TALLINNA TEHNIKAÜLIKOOL
LOODUSTEADUSKOND
KÜBERNEETIKA INSTITUUT

TURBULENTSE SEGUNEMISE MODELLEERIMINE ÜHEDIMENSIONAALSE MUDELI ABIL

Bakalaureusetöö

Üliõpilane: Karl Kukk
Üliõpilaskood: 232832YAFB
Juhendaja: Jaan Kalda, Küberneetika Instituut, täisprofessor tenuuris
Tallinna Tehnikaülikool
Õppekava: Rakendusfüüsika

Tallinn 2026

TALLINN UNIVERSITY OF TECHNOLOGY
SCHOOL OF SCIENCE
DEPARTMENT OF CYBERNETICS

MODELLING TURBULENT MIXING USING A ONE-DIMENSIONAL MODEL

Bachelor's thesis

Student: Karl Kukk
Student code: 232832YAFB
Supervisor: Jaan Kalda, Department of Cybernetics, Tenured Full Professor
Tallinn University of Technology
Study program: Applied Physics

Tallinn 2026

Declaration

Hereby I declare that I have compiled the paper independently and all works, important standpoints and data by other authors have been properly referenced and the same paper has not been previously been presented for grading.

Author: Karl Kukk .....

(signature)

Date: .....

The paper conforms to requirements in force.

Supervisor: Jaan Kalda .....

Date: .....

(signature)

Table of contents

Abstract

Turbulence remains one of the last major unsolved problems in classical physics. A key open question concerns the high-order behaviour of passive scalar structure function scaling exponents ζ_p and whether they grow logarithmically without bound or saturate at a finite value. The one-dimensional Baker's map model of Kalda and Morozenko [1] predicts unbounded logarithmic growth according to $\zeta_p = \frac{2}{3}\log_3(p+1)$, but was previously validated only up to $p \approx 8$, a range in which the two competing predictions are nearly indistinguishable.

This thesis revisits the model using modern computing resources, collecting 86 GB of simulation data across 20 independent parallel chains. The decisive test is the saturation ceiling: a finite asymptote $\zeta_\infty \approx 1.4$, the value implied by mature-front arguments, requires $\zeta_p < 1.4$ at every finite order, yet the reliable exponents reach $\zeta_{12} = 1.54 \pm 0.02$, multiple standard errors above this ceiling, while also continuing to rise.

The model does not saturate at $\zeta_\infty \approx 1.4$. Given its validated agreement with experimental data at low p, this argues against a finite ceiling in that range for real scalar turbulence. In the same reliable range, the measured exponents track the logarithmic prediction

$$\zeta_p = \frac{2}{3}\log_3(p+1)$$

to within 3.6% across $p \leq 10$, with the direct power-law fit and global Extended Self-Similarity in agreement, providing a cleaner cross-validated confirmation of the model than was previously available.

At $p = 14$–16, the results remain consistent with logarithmic growth but with weakening statistical reliability, beyond $p = 16$, the moment estimator does not converge for the present data volume and ζ_p is not quantitatively resolvable.

Conclusively describing the high-p behaviour will require a larger array ($N \gg 40960$) with the mean gradient removed prior to computing structure functions.

Annotatsioon

Turbulentsus on siiani üks viimaseid lahendamata probleeme klassikalises füüsikas. Üks avatud küsimustest puudutab voolu poolt kantava passiivse skalaari kontsentratsiooni kõrget järku struktuurifunktsioonide eksponente ζ_p ja kas need väärtused kasvavad logaritmiliselt piiramatult või koonduvad lõplikule väärtusele. Ühedimensionaalne Kalda ja Morozenko *Baker's map* mudel [1] ennustab logaritmilist kasvu

$$\zeta_p = \frac{2}{3}\log_3(p+1),$$

kuid on varasemalt kontrollitud vaid kuni $p \approx 8$, vahemikus, kus kaks konkureerivat ennustust on praktiliselt üksteisest eristamatud.

See töö naaseb antud mudeli juurde kasutades tänapäevaseid arvutusvõimsusi. Koguti 86 GB simulatsiooniandmeid üle 20 iseseisva paralleelse ahela. Mudel ei küllasu väärtuseni $\zeta_\infty \approx 1{,}4$: usaldusväärsed eksponendid jõuavad väärtuseni $\zeta_{12} = 1{,}54 \pm 0{,}02$, ületades selle piiri mitme standardhälbe võrra, ning jätkavad kasvamist. Arvestades mudeli kooskõla eksperimentaalsete andmetega väikeste p väärtuste juures, viitavad need tulemused sellele, et lõplik küllastuslävi reaalse turbulentsi jaoks pakutud vahemikus ei ole tõenäoline. Samas vahemikus järgivad mõõdetud eksponendid logaritmilist ennustust 3,6% täpsusega kuni $p \leq 10$, kusjuures astmeseaduse andmepunktide lähendamine ja globaalne *Extended Self-Similarity* on omavahel kooskõlas, pakkudes varasemast selgemat ristvalideeritud kinnitust Kalda ja Morozenko teooriale.

Väärtustel $p = 14$–16 jäävad tulemused logaritmilise kasvuga kooskõlla, kuid nende statistiline usaldusväärsus väheneb. Kõrgemate järkude $p > 16$ korral muutub statistiline koonduvus antud andmemahu juures ebausaldusväärseks ning ζ_p väärtusi ei saa kvantitatiivselt usaldada.

Kõrge p käitumise lõplikuks selgitamiseks on vajalik oluliselt suurema massiivi kasutamine ($N \gg 40960$) koos enne struktuurifunktsioonide arvutamist läbi viidava keskmise gradiendi eemaldamisega.

Abbreviations and terms

GB	abbreviation for gigabyte
CPU	abbreviation for central processing unit
ESS	abbreviation for Extended Self-Similarity
PDF	abbreviation for probability density function

Introduction

After years of research, turbulence remains the last major unsolved problem in classical physics. A better understanding of this problem could contribute to advancements in studying the atmospheric dispersion of pollutants or pollen, where passive scalars are carried by turbulent flows.

While fully developed turbulence is intermittent and extreme events occur far more often than what would be expected from Gaussian statistics, passive scalar turbulence provides a cleaner framework for studying intermittency, exhibiting stronger intermittency and simpler equations.

The structure functions of turbulent flow at high orders are yet to be fully understood. Two competing versions exist, logarithmic and saturation at high orders. These two are practically very hard to distinguish because of the extremely slow characteristic of logarithmic growth.

The one-dimensional Baker's map model proposed by Kalda and Morozenko [1] predicts $\zeta_p = (2-\xi)\log_3(p+1)$, where $\xi = 4/3$ for Kolmogorov turbulence [2], which forever exhibits logarithmic growth. The model was shown to agree well with existing experimental and numerical data, but the large order of p question was not covered, only validating to about p=8.

The aim of this thesis is to attempt to determine whether ζ_p grows logarithmically or saturates at large p, by revisiting the one-dimensional model, this time, thanks to the increase of computing power available during the last 18 years, including large-p calculations up to p=24. 86 GB of data was collected for reliable results at high orders.

1. Literature Review

This chapter introduces the theoretical background and research problem addressed in this thesis. Section 1.1 reviews the fundamental concepts of turbulence and intermittency, providing the classical framework established by Kolmogorov and the origin of anomalous scaling in turbulent systems. Section 1.2 introduces passive scalar turbulence and the advection–diffusion equation governing tracer transport. Section 1.3 presents structure functions and scaling exponents, which are the primary statistical tools used to characterise intermittency. Section 1.4 examines the one-dimensional Baker's map model proposed by Kalda and Morozenko, including the mechanism of discontinuity front formation and the derivation of the logarithmic scaling prediction. Section 1.5 discusses the competing hypotheses of logarithmic growth and saturation of the scaling exponents at high orders, as well as the challenges involved in distinguishing between them. Finally, Section 1.5.4 outlines the approach of this thesis and the numerical strategy used to investigate the asymptotic behaviour of the structure function exponents up to order $p = 24$.

1.1 Turbulence and Intermittency

Fully developed turbulence refers to a regime of fluid motion at very high Reynolds numbers where energy is injected at large scales, cascades through a hierarchy of eddies, and is ultimately dissipated at small scales, where viscous forces take over. In this state the flow exhibits statistical regularities that are approximately universal. A central theoretical framework for describing this regime is presented by Andrei Kolmogorov in his 1941 paper [2], which assumes homogeneity, isotropy, and self-similarity in the inertial range. Under these assumptions, the paper predicts simple power-law scaling for velocity increments, leading to the $E(k) \sim k^{-\frac{5}{3}}$ energy spectrum and linear scaling of structure function exponents.

Experimental and numerical studies have consistently shown deviations from these simple scalings. These discrepancies are attributed to intermittency, which is a feature of turbulence according to which irregular fluctuations of large scales, large deviations from the mean, happen inconsistently and much more frequently than expected from Gaussian statistics. Physically, intermittency means that quantities such as velocity gradients or energy dissipation are not uniformly distributed but instead concentrate in localized structures, for example vortex filaments.

Kolmogorov's refined similarity hypothesis [3] accounts for these effects by relaxing the strict self-similarity assumption of [2] and introducing fluctuations in the local energy dissipation rate.

This leads to corrections in the scaling exponents of structure functions, which exhibits itself in anomalous scaling. Intermittency is often described using multifractal models, which capture the heterogeneous distribution of dissipation across scales. Thus Andrei Kolmogorov's 1941 paper provides a foundational baseline for turbulence scaling, but his later 1962 paper and subsequent developments highlight the essential role of intermittency in shaping the statistical structure of fully developed turbulent flows.

1.2 Passive Scalar Turbulence

A passive scalar is essentially any quantity carried by fluid flow, that does not affect it. Temperature, dye concentration, pollutant or pollen density are all great examples. Its evolution is governed by the advection and diffusion equation, presented in [1]:

$$\partial_t \theta + \mathbf{v} \cdot \nabla \theta = \kappa \nabla^2 \theta + \mathbf{f}(\mathbf{r}, t), \tag{1.1}$$

where $\theta(r, t)$ is the tracer density, $\mathbf{v}(\mathbf{r}, t)$ is a turbulent velocity field in a 3D or 2D space, $\mathbf{f}(\mathbf{r}, t)$ is a forcing, and the seed diffusivity κ is assumed to be very small, but not zero.

1.3 Structure Functions and Anomalous Scaling

The statistical properties of a turbulent passive scalar field are characterised by structure functions. The p-th order structure function is defined as:

$$S_p(r) = \langle |\theta(x + r) - \theta(x)|^p \rangle, \tag{1.2}$$

where $\theta(x)$ is the tracer density at position x, r is the spatial separation, and $\langle \cdot \rangle$ denotes an ensemble average. In the inertial range, structure functions follow a power law:

$$S_p(r) \sim r^{\zeta_p}, \tag{1.3}$$

where ζ_p are the scaling exponents that characterise the statistical structure of the field at different orders p.

For a simple, non-intermittent field, the scaling exponents would grow linearly with order: $\zeta_p = p\zeta_1$. This would correspond to a self-similar field where fluctuations at all scales follow the same statistical distribution. However, in fully developed turbulence, the exponents grow more slowly than linearly, a phenomenon known as anomalous scaling. Anomalous scaling is

a direct signature of intermittency, the tracer field is not self-similar, and extreme events such as sharp discontinuity fronts contribute disproportionately to high-order moments.

The classical result for the second-order exponent, derived independently by Obukhov [4] and Corrsin [5], gives:

$$\zeta_2 = 2 - \xi, \tag{1.4}$$

where ξ is the velocity field smoothness exponent, defined as the scaling exponent of the non-constant part of the velocity correlation function [1]. For Kolmogorov turbulence, the velocity field follows the well-known $r^{1/3}$ scaling, which corresponds to $\xi = 4/3$, giving $\zeta_2 = 2/3$.

The central open question in passive scalar turbulence concerns the behaviour of ζ_p at large orders p. The deviation of the full ζ_p curve from linear growth quantifies how intermittent the field is. Understanding the functional form of this deviation, whether it grows indefinitely or saturates, is the primary motivation of this thesis.

1.4 1D Baker's Map Model

1.4.1 Formation of Discontinuity Fronts

The fundamental mechanism driving intermittency in passive scalar turbulence is the formation of sharp discontinuity fronts in the tracer density field. To understand how these fronts emerge, consider an initially smooth tracer field with a constant gradient, and decompose the turbulent velocity field into components of characteristic scale a:

$$\mathbf{v}_a(\mathbf{r}, t) = \int_{a \leq |\mathbf{k}|^{-1} < 2a} \mathbf{v}(\mathbf{k}, t) e^{i\mathbf{k}\cdot\mathbf{r}} \mathrm{d}\mathbf{k}. \tag{1.5}$$

Each such component has a characteristic timescale $\tau_a \approx a^{2-\xi}$, defined as the time needed to transport a tracer particle by a distance of order a [1]. Initially, the largest eddies dominate the mixing, bringing regions of different tracer density closer together. Once the separation between these regions is halved, smaller eddies take over, operating on a timescale reduced by a factor of $2^{2-\xi}$. This process continues hierarchically through successively smaller scales, with the timescales forming a converging geometric series. As a result, regions of vastly different tracer density are brought into direct contact within a *finite* time, forming a sharp discontinuity front [1]. This mechanism is illustrated schematically in Figure 1.

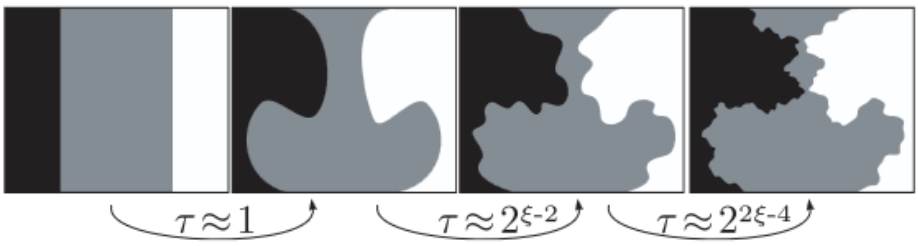

Figure 1. Simplified scheme of the formation of tracer discontinuities, reproduced from [1]. The characteristic time of eddies of size a scales as $\tau \approx a^{2-\xi}$. Due to the combined effect of large and small eddies, low- and high-density regions are brought into contact within a finite time.

1.4.2 Baker's Map and the Mapping $\mathcal{M}_{a,c}$

Baker's map is a classical tool for modelling chaotic mixing. It performs an area-preserving affine transformation on a region of space and then folds it to restore the original shape, mimicking the stretching and folding action of a chaotic fluid [1]. This stretching-folding motion is the fundamental kinematic process responsible for mixing in incompressible flows.

To model the effect of a single vortex of size a centred at position c on the tracer density profile $\theta(x)$ along a one-dimensional cross-section, Kalda and Morozenko introduce the mapping $\mathcal{M}_{a,c} : \theta(x) \to \theta'(x)$. The physical picture is illustrated in Figure 2: a shear flow caused by the vortex deforms initially straight isolines of the tracer field into S-shaped curves. As a result, a previously monotone segment of the profile $\theta(x)$ is replaced by a sequence of descending, ascending, and descending segments or what could be described as a "kink" in the profile. In the idealised version, these three segments are mirror images of each other and represent a threefold compression of the original segment along the x-axis. This compression is the key feature of the mapping: it reduces the spatial scale of tracer variations by a factor of three while preserving the amplitude of the density differences.

The full turbulent mixing process is then modelled by the iterative application of $\mathcal{M}_{a_t,c_t}$ to an initial profile $\theta_0(x)$, with the vortex size a_t and position c_t drawn randomly at each discrete time step t:

$$\theta_{t+1}(x) = \mathcal{M}_{a_t,c_t}[\theta_t(x)]. \tag{1.6}$$

To match the statistics of a homogeneous, scale-invariant turbulent flow, the position c_t is drawn from a uniform distribution, while the vortex size a_t follows a power-law distribution chosen such that the waiting time between successive mappings of scale a scales as $T_a \approx a^{2-\xi}$, consistent with the characteristic timescales of turbulent eddies [1].

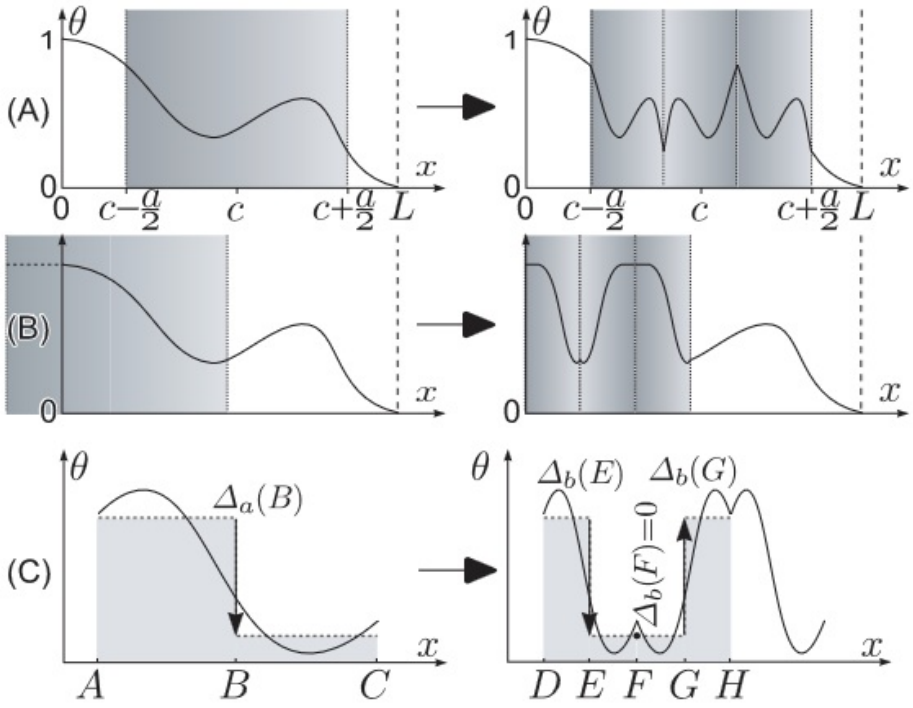

Figure 2. Mapping M_a, modelling the effect of a single vortex of size a on the tracer profile $\theta(x)$. The vortex may be far from the vessel boundary (case (A)), or it may be close to the boundary, at which a fixed tracer density value $\theta|_{x=0} \equiv 1$ is kept (case (B)). (C) As a result of the mapping $M_{2a,B}$, the old value of the mean density difference Δ_a (B) defines the possible range of new values at thrice smaller scale $b = a/3$. The smallest value is Δ_b (F) $= 0$, and the largest one is Δ_b (E) $= \Delta_b$ (G) $= \Delta_a$ (B).

1.4.3 Derivation of the Scaling Exponents

The theoretical analysis of the model is based on the probability density function (PDF) $f_a(\Delta_a)$ of the tracer density difference $\Delta_a(x) = |\bar{\theta}_a(x + a/2) - \bar{\theta}_a(x - a/2)|$ between neighbouring segments of length a, where $\bar{\theta}_a$ denotes a local average over a segment of length a.

When the mapping $\mathcal{M}_{a,c}$ acts on a segment, it compresses it into three sub-segments of length $a/3$. The density drop between two of the resulting sub-segments equals the original drop Δ_a, while between the other two there is no density drop. Consequently, as position varies across the compressed region, the density drop at scale $a/3$ takes all values between 0 and Δ_a approximately uniformly. This means the probability $f_a(\Delta_a)\mathrm{d}\Delta_a$ associated with the drop Δ_a at scale a contributes evenly to the PDF at the smaller scale $b = a/3$. This relationship is captured by the integral equation [1]:

$$f_b(\Delta) = \int_{\Delta}^{1} f_a(\Delta') \frac{\mathrm{d}\Delta'}{\Delta'}. \tag{1.7}$$

With the boundary condition $f_1(\Delta) \equiv 1$, direct integration of equation (1.7) yields:

$$f_a(\Delta_a) = \frac{|\ln(\Delta_a)|^n}{\Gamma(n+1)}, \qquad n = -(2-\xi)\log_3 a, \tag{1.8}$$

where Γ denotes the gamma function and n is the effective number of compression iterations [1].

The structure function scaling exponents ζ_p are then obtained from the condition $\int f_a(\Delta)\Delta^p \mathrm{d}\Delta \propto a^{\zeta_p}$. Evaluating this integral and comparing with the classical second-order result $\zeta_2 = 2 - \xi$ to fix the effective compression factor, one obtains the central result of the model [1]:

$$\zeta_p = (2-\xi)\log_3(p+1). \tag{1.9}$$

For Kolmogorov turbulence with $\xi = 4/3$, this reduces to $\zeta_p = \frac{2}{3}\log_3(p+1)$. This expression predicts unbounded logarithmic growth of the scaling exponents with order p, in contrast to the competing hypothesis of saturation at large p, which is the central question this thesis addresses.

1.5 Logarithmic Growth or Saturation

Two competing predictions exist for the asymptotic behaviour of ζ_p as $p \to \infty$. The first predicts unbounded logarithmic growth according to Eq. (1.9). The second predicts that ζ_p saturates at a finite ceiling. Saturation is rigorously established in the large-dimensionality limit $d(2-\xi) \gg 1$ [6], and the data of [7] suggest a saturation value of approximately $\zeta_p \approx 1.4$ when read from their figure. Distinguishing between the two predictions is the primary motivation of this thesis, as the logarithmic growth is extremely slow, rendering the two curves practically indistinguishable within the range of p previously studied.

1.5.1 Logarithmic Growth

The Kalda (2008) model predicts that ζ_p grows without bound according to equation (1.9):

$$\zeta_p = (2-\xi)\log_3(p+1), \tag{1.10}$$

meaning the scaling exponents continue to increase indefinitely with order p, albeit increasingly slowly. This prediction follows directly from the hierarchical Baker's map model and is consistent with the physical picture of a cascade of discontinuity fronts at all scales. The model was shown to agree well with existing experimental and numerical data up to approximately $p \approx 8$ [1].

1.5.2 Saturation

The competing hypothesis predicts that ζ_p approaches a finite ceiling as $p \to \infty$, meaning the exponents stop growing beyond some limiting value ζ_∞. This prediction is supported by rigorous analytical results obtained by Balkovsky and Lebedev [6], who showed that saturation indeed occurs in the limit of large spatial dimensionality d, specifically when $d(2-\xi) \gg 1$. Whether the reliable exponents measured here remain below this ceiling is examined in Section 3.4.

The saturation hypothesis is also consistent with the existence of mature fronts, discontinuity fronts across which the tracer density drops by a non-negligible fraction of the total density difference across the entire domain. The existence of such mature fronts has been confirmed numerically [1], and their presence would imply the saturation of ζ_p at high orders.

1.5.3 Distinguish between the two predictions

Despite being qualitatively different predictions, the two hypotheses are practically very difficult to distinguish from data. The fundamental reason is that logarithmic growth is extremely slow. For Kolmogorov turbulence with $\xi = 4/3$, the predicted values are:

$$\zeta_p = \frac{2}{3}\log_3(p+1), \tag{1.11}$$

which increases by only $\frac{2}{3}\log_3 2 \approx 0.14$ each time p doubles. A saturation curve, chosen to match at low orders, would deviate from this by an amount that is smaller than typical statistical uncertainties in the data at moderate values of p. This difficulty is explicitly acknowledged by Jaan Kalda in 2008, where it is noted that the existing numerical and experimental data are not precise enough to exclude either asymptotic form [1]. The problem is compounded by the fact that statistical uncertainties in ζ_p grow rapidly with increasing order p, since high-order moments are increasingly dominated by rare extreme events that are difficult to sample reliably.

1.5.4 Approach of This Thesis

The original Kalda paper published in 2008 validated the model only up to approximately $p \approx 8$, a range in which both predictions are nearly indistinguishable. To make meaningful progress on this question, two conditions must be met: the calculation must be extended to sufficiently large p, and the statistical precision must be high enough that the difference between the two curves exceeds the measurement uncertainty.

This thesis addresses both conditions. Structure functions are computed up to order $p = 24$, a range not previously studied in the context of this model. To ensure reliable statistics at such high orders, 86 GB of simulation data was collected across 20 independent chains. The key diagnostic tools used to compare the two predictions are a direct comparison of the measured ζ_p curve against both functional forms, a linearity test in $\log_3(p+1)$ coordinates, which should yield a straight line if the Kalda prediction is correct and an analysis of the successive increments $\zeta_p - \zeta_{p-2}$, which decay as $\frac{1}{p}$ under logarithmic growth but converge to zero faster under saturation. The scaling exponents are extracted using two complementary methods. The first is a direct power-law fit in log-log space over an inertial-range window selected by maximising the coefficient of determination R^2, with no reference to the theoretical prediction, ensuring the procedure cannot circularly favour either hypothesis. The second is Extended Self-Similarity (ESS), in which $\log S_p$ is fitted against $\log S_2$ globally over all lag values, recovering the relative exponent ζ_p/ζ_2. Because both S_p and S_2 are affected by the same finite-step distortions, ESS cancels much of this systematic error and provides an independent cross-check on the direct fit.

2. Methodology

This chapter presents the data and methodology used in the one-dimensional model of scalar turbulence. Section 2.1 details the data generation process, describing how the iterative mapping $M_{a,c}$ is applied to the tracer density profile. This includes the choice of initial conditions and boundary conditions, the probability distributions of the vortex size and position parameters, and the array discretization at the diffusion scale $\delta = \kappa^{1/\xi}$. Section 2.2 prepares an overview of the post processing of generated data.

All data generation was executed using the C++ programming language and its libraries, because of its precompiled nature and therefore superior speed when compared to python, while post processing and visualizations were executed using the Python programming language and its relevant libraries.

2.1 Data Generation

This section presents how the data for this thesis was generated. Subsection 2.1.1 goes over the initial conditions used, subsection 2.1.2 presents the boundary conditions used and how they were implemented and 2.1.3 talks about the implementation of the mapping $\mathcal{M}_{a,c}$.

2.1.1 Initial Conditions

An initial tracer density profile was generated as a simple

$$T(x) = -x, \quad x \in \begin{cases} -20481, -20480, -20479, \ldots, \\ -1, 0, 1, \ldots, \\ 61439, 61440, 61441 \end{cases}$$

array.

The possible vortexes generated by the simulation were as follows:

$$V = \{\, 3 \cdot 2^n + 2 \mid n \in \mathbb{N},\ n \geq 1,\ 3 \cdot 2^n + 2 < \mathsf{N} \,\} \tag{2.1}$$

, where $N = 40960$.

2.1.2 Boundary Conditions

Differing from the boundary conditions of the 2008 paper, a cyclic model, such that

$$T(x) + N = T(x + N) \tag{2.2}$$

was chosen, for its ease of implementation and smaller impact on tracer density near the edges. The implementation was achieved by creating a $2N$ long array of T values, and computing on the second and third quarters. The remaining first and fourth quarters were filled with values from the third and second quarters respectively, with a shift of N, as seen above (2.2).

However, this choice introduces a known systematic limitation. The initial condition $\theta_0(x) = -x$ combined with the periodicity constraint preserves a mean gradient of order $G \approx 1$ across the domain throughout the simulation. The increment $|\theta(x+a)-\theta(x)|$ then contains a deterministic contribution $\sim Ga$ in addition to the turbulent fluctuations $\sim a^{\zeta_1}$. Since $\zeta_1 \approx 0.42 < 1$, this ramp term dominates at large a and does so more strongly for higher-order moments, producing an upward bias in ζ_p that grows with p. This effect is most pronounced at high orders $p > 12$, where it is the primary driver of the systematic overestimation. Removing it would require subtracting the mean gradient from the field before computing structure functions, which is left for future work.

2.1.3 Mapping

The core of the simulation is the iterative application of the mapping $\mathcal{M}_{a,c}$ to the tracer density profile $\theta(x)$. At each discrete time step t, a vortex of size a_t is drawn from the set V according to the weight distribution, and a position c_t is drawn uniformly from $[0, N]$. The mapping then acts on the symmetric slice of the profile centred at c_t with half-width $a_t/2$.

The operation proceeds in three steps. First, the slice of $\theta(x)$ centred at c_t is extracted. Second, the slice is centred by subtracting its mean value, ensuring the mapping preserves the mean tracer density of the profile. Third, the centred slice is folded: the profile within the slice is compressed by a factor of three along the x-axis and reflected, producing the descending-ascending-descending kink structure described in Section 1.4.2. The result replaces the original slice in the profile.

The vortex sizes are drawn from the discrete set described in (2.1). Each vortex size $v \in V$ is assigned a weight proportional to $v^{-5/3}$, implementing the power-law frequency distribution required to match the statistics of Kolmogorov turbulence. Specifically, the waiting time between successive mappings of scale a must scale as $T_a \approx a^{2-\xi}$ with $\xi = 4/3$, which is satisfied by the weight $v^{-5/3} = v^{-(1+2/3)} = v^{-(1+\xi/2)}$ [1]. The position c_t is drawn from a discrete uniform distribution over $[0, N]$, consistent with the assumption of statistically

homogeneous turbulence.

The diffusion scale is implemented implicitly through the array discretisation. Rather than explicitly smoothing the profile at each step, the array is stored at resolution $\delta = \kappa^{1/\xi}$, so that sub-diffusion-scale fluctuations are never represented. When the mapping transforms a point i into a point j, the update rule:

$$\theta_{j,\,t+1} = \frac{1}{3} \sum_{|k-i|\leq 1} \theta_{k,\,t} \tag{2.3}$$

provides an effective averaging over the three nearest neighbours, which acts as a minimal smoothing at the diffusion scale [1]. The array length $N = 40960$ therefore corresponds to the ratio of the domain width to the diffusion scale.

The simulation is run for a total of 1679360000 iterations per chain. Structure functions are recorded every 40960 iterations, providing snapshots of the fully developed field at regular intervals. A burn-in diagnostic confirmed that S_2 is stationary from the first recorded snapshot across all chains, indicating the field reaches statistical equilibrium within the first recorded snapshot, after 40960 iterations.

2.1.4 Computing $S_p(a)$ and Chain Structure

The p-th order structure function $S_p(a)$ is estimated directly from the simulated tracer density profile $\theta(x)$ as a spatial average over all pairs of points separated by lag a:

$$S_p(a) = \frac{1}{N_a} \sum_{x} |\theta(x+a) - \theta(x)|^p, \tag{2.4}$$

where the sum runs over all valid positions x and N_a is the number of such pairs. This estimator converges to the true ensemble average $\langle |\theta(x+a) - \theta(x)|^p \rangle$ as the number of samples increases.

Structure functions are computed at a discrete set of lag values a, chosen to provide dense coverage at small scales while remaining computationally tractable at large scales. The full set of lags used were:

$$a \in \{2^k \mid 0 \leq k \leq 15\} \cup \{3 \cdot 2^k \mid 0 \leq k \leq 13\}. \tag{2.5}$$

This set spans nearly five decades in scale, from $a = 1$ (the diffusion scale) to $a = 32768$,

which is 80% of the array length $N = 40960$. The inclusion of both powers of two and their multiples by three ensures that the inertial range is sampled at sufficient density for reliable power-law fitting. It should be mentioned that three of the lags $a \in \{4, 16, 192\}$, were omitted due to a coding mistake and were discovered too late for correction, however, three missing lags at such a low value should not significantly affect the results.

Structure functions are computed at regular intervals throughout the simulation and written to output files, with one file per order p per chain. The orders computed are $p \in \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 14, 16, 18, 20, 22, 24\}$, covering all integers up to $p = 12$ and then even values up to $p = 24$.

2.1.4.1 Chain Structure and Ensemble Averaging

A single long simulation run would in principle provide sufficient statistics, but is impractical for two reasons. First, the decorrelation time of the tracer field, the time required for the field to lose memory of its initial condition, is long, meaning that successive snapshots within a single run are statistically correlated. Secondly, a single run cannot exploit modern multicore processors.

Instead, the simulation is run as a collection of independent chains, each starting from the same initial condition $\theta_0(x) = -x$ but using a different random seed. Each chain runs on a dedicated CPU thread in parallel using OpenMP, so the total wall-clock time is determined by the runtime of a single chain rather than the sum of all chains. This means that doubling the number of chains doubles the total statistics while keeping the runtime approximately constant, as long as the number of chains does not exceed the number of available CPU threads.

The structure functions from all chains and all time snapshots are then combined into a single ensemble average. If chain c contributes M_c snapshots, each yielding an estimate $S_p^{(c,m)}(a)$, the final estimate is:

$$\hat{S}_p(a) = \frac{1}{\sum_c M_c} \sum_c \sum_{m=1}^{M_c} S_p^{(c,m)}(a). \tag{2.6}$$

The statistical uncertainty of this estimate decreases as $1/\sqrt{\sum_c M_c}$, so collecting more chains directly reduces the error on $\hat{S}_p(r)$ and consequently on the fitted exponent ζ_p. This is particularly important at high orders p, where the structure function is dominated by rare extreme events and converges slowly. In this work, 20 independent chains were run, producing a total data output of 86 GB.

2.2 Post Processing

2.2.1 Extraction of ζ_p

The scaling exponent ζ_p is extracted from the ensemble averaged structure function $\hat{S}_p(a)$ by fitting a power law in the inertial range, the range of scales over which the relation $S_p(a) \sim a^{\zeta_p}$ is expected to hold. Taking the logarithm of both sides gives:

$$\log \hat{S}_p(a) = \zeta_p \log a + C, \tag{2.7}$$

where C is a constant. The exponent ζ_p is therefore the slope of $\log \hat{S}_p(a)$ plotted against $\log a$, and can be estimated by linear regression in log-log space.

2.2.1.1 Inertial Range and Window Selection

Not all lag values a are suitable for fitting. At very small scales, a approaches the diffusion scale δ and the structure function is influenced by the smoothing effect of diffusion rather than by inertial-range scaling. At very large scales, a becomes a significant fraction of the array length N, and two effects degrade the estimate: the number of available pairs of points decreases, reducing statistical precision, and the boundary conditions begin to influence the result. The inertial range therefore occupies an intermediate window of scales, and the precise choice of this window affects the fitted value of ζ_p.

To select the fitting window in a principled and hypothesis-independent way, a best-window algorithm is employed. For each possible contiguous subset of the available lag values of length at least $w_{\min}$ points, an ordinary least squares regression is performed in log-log space. The window whose regression yields the highest coefficient of determination R^2 is selected as the inertial range. To prevent the algorithm from selecting plateau regions where the structure function has saturated, windows whose fitted slope falls below a minimum physically plausible value are rejected. This floor is set to half the Kalda prediction for $p = 2$, giving a threshold of approximately 0.33, well below any expected ζ_p. Crucially, no reference to the theoretical prediction ζ_p^{theory} is made at any point in the window selection, ensuring the procedure cannot circularly bias the results toward the Kalda model.

To further reduce the influence of finite-step artefacts that affect both S_p and S_2 in a correlated way, the scaling exponents are also extracted using ESS. Rather than fitting $\log S_p$ against $\log a$ directly, ESS fits:

$$\log S_p = \mu_p \cdot \log S_2 + \text{const}, \tag{2.8}$$

where S_2 is the second-order structure function evaluated at the same lag values. The slope

$\mu_p = \zeta_p/\zeta_2$ is then multiplied by the known value $\zeta_2 = \frac{2}{3}$ to recover ζ_p. Because both S_p and S_2 are subject to the same finite-step distortions, plotting one against the other cancels much of this systematic error, yielding more reliable exponent estimates than direct log-log fitting against lag. The ESS fit is performed globally over all available lag values without any window restriction, giving a single slope per order p. This global ESS estimate is taken as the primary result.

The stability of each direct fit is assessed by a jackknife procedure in which the selected window is re-fitted after successively removing one or two points from each edge. A genuine scaling region produces slopes that are mutually consistent within twice the standard error of the regression.

2.2.1.2 Linear Regression and Error Estimation

Within the selected inertial range, the exponent ζ_p is estimated by ordinary least squares regression of $\log \hat{S}_p(a)$ on $\log a$:

$$\zeta_p = \frac{n \sum_i \log a_i \cdot \log \hat{S}_p(a_i) - \sum_i \log a_i \sum_i \log \hat{S}_p(a_i)}{n \sum_i (\log a_i)^2 - \left(\sum_i \log a_i\right)^2}, \tag{2.9}$$

where the sum runs over the n lag values in the selected inertial range. The standard error of the slope provides an estimate of the uncertainty on ζ_p:

$$\sigma_{\zeta_p} = \hat{\sigma} \sqrt{\frac{1}{n \sum_i (\log a_i)^2 - \left(\sum_i \log a_i\right)^2}}, \tag{2.10}$$

where $\hat{\sigma}^2$ is the residual variance of the regression. These error estimates are propagated into the final ζ_p curve and are used to assess whether the difference between the logarithmic and saturation predictions exceeds the measurement uncertainty at high orders p.

3. Results

This chapter presents the results of the numerical simulations of the one-dimensional Baker's map model. Section 3.1 provides a visual verification that the simulation produces the expected physical behaviour. Section 3.2 presents the measured scaling exponents ζ_p for orders $p = 1$ to $p = 24$. Section 3.3 applies two diagnostic tests to distinguish between the competing theoretical predictions. Section 3.4 assesses which model the data supports and how conclusively.

3.1 Tracer Density and Structure Function Profiles

Before extracting scaling exponents, it is important to verify that the simulation produces the qualitative behaviour expected from the model. Two controls are performed.

Figure 3 shows a snapshot of the tracer density profile $\theta(x)$ after the field has fully developed. The profile exhibits the characteristic sharp discontinuity fronts or regions where the tracer density changes abruptly over a very short distance. These fronts are the physical signature of intermittency and confirm that the simulation is operating in the correct regime. The fronts emerge at all scales, consistent with the hierarchical cascade mechanism described in Section 1.4.1.

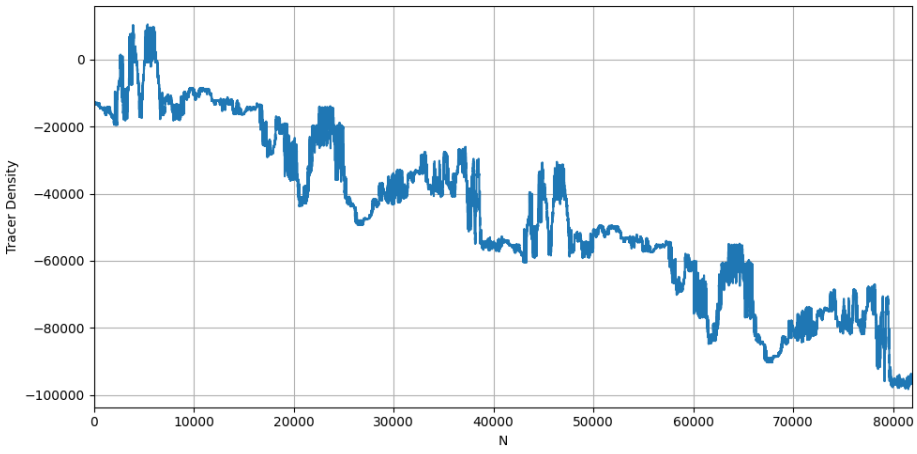

Figure 3. An example of a tracer density profile after 1679360000 iterations.

Figure 4 shows the second-order structure function $S_2(a)$ plotted against lag a on a log-log scale for a representative chain. A clear power-law scaling region is visible over approximately 3 decades in scale, confirming the existence of an inertial range suitable for fitting. The slope of this region is consistent with the theoretical prediction $\zeta_2 = \frac{2}{3}$, providing a first validation of

the simulation. At small a the curve flattens due to diffusion scale smoothing, and at large a it deviates due to finite array size effects, consistent with the discussion in Section 2.2.1.1. At large values of p, the curve quality falls, as seen in figure 5.

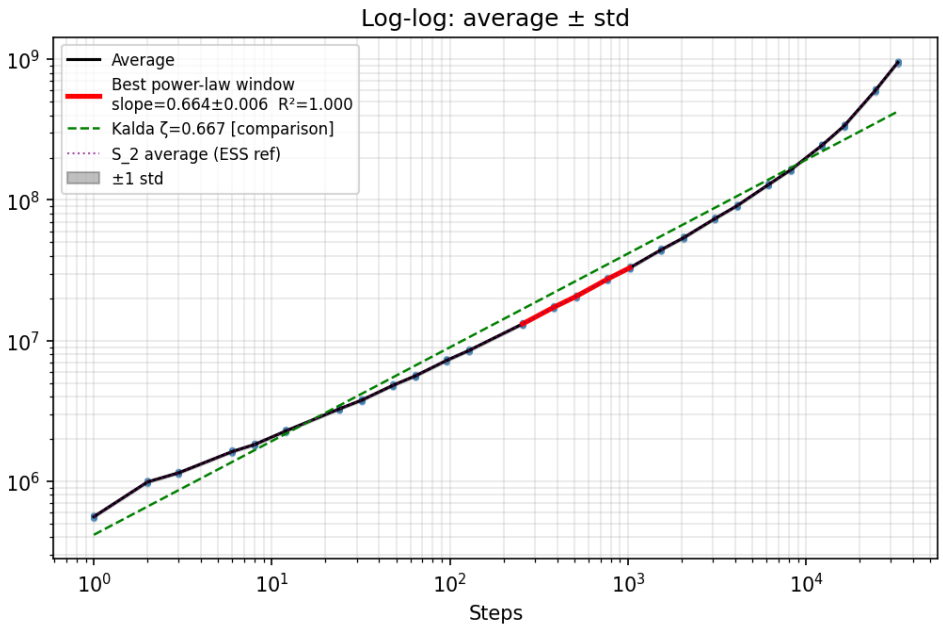

Figure 4. Second-order structure function $S_2(a)$ plotted against lag (steps) a, averaged over all chains and snapshots. The black curve shows the full ensemble average and the red segment indicates the best-window fit used to extract ζ_2. A clear power-law scaling region is visible over almost the full scale. The fitted slope is consistent with the theoretical prediction $\zeta_2 = \frac{2}{3}$. At small a the curve flattens due to diffusion scale smoothing, and at large a it steepens due to finite array size effects.

3.2 Measured ζ_p Values

Figure 6 shows the measured scaling exponents ζ_p for orders $p = 2$ to $p = 24$, extracted using the best-window regression procedure described in Section 2.2.1. Error bars represent the standard error of the linear regression slope within the selected inertial range. Four reference curves are shown alongside the measurements: the Kalda model $\zeta_p = \frac{2}{3}\log_3(p+1)$ [1], the Kraichnan prediction [8], the stagnating model [7], and the simulation data of Antonia et al. [9]. The present results are shown as two separate estimates, global ESS and window ESS.

The measured exponents follow the theoretical curve closely at low orders $p \leq 12$, consistent with the validation of the model against experimental data in [1]. At $p = 14$–16 the exponents remain consistent with logarithmic growth but with increasing uncertainty, and these values should be treated with caution. At orders $p > 16$ the measurements are dominated by statistical and systematic effects that the present setup cannot resolve, the effective number of independent samples contributing to the moment estimator falls below a reliable threshold, and a mean gradient preserved by the cyclic boundary conditions contaminates high-order moments at large lags. Consequently, the results at $p > 16$ are not reported as quantitative estimates of ζ_p.

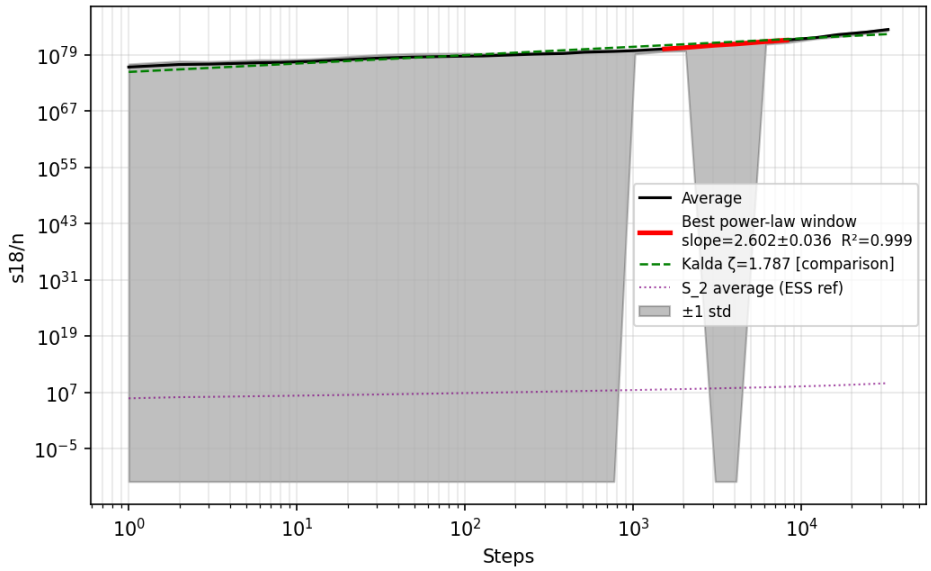

Figure 5. Structure function $S_{18}(a)$ plotted against lag a, averaged over all chains and snapshots. The black curve shows the full ensemble average and the red window indicates the best-window fit used to extract ζ_{18}. The standard deviation has become significantly larger than for $p = 2$. The direct window fit yields a slope of 2.602 ± 0.035 over steps 1536–8192, while the global ESS estimate gives $\zeta_{18} = 1.90 \pm 0.04$. The deviation from the Kalda prediction $\zeta_{18} \approx 1.787$ is attributed to finite-size effects at high order.

3.3 The Linearity and Increments Tests

Figure 7 shows the measured ζ_p plotted against $\log_3(p+1)$. The data points follow a linear trend up to $p \approx 12$, with no downward flattening at high orders. Instead, the high-p points depart upward from the line, consistent with finite-size effects rather than saturation. The linearity of the relationship up to moderate p is the key diagnostic for logarithmic growth, if saturation were occurring, the points would curve downward and fall below the line at high p.

Figure 8 shows the successive increments $\zeta_p - \zeta_{p-2}$ plotted against p. At low orders $p \leq 10$ the measured increments follow the theoretical $\frac{2}{3}\log_3 \frac{p+1}{p-1}$ decay closely, consistent with logarithmic growth. At $p = 12$–16 the increments level off rather than continuing to decay, which reflects the onset of the systematic overestimation discussed in Section 3.2. No convergence toward zero is observed within the reliable range, which would be the expected signature of saturation. Beyond $p = 16$ the increments are not a reliable diagnostic for either hypothesis.

3.4 Conclusion

The central question of this thesis is whether ζ_p grows logarithmically without bound or saturates at a finite ceiling. At the level of ζ_p, the two hypotheses are nearly indistinguishable across the low orders where both this work and the original validation [1] are reliable, so the conclusion is not built on matching one curve more closely than the other there, but rather

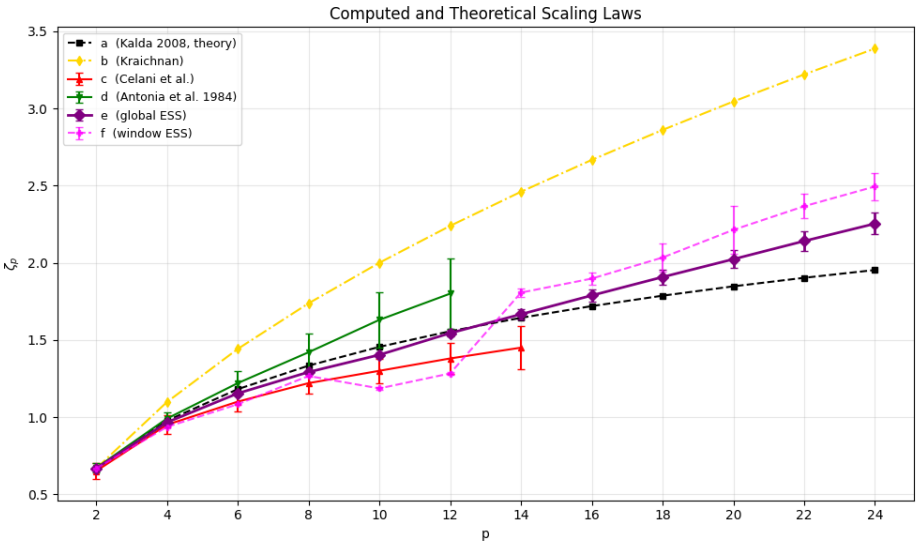

Figure 6. Theoretical and experimental curves of the ζ_p functions. The black line represents the theoretical curve by Kalda [1] (a), the yellow lines represent the theoretical Kraichnan curve, formula from [8] (b), the red line represents the stagnating model, data from [7] (c), the green line represent simulation data from [9] (d), the purple line (e) represents global ESS estimates from this work and the pink line (f) represents window ESS estimates from this work. The erratic behaviour of the window ESS at high orders reflects the absence of a shared inertial range between S_p and S_2, confirming that the global ESS is the more reliable primary estimator at $p \leq 12$.

is built on a feature that does not require the contaminated high-p points, that being the saturation ceiling itself.

A saturating ζ_p approaches its asymptote ζ_∞ , a finite ceiling at $\zeta_\infty \approx 1.4$, from below. The value implied by the mature-front data of [7] requires $\zeta_p < 1.4$ at every finite p. The reliable exponents violate this directly. The global ESS estimates reach

$$\zeta_{10} = 1.40 \pm 0.02,$$

and

$$\zeta_{12} = 1.54 \pm 0.02,$$

the latter exceeding the proposed ceiling by multiple standard errors, and they are still rising, with no sign of a downward turn that an approach to a finite asymptote would require.

Crucially, the ceiling-crossing does not hinge on trusting the anomalously large $\zeta_{12} - \zeta_{10}$ increment. Because ζ_p is non-decreasing in p, the reliable value $\zeta_{10} = 1.40 \pm 0.02$ already places a floor under ζ_{12} at the proposed ceiling itself, and the positive ramp bias of Section 4.2 can only raise the measured ζ_{12} further, the exceedance therefore survives even if the $p = 12$ increment is treated as fully contaminated. Moreover, a curve saturating at $\zeta_\infty \approx 1.4$ must lie strictly below 1.4 at every finite order, yet ζ_p has already reached the asymptote by $p = 10$ while its increments remain ≈ 0.11 — by itself difficult to reconcile with imminent

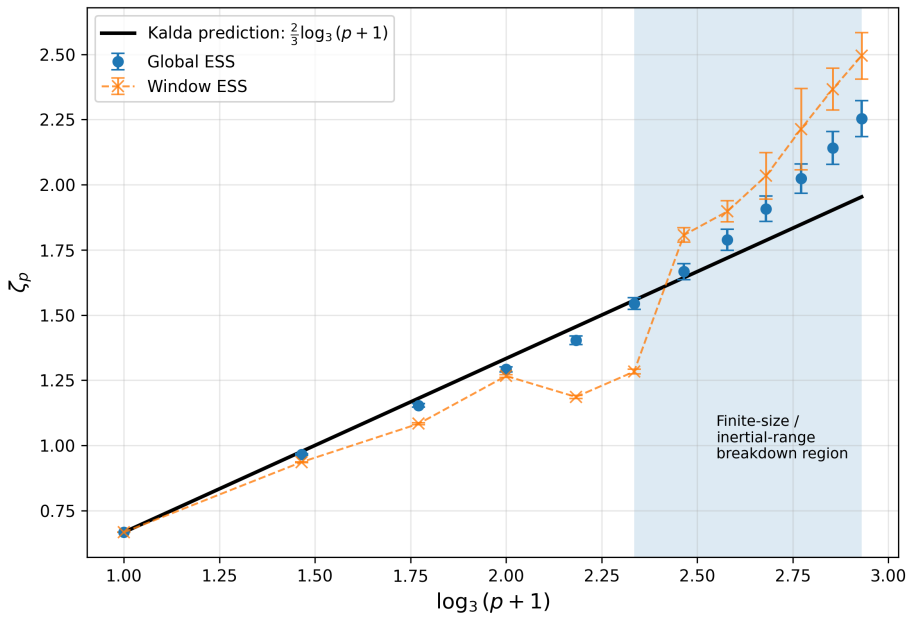

Figure 7. Measured ζ_p plotted against $\log_3(p+1)$ for $p = 2$ to $p = 24$. The data exhibit a clear linear trend up to $p = 12$, but due to finite-size effect starts growing rapidly. The black line follows the theoretical curve, blue datapoints representing the full range ESS data and orange datapoints represent the window ESS data.

saturation at 1.4, independently of the higher orders.

These points lie within the range where the moment estimator is converged, as discussed in Section 4.1, and where the direct fit and ESS agree, so the exceeding cannot be attributed to the high-order artifacts discussed in Section 4.2. In addition, the conclusion is insensitive to the precise ceiling value and even when reading the saturation literature as generously as $\zeta_\infty \approx 1.5$, the continued rise of ζ_p through $p = 12$ and beyond is difficult to reconcile with imminent saturation. The model does not saturate at $\zeta_\infty \approx 1.4$; the basis for this claim rests on the reliable data and does not depend on the contaminated high-p points.

The relevance of this result to real turbulence rests on two further premises. First, the model has been validated against experimental scalar turbulence data up to $p \approx 8$ [1]. Second, as argued in Section 4.4, the rigorous saturation result of Balkovsky and Lebedev [6] holds in the large-dimensionality limit $d(2-\xi) \gg 1$, a regime far from physical two- or three-dimensional turbulence. Together these support the reasoning that the absence of saturation in the 1D model reflects the behaviour of real scalar turbulence rather than a model artefact. Therefore, the model's failure to saturate at $\zeta_\infty \approx 1.4$ argues against a finite ceiling in that range for real scalar turbulence.

The remaining diagnostics are consistent with this conclusion. The global ESS exponents track the logarithmic form

$$\zeta_p = \frac{2}{3}\log_3(p+1)$$

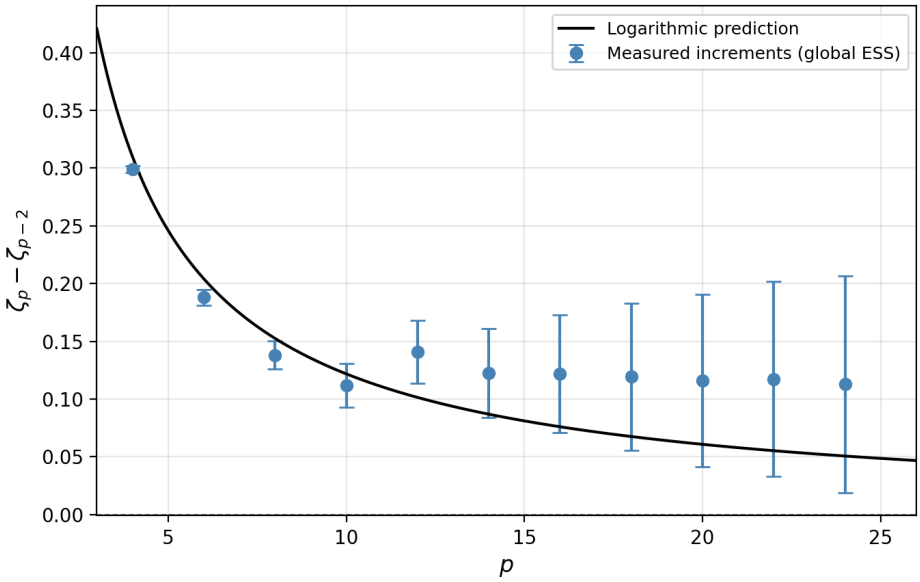

Figure 8. Successive increments $\zeta_p - \zeta_{p-2}$ as a function of p, extracted using global ESS. The black curve shows the theoretical prediction $\frac{2}{3}\log_3 \frac{p+1}{p-1}$ for logarithmic growth. At low orders $p \leq 10$ the measured increments follow the theoretical decay closely. At higher orders the increments level off at approximately 0.12 rather than continuing to decay, which is attributed to the finite-size overestimate of ζ_p at high orders. Crucially, no convergence toward zero is observed within the measured range, which would be the expected signature of saturation.

to within 3.6% across $p \leq 10$, in agreement with the direct power-law fit, a cleaner and better cross-validated confirmation of the model than was previously available. The increment test, displayed in Figure 8, shows the successive differences

$$\zeta_p - \zeta_{p-2}$$

following the predicted decay

$$\frac{2}{3}\log_3\left(\frac{p+1}{p-1}\right)$$

up to $p \approx 10$, with no convergence towards zero, which would be the signature of saturation.

At $p = 14$–16 the exponents remain consistent with logarithmic growth but with weakening statistical reliability. Beyond $p = 16$ the present dataset cannot be used to determine ζ_p quantitatively, and the interpretation of the high-p overshoot is discussed in Section 4.4. The claim that saturation is excluded rests on the $p \leq 12$ ceiling comparison, not on the high-order points.

Table 1. Measured scaling exponents ζ_p from global ESS, compared to the Kalda prediction $\frac{2}{3}\log_3(p+1)$. Results at $p \leq 12$ are considered reliable. Values at $p = 14$–16 are consistent with logarithmic growth but have decreasing statistical reliability (noted †). Values at $p \geq 18$ are not interpreted as quantitative estimates and are shown for completeness only (noted ‡).

p	ζ_p (ESS)	$\pm 1\sigma$	ζ_p (Kalda)	deviation
2	0.6667^*	0.0000	0.6667	$+0.0\%$
4	0.9655	0.0031	0.9767	-1.1%
6	1.1535	0.0064	1.1808	-2.3%
8	1.2916	0.0101	1.3333	-3.1%
10	1.4033	0.0162	1.4551	-3.6%
12	1.5441	0.0222	1.5565	-0.8%
14	$1.6665^\dagger$	0.0313	1.6433	$+1.4\%$
16	$1.7884^\dagger$	0.0405	1.7193	$+4.0\%$
18	$1.9077^\ddagger$	0.0490	1.7868	$+6.8\%$
20	$2.0234^\ddagger$	0.0565	1.8475	$+9.5\%$
22	$2.1409^\ddagger$	0.0631	1.9027	$+12.5\%$
24	$2.2537^\ddagger$	0.0694	1.9533	$+15.4\%$

* The apparent zero uncertainty at $p = 2$ is an artifact of ESS normalization: ζ_2 is fixed by construction and therefore carries no fitted variance.

4. Discussion

In this chapter, we go over why some choices were made, and how the paper could be improved further. Section 4.1 discusses, why the upper limit of p=24 was chosen, 4.2 goes over some of the limitations that remain, in section 4.3, further improvements are considered and section 4.4 talks about the broader implications of this paper.

4.1 Choice of Maximum Order $p = 24$

The simulation was carried out up to $p = 24$ in order to maximise the separation between the logarithmic and saturation curves. However, statistical reliability decreases rapidly with increasing p, high-order moments are dominated by rare extreme events that are undersampled even with 86 GB of data across 20 chains. Based on the effective independent sample count across chains, the results are considered reliable up to $p = 12$, and consistent but increasingly uncertain at $p = 14$–16. At $p \geq 18$ the variance of the moment estimator itself is unreliable, and these points are not used as quantitative claims. This is consistent with the general expectation that for a dataset of this size the honest convergence horizon lies around $p \approx 12$–14, well below the $p = 24$ ceiling that motivated the data collection.

The choice is also motivated by computational constraints. The orders used $p \in \{2, 4, \ldots, 22, 24\}$ were computed simultaneously within each chain iteration, meaning that extending to higher orders would have increased the per-iteration computation time without a corresponding improvement in the statistical precision of the result.

4.2 Sources of Uncertainty and Finite-Size Effects

Several sources of uncertainty affect the measured ζ_p values.

The dominant source at high orders is statistical uncertainty from finite sampling. High-order structure functions $S_p(a)$ are sensitive to the tails of the distribution of tracer density differences, which are populated by rare extreme events. Even with 86 GB of data across 20 chains, the tails are undersampled at high p, leading to the growing error bars visible in Figure 5. This effect is irreducible without either a larger ensemble or a longer simulation.

A second source is finite array size. The array length $N = 40960$ limits the width of the inertial range to approximately 3 decades in scale, compared to $N = 10^6$ in the original paper [1]. An additional systematic arises from the cyclic boundary conditions: the initial condition $\theta = -x$

together with the periodicity constraint preserves a mean gradient of order $G \approx 1$ across the domain throughout the simulation. The increment $|\theta(x+a)-\theta(x)|$ then contains a deterministic contribution $\sim Ga$ in addition to the turbulent fluctuations $\sim a^{\zeta_1}$. Since $\zeta_1 \approx 0.42 < 1$, the ramp term dominates at large a and does so more strongly for higher-order moments, producing an upward bias in ζ_p that grows with p. This ramp contamination is the primary driver of the systematic upward deviation of the measured ζ_p from the Kalda prediction at orders $p > 12$: in the ramp-dominated tail $S_p \propto a^p$ and $S_2 \propto a^2$, so the ESS slope tends toward $p/2$, which exceeds the true ζ_p and does so more strongly as p increases. Removing this effect would require subtracting the mean gradient from the field before computing structure functions, which is left for future work.

A minor additional source is three missing lags: $a \in \{4, 16, 192\}$ were omitted due to coding errors discovered after data collection. All three fall within the lower part of the inertial range, and their absence is not expected to significantly affect the fitted values ζ_p, as the regression uses many lag points and removing a small number does not significantly alter the slope. Note that Eq. 2.5 as written includes these values, the lag set actually used in the code differs from that equation by their omission.

Finally, the cyclic boundary conditions used in this work differ from the fixed boundary conditions used in the original paper [1]. Although cyclic boundaries reduce edge effects and are easier to implement, they alter the large-scale structure of the tracer field in a way that may slightly affect the structure functions at the largest lags. This is consistent with the observed deviation of $S_2(a)$ from power-law behaviour at large a seen in 4.

4.3 Odd-Order Structure Functions

This work computed structure functions only for even orders p, since for odd orders the absolute value in Eq. 2.4 is essential and without it, positive and negative increments partially cancel, the resulting signed moment grows more slowly, and the statistical convergence is far worse than for even p. Resolving odd-order exponents reliably would require substantially more data than collected here and is left for future work.

4.4 Broader Implications

The high-order overshoot, although not usable as a quantitative measurement of ζ_p, still carries information once the sign of its bias is accounted for, and that information points away from saturation rather than toward it. The contamination identified in Section 4.2 is strictly positive, the mean-gradient ramp pushes the ESS slope toward $p/2$ in the ramp-dominated tail, so its contribution to the increment $\zeta_p - \zeta_{p-2}$ is positive and grows with p. The two hypotheses therefore make opposite predictions for the measured increments in the contaminated region.

Under logarithmic growth, the true increments are already decaying slowly, and adding a positive bias yields a roughly flat plateau. Under saturation, the true increments would be near zero, and the same positive bias would instead drive the measured increments upward, increasingly so with p. The observed increments in Fig. 8 sit on a flat, gently declining plateau near 0.12 and show no such rise. The shape of the contaminated data is thus consistent with logarithmic growth viewed through a positive bias, and inconsistent with saturation viewed through the same bias. This is not an independent measurement, but it means that even the high-order points excluded from quantitative interpretation do not provide support for saturation.

The exclusion of saturation also is relevant to the broader problem. The model of [1] is a simplified 1D representation of a fundamentally 2D or 3D phenomenon, and the rigorous saturation result of Balkovsky and Lebedev [6] holds in the large-dimensionality limit $d(2-\xi) \gg 1$, a regime far from physical turbulence in two or three dimensions. The present results are consistent with the argument in [1], that saturation is a feature of that large-d limit rather than of real turbulence, and that the logarithmic form is the relevant description at physically accessible dimensionalities.

It should also be noted that the model studied here uses cyclic rather than fixed boundary conditions and a discrete rather than continuous vortex-size distribution. The fact that the results in this thesis at hand follow the logarithmic prediction despite these differences supports the robustness of the model with respect to implementation details, as anticipated in [1].

Acknowledgements

The author acknowledges the supervision, contributions and support of Jaan Kalda in the creation of this thesis. Professor Kalda's guidance and interpretations of results were instrumental in developing the work at hand. The author also thanks his mother for proofreading.

Additionally, it is noted here that Claude AI was used for both language and code correction purposes.

Bibliography

[1] J. Kalda and A. Morozenko. “Turbulent mixing: the roots of intermittency”. In: *New Journal of Physics* 10.9 (2008). DOI: 10.1088/1367-2630/10/9/093003.

[2] A. N. Kolmogorov. “The local structure of turbulence in incompressible viscous fluid for very large Reynolds numbers”. In: *Doklady Akademii Nauk SSSR* 30 (1941), pp. 301–305.

[3] A. N. Kolmogorov. “A refinement of previous hypotheses concerning the local structure of turbulence in a viscous incompressible fluid at high Reynolds number”. In: *Journal of Fluid Mechanics* 13 (1962), pp. 82–85. DOI: 10.1017/S0022112062000518.

[4] A. M. Obukhov. “Structure of the temperature field in turbulent flows”. In: *Izvestiya Akademii Nauk SSSR* 13 (1949), pp. 58–69.

[5] S. Corrsin. “On the spectrum of isotropic temperature fluctuations in an isotropic turbulence”. In: *Journal of Applied Physics* 22 (1951), pp. 469–473. DOI: 10.1063/1.1699986.

[6] E. Balkovsky and V. Lebedev. “Instanton for the Kraichnan passive scalar problem”. In: *Physical Review E* 58.5 (1998), pp. 5776–5795. DOI: 10.1103/PhysRevE.58.5776.

[7] Antonio Celani et al. “Fronts in passive scalar turbulence”. In: *Physics of Fluids* 13.6 (2001), pp. 1768–1783. DOI: 10.1063/1.1367325.

[8] Robert H. Kraichnan. “Anomalous Scaling of a Randomly Advected Passive Scalar”. In: *Physical Review Letters* 72.7 (1994), pp. 1016–1019. DOI: 10.1103/PhysRevLett.72.1016.

[9] R. A. Antonia et al. “Temperature structure functions in turbulent shear flows”. In: *Physical Review A* 30.5 (1984), pp. 2704–2707. DOI: 10.1103/PhysRevA.30.2704.

Appendices

Appendix 1 – Non-Exclusive License for Reproduction and Publication of a Graduation Thesis

I, Karl Kukk

1. Grant Tallinn University of Technology free licence (non-exclusive licence) for my thesis *"Modelling Turbulent Mixing Using A One-Dimensional Model"*,
   supervised by Jaan Kalda
   (a) to be reproduced for the purposes of preservation and electronic publication of the graduation thesis, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright;
   (b) to be published via the web of Tallinn University of Technology, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright.
2. I am aware that the author also retains the rights specified in clause 1 of the non-exclusive licence.
3. I confirm that granting the non-exclusive licence does not infringe other persons' intellectual property rights, the rights arising from the Personal Data Protection Act or rights arising from other legislation.

25.05.2026

Appendix 2 – Used code

```
#include <iostream>
#include <vector>
#include <cmath>
#include <random>
#include <fstream>
#include <algorithm>
#include <chrono>
#include <omp.h>

#include "transform.h"
#include "vortexlength.h"
#include "findyorig.h"
#include "uuedloigud.h"
#include "loigumuutmine2.h"

using clk = std::chrono::high_resolution_clock;
using ms = std::chrono::duration<double, std::milli>;

void log_runtime(const std::string& filename, int iteration, double
    elapsed_sec)
{
   std::ofstream f(filename, std::ios::app);
   if (f.is_open())
      f << iteration << "\t" << elapsed_sec << "\n";
}

std::vector<int> find_symmetric_slice(int x0, int d, int step = 1)
{
   int left = x0 - d / 2 + 1;
   int right = x0 + d / 2;
   std::vector<int> xs;
   xs.reserve(d);
   for (int xi = left; xi <= right; xi += step)
      xs.push_back(xi);
   return xs;
}

void save_to_file(const std::string& name, const std::vector<double>& v)
{
   std::ofstream f(name);
   for (double val : v)
      f << static_cast<int>(val) << "\n";
}
```

```
void compute_and_save_sp(
   const std::vector<double>& y,
   int x0,
   int chain_id,
   int iteratsioon,
   int loigupikkus)
{
   const std::vector<int> lags = { 1, 2, 3, 6, 8, 12, 24, 32, 48, 64,
      96, 128, 256, 384, 512,
                           768, 1024, 1536, 2048, 3072, 4096, 6144,
                              8192, 12288, 16384, 24576, 32768};
   const std::vector<int> orders = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11,
      12, 14, 16, 18, 20, 22, 24 };
   const int x1 = x0 + loigupikkus;

   for (int pi = 0; pi < (int)orders.size(); ++pi)
   {
      int p = orders[pi];
      std::string fname = "sp" + std::to_string(p)
         + "_chain" + std::to_string(chain_id)
         + ".txt";
      std::ofstream f(fname, std::ios::app);

      f << iteratsioon;
      for (int a : lags)
      {
         if (x0 + a >= x1) break;
         double S = 0.0;
         int count = 0;
         for (int xi = x0; xi + a < x1; ++xi)
         {
            double diff = y[xi] - y[xi + a];
            double val = diff;
            for (int k = 1; k < p; ++k) val *= diff;
            S += val;
            ++count;
         }
         f << "\t" << S / count;
      }
      f << "\n";
   }
}

std::vector<double> run_chain(
   int chain_id,
   const std::vector<int>& x,
   const std::vector<int>& vortexlengths,
```

```
    const std::vector<double>& weights,
    int loigupikkus,
    int iterations,
    std::chrono::time_point<clk> timer_start)
{
    std::mt19937 rng(std::random_device{}() ^ (uint32_t)(chain_id *
        2654435761u));
    std::discrete_distribution<> vlen_dist(weights.begin(),
        weights.end());
    std::uniform_int_distribution<int> pos_dist(0, loigupikkus);

    // Initial condition: y = -x
    std::vector<double> y_shifted(x.size());
    for (size_t i = 0; i < x.size(); ++i)
        y_shifted[i] = -x[i];

    // Find x0 once before the main loop
    int x0 = 0;
    while (x0 < (int)x.size() && x[x0] != 0) ++x0;

    const int sp_interval = loigupikkus; // 40960

    for (int iter = 0; iter < iterations; ++iter)
    {
        int vlen = vortexlengths[vlen_dist(rng)];
        int vpos = pos_dist(rng);

        auto slice_points = find_symmetric_slice(vpos, vlen);
        auto y_slice = fetch_y_coordinates(x, slice_points, y_shifted);
        auto centered = keskmestamine(y_slice);
        auto new_y = new_slice_creation(centered);
        loigumuutmine2(x, y_shifted, slice_points, new_y, loigupikkus);

        int iteratsioon = iter + 1;

        if (iteratsioon % sp_interval == 0)
        {
            compute_and_save_sp(y_shifted, x0, chain_id, iteratsioon,
                loigupikkus);

            if (chain_id == 0)
            {
                auto now = clk::now();
                std::chrono::duration<double> elapsed = now - timer_start;
                log_runtime("runtime.txt", iteratsioon, elapsed.count());
            }
        }
    }
```

```
    return y_shifted;
}

int main()
{
    auto timer_start = clk::now();

    const int loigupikkus = 40960;
    const int iterations = 40960*41000;
    const int num_chains = 20;

    std::cout << "Running " << num_chains << " chains on " << num_chains
        << " threads\n";

    // Build x vector
    std::vector<int> x;
    for (int i = -loigupikkus / 2 - 1; i <= (int)(loigupikkus * 1.5) + 1;
        ++i)
        x.push_back(i);

    // Vortex lengths
    std::vector<int> vortexlengths;
    for (int n = 1; ; n++) {
        int v = 3 * (1 << n) + 2;
        if (v >= loigupikkus) break;
        vortexlengths.push_back(v);
    }

    // Weights
    std::vector<double> weights;
    for (double v : vortexlengths)
        weights.push_back(std::pow(v, -5.0 / 3.0));

    // Storage for each chain's final result
    std::vector<std::vector<double>> chain_results(num_chains);

    // Launch all chains in parallel, one per thread
#pragma omp parallel for num_threads(num_chains) schedule(static, 1)
    for (int c = 0; c < num_chains; ++c)
    {
        chain_results[c] = run_chain(c, x, vortexlengths, weights,
            loigupikkus, iterations, timer_start);
    }

    // Average all chains for ensemble result
    std::vector<double> averaged(x.size(), 0.0);
    for (int c = 0; c < num_chains; ++c)
```

```
            for (size_t i = 0; i < x.size(); ++i)
                averaged[i] += chain_results[c][i];
        for (size_t i = 0; i < x.size(); ++i)
            averaged[i] /= num_chains;

        save_to_file("final_averaged.txt", averaged);

        // Save each chain's final state individually
        for (int c = 0; c < num_chains; ++c)
            save_to_file("final_chain" + std::to_string(c) + ".txt",
                chain_results[c]);

        auto now = clk::now();
        std::chrono::duration<double> elapsed = now - timer_start;
        log_runtime("runtime.txt", iterations, elapsed.count());

        std::cout << "Done in " << elapsed.count() << "s\n";
        std::cout << "Equivalent single-chain iterations: " << (long
            long)num_chains * iterations << "\n";

        return 0;
}

// g++ -O3 -march=native -ffast-math -fopenmp main.cpp transform.cpp
    vortexlength.cpp findyorig.cpp uuedloigud.cpp loigumuutmine2.cpp -o
    simulation
```

```
#pragma once
#include <vector>

// For each x in x_array, interpolate the corresponding y from
    (x_full_array, y_full_array)
std::vector<double> fetch_y_coordinates(
    const std::vector<int>& x_full_array,
    const std::vector<int>& x_array,
    const std::vector<double>& y_full_array
);
```

```
#include "findyorig.h"
#include <vector>
#include <algorithm> // for std::lower_bound

std::vector<double> fetch_y_coordinates(
    const std::vector<int>& x_full_array,
    const std::vector<int>& x_array,
    const std::vector<double>& y_full_array
)
{
    std::vector<double> y_result;
```

```
    y_result.reserve(x_array.size());

    for (auto x : x_array)
    {
        // Find the interval [x0, x1] where x0 <= x <= x1
        auto it = std::lower_bound(x_full_array.begin(),
            x_full_array.end(), x);

        if (it == x_full_array.begin())
        {
            y_result.push_back(y_full_array.front());
        }
        else if (it == x_full_array.end())
        {
            y_result.push_back(y_full_array.back());
        }
        else
        {
            size_t idx1 = std::distance(x_full_array.begin(), it);
            size_t idx0 = idx1 - 1;

            double x0 = x_full_array[idx0];
            double x1 = x_full_array[idx1];
            double y0 = y_full_array[idx0];
            double y1 = y_full_array[idx1];

            // Linear interpolation formula
            double y = y0 + (y1 - y0) * (x - x0) / (x1 - x0);
            y_result.push_back(y);
        }
    }

    return y_result;
}

#pragma once
#include <vector>

void loigumuutmine2(
    const std::vector<int>& x,
    std::vector<double>& y,
    const std::vector<int>& x_pos,
    const std::vector<double>& y_uus,
    int loigupikkus
);

#include <vector>
#include <cmath>
#include <algorithm>
```

```cpp
#include "loigumuutmine2.h"

void loigumuutmine2(
   const std::vector<int>& x,
   std::vector<double>& y,
   const std::vector<int>& x_pos,
   const std::vector<double>& y_uus,
   int loigupikkus
) {
   const int start = loigupikkus / 2;
   const int end = (int)(loigupikkus * 1.5);
   const int mid = (start + end) / 2;
   const int x_offset = x[0];

   const int k_max = (int)std::min(x_pos.size(), y_uus.size());
   for (int k = 0; k < k_max; ++k) {
      int idx = x_pos[k] - x_offset;
      if (idx >= start && idx < end)
         y[idx] = y_uus[k];
   }

   // Left section: read from mid..end, write to 0..start (no overlap)
   for (int i = 0; i < start; ++i)
      y[i] = y[mid + i] + loigupikkus;

   // Right section: read from start..mid, write to end.. (no overlap)
   int right_len = (int)y.size() - end;
   for (int i = 0; i < right_len; ++i)
      y[end + i] = y[start + i] - loigupikkus;
}
```

```cpp
#pragma once
#include <vector>

// Compute the "triplet-centered" mean as in Python's keskmestamine
std::vector<double> keskmestamine(const std::vector<double>&
   y_koordinaatide_array);
```

```cpp
#include "transform.h"
#include <vector>
#include <numeric> // for std::accumulate

std::vector<double> keskmestamine(const std::vector<double>& y)
{
   size_t n = y.size();
   size_t triplets_count = n / 3;

   std::vector<double> result(triplets_count);
```

```
    for (size_t i = 0; i < triplets_count; ++i)
    {
        // sum elements y[i*3 + 1], y[i*3 + 2], y[i*3 + 3]
        double sum = 0.0;
        for (size_t j = 1; j <= 3; ++j)
        {
            sum += y[i * 3 + j];
        }
        result[i] = sum / 3.0; // mean
    }

    // Create new vector with first element prepended and last element
        appended
    std::vector<double> newer_result;
    newer_result.reserve(result.size() + 2);
    newer_result.push_back(y[0]);
    newer_result.insert(newer_result.end(), result.begin(), result.end());
    newer_result.push_back(y[n - 1]);

    return newer_result;
}

#pragma once
#include <vector>

// Create a new slice by mirroring the middle part of the input array
std::vector<double> new_slice_creation(const std::vector<double>&
    allycoordinates);

#include "uuedloigud.h"
#include <vector>
#include <algorithm> // for std::reverse_copy

std::vector<double> new_slice_creation(const std::vector<double>&
    allycoordinates)
{
    if (allycoordinates.size() < 3)
        return allycoordinates; // not enough elements to mirror

    // first element
    std::vector<double> first(allycoordinates.begin(),
        allycoordinates.begin() + 1);
    // last element
    std::vector<double> last(allycoordinates.end() - 1,
        allycoordinates.end());
    // middle elements
    std::vector<double> middle(allycoordinates.begin() + 1,
        allycoordinates.end() - 1);
```

```
    // mirrored middle
    std::vector<double> mirrored(middle.size());
    std::reverse_copy(middle.begin(), middle.end(), mirrored.begin());

    // concatenate: first + middle + mirrored + middle + last
    std::vector<double> new_slice;
    new_slice.reserve(first.size() + middle.size() + mirrored.size() +
        middle.size() + last.size());

    new_slice.insert(new_slice.end(), first.begin(), first.end());
    new_slice.insert(new_slice.end(), middle.begin(), middle.end());
    new_slice.insert(new_slice.end(), mirrored.begin(), mirrored.end());
    new_slice.insert(new_slice.end(), middle.begin(), middle.end());
    new_slice.insert(new_slice.end(), last.begin(), last.end());

    return new_slice;
}
```

```
#pragma once
#include <vector>

std::vector<int> random_vortex_position(
    const std::vector<int>& vortexlengths,
    int loigupikkus
);
```

```
#include "vortexlength.h"
#include <random>
#include <vector>

std::vector<int> random_vortex_position(
    const std::vector<int>& vortexlengths,
    int loigupikkus
) {
    std::mt19937 rng(std::random_device{}());
    std::uniform_int_distribution<int> dist(0, loigupikkus); // inclusive
        0 and loigupikkus
    std::vector<int> positions;

    for (size_t i = 0; i < vortexlengths.size(); ++i) {
        positions.push_back(dist(rng));
    }

    return positions;
}
```