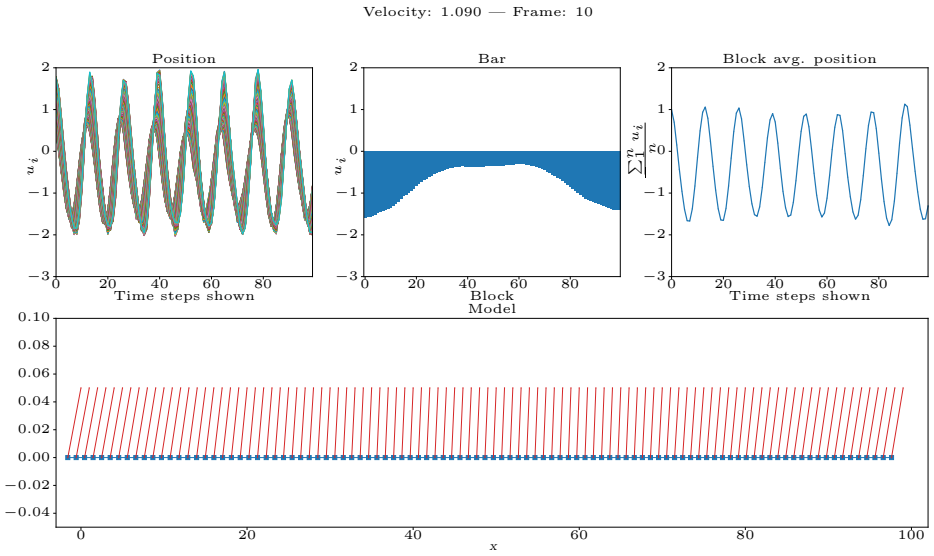


Figure 5.22: Animation interface showing frame 10 of the same run as in Figure 5.21. The animation frame shows a later stage in the simulation at $\nu = 1.09$.

to harmonious wave pattern and ended when exiting this pattern.

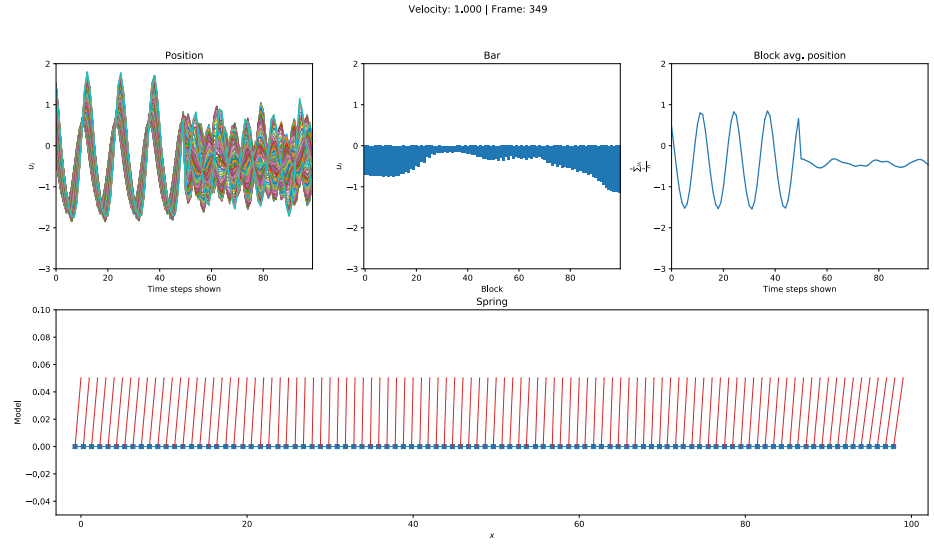


Figure 5.23: Animation interface showing frame 349 of the same run as in Figures 5.21 and 5.22. The animation frame shows when the run exits the ourlier region.

5.4.1 Dips

In Section 5.2 two dips in the in the friction amplitude are presented. It was also shown that the second of these dips were not always present. In order to inspect the properties of the different runs, animations of the system were created for the region of interest. In Figure 5.24 two frames are plotted of runs behaving differently at the same velocity ν. This figure shows two simulations of the BKP with the slider having positive acceleration.

From the animation frames, it is possible to see that the position of the blocks (upper left in Figure 5.24 has a more noisy shape as well as the pad. In the run with seed 116, the pad is hardly moving in this

region compared to the run with seed 112. This difference breaks with the behaviour of the systems in the first dip. The animations reveal that they here behave very similar, something also shown in the friction amplitude, see Figure 5.15.

5.4.2 Animation videos

In the appendix presented in Chapter B, a quick guide and summary of a web page hosted on a repository belonging the authors GitHub profile. The page contains videos of animations used in the Thesis.

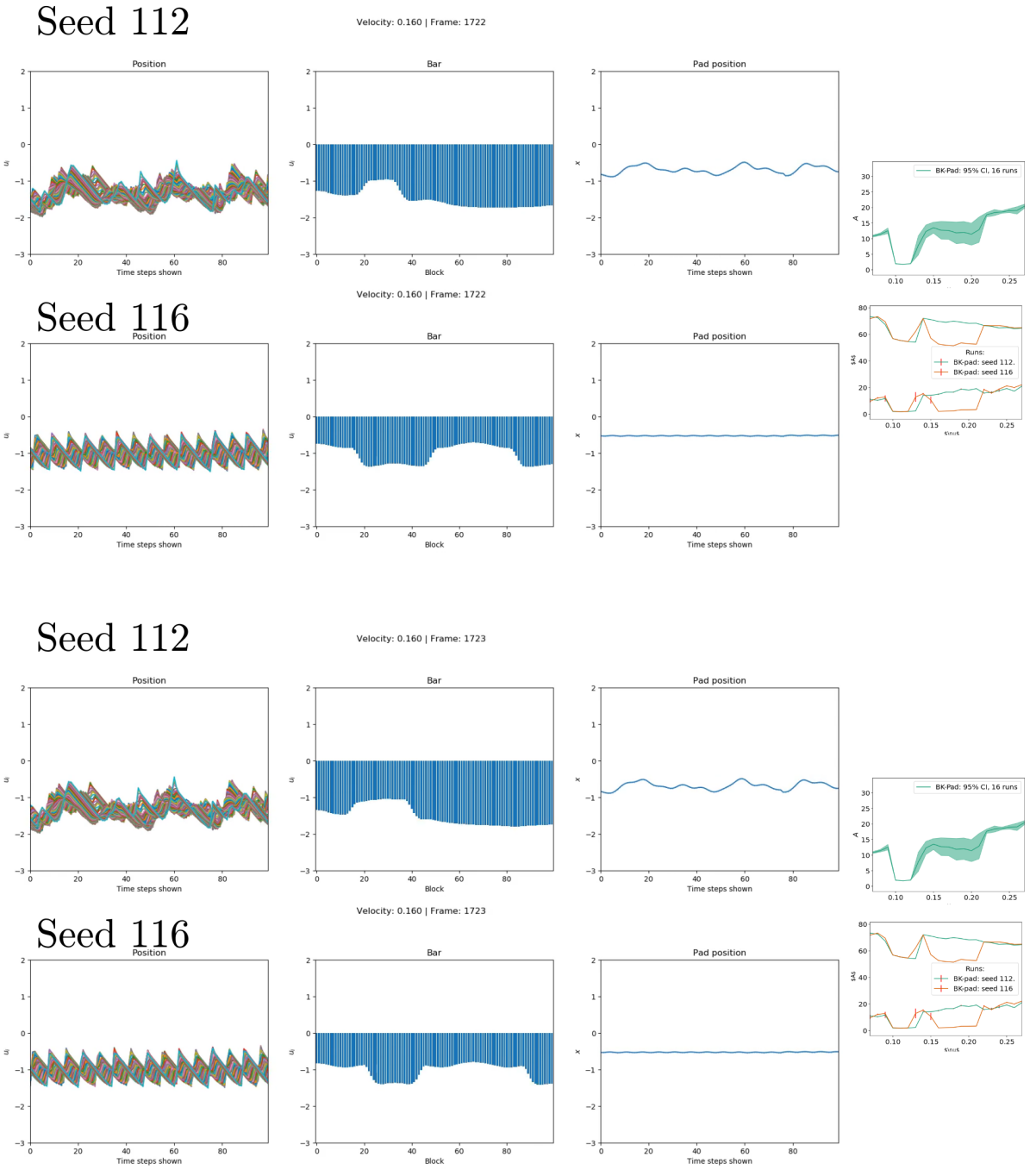


Figure 5.24: Two frames from combined animation plot. Here two runs of the BKP model is compared. The slider velocity is increasing step-wise, as shown in Section 5.1.2. The combined shown as well as a corresponding animation for the BK model is hosted on a GitHub pages as is detailed in Section 5.4.2.

Chapter 6

Discussion & Conclusion

6.1 Increasing vs decreasing slider velocity

In Section 5.1.1, the difference between decreasing and increasing the slider velocity is clear. It is also possible to see that this difference changes with the slider velocity. At high velocity, it affects the magnitude of the pad position to a lesser degree than at lower velocities. However, as seen in Figure 5.8 with the mean friction amplitude, the consistency of the different directions also varies with the velocity. An example of this is around the second dip in the friction amplitude, again looking at Figure 5.8. Here the CI is wider for the increasing than the decreasing runs. The wider CI might be due to the amount of time step before this point in the simulation. Since log scaled is used, lower values of ν is *stretched* out. This stretch means the first and second dip occurs in the first 20% of the simulation for increasing runs and the last 20% for decreasing run. The specific percentages are specific to the simulations presented in Chapter 5. Due to the general stability of the increasing simulations at lower velocity, the potential contribution of this is likely little, if any. However, it should be considered in further work. Doing bigger batches could reveal more on this matter. Furthermore, Figures 5.4 and 5.5 show that the general behaviour of the simulation is intact when running step-wise increases in the slider velocity. Nevertheless, it is also seen that there are evident differences between step-wise and continuous change in slider velocity. This difference is mainly shown in less positional magnitude change for the step-wise runs. Reason for this can be; more time to stabilise at the respective velocities, the bigger increases or decreases in velocity for the step-wise runs is more disruptive than it is for continuous runs,or a combination of the two. For further work, it is worthwhile to use finer changes in slider velocity steps to investigate if there are any new behaviours discovered.

The phase plots in Figures 5.6 and 5.7 builds on the observations made for the pads positional plots in Figures 5.3 to 5.5. Especially the showing where the pad have a more chaotic behaviour and the general domains where it is consistent. It shows that there is a connection between the pads positional range and how harmoniously the pad moves. Combining this connection and the findings in the friction plot, emerge a behavioural pattern. At lower friction and positional amplitude, the system behaves more harmoniously and periodically — a behaviour which is consistent for both the BK and BKP model. Though they are very similar, they evolve towards it at different time steps or slider velocities.

Most visible in Figure 5.12 showing the averaged run types, is how the increasing and decreasing runs converge to different values at the higher end of the slider velocity range. This converged value is independent of the model. The independence could be due to the decreasing runs starting with the position and velocity of the pad being zero. This explanation is the most could be right for the BKP model, but not for the BK model, which has no pad. It should also be noted that the increasing BK line shows a decreasing tendency before converging to $A \approx 11$ like the corresponding BKP line.

6.2 BK vs. BKP

With a basis in 16 runs of each the BK and BKP model with increasing and decreasing slider velocity, some difference can be described between the two models. As seen in the log-scaled curves in Figures 5.8 and 5.12, all of the four run types contain two dips in around $\nu = 0.11$ and $= 0.16$, with the first dip

being complementary among the four and the second less so. The second drop is evident in all the curves, but is deeper and sharper for the BK model. The CI interval is also wider for the BKP model, suggesting the simulations do not always go into the dip. This is confirmed in Figures 5.11 and 5.13. While all the runs to some degree enter the first dip, not all enter the second. Further, skipping the second is more prominent for the BKP model. In H. Ferre's Thesis, he stated that it would be interesting to figure out why the BKP did not go into a dip around the same velocity as discussed here, $\nu = 0.17$[1]. As is seen in the figures listed, the BKP model does experience a dip, but it does it less frequent than the BK model. This difference points to the need for running more significant averages in order to reveal the true frequency and behaviour of the dips shown at mentioned slider velocities. Hence, running bigger batches for both models is listed as further work.

Looking at Section 5.2.3 comparing the BK and BKP model, there are notable differences in position and velocity, as well as the frequency and amplitude of the position from different blocks in the BK and BKP models. Furthermore, the phase and Fourier spectrum show very similar shapes, frequencies and amplitudes at $\nu = 0.11$, where the first dip is experienced. The similarity is apparent for both increasing and decreasing ν. As noted, there are differences for the BK and BKP model at this slider velocity with the BK model showing a more condense pattern. The BK model also shows fewer peaks in the Fourier spectrum, sharing the ones it has with the BKP model. These new peaks in the BKP model are probably a result of the pad, making the blocks oscillation dependent on more than its closest neighbours and friction. Higher velocities ν also show consistent differences between the BK and BKP model, but to a more substantial degree than shown at the dip. There are also apparent differences here between runs where the slider is increasing and decreasing in velocity. There is a possibility that these results are run specific, and it is hard to conclude specific features without analysing more significant averages. It can, however, tell that the pad has an impact on the system which is detectable. Further, the Fourier spectrum for the sample of blocks show each share the same frequencies with varying amplitude. Investigating these data closer for bigger batches and potentially more block could yield a better understanding.

The pad seems to have a very consistent effect of lowering the friction amplitude in the slider velocity interval of $\nu \in [0.25, 0.68]$ compared to the BK model. Figures 5.8 and 5.12 clearly show this difference between the two models. The blocks in the BK model has also shown a tendency of more often obtaining a positive position relative to where their pulling (top) spring is attached. This tendency is visible in the domain where the blocks are oscillating in a more chaotic pattern. The domain in question is also an interval where the pad is showing a high positional amplitude, as seen in Figure 5.4. Interestingly, it is the BK-pad with the decreasing slider velocity, which has the highest mean friction amplitude in the interval of $\nu \in [0.86, 1.06]$. This observation is interesting due to the pad position consistently has a low amplitude in this interval. This interval could be interesting to look into for further work, generating animations to try to see what is happening in the interval. The previous interval $\nu \in [0.25, 0.68]$, the pad is likely limiting the range of motion for the blocks causing a lower frictional amplitude. As for all the models, the general behaviour of the mean friction is quite similar except for the second dip. Further, all follow the general shape of the scaled friction law.

6.3 Animation

Another thing that is important to mention is how useful animation has been for studying and inspecting different parts of the system. Adding the option to view large parts of the system and how they behave over time is a useful asset to have, especially when looking at specific regions in, i.e. slider velocity. The system does require more fine-tuning and further development in order to increase the user-friendliness of it. It is worth noting that the animation is written in the same language used for creating the majority of the plots and figures, namely Python. Being written in this language making it very applicable for including more methods described in the Thesis into the animation, i.e. the Fourier spectrum. Implementing a live Fourier spectrum could reveal more on how the frequencies and amplitudes change and why.

A Fourier spectrum would give more insight into the animation of the outlier detailed in Section 5.4. The animation shows that the blocks oscillate in a synchronous pattern showing a periodic wave behaviour in the position of the blocks. This wave behaviour is present in the blocks until obtaining the slider velocity where the drop in friction amplitude occurs. The close to harmonious wave pattern is replaced by a more chaotic pattern consisting of multiple waves. The animation indicates that the outlier occurred when the blocks started to oscillate in a synchronous and repetitive pattern. There are similarities in this behaviour

and the one shown for dips in Section 5.4.1. In both situations, there is a clear and repetitive pattern in the blocks. The difference is that instead of increasing the pads or blocks averaged position magnitude, the pattern in dips has the opposite effect. The pad or blocks averaged position show an almost flat line during the dips. If more outliers are detected in possible future work, the difference between them and the dips could be material for further investigation.

Due to the current situation of video editing, it is also possible to combine runs with simple and often free tools such as the editor KdenLive (Section 4.3.2). KdenLive is used to compared different runs of the BK and BKP model where different runs of the respective models show different behaviour. In Figure 5.24, two animations of the BKP model where one of them enters the second dip is edited together. The animations clearly show that the blocks and pad display quite different behaviour in the two cases. The animation where the dip is present show a tidier and repeating positional change in the blocks with little movement in the pad. The other has a quite messy behaviour in the blocks which is reflected onto the pad. The indicates that the dip is likely due to the blocks getting into a synchronous and repetitive pattern. The frequency this happens in the different models tells something about the probability of it happening. These probabilities are hard to conclude from an average of 16 runs, but could be discovered with more runs used in the average. Running bigger batches for averages could be looked into in further work. More significant averages will also reveal the presence of different outliers.

An interesting aspect of both models friction amplitude paths as in Figure 5.8 is the fact that they seem to follow two underlying functions. One is decreasing, and the other function is increasing from the start and cross in $\nu = 0.05$. This shape might be an underlying behaviour in the models where the highest value of the two functions is taken. Such an underlying behaviour is not possible to conclude or debunk with the results in this Thesis and is listed in Section 6.5.

6.4 Conclusion

In this Thesis, the BK and BKP model was successfully implemented for running batch simulations. These batches proved useful for investigating and comparing the BK and BKP model. The comparison was made using step-wise changing slider velocity ν for the models, which was confirmed to be sufficiently close in behaviour to continuous change. Through phase plots of the pads position, it is clear that there is a connection between the pads range of motion and how stable its oscillation is, with higher range leading to less stability. Further, the pad is stable for a more significant velocity range when ν is decreasing compared to increasing.

Through the use of averages, the BK and BKP model show some apparent similarities and differences. They both share two dips in the mean friction and fiction amplitude at low slider velocity. It is shown that both models share the same general behaviour in these measurements for the first dip, for both increasing and decreasing ν. The second dip has a clear difference between the models. One difference is that the BK model shows a more significant decrease in the mean friction and friction amplitude. This difference is caused by the single BKP runs, which more seldom enters the second dip. This behaviour, together with the fact the BKP runs that enter the second dip with increasing ν, has a wider dip, show a clear difference from the pad. Further, the pad causes a lower friction amplitude in the interval of $\nu \in [0.25, 0.68]$ through limiting the range of motion for the blocks.

The Thesis also concludes that the animation of the model has been a great asset in gaining an understanding of the model. A large amount of output can be visually inspected efficiently while also providing information that is hard to inspect in still figures. New video editing tools make it simple to combine multiple animations to highlight some difference or compare them to each other. The project as a whole has also shown the value of using modern developing tools and mindsets, especially through the animation.

By continuing on the implementation and results presented in this Thesis, a clear understanding of the applicability of the BKP model to real-life problems such as brake squeal can be determined.

6.5 Further work

The further work sections are split up in three sections. The first section spells out ideas for further investigation. The second section is a list of smaller ideas and improvements concerning the simulations themselves and the analysis of the output. The last section concern technical improvements, such as ones that could improve the workflow running simulations and increase user-friendliness.

Ideas

- Run the same simulations as shown in the papers by A. Papangelo et al. presented in Section 2.4 and compare them to the BKP model. The comparison could link the literature better together and potentially give more insight into the viability of the BKP model.
- Use the implementation to study how the noise in a brake system can be controlled.
- Implement and inspect a more advanced friction law to map the BKP models applications in real life problems.

Simulation

- Run bigger batches to get a better picture of the general behaviour of the BK and BKP model.
- Look at the ending friction force for the BK and BKP model for the increasing slider velocity.
- Make animations for the BK and BKP model with decreasing slider velocity.
- Run averages to find or confirm the general frequencies of the pad and blocks in the BK and BKP model at different slider velocities.
- Investigate the possible underlying functions present in the friction amplitude of the BK and BKP model.
- Do step-wise runs with finer increases in slider velocity.
- Investigate further accelerating the slider to a threshold as an alternative to step-wise acceleration described in Section 5.1.2.

Technical

- Develop a standard for output to improve the system overall and reduce the time required to interpret it.
- Develop a GUI that fuse the different visualisation possibilities. The GUI should make it possible to interpret the system with little prior knowledge of the implementation.
- Make the animation interactive. It has been shown that the animation can support different visualisation of the output. With interaction, these could be interpreted more efficiently. This point is a continuation of the previous point.
- Develop a more sophisticated python wrapper for running and setting up the simulations.

Bibliography

[1] Hermann Nylund Ferre. The burridge-knopoff-pad model: A new model for studying noise generation & brake dynamics. February 2018.

[2] Bo NJ Persson. *Sliding friction: physical principles and applications*. Springer Science & Business Media, 2013.

[3] Ian Ridpath. Newton's laws of motion, 01 2012.

[4] Michael Marder. Terms of detachment. *Nature materials.*, 3(9):583–584, 2004.

[5] Bastiaan Antonius Hermanus Huisman et al. *Dynamical and structural self-organization: a study of friction, liquid-crystal nucleus growth, and supramolecular polymers through simple models*. PhD thesis, Universiteit van Amsterdam [Host], 2008.

[6] L. Knopoff and R. Burridge. Theoretical and model seismicity. *Geophysical Journal International*, 11(1):265, 1966.

[7] R Burridge and Leon Knopoff. Model and theoretical seismicity. *Bulletin of the seismological society of america*, 57(3):341–371, 1967.

[8] Ugo Galvanetto. Non-linear dynamics of multiple friction oscillators. *Computer methods in applied mechanics and engineering*, 178(3-4):291–306, 1999.

[9] Carlo Cantoni, Riccardo Cesarini, Giampiero Mastinu, Gianpiero Rocca, and Roberto Sicigliano. Brake comfort–a review. *Vehicle System Dynamics*, 47(8):901–947, 2009.

[10] A Papangelo, A Grolet, L Salles, N Hoffmann, and M Ciavarella. Snaking bifurcations in a self-excited oscillator chain with cyclic symmetry. *Communications in Nonlinear Science and Numerical Simulation*, 44:108–119, 2017.

[11] A Papangelo, N Hoffmann, A Grolet, M Stender, and M Ciavarella. Multiple spatially localized dynamical states in friction-excited oscillator chains. *Journal of Sound and Vibration*, 417:56–64, 2018.

[12] Endre Süli and David F Mayers. *An introduction to numerical analysis*. Cambridge university press, 2003.

[13] Richard P Feynman, Robert B Leighton, and Matthew Sands. The feynman lectures on physics; vol. i. *American Journal of Physics*, 33(9):750–752, 1965.

[14] H Strogatz Steven. *Nonlinear dynamics and chaos: with applications to physics, biology, chemistry, and engineering*. Westview press, 1994.

[15] Ronald E Walpole, Raymond H Myers, Sharon L Myers, and Keying Ye. *Probability and statistics for engineers and scientists*, volume 5. Macmillan New York, 1993.

[16] E Oran Brigham. *The fast Fourier transform and its applications*, volume 448 of *Prentice-Hall Signal Processing Series: Advanced monographs*. prentice Hall Englewood Cliffs, NJ, 1988.

[17] Travis E Oliphant. Python for scientific computing. *Computing in Science & Engineering*, 9(3):10–20, 2007.

[18] Yakov Shafranovich. Common format and mime type for comma-separated values (csv) files, 2005.

[19] F. James. A review of pseudorandom number generators. *Computer Physics Communications*, 60(3):329 – 344, 1990.

[20] Mark Pilgrim and Simon Willison. *Dive Into Python 3*, volume 2. Springer, 2009.

[21] John D Hunter. Matplotlib: A 2d graphics environment. *Computing in science & engineering*, 9(3):90–95, 2007.

[22] K. J. Millman and M. Aivazis. Python for scientists and engineers. *Computing in Science Engineering*, 13(2):9–12, March 2011.

[23] Oren Ben-Kiki, Clark Evans, and Brian Ingerson. Yaml ain't markup language (yaml^{TM}) version 1.2. *yaml. org, Tech. Rep*, pages 1,2, 2009.

[24] Tim Bray, Jean Paoli, C Michael Sperberg-McQueen, Eve Maler, and François Yergeau. Extensible markup language (xml). *World Wide Web Journal*, 2(4):27–66, 1997.

[25] Jesse Beder. yaml-cpp. https://github.com/jbeder/yaml-cpp, 2018.

[26] Kirill Simonov. Pyyaml, ver. 3.12. https://pyyaml.org/wiki/PyYAML, 2016.

[27] J Wood. Free and open source video editor for gnu/linux, mac os x and freebsd. kdenlive, 2002.

[28] Inkscape Team. Inkscape: A vector drawing tool, 2004.

[29] Linus Torvalds, J Hamano, et al. Git, 2005.

[30] Dimitri Van Heesch. Doxygen, 2004.

Appendices

Appendix A

YAML Appendix

This appendix includes yaml-files containing parameters and configurations for runs shown in the thesis.

A.1 Blocks: only stationary springs

```yaml
---
Parameters:
  dt : 0.005
  seed : 101
  num_events : 1
  N : 3
  max_time : 50
  slider_speed : 0.000
  increment : 0.000
  interval : 1000
  file_name : test_solutions
  progress_indicator : true
  m_F0 : 1
  m_alpha : 0.5
  m_sigma : 0.01
  m_mass_x : 100
  m_scale_mass : 1
  m_zeta : 0.0833
  m_k_P0 : 100
  m_scale_P : 1
  m_scale_C : 0.01
  m_t : 0.0
  m_v0 : 0.0001
  m_u_min : 0
  blocks : []
  start_speed_continuous: 0.0
  end_speed_continuous: 1.0
  save_interval_dt: 1
  threshold_speed: 0.1
Debug:
  # Debug variables. Should normally be false
  debug_no_friction :                true
  debug_no_neighbor_springs :        true
  debug_no_stationary_springs :      false
  debug_no_damper :                  false
  debug_no_min_speed :               false
  debug_no_pad :                     true
  debug_negative_initial_values :    true
  debug_only_negative_initial :      false
  debug_no_random_displacements :    false
  debug_special_phi :                false
  debug_pad_as_block :               false
  debug_stop_slider :                false
  debug_stick_blocks :               false
  debug_write_blocks :               false
  debug_only_write_friction :        false
  debug_continuous_slider_speed :    false
  debug_one_degree_freedom_mode :    false
...
```

A.2 Blocks: only neighbouring springs

```yaml
---
Parameters:
  dt : 0.005
  seed : 103
  num_events : 1
  N : 3
  max_time : 100
  slider_speed : 0.000
  increment : 0.000
  interval : 1000
  file_name : test_solutions
  progress_indicator : true
  m_F0 : 1
  m_alpha : 0.5
  m_sigma : 0.01
  m_mass_x : 100
  m_scale_mass : 1
  m_zeta : 0.0833
  m_k_P0 : 100
  m_scale_P : 1
  m_scale_C : 0.01
  m_t : 0.0
  m_v0 : 0.0001
  m_u_min : 0
  blocks : []
  start_speed_continuous: 0.0
  end_speed_continuous: 1.0
  save_interval_dt: 1
  threshold_speed: 0.1
Debug:
  # Debug variables. Should normally be false
  debug_no_friction :                true
  debug_no_neighbor_springs :        false
  debug_no_stationary_springs :      true
  debug_no_damper :                  false
  debug_no_min_speed :               false
  debug_no_pad :                     true
  debug_negative_initial_values :    true
  debug_only_negative_initial :      false
  debug_no_random_displacements :    false
  debug_special_phi :                false
  debug_pad_as_block :               false
  debug_stop_slider :                false
  debug_stick_blocks :               false
  debug_write_blocks :               false
  debug_only_write_friction :        false
  debug_continuous_slider_speed :    false
  debug_one_degree_freedom_mode :    false
...
```

A.3 Animation: BK decrease seed 106

```yaml
Debug:
  debug_continuous_slider_speed: false
  debug_negative_initial_values: true
  debug_no_damper: false
  debug_no_friction: false
  debug_no_min_speed: false
  debug_no_neighbor_springs: false
  debug_no_pad: true
  debug_no_random_displacements: false
  debug_no_stationary_springs: false
  debug_only_negative_initial: false
  debug_only_write_friction: false
  debug_pad_as_block: false
  debug_special_phi: false
  debug_stick_blocks: false
  debug_stop_slider: false
  debug_write_blocks: false
Parameters:
  N: 100
  blocks: []
  dt: 0.005
  end_speed_continuous: 1.0
  file_name: test_solutions
  increment: -0.01
  interval: 2000
  m_F0: 1
  m_alpha: 0.5
  m_k_P0: 100
  m_mass_x: 100
  m_scale_C: 0.01
  m_scale_P: 1
  m_scale_mass: 1
  m_sigma: 0.01
  m_t: 0.0
  m_u_min: 0
  m_v0: 0.0001
  m_zeta: 0.0833
  max_time: 300000
  num_events: 1
  progress_indicator: true
  save_interval_dt: 100
  seed: 106
  slider_speed: 1.49
  start_speed_continuous: 0.0
  threshold_speed: 0.1
```

A.4 Animation: BK increase seed 101

```yaml
Debug:
  debug_continuous_slider_speed: false
  debug_negative_initial_values: true
  debug_no_damper: false
  debug_no_friction: false
  debug_no_min_speed: false
  debug_no_neighbor_springs: false
  debug_no_pad: true
  debug_no_random_displacements: false
  debug_no_stationary_springs: false
  debug_only_negative_initial: false
  debug_only_write_friction: false
  debug_pad_as_block: false
  debug_special_phi: false
  debug_stick_blocks: false
  debug_stop_slider: false
  debug_write_blocks: false
Parameters:
  N: 100
  blocks:
  - 1
  - 2
  - 3
  dt: 0.005
  end_speed_continuous: 1.0
  file_name: test_solutions
  increment: 0.01
  interval: 2000
  m_F0: 1
  m_alpha: 0.5
  m_k_P0: 100
  m_mass_x: 100
  m_scale_C: 0.01
  m_scale_P: 1
  m_scale_mass: 1
  m_sigma: 0.01
  m_t: 0.0
  m_u_min: 0
  m_v0: 0.0001
  m_zeta: 0.0833
  max_time: 300000
  num_events: 1
  progress_indicator: true
  save_interval_dt: 100
  seed: 101
  slider_speed: 0.0
  start_speed_continuous: 0.0
  threshold_speed: 0.1
```

A.5 Animation: BKP decrease seed 116

```yaml
Debug:
  debug_continuous_slider_speed: false
  debug_negative_initial_values: true
  debug_no_damper: false
  debug_no_friction: false
  debug_no_min_speed: false
  debug_no_neighbor_springs: false
  debug_no_pad: false
  debug_no_random_displacements: false
  debug_no_stationary_springs: false
  debug_only_negative_initial: false
  debug_only_write_friction: false
  debug_pad_as_block: false
  debug_special_phi: false
  debug_stick_blocks: false
  debug_stop_slider: false
  debug_write_blocks: false
Parameters:
  N: 100
  blocks: []
  dt: 0.005
  end_speed_continuous: 1.0
  file_name: test_solutions
  increment: -0.01
  interval: 2000
  m_F0: 1
  m_alpha: 0.5
  m_k_P0: 100
  m_mass_x: 100
  m_scale_C: 0.01
  m_scale_P: 1
  m_scale_mass: 1
  m_sigma: 0.01
  m_t: 0.0
  m_u_min: 0
  m_v0: 0.0001
  m_zeta: 0.0833
  max_time: 300000
  num_events: 1
  progress_indicator: true
  save_interval_dt: 100
  seed: 116
  slider_speed: 1.49
  start_speed_continuous: 0.0
  threshold_speed: 0.1
```

A.6 Animation: BKP insrease seed 116

```yaml
Debug:
  debug_continuous_slider_speed: false
  debug_negative_initial_values: true
  debug_no_damper: false
  debug_no_friction: false
  debug_no_min_speed: false
  debug_no_neighbor_springs: false
  debug_no_pad: false
  debug_no_random_displacements: false
  debug_no_stationary_springs: false
  debug_only_negative_initial: false
  debug_only_write_friction: false
  debug_pad_as_block: false
  debug_special_phi: false
  debug_stick_blocks: false
  debug_stop_slider: false
  debug_write_blocks: false
Parameters:
  N: 100
  blocks: []
  dt: 0.005
  end_speed_continuous: 1.0
  file_name: test_solutions
  increment: 0.01
  interval: 2000
  m_F0: 1
  m_alpha: 0.5
  m_k_P0: 100
  m_mass_x: 100
  m_scale_C: 0.01
  m_scale_P: 1
  m_scale_mass: 1
  m_sigma: 0.01
  m_t: 0.0
  m_u_min: 0
  m_v0: 0.0001
  m_zeta: 0.0833
  max_time: 300000
  num_events: 1
  progress_indicator: true
  save_interval_dt: 100
  seed: 116
  slider_speed: 0.0
  start_speed_continuous: 0.0
  threshold_speed: 0.1
```

Appendix B

Animation and Code Appendix

B.1 Web-hosted videos

Due to it being difficult to get a view of the animations in a PDF-format, a GitHub pages hosting some of the videos is made. Under is a link together with a small guide of how to use the page.

The page itself is written in markdown and contains links to different pages showing specific videos mentioned in the thesis among with some explanation as to how they where made. To get back to the main page the title *Simulating Brake Noise by Friction (Home)* can be clicked or navigation arrows used.

The page also host the documentation for the code together with the source files. As of this moment a button or link for going back to the main page is not implemented. To get back to the main page, use the navigation arrows or change the URL back to the one given below.

Link to the GitHub pages page is given by the following link.

https://izome.github.io/burridge-knopoff-pad-karsten/

B.2 Code

This Thesis is submitted together with a zip-file. The zip contains a *readme* explaining its content. Further, it contains documentation of the code used in the project created in HTML by Doxygen[30]. In addition to the documentation, it contains the plain code for the simulation. This code is also documented in the Doxygen documentation, which contain the source files for both C++ and Python code.

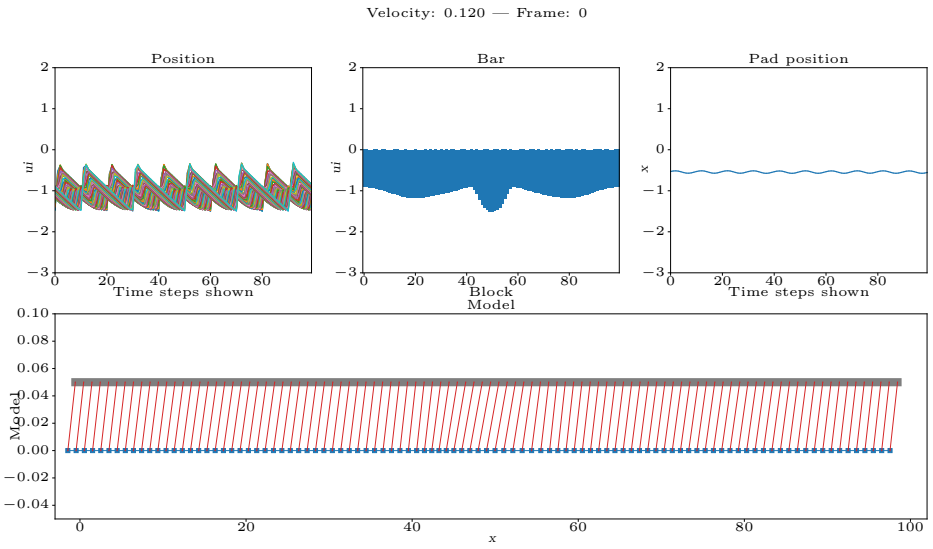


Figure B.1: Animation interface showing frame 0 of an increasing run of the BKP model. The animation frame shows when the run in the first dip seen in Figure 5.13.

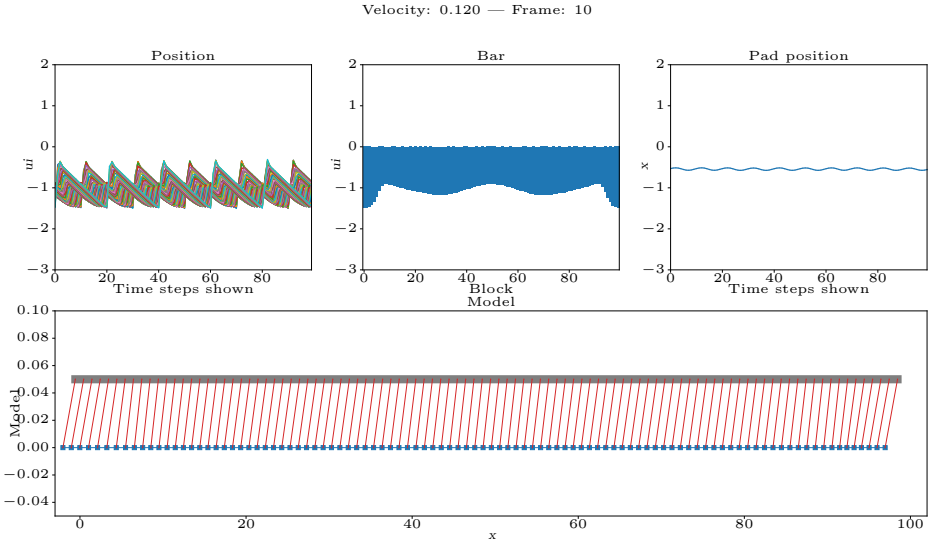


Figure B.2: Animation interface showing frame 10 of the same run as in Figure B.1.

Appendix C

Supporting Figures Appendix

C.1 Implementation

This appendix contains some additional plots that are supporting, but not essential of the ones found in the Thesis.

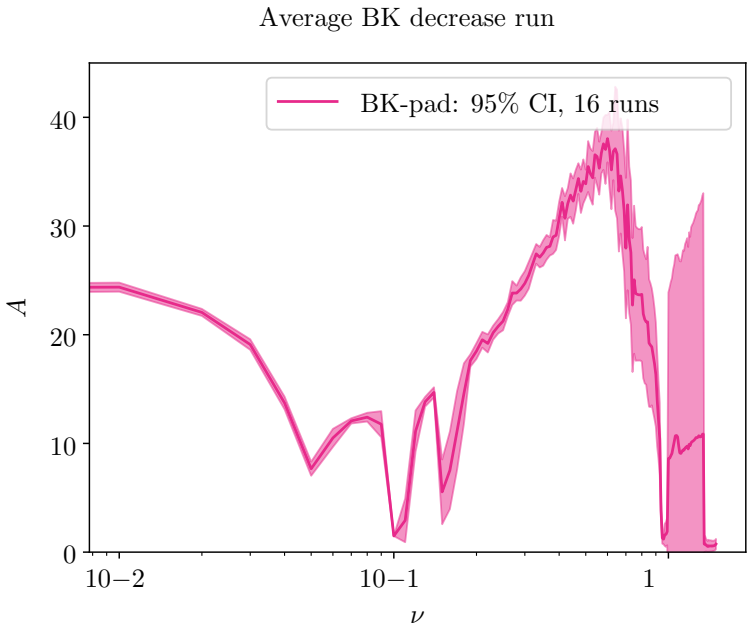


Figure C.1: Average with 95% confidence interval of the friction amplitude runs in in Figure 3.3 plotted with log-scale on the ν-axis.

C.2 Fourier heat map

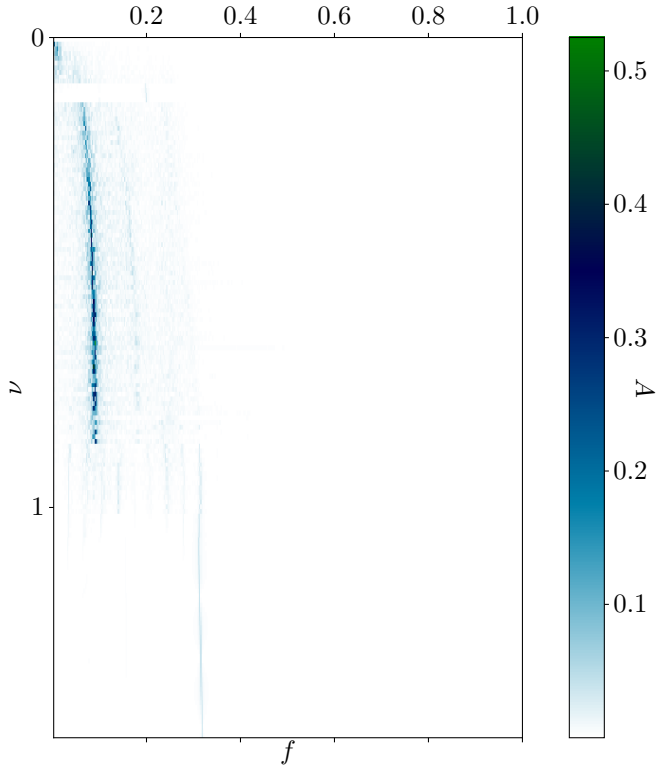


Figure C.2: Fourier spectrum heat map for the BKP model with increasing slider velocity. Seed value is 112. The amplitude is give by the heat bar on the right. Some lines are visible showing which frequencies show the highest amplitude at different velocities. It is also possible to see where multiple frequencies are present.

C.3 Fourier spectrum

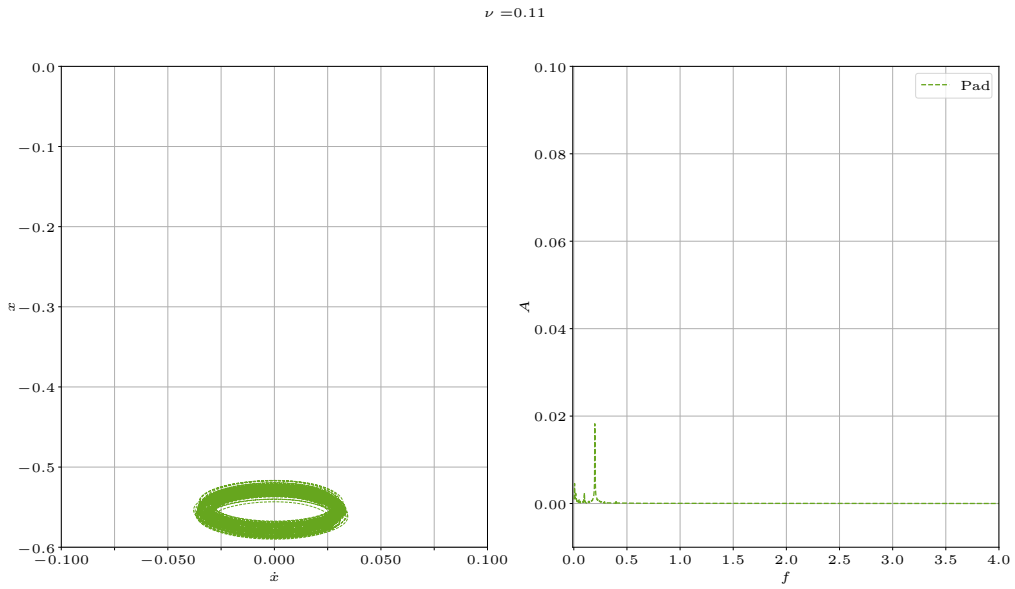


Figure C.3: Showing the phase plot to the left and Fourier spectrum to the right of the pad for the BKP model with increasing slider velocity $\nu = 0.11$. Note the axis are different that those of Figures 5.16 and 5.18 in order to see the values of the two plots. Seed value is 112.

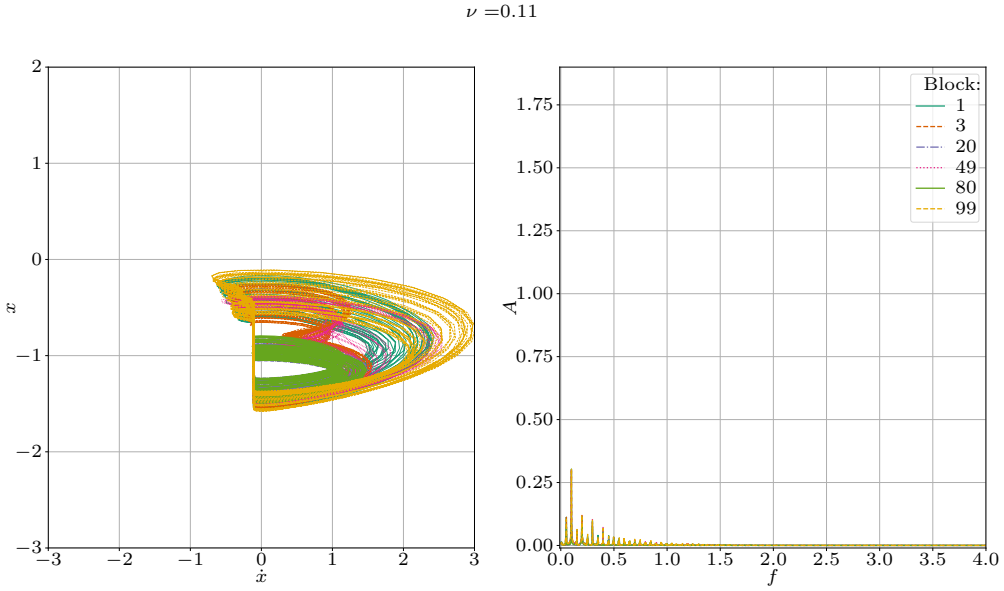


Figure C.4: Showing the phase plot to the left and Fourier spectrum to the right of six different blocks for the BKP model with increasing slider velocity $\nu = 0.11$. Seed value is 112.

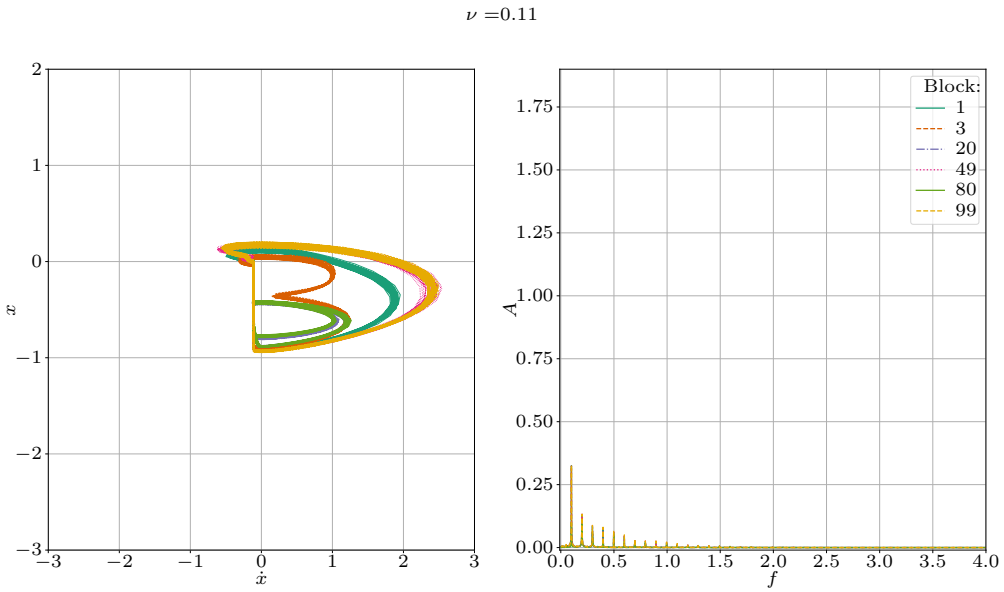
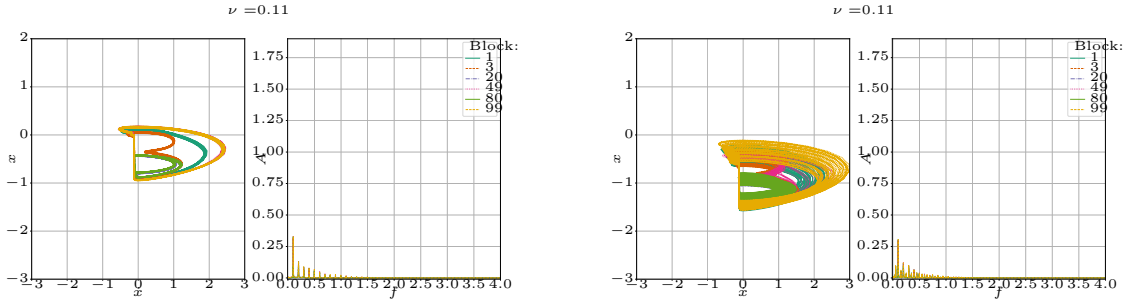


Figure C.5: Showing the phase plot to the left and Fourier spectrum to the right of six different blocks for the BK model with increasing slider velocity $\nu = 0.11$. Seed value is 113.



(a) BK decreasing (b) BKP decreasing

Figure C.6: Showing the phase plot to the left and Fourier spectrum to the right of six different blocks for the BK and BKP model with decreasing slider velocity $\nu = 0.11$.

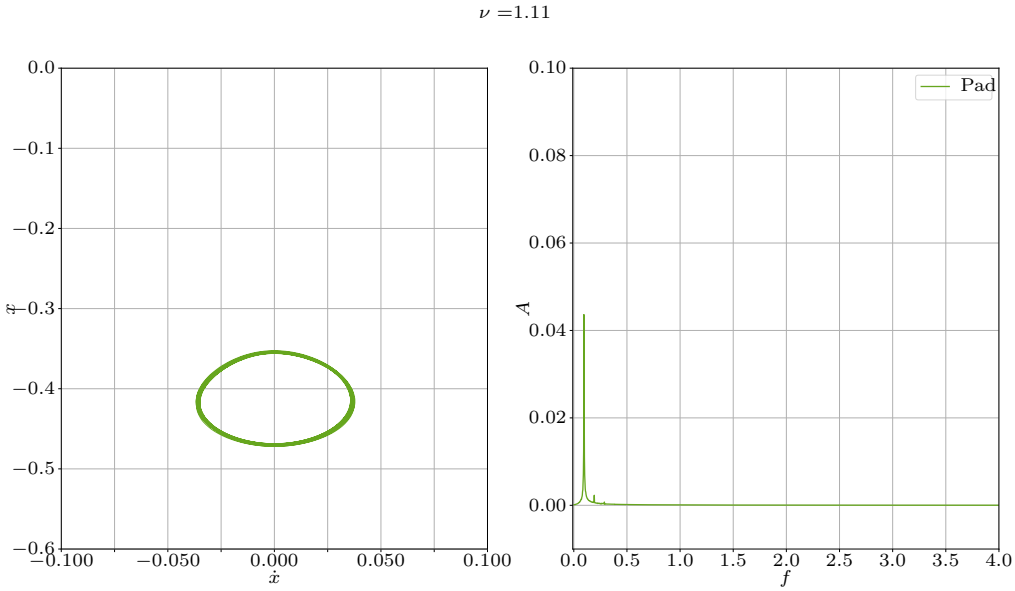
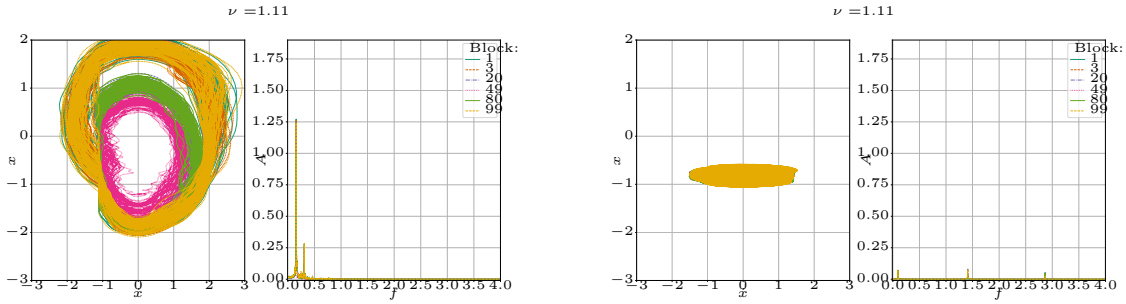


Figure C.7: Showing the phase plot to the left and Fourier spectrum to the right of the pad for the BKP model with decreasing slider velocity $\nu = 1.11$.



(a) BK decreasing (b) BKP decreasing

Figure C.8: Showing the phase plot to the left and Fourier spectrum to the right of six different blocks for the BK and BKP model with decreasing slider velocity $\nu = 1.11$.

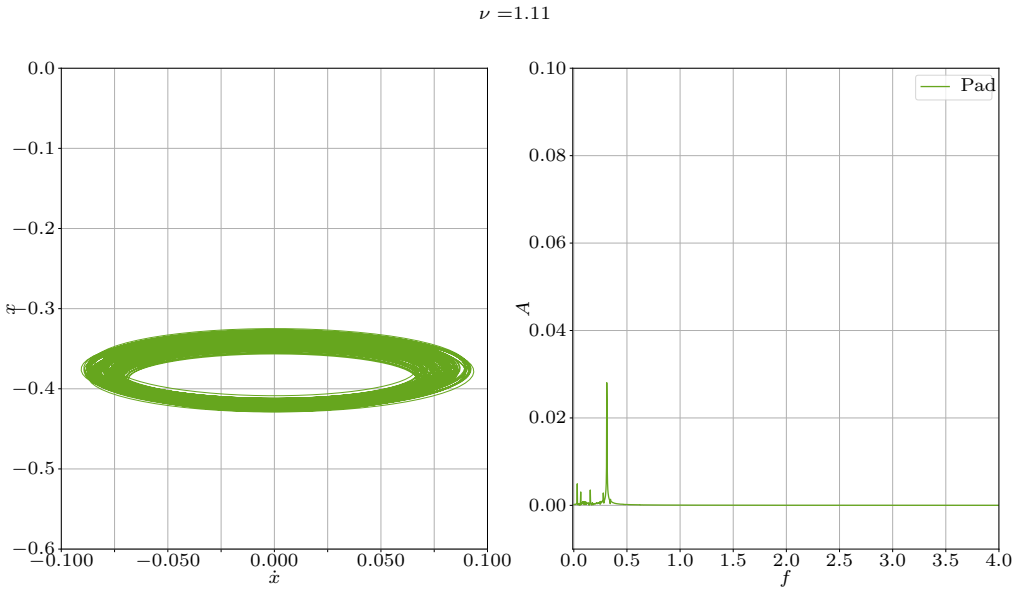
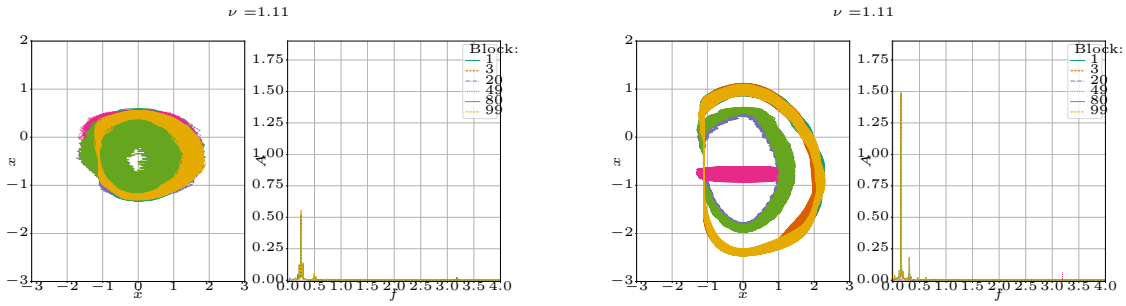


Figure C.9: Showing the phase plot to the left and Fourier spectrum to the right of the pad for the BKP model with increasing slider velocity $\nu = 1.11$.



(a) BK increasing

(b) BKP increasing

Figure C.10: Showing the phase plot to the left and Fourier spectrum to the right of six different blocks for the BK and BKP model with increasing slider velocity $\nu = 1.11$.

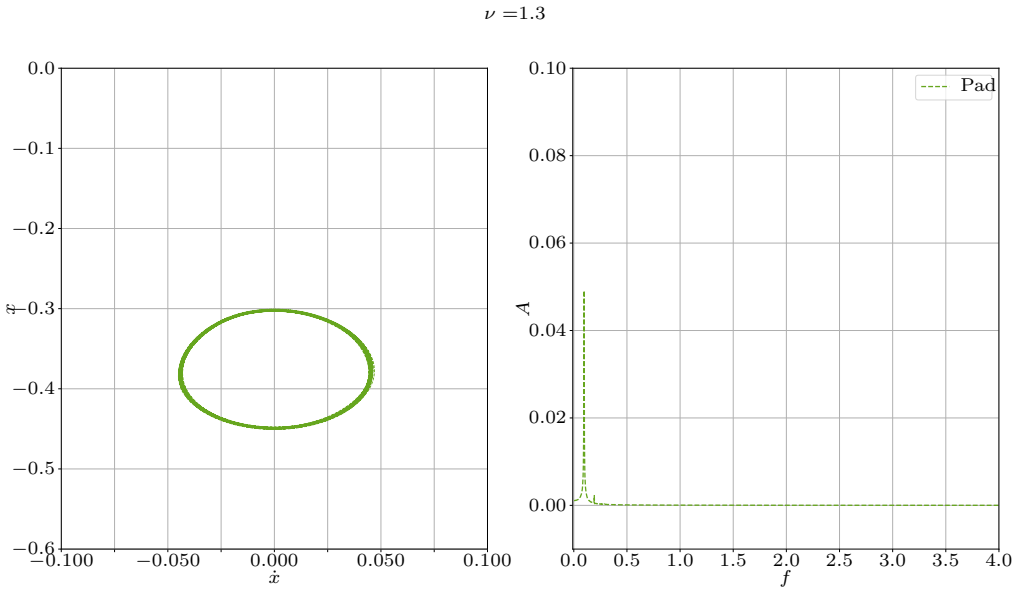
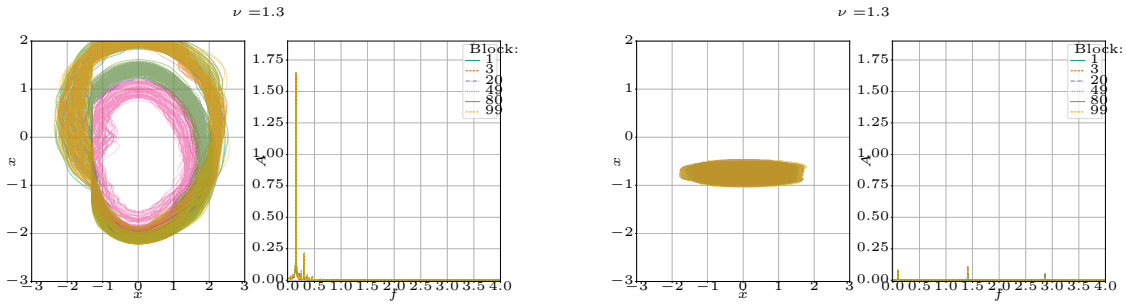


Figure C.11: Showing the phase plot to the left and Fourier spectrum to the right of the pad for the BKP model with decreasing slider velocity $\nu = 1.30$.



(a) BK decreasing

(b) BKP decreasing

Figure C.12: Showing the phase plot to the left and Fourier spectrum to the right of six different blocks for the BK and BKP model with decreasing slider velocity $\nu = 1.30$.

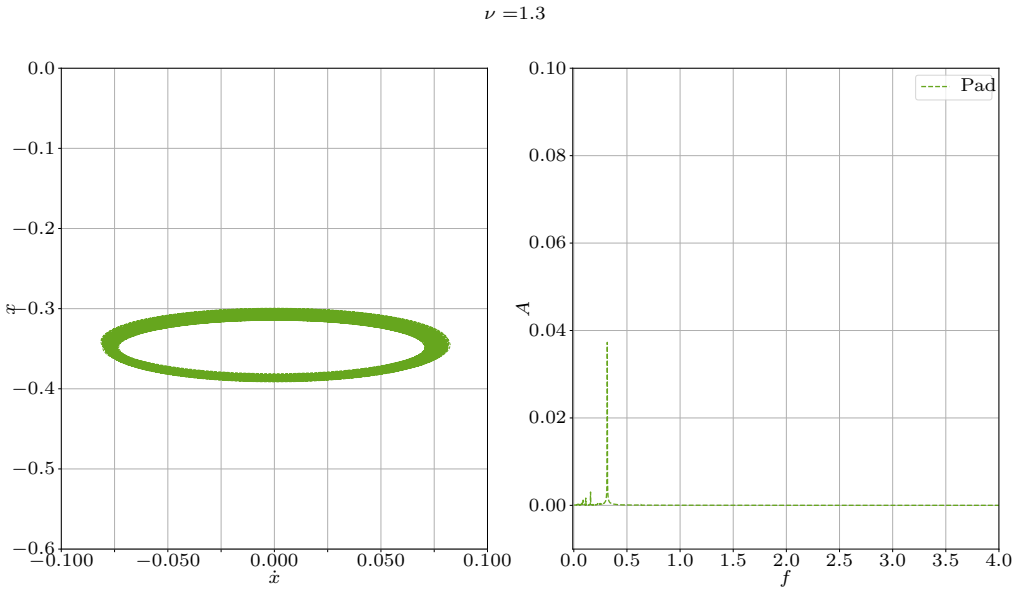
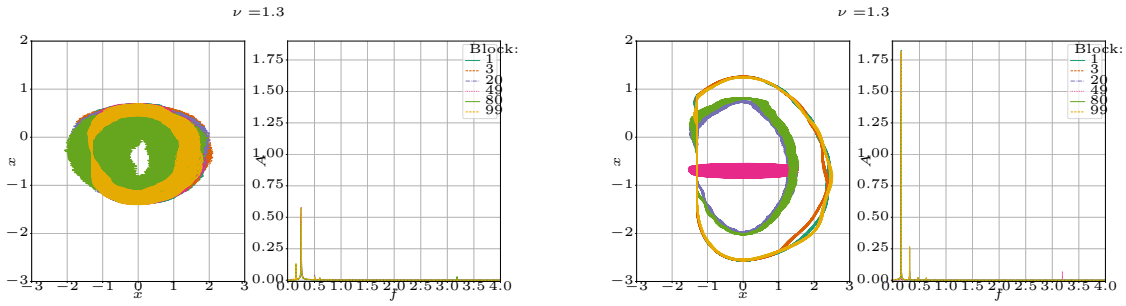


Figure C.13: Showing the phase plot to the left and Fourier spectrum to the right of the pad for the BKP model with increasing slider velocity $\nu = 1.30$.



(a) BK increasing

(b) BKP increasing

Figure C.14: Showing the phase plot to the left and Fourier spectrum to the right of six different blocks for the BK and BKP model with increasing slider velocity $\nu = 1.30$.

C.4 Comparing BK and BKP

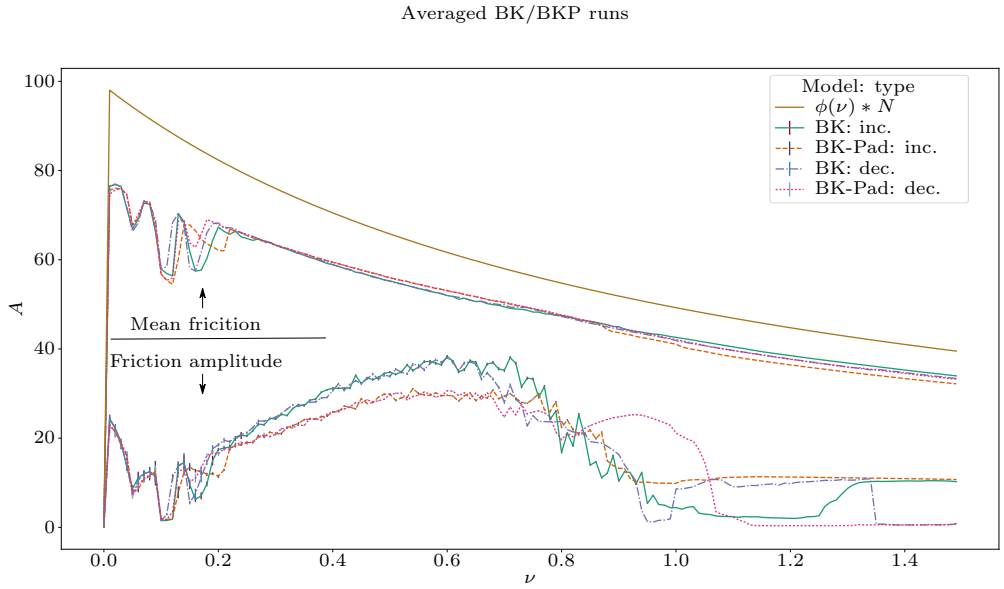


Figure C.15: Mean and error for the friction amplitude of increasing and decreasing simulations of BK and BKP model. Each mean and error is produced from 16 runs with different initial position and velocity for the blocks and is represented by the lower lines. Each mean shown in the lower line has a corresponding mean of mean friction line among the upper lines with the same colour and line style. The topmost line is the friction law shown in Figure 3.2 scaled with the number of blocks for further comparison. Non-logarithmic version of Figure 5.12

C.4.1 BK decreasing

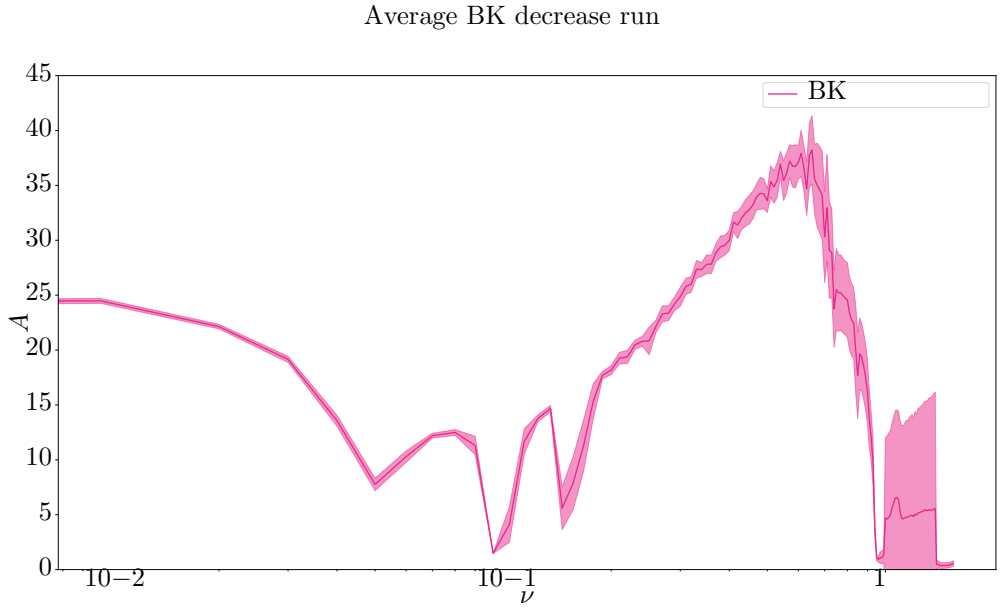


Figure C.16: Average friction amplitude with 95% confidence interval based on 32 runs of the decreasing BK model plotted with log-scale on the ν-axis.

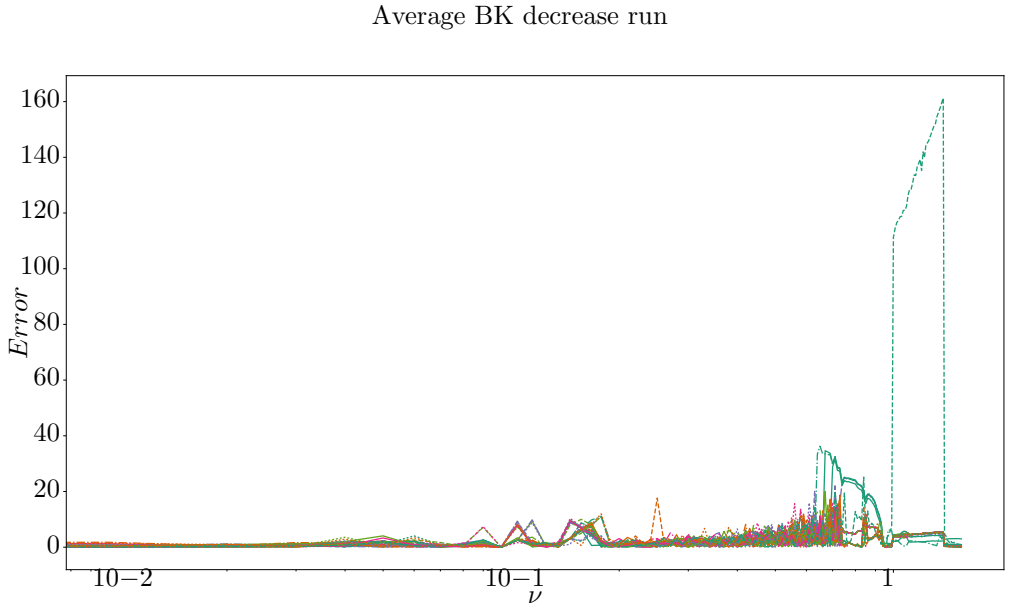
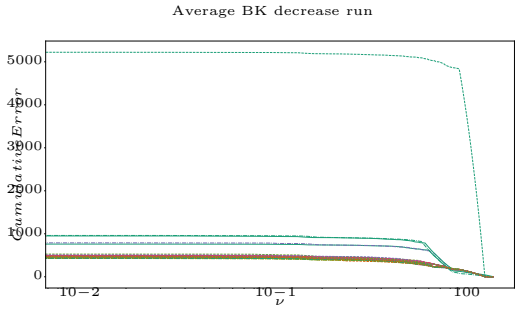
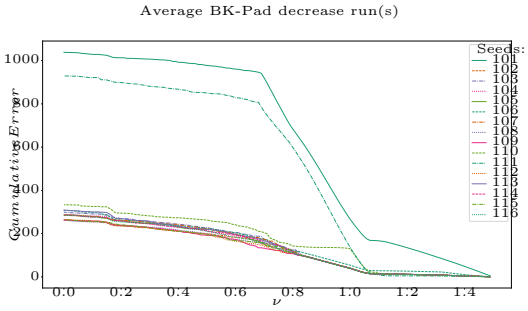


Figure C.17: Error for each run at different slider velocity



(a) 32 runs of the BK model plotted with log-scale on the ν-axis.



(b) 16 runs of the BKP model.

Figure C.18: Cumulative error following the slider velocity. The slider is decreasing in velocity.

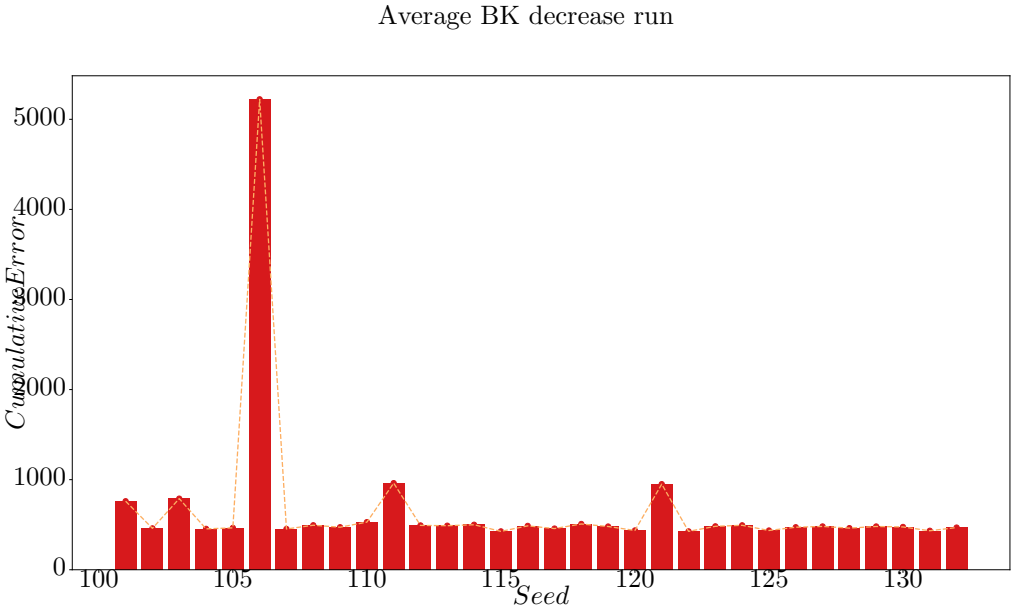


Figure C.19: Total error for each run in order of increasing seed used in the runs.

C.5 BK increasing

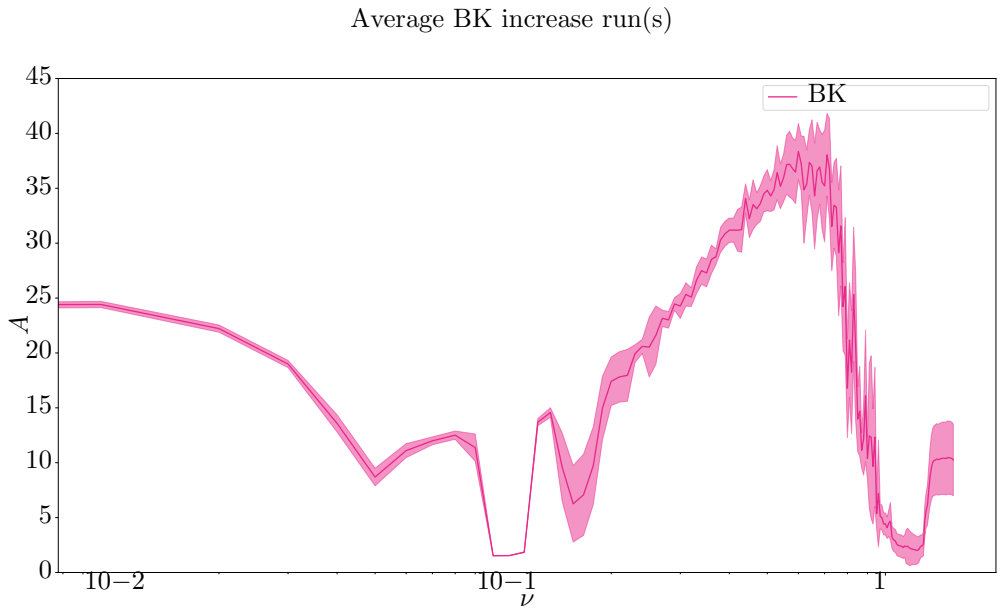


Figure C.20: Average friction amplitude with 95% confidence interval based on 16 runs of the increasing BK model plotted with log-scale on the ν-axis.

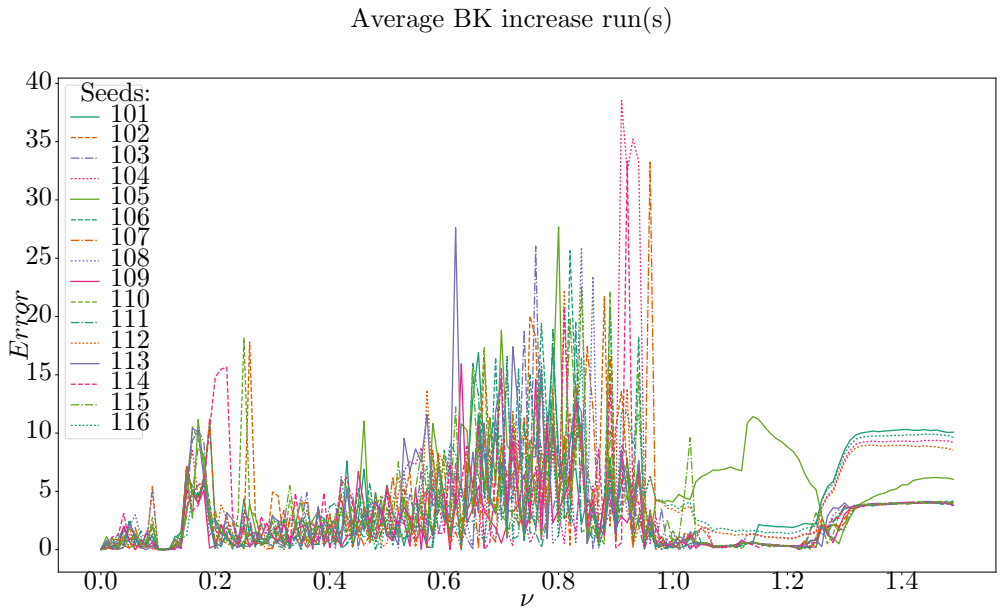


Figure C.21: Error for each run at different slider velocity for increasing BK runs

Average BK increase run(s)

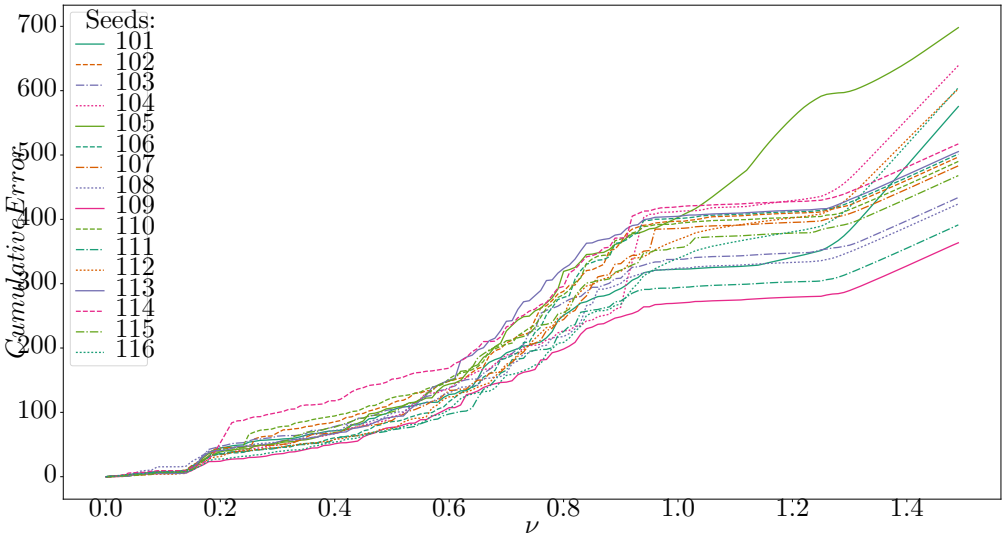


Figure C.22: Cumulative error following the slider velocity of increasing BK runs.

Average BK increase run(s)

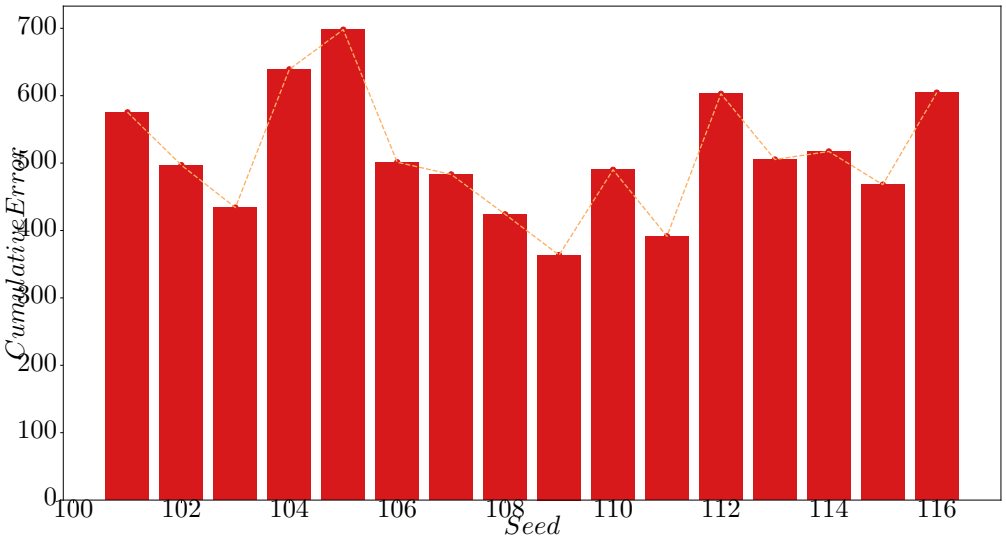


Figure C.23: Total error for each run in order for increasing seed used in the runs.

C.5.1 BKP decreasing

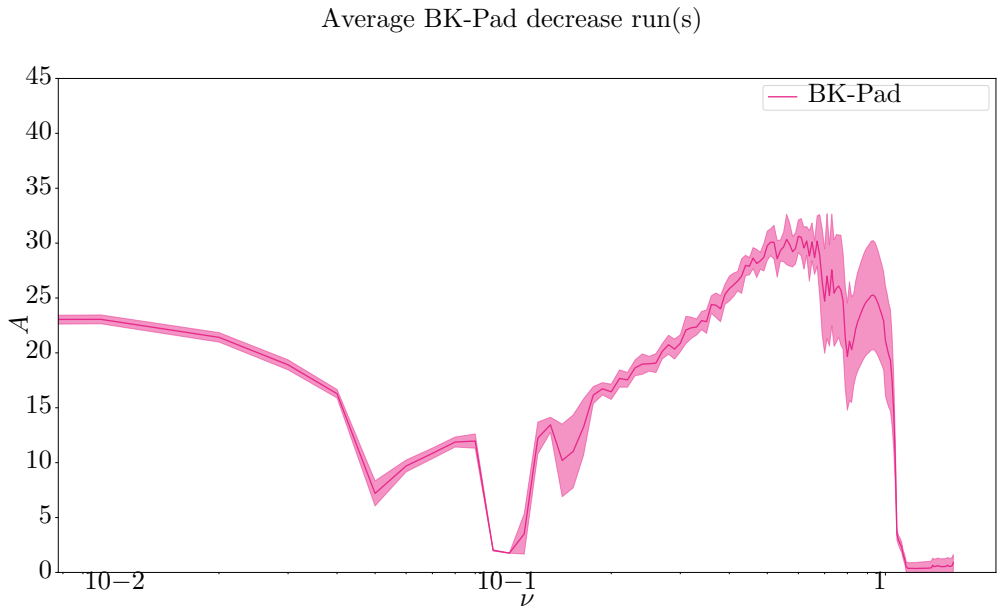


Figure C.24: Average friction amplitude with 95% confidence interval based on 316 runs of the decreasing BKP model plotted with log-scale on the ν-axis.

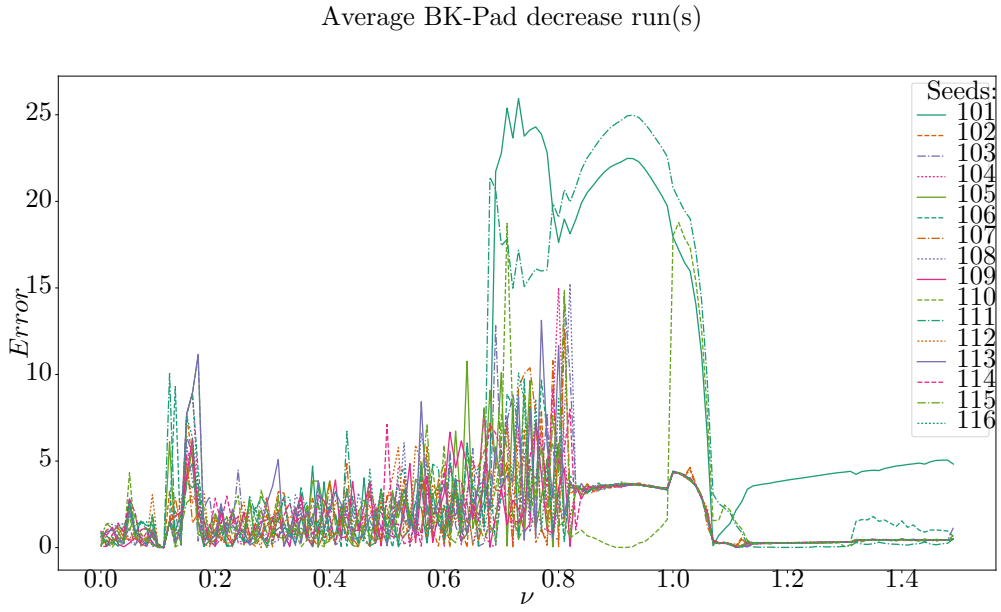


Figure C.25: Error for each run at different slider velocities for decreasing BKP runs.

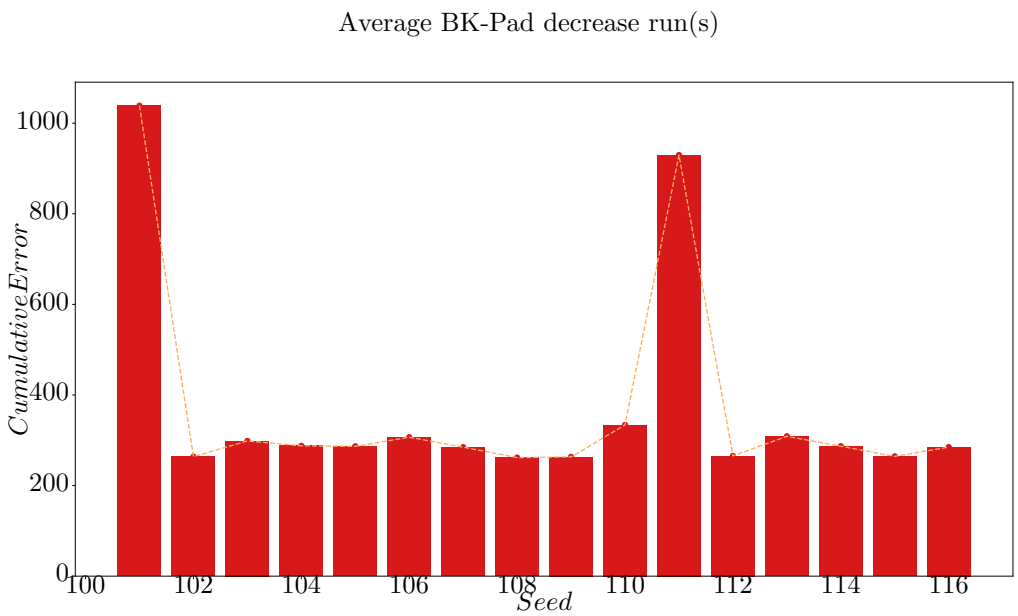


Figure C.26: Total error for each run in order of increasing seed used in the runs for decreasing BK runs.

C.5.2 BKP increasing

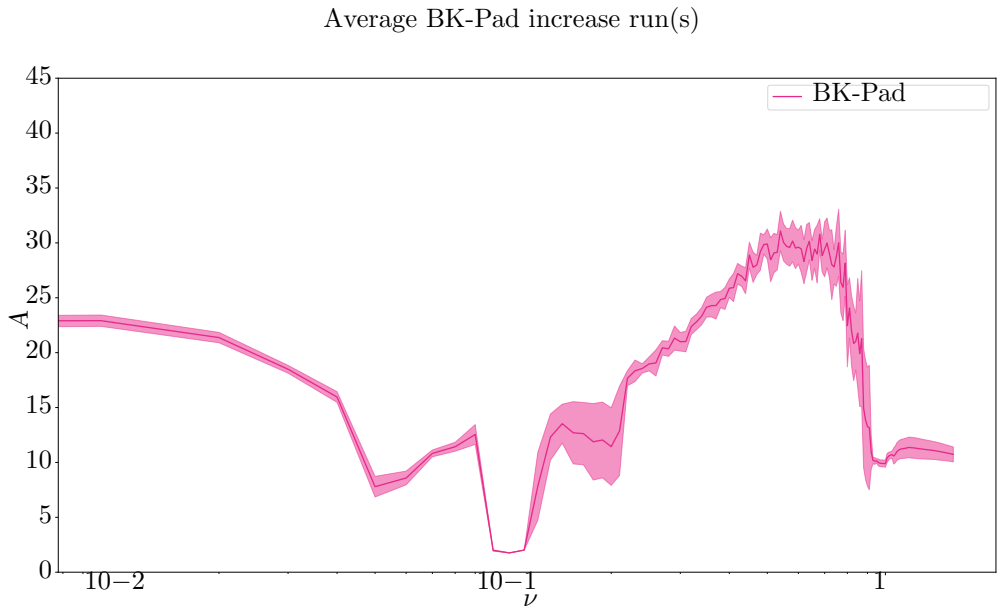


Figure C.27: Average friction amplitude with 95% confidence interval based on 16 runs of the increasing BKP model plotted with log-scale on the ν-axis.

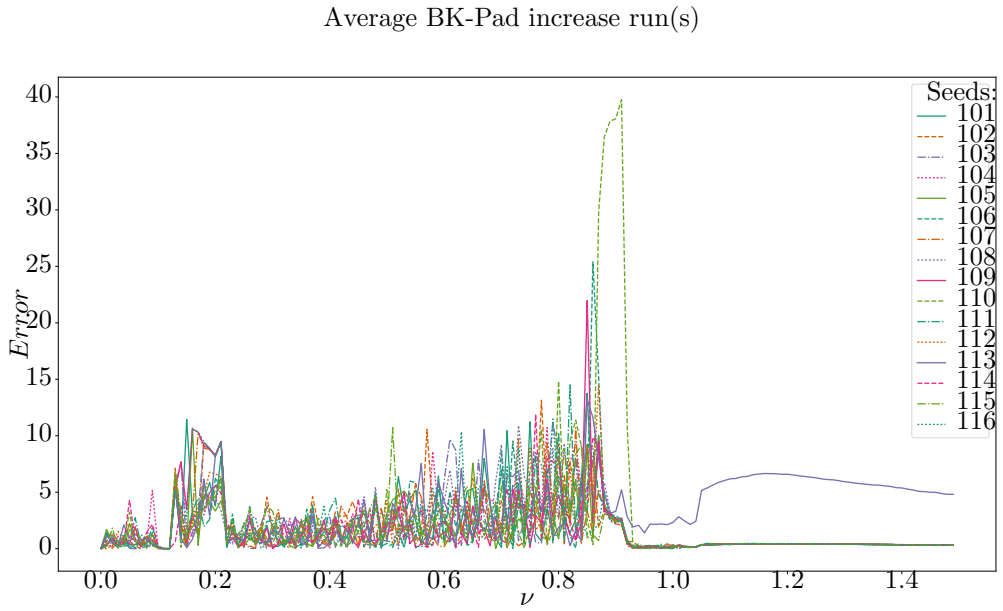


Figure C.28: Error for each run at different slider velocity for increasing BKP runs.

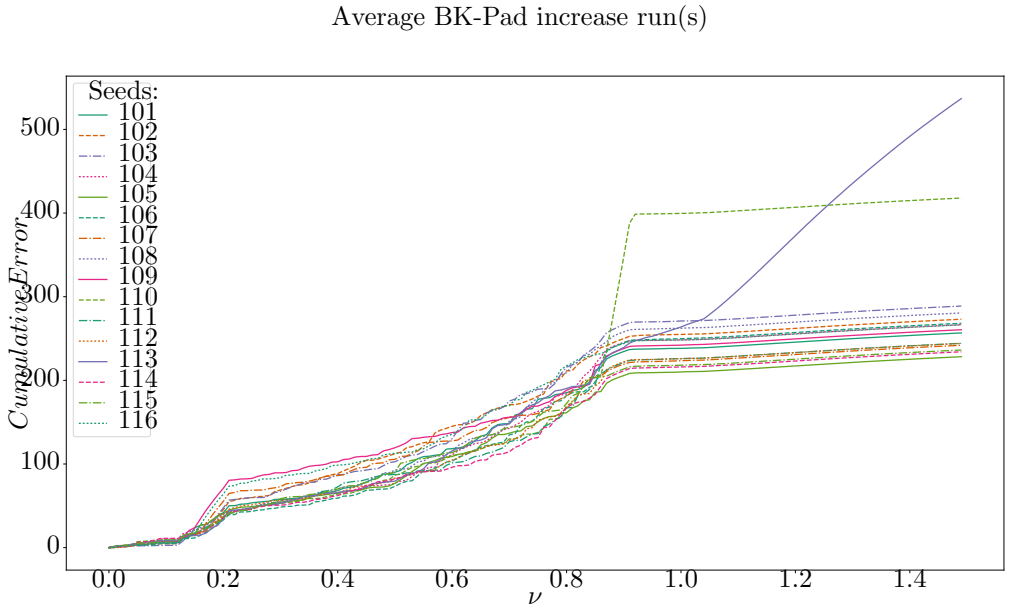


Figure C.29: Cumulative error following the slider velocity for increasing BKP runs.

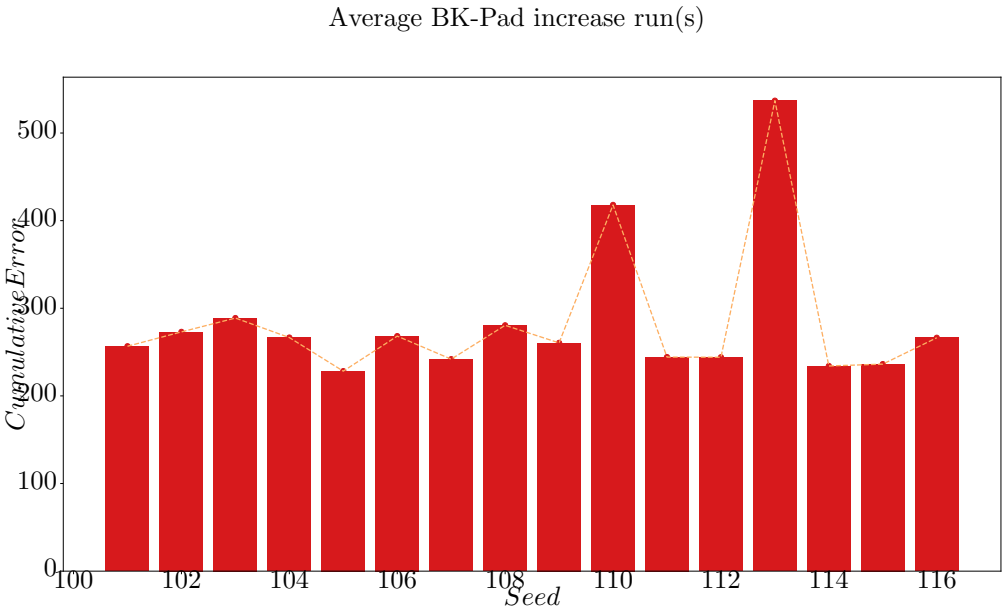
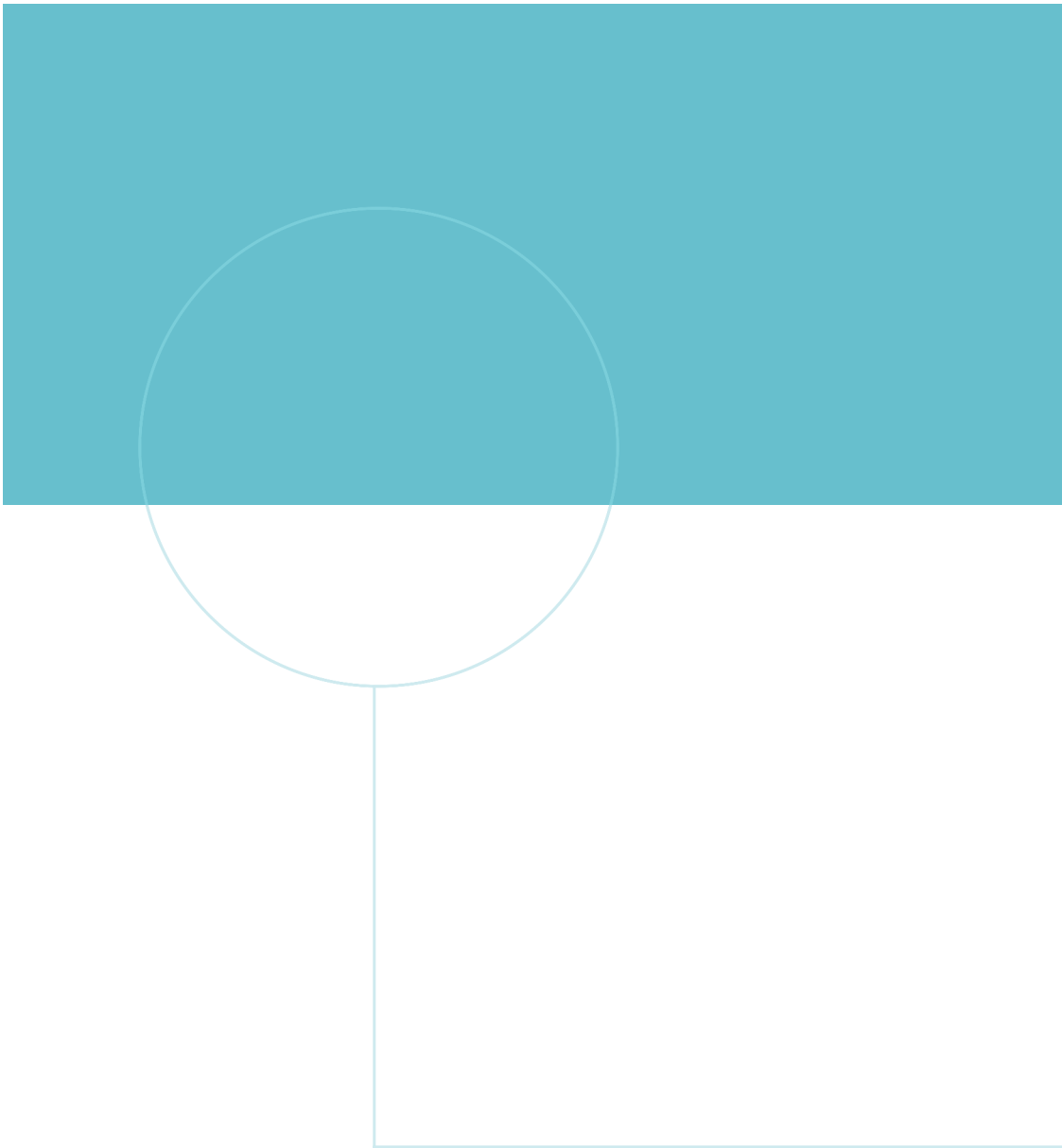


Figure C.30: Total error for each run in order of increasing seed used in the runs.



NTNU

Kunnskap for en bedre verden