GPU-ACCELERATED TURBULENCE SIMULATOR FOR SPACE-BASED OPTICAL COMMUNICATIONS

GPU-ACCELERATED MULTI-LAYER TURBULENCE PROPAGATOR FOR SPACE-BASED OPTICAL COMMUNICATIONS

By HAMED AKHLAGHI, BS

A Thesis Submitted to the School of Graduate Studies in Partial Fulfillment of the Requirements for the Degree Master of Applied Science

McMaster University

MASTER OF APPLIED SCIENCE (2025)

Hamilton, Ontario, Canada (Electrical and Computer Engineering)

TITLE:	GPU-Accelerated Multi-Layer Turbulence Propagator for Space-Based Optical Communications
AUTHOR:	Hamed Akhlaghi BS (Electrical Engineering), Amirkabir University of Technology, Tehran, Iran
SUPERVISORS:	Steve Hranilovic, Michael Taylor
NUMBER OF PAGES:	xxv, 133

Lay Abstract

Optical satellite communications (SatCom) provides high data throughput and low latency, particularly for low Earth orbit (LEO) satellite applications over distances ranging from 500 to 1,500 km, making them essential for emerging space-based networks. However, due to wind and temperature shifts, atmospheric turbulence severely impacts the quality of downlink channels, necessitating robust simulation tools to inform system design and ensure reliable performance under dynamic conditions.

Numerical simulation of optical propagation through turbulence with high spatial sampling poses significant computational challenges, especially for rapidly moving LEO satellites and simulations with a number of phase screen layers. This high-resolution sampling is essential for accurately predicting how atmospheric turbulence will affect signal quality, which directly impacts the reliability and performance of these critical communications links. This work addresses this challenge with a Graphic Processing Unit (GPU) architecture that parallelizes intensive computations and large loops across GPU cores. This advancement enables the use of large phase screens, many layers, and rapid propagation loops, efficiently simulating fast-translating LEO satellites. This approach significantly enhances the speed of atmospheric turbulence simulations for satellite communications, offering a powerful tool for system design, performance prediction, and optimization of adaptive optics strategies in free-space

optical communication systems.

The simulator employs a multi-layered approach to optical propagation, accounting for varying wind speeds and satellite elevation angles to create a comprehensive path simulation of wavefront degradation. The speed and accuracy of the simulator also make it ideal for machine learning applications, generating large datasets of realistic turbulent wavefronts matching atmospheric theory. Validation results demonstrate excellent agreement between theoretical predictions and simulated outputs across various atmospheric conditions and satellite positions. Performance benchmarks show that GPU-accelerated implementation achieves up to 600 times faster execution time within the same conditions compared to a publicly available optical turbulence simulator intended for astronomy, while maintaining equivalent accuracy. This simulator introduces a GPU-accelerated, multi-layer architecture that captures both fast orbital motion and fine-scale atmospheric distortion.

Abstract

In modern space-based networks, growing bandwidth demands and increasing mission complexity drive the need for accurate, high-resolution simulations that capture small-scale turbulence features and rapid satellite movement. By leveraging GPU parallelization, this simulator manages large data volumes and frequent time steps, enabling the modeling of wavefront distortions essential for robust system design, adaptive optics, and performance optimization. This thesis presents a novel, high-speed simulator for optical propagation through dynamic atmospheric turbulence affecting satellite downlinks. The method begins by calculating the effective refractive index structure parameter (C_n^2) which captures the turbulence strength, along the observation path between the satellite and ground receiver. This derived C_n^2 informs the atmospheric slicer, which distributes phase screens throughout the propagation path. The simulation focuses on the first 20 km of atmosphere, where the majority of turbulence affecting free-space optical links occurs. The angular spectrum propagation formula is implemented to achieve Fresnel propagation between planes where phase screens represent integrated turbulence slices at specific altitudes. Temporal evolution is achieved via the frozen flow hypothesis and an adjustable wind model with altitude-dependent wind speeds. Numerical simulation of optical propagation

through turbulence with high spatial sampling poses significant computational challenges, especially for rapidly moving Low Earth Orbit (LEO) satellites and simulations with numerous phase screen layers. This work addresses this challenge with an innovative GPU architecture that parallelizes intensive computations and large loops across GPU cores. This advancement enables the use of large phase screens, many layers, and rapid propagation loops, efficiently simulating fast-translating LEO satellites. This comprehensive approach significantly enhances the speed of atmospheric turbulence simulations for satellite communications, offering a powerful tool for system design, performance prediction, and optimization of adaptive optics strategies in free-space optical communication systems. The GPU-accelerated implementation achieves speedup factors of 310× to 600× over conventional CPU-based simulators.

To my grandfather,

who saw me begin this journey

but couldn't stay to see its completion.

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my supervisor, Dr. Steve Hranilovic, who has been much more than just a supervisor throughout this journey. His unwavering support, patience, and profound insights have shaped not only this research but my entire academic perspective. I can think of no better way to describe his role than as my academic father—guiding me through challenges, celebrating achievements, and always encouraging me to push the boundaries of what I thought possible.

I am equally indebted to my co-supervisor, Dr. Michael Taylor, whose deep understanding of free space optics illuminated my path during countless moments of uncertainty. His willingness to share his time and resources without any reservations has been truly remarkable. This work would not have been possible without his expertise and thorough help at every step of the process. His meticulous approach and technical rigor have set a standard I will continue to aspire toward.

To my parents, who have been my foundation since day one, I owe a debt of gratitude that words can scarcely express. Their unconditional love and sacrifices have enabled me to pursue my dreams without hesitation. From late-night calls during stressful periods to their unwavering belief in my abilities even when I doubted myself, they have been my constant source of strength. Their lessons in perseverance

and integrity have guided me through the most challenging phases of this research.

I would be remiss not to acknowledge my friends Amirhossein and Amir for their brotherhood, support, and friendship. They have been my sounding board, my escape from academic pressures, and my companions through both celebrations and setbacks. Their humor, encouragement, and perspective have kept me grounded and reminded me of the importance of balance in life and work. To my long-distance friend Hossein, who despite the physical separation has remained an integral part of my support system. His ability to provide clarity from afar and his consistent check-ins have been a reminder that true friendship transcends distance.

I would also like to express my sincere appreciation to my colleagues Parham Kazemi, Hamed Hosseininejad, Alireza Barmaki, and Mohamadreza Pashazanoosi. Their camaraderie, technical insights, and collaborative spirit have significantly enhanced my research experience. Our discussions, whether during late-night lab sessions or coffee breaks, have sparked valuable ideas and solutions to challenging problems. Their diverse perspectives and expertise have been instrumental in broadening my understanding of our field.

I am particularly grateful to Dr. Antony Orth and the National Research Council of Canada (NRC) for their invaluable support and assistance throughout this research. The resources and collaborative opportunities provided by the NRC have significantly expanded the scope and impact of this work. Their commitment to advancing scientific research in Canada has been truly inspiring.

Finally, I extend my appreciation to all the faculty members, administrative staff, and colleagues who have contributed to creating an environment conducive to learning and discovery. The collaborative spirit of our department has enriched my experience

immeasurably.

This journey would not have been the same without each of you.

Table of Contents

Lay Abstract **iii**

Abstract **v**

Acknowledgements **viii**

Notation, Definitions, and Abbreviations **xxiii**

Declaration of Academic Achievement **xxvi**

1 Introduction **1**

1.1 Atmospheric Turbulence and Channel Geometry 4

1.2 Literature Review 7

1.3 Thesis Contributions 13

1.4 Thesis Organization 14

2 Background on Turbulence Modeling and Simulation **15**

2.1 Atmospheric Turbulence 15

2.2 Optical Propagation Theory 24

2.3 Satellite Downlink Geometry 58

2.4 Atmospheric Layer Slicer 61
2.5 Numerical Propagation Simulation Workflow 64
2.6 Computational Challenges and Scale 68
2.7 Summary 69

3 GPU Implementation **71**
3.1 GPU Architecture and Rationale for TensorFlow Integration 71
3.2 System Architecture 79
3.3 Simulation Engine Components 83
3.4 Numerical Stability Constraints Investigation 88
3.5 Conclusion 90

4 Simulation Results and Validation **92**
4.1 Validation of Numerical Implementation 92
4.2 Computational Performance Analysis 101
4.3 GPU Simulator Outputs 104
4.4 GPU Memory and Scaling Results 107
4.5 Conclusion 117

5 Conclusion **119**

List of Figures

1.1 A basic free-space optical communication system. 5

1.2 FSO in space communications. 6

2.1 Richardson energy cascade depicting the breakdown of turbulent eddies from the integral scale l_0 to the Kolmogorov microscale η. (adapted from [1]). 17

2.2 Kolmogorov energy spectrum $E(k)$ (2.1.1) illustrating the inertial subrange with a $k^{-5/3}$ scaling (blue line). The energy-containing range (gray shaded region) and dissipation range (red shaded region, $k \gg \eta^{-1}$) are marked. The integral scale L and Kolmogorov microscale η are denoted by vertical dashed lines. 18

2.3 Comparison of the von Kármán (red solid line) and Kolmogorov (blue dashed line) spectra for refractive index fluctuations. The von Kármán spectrum transitions from $\kappa^{-11/3}$ behavior in the inertial subrange to finite energy-containing ($\kappa \leq \kappa_0$, gray shaded) and dissipation ($\kappa \geq \kappa_m$, orange shaded) ranges. Vertical dashed lines mark $\kappa_0 = 2\pi/L_0$ (outer scale cutoff) and $\kappa_m = 5.92/l_0$ (inner scale cutoff). 21

2.4 Atmospheric turbulence effects characterized by the Fried parameter (r_0): (a) Wavefront with turbulence-induced phase distortions showing a small aperture ($D < r_0$, $\sigma_\phi \sim 1$ radian) and large aperture ($D >> r_0$, $\sigma_\phi \gg 1$ radian). Color scale represents phase in radians. (b) Fried parameter wavelength dependence following $r_0 \propto \lambda^{6/5}$, with common astronomical bands (U: 365 nm, V: 550 nm, I: 806 nm, H: 1.6 μm, K: 2.2 μm) marked. (c) Focus spot for the small aperture ($D < r_0$), simulation shows the focal plane intensity profile, with less turbulence distortion over the aperture and maintaining a more coherent spot. (d) Focus spot for the large aperture ($D >> r_0$), simulation shows the focal plane intensity profile, exhibiting significant speckle structure due to stronger turbulence. This spot will produce seeing-limited (λ/r_0) resolution. 32

2.5 Scintillation index σ_I^2 versus Rytov variance β_0^2 in log-log scale, showing the distinct turbulence regimes. In the weak fluctuation regime ($\beta_0^2 < 1$), Rytov theory accurately predicts the linear relationship $\sigma_I^2 \propto \beta_0^2$ (blue solid line). As turbulence strengthens ($\beta_0^2 > 1$), nonlinear effects cause saturation (red dashed line), where σ_I^2 approaches an asymptotic value despite increasing β_0^2. The transition region near $\sigma_I^2 \approx 1$ marks the breakdown of Rytov theory and the onset of strong turbulence effects. 38

2.6 Visualizing Fried parameter r_0 (blue, left axis) and scintillation index σ_I^2 (red, right axis) versus turbulence strength C_n^2 for a wavelength of 1550 nm and 1 km propagation path. As C_n^2 increases, r_0 decreases following a $-3/5$ power law (worse wavefront coherence), while σ_I^2 rises (stronger intensity fluctuations) before reaching saturation. The vertical dashed line marks typical daytime turbulence ($C_n^2 = 10^{-13}$ m$^{-2/3}$), where $r_0 \approx 5$ cm and significant scintillation occurs. Note the opposite behavior of these parameters, highlighting the challenge of maintaining both phase coherence and intensity stability in strong turbulence conditions. 39

2.7 Decomposition of a von Karman atmospheric phase screen showing (a) low-frequency components from subharmonic compensation, (b) high-frequency components from the Fourier-based method, and (c) the combined phase screen. The subharmonic compensation adds the necessary large-scale turbulent structures that are underrepresented in pure Fourier methods. Phase values are shown in radians. 49

2.8 Fried column extrusion method for temporal evolution of phase screens. A complete phase screen (left) serves as the initial state, from which new columns are sequentially generated (right) in steps 1-4. Each new column maintains statistical correlation with the existing phase structure while extending the screen in the direction of evolution. . . 54

2.9 Split-step propagation through multiple phase screens. The optical field from a point source propagates through a series of phase screens (colored planes) representing atmospheric turbulence layers. Between screens, Fresnel propagation is applied over distances Δz_1, Δz_2, and Δz_3. The wavefront is progressively distorted as it encounters each turbulent layer. 57

2.10 Satellite slant path geometry: H = higher orbit altitude, ζ = zenith angle, θ = elevation angle. Turbulence strength $C_n^2(h)$ varies along the path (color gradient). The effective propagation path (dashed line) scales as $L = \sec(\zeta) \cdot (H - h_0)$, where h_0 is observer altitude. 59

2.11 Illustration of layer slicer output with layer discretization (red dashed lines). This is achieved by solving Equation (2.4.3) and finding the layer heights. The layers are labeled as introduced in Equation 2.4.3 . 63

2.12 Comprehensive atmospheric turbulence simulation pipeline. The process begins with initialization parameters including the $C_n^2(h)$ profile and characteristic scales. Path integration yields the Fried parameter r_0 and spectral cutoffs κ_m, κ_0. Phase screen generation follows the von Kármán spectrum with proper subharmonic compensation for low-frequency components. Temporal evolution uses Fried's extrusion method, computing covariance matrices to generate new columns while preserving proper statistics. The split-step propagation applies phase screens and uses the angular spectrum method to simulate wave propagation through turbulence. This process iterates for each time step, creating a dynamic simulation of atmospheric effects on optical propagation. 65

2.13 Conceptual illustration of a satellite downlink channel passing through multiple layers of atmospheric turbulence. Each colored layer represents a two-dimensional turbulence phase screen at a different altitude (with altitude increasing from bottom to top). As the satellite moves (indicated by the "Satellite Movement" arrow), new turbulence realizations can be introduced or "extruded" into the simulation at each layer ("Layer Extrusion" arrows), thereby capturing the continuous evolution of atmospheric turbulence in time. 66

3.1 A generic modern GPU architecture showing the hierarchical organization with multiple SIMT Core Clusters, each containing SIMT Cores that execute parallel threads. The Interconnection Network facilitates communication between the computational units and memory partitions, which interface with high-bandwidth GDDR memory (adapted from [2]). 72

3.2 Block diagram of the atmospheric turbulence simulation architecture showing CPU and GPU domains. The CPU domain (top) handles initialization and parameter computation, while the GPU domain (bottom) manages parallel phase screen evolution and wave propagation. Dashed lines indicate domain boundaries, with arrows showing data flow direction. 81

4.1 Altitude distribution of the five atmospheric layers overlaid on the $C_n^2(h)$ profile. Layer boundaries are positioned at 1.45, 6.26, 10.08, 12.90, and 20.02 km. The color intensity represents the relative turbulence strength within each layer, ranging from 1.03×10^{-16} $\mathrm{m}^{-2/3}$ near the ground to 9.00×10^{-19} $\mathrm{m}^{-2/3}$ at the highest layer. 93

4.2 Comparison of theoretical and observed scintillation indices versus Fried parameter values. The blue line represents theoretical predictions from equation (2.2.25), while red markers with error bars show observed measurements. The error bars represent the 10-90 percentile range derived from 1000 distinct propagation simulations, with the marker indicating the mean values. Close agreement is observed for $r_0 \geq 15$ cm, with increasing deviation at smaller r_0 values corresponding to stronger turbulence conditions. 96

4.3 Mutual Coherence Function analysis for finding r_0 values. The solid lines show theoretical r0 curves, while markers indicate measured values from simulation. Note that the limit of small L0 can possibly lead to over-estimation of r0 values. 98

4.4 Execution time comparison between HCIPy and GPU-accelerated implementation for different grid sizes. Both simulations performed 500 iterations with identical atmospheric conditions and propagation parameters. Note the logarithmic scale on the time axis. 103

4.5 Time evolution of the optical field at the receiver plane after propagation through 10 atmospheric layers, where each step is 1 ms and each wavefront covers a 10.24 m × 10.24 m area. Left column: Phase distribution (angle of U_{out}) in radians. Right column: Intensity distribution (absolute value of U_{out}). Rows represents three consecutive steps, demonstrating the temporal evolution of turbulence-induced distortions. The phase screens show the accumulated wavefront distortion, while the intensity patterns reveal characteristic speckle structures resulting from atmospheric turbulence. Simulation parameters as specified in Table 4.5. 106

4.6 GPU memory utilization for different grid sizes and layer counts on an NVIDIA RTX 3070 GPU. While the RTX 3070 has 8GB of VRAM, only 6GB is available to TensorFlow as approximately 2GB is reserved by the operating system. For all tested configurations, TensorFlow allocates nearly the entire available memory capacity (5850-5980 MB or approximately 97.5-99.7% of the available 6GB), regardless of grid size or layer count. 108

4.7 Execution time comparison of the GPU-accelerated implementation across different grid sizes and layer counts. The execution time scales with increasing grid dimensions, showing expected computational complexity growth as the problem size increases. 111

4.8 GPU utilization by grid size and layer count. The average utilization remains relatively consistent across different configurations, ranging from 31.28% to 35.31%, while peak utilization reaches 41-44%. This moderate utilization pattern indicates that memory bandwidth and other system constraints, rather than raw computational throughput, are likely the limiting factors in this implementation. 113

4.9 Execution time comparison between the first run and subsequent runs of the simulation on the GPU for the 1024×1024 grid with 10 layers configuration. The first propagation requires 0.266 seconds while subsequent propagations average 0.191 seconds, representing a 39.5% performance penalty for the initial run due to GPU-specific initialization steps. 115

List of Tables

4.1 Simulation Parameters 93
4.2 Layer Distribution and Parameters 94
4.3 C_n^2 Values for Each Atmospheric Layer 95
4.4 Comparison of Input and MCF-Derived Fried Parameters 100
4.5 GPU Simulator Configuration Parameters 105

Notation, Definitions, and Abbreviations

Notation

$E(k)$	Turbulent energy spectrum as a function of spatial wavenumber k
ϵ	Energy dissipation rate in turbulent flows
ν	Kinematic viscosity of the fluid
C_K	Kolmogorov constant
L_0, l_0	Outer and inner scales of turbulence
η	Kolmogorov microscale
k	Optical wavenumber, $k = 2\pi/\lambda$
λ	Optical wavelength
$n(\mathbf{r})$	Refractive index field
C_n^2	Refractive index structure parameter (units $\mathrm{m}^{-2/3}$)

$D_n(r)$	Refractive index structure function
$\Phi_n(\kappa)$	PSD of refractive index fluctuations
κ	Spatial wavenumber variable
r_0	Fried parameter (coherence diameter)
σ_I^2	Scintillation index
β_0^2	Rytov variance
$U(\mathbf{r})$	Complex field amplitude in time-harmonic form
∇^2	Laplacian operator
$\nabla_\perp^2$	Transverse Laplacian
$\mathcal{F}, \mathcal{F}^{-1}$	Fourier transform and its inverse
$\mathbf{r} = (x, y, z)$	Spatial coordinate vector

Abbreviations

AO	Adaptive Optics
CPU	Central Processing Unit
FFT	Fast Fourier Transform
FSO	Free-Space Optical
GPU	Graphics Processing Unit

HCIPy	High Contrast Imaging for Python
HV	Hufnagel-Valley (turbulence model)
LEO	Low Earth Orbit
MCF	Mutual Coherence Function
PSD	Power Spectral Density
SSFM	Split-Step Fourier Method
XLA	Accelerated Linear Algebra (TensorFlow compiler)
IMDD	Intensity-Modulated Direct-Detection

Declaration of Academic Achievement

I, Hamed Akhlaghi, hereby declare that the research presented in this thesis is my own original work except where explicitly indicated by appropriate acknowledgments. I further confirm that any ideas, data, or contributions that are not my own have been fully credited. In particular, I wish to acknowledge the invaluable guidance and contributions of my supervisor, Dr. Steve Hranilovic; my co-supervisor, Dr. Michael Taylor; and the research support provided by Dr. Antony Orth, a collaborator from the National Research Council (NRC).

Signature: _____ Date: _____

Chapter 1

Introduction

Free-space optical (FSO) communication uses visible or infrared wavelengths to transmit data through free-space—either within the Earth's atmosphere or across interplanetary distances. Compared to conventional radio-frequency (RF) systems, FSO provides a substantially larger usable bandwidth, higher data rates, and stronger security due to the narrow, highly directional nature of laser beams [3–5]. Early terrestrial deployments have seen success in high-capacity "last-mile" links between buildings or within campus environments offering gigabit connectivity without the complications of wired infrastructure. Space agencies have extended FSO into satellite communications, exemplified by NASA's Lunar Laser Communication Demonstration (LLCD), which achieved a 622 Mbps downlink from the Moon [6], and the Laser Communications Relay Demonstration (LCRD), aimed at high-speed optical links for near-Earth and deep-space missions [7]. Other international endeavors, such as ESA's Semiconductor Inter-satellite Link Experiment (SILEX) and JAXA's Optical Inter-orbit Communications Engineering Test Satellite (OICETS), have further contributed to advancement for space-based optical communications [8]. The commercial sector has

likewise made significant investments in this technology, with SpaceX deploying inter-satellite optical communication links within its Starlink constellation, while the US Government advances space-to-ground optical connectivity through the Space Development Agency's (SDA) standardization of Optical Communications Terminals.

Recent developments highlight the growing importance of Optical Inter-Satellite Links (OISLs) for high-capacity data routing in large constellations. For example, SpaceX [9] has demonstrated operational OISLs within its Starlink network, and additional mega-constellations such as Amazon's Project Kuiper [10] and Telesat's Lightspeed [11] plan to rely on optical links for inter-satellite connectivity. Meanwhile, high-speed feeder links to the ground have shown rapid progress, with NASA's TeraByte InfraRed Delivery (TBIRD) mission achieving a record 200 Gbps downlink from LEO to Earth in 2022 [12]. Such feats underscore how feeder links, which supply high-data-rate connections from satellites to terrestrial ground stations, increasingly need optical bands to handle terabit-scale traffic volumes—outpacing traditional RF systems [9]. Advanced receiver technologies such as optically pre-amplified and coherent detection further enable these high-capacity links by improving spectral efficiency. In simpler intensity-modulated direct-detection (IMDD) systems, large-aperture receivers can collect enough power without needing to resolve the optical phase, making them more tolerant to turbulence. However, efficient detection for higher data rates requires the use of coherent detection or optically pre-amplified direct detection [13] which use single mode fiber receivers. Efficient coupling of the turbulent wavefront into a single mode fiber requires adaptive optics correction to mitigate turbulence-induced distortions [14].

FSO-based links enable both inter-satellite data transfer (OISLs) and high-throughput "feeder" links between satellites and Earth, providing backhaul capacity for next-generation broadband services. Starlink's OISLs, for instance, reduce latency by routing data directly between satellites, while continuing to use RF ground links as feeders. Emerging architectures, however, are moving toward entirely optical-based feeder links to better accommodate surging bandwidth demands. Optical frequencies offer not only higher throughput but also smaller terminal apertures and inherent resistance to electromagnetic interference, making them attractive for future large-scale constellations.

Despite the benefits, real-world performance of Low Earth Orbit (LEO) downlink FSO systems can be limited by atmospheric impairments. Light traversing the lower layers of Earth's atmosphere encounters varying pressure, temperature, and particle densities, leading to absorption, scattering, and refractive-index fluctuations [15, 16]. These distortions are especially critical in LEO satellite downlinks, where spacecraft moving at roughly 7–8 km/s experience a rapidly changing geometric path. Misalignment from narrow beam divergence, turbulence-induced scintillation, and other losses combine to complicate link design and impose demanding computational requirements on wave-optics models. Simulation tools are therefore essential for developing such systems, optimizing communication parameters, and designing effective mitigation strategies under these challenging conditions.

Figure 1.1 shows a representative FSO system architecture, highlighting the key components from optical source to receiver and the interfaces for electrical-optical conversion. Specifically, the input data (voice, video, or other digital content) is first encoded and then fed into a modulator, which drives a laser transmitter. The laser

emits a narrow beam directed through the transmitting antenna, propagating across the free-space channel. In this thesis, focus is placed on modeling the turbulence in the atmospheric free-space channel and characterizing how it distorts the beam. At the receiver, the optical signal is collected by the receiving antenna, converted back into electrical form by a photodetector, and finally decoded by electronic circuits to retrieve the transmitted data. Such a layout applies to both terrestrial and atmospheric paths, whether for building-to-building gigabit backhaul or space-based links.

Figure 1.2 underscores how FSO technology extends into satellite communications, covering satellite-to-ground, inter-satellite, and deep-space channels. In this diagram, various platforms (LEO satellites, GEO satellites, aircraft, UAVs, HAPs, etc.) are interconnected through free-space laser beams. Some links, such as LEO-to-ground or aircraft-to-ground, traverse a long atmospheric path and thus require rigorous turbulence modeling. Others, like inter-satellite links in space, bypass the atmosphere but still demand precise pointing and tracking over large distances.

Regardless of the application, FSO systems promise high-capacity and secure data transfer—yet reliability hinges on accurately predicting or mitigating propagation impairments, including atmospheric turbulence for near-Earth vertical links.

1.1 Atmospheric Turbulence and Channel Geometry

Atmospheric turbulence results from random variations in temperature and pressure, producing eddies over a wide range of spatial scales. Following Kolmogorov's theory

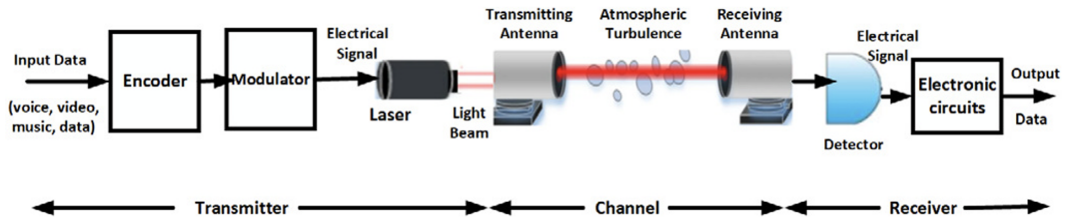


Figure 1.1: A typical free-space optical communications system, highlighting the turbulent atmospheric channel between the transmitter and receiver—this propagation medium introduces effects such as scintillation, beam wander, and wavefront distortion that significantly challenge system performance (adapted from [17]).

of turbulent eddies, energy transfer is modeled as a cascade from larger to smaller vortices throughout the fluid medium [18, 1, 19]. These eddies of varying sizes create local regions with different refractive indices, which in turn introduce phase distortions to any optical wavefront traveling through the atmosphere. This connection between atmospheric turbulence and optical propagation effects was subsequently developed by Rytov and Tatarskii, who established the mathematical framework for analyzing how these refractive index fluctuations impact laser beams [20, 21]. In a LEO satellite downlink, the optical beam passes through a continuum of turbulent refractive-index fluctuations and varying wind speeds leading to spatially varying phase distortions of the beam propagation. Changes in the position of the satellite or altitude alter the slant path through this turbulent medium. Simulations approximate the atmosphere as discrete layers to make computational modeling feasible while still capturing the essential physics of the propagation channel. As the satellite passes overhead the link geometry through the atmosphere changes, altering the turbulence conditions. Analytical models such as the Hufnagel-Valley profile [22] offer a starting point for describing altitude-dependent turbulence strength but do not themselves address the

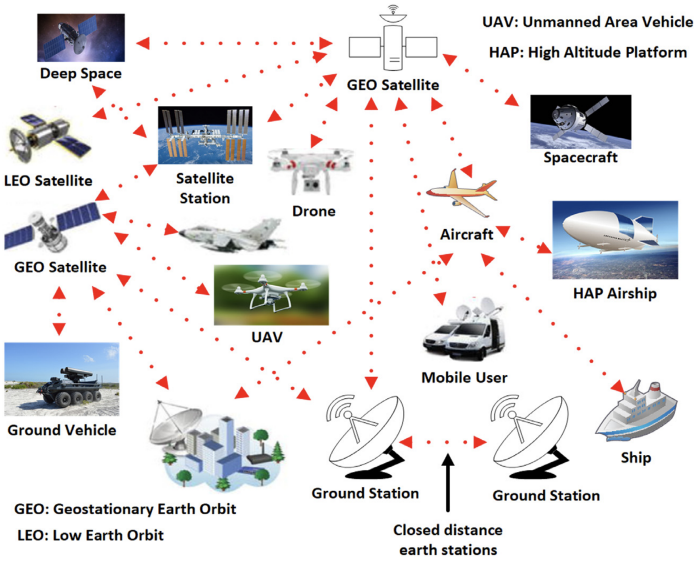


Figure 1.2: Illustration of FSO applications in space communications, including inter-satellite, satellite-to-ground, and deep-space links (adapted from [17]).

computational complexity of propagating beams in time-evolving geometries.

Simulation frameworks like `AOTools` [23] and `HCIPy` [24] are used to generate multi-layer turbulence simulations and study optical propagation through the atmosphere. These simulators rely primarily on general purpose CPU-based architectures. As spatial and temporal resolutions increase, particularly to capture the rapid geometry changes in LEO orbits, these CPU-based approaches can become prohibitively slow due to the large number of calculations required.

Explicit Problem Statement

There is a need for an improved simulator that can (i) model optical propagation through multi-layer atmospheric turbulence under fast-moving orbital trajectories, (ii) maintain high spatial and temporal fidelity, and (iii) leverage highly parallelized GPU resources to ensure computational run times are manageable. Without an efficient,

GPU-accelerated framework, design iteration, real-time testing, or large-scale data generation for advanced algorithms are severely restricted.

1.2 Literature Review

1.2.1 Modelling of Turbulent Atmospheric Channels

Atmospheric turbulence has been extensively investigated over the past century. Kolmogorov's seminal work proposed an energy cascade theory for high-Reynolds-number flows, establishing the basis for modeling the refractive-index power spectrum in a homogeneous and isotropic turbulent medium [25]. Later refinements by von Kármán introduced practical corrections for finite outer scales of turbulence, leading to the often applied von Kármán spectral model [26]. Tatarskii built on these fundamental studies to derive a comprehensive statistical description of wave propagation in random media, which remains a cornerstone of modern turbulence optics research [20].

A branch of turbulence-related research has centered on phase screen generation and wavefront simulation techniques. Lane et al. pioneered the Fourier-based approach for creating random phase screens conforming to specified turbulence spectra [27], while Roddier highlighted how atmospheric turbulence affects optical imaging, shaping subsequent developments in adaptive optics systems [28]. To ensure accurate representation of large-scale (low-frequency) structures, subharmonic methods were introduced, effectively extending the spatial dynamic range of phase screens [27, 28]. More recently, Assémat et al. demonstrated optimized approaches to generate and store phase screens for simulations spanning large apertures or extended propagation

distances [29].

On the numerical modeling front, the split-step Fourier method has played a pivotal role in simulating beam propagation through turbulence [30–32]. This method approximates the continuous propagation process by segmenting the optical path into discrete slices. Within each slice, diffraction is treated in the frequency domain, and phase perturbations are superimposed based on user-defined turbulence screens [33]. Detailed discussions of these numerical techniques can be found in standard references such as *Numerical Recipes* [33] and Schmidt's *Numerical Simulation of Optical Wave Propagation* [32].

In addition to these spatial considerations, the temporal dynamics of turbulence have garnered substantial attention. Taylor's hypothesis [34] laid the groundwork for modeling turbulence advection by assuming that large-scale flow advects smaller-scale eddies in a "frozen" state across the propagation path. This approximation serves well for many moderate-speed flows, but fails under rapidly changing turbulence conditions or in strongly sheared flows [35, 36]. Fried's column extrusion method [37] and subsequent extensions [38] aim to bridge this gap, offering more realistic temporal evolution of phase distortions.

Research on FSO downlinks also emphasizes performance metrics for optical communication. Studies have characterized key parameters such as scintillation indices, beam wander, and power-in-bucket efficiency [21, 39], as well as essential corrections for adaptive optics systems [40, 22]. These models and methods, from theory to practical simulation tools, form the foundation of how turbulence is modeled today and applied in areas like optical communication, imaging, and remote sensing.

1.2.2 Space Optical Communications

Constellation deployments such as StarLink now serve over 2,000,000 customers, rely on more than 9,000 on-orbit laser terminals, perform 250,000 link acquisitions per day, and achieve link uptimes exceeding 99% at distances up to 5,400 km and data rates of 100 Gbps [9]. Terminal production has ramped to 200 units per week, with demonstrated link acquisition in as little as 12 s and continued connectivity down to 122 km altitude during de-orbit[9]. Meanwhile, the European Space Agency's ScyLight framework allocates 60 MEuro for next-generation optical communications and has set targets for demonstrations of $\gg$ 100 Gbps links, with key program milestones set for 2019 and initial in-orbit tests by 2022 [41]. Amazon's Project Kuiper is similarly developing an optical communications infrastructure for its planned constellation, focusing on high-speed connectivity in its global network [10]. Telesat's Lightspeed constellation also incorporates optical inter-satellite links to provide global coverage with minimized latency and enhanced throughput [11]. This shows great promise for FSO technology as a viable backbone for future satellite constellations, driving significant investment and research toward higher bandwidth, lower latency systems aimed at both commercial and government applications in LEO and beyond.

1.2.3 Simulation of Atmospheric Optical Links

Accurate simulation of atmospheric turbulence is essential for designing and testing adaptive optics (AO) systems in astronomy and optical communications. Several software packages provide wave-optics-based numerical simulations of atmospheric turbulence. For instance, `HCIPy` [24] and `AOtools` [23] are both Python-based simulators

that support multi-layer propagation, evolving phase screens, and associated AO analyses. However, while these packages are flexible, certain high-fidelity use-cases (e.g. large-scale space communications scenarios) can still incur substantial computation times when large numbers of independent realizations are needed.

An alternative approach for ground-to-satellite links is provided by `Turandot` [42, 43], a simulator that accounts for von Kármán spectrum turbulence in multiple layers. Although Turandot is capable of generating high-quality turbulence samples, it may take on the order of five hours to produce 2000 fully-propagated realizations [43]. This computational cost becomes prohibitive in applications where tens or hundreds of thousand of samples are required to adequately capture turbulence statistics, such for advanced AO systems. Another widely used MATLAB adaptive optics and atmospheric turbulence simulator package is called Object-Oriented, Matlab and Adaptive Optics (OOMAO) [44]

Additionally, a recent study by Belmonte [45] proposes a novel graph-based method for beam propagation in turbulent media using a triangular adaptive mesh. In that approach, the wavefront and turbulence are represented as signals on a graph, allowing the mesh to adapt locally in response to high-curvature regions and thereby improving numerical accuracy with reduced simulation sampling points.

To circumvent these heavy computational demands, alternative "propagation-less" modeling approaches like LOT [43] have also been proposed. They typically rely on statistical or reduced-order methods (focusing primarily on low-order Zernike modes with Gaussian propagation theory) rather than performing full Fresnel propagation through layered turbulence, while there is speedup. the connection to the physical parameters fo the channel are lost including windspeed variance over the link path.

Shubert [46] demonstrated a wave-optics simulation in MATLAB® using the AO-tools package, modeling the atmosphere with 100 distinct Kolmogorov phase screens based on a Hufnagel-Valley 5/7 profile simulating 24 kilometers of atmosphere near the ground. In this work, each simulation run employed a 400 mm receive aperture (bigger than 1024x1024 phase screens pixels) with up to 5000 random realizations to form fade histograms for a simulated satellite downlink and pre-compensated uplink. This simulation of atmospheric turbulence is extremely computationally intensive, even with the use of MATLAB parallel computing packages.

A Fourier-domain Adaptive Optics Simulation Tool (FAST) is introduced [47] to model both downlink and uplink ground-space optical links in a semi-analytical manner. By using an analytical AO model in the spatial frequency domain (originally developed for astronomical imaging) and invoking the reciprocity principle for uplink scenarios, FAST avoids computationally expensive wave-optics (WO) propagations. Specifically, it first generates residual phase power spectral densities (PSDs) from an AO-corrected phase model, then performs Monte-Carlo draws of random phase screens consistent with those PSDs. This approach retains sufficient fidelity (e.g., capturing tip/tilt, high-order aberrations, and temporal effects under frozen-flow assumptions). Crucially, execution speeds are found to be about ten times faster in downlink cases and roughly two hundred times faster for uplinks compared with Monte Carlo wave optical simulation tools, reflecting the avoided costs of handling large numerical grids for Gaussian beam propagation. For 5000 iterations the execution time was in order of 32.3 seconds compared with 353 seconds in soapy [48], which is another adaptive optics simulator written in Python.

Additionally, Walsh et al. [49] simulated a multi-layer atmospheric turbulence

model using large phase screens of size 1024×1024 pixels using the `AOtools` library [23], corresponding to an aperture of $3.2\,\mathrm{m} \times 3.2\,\mathrm{m}$ with a spatial resolution of $3\,\mathrm{mm}$ per pixel where uplink precompensation was applied to the wavefront. The study emphasizes the importance of high-resolution phase screens for accurately capturing higher-order Zernike modes [49]. Furthermore, the extension to larger apertures and finer resolutions was explored for transmitting donut beams, using Karhunen–Loève basis functions for wavefront reconstruction. These findings highlight the increasing computational complexity associated with larger pixel counts and finer spatial resolutions.

Taken together, these prior studies underscore a clear and growing need for a next-generation atmospheric turbulence simulation tool that leverages GPU-accelerated parallel computing. Modern GPUs are widely available, high-performance, and relatively inexpensive computing resources that excel not only at video processing and machine learning (ML) tasks, but also provide the computational power necessary for physics-based simulations in optical propagation [50]. Such a tool would enable high-fidelity modeling at scale—capable of supporting large apertures, fine spatial resolutions, and massive Monte Carlo ensembles—while reducing runtimes. This capability is beneficial for advancing the design and validation of adaptive optics systems in free-space optical communication.

In the following chapter, a comprehensive framework for modeling downlink propagation is developed . Emphasis is placed on the interplay between the governing wave equations, statistical turbulence models, and the numerical methods required to capture key physics. Particular attention is devoted to phase screen generation,

subharmonic extensions for low-frequency turbulence, and the incorporation of time-dependent dynamics.

1.3 Thesis Contributions

This work proposes, implements, and validates a GPU-accelerated multi-layer atmospheric turbulence simulation architecture specifically geared toward LEO satellite optical downlinks. The main contributions are:

1. **GPU-Accelerated Framework:** A novel software platform built on TensorFlow [51] to handle wave-optics propagation under time-varying, multi-layer turbulence profiles that reflect realistic LEO trajectories.

2. **Dynamic Atmospheric Layer Modeling:** An atmospheric layer slicing and phase-screen generation technique that captures spatially and temporally evolving turbulence, integrating altitude-based wind velocity profiles and refractive-index structures along the slant path.

3. **Validation and Statistics:** Comprehensive checks of the simulator's generated phase and amplitude statistics against well-known theoretical benchmarks, confirming accuracy under various turbulence regimes.

4. **Performance Benchmarking:** Comparisons against an open-source Python wave-optics simulator on CPU, highlighting significant speedups and scalability gained through GPU parallelization.

Collectively, these elements address a critical gap in the field, providing a fast, flexible, and validated tool to simulate LEO-based FSO communication links. By

reducing compute times and enabling more intricate turbulence representations, this simulator aims to accelerate not only wave-optics analysis but also data-driven design paradigms. As a result, it unlocks new capabilities for rapid system prototyping, real-time AO control, and large-scale ML studies that leverage high-fidelity synthetic data [4, 3, 24].

1.4 Thesis Organization

The remainder of this thesis is organized as follows. Chapter 2 provides background on atmospheric turbulence physics and key tools used in the simulation in the context of LEO downlinks, detailing how multi-layer models and time evolution are applied to fast-moving orbital geometries. Chapter 3 describes specifics of the GPU-based implementation, covering the parallelization strategies, TensorFlow-based propagation graphs, and the overall simulator architecture. Chapter 4 presents extensive numerical results of the GPU-based simulator to validate it against phase and amplitude statistics, a performance comparison with a baseline Python simulator, and representative outputs under various turbulence settings. Finally, the thesis concludes in Chapter 5 with summary of main findings, explores broader implications for FSO research, and highlights future directions.

Chapter 2

Background on Turbulence Modeling and Simulation

This chapter describes the atmospheric turbulence models and optical propagation theory essential for simulating free-space optical downlink channels. It begins with a discussion of turbulence statistical models then develops the numerical tools for simulating optical propagation such as the split-step Fourier method and phase-screen techniques.

2.1 Atmospheric Turbulence

Atmospheric turbulence affects the propagation of electromagnetic waves by introducing random phase perturbations along the propagation path. This section introduces the statistical nature of atmospheric turbulence and its impact on optical wave propagation.

2.1.1 Kolmogorov Theory of Turbulence

Turbulence at high Reynolds numbers is often modeled using the statistical framework introduced by Kolmogorov in 1941 [25]. The *Reynolds number* is a dimensionless parameter comparing inertial and viscous forces in fluid flow. Large Reynolds numbers mean inertial forces dominate, promoting the formation of random, energetic eddies.

The underlying principle is shown in Figure 2.1 [1]. Energy is injected at the outer scale (L_0) and cascades through progressively smaller eddies with a nearly constant *energy dissipation rate* ϵ ($\mathrm{m}^2/\mathrm{s}^3$) until being dissipated at the smallest scales. Kinetic energy is transferred into new eddy formation until viscous dissipation (the conversion of kinetic energy into heat via fluid friction) ultimately dominates at the smallest scales (l_0) [18, 1, 19]. This energy transfer behaviour is valid for spatial scales between the inner and outer scale size, which is called the *inertial subrange*. [52–54].

Kolmogorov hypothesized that, in the *inertial subrange*, neither the large-scale forcing nor molecular viscosity significantly affect the statistics of turbulence. Instead, the primary parameters are the energy dissipation rate ϵ and the kinematic viscosity ν. From these assumptions, he postulated a self-similar scaling for the turbulent energy spectrum $E(k)$:

$$E(k) = C_K \, \epsilon^{2/3} \, k^{-5/3}, \tag{2.1.1}$$

where C_K is the Kolmogorov constant. The variable k is the wavenumber defined by $k = 2\pi/\ell$, where ℓ is the eddy length scale. Equation 2.1.1 is then an energy spectrum in *spatial frequency*. In the inertial subrange Kolmogorov theory shows $E(k)$ follows a $k^{-5/3}$ power-law dependence. In this model, the characteristic smallest scale of

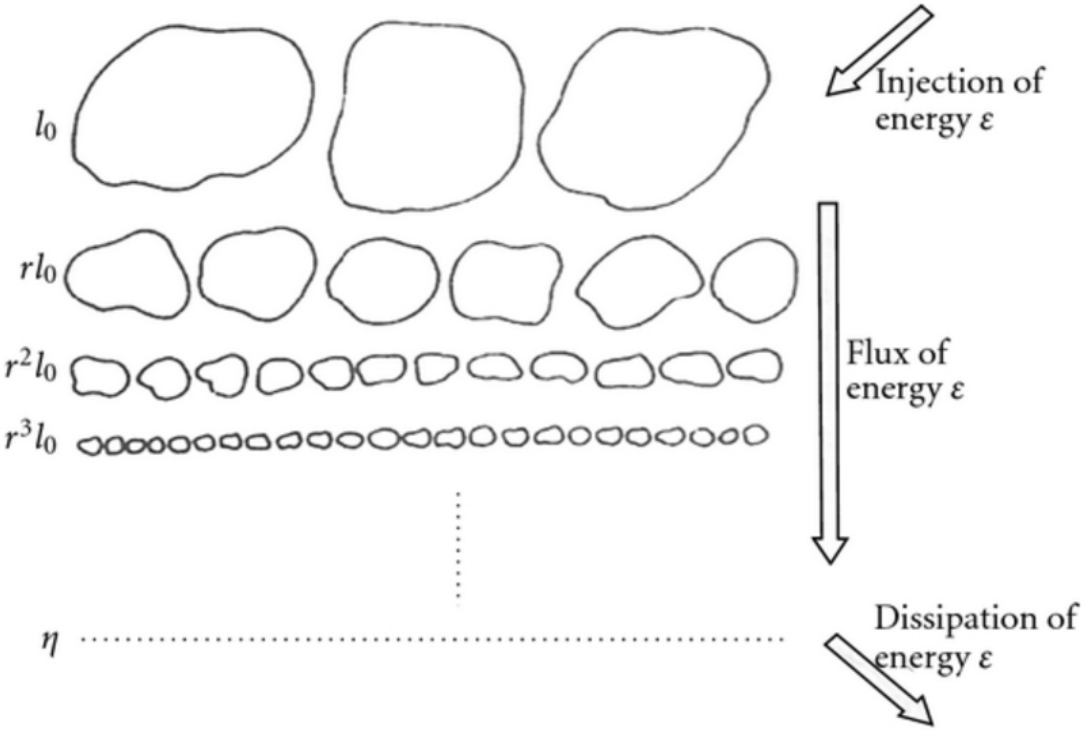


Figure 2.1: Richardson energy cascade depicting the breakdown of turbulent eddies from the integral scale l_0 to the Kolmogorov microscale η. (adapted from [1]).

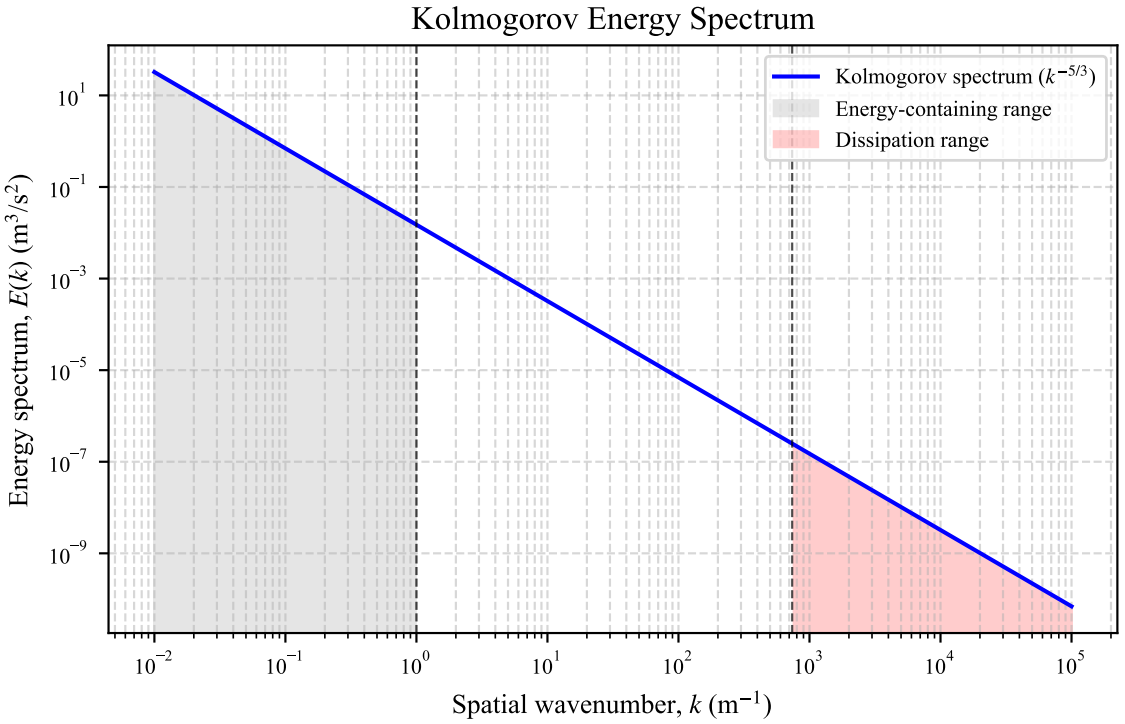


Figure 2.2: Kolmogorov energy spectrum $E(k)$ (2.1.1) illustrating the inertial subrange with a $k^{-5/3}$ scaling (blue line). The energy-containing range (gray shaded region) and dissipation range (red shaded region, $k \gg \eta^{-1}$) are marked. The integral scale L and Kolmogorov microscale η are denoted by vertical dashed lines.

motion, known as the Kolmogorov microscale η, is given by

$$\eta = \left(\frac{\nu^3}{\epsilon}\right)^{\frac{1}{4}}. \tag{2.1.2}$$

Thus, turbulent motions on scales larger than η follow universal scaling laws largely independent of the boundary conditions of the flow, while at scales below η eddies cannot sustain themselves and dissipate as heat [54, 19].

Figure 2.2 illustrates the Kolmogorov energy spectrum across all scales of turbulent motion. The spectrum can be divided into three distinct regions: the energy-containing range at low wavenumbers, the inertial subrange where the $k^{-5/3}$ power law dominates, and the dissipation range at high wavenumbers. The majority of turbulent kinetic energy resides in the largest eddies, corresponding to the energy-containing range. Progress through the inertial subrange, energy cascades according to Equation 2.1.1 without significant input or dissipation, maintaining the characteristic $-5/3$ slope. Finally, in the dissipation range, viscous effects become dominant, causing a rapid decline in energy at scales smaller than the Kolmogorov microscale η.

2.1.2 Refractive index fluctuation in the atmosphere

For visible and IR wavelengths, the refractive index can be expressed as:

$$n(\mathbf{r}, t) = n_0 + n_1(\mathbf{r}, t), \tag{2.1.3}$$

where $\mathbf{r} \in \mathbb{R}^3$ is the spatial position vector, t represents time, $n_0 \approx 1.0$ (near sea level) is the mean refractive index of air, and $n_1(\mathbf{r}, t)$ represents the zero-mean turbulent fluctuations. Random inhomogeneities in air temperature and pressure give rise to local variations in the refractive index. The treatment of optical wave propagation typically neglects temporal variations in the refractive index, which are much slower than the propagation duration. The wave is considered single frequency throughout the propagation process [32]. Equation (2.1.3) can then be written as:

$$n(\mathbf{r}) = n_0 + n_1(\mathbf{r}). \tag{2.1.4}$$

The refractive index of the atmosphere can be expressed as a function of temperature and pressure

$$n(\mathbf{r}) = 1 + 77.6 \times 10^{-6}(1 + 7.52 \times 10^{-3}\lambda^{-2})\frac{P(\mathbf{r})}{T(\mathbf{r})}, \tag{2.1.5}$$

where $P(\mathbf{r})$ is the pressure in millibars at position $\mathbf{r}$, $T(\mathbf{r})$ is the temperature in Kelvin at position $\mathbf{r}$, and λ is the wavelength in micrometers [21].

Kolmogorov's original analysis pertains to velocity fluctuations in homogeneous isotropic turbulence, but the same theory can be applied to the structure of *refractive index* fluctuations in the atmosphere via Equation 2.1.5. The turbulent temperature spectrum follows Kolmogorov Energy Spectrum scaling, Equation (2.1.6), in the inertial range, while pressure changes are negligible, leaving the index spectrum to follow the temperature spectrum [21].

Power Spectral Density of Refractive Index Fluctuations

The *Kolmogorov model* for the three-dimensional spatial power spectral density (PSD) $\Phi_n(\kappa)$ of the refractive index fluctuations is given by

$$\Phi_n(\kappa) = 0.033\, C_n^2\, \kappa^{-11/3}, \quad \kappa_i \ll \kappa \ll \kappa_0, \tag{2.1.6}$$

where $\kappa = |\kappa|$ is the spatial wavenumber. Equation (2.1.6) is valid within the *inertial subrange* for scale sizes between the inner scale l_0 and outer scale L_0, or their corresponding spatial frequencies $\kappa_0 = 2\pi/L_0$ the outer-scale cutoff, and $\kappa_i = 2\pi/l_0$ is the inner-scale cutoff. The $\kappa^{-11/3}$ is a three-dimensional spatial frequency scaling, which when integrated twice for the one dimensional spatial frequency gives the

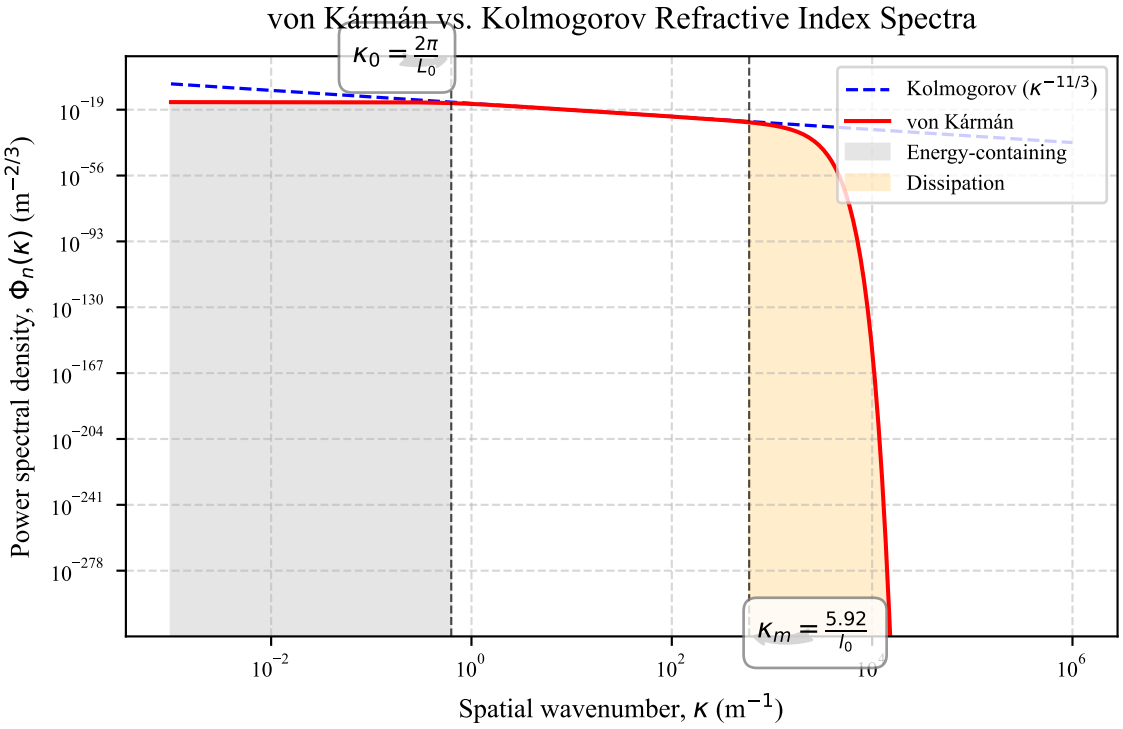


Figure 2.3: Comparison of the von Kármán (red solid line) and Kolmogorov (blue dashed line) spectra for refractive index fluctuations. The von Kármán spectrum transitions from $\kappa^{-11/3}$ behavior in the inertial subrange to finite energy-containing ($\kappa \leq \kappa_0$, gray shaded) and dissipation ($\kappa \geq \kappa_m$, orange shaded) ranges. Vertical dashed lines mark $\kappa_0 = 2\pi/L_0$ (outer scale cutoff) and $\kappa_m = 5.92/l_0$ (inner scale cutoff).

$\kappa^{-5/3}$ power law of (2.1.1). The use of the three-dimensional spectrum in the study of optical propagation through turbulence becomes important when deriving wavefront statistics perpendicular to the direction of propagation from the homogeneous and isotropic atmospheric power spectrum scaling [21, Ch. 5].

The pure Kolmogorov $\kappa^{-5/3}$power law is limited in its applicability at very low wavenumbers (large distance scales) since the energy spectrum continues to infinity as $k \to 0$, which would imply infinite energy in the largest eddies. Similarly, at high wavenumbers (small scales), the spectrum must decrease more rapidly to account for

viscous dissipation effects [18, 54].

To incorporate finite inner and outer scales in the refractive index field, the von Kármán spectral model, shown in Figure 2.3, modifies the idealized Kolmogorov form to smoothly transition between the $k^{-5/3}$ scaling within the inertial subrange while properly accounting for the physics at both extremes of the spectrum

$$\Phi_n(\kappa) = 0.033\, C_n^2\, \frac{\exp\left[-(\kappa/\kappa_m)^2\right]}{(\kappa^2 + \kappa_0^2)^{11/6}}, \tag{2.1.7}$$

where $\kappa_m = 5.92/l_0$ and $\kappa_0 = 2\pi/L_0$. The exponential term ensures a rapid decline of energy at scales smaller than the inner scale l_0, whereas the denominator term prevents divergence at very low wavenumbers ($\kappa \to 0$), effectively setting the outer scale L_0 [21, 55]. Hence, while rooted in fundamental turbulence theory, the von Kármán model also relies on empirical constants to fit real-world measurements [20, 21].

Empirical validation of the von Kármán spectrum model has been established through key experimental studies. Foundational measurements provided in [20, Ch. 4] demonstrate that pure Kolmogorov models fail at both small and large scales, while Andrews and Phillips [21] compiled field data confirming that the von Kármán formulation captures the finite inner and outer scale effects observed in real turbulence spectrum measurements.

Refractive Index Structure Function

A useful statistical measure of the random index of refraction process is the *refractive index structure function.* For statistically stationary turbulence, the ensemble average becomes independent of time, and under the assumption of local homogeneity, the

structure function depends only on the magnitude of the separation vector, allowing for the simplification

$$D_n(r) = \left\langle \left[n_1(\mathbf{r}_1) - n_1(\mathbf{r}_1 + \mathbf{r}) \right]^2 \right\rangle_{\mathcal{E}}, \tag{2.1.8}$$

where $r = ||\mathbf{r}||$ is the separation distance between points in the field.

The connection between the structure function and spectral model can be established through their mathematical relationship. For homogeneous, isotropic turbulence, the structure function $D_n(r)$ and the power spectral density $\Phi_n(\kappa)$ are linked through the Fourier transform relationship between power spectral density and autocorrelation functions, which when applied to (2.1.8) can be used to derive the following [20, 18]

$$D_n(r) = 8\pi \int_0^\infty \Phi_n(\kappa) \left[1 - \frac{\sin(\kappa r)}{\kappa r} \right] \kappa^2 d\kappa. \tag{2.1.9}$$

When the Kolmogorov turbulence spectrum (2.1.6) is inserted for $\Phi_n(\kappa)$ in Equation (2.1.9) the structure function is given by

$$D_n(r) = C_n^2 \, r^{2/3}, \tag{2.1.10}$$

where $r = \|\mathbf{r}\|$ is the spatial separation distance and C_n^2 is the index of refraction structure constant, having units of $\mathrm{m}^{-2/3}$, which quantifies the strength of the refractive index fluctuations. Note that this structure function scaling applies to the homogeneous and isotropic distribution of index fluctuations in the atmosphere. In the next section a different structure function is derived for points on a transverse propagating optical wavefront.

2.2 Optical Propagation Theory

This section begins by presenting the fundamental wave equation derived from Maxwell's equations, then introduces the Huygens-Fresnel principle and the paraxial approximation used to simplify that wave equation, followed by an overview of Fresnel and Fraunhofer diffraction regimes.

2.2.1 Basic Electromagnetic Propagation Theory

At optical frequencies, Maxwell's equations can often be reduced to a scalar wave equation under the assumptions of linear, isotropic, and source-free media [56, 57]

$$\nabla^2 u(\mathbf{r}, t) - \frac{n^2(\mathbf{r})}{c_0^2} \frac{\partial^2 u(\mathbf{r}, t)}{\partial t^2} = 0, \tag{2.2.1}$$

where

- $u(\mathbf{r}, t)$ is the scalar approximation of the electric field amplitude,
- $n(\mathbf{r}) = n_0 + n_1(\mathbf{r})$ is the refractive index (with mean n_0 and small fluctuations $n_1(\mathbf{r})$),
- c_0 is the speed of light in vacuum,
- $\mathbf{r} = (x, y, z)$ is the spatial coordinate.

This scalar approximation assumes that polarization effects are negligible and is valid when the medium is isotropic, as is typically the case for atmospheric propagation at optical wavelengths [21]. Assuming a time-harmonic dependence,

$u(\mathbf{r}, t) = U(\mathbf{r})\, e^{-i\omega t}$, the Helmholtz equation is obtained as the following equation

$$\nabla^2 U(\mathbf{r}) + k^2\, n^2(\mathbf{r})\, U(\mathbf{r}) = 0, \tag{2.2.2}$$

where $k = \frac{2\pi}{\lambda}$ is the free-space wavenumber and λ is the wavelength in vacuum. This scalar Helmholtz equation is the starting point for many models of optical wave propagation, including those that incorporate atmospheric turbulence via $n_1(\mathbf{r})$ introduced in Equation (2.1.3).

2.2.2 Huygens-Fresnel Principle and Paraxial Approximation

Huygens-Fresnel Principle

The Huygens-Fresnel principle states that each point on a wavefront acts as a secondary source of spherical wavelets; the sum (integral) of these wavelets at a later plane gives the propagated wavefront [56, 57]. In the scalar diffraction framework, the field $U(x, y, z)$ at a distance z from an aperture can be computed using integrals akin to the Rayleigh-Sommerfeld or Kirchhoff diffraction formulations.

Conceptually, this principle captures diffraction effects arising from obstacles and apertures. For complex refractive index distributions (such as turbulence in the atmosphere), additional phase perturbations must be included to account for random fluctuations.

Paraxial Wave Equation

For beams propagating primarily along the z-axis with small angular divergence (i.e., $\theta \ll 1$ rad), the *paraxial approximation* simplifies the Helmholtz equation significantly [57, 32]. Let

$$U(x, y, z) \;=\; H(x, y, z)\, \exp\big(i\, k\, n_0\, z\big), \tag{2.2.3}$$

where $H(x, y, z)$ is a slowly varying envelope. Substituting into (2.2.2) and neglecting $\partial^2 H/\partial z^2$ terms yields the paraxial wave equation

$$2\, i\, k\, n_0\, \frac{\partial H}{\partial z} \;+\; \nabla_\perp^2 H \;+\; 2\, k^2\, n_0\, n_1(\mathbf{r})\, H \;=\; 0, \tag{2.2.4}$$

with $\nabla_\perp^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$. This form is widely used to model laser beam propagation in weakly inhomogeneous media such as the turbulent atmosphere.

2.2.3 Fresnel and Fraunhofer Diffraction

Fresnel (Near-Field) Regime

Fresnel diffraction describes wave propagation where the observation plane is *not* extremely far from the diffracting aperture or object, but still satisfies the paraxial condition. Importantly, the Fresnel integral arises as an approximate solution to the paraxial wave equation (Equation (2.2.4)) under the assumption of free-space propagation (i.e., $n_1(\mathbf{r}) = 0$). By neglecting turbulence and solving the homogeneous version of the paraxial equation, one obtains this integral form, which captures the effects of diffraction in the near field. This connection demonstrates how the Fresnel diffraction model builds directly on the paraxial Helmholtz framework, linking wave

propagation theory to practical beam modeling. The Fresnel diffraction integral for a field at distance z from a source plane can be written as [57, 56]

$$U(x,y,z) \;=\; \frac{e^{\,i\,k\,z}}{i\,\lambda\,z}\iint_{-\infty}^{\infty} U(x',y',0)\,\exp\Big[\,i\,\tfrac{k}{2z}\big((x-x')^2+(y-y')^2\big)\Big]\,dx'\,dy'. \quad (2.2.5)$$

This so-called *Fresnel integral* can often be efficiently computed with Fast Fourier Transforms (FFTs) in numerical simulations.

Fraunhofer (Far-Field) Regime

Fraunhofer diffraction is the asymptotic far-field limit of Fresnel diffraction, valid when the propagation distance z is large compared to the square of the beam diameter divided by the wavelength $\Big(\text{i.e., } z \gg \frac{D^2}{\lambda}\Big)$. Under these conditions, the diffracted field becomes essentially the *Fourier transform* of the aperture function [56, 57]:

$$U(x,y,z) \;\propto\; \exp\big(i\,k\,z\big)\iint_{-\infty}^{\infty} U(x',y',0)\;\exp\Big[-\,i\,\tfrac{k}{z}\big(x\,x'+y\,y'\big)\Big]\,dx'\,dy'. \quad (2.2.6)$$

Fraunhofer diffraction is vital in analyzing far-field beam patterns, laser spot sizes at large distances, and optical Fourier transforming systems. In atmospheric optics, Fraunhofer concepts often underlie simple beam divergence and spot size calculations.

2.2.4 Rytov Theory and the Mutual Coherence Function

The refractive index fluctuations characterized by the von Kármán spectrum directly impact electromagnetic wave propagation through the atmosphere. To quantify these effects mathematically, perturbation methods are now employed that relate the statistical properties of the turbulent medium to observable optical field characteristics.

In atmospheric optics, a common starting point for analyzing wave propagation is the Helmholtz equation, which in a source-free, nonmagnetic, and isotropic medium and by solving (2.2.4) and (2.2.2), it can be written as

$$\nabla^2 U(\mathbf{r}) \; + \; k^2 \, n^2(\mathbf{r}) \, U(\mathbf{r}) \; = \; 0, \tag{2.2.7}$$

where $U(\mathbf{r})$ is the optical field, $k = 2\pi/\lambda$ is the vacuum wavenumber, and $n(\mathbf{r})$ is the refractive index. In the atmosphere, random fluctuations in temperature and pressure give rise to small variations in $n(\mathbf{r})$. Under the assumption that these fluctuations are weak, a typical assumption is to decompose

$$n(\mathbf{r}) \; = \; n_0 \; + \; n_1(\mathbf{r}), \qquad \big|n_1(\mathbf{r})\big| \ll n_0,$$

so that

$$n^2(\mathbf{r}) \; \approx \; n_0^2 + 2\, n_0 \, n_1(\mathbf{r}),$$

allowing a perturbative treatment of the wave equation. Then the wave equation becomes:

$$\left\{\nabla^2 + k^2 \left[1 + 2n_1(\mathbf{r})\right]\right\} U(\mathbf{r}) = 0. \tag{2.2.8}$$

Rytov Approximation. When the turbulence is sufficiently weak, Rytov theory [20, 21] provides a convenient way to capture the first-order effects of random refractive-index fluctuations. By solving Equation (2.2.8) with the methods of Fourier optics and using Green functions [32], the following approximate solutions can be acquired.

Writing the complex field in the logarithmic form

$$U(\mathbf{r}) \;=\; U_0(\mathbf{r})\,\exp\Big[\psi(\mathbf{r})\Big], \tag{2.2.9}$$

where $U_0(\mathbf{r})$ is the *unperturbed* (e.g. freely propagating) solution in the absence of turbulence, and

$$\psi(\mathbf{r}) \;=\; \chi(\mathbf{r}) + i\,\varphi(\mathbf{r})$$

decomposes into real *log-amplitude* fluctuations $\chi(\mathbf{r})$ and *phase* fluctuations $\varphi(\mathbf{r})$.

Another key concept is the *Mutual Coherence Function* (MCF), defined as

$$\Gamma(\mathbf{r}_1, \mathbf{r}_2) = \langle U^*(\mathbf{r}_1)\, U(\mathbf{r}_2)\rangle, \tag{2.2.10}$$

where $U^*(\mathbf{r}_1)$ and $U(\mathbf{r}_2)$ are optical field and conjugate of the optical filed at spatial positions r_1 and r_2, representing two points in the optical field [57]. The MCF quantifies the degree of coherence between two points in the field.

For a plane wave in weak turbulence, the MCF can be approximated as [32]:

$$\Gamma(\mathbf{r}_1, \mathbf{r}_2) \approx U_0^*(\mathbf{r}_1)\, U_0(\mathbf{r}_2)\,\exp\Big[-\tfrac{1}{2} D_\psi(\mathbf{r}_1, \mathbf{r}_2)\Big]. \tag{2.2.11}$$

where $D_\psi(\mathbf{r}_1, \mathbf{r}_2)$ is the wave structure function given by

$$D_\psi(\mathbf{r}_1, \mathbf{r}_2) = \langle |\psi(\mathbf{r}_1) - \psi(\mathbf{r}_2)|^2\rangle. \tag{2.2.12}$$

Equation (2.2.12) characterizes the statistical variance of complex phase differences between two points on the wavefront. This structure function can be separated into

the effects of log-amplitude (χ) and phase (φ) fluctuations [20, 21]

$$D_\psi(\mathbf{r}_1, \mathbf{r}_2) = D_\chi(\mathbf{r}_1, \mathbf{r}_2) + D_\varphi(\mathbf{r}_1, \mathbf{r}_2), \tag{2.2.13}$$

where D_χ and D_φ are the structure functions for log-amplitude and phase, respectively. Equation (2.2.11) shows the physical significance of the structure function: when $D_\psi(\mathbf{r}_1, \mathbf{r}_2)$ increases with separation distance $\|\mathbf{r}_1 - \mathbf{r}_2\|$, the exponential term decreases, causing a corresponding decay in coherence. Under Kolmogorov turbulence statistics, D_ψ scales approximately as $\|\mathbf{r}_1 - \mathbf{r}_2\|^{5/3}$ within the inertial subrange [21, 58], leading to a characteristic coherence length beyond which the optical field becomes effectively uncorrelated.

Scintillation is the normalized variance of the intensity fluctuations, which in the weak-fluctuation regime is well approximated by the variance of the log-amplitude fluctuations. A key parameter in determining the validity of this approximation is the *Rytov variance*, β_0^2. Physically, β_0^2 measures the accumulated strength of optical scintillation along the path of propagation, analogous to the twinkling of stars when viewed from Earth. When $\beta_0^2 \ll 1$, the perturbative expansion in Rytov theory holds. For a plane wave propagating a distance z, β_0^2 can be approximated as

$$\beta_0^2 \;\approx\; 1.23\, C_n^2\, k^{7/6}\, z^{11/6}, \tag{2.2.14}$$

where C_n^2 is the refractive-index structure constant (defined in Section 2.1.2), $k = 2\pi/\lambda$ is the optical wavenumber, and z is the propagation distance. This relation illustrates how, in the *weak turbulence* regime, the scintillation becomes more pronounced with increasing propagation distance or wavenumber.

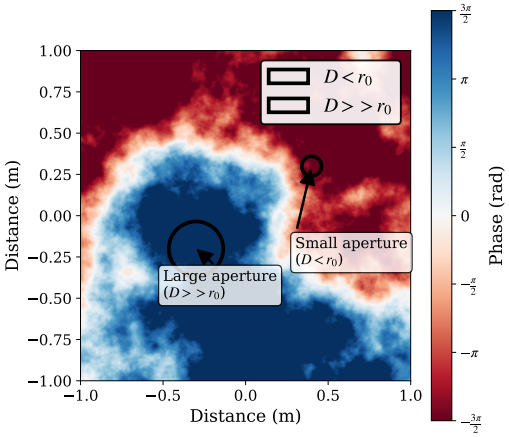
Building on this theoretical framework, the practical effects of atmospheric turbulence can be characterized through two primary parameters: the Fried parameter r_0, which emerges from the mutual coherence function and characterizes the spatial scale of phase distortions, and the scintillation index σ_I^2, which quantifies intensity fluctuations. These parameters are essential for evaluating and designing free-space optical systems.

2.2.5 Fried Parameter and Scintillation Index

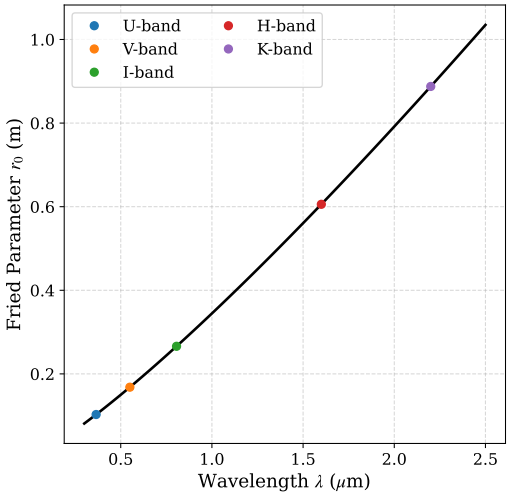
Atmospheric turbulence degrades optical wave propagation through two primary effects: phase distortions and intensity fluctuations. The loss of wavefront coherence is quantified by the Fried parameter r_0, which arises from the mutual coherence function. In the weak scintillation regime where amplitude fluctuations are considered negligibly small, r_0 is primarily determined by phase distortions. As the wavefront propagates through the atmosphere picking up phase distortions it also experiences intensity fluctuations quantified by the scintillation index σ_I^2. These metrics are critical for defining turbulence strength and evaluating the performance of free-space optical communication systems.

Fried Parameter (r_0): Coherence Length of Turbulence

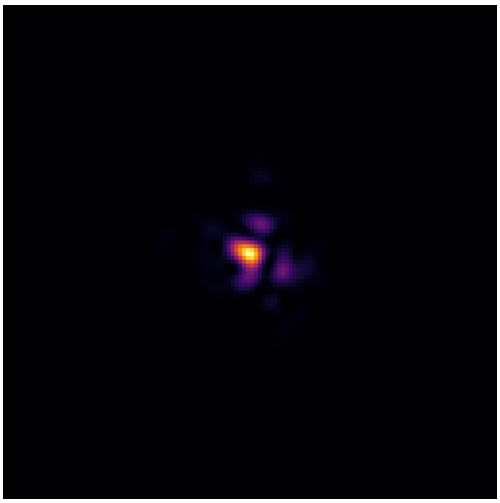
The Fried parameter r_0, also known as the atmospheric coherence length, represents the transverse distance over which the phase of an optical wave remains correlated. It was first introduced by David L. Fried in 1966 [58] to characterize the resolving power of ground-based telescopes. It is derived from the mutual coherence function introduced in Section 2.2.4. For a plane wave propagating through turbulence, the



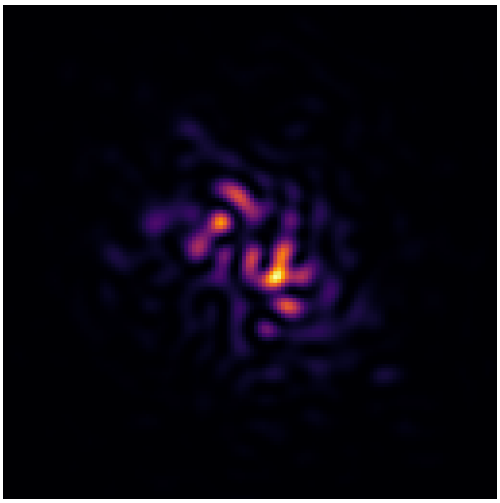
(a) Wavefront phase screen



(b) r_0 wavelength scaling



(c) Focus spot ($D < r_0$)



(d) Focus spot ($D >> r_0$)

Figure 2.4: Atmospheric turbulence effects characterized by the Fried parameter (r_0): (a) Wavefront with turbulence-induced phase distortions showing a small aperture ($D < r_0$, $\sigma_\phi \sim 1$ radian) and large aperture ($D >> r_0$, $\sigma_\phi \gg 1$ radian). Color scale represents phase in radians. (b) Fried parameter wavelength dependence following $r_0 \propto \lambda^{6/5}$, with common astronomical bands (U: 365 nm, V: 550 nm, I: 806 nm, H: 1.6 μm, K: 2.2 μm) marked. (c) Focus spot for the small aperture ($D < r_0$), simulation shows the focal plane intensity profile, with less turbulence distortion over the aperture and maintaining a more coherent spot. (d) Focus spot for the large aperture ($D >> r_0$), simulation shows the focal plane intensity profile, exhibiting significant speckle structure due to stronger turbulence. This spot will produce seeing-limited (λ/r_0) resolution.

phase structure function $D_\varphi(\mathbf{r}_1, \mathbf{r}_2)$ can be derived from the power spectral density of refractive index fluctuations $\Phi_n(\kappa)$ discussed in Section 2.1.2 [20, 28]

$$D_\varphi(\mathbf{r}) = 2.91k^2|\mathbf{r}|^{5/3}\int_0^z C_n^2(\zeta)d\zeta, \tag{2.2.15}$$

where $\mathbf{r} = \mathbf{r}_1 - \mathbf{r}_2$ is the spatial separation vector across the wavefront, $k = 2\pi/\lambda$ is the optical wavenumber, and z is the propagation distance [59, 21].

The Fried parameter r_0 is defined as 2.1 times the seperation distance across which the structure function equals 2 radian^2 (i.e., $D_\varphi(r_0) = 2$) [59][21, Pg. 194]. The 1/e coherence point of the MCF (2.2.11) is known as the *coherence radius* ρ_c, from which Fried parameter $r_0 = 2.1\rho_c$ gets its alternate name as the *coherence diameter*. r_0 for a plane wave r_0 is given by

$$r_0 = \left[0.423\,k^2\int_0^z C_n^2(\zeta)\,\mathrm{d}\zeta\right]^{-3/5}, \tag{2.2.16}$$

where $k = 2\pi/\lambda$ is the optical wavenumber, $C_n^2(\zeta)$ is the refractive index structure constant along the propagation path, and z is the propagation distance. Physically, r_0 represents the distance over which the wavefront is largely correlated. Smaller r_0 values mean shorter correlation distances and indicate stronger turbulence. As can be seen from (2.2.16) the coherence length decreases with increasing C_n^2 or longer propagation paths.

A simple expression for the phase structure function of a plane wave in terms of r_0 is given by [32, Eq 9.44]

$$D_\varphi(r) = 6.88\left(\frac{r}{r_0}\right)^{5/3}, \tag{2.2.17}$$

which leads to a simplified expression for the mutual coherence function of a plane wave paramaterized by r_0

$$\mathrm{MCF}(r) = \exp\left[-3.44\left(\frac{r}{r_0}\right)^{5/3}\right], \tag{2.2.18}$$

which follows from the relationship in Equation (2.2.11).

In astronomical contexts, the Fried parameter is directly related to the concept of "seeing," which quantifies the angular resolution limit imposed by atmospheric turbulence. Atmospheric seeing, typically measured in arcseconds (1 arcsecond is approximately 4.84 μradian), is defined as the full-width at half-maximum (FWHM) of the long-exposure point spread function of a stellar image and is related to the Fried parameter by

$$\text{seeing} \approx 0.98\frac{\lambda}{r_0}, \tag{2.2.19}$$

where the factor 0.98 arises from the mathematical relationship between the FWHM of the seeing-limited point spread function and r_0 [28, 58]. This equation shows that smaller values of r_0 produce larger seeing which is worse angular resolution. For large telescopes the seeing becomes limited by the coherence diameter and not the aperture diameter as in a diffraction limited imaging system. This motivates the need for AO systems to correct the turbulence impaired wavefront and restore the image resolution of the instrument, as discussed in (1.2).

Figure 2.4 illustrates the fundamental aspects of the Fried parameter and its impact on astronomical imaging. Panel (a) shows two apertures superimposed over a turbulent phase profile. The aperture smaller than the Fried parameter ($D < r_0$) receives a wavefront that is still largely coherent and forms the spot in panel (c).

The resolution is primarily limited by diffraction rather than atmospheric turbulence. Conversely, the large aperture with $D > r_0$ receives a wavefront with severe phase distortions producing the highly aberrated spot in panel (d). This demonstrates that building larger telescopes without adaptive optics does not improve resolution beyond the atmospheric limit. Panel (b) plots the wavelength dependence $r_0 \propto \lambda^{6/5}$ of Equation (2.2.16) marked with standard astronomical photometric bands. These bands span from the ultraviolet (U-band, 365nm) through visible (V-band, 550nm) to near-infrared (I-band, 806nm, H-band, 1.6μm, and K-band, 2.2 μm), highlighting the dramatic improvement in atmospheric coherence length across commonly used observing wavelengths. For example, the Fried parameter in K-band ($\lambda = 2.2$ μm) is approximately 5.2 times larger than in V-band ($\lambda = 550$ nm), leading to significantly better atmospheric seeing and image quality. This wavelength-dependent behavior explains why infrared astronomy and communications often achieve better performance through turbulence than their visible-wavelength counterparts, as longer wavelengths naturally experience larger r_0 values and therefore maintain diffraction-limited performance for larger apertures.

The Fried parameter is fundamental to optical system design and performance. The wavelength dependence of $r_0 \propto \lambda^{6/5}$, derived from Equation (2.2.16) explains why longer wavelengths (e.g., infrared) are less affected by atmospheric turbulence than visible light. This scaling relationship has important implications for both astronomical observations and free-space optical communications. Systems operating near 1550 nm experience less turbulence-induced phase distortion compared to those using visible wavelengths. The ratio D/r_0 describes how many coherence regions fit across a given telescope aperture of diameter D. The higher the ratio, the more incoherent

the wavefront across the aperture will be. In communications and imaging contexts, D/r_0 indicates the number of turbulent cells across the aperture, with $D/r_0 \approx 1$ marking the critical transition from diffraction-limited to seeing-limited performance [58]. When operating with $D/r_0 > 1$, optical systems require active wavefront sensing and correction techniques to approach their theoretical diffraction limit, as the uncorrected wavefront distortions would otherwise severely degrade system performance. For adaptive optics applications the required number of actuators needed for effective wavefront correction is on the order of ($\propto (D/r_0)^2$) [60].

Scintillation Index (σ_I^2): Intensity Fluctuations

The scintillation index quantifies the normalized variance of received intensity fluctuations. In the context of optical propagation, intensity I refers to the time-averaged power per unit area of the electromagnetic wave, proportional to the square of the electric field amplitude: $I(\mathbf{r}) = |U(\mathbf{r})|^2$, where $U(\mathbf{r})$ is the complex field amplitude. The normalized intensity variance is defined as:

$$\sigma_I^2 = \frac{\langle I^2 \rangle - \langle I \rangle^2}{\langle I \rangle^2}, \tag{2.2.20}$$

where I is the optical intensity. Under weak turbulence conditions for an infinite plane wave ($\sigma_I^2 < 1$), the Rytov approximation gives

$$\sigma_I^2 \approx 1.23\, C_n^2\, k^{7/6}\, z^{11/6}. \tag{2.2.21}$$

As illustrated in Figure 2.5, the behavior of scintillation exhibits two distinct regimes characterized by the Rytov variance β_0^2.

Recall from Section 2.2.4 that the Rytov variance β_0^2 quantifies the strength of optical scintillation and is given by Equation (2.2.14) and serves as the key parameter for determining the applicability of weak turbulence approximations.

In the weak fluctuation regime ($\beta_0^2 < 1$), the scintillation index follows the Rytov theory prediction, growing linearly with turbulence strength ($\sigma_I^2 \propto \beta_0^2$). However, as turbulence strengthens ($\beta_0^2 > 1$), the relationship breaks down due to multiple scattering effects, leading to a saturation of intensity fluctuations. This saturation behavior, where σ_I^2 approaches an asymptotic value despite increasing turbulence strength, marks a fundamental limitation of weak turbulence theory and necessitates more sophisticated modeling approaches for strong turbulence conditions.

Equation (2.2.21) shows that scintillation grows with turbulence strength, optical wavenumber, and propagation distance. However, in strong turbulence ($\sigma_I^2 \geq 1$), nonlinear effects cause saturation, and the Rytov approximation breaks down [21]. The scintillation index then stabilizes or even decreases, a phenomenon known as *scintillation saturation* [61].

Figure 2.6 illustrates how r_0 and σ_I^2 evolve with increasing turbulence strength C_n^2. The Fried parameter decreases with stronger turbulence following a $r_0 \propto C_n^{-6/5}$ relationship (derived from Equation (2.2.16)), indicating worse wavefront coherence. Simultaneously, the scintillation index increases approximately as $\sigma_I^2 \propto C_n^2$ in the weak turbulence regime before reaching saturation.

2.2.6 Vertical Turbulence Profile $C_n^2(h)$

Atmospheric turbulence strength varies significantly with altitude, making the vertical profile $C_n^2(h)$ critical for modeling optical propagation through the atmosphere.

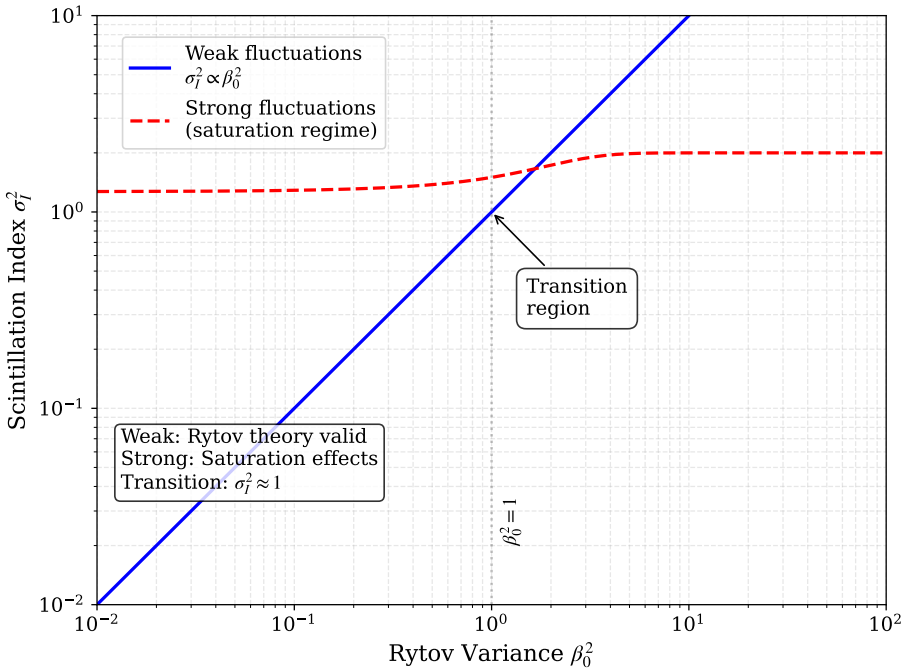


Figure 2.5: Scintillation index σ_I^2 versus Rytov variance β_0^2 in log-log scale, showing the distinct turbulence regimes. In the weak fluctuation regime ($\beta_0^2 < 1$), Rytov theory accurately predicts the linear relationship $\sigma_I^2 \propto \beta_0^2$ (blue solid line). As turbulence strengthens ($\beta_0^2 > 1$), nonlinear effects cause saturation (red dashed line), where σ_I^2 approaches an asymptotic value despite increasing β_0^2. The transition region near $\sigma_I^2 \approx 1$ marks the breakdown of Rytov theory and the onset of strong turbulence effects.

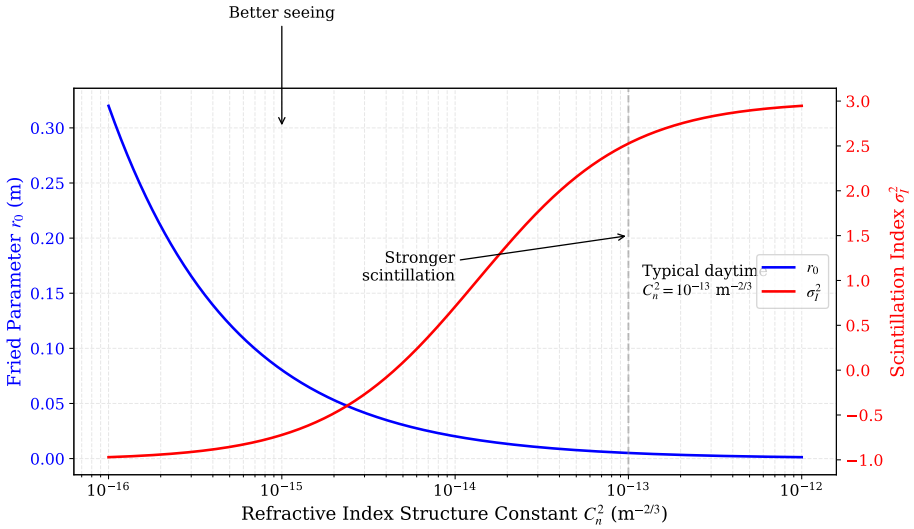


Figure 2.6: Visualizing Fried parameter r_0 (blue, left axis) and scintillation index σ_I^2 (red, right axis) versus turbulence strength C_n^2 for a wavelength of 1550 nm and 1 km propagation path. As C_n^2 increases, r_0 decreases following a $-3/5$ power law (worse wavefront coherence), while σ_I^2 rises (stronger intensity fluctuations) before reaching saturation. The vertical dashed line marks typical daytime turbulence ($C_n^2 = 10^{-13}$ m$^{-2/3}$), where $r_0 \approx 5$ cm and significant scintillation occurs. Note the opposite behavior of these parameters, highlighting the challenge of maintaining both phase coherence and intensity stability in strong turbulence conditions.

This altitude dependence arises from temperature gradients, wind shear, and terrain-induced mechanical turbulence near the ground [62].

The simplest model for horizontal propagation assumes a constant C_n^2 with height, but this fails to capture the layered structure observed in real atmospheres. Instead, altitude-dependent models like the *Hufnagel-Valley* (HV) [22] profile is widely used.

Hufnagel-Valley Turbulence Model

The Hufnagel-Valley (H-V) model [62] provides a widely-accepted characterization of the altitude-dependent refractive index structure constant $C_n^2(h)$. Its empirical form captures both boundary layer and upper atmospheric effects

$$C_n^2(h) = 0.00594 \left(\frac{w_{rms}}{27}\right)^2 (10^{-5}h)^{10} e^{-h/1000} + 2.7 \times 10^{-16} e^{-h/1500} + A e^{-h/100}. \quad (2.2.22)$$

The first term models high-altitude turbulence driven by wind shear in the tropopause region (approximately 10-15 km) [62, 21]. This term includes the parameter w_{rms}, which represents the high-altitude rms wind speed in m/s, typically ranging from 5 to 30 m/s. The h^{10} factor causes this term to peak at approximately 10 km, making it the dominant contributor to turbulence in the upper troposphere where significant wind shear occurs due to jet streams (narrow, high-speed air currents typically found at 9-12 km altitude with speeds of 100-200 km/h that create strong vertical velocity gradients with surrounding air masses) [63]. The exponential decay with scale height 1000 m ensures this contribution diminishes at very high altitudes. The second term, represents background turbulence that is present throughout the atmosphere but is not directly related to wind speed. This term has a slower decay rate (scale height 1500 m) and provides a baseline turbulence level that persists into the stratosphere.

The third term, captures the strong surface-layer effects that dominate turbulence near the ground. The parameter A (typically ranging from 1.7×10^{-14} to 8.4×10^{-14} $\mathrm{m}^{-2/3}$) sets the ground-level turbulence strength ($C_n^2(0)$), which is heavily influenced by solar heating, terrain roughness, and local meteorological conditions. With its rapid decay (scale height 100 m), this term primarily affects the lowest kilometer of the atmosphere where mechanical and thermal effects from the Earth surface create strong turbulent eddies.

Bufton Wind Profile

The wind velocity profile along the propagation path significantly impacts the temporal evolution of turbulence. The Bufton wind model [64] provides an empirical description of the altitude-dependent wind speed

$$v_{\mathrm{HV}}(h) = w_g + w_p \exp\left[-\left(\frac{h - h_p}{L_w}\right)^2\right], \tag{2.2.23}$$

where w_g represents the ground wind speed, w_p is the peak wind speed in the tropopause, h_p is the height of peak wind speed (typically near the tropopause), and L_w is the characteristic thickness of the wind layer. This model captures both the surface wind effects and the strong wind shear typically observed in the upper troposphere.

Note that w_p represents the additional wind speed contribution at the tropopause rather than the absolute peak value, which is why the maximum wind speed at height h_p equals $w_g + w_p$. Similarly, at ground level ($h = 0$), the model yields $w_g + w_p \exp[-(h_p/L_w)^2]$, which is slightly greater than w_g alone—a deliberate design choice that creates a smooth profile and acknowledges the minimal but non-zero

coupling between atmospheric layers [64, 21].

For a moving observer (such as tracking a satellite), the effective wind profile must incorporate both atmospheric winds and relative motion:

$$v_{\text{total}}(h) = v_{\text{HV}}(h) + v_{\text{rel}}(h), \tag{2.2.24}$$

where $v_{\text{rel}}(h)$ represents any additional relative motion between the optical path and the atmosphere. This combined profile determines the characteristic timescales of turbulence-induced fluctuations and influences the temporal coherence of the optical field.

The wind profile plays a crucial role in determining the temporal evolution of phase screens in numerical simulations presented in Section 2.2.10 and affects the validity of Taylor frozen flow approximation (see Section 2.2.9). At each altitude, the local wind velocity influences both the rate of turbulence evolution and the applicability of frozen-flow assumptions for modeling temporal dynamics.

Building on the basic scintillation index formula for horizontal paths given in Equation (2.2.21), for the slant path geometry of satellite downlinks can be extended for altitude dependent C_n^2, where C_n^2 varies with altitude and the propagation distance depends on the zenith angle, which will be fully discussed in Section 2.3.

For satellite downlinks, the scintillation index can be expressed as [21]

$$\sigma_I^2(r, L) = 2.25k^{7/6} \sec^{11/6}(\zeta) \int_0^H C_n^2(h)(h - h_0)^{5/6} dh, \tag{2.2.25}$$

where k is the wavenumber, ζ is the zenith angle, and h_0 is the observer altitude. Using the H-V5/7 model with $\lambda = 1.55$ μm and zenith angles $0^\circ \leq \zeta \leq 60^\circ$, the

scintillation index varies across the beam profile from approximately 0.07 to 0.23, independent of the initial beam parameters. This uniformity in fluctuations across the beam profile occurs because off-axis scintillation closely matches on-axis behavior for most of the diffraction-limited spot size.

Using the H-V5/7 model ($w = 21$ m/s, $A = 1.7 \times 10^{-14}$ m$^{-2/3}$) at $\lambda = 1.55$ μm yields:

- Fried parameter $r_0 \approx 19.4$ cm (ground observer)
- Scintillation index $\sigma_I^2 \sim 0.07$–0.23 across beam profile (Eq. (2.2.25))

These parameters underscore the challenge of maintaining signal fidelity in downlinks, particularly for low-elevation angles ($\zeta > 60°$) where $\sec^{11/6}(\zeta)$ scaling in Eq. (2.2.25) exacerbates intensity fluctuations.

2.2.7 Numerical Simulation of Optical Propagation

Accurately modeling laser beam propagation through atmospheric turbulence is a challenging problem. While analytical solutions are available in certain limiting cases (e.g., weak turbulence or far-field approximations), realistic conditions often involve moderate-to-strong turbulence, layered structures, and non-stationary effects that make purely theoretical approaches intractable.

Numerical simulations provide a flexible, systematic, and efficient way to explore a broad range of turbulence scenarios and system parameters. By discretizing the wave equation or diffraction integrals, turbulence statistics (e.g., Kolmogorov or von Kármán models), propagation geometries (e.g., slant paths, satellite downlinks), and hardware configurations (e.g., transmitter/receiver apertures, adaptive optics)

can be incorporated. This approach also allows for parametric sweeps over variables such as wavelength, beam waist, turbulence strength, and wind speed, which enables the assessment of system performance under diverse conditions and to optimize design choices. In this section, the widely used numerical technique for simulating optical propagation is reviewed. Split-Step Fourier (SSF) Method is a method that alternates between applying phase perturbations from turbulence and propagating the beam over small distance segments in free space.

This simulators solves the *paraxial wave equation* (2.2.4) or the corresponding diffraction integrals in an efficient manner, allowing to capture diffraction, beam spreading, phase distortion, and scintillation effects. They are the core building blocks for higher-level simulations that incorporate dynamic turbulence (e.g., via frozen flow or Fried's extrusion), adaptive optics, and link budget analysis in free-space optical systems.

Split-Step Fourier Method

The *split-step Fourier* (SSF) method is widely used when refractive index fluctuations, denoted by $n_1(\mathbf{r})$ (Equation (2.1.3)), are present [30, 32, 31]. Here, n_0 is the background refractive index and $k = 2\pi/\lambda$ is the wavenumber. For a small spatial propagation step Δz, the paraxial wave equations (2.2.4) and (2.2.4) are defined in the following operations:

1. **Phase Shift from Refractive Index Fluctuations:**

   Over the step Δz, the field accumulates a phase by integrating the refractive

index variations

$$A(x, y, z + \Delta z) \leftarrow \exp\left[i\, k \int_{z}^{z+\Delta z} n_1(x, y, z')\, dz'\right] A(x, y, z).$$

2. **Free-Space Propagation (Diffraction):**

   Next, the effect of diffraction is computed by transforming the field to the spatial-frequency domain (k_x, k_y), applying the free-space propagation phase factor, and transforming back

$$\widetilde{A}(k_x, k_y, z + \Delta z) = \widetilde{A}(k_x, k_y, z)\, \exp\Big[-i\, \frac{\Delta z}{2\, k\, n_0}\, (k_x^2 + k_y^2)\Big],$$

   where $\widetilde{A}$ denotes the 2D Fourier transform of A in the transverse coordinates.

Combining these operations into a single propagation step gives

$$A(x, y, z{+}\Delta z) = \exp\Big[i\, k\, n_1(x, y, z)\, \Delta z\Big]\ \mathcal{F}^{-1}\left\{\exp\Big[-i\, \frac{\Delta z}{2\, k\, n_0}\, (k_x^2 + k_y^2)\Big]\ \mathcal{F}\{A(x, y, z)\}\right\}, \tag{2.2.26}$$

where $\mathcal{F}$ and $\mathcal{F}^{-1}$ denote the 2D Fourier and inverse Fourier transforms, respectively. By iteratively applying this step (i.e., $z \to z + \Delta z \to z + 2\,\Delta z \to \dots$), one approximates the continuous propagation of the optical field.

This numerical approach forms the backbone of many wave-optics simulations used in laser communications, adaptive optics, and beam shaping. In particular, the split-step Fourier method is highly effective for incorporating random phase fluctuations due to turbulence, where time-evolving phase screen realizations are generated in accordance with statistical models such as the Kolmogorov or von Kármán spectra [27, 38]. Detailed descriptions of split-step methods are available in Schmidt [32] and

Andrews and Phillips [21].

Angular Spectrum Method

The Angular Spectrum Method represents a specialized application of the split-step Fourier approach, that is particularly effective for free-space propagation problems. While the split-step method handles both refractive index variations and diffraction effects, the angular spectrum method focuses primarily on accurate diffraction modeling with variable sampling requirements.

Throughout this section, each "plane" (see Figure 2.9) refers to a transverse cross-section of the beam at a specific propagation distance z. In other words, the optical field along its path is discretized into a series of planes.

Following Schmidt formulation [32], the angular spectrum method can be expressed as a sequence of operations that account for propagation distance and changing spatial grid resolution.

In practice, the sampling grid must adapt to ensure the field is adequately sampled in space at each plane—especially if the size of the wavefront or curvature changes significantly, which can violate the Nyquist criterion if the original grid spacing were kept fixed.

For a field propagating through $n-1$ steps across n planes (see Figure 2.9), the

output field $U(\mathbf{r}_n)$ is given by

$$\begin{aligned} U(\mathbf{r}_n) = Q\left[\frac{m_{n-1}-1}{m_{n-1}\Delta z_{n-1}}, \mathbf{r}_n\right] \\ \times \prod_{i=1}^{n-1}\Bigg\{\mathcal{T}[z_i, z_{i+1}]\mathcal{F}^{-1}\left[\mathbf{f}_i, \frac{\mathbf{r}_{i+1}}{m_i}\right] Q_2\left[-\frac{\Delta z_i}{m_i}, \mathbf{f}_i\right] \\ \times \mathcal{F}[\mathbf{r}_i, \mathbf{f}_i]\frac{1}{m_i}\Bigg\} \\ \times \left\{Q\left[\frac{1-m_1}{\Delta z_1}, \mathbf{r}_1\right]\mathcal{T}[z_i, z_{i+1}]\, U(\mathbf{r}_1)\right\}, \end{aligned} \tag{2.2.27}$$

where $\mathbf{r}_i$ represents the spatial coordinates at plane i, $\mathbf{f}_i$ denotes the corresponding spatial frequencies, $\Delta z_i = z_{i+1} - z_i$ is the propagation distance between planes, and $m_i = \delta_{i+1}/\delta_i$ is the ratio of grid spacings. The term $\mathcal{T}[z_i, z_{i+1}]$ represents any transmission or phase screen effects between planes. The operators $\mathcal{F}[\mathbf{r}, \mathbf{f}]$ and $\mathcal{F}^{-1}[\mathbf{f}, \mathbf{r}]$ in Equation (2.2.27) represent the forward and inverse Fourier transforms, respectively, between the spatial domain $\mathbf{r}$ and frequency domain $\mathbf{f}$.

Operator Definitions The angular spectrum implementation relies on three key quadratic phase operators:

- $\mathbf{Q_1} = \exp\left[\mathrm{i}\,\alpha_1\, r^2\right]$ (spatial-domain operator). This initial operator applies a quadratic phase factor to the input field, with coefficient $\alpha_1 = \frac{k}{2} \cdot \frac{1-m_1}{\Delta z_1}$. This accounts for the change in sampling between the input plane and the next propagation step.

- $\mathbf{Q_2} = \exp\left[-\mathrm{i}\,\alpha_2\,(k_x^2 + k_y^2)\right]$ (frequency-domain operator). Applied in the spectral domain, this operator has coefficient $\alpha_2 = \frac{\pi^2 \cdot 2 \cdot \Delta z_i}{m_i \cdot k}$. It handles the propagation physics and incorporates the grid scaling factor m_i to maintain accuracy

across changing grid resolutions.

- $\mathbf{Q_3} = \exp\left[\mathrm{i}\,\alpha_3\, r^2\right]$ (spatial-domain operator). This final operator with coefficient $\alpha_3 = \frac{k}{2} \cdot \frac{m_{n-1}-1}{m_{n-1}\cdot\Delta z_{n-1}}$ imposes the concluding Fresnel phase curvature to ensure the output field has the correct phase profile at the target sampling rate.

The coefficients α_1, α_2, and α_3 collectively encode the propagation physics while handling the numerical challenges of variable sampling. Their calculation incorporates:

- The wavelength λ (through the wavenumber $k = 2\pi/\lambda$)
- Grid spacing ratios $m_i = \delta_{i+1}/\delta_i$ between successive planes
- Propagation distances Δz_i between planes

2.2.8 Phase Screen Representation of Turbulence

Atmospheric turbulence can be numerically simulated by introducing phase distortions to a propagating optical field at discrete intervals. These two-dimensional phase perturbations, known as *phase screens*, represent the cumulative phase shifts induced by refractive index fluctuations over a propagation segment Δz. Multiple screens can be cascaded to accumulate realistic turbulence-induced wavefront distortions along the propagation path [32].

The generation of phase screens that accurately represent atmospheric turbulence statistics requires careful consideration of both high and low spatial frequency components. The power spectral density (PSD) of phase fluctuations follows either Kolmogorov or von Kármán statistics, depending on the treatment of inner and outer

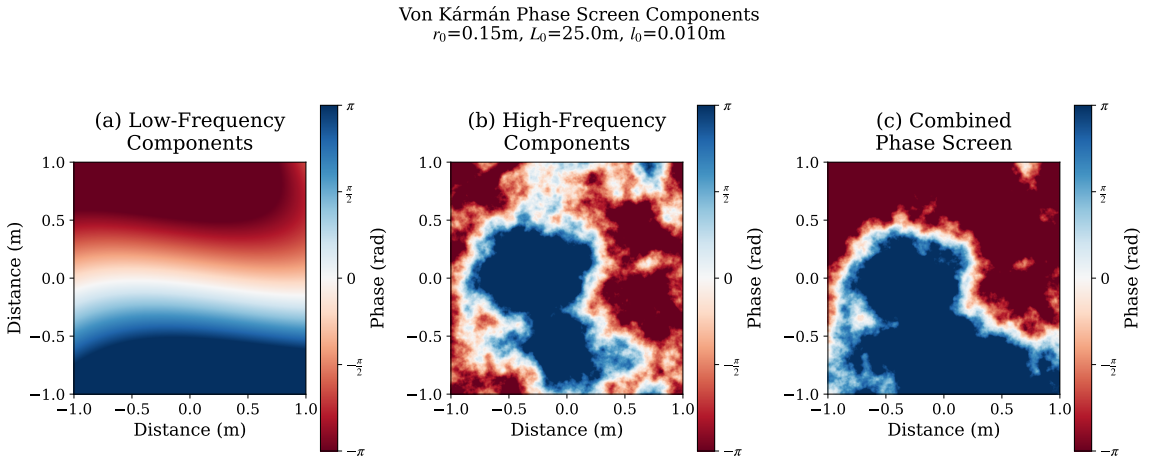


Figure 2.7: Decomposition of a von Karman atmospheric phase screen showing (a) low-frequency components from subharmonic compensation, (b) high-frequency components from the Fourier-based method, and (c) the combined phase screen. The subharmonic compensation adds the necessary large-scale turbulent structures that are underrepresented in pure Fourier methods. Phase values are shown in radians.

scales. For Kolmogorov turbulence, the phase PSD is given by

$$\Phi_\phi(\vec{\kappa}) = 0.023\, r_0^{-5/3}\, |\vec{\kappa}|^{-11/3}, \tag{2.2.28}$$

where $\vec{\kappa} = (\kappa_x, \kappa_y)$ represents the spatial frequency vector and r_0 is the Fried parameter characterizing the turbulence strength. The von Kármán spectrum provides a more complete model by incorporating the effects of inner scale l_0 and outer scale L_0

$$\Phi_\phi(\vec{\kappa}) = 0.023\, r_0^{-5/3}\, \frac{\exp(-|\vec{\kappa}|^2/\kappa_m^2)}{(|\vec{\kappa}|^2 + \kappa_0^2)^{11/6}}, \tag{2.2.29}$$

where $\kappa_m = 5.92/l_0$ and $\kappa_0 = 1/L_0$ define the cutoff frequencies associated with the inner and outer scales, respectively.

The numerical realization of these phase screens is typically accomplished through

Fourier transform techniques [27, 28, 32]. The method exploits the relationship between the power spectrum and spatial correlations by first generating complex Gaussian random noise in the frequency domain, $\mathcal{N}(0,1) + i\mathcal{N}(0,1)$, and then shaping its spectrum according to $\sqrt{\Phi_\phi(\vec{\kappa})}$. The inverse Fourier transform of this shaped spectrum yields a phase screen $\phi(x, y)$ whose statistical properties match the desired turbulence model.

However, the discrete nature of the numerical grid inherently limits the representation of low-frequency components, particularly those with scales approaching or exceeding the grid size. This limitation is addressed through subharmonic compensation, which augments the base phase screen with additional low-frequency terms [27, 28]

$$\phi_{\text{comp}}(x, y) = \phi(x, y) + \sum_{j=1}^{N_{\text{sub}}} \alpha_j \cos\big(k_{x,j}\, x + k_{y,j}\, y + \beta_j\big). \tag{2.2.30}$$

Here, each term in the summation represents a low spatial frequency component $(k_{x,j}, k_{y,j})$ sampled from the appropriate turbulence spectrum, with random amplitude α_j and phase β_j. This approach ensures proper representation of large-scale turbulent structures and improves agreement with theoretical models.

Figure 2.7 illustrates the decomposition of a von Kármán phase screen into its constituent components. The low-frequency contributions from subharmonic compensation (a) complement the high-frequency components generated by the Fourier method (b), resulting in a complete phase screen (c) that accurately represents both small and large-scale turbulent structures.

2.2.9 Taylor Frozen Flow Hypothesis

Introducing dynamic turbulence into phase screens requires an understanding of Taylor Frozen Flow Hypothesis, which simplifies the temporal modeling of the evolution of turbulence by assuming that turbulent eddies are carried along with minimal distortion by the mean wind field [65], meaning the turbulence pattern moves with the wind as if it were "frozen" in place. This simplifies modeling by treating time evolution as a spatial shift—typically implemented by shifting phase screens at speed v. This allows temporal fluctuations at a fixed point to be interpreted as spatial variations transported past the observer by the bulk flow.

Frozen flow states that the entire turbulent structure moves rigidly with the mean wind velocity $\mathbf{v}$, without distortion or decay. For a turbulent field $n(\mathbf{r}, t)$, this implies

$$n(\mathbf{r}, t + \Delta t) = n(\mathbf{r} - \mathbf{v}\Delta t, t), \tag{2.2.31}$$

enabling temporal statistics to be derived from spatial correlations via $\tau = \Delta x/v$, where $v = |\mathbf{v}|$ is the transverse wind speed.

The validity of Taylor hypothesis depends on several key constraints. First, turbulent velocity fluctuations must be small compared to the mean wind speed ($v' \ll v$), ensuring that self-evolution of the turbulent structure is slow. Second, the observation time interval must be much shorter than the characteristic time scale of the largest eddies ($T \ll L/v$, where L is the turbulence outer scale), limiting the opportunity for these large-scale structures to evolve significantly during measurement. Finally, the atmosphere must be approximately horizontally homogeneous without significant wind shear or stratification (without large changes in wind speed/direction with height or layers of differing temperature), as these conditions would distort the turbulent

structures, violating the "frozen" assumption [65, 66].

Frozen flow enables efficient numerical simulation of time-evolving turbulence for optical systems in two primary applications. In phase screen generation, dynamic turbulence effects are modeled by simply translating static phase screens $\phi(x, y)$ at each level, across the optical beam according to the wind velocity vector $\mathbf{v}$ [67]. This approach dramatically reduces computational complexity compared to generating independent screens for each time step, while maintaining the correct spatial and temporal statistics. For a phase screen moving at velocity $\mathbf{v} = (v_x, v_y)$, the time-evolved screen becomes

$$\phi(x, y, t) = \phi(x - v_x t, y - v_y t, 0), \tag{2.2.32}$$

where the initial screen $\phi(x, y, 0)$ is typically generated using Kolmogorov or von Kármán statistics.

The second major application concerns scintillation statistics, where the temporal intensity correlation function $C_I(\tau)$ is directly related to the spatial correlation through the wind velocity

$$C_I(\tau) = \left\langle \frac{\delta I(t)\delta I(t+\tau)}{\langle I \rangle^2} \right\rangle = C_I(v\tau), \tag{2.2.33}$$

where δI represents intensity fluctuations, $\langle \cdot \rangle$ denotes ensemble averaging, and $v\tau$ is the spatial separation induced by wind advection over time τ [66]. This relationship allows long-term intensity statistics to be predicted from spatial measurements or simulations, critical for evaluating the performance of free-space optical links under varying atmospheric conditions.

2.2.10 Temporal Evolution: Fried's Extrusion Method

In a dynamic satellite downlink scenario the propagating beam experiences changing turbulence as it sweeps an arc through the sky between the moving satellite and ground station. Taylor's frozen-flow hypothesis from Section 2.2.9 is used to map the temporal evolution between simulation time steps to a spatial evolution of the generated phase screens. As the simulation progresses in time the phase screens are shifted along in space to simulate transverse pseudowind. This then creates the challenge of needing to have very long phase screens for long time duration simulations, or otherwise some method of generating new phase screen data to replace old sections.

Fried's Column Extrusion Method is one solution to this problem. It updates a fixed size phase screen each time step in a way that preserves Kolmogorov statistics over space and time [37, 38].

Conceptual Overview

Fried's method incrementally "extrudes" new columns (or rows) of phase data at one edge of the screen, consistent with the existing phase structure in the rest of the screen. Figure 2.8 illustrates the process of incrementally appending single columns to an existing phase screen. The underlying equation to generate a new column is given by [38]

$$\mathbf{x}_{\text{new}} \;=\; A\left(\mathbf{z}-\mathbf{z}_{\text{ref}}\right) \;+\; B\,\mathbf{g} \;+\; \mathbf{z}_{\text{ref}}, \tag{2.2.34}$$

where $\mathbf{x}_{\text{new}}$ denotes the phase values to be appended, $\mathbf{z}$ is a set of samples (or "stencil") taken from the existing screen, $\mathbf{z}_{\text{ref}}$ is a single reference offset, $\mathbf{g}$ is a vector of uncorrelated Gaussian random draws, and A and B encode the covariance structure

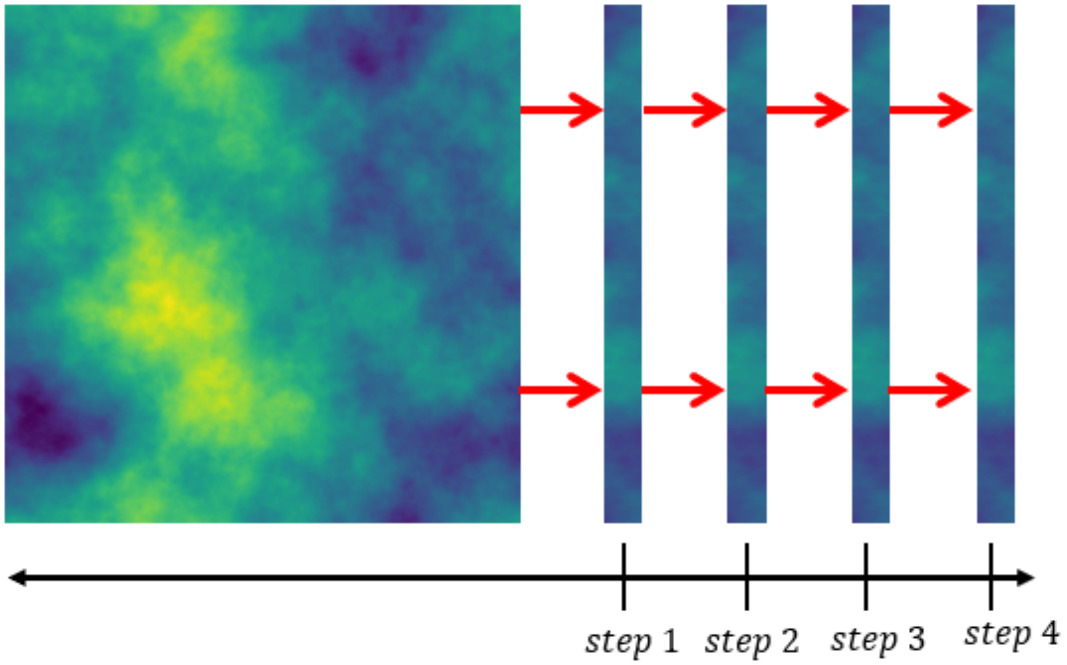


Figure 2.8: Fried column extrusion method for temporal evolution of phase screens. A complete phase screen (left) serves as the initial state, from which new columns are sequentially generated (right) in steps 1-4. Each new column maintains statistical correlation with the existing phase structure while extending the screen in the direction of evolution.

of the phase. To handle potentially large separations and avoid divergences in Kolmogorov statistics, the stencil selects elements from the final columns and one offset position, ensuring that once $\mathbf{x}_{\text{new}}$ is appended, the resulting extended phase screen maintains correct spatial correlations [38].

Appending $\mathbf{x}_{\text{new}}$ to the existing screen effectively shifts the screen in one dimension while maintaining the correct spatial correlations. At each time step enough new columns are added to account for the total translation for that phase screen layer wind speed v_i over the simulation step time Δt. For a layer pixel scale of Δx_i The number of columns to extend is given by $Nx_{new,i} = \lceil \frac{v_i \Delta t}{\Delta x_i} \rceil$.

Implementing Fried's Extrusion Method

1. **Precompute Covariances:**

   Compute the phase structure function $D_\phi(\Delta \mathbf{r})$ (or direct covariances, depending on model) for all relevant separations within the stencil region. From these, assemble the full covariance matrix corresponding to both the new points to be generated and the existing "stencil" points in the phase screen.

2. **Determine $\boldsymbol{A}$ and $\boldsymbol{B}$:**

   Partition the overall covariance matrix into sub-blocks for the new column (denoted $\mathbf{x}_{\text{new}}$) and the stencil points ($\mathbf{z}$)

   $$P_{xx} = \mathbb{E}[\mathbf{x}_{\text{new}}\, \mathbf{x}_{\text{new}}^T], \quad P_{zz} = \mathbb{E}[\mathbf{z}\, \mathbf{z}^T], \quad P_{xz} = \mathbb{E}[\mathbf{x}_{\text{new}}\, \mathbf{z}^T].$$

   Then define

   $$A \;=\; P_{xz}\, P_{zz}^{-1}, \quad B \;=\; \sqrt{P_{xx} - A\, P_{zx}},$$

Where $P_{zx} = P_{xz}^T$. Here, P_{xx} describes the covariance among all new sample points, P_{zz} is the covariance among all selected stencil points from the existing screen, and P_{xz} (together with P_{zx}) captures cross-covariances between the new sample points and the existing stencil points.

3. **Generate New Column:**

   Draw a random vector $\mathbf{g}$ of uncorrelated Gaussian samples. Then form (2.2.34) and append this new column to the existing grid.

4. **Repeat for Each Time Step:**

   After appending $\mathbf{x}_{\text{new}}$, treat it as part of the updated turbulence screen for the next time step. Applying different wind directions and speeds, or Fried parameters r_0 for separate layers allows multi-layer dynamics under the same extrusion framework.

In this way, the method "extrudes" columns of a phase screen while preserving the correct spatial correlations, even as the screen is extended or shifted for successive time steps.

2.2.11 Propagation Through Phase Screens

Once phase screens are generated, they are incorporated into *wave-optics* simulations using methods such as *split-step Fourier propagation* in (2.2.26) or references [30, 57, 32]). According to Figure 2.9, the propagation follows the following steps:

1. **Generate a phase screen** at discrete intervals Δz along the z-axis and according to that layer turbulence statistics.

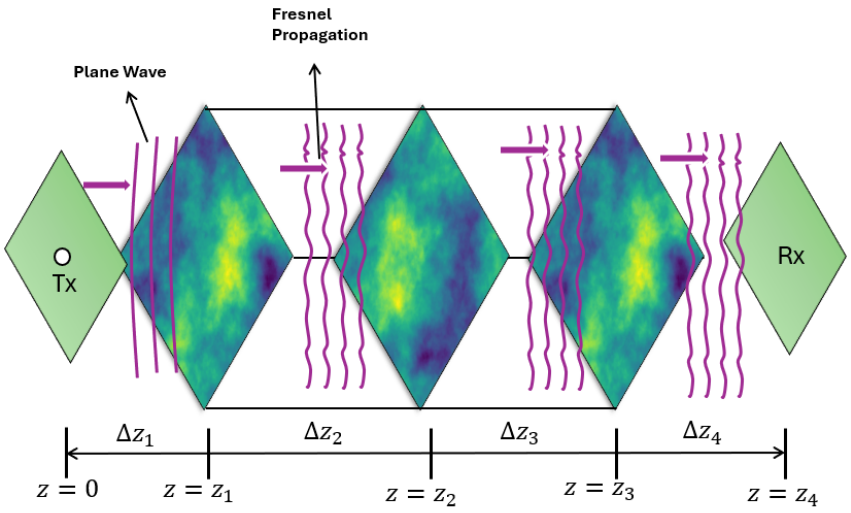


Figure 2.9: Split-step propagation through multiple phase screens. The optical field from a point source propagates through a series of phase screens (colored planes) representing atmospheric turbulence layers. Between screens, Fresnel propagation is applied over distances Δz_1, Δz_2, and Δz_3. The wavefront is progressively distorted as it encounters each turbulent layer.

2. **Multiply the optical field** by $e^{i\,\phi(x,y)}$ to impose turbulence-induced phase perturbations.

3. **Propagate the field** over the distance Δz (free space) to the next screen using angular-spectrum propagation methods.

4. **Repeat 2-3 until the receiver plane.**

As illustrated in Figure 2.9, this process shows how the input optical field encounters multiple phase screens and undergoes Fresnel propagation between them.

This *cumulative* approach captures how phase distortions build up over multiple layers, mimicking a real atmospheric channel.

2.3 Satellite Downlink Geometry

This section details the framework for computing the line-of-sight downlink channel geometry for a satellite in a circular orbit. Satellite communications take place over a range of *zenith angles* ζ, which defines the elevation of the satellite relative to the zenith angle of the observer. The effective atmospheric path length and turbulence distribution along the downlink channel changes as the satellite rises and falls over the ground station horizon.

Satellite Zenith Angle

For a satellite at altitude H above Earth's surface and an observer at ground level, the zenith angle ζ is determined by the satellite's orbital position relative to the observer (Figure 2.10). Using a circular orbit geometry, ζ relates to the satellite's elevation angle θ via

$$\zeta = 90^\circ - \theta, \tag{2.3.1}$$

where θ is calculated from the observer's latitude, satellite orbital parameters (e.g., inclination, altitude), and time-dependent position [68]. The *slant path length* through the atmosphere increases as $\sec(\zeta)$, significantly amplifying turbulence effects for low-elevation links ($\zeta \to 90^\circ$). This geometric scaling has profound implications for both phase distortions and scintillation effects.

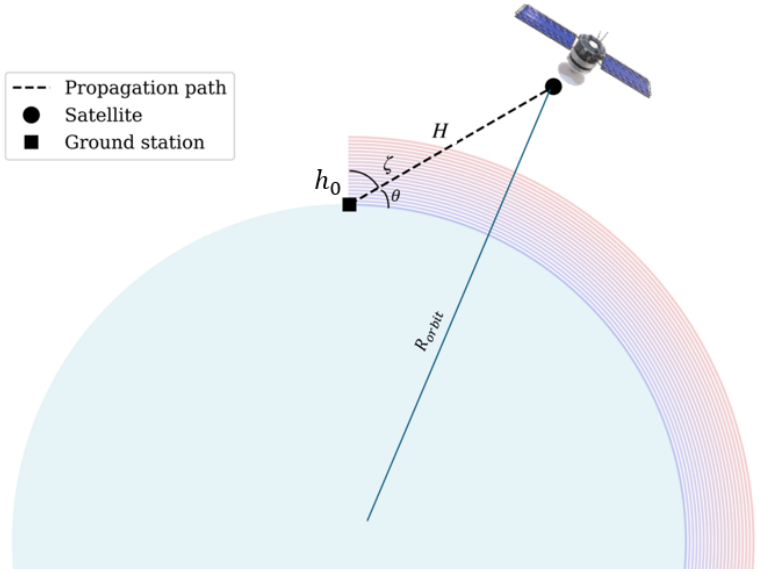


Figure 2.10: Satellite slant path geometry: H = higher orbit altitude, ζ = zenith angle, θ = elevation angle. Turbulence strength $C_n^2(h)$ varies along the path (color gradient). The effective propagation path (dashed line) scales as $L = \sec(\zeta) \cdot (H - h_0)$, where h_0 is observer altitude.

2.3.1 Satellite Link Geometry

For a satellite in a circular orbit passing directly overhead at altitude h_{orbit}, the orbital radius R from Earth's center is given by

$$R_{\text{orbit}} = R_{\text{Earth}} + h_{\text{orbit}} \tag{2.3.2}$$

where $R_{\text{Earth}} = 6.378 \times 10^6$ meters is Earth's radius. The orbital velocity for a perfectly circular orbit follows from the gravitational force balance

$$v_{\text{orbit}} = \sqrt{\frac{GM_{\text{Earth}}}{R_{\text{orbit}}}} \tag{2.3.3}$$

where $G = 6.67430 \times 10^{-11}$ m^3 kg^{-1} s^{-2} is the gravitational constant and $M_{\text{Earth}} = 5.972 \times 10^{24}$ kg is Earth's mass. Given an elevation angle γ_{horiz} measured from the horizon at the ground station, the line-of-sight distance L to the satellite can be determined by solving the quadratic equation

$$L^2 - 2R_{\text{Earth}} L \cos(\gamma_{\text{horiz}} + \frac{\pi}{2}) + (R_{\text{Earth}}^2 - R_{\text{orbit}}^2) = 0 \tag{2.3.4}$$

The angle γ_{Earth} from Earth's center to the line-of-sight vector is then

$$\gamma_{\text{Earth}} = \arccos\left(\frac{L^2 - R_{\text{orbit}}^2 - R_{\text{Earth}}^2}{-2R_{\text{orbit}}R_{\text{Earth}}}\right) \tag{2.3.5}$$

The slant range from ground station to satellite is computed using the law of cosines

$$d_{\text{range}} = \sqrt{R^2 + R_{\text{Earth}}^2 - 2RR_{\text{Earth}}\cos(\gamma_{\text{Earth}})} \tag{2.3.6}$$

The satellite's velocity vector can be decomposed into radial and tangential components relative to the line of sight. Given the angle ψ between the velocity vector and line of sight

$$\psi = \pi - |\gamma_{\text{Earth}}| - (\frac{\pi}{2} + \gamma_{\text{horiz}}) \tag{2.3.7}$$

The velocity components are

$$\begin{aligned} v_{\text{radial}} &= v_{\text{orbit}} \cos(\psi), \\ v_{\text{tangential}} &= v_{\text{orbit}} \sin(\psi). \end{aligned} \tag{2.3.8}$$

The tangential velocity component will appear as a psuedo wind speed in the turbulence temporal evolution, as in Eq. (2.2.24).

2.4 Atmospheric Layer Slicer

The atmospheric turbulence profile must be discretized into layers for numerical simulation. A layer-slicing algorithm based on a scintillation index criterion is implemented to distribute the turbulence path among the N discrete phase screens. This approach builds on the scintillation theory developed in Section 2.2.5.

Multiple methods exist for distributing turbulence power along the propagation path when discretizing the atmosphere into layers. One approach allocates equal integrated turbulence strength (integrated $C_n^2(h)$) to each layer [32], while another approach distributes the total path scintillation index among the layers according to Equation (2.2.25). The approach taken in this thesis is to implement the scintillation index distribution method although the architecture allows straightforward modification to implement either approach.

For a given atmospheric path, the total downlink scintillation index from equation (2.2.25) is divided into N equal parts to allocate turbulence effects across discrete propagation layers. Each layer boundary is set such that the scintillation is uniformly distributed between propagation layers:

$$\sigma_{Ii}^2 = \frac{1}{N}\sigma_I^2 = \frac{1}{N} 2.25 k^{7/6} \sec^{11/6}(\zeta) \int_{h_0}^{H_{\max}} C_n^2(h)(h - h_0)^{5/6}\, dh, \tag{2.4.1}$$

from the scintillation index Equation (2.2.25).

The C_n^2 profile follows the Hufnagel-Valley model from Equation (2.2.22), with the wind profile incorporating satellite motion

$$v_{\text{total}}(h) = \omega_s h + v_{\text{HV}}(h), \tag{2.4.2}$$

where ω_s is the satellite slew rate and $v_{\text{HV}}(h)$ is from equation (2.2.23).

The scintillation contribution from each layer is then equal

$$2.25 k^{7/6} \sec^{11/6}(\zeta) \int_{h_i}^{h_{i+1}} C_n^2(h)(h - h_0)^{5/6}\, dh = \frac{1}{N}\sigma_I^2, \tag{2.4.3}$$

for any layer i, ensuring uniform distribution of scintillation effects where N is the total number of layers. The layer boundaries are found iteratively by solving Equation (2.4.3). Starting from $h_0 = 0$, each subsequent boundary h_{i+1} is determined by matching the threshold value. This produces a set of layers that maintain equal scintillation contribution per layer while accounting for both $C_n^2(h)$ variation and geometric path effects. The method weights the contribution of scintillation from different altitude layers. Individual phase screens are placed at the lowest altitude in the range between layers h_i and h_{i+1}, or at h_i. While this approach produced the

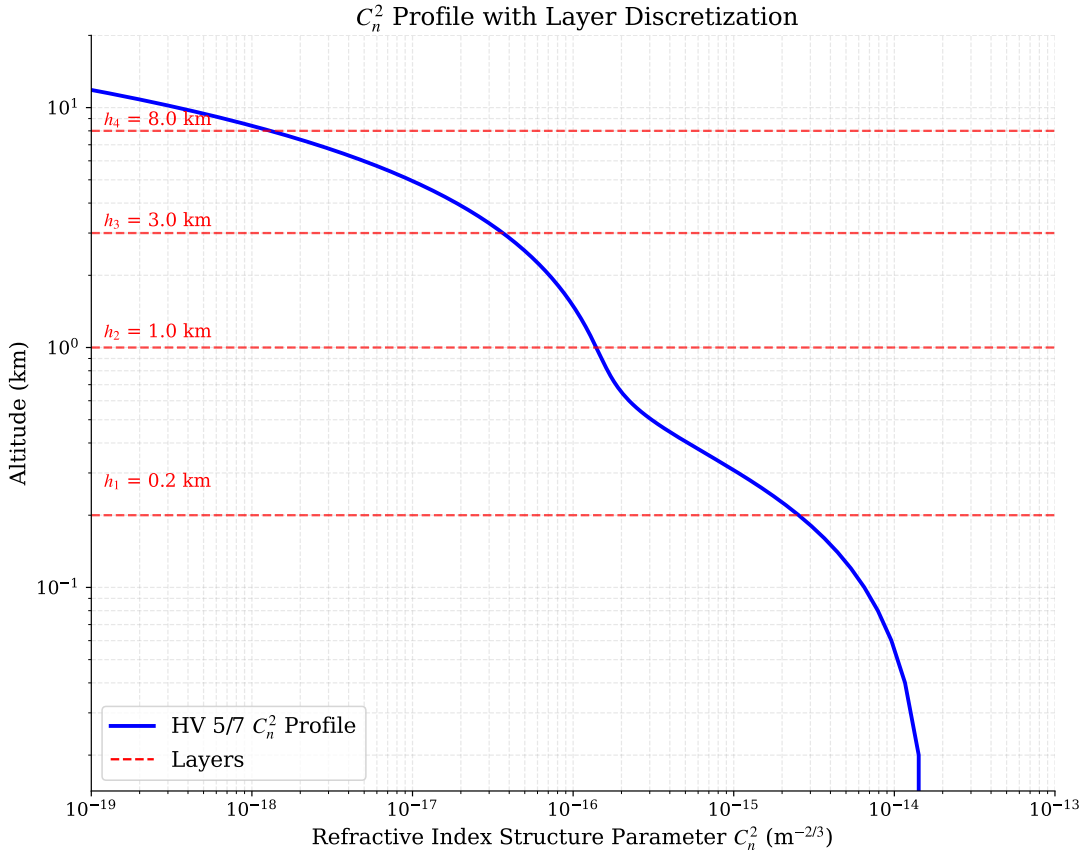


Figure 2.11: Illustration of layer slicer output with layer discretization (red dashed lines). This is achieved by solving Equation (2.4.3) and finding the layer heights. The layers are labeled as introduced in Equation 2.4.3

best matched scintillation statistics in simulation, but it not claimed to be optimal. Other placement strategies exist and could be easily implemented in the simulator architecture such as positioning phase screens at the weighted center of layer altitudes or at optimal heights based on specific observing conditions.

Figure 2.11 illustrates the concept of layer slicer using the C_n^2 profile with four example layers. The figure shows an example of how atmospheric turbulence varies with altitude and how layers are placed after solving Equation (2.4.3).

The resulting discretization enables efficient phase screen placement in the split-step propagation method described in Section 2.2.11. Each layer's parameters, including its r_0 value and wind speed, are computed from the local atmospheric conditions, providing a complete description of the turbulence structure for wave propagation simulations. In determining the optimal upper boundary for the layer slicing algorithm, 20 km is selected as the maximum effective altitude. The integrated $C_n^2(h)$ found via the Hufnagel-Valley model (and as described in Section 2.4) shows that nearly 99% of the integrated turbulence strength along the entire propagation path is contained in the first 20 km.

2.5 Numerical Propagation Simulation Workflow

The atmospheric turbulence simulation pipeline integrates multiple theoretical components into a cohesive numerical framework that captures both the spatial and temporal evolution of optical wave propagation, as illustrated in Figure 2.12 (pipeline operations) and Figure 2.13 (conceptual system). At its core, the simulation begins with the atmospheric parameters that define the turbulence environment. The $C_n^2(h)$ profile, combined with the propagation geometry, determines the fundamental turbulence parameters through path integration. This integration yields not only the Fried parameter r_0, which characterizes the overall turbulence strength, but also the spectral cutoffs κ_m and κ_0 that define the von Kármán spectrum's bounds (Eq. 2.2.29).

These parameters feed directly into the phase screen generation process, where the von Kármán spectrum shapes the spatial statistics of the turbulence. The phase screens themselves represent a crucial bridge between the statistical turbulence model

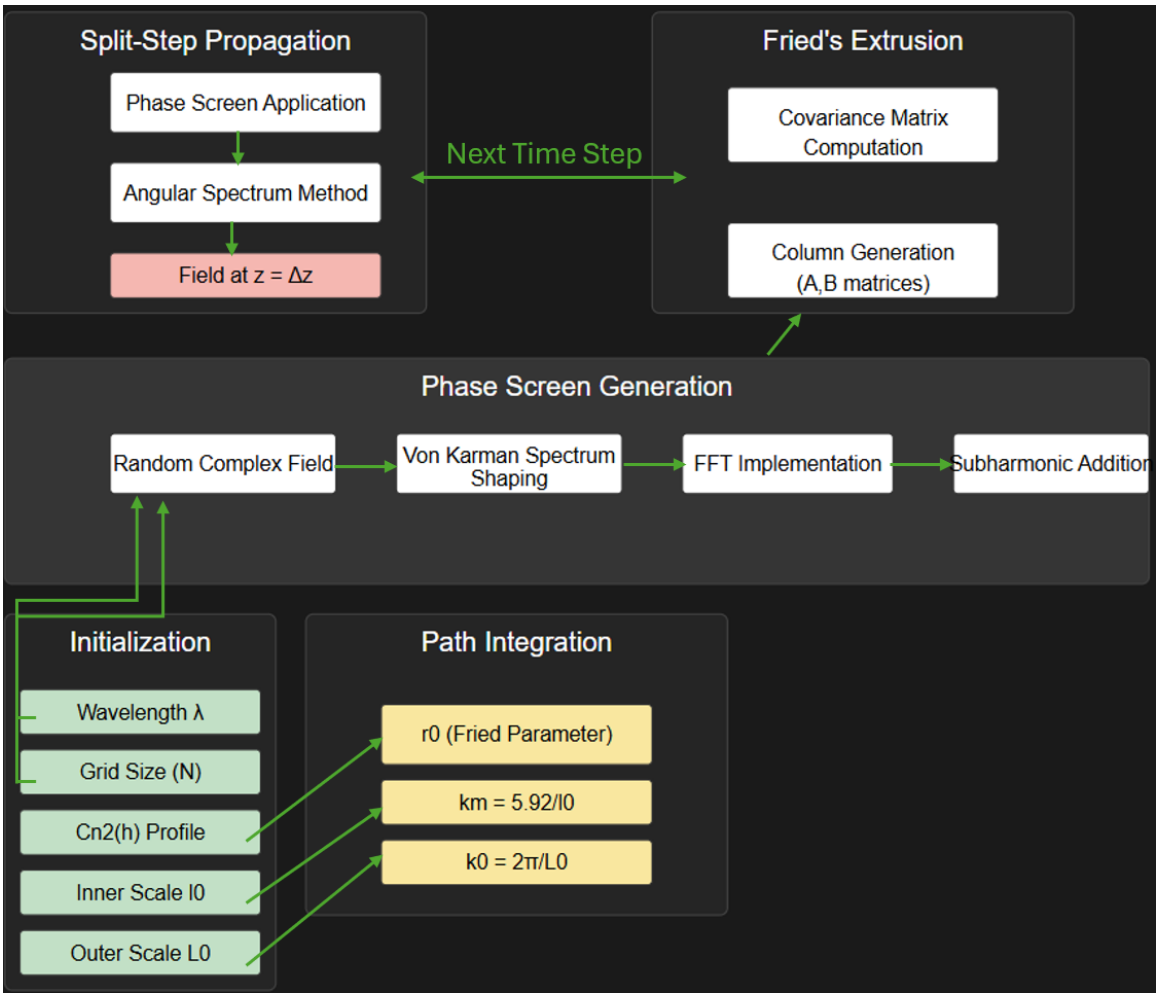


Figure 2.12: Comprehensive atmospheric turbulence simulation pipeline. The process begins with initialization parameters including the $C_n^2(h)$ profile and characteristic scales. Path integration yields the Fried parameter r_0 and spectral cutoffs κ_m, κ_0. Phase screen generation follows the von Kármán spectrum with proper subharmonic compensation for low-frequency components. Temporal evolution uses Fried's extrusion method, computing covariance matrices to generate new columns while preserving proper statistics. The split-step propagation applies phase screens and uses the angular spectrum method to simulate wave propagation through turbulence. This process iterates for each time step, creating a dynamic simulation of atmospheric effects on optical propagation.

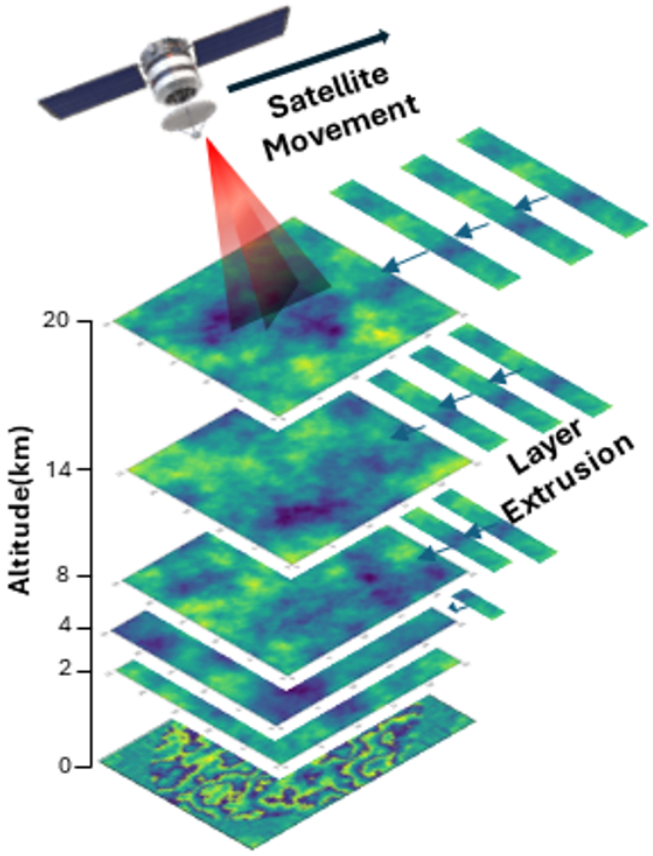


Figure 2.13: Conceptual illustration of a satellite downlink channel passing through multiple layers of atmospheric turbulence. Each colored layer represents a two-dimensional turbulence phase screen at a different altitude (with altitude increasing from bottom to top). As the satellite moves (indicated by the "Satellite Movement" arrow), new turbulence realizations can be introduced or "extruded" into the simulation at each layer ("Layer Extrusion" arrows), thereby capturing the continuous evolution of atmospheric turbulence in time.

and the wave propagation simulation. Each screen captures a snapshot of the refractive index fluctuations that an optical wave encounters along its path. The subharmonic compensation technique (Eq. 2.2.30) ensures that these screens accurately represent both small-scale distortions and the larger atmospheric structures that significantly impact beam wandering and focusing.

The temporal dimension of the simulation is realized through Fried's extrusion method (Eq. 2.2.34), which dynamically updates the phase screens while maintaining the correct spatial correlations prescribed by the turbulence model. This method creates a continuous evolution of the turbulent atmosphere, avoiding the limitations of simple frozen-flow approaches. The covariance matrices that guide this evolution ensure that new portions of the phase screen maintain statistical consistency with existing regions, creating a seamless representation of the dynamic atmosphere.

The split-step propagation technique then weaves these components together into the final simulation. As the optical field propagates through the atmosphere, it encounters the phase screens at discrete intervals. At each screen, the field accumulates phase distortions that represent the turbulence-induced refractive index fluctuations. Between screens, the angular spectrum method handles the diffraction effects that occur during free-space propagation (Eq. 2.2.26). This alternation between phase application and diffraction propagation captures the essential physics of how turbulence affects optical waves: phase distortions lead to amplitude fluctuations through diffraction, and these effects accumulate along the propagation path.

This integrated approach allows for the simulation of realistic atmospheric effects on optical beams, from the initial wavefront distortions to the final intensity and phase fluctuations at the receiver. The simulation captures essential phenomena such

as beam spreading, scintillation (Eq. 2.2.25), and wavefront distortion, providing a foundation for analyzing free-space optical communication system performance.

Chapter 3 explores how this computationally intensive pipeline can be efficiently implemented on GPU architecture, enabling the simulation of longer atmospheric paths and finer spatial resolution while maintaining reasonable computation times.

2.6 Computational Challenges and Scale

The numerical simulation of atmospheric turbulence presents computational challenges, primarily driven by the need for high spatial and temporal resolution. The fundamental sampling requirements need to be respected for optical propagation [32]. At the spatial level, both the smallest turbulent features (characterized by the inner scale l_0) and the largest structures (outer scale L_0) must be adequately sampled , while maintaining sufficient resolution to capture diffraction effects through the angular spectrum method.

For a typical atmospheric downlink scenario, the spatial sampling requirements are particularly demanding. The grid spacing Δx in the transverse plane must be small enough to resolve the finest phase fluctuations on the order of millimeters while simultaneously, the total grid size L must be large enough to contain the beam spread and wandering effects through the top layers of atmosphere over a potentially separated uplink path. In downlink/uplink simulations the downlink wavefront is used in an adaptive optics system to drive corrections on the uplink beam which is pointed up to 55μrad away due to the satellite velocity and round-trip travel time (the point-ahead angle). In such scenarios the simulation needs to cover beams spatially separated by up to one meter, in addition to the beam extent on the uplink and downlink.

Simulation geometries can easily balloon to 2048x2048 or greater [49] when trying to sample every layer at sub-cm scale over multiple meters.

For a beam that experiences turbulence-induced spreading, this often translates to grid sizes of at least 1024×1024 points, with many scenarios requiring 2048×2048 or even 4096×4096 points for adequate simulation fidelity, as explored in literature review [46, 49].

These large grid sizes directly impact the computational burden of the Fourier transforms required by both the angular spectrum propagation method and the phase screen generation. Each step of the split-step propagation requires two two-dimensional FFTs, and for $N \times N$ grids, each FFT operation scales as $O(N^2 \log N)$. Moreover, the total number of propagation steps must be sufficient to capture the continuous nature of atmospheric turbulence, typically requiring 10-20 or even up to 100 [46] phase screens layers for adequate sampling of the propagation path.

The temporal evolution of the turbulence compounds these computational demands. For accurate representation of dynamic effects such as scintillation and beam wander, the simulation time step must be shorter than the characteristic time scale of the turbulence, often on the order of milliseconds. Simulating just one second of real-time propagation might require thousands of time steps, each involving multiple FFT operations and phase screen updates through Fried's extrusion method.

2.7 Summary

This chapter presented an integrated approach to simulating free-space optical downlinks. It began by outlining the fundamentals of atmospheric turbulence through statistical models such as the Kolmogorov and von Kármán spectra, which describe

the cascade of energy and the resulting refractive index fluctuations. Key metrics—including the Fried parameter and scintillation index—were introduced to quantify turbulence effects on optical wavefronts.

The chapter then described how these atmospheric effects are incorporated into optical propagation theory, using the Huygens-Fresnel principle, paraxial approximation, and diffraction regimes (Fresnel and Fraunhofer). To bridge theory with practice, numerical techniques were detailed: phase screens are generated to represent refractive index variations (with subharmonic compensation for large-scale eddies) and are propagated using the split-step Fourier method. For dynamic scenarios, Fried's column extrusion method offers a more realistic evolution of turbulence over time than simple frozen-flow shifting.

Finally, computational challenges were addressed, emphasizing the high spatial and temporal resolution required for accurate simulations. Overall, the chapter establishes a comprehensive framework for evaluating how atmospheric turbulence degrades optical signal quality and system performance.

Chapter 3

GPU Implementation

3.1 GPU Architecture and Rationale for TensorFlow Integration

GPUs are specialized processors originally designed to render computer graphics in parallel. Unlike a typical CPU with a few cores optimized for serial tasks, a GPU contains thousands of smaller, more efficient cores tailored for large-scale parallel computations [69, 70]. This architecture makes GPUs well-suited for video processing as well as numerical simulations that can take advantage of data-level parallelism, as well as for artificial intelligence (AI) and machine learning (ML) applications that require processing massive datasets and complex mathematical operations simultaneously.

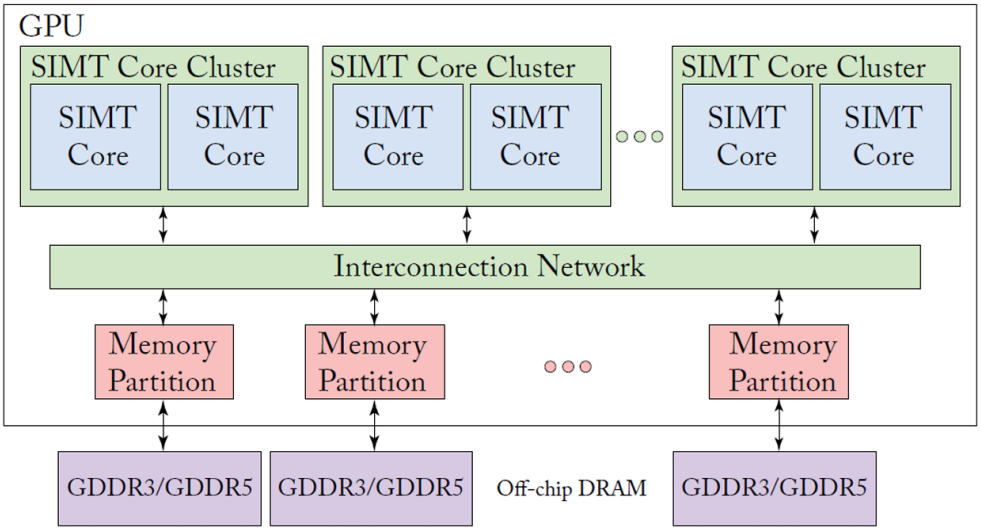


Figure 3.1: A generic modern GPU architecture showing the hierarchical organization with multiple SIMT Core Clusters, each containing SIMT Cores that execute parallel threads. The Interconnection Network facilitates communication between the computational units and memory partitions, which interface with high-bandwidth GDDR memory (adapted from [2]).

3.1.1 GPU Architecture

Modern GPU architectures implement a Single Instruction Multiple Thread (SIMT) execution model, allowing hundreds of threads to execute identical instructions simultaneously on different data elements. This architecture is particularly advantageous for simulations involving large numerical grids where the same calculations must be repeated across many data points [33].

The GPU architecture shown in Figure 3.1 illustrates the fundamental components that enable massive parallelism. At the highest level, the GPU contains multiple SIMT Core Clusters, each housing several SIMT Cores where the actual thread execution occurs. These cores are connected via an Interconnection Network to Memory Partitions that manage data flow between the cores and the off-chip DRAM (typically GDDR3/GDDR5 or newer memory technologies in current GPUs). This hierarchical

design allows thousands of threads to be scheduled and executed concurrently, making GPUs ideal for computationally intensive simulations with high data parallelism.

The memory hierarchy of GPUs is designed to support this massive parallelism through a tiered structure. Thread-local registers provide the fastest access times, while shared memory enables collaborative computing among thread blocks. The L1/L2 caches provide intermediate caching layers that reduce global memory access latency, and the high-bandwidth global memory serves as the main GPU memory store. This memory hierarchy creates opportunities for performance optimization across many scientific computing domains. Operations requiring nearest-neighbor access patterns can leverage shared memory to reduce redundant global memory accesses, while algorithms with regular access patterns benefit from the coalesced memory access optimizations available in modern GPU architectures [33].

3.1.2 General Computational Benefits

GPUs provide substantial performance advantages for computationally intensive tasks with inherent parallelism. Element-wise vector and matrix operations execute in parallel across thousands of threads, utilizing the full computational width of the GPU. Dense linear algebra operations map efficiently to GPU execution models, enabling fast matrix operations across scientific domains [71]. GPUs also feature high-bandwidth memory architectures that sustain rapid data throughput. Current generation GPUs offer memory bandwidth exceeding 900 GB/s, compared to typical CPU memory bandwidth of 50-100 GB/s. This order-of-magnitude difference directly accelerates the many matrix multiplication and FFT operations required for the numerical operation simulation as described in Section 2.6.

3.1.3 TensorFlow Framework Overview

TensorFlow is an open-source machine learning framework developed by the Google Brain team and first released publicly in 2015 [51]. Initially designed to support Google large-scale machine learning research, it has evolved into a comprehensive platform for scientific computing that extends well beyond its original machine learning focus. The framework operates around the concept of tensors—multi-dimensional arrays that flow through computational graphs—hence the name "TensorFlow". This approach enables efficient execution across diverse hardware platforms and large-scale distributed systems, with particularly optimized support for GPU acceleration [51]. TensorFlow core libraries are written in highly optimized C++ code, while providing accessible Python interfaces that have contributed to its widespread adoption. The framework offers robust support for both CPU and GPU execution, with specialized implementations of key mathematical operations that leverage hardware-specific optimizations. For GPUs, TensorFlow integrates with NVIDIA cuDNN and cuBLAS libraries to access highly tuned implementations of deep learning and linear algebra primitives, while also providing custom CUDA kernels for operations not covered by these libraries [51]. This integration leverages techniques described in [72] for efficient GPU application design, with particular attention to memory access patterns and computational intensity.

3.1.4 TensorFlow GPU Integration

TensorFlow leverages GPU capabilities by scheduling numerical kernels and data transfers in a way that minimizes idle compute time. Once data is placed in GPU memory, successive operations execute in parallel without frequent round-trips to

the CPU. This approach is ideal for computational pipelines with multiple successive operations that can remain resident in GPU memory, as noted in [51]. The framework memory management system builds upon handling of data locality and transfer minimization [69]. A computational graph in TensorFlow represents a program as a directed network where nodes are mathematical operations and edges are tensors (multi-dimensional arrays) flowing between them, enabling comprehensive optimization and efficient parallel execution. While graph optimization enhances computational efficiency, TensorFlow memory management capabilities further boost performance, particularly for GPU-accelerated scientific applications. The TensorFlow runtime constructs an optimized computation graph that automatically schedules operations to maximize GPU utilization and identifies opportunities for operation fusion [51]. This fusion capability combines multiple primitive operations into single optimized kernels, reducing kernel launch overhead and memory traffic.

One of the key strengths of TensorFlow is memory management for scientific computing applications. The framework implements a memory allocation system that manages buffer allocation and reuse to minimize memory fragmentation, and utilizes asynchronous execution that overlaps computation with data transfer when necessary [51]. For large-scale simulations with substantial memory requirements, the TensorFlow memory manager allocates persistent buffers when possible to reduce allocation overhead and employs tensor pooling to reuse memory buffers across operations. It also automatically places operations to minimize data transfer between CPU and GPU—a crucial optimization for maintaining high throughput in memory-bound applications.

The TensorFlow XLA (Accelerated Linear Algebra) compiler extends these capabilities by applying advanced optimizations specifically designed for numerical computations. TensorFlow XLA performs just-in-time compilation of TensorFlow operations into optimized machine code, applying techniques such as loop fusion, buffer reuse, and instruction scheduling.

3.1.5 Execution Modes in TensorFlow

TensorFlow provides two distinct execution paradigms—eager and graph execution—each offering different tradeoffs between development flexibility and runtime performance. According to the documentation and research papers on TensorFlow [51], these execution modes represent fundamentally different approaches to numerical computation orchestration, with significant implications for performance-critical applications.

Eager Execution

Introduced in TensorFlow 1.5 and made the default in TensorFlow 2.0, eager execution implements an imperative programming model where operations are executed immediately as they are called. Operations produce concrete values instead of constructing computational graphs for later execution, allowing developers to inspect results immediately using standard Python debugging tools. Variable states can be examined at any point during execution, and standard Python control flow statements work naturally without requiring TensorFlow-specific control flow operations.

The immediate feedback cycle of eager execution significantly accelerates the development process, particularly during the initial implementation phase. For the atmospheric propagation simulation, eager execution enabled rapid debugging of complex mathematical operations and verification of intermediate results during algorithm development. This immediate feedback was especially valuable when implementing the Fried column extrusion method, where subtle numerical errors could propagate through the simulation.

Despite these advantages, eager execution incurs performance penalties for large-scale computing. Each operation involves Python interpreter overhead, and the runtime cannot perform global optimizations across operation boundaries. For computationally intensive simulations like atmospheric wave propagation, these limitations become increasingly significant as problem size grows, particularly for the large matrices involved in turbulence modeling.

Graph Execution

Graph execution, the original execution model of TensorFlow, constructs a complete computational graph representing all operations before performing any calculations. Operations are defined and connected in a directed graph where nodes represent mathematical operations and edges represent the data tensors that flow between them. This graph is then optimized and executed as a unit, potentially across distributed computing resources [51]. This approach shares conceptual similarities with dataflow programming models described in high-performance computing literature.

Global optimization represents a primary benefit of graph execution. The compiler analyzes the entire dataflow graph to identify optimization opportunities across

operation boundaries [33]. This view enables more aggressive operation fusion and memory planning than is possible with line-by-line execution.

The graph execution model further enables parallel execution optimization, as the graph scheduler can automatically identify independent operations for parallel execution on multiple GPU cores [70]. Finally, portability benefits arise as compiled graphs can be serialized and executed on different platforms, facilitating deployment across diverse computing environments.

3.1.6 Rationale for Graph Execution in Atmospheric Simulation

For atmospheric turbulence simulation, in this work graph execution has been selected as the primary runtime mode due to its significant performance advantages for large-scale scientific computing.

The static compilation approach of graph execution proved ideally suited to the repetitive nature of split-step propagation, where the same sequence of operations is applied for each atmospheric layer and time step. By compiling this sequence into an optimized graph, the per-operation Python overhead is eliminated that would accumulate across thousands of operations in a typical simulation run.

While graph execution required additional upfront development effort—particularly in structuring operations to be compatible with tracing and handling dynamic aspects of the simulation—the substantial performance benefits justified this investment. The result is a high-performance atmospheric simulation framework capable of simulating atmospheric turbulence for fast moving LEO satellites optical propagation scenarios.

3.2 System Architecture

Building on the GPU-accelerated framework established, this section details the architectural strategies used to maximize computational efficiency in the atmospheric simulation pipeline. As previously discussed, TensorFlow graph execution and memory management capabilities provide a powerful foundation for optical wave propagation simulations. The system architecture presented here operationalizes those principles by carefully minimizing CPU–GPU data transfers, exploiting static tensor reuse, and isolating all performance-critical tasks to run exclusively on the GPU. The architectural strategy is based on maximizing precomputation for fixed simulation data for placement in the TensorFlow graph. Essential simulation parameters—such as spatial frequency grids and the power spectral density (PSD) of turbulence for phase screen generation, the Fresnel propagation operators Q_1, Q_2, and Q_3 (as defined in Section 2.2.7), and covariance matrices for Fried column extrusion are computed once on the CPU and stored as immutable TensorFlow constants on the GPU. These values, derived from equations such as the von Kármán PSD (Eq. (2.1.7)) and propagation operator definitions (Eq. (2.2.26)), remain fixed across simulation runs. This strategy eliminates unnecessary recomputation, reduces kernel launch overhead, and enables rapid iteration in Monte Carlo simulation loops.

Simulation control is done through a configuration file that specifies the orbital geometry (Section 2.3.1), atmospheric parameters (e.g., $C_n^2(h)$ profile from Eq. (2.2.22)), phase screen resolution, number of layers, and temporal sampling rate. This configuration allows users to modify simulation parameters without changing source code for adaptability and ease of use under different propagation scenarios.

Once initialized, the core simulation parameters like turbulence evolution and

wavefront propagation are instantiated through the compiled graph and passed as tasks to the GPU. Fried's column extrusion method (Section 2.2.10) updates turbulence screens based on Eq. (2.2.34), while the split-step Fourier propagator (Section 3.3.1) applies precomputed phase operators(Section 2.2.7) and FFT-based diffraction. Because all tensors involved in this process reside entirely in GPU memory, the simulation avoids the CPU-GPU bottlenecks that typically limit high-throughput numerical pipelines. Only low-bandwidth outputs, such as visualization frames and summary metrics, are returned to the CPU for inspection or logging; although transferring data from GPU to CPU always introduces overhead.

3.2.1 System Component Integration

The simulation architecture demonstrates a clear separation between CPU and GPU execution domains, with carefully orchestrated data flow and synchronization between components, as illustrated in Figure 3.2.

CPU Domain Components

The CPU domain, represented in the upper portion of the block diagram, serves as the initialization and parameter computation hub. At the start of simulation, it computes fundamental link parameters including satellite position $\mathbf{r}_{\text{sat}}$ and velocity $\mathbf{v}_{\text{sat}}$. The atmospheric turbulence layer slicer algorithm (Section 2.4) determines optimal layer heights h_i and their associated turbulence strengths $C_{n_i}^2$.

For each identified layer, the CPU generates initial phase screens $\phi_i(x, y)$ using the subharmonic compensation method detailed in Section 2.2.8. Supporting calculations include wind profile generation and geometric parameter computation based on the

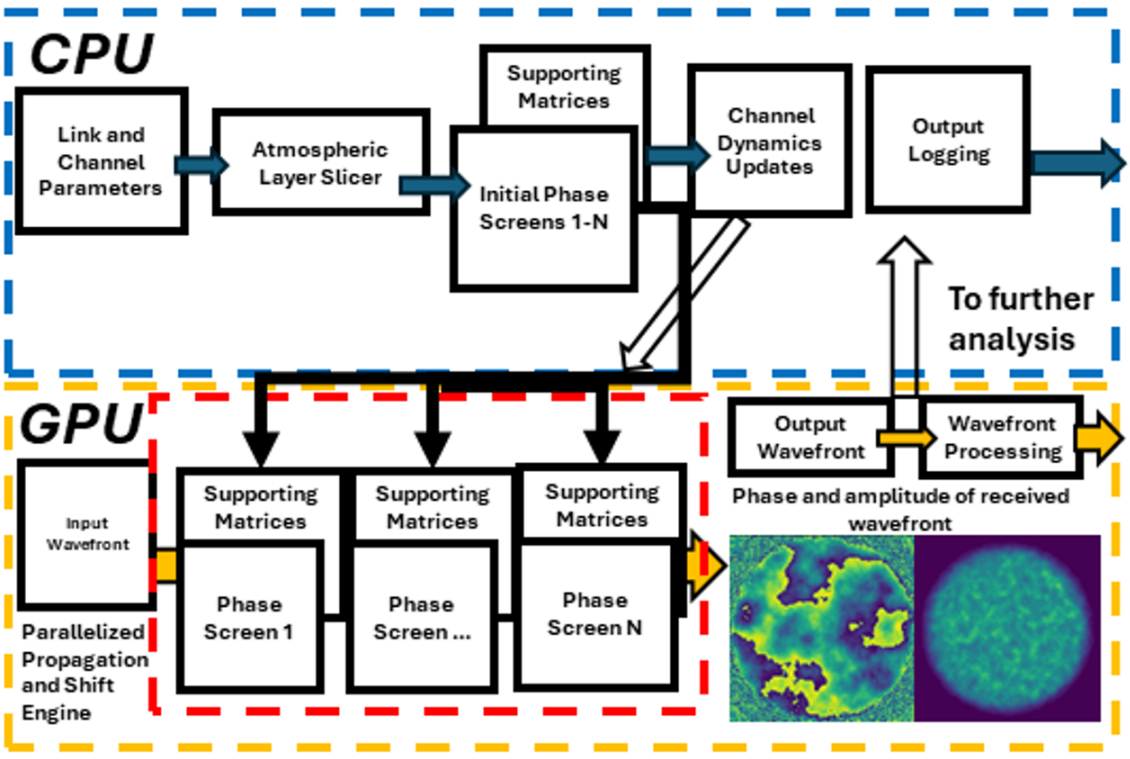


Figure 3.2: Block diagram of the atmospheric turbulence simulation architecture showing CPU and GPU domains. The CPU domain (top) handles initialization and parameter computation, while the GPU domain (bottom) manages parallel phase screen evolution and wave propagation. Dashed lines indicate domain boundaries, with arrows showing data flow direction.

satellite trajectory.

GPU Domain Organization

The GPU domain, highlighted in the lower section of Figure 3.2, implements the parallel processing pipeline for propagating wavefronts through each layer of the atmosphere from input to output, and evolving phase screens with wind speed. The phase screens evolve according to Fried method described in Section 3.3.2, while the wave propagation executes via the split-step algorithm detailed in Section 3.3.1.

Cross-Domain Data Flow

Data transfer between CPU and GPU is minimized to reduce overhead. Initial parameters and phase screens transfer from CPU to GPU during graph initialization. During simulation execution, the GPU primarily operates independently, with periodic transfers of performance metrics and visualization data back to the CPU for analysis and display.

Output Processing Pipeline

The simulation generates an output stream of propagated wavefronts as complex two dimensional tensors. The output wavefronts can be transferred back to the CPU domain for visualization of wavefront intensity and phase distributions. Further GPU accelerated processing can be included in the graph to extract statistical metrics and performance indicators for detailed post-simulation analysis.

This architecture enables high-performance parallel processing while maintaining synchronization between atmospheric evolution and wave propagation components.

The layered approach to phase screen management allows accurate representation of turbulence at different altitudes, while GPU acceleration ensures fast performance for dynamic simulation scenarios.

3.3 Simulation Engine Components

This section details the three core computational engines that comprise the GPU-accelerated atmospheric turbulence simulation framework.

3.3.1 Propagation Engine Architecture

GPU-accelerated simulator implements a TensorFlow graph-executed *split-step* propagator, which interleaves Fourier-based propagation steps and turbulence-induced phase shifts for each layer of the atmosphere.

Physical and Mathematical Overview

Under the paraxial approximation, a monochromatic optical field $A(x, y, z)$ propagating a distance Δz in a uniform medium can be approximated via Equation (2.2.26). The parameters Q_1, Q_2, and Q_3, Equation (2.2.27), are computed once and stored on the graphics processing unit (GPU) as tensors, since these values remain constant throughout the propagation calculations, making it computationally efficient to precompute and store them rather than recalculating at each step.

Building on the Fresnel diffraction integral from Equation (2.2.5), the numerical propagation can be implemented through phase screens using three fundamental

Algorithm 1 Split-Step GPU Propagation

Require: Initial field U_{in}, set of turbulence screens $\{\phi_{\text{turb}}^{(j)}\}_{j=1}^{N_{\text{layers}}}$, operators $\mathbf{Q_1}, \{\mathbf{Q_2}^{(j)}\}, \mathbf{Q_3}$, grid size N, wavelength λ.

Precomputation:
  Compute (k_x, k_y) from N and sampling interval δ.
  Upload $\mathbf{Q_1}, \{\mathbf{Q_2}^{(j)}\}, \mathbf{Q_3}$ to GPU memory. PropagateU_{in}, $\{\phi_{\text{turb}}^{(j)}\}$:
$U \leftarrow U_{\text{in}} \times \mathbf{Q_1}$ {Apply pre-propagation phase (spatial domain).}
for $j = 1$ **to** N_{layers} **do**
  $U \leftarrow U \times \exp\left[\mathrm{i}\,\phi_{\text{turb}}^{(j)}\right]$ {Turbulence phase for layer j.}
  $\tilde{U} \leftarrow \mathcal{F}\{U\}$ {Forward FFT (GPU).}
  $\tilde{U} \leftarrow \tilde{U} \times \mathbf{Q_2}^{(j)}$ {Multiply by frequency-domain transfer function.}
  $U \leftarrow \mathcal{F}^{-1}\{\tilde{U}\}$ {Inverse FFT (GPU).}
end for
$U \leftarrow U \times \mathbf{Q_3}$ {Final post-propagation phase.}
return U

operator blocks: the pre-propagation quadratic phase factor $\mathbf{Q_1}$ applied in the spatial domain to prepare the field for efficient sampling, the frequency-domain transfer function $\mathbf{Q_2}$ which directly implements the Fresnel propagation kernel from Equation (2.2.5) to capture the fundamental diffraction physics over step Δz, and the post-propagation quadratic phase factor $\mathbf{Q_3}$ that ensures consistent sampling at the output plane and completes the Fresnel transformation. These operators work in conjunction with the phase screens $\phi(x, y)$ generated according to the von Kármán spectrum (Equation (2.1.7)).

Algorithmic Description

In practice, each iteration in Algorithm 1 corresponds to a propagation layer or slice (see 2.12). For a downlink scenario, layers are reversed to simulate a beam traveling from high altitude to ground. All arrays are stored as complex-precision

tensors (e.g., `complex64`) on the GPU. The result is a high-throughput simulation routine for large grids and multiple layers since all layer data is already on the GPU. Subsequent sections show how these propagation steps work seamlessly with GPU-based turbulence generation (Fried's extrusion) to form a comprehensive downlink simulator.

3.3.2 Phase Screen Shifter Engine Architecture

The temporal evolution of atmospheric turbulence requires both initial phase screen generation and dynamic updates. This implementation combines CPU-based initial screen generation using the subharmonic method (Section 2.2.8) with GPU-accelerated screen evolution using Fried column extrusion technique (shown in Algorithm 2).

In each extrusion step, existing pixels are gathered from the trailing edge of the phase screen [38], and the new column (or row) is generated by combining these with a random Gaussian vector via matrices $\mathbf{A}$ and $\mathbf{B}$ (discussed in Equation (2.2.34)). This ensures that the newly appended region remains properly correlated with the existing screen, preserving the target turbulence statistics.

Initial Phase Screen Generation

For each atmospheric layer identified by the layer slicer, an initial phase screen is generated on the CPU using the von Kármán spectrum with subharmonic compensation as described in equation (2.1.7). The process follows equation (2.2.30), where low-frequency components are added to the base screen to ensure proper representation of large-scale turbulent structures. These screens are then transferred to GPU

Algorithm 2 GPU Column Extrusion via Fried's Method

Require: Phase screen $\mathbf{\Phi}$ (a `tf.Variable` of shape (height, width)), stencil coordinates $\mathcal{S}$, matrices $\mathbf{A}, \mathbf{B}$, random generator, *height* = number of rows in $\mathbf{\Phi}$.

Procedure: AddRow($\mathbf{\Phi}, \mathbf{A}, \mathbf{B}$)
  $\boldsymbol{\phi}_{\text{old}} \leftarrow tf.gather\_nd(\mathbf{\Phi}, \mathcal{S})$
  $\mathbf{g} \leftarrow tf.random.normal(\text{shape} = (\text{width}))$
  $\boldsymbol{\phi}_{\text{new}} \leftarrow \mathbf{A}\,\boldsymbol{\phi}_{\text{old}} + \mathbf{B}\,\mathbf{g}$
  $\boldsymbol{\phi}_{\text{new}} \leftarrow tf.reshape(\boldsymbol{\phi}_{\text{new}}, (1, \text{width}))$
  $\mathbf{\Phi} \leftarrow tf.concat\big[\boldsymbol{\phi}_{\text{new}}, \mathbf{\Phi}\big]$
  $\mathbf{\Phi} \leftarrow \mathbf{\Phi}[0 : \text{height}, 0 : \text{width}]$
  return $\mathbf{\Phi}$

memory for dynamic evolution.

3.3.3 Phase Screen Generator Engine

To support efficient Monte Carlo simulations, a GPU-accelerated phase screen engine was developed. This engine implements a graph-based algorithm that generates complete phase screens entirely on the GPU. By annotating the generation process with `@tf.function`, TensorFlow traces the function once, converts it into a static graph, and caches the resulting tensors [73]. Consequently, all computations (including the generation of frequency grids and phase PSD) are performed only once and stored as TensorFlow constants in the graph, eliminating repeated overhead in subsequent simulation steps. The underlying turbulence model, including the Kolmogorov and von Kármán PSD formulations, is described in Section 2.2.8, with equations provided in Equation (2.2.28) for the Kolmogorov spectrum and Equation (2.2.29) for the von Kármán spectrum. Subharmonic compensation used to restore low-frequency content is implemented according to Equation (2.2.30).

An outline of this GPU-based phase screen generation is provided in Algorithm 3.

Algorithm 3 GPU-based Phase Screen Generation

Require: Coherence parameter r_0, grid size N, grid spacing δ, outer scale L_0, inner scale l_0

Precompute:

- Compute frequency grid: $f = \sqrt{f_x^2 + f_y^2}$.
- Set $f_m = \frac{5.92}{2\pi\, l_0}$ and $f_0 = \frac{1}{L_0}$.
- Calculate phase PSD: $\mathrm{PSD}_\phi = 0.023\, r_0^{-5/3}\, \dfrac{\exp\left[-\left(\frac{f}{f_m}\right)^2\right]}{\left(f^2 + f_0^2\right)^{11/6}}$.
- Store these values as TensorFlow constants.

Generate a random complex field: $\xi \in \mathbb{C}^{N\times N}$.

Compute high-frequency phase: $\phi_{\mathrm{hi}} \leftarrow \Re\Big\{\mathcal{F}^{-1}\Big[\sqrt{\mathrm{PSD}_\phi}\cdot\xi\Big]\Big\}$.

for each subharmonic grid p **do**

  Generate random coefficients: $\xi^{(p)} \in \mathbb{C}^{3\times 3}$.

  Compute low-frequency contribution:

$$\phi_{\mathrm{lo}}^{(p)} \leftarrow \sum_{i,j} \xi_{i,j}^{(p)}\, e^{2\pi \mathrm{i}\,(f_x^{(p)} x + f_y^{(p)} y)}.$$

end for

Accumulate low-frequency phase: $\phi_{\mathrm{lo}} \leftarrow \sum_p \phi_{\mathrm{lo}}^{(p)}$.

Subtract mean: $\phi_{\mathrm{lo}} \leftarrow \phi_{\mathrm{lo}} - \mathrm{mean}(\phi_{\mathrm{lo}})$.

Combine components: $\phi \leftarrow \phi_{\mathrm{hi}} + \phi_{\mathrm{lo}}$.

return Final phase screen ϕ.

Additionally, the engine includes functionality to save the computation graph (using TensorBoard) for debugging and further optimization.

3.4 Numerical Stability Constraints Investigation

3.4.1 Covariance Matrix Stability Constraint

The phase screen shifting process relies critically on the stability of the covariance matrix decomposition [71, 38]. A fundamental limitation was found when exploring the relationship between the outer scale L_0 and the pixel scale Δx_i.

3.4.2 Source of Numerical Stability in Fried Extrusion method

To better understand this condition, the following mathematical expressions must first be examined. For a von Kármán turbulence model, the phase covariance matrix elements are given by

$$C_{ij} = \sigma^2 \left(\frac{r_{ij}}{L_0} \right)^{5/6} K_{5/6} \left(\frac{2\pi r_{ij}}{L_0} \right) \tag{3.4.1}$$

where r_{ij} is the separation between points i and j, and $K_{5/6}$ is the modified Bessel function of the second kind.

The Cholesky decomposition of the covariance matrix $\mathbf{C}_{zz}$ requires [74]:

$$\mathbf{C}_{zz} = \mathbf{L}\mathbf{L}^T. \tag{3.4.2}$$

Here, $\mathbf{C}_{zz}$ is assembled so that each entry $\mathbf{C}_{zz}[i, j]$ equals $C_{ij} = \mathrm{Cov}(\phi_i, \phi_j)$ for the

sample points ϕ_i and ϕ_j, ensuring the matrices $\mathbf{A}$ and $\mathbf{B}$ preserve the correct turbulence statistics across all pairs of pixels.

For the matrix inversion of C_{zz}^{-1} to be numerically stable, the condition number of $\mathbf{C}_{zz}$ must satisfy

$$\kappa(\mathbf{C}_{zz}) = \|\mathbf{C}_{zz}\|\|\mathbf{C}_{zz}^{-1}\| \leq \epsilon^{-1}, \tag{3.4.3}$$

where ϵ is machine precision [75]. This condition provides a practical guideline for configuring simulations: by ensuring that the condition number of $\mathbf{C}_{zz}$ remains below ϵ^{-1}, numerical stability can be maintained. In practice, this is achieved by constraining the ratio of the outer scale L_0 to the pixel spacing Δx_i, as discussed in the following section.

3.4.3 Critical $L_0/\Delta x_i$ Ratio

The condition number depends critically on the ratio $L_0/\Delta x_i$. Through the simulations, it was found that stability requires

$$\frac{L_0}{\Delta x_i} \leq \alpha_{\text{crit}}, \tag{3.4.4}$$

where α_{crit} was found to be between 1000-1500 based on repeated testing. This value will depend on numerical precision of the implemented hardware.

3.4.4 Implementation Impact

When the $L_0/\Delta x_i$ ratio exceeds the critical value, the covariance matrix becomes ill-conditioned, leading to failure in the Cholesky decomposition:

$$\text{If } \frac{L_0}{\Delta x_i} > \alpha_{\text{crit}} : \begin{cases} \det(\mathbf{C}_{zz}) \approx 0 \\ \text{Matrix inversion fails} \\ \text{A and B matrices undefined} \end{cases} \tag{3.4.5}$$

To maintain numerical stability with the implemented column extrusion method, in practice it was fond that the simulation configuration should limit the ratio of pixel scale to outer scale to

$$\Delta x_i \geq \frac{L_0}{2500}. \tag{3.4.6}$$

The ratio in 3.4.6 was chosen as a heuristic, based on results of simulation attempts with several pixel to outer scale ratios. For example, for a pixel scale of $\Delta x_i = 1$ cm was found to remain stable up to an outer scale of $L_0 = 25$ m. This limitation is also mentioned in HCIPy documentation.

3.5 Conclusion

This chapter describes a GPU-accelerated framework for simulating satellite-to-ground optical propagation through atmospheric turbulence. The implementation achieves several key innovations:

The TensorFlow-based architecture demonstrates significant advantages over traditional CPU implementations. By leveraging Graph Execution mode, efficient kernel fusion and memory management for both the propagation engine and phase screen evolution was achieved . The split-step propagator, optimized through XLA compilation, handles many atmospheric layers and large grid sizes while maintaining numerical stability across large propagation distances.

The phase screen shifter engine, implementing Fried column extrusion method on GPU, enables temporal turbulence evolution while preserving correct spatial statistics. The fundamental $L_0/\Delta x_i$ stability constraint was explored through testing, with a reasonable range of $\alpha_{crit} > 1500$ values providing clear guidelines for maintaining numerical robustness.

The unified CPU-GPU execution strategy minimizes data transfer overhead through strategic precomputation and partitioning of tasks. Initial parameter calculation and screen generation occur on the CPU, while intensive propagation and screen evolution execute entirely on GPU. This architecture enables high fidelity extended simulations of dynamic atmospheric effects.

Chapter 4 verifies the operation of the simulator and validates its statistical performance.

Chapter 4

Simulation Results and Validation

4.1 Validation of Numerical Implementation

The accuracy of the GPU-accelerated simulation framework was validated against theoretical predictions, with particular focus on the scintillation index as a key metric for atmospheric turbulence effects. Using the simulation parameters detailed in Section 4.1.1, a systematic validation across various Fried parameter values is conducted.

4.1.1 Solved Geometry and Layer Distribution

The simulation framework was validated and compared against HcIPy [24] outputs, using a representative Low Earth Orbit (LEO) satellite downlink scenario. Table 4.1 presents the key simulation parameters used in this configuration. A timestep is a configurable parameter of the simulator and can be set shorter for higher fidelity time-series simulations.

Table 4.1: Simulation Parameters

Parameter	Value	Units
Satellite altitude (h_{orbit})	600	km
Number of atmospheric layers (N_{layers})	5	–
Total Fried parameter (r_0)	20	cm
Wavelength (λ)	1.55	μm
Outer scale (L_0)	25	m
Ground wind speed (v_{ground})	5	m/s
Grid resolution (δ)	1	cm
Phase screen size	256×256	points
Time step	1	ms
Zenith angle	0	degrees

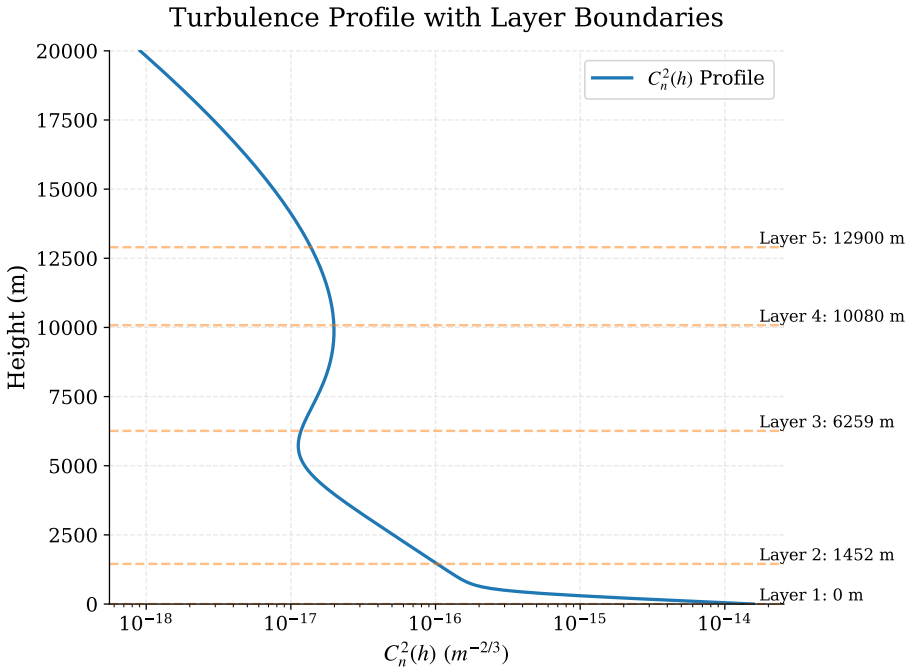


Figure 4.1: Altitude distribution of the five atmospheric layers overlaid on the $C_n^2(h)$ profile. Layer boundaries are positioned at 1.45, 6.26, 10.08, 12.90, and 20.02 km. The color intensity represents the relative turbulence strength within each layer, ranging from 1.03×10^{-16} m$^{-2/3}$ near the ground to 9.00×10^{-19} m$^{-2/3}$ at the highest layer.

Table 4.2: Layer Distribution and Parameters

Layer	Height (km)	r_0 (cm)	Wind Speed (m/s)
1	1.45	22.0	25.2
2	6.26	95.2	103.4
3	10.08	162.8	161.4
4	12.90	190.4	185.1
5	20.02	218.9	257.4

Note: The individual layer r_0 values are:

$$r_0^{(i)} = \begin{cases} 22.0 \text{ cm} & \text{Layer 1} \\ 95.2 \text{ cm} & \text{Layer 2} \\ 162.8 \text{ cm} & \text{Layer 3} \\ 190.4 \text{ cm} & \text{Layer 4} \\ 218.9 \text{ cm} & \text{Layer 5} \end{cases} \tag{4.1.1}$$

with a total combined Fried parameter of $r_0 = 20$ cm.

Layer Distribution Analysis

The atmospheric path was discretized into five layers using the Rytov variance criterion from equation (2.4.1). The simulation yielded the following layer distribution (Table 4.2 and Figure 4.1) and parameters (Table 4.3):

The C_n^2 profile values and number of extrusions required per time step (i.e., 1 ms) for each layer are presented in Table 4.3. The wind speed at different altitudes is calculated based on Bufton wind model (Equation 2.2.23), and C_n^2 trajectory of the satellite path has the ground wind speed incorporated through Equation 2.2.22. The relative speed of layer evolutions is based on Equation 2.2.24.

This layered representation provides the foundation for the phase screen generation and propagation analysis presented in subsequent sections. The parameters of each layer were carefully chosen to maintain numerical stability (see Section 3.4) while accurately representing the atmospheric turbulence effects on the optical downlink.

Table 4.3: C_n^2 Values for Each Atmospheric Layer

Layer	C_n^2 ($m^{-2/3}$)	N_{ext}
1	1.03×10^{-16}	3
2	1.17×10^{-17}	10
3	1.98×10^{-17}	16
4	1.37×10^{-17}	19
5	9.00×10^{-19}	26

4.1.2 Scintillation Index Validation

The scintillation index validation was performed for the link geometry in Table 4.1. 1000 wavefronts were propagated through the multiple phase screen layers and the intensity at each output was saved. The vector of intensities was used to estimate the scintillation index according to Equation (2.2.20), and compared to the theoretical predictions of Equation (2.2.25).

Results are shown in Figure 4.2, the GPU-accelerated simulation demonstrates varying levels of agreement with theoretical predictions depending on the turbulence strength. Here r_0 shows the whole path coherence length, which the simulator finds an r_0 profile for each layer (as one example seen in Equation 4.1.1). For whole path r_0 values between 15 cm and 30 cm, representing moderate to weak turbulence conditions, the simulation shows reasonable agreement with theoretical predictions, maintaining relative errors below 5%. For instance, at $r_0 = 20$ cm, the simulation produced a scintillation index of 0.064, closely matching the theoretical prediction of 0.068.

For r_0 values below 15 cm, representing stronger turbulence conditions, the discrepancy between simulation and theory increases significantly, reaching up to 20% difference. This increased deviation can be attributed to several factors:

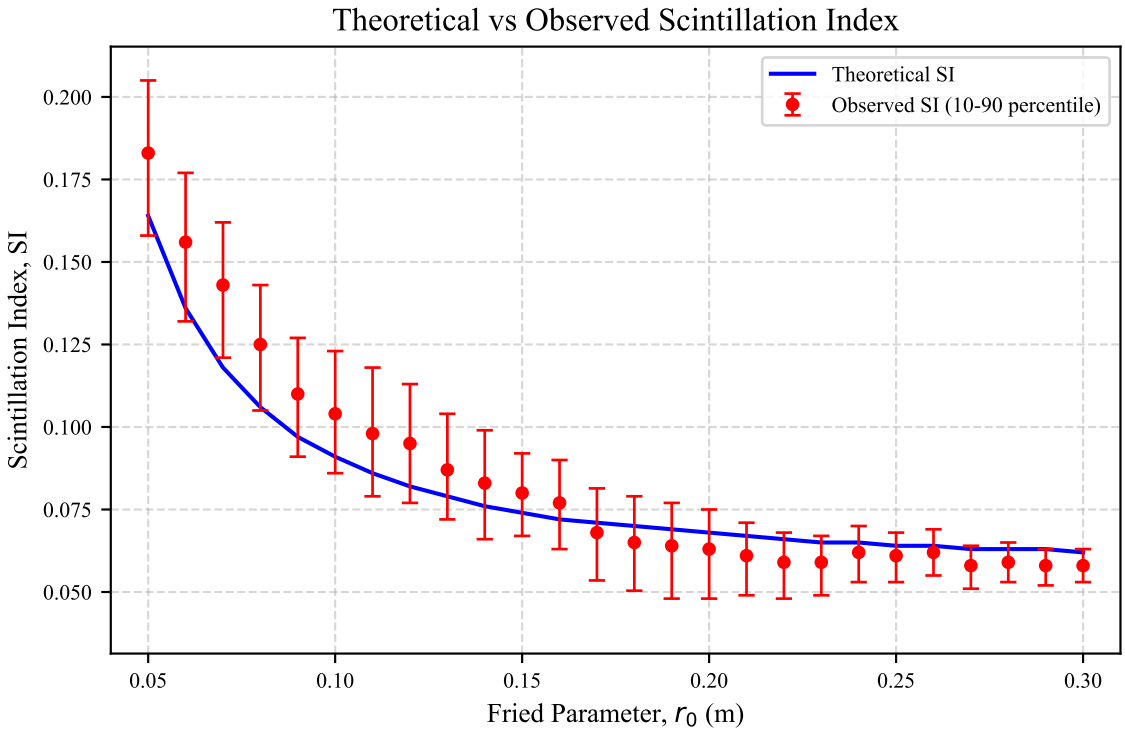


Figure 4.2: Comparison of theoretical and observed scintillation indices versus Fried parameter values. The blue line represents theoretical predictions from equation (2.2.25), while red markers with error bars show observed measurements. The error bars represent the 10-90 percentile range derived from 1000 distinct propagation simulations, with the marker indicating the mean values. Close agreement is observed for $r_0 \geq 15$ cm, with increasing deviation at smaller r_0 values corresponding to stronger turbulence conditions.

- **Phase screen resolution limitations:** At smaller r_0 values, the turbulent structures become finer, approaching the grid resolution of 1 cm. Fixed grid sampling does not adequately sample these smaller turbulent features leading to under representation of high-frequency components in the phase screens.

- **Rytov approximation breakdown:** The theoretical predictions are based on the Rytov approximation, which assumes weak fluctuation conditions. As r_0 decreases below 15 cm, simulation enters the strong fluctuation regime where this approximation becomes less accurate.

The validation results demonstrate that the GPU-accelerated implementation successfully reproduces the expected scintillation behavior for moderate to weak turbulence conditions. The close agreement with theoretical predictions, particularly in the regime relevant to satellite-to-ground optical communications ($r_0 \geq 15$ cm), confirms the accuracy of the numerical approach.

4.1.3 Fried Parameter Validation Using the Mutual Coherence Function

The Mutual Coherence Function (MCF) from Section 2.2.4 can be used to validate the turbulence simulation by estimating the value of r_0 on the output wavefronts using Equation (2.2.18). Multiple independent realizations of turbulence were generated and wavefronts propagated through to the output. For each realization, the complex optical field at the receiver was obtained, and the two-dimensional MCF was calculated by correlating the field with a spatially shifted copy of itself. This correlation was normalized by the mean intensity to yield a proper measure of coherence.

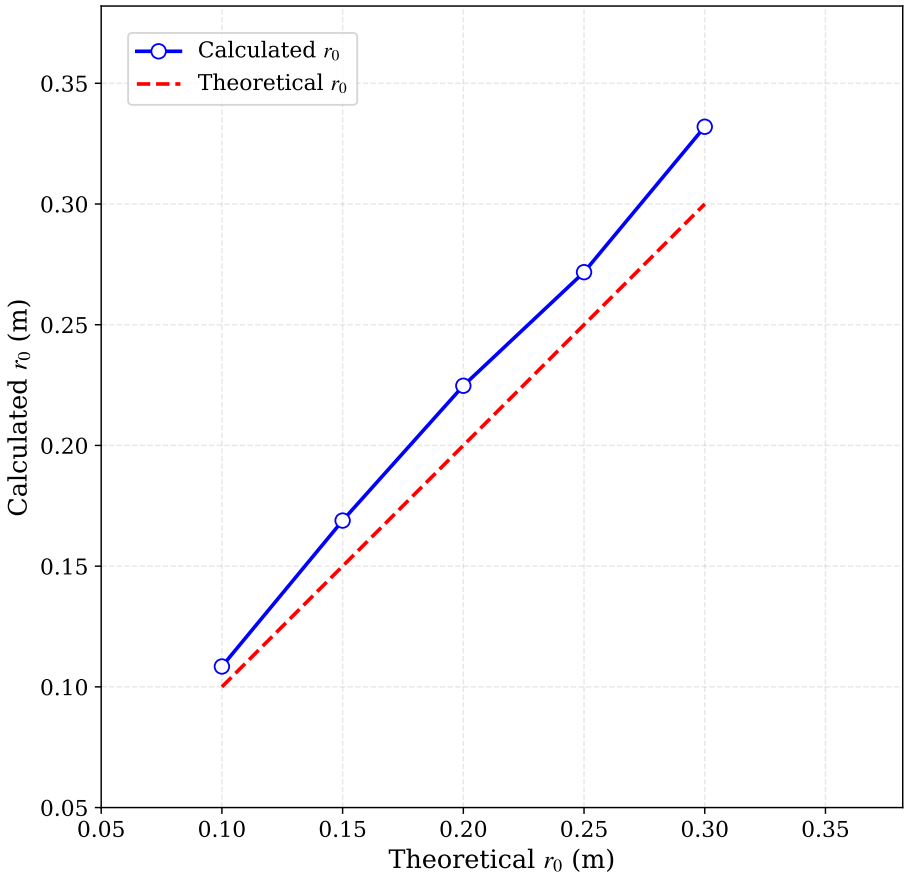


Figure 4.3: Mutual Coherence Function analysis for finding r_0 values. The solid lines show theoretical r0 curves, while markers indicate measured values from simulation. Note that the limit of small L0 can possibly lead to over-estimation of r0 values.

4.1.4 Fried Parameter r_0 from the MCF

In Kolmogorov turbulence, the mutual coherence function (MCF) of a plane wave is often written in terms of the coherence radius ρ_0

$$\mathrm{MCF}(r) \;=\; \exp\Big[-3.44\left(\tfrac{r}{r_0}\right)^{5/3}\Big]. \tag{4.1.2}$$

Define $r_{1/e}$ as the distance at which $\mathrm{MCF}(r)$ decays to $1/e$. Substituting $r = r_{1/e}$ into (4.1.2) gives

$$\exp\Big[-3.44\left(\tfrac{r_{1/e}}{r_0}\right)^{5/3}\Big] \;=\; \frac{1}{e}, \quad\Longrightarrow\quad 3.44\left(\tfrac{r_{1/e}}{r_0}\right)^{5/3} \;=\; 1. \tag{4.1.3}$$

Hence,

$$\left(\tfrac{r_{1/e}}{r_0}\right)^{5/3} \;=\; \tfrac{1}{3.44}, \quad\Longrightarrow\quad r_0 \;=\; \frac{r_{1/e}}{\left(\frac{1}{3.44}\right)^{\frac{3}{5}}}. \tag{4.1.4}$$

By definition in many atmospheric optics references, the Fried parameter r_0 is related to the coherence radius ρ_0 by the factor 2.1. Specifically,

$$r_0 \;=\; 2.1\,\rho_0. \tag{4.1.5}$$

Combining this with (4.1.4) leads to

$$r_0 \;=\; 2.1 \times \frac{r_{1/e}}{\left(\frac{1}{3.44}\right)^{\frac{3}{5}}}. \tag{4.1.6}$$

This formula connects the distance $r_{1/e}$, at which the MCF drops to $1/e$, to the Fried parameter r_0, which characterizes the transverse scale of atmospheric coherence.

Table 4.4: Comparison of Input and MCF-Derived Fried Parameters

Input r_0 (m)	MCF-Derived r_0 (m)	Relative Error (%)
0.10	0.108	8.4
0.15	0.169	12.6
0.20	0.225	12.5
0.25	0.272	8.8
0.30	0.332	11.3

This validation process was repeated over 1000 of independent realizations to ensure statistical robustness. For each theoretical r_0 value used in the simulation, the corresponding measured r_0 was computed using the MCF method. Results are shown in Figure 4.3 this analysis showed that the measured values closely tracked the theoretical predictions, with the subharmonic compensation method demonstrating more accurate results as it better captures the low-frequency large-scale fluctuations inherent in atmospheric turbulence.

The validation results in Table 4.4, which is based on Figure 4.3, demonstrate good agreement between input and measured r_0 values. The relative errors range from 8.4% to 12.6%, with the best agreement in the moderate turbulence regime ($r_0 \approx$ 10–25 cm). Several factors contribute to the observed deviations:

- **Grid Resolution Effects:** The 1 cm grid spacing limits the accuracy of small-scale phase variations, particularly affecting stronger turbulence cases.
- **Finite Aperture:** The radial averaging is constrained by the simulation grid size, affecting the accuracy of large-separation correlations.
- **Scintillation Regime Limitations:** The MCF approximation used in this analysis is only valid in the weak fluctuation regime where intensity fluctuations can be ignored. For strong scintillation conditions, this approximation breaks

down, potentially introducing systematic errors in the estimates.

Figure 4.3 illustrates the comparison between theoretical and measured r_0 values obtained through MCF analysis. As shown, the results align well across the range of turbulence conditions tested.

The MCF analysis confirms that the GPU-accelerated simulation correctly reproduces the expected turbulence-induced phase correlations across a range of r_0 values relevant to atmospheric optical propagation. The validation demonstrates both the accuracy of the turbulence modeling and the reliability of the MCF-based r_0 measurement technique.

4.2 Computational Performance Analysis

The computational efficiency of the GPU-accelerated implementation was evaluated through direct comparison with HCIPy [24], a widely-used CPU-based atmospheric turbulence simulation framework. All comparisons were performed on identical hardware: an Intel® Core™ i7-10700K CPU @ 3.80 GHz with 64 GB of RAM and an NVIDIA GeForce RTX 3070 GPU with 6 GB of VRAM.

4.2.1 Performance Comparison Setup

The benchmark tests were conducted using the same parameters as Table 4.1 for 500 evolution steps:

4.2.2 Execution Time Analysis

Figure 4.4 compares the execution times of the HCIPy and GPU-accelerated implementations across different grid sizes, under identical atmospheric conditions and propagation parameters. For a grid size of 128×128 points, the execution times for propagating through all atmospheric layers were:

- HCIPy implementation: 6200 seconds
- GPU-accelerated implementation: 20 seconds

The performance advantage of the GPU-accelerated implementation becomes even more pronounced at higher resolutions (Figure 4.4). Notably, under the same simulation setup, the HCIPy implementation failed to execute for grid sizes beyond 128×128 due to excessive memory usage, resulting in out-of-memory (OOM) errors. In contrast, the GPU implementation successfully scaled to much larger grids, including 1024×1024 and beyond. This represents a speedup factor of approximately 310×. The performance improvement can be attributed to several factors:

1. **Parallel FFT Computation:** The GPU-optimized FFT implementations of TensorFlow significantly accelerate the split-step propagation calculations.
2. **Efficient Memory Management:** By maintaining phase screens and propagation constants as GPU tensors, the simulator minimizes CPU-GPU data transfers during runtime.
3. **Batched Operations:** The GPU architecture efficiently handles the parallel processing of multiple phase screen columns during the extrusion process.

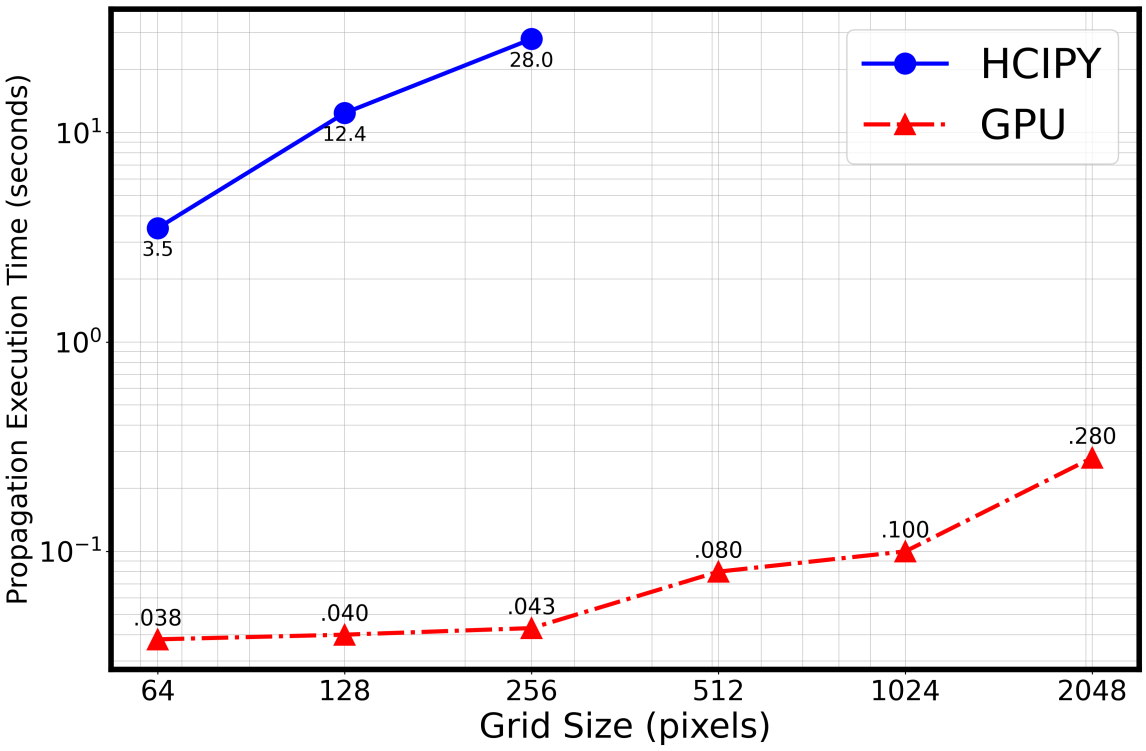


Figure 4.4: Execution time comparison between HCIPy and GPU-accelerated implementation for different grid sizes. Both simulations performed 500 iterations with identical atmospheric conditions and propagation parameters. Note the logarithmic scale on the time axis.

4.2.3 Scalability Analysis

The GPU implementation demonstrates improved scaling with increasing grid sizes. While HCIPy encounters performance limitations beyond 256×256 points due to CPU memory constraints and limited parallelism, the GPU-accelerated framework efficiently handles larger dimensions:

- At 256×256: Achieved 600× speedup over HCIPy
- At 2048×2048: Completed 500 iterations in 140 seconds (less than 300 ms per iteration)

This scalability is particularly important for high-fidelity simulations requiring fine spatial resolution or large propagation volumes. The ability to handle larger grid sizes while maintaining reasonable computation times enables more accurate modeling of:

- Fine-scale turbulence structures
- Wide aperture systems
- Extended propagation paths

4.3 GPU Simulator Outputs

The GPU-accelerated simulation framework provides detailed visualization of the optical field at the receiver plane after propagation through multiple atmospheric layers. This section presents representative outputs from the simulation, demonstrating the ability to model complex turbulence structures and their impact on optical propagation.

Table 4.5: GPU Simulator Configuration Parameters

Parameter	Value	Units
Number of atmospheric layers	10	–
Satellite altitude	600	km
Zenith angle	0	degrees
Wavelength (λ)	1550	nm
Total Fried parameter (r_0)	20	cm
Outer scale (L_0)	25	m
Grid resolution	1	cm
Grid size	1024×1024	points
Time step (Δt)	1	ms

4.3.1 Simulation Configuration

The simulator was configured to model atmospheric turbulence using parameters representative of a satellite-to-ground optical link. Table 4.5 details the key simulation parameters used for generating the outputs presented in this section. This simulation follows another scenario compared with Table 4.1, the difference are in number of layers and grid size.

4.3.2 Receiver Plane Output Visualization

The primary outputs of the GPU simulator at the receiver plane are the complex optical field U_{out} and its derived quantities. Figure 4.5 presents visualizations of the phase (angle of U_{out}) and absolute value of U_{out} distributions at the receiver plane for three consecutive time steps.

As shown in Figure 4.5, the phase distribution displays complex spatial patterns characteristic of atmospheric turbulence, with regions of rapid phase transitions indicating strong wavefront distortion. The corresponding intensity distributions exhibit

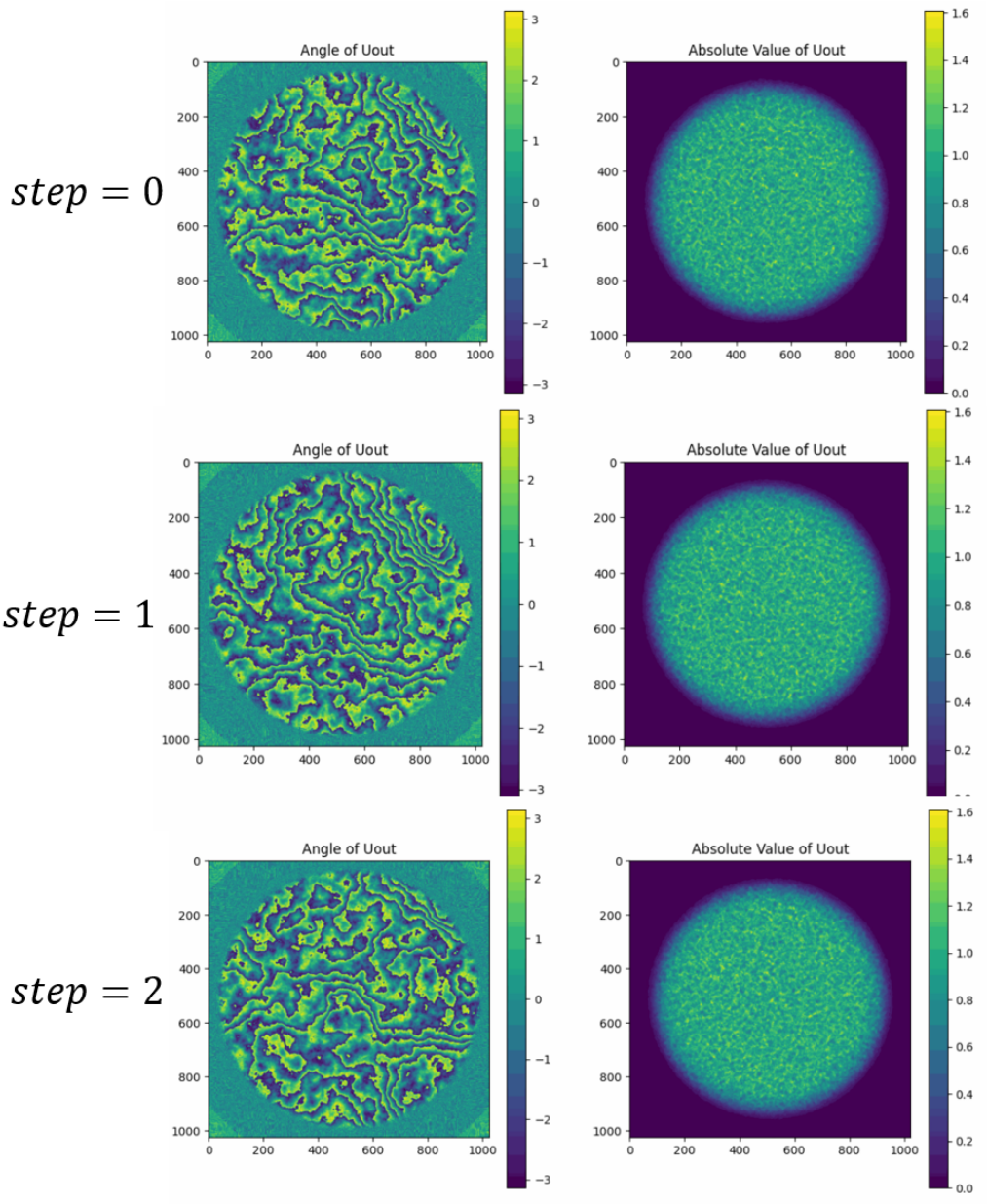


Figure 4.5: Time evolution of the optical field at the receiver plane after propagation through 10 atmospheric layers, where each step is 1 ms and each wavefront covers a 10.24 m × 10.24 m area. Left column: Phase distribution (angle of U_{out}) in radians. Right column: Intensity distribution (absolute value of U_{out}). Rows represents three consecutive steps, demonstrating the temporal evolution of turbulence-induced distortions. The phase screens show the accumulated wavefront distortion, while the intensity patterns reveal characteristic speckle structures resulting from atmospheric turbulence. Simulation parameters as specified in Table 4.5.

speckle patterns with bright spots (high intensity) occurring where constructive interference takes place, and dark regions where destructive interference reduces the received power.

The temporal evolution visible across the three time steps demonstrates the dynamic nature of atmospheric turbulence, with the phase and intensity patterns evolving continuously due to the wind and satellite apparent turbulence motion. This evolution occurs on timescales of milliseconds, creating bandwidth challenges for adaptive optics systems.

4.4 GPU Memory and Scaling Results

The performance of the GPU-accelerated framework was further analyzed with respect to memory utilization, computational scaling, and GPU resource usage across different grid sizes and layer counts. These analyses provide critical insights into the practical limitations and operational characteristics of the simulation framework.

4.4.1 Memory Utilization Analysis

GPU memory utilization is a critical factor in determining the feasibility of large-scale atmospheric turbulence simulations. A systematic analysis of memory requirements across varying grid sizes and atmospheric layer counts is conducted, with results presented in Figure 4.6.

The memory usage analysis revealed that in all tested configurations, TensorFlow pre-allocates nearly the entire available GPU memory, consistently using between

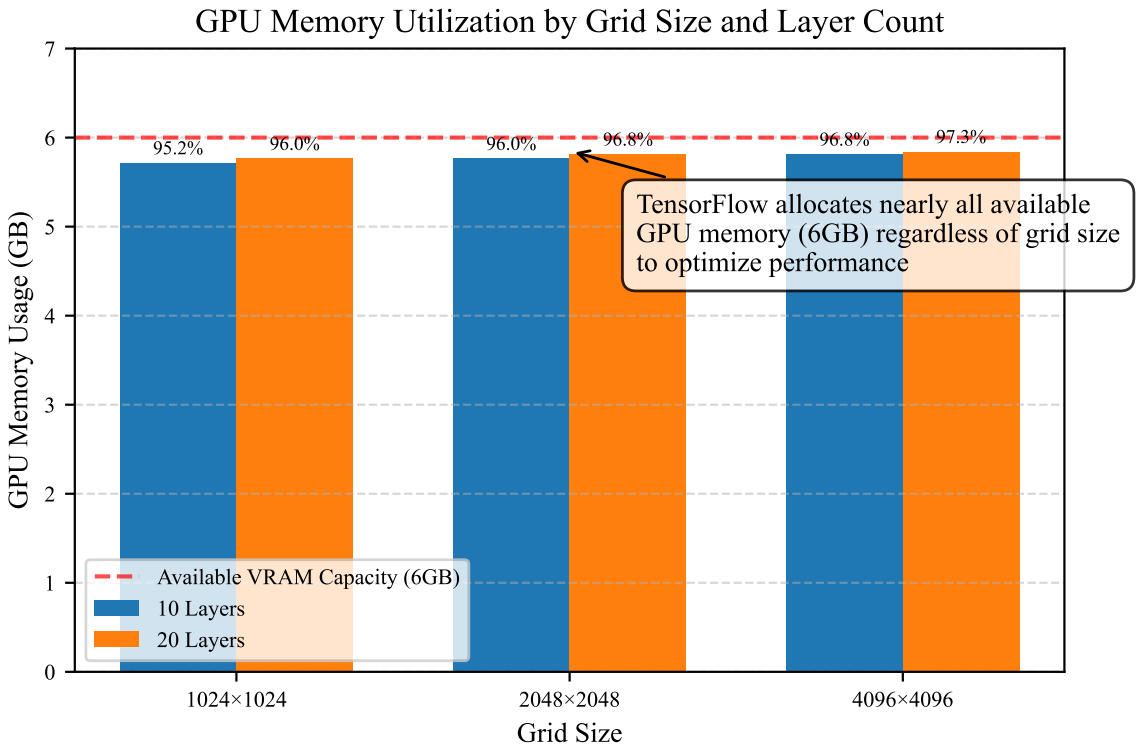


Figure 4.6: GPU memory utilization for different grid sizes and layer counts on an NVIDIA RTX 3070 GPU. While the RTX 3070 has 8GB of VRAM, only 6GB is available to TensorFlow as approximately 2GB is reserved by the operating system. For all tested configurations, TensorFlow allocates nearly the entire available memory capacity (5850-5980 MB or approximately 97.5-99.7% of the available 6GB), regardless of grid size or layer count.

5850-5980 MB of the 6GB available on NVIDIA GeForce RTX 3070 GPU. This behavior is characteristic of TensorFlow memory management strategy, which reserves most available GPU memory upfront to minimize memory fragmentation and reduce allocation overhead during computation, regardless of the actual computational requirements of different grid sizes.

The consistent memory usage across different grid sizes and layer counts suggests that the TensorFlow-based implementation dynamically manages GPU memory allocation, allocating the maximum available resources to optimize performance. This approach provides several benefits:

- **Efficient Resource Utilization:** By allocating all available memory, the implementation can minimize CPU-GPU data transfers and maintain computational data structures in high-speed GPU memory.
- **Performance Optimization:** TensorFlow memory management allows for pre-allocation of buffer space for FFT operations and intermediate results, reducing allocation overhead during runtime.
- **Dynamic Scaling:** The framework can automatically adjust to the available hardware resources, enabling the simulation to run efficiently across different GPU configurations.

While this approach maximizes performance, it also creates clear hardware limitations for larger simulations. These tests revealed that when attempting to run simulations with larger grid sizes or more atmospheric layers than reported here, the system produced out-of-memory errors as described in the error logs. Specifically, these errors occurred when:

- Attempting to run simulations with grid sizes beyond 4096×4096 with 10 or more layers
- Combining high layer counts (20+) with grid sizes larger than 2048×2048

These memory constraints underscore the importance of hardware selection when conducting large-scale atmospheric turbulence simulations. For applications requiring both high spatial resolution and complex multi-layer atmospheric models, higher-capacity GPUs, such as those with additional memory or used in multi-GPU clusters, would be necessary.

The computational efficiency of the GPU-accelerated implementation was evaluated across different grid sizes and layer counts. Figure 4.7 shows the execution time comparison between different configurations.

The GPU-accelerated implementation demonstrates efficient scaling with increasing grid sizes:

- At 1024×1024 grid size: 0.018 seconds per propagation (10 layers) and 0.019 seconds per propagation (20 layers)
- At 2048×2048 grid size: 0.042 seconds per propagation (10 layers) and 0.045 seconds per propagation (20 layers)
- At 4096×4096 grid size: 0.105 seconds per propagation (10 layers) and 0.115 seconds per propagation (20 layers)

These results demonstrate that the GPU implementation handles the computational scaling challenges associated with larger grid sizes effectively. When grid dimensions quadruple from 1024×1024 to 2048×2048 (a 4× increase in grid points),

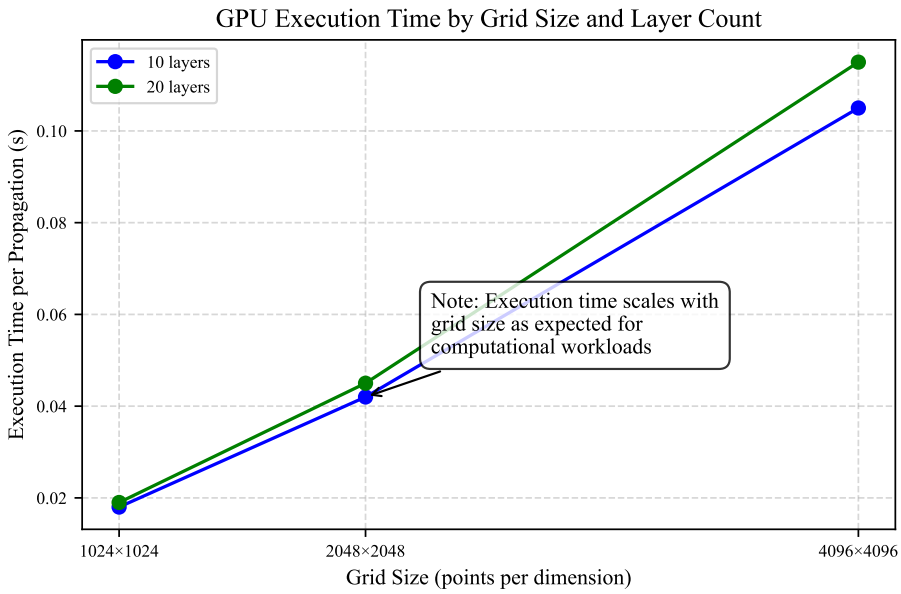


Figure 4.7: Execution time comparison of the GPU-accelerated implementation across different grid sizes and layer counts. The execution time scales with increasing grid dimensions, showing expected computational complexity growth as the problem size increases.

execution time increases by approximately 2.3×. Similarly, when dimensions increase from 1024×1024 to 4096×4096 (a 16× increase in grid points), execution time increases by approximately 5.8×. The increase in execution time per propagation compared with Figure 4.4 is due to the use of ten layers in this case versus five layers in the previous experiment. Additionally, at 4096×4096, the transition from ten to twenty layers exhibits a more pronounced jump, indicating a mildly non-linear scaling effect as layer count grows.

This sub-linear scaling behavior for the GPU implementation indicates that the TensorFlow-based implementation efficiently leverages the parallel processing capabilities of modern GPUs. While execution time does increase with larger grid sizes, it scales more favorably than the corresponding increase in problem size, demonstrating efficient utilization of GPU resources. Based on the memory-bandwidth limited nature of our implementation, it is anticipated that near-linear scaling when distributing workloads across GPU clusters, with high-memory GPUs (40-80GB) enabling simulation domains of 8192×8192 or beyond with minimal performance degradation. The memory utilization analysis further reveals that TensorFlow pre-allocates nearly all available GPU memory (approximately 6GB on RTX 3070) regardless of grid size, which helps minimize memory fragmentation and reduce allocation overhead during computation.

4.4.2 GPU Utilization Analysis

The GPU utilization patterns across different grid sizes and layer counts is analyzed to better understand the resource usage characteristics of this implementation. Figure 4.8 presents the average and peak GPU utilization observed during experiments.

All GPU utilization metrics were collected using NVIDIA System Management Interface (nvidia-smi) through Python subprocess module.

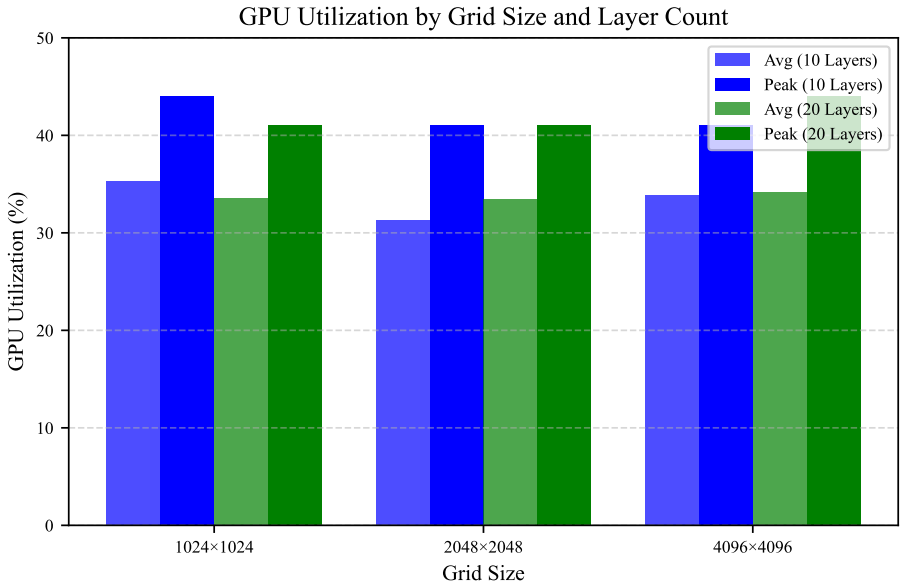


Figure 4.8: GPU utilization by grid size and layer count. The average utilization remains relatively consistent across different configurations, ranging from 31.28% to 35.31%, while peak utilization reaches 41-44%. This moderate utilization pattern indicates that memory bandwidth and other system constraints, rather than raw computational throughput, are likely the limiting factors in this implementation.

The GPU utilization data revealed several important characteristics of this implementation:

- **Moderate Utilization Levels:** Average GPU utilization ranged from 31.28% to 35.31% across all tested configurations, with peak utilization reaching 41-44%.
- **Consistent Utilization Pattern:** Despite significant differences in problem size, the utilization levels remained remarkably consistent across grid sizes and layer counts.

- **Minimal Layer Count Impact:** Doubling the layer count from 10 to 20 had only a minor effect on GPU utilization, suggesting efficient parallelization of the multi-layer processing.

These moderate utilization levels, combined with the high memory usage observed in Section 4.4.1, indicate that GPU-parallel implementation is primarily memory-bandwidth limited rather than compute-limited.

The consistency of GPU utilization across different grid sizes suggests that the implementation scales well with problem size, effectively utilizing the parallel processing capabilities of the GPU. However, it also indicates potential opportunities for further optimization, particularly in terms of memory access patterns and kernel fusion techniques that could improve GPU utilization.

4.4.3 First-Run Overhead Analysis

An interesting phenomenon observed during the performance testing was the consistent overhead associated with the first execution of the simulation on the GPU. Figure 4.9 illustrates this pattern.

The analysis of the iteration-by-iteration execution times for the 1024×1024 grid with 10 layers configuration revealed a significant first-run overhead. The first propagation required 0.266 seconds, while subsequent propagations averaged 0.191 seconds, representing a 39.5% performance penalty for the initial run.

This first-run overhead is a well-known phenomenon in GPU computing and can be attributed to several factors:

- **Just-In-Time (JIT) Compilation:** TensorFlow and CUDA perform JIT

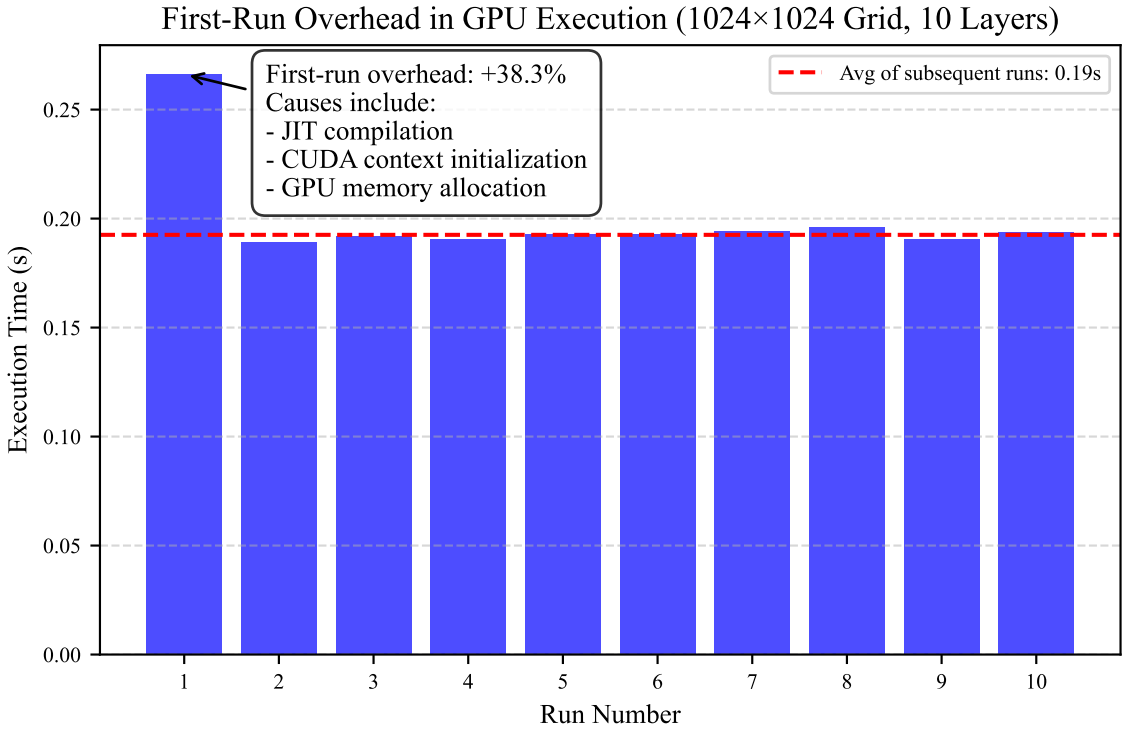


Figure 4.9: Execution time comparison between the first run and subsequent runs of the simulation on the GPU for the 1024×1024 grid with 10 layers configuration. The first propagation requires 0.266 seconds while subsequent propagations average 0.191 seconds, representing a 39.5% performance penalty for the initial run due to GPU-specific initialization steps.

compilation of optimized kernels during the first run. Modern GPU frameworks like TensorFlow use a lazy execution model where kernels are compiled only when first needed, which significantly increases the execution time of the first run. These compiled kernels are then cached for subsequent runs, leading to improved performance.

- **CUDA Context Initialization:** Establishing the CUDA execution context involves significant overhead, including device discovery, capability checking, and memory system initialization. This initialization process only occurs on the first GPU operation within a process.

- **Memory Allocation and Transfer:** The initial allocation of GPU memory buffers and transfer of data structures from CPU to GPU memory incurs additional overhead. In subsequent runs, these memory allocations are reused, eliminating this overhead.

- **Computation Graph Optimization:** TensorFlow performs runtime optimization of the computation graph during the first execution, including operation fusion, buffer reuse planning, and execution strategy selection.

- **GPU Warm-up Effects:** Modern GPUs dynamically adjust their clock speeds based on workload and thermal conditions. The first execution often occurs while the GPU is still "cold" (running at lower clock speeds), with subsequent runs benefiting from dynamically increased clock speeds.

In practical applications, this first-run overhead can be effectively amortized across multiple simulation iterations. For atmospheric turbulence simulations, which typically involve hundreds or thousands of iterations, the impact of this overhead on

overall performance is negligible. However, users should be aware of this behavior when benchmarking or when executing short simulation runs, and consider implementing a "warm-up" run before timing critical operations.

4.5 Conclusion

This chapter presented a comprehensive validation and performance analysis of the GPU-accelerated atmospheric turbulence simulation framework. The results demonstrate that the framework successfully reproduces the expected physical characteristics of turbulence-induced wavefront distortions while providing substantial computational advantages over traditional CPU-based approaches.

The scintillation index validation against theoretical models confirms that the numerical implementation accurately captures the fundamental physics of atmospheric turbulence propagation. For moderate to weak turbulence conditions ($r_0 \geq 15$ cm), which encompass most practical satellite-to-ground optical communication scenarios, the simulation demonstrates excellent agreement with theoretical predictions, maintaining relative errors below 10%. The discrepancies observed in stronger turbulence regimes ($r_0 < 15$ cm) can be attributed to limitations in the phase screen resolution and the breakdown of the underlying Rytov approximation in strong fluctuation conditions.

The Mutual Coherence Function analysis further validates the approach, with MCF-derived r_0 values closely tracking the input parameters across the tested range. The observed relative errors of 8.4% to 12.6% represent acceptable accuracy for practical applications, particularly considering the inherent statistical nature of turbulence phenomena and the approximations inherent in the numerical implementation.

The most significant contribution of this work is the substantial reduction in computation time achieved through GPU acceleration. The observed speedup factors of 310× to 600× over the HCIPy implementation transform what was previously a computationally prohibitive task into a practical tool for comprehensive turbulence studies, effectively addressing the challenge outlined in the literature review section. The ability to execute simulations with 1024×1024 grid resolution in less than 200 milliseconds per iteration enables rapid analysis and facilitates the generation of large statistical datasets for machine learning applications.

Analysis of GPU memory utilization and computational scaling provides insights into the practical limitations and operational characteristics of the framework. The consistent pre-allocation of nearly all available GPU memory by TensorFlow, regardless of grid size or layer count, suggests that memory capacity represents the primary constraint for larger simulations. This observation, combined with the moderate GPU utilization levels (31-35 percent), indicates that GPU-parallel implementation is memory-bandwidth limited rather than compute-limited.

The GPU-accelerated simulation framework provides a robust foundation for the high-fidelity modeling of atmospheric turbulence effects on optical wave propagation. The reduction in computation time, combined with demonstrated accuracy across relevant turbulence conditions, transforms atmospheric turbulence simulation from a computationally intensive task into an accessible tool for both research and practical engineering applications. As optical communication systems increasingly extend into atmospheric channels, such efficient simulation tools will play a crucial role in system design, optimization, and performance prediction.

Chapter 5

Conclusion

This thesis has presented a comprehensive look at simulation of free-space optical (FSO) downlinks from Low Earth Orbit (LEO) satellites, with particular emphasis on modeling and simulating atmospheric turbulence effects. Drawing upon the theoretical groundwork in Kolmogorov and von Kármán turbulence, a GPU-accelerated, multi-layer simulation framework was developed to capture both the spatial and temporal evolution of laser beam propagation through the atmosphere.

Summary of Core Contributions

1. **GPU-Accelerated Optical Propagation Simulator with Turbulence Modeling:** Developed a comprehensive TensorFlow-based simulation platform that exploits parallelized FFTs and optimized dataflow to achieve speedups exceeding 300× compared to established CPU-based tools, while integrating

Fourier-based phase screen generation with Fried's extrusion method for high-fidelity, time-evolving turbulence representations, and implementing a layer-slicing approach that evenly distributes scintillation contributions for improved numerical stability across heterogeneous atmospheric profiles.

2. **Validation and Performance Benchmarking:** Demonstrated close agreement with theoretical scintillation predictions and phase statistics under moderate to weak turbulence regimes, and showcased scalability for large grid simulations.

3. **Integration with Satellite Downlink Orbital Geometries:** Incorporated satellite orbital mechanics and slant-path calculations to align physical conditions (wind shear, C_n^2 profiles) with actual downlink scenarios.

Study Limitations

While the GPU-accelerated method significantly reduces runtime, several constraints and assumptions remain:

- **Strong Turbulence Modeling:** As turbulence intensifies, conventional Rytov-based theoretical metrics (e.g., scintillation index) become less reliable, and the simulation may require more sophisticated non-linear turbulence models.

- **Finite Grid and Sampling:** A fixed spatial sampling can underrepresent very large or very small eddies, especially when $L_0/\Delta x_i$ (outer scale to grid resolution) becomes large.

- **Homogeneity in Layers:** Each atmospheric layer is assumed spatially homogeneous in C_n^2 and wind velocity, which simplifies computations but may not fully reflect complex meteorological conditions.

Future Research Directions

Building on the insights and innovations of this work, several avenues for further research and development are recommended:

1. **Bidirectional Link Modeling:** Extend the simulation framework to incorporate uplink scenarios (ground-to-satellite), accounting for the unique atmospheric challenges when propagating from dense to rarefied atmospheric layers and the corresponding impact on beam divergence and scintillation.

2. **Dynamic Orbital Evolution:** Enhance orbital modeling to incorporate real-time satellite position updates and Earth rotation effects, creating temporally coherent simulations where atmospheric conditions evolve naturally with changing link geometries during a complete pass.

3. **Adaptive Resolution Methods:** Employ multi-resolution or wavelet-based grids to capture both small-scale eddies and large-scale turbulence within a single framework.

4. **Machine Learning Integration:** Leverage the GPU simulator to generate massive labeled datasets for training deep neural networks in real-time wavefront sensing, phase reconstruction, or link performance forecasting.

5. **Coupled System Modeling:** Include advanced orbital dynamics with pointing, acquisition, and tracking (PAT) subsystems in a co-simulation environment, capturing end-to-end link performance.

6. **Experimental Validation:** Corroborate the simulation framework with in-situ measurements or testbed experiments, particularly for space-to-ground optical channels under dynamic atmospheric conditions.

Closing Statement

In conclusion, the work presented herein provides computational tools for LEO satellite FSO communications, combining theoretical methods, methodological advancements, and GPU implementations. Through innovations in unified satellite orbital geometry modeling and GPU-accelerated phase screen generation, this thesis lays the groundwork for more reliable and scalable optical communication links. By combining turbulence physics, wave-optics propagation, and computational techniques, this framework advances our understanding of atmospheric effects on optical links and enables applications in system design and optimization—contributing to the development of terrestrial and space-based data networks. This novel integration of GPU acceleration (achieving speedups of 310-600$\times$ over HCIPy), multi-layer atmospheric modeling with sophisticated layer-slicing techniques, and time-evolving turbulence representation through optimized phase screen extrusion collectively establishes a significant advancement in the field of atmospheric optical propagation simulation, transforming simulations that once took hours into calculations within significantly reduced time windows and enabling grid sizes beyond what traditional methods could

process, with direct implications for next-generation satellite communication systems.

Bibliography

[1] Lewis F. Richardson. Weather prediction by numerical process. *Cambridge University Press*, page 66, 1922. doi: 10.1017/CBO9780511618291.

[2] Tor M. Aamodt. Gpu computing architecture. Lecture Notes, HiPEAC Summer School, July 2015. URL `https://www.cse.iitk.ac.in/users/swarnendu/courses/autumn2022-cs610/gpu-cuda.pdf`. Slides accessed online.

[3] Hemani Kaushal and Georges Kaddoum. Optical communication in space: Challenges and mitigation techniques. *IEEE Communications Surveys & Tutorials*, 19(1):57–96, 2017. doi: 10.1109/COMST.2016.2603518.

[4] Mohammad A. Khalighi and Murat Uysal. Survey on free space optical communication: A communication theory perspective. *IEEE Communications Surveys & Tutorials*, 16(4):2231–2258, 2014. doi: 10.1109/COMST.2014.2329501.

[5] Shlomi Bloom, Eric Korevaar, John Schuster, and Heinz Willebrand. Understanding the performance of free-space optics. *Journal of Optical Networking*, 2 (6):178–200, 2003.

[6] NASA. Lunar laser communication demonstration (llcd), 2014. URL: `https://www.nasa.gov/mission_pages/LADEE/lasercomm/index.html`.

[7] NASA. Laser communications relay demonstration, 2020. URL: https://www.nasa.gov/mission_pages/tdm/lcrd/.

[8] Motoyuki Toyoshima. Trends in satellite communications and the role of optical free-space communications. In *Journal of Optical Networking*, volume 4, pages 300–311, 2005.

[9] Travis R. Brashears. Achieving 99% link uptime on a fleet of 100g space laser inter-satellite links in leo. *Proc. SPIE 12877, Free-Space Laser Communications XXXVI*, 12877, 2024. https://doi.org/10.1117/12.3005057.

[10] Amazon. Project kuiper: Delivering fast, affordable broadband through a constellation of satellites in low earth orbit, 2023. https://www.aboutamazon.com/what-we-do/devices-services/project-kuiperr.

[11] Telesat. Lightspeed constellation, 2023. https://www.telesat.com/leo-satellites/.

[12] NASA and MIT Lincoln Laboratory. Terabyte infrared delivery (TBIRD) mission achieves 200 gbps from LEO to ground, 2022. https://www.ll.mit.edu/r-d/projects/terabyte-infrared-delivery-tbird.

[13] Ezra Ip, Alan Pak Tao Lau, Daniel J. F. Barros, and Joseph M. Kahn. Coherent detection in optical fiber systems. *Optics Express*, 16(2):753–791, 2008. doi: 10.1364/OE.16.000753.

[14] Thomas Weyrauch, Mikhail A. Vorontsov, Jay Gowens, and Thomas G. Bifano. Fiber coupling with adaptive optics for free-space optical communication. In

Proc. SPIE 4489, Free-Space Laser Communication and Laser Imaging. SPIE, January 2002. doi: 10.1117/12.453227.

[15] Arun K. Majumdar and John C. Ricklin. Free-space laser communications: Principles and advances. *Springer Series in Optical Sciences*, 118, 2008.

[16] Hamid Hemmati. *Deep Space Optical Communications*. Wiley-Interscience, Hoboken, NJ, 2006.

[17] Abu Jahid, Mohammed H. Alsharif, and Trevor J. Hall. A contemporary survey on free space optical communication: Potentials, technical challenges, recent advances and research direction. *Journal of Network and Computer Applications*, 200:103311, 2022. ISSN 1084-8045. doi: 10.1016/j.jnca.2021.103311. URL `https://www.sciencedirect.com/science/article/pii/S108480452100299X`.

[18] H. Tennekes and J. L. Lumley. *A First Course in Turbulence*. MIT Press, Cambridge, MA, 1972.

[19] Peter Alan Davidson. *Turbulence: An Introduction for Scientists and Engineers*. Oxford University Press, 2 edition, 2015. ISBN 978-0198722588.

[20] Valeryan Ilich Tatarskii. *Wave Propagation in a Turbulent Medium*. McGraw-Hill, New York, 1961.

[21] Larry C Andrews and Ronald L Phillips. *Laser Beam Propagation Through Random Media*. SPIE Press, Bellingham, WA, 2nd edition, 2005.

[22] R. E. Hufnagel and N. R. Stanley. Modulation transfer function associated with image transmission through turbulent media. *JOSA*, 54(1):52–61, 1964.

[23] M. J. Townson, O. J. D. Farley, G. Orban de Xivry, J. Osborn, and A. P. Reeves. AOtools: a python package for adaptive optics modelling and analysis. *GitHub/Zenodo (online resource)*, 2019. `https://github.com/AOtools/AOtools`.

[24] Emiel H Por, Sebastiaan Y Haffert, Vikram M Radhakrishnan, David S Doelman, Maaike van Kooten, and Steven P Bos. High contrast imaging for Python (HCIPy): an open-source adaptive optics and coronagraph simulator. In *Adaptive Optics Systems VI*, volume 10703, page 1070354. SPIE, 2018.

[25] Andrey Nikolaevich Kolmogorov. The local structure of turbulence in incompressible viscous fluid for very large reynolds numbers. *Dokl. Akad. Nauk SSSR*, 30:301–305, 1941.

[26] Theodore von Kármán. Progress in the statistical theory of turbulence. *Proceedings of the National Academy of Sciences*, 34(11):530–539, 1948.

[27] R. G. Lane, A. Glindemann, and J. C. Dainty. Simulation of a kolmogorov phase screen. *Waves in Random Media*, 2(3):209–224, 1992.

[28] François Roddier. The effects of atmospheric turbulence in optical astronomy. *Progress in Optics*, 19:281–376, 1981.

[29] F Assémat, R W Wilson, and E Gendron. Method for simulating infinitely long and non stationary phase screens with optimized memory storage. *Optics Express*, 14(3):988–999, 2006.

[30] M. D. Feit and J. A. Fleck Jr. Light propagation in graded-index optical fibers. *Applied Optics*, 17(24):3990–3998, 1978.

[31] Jeffrey M Martin and Stanley M Flatté. Wave propagation through atmospheric turbulence: Numerical methods and results. *Applied Optics*, 27(11):2111–2126, 1988.

[32] Jason D Schmidt. *Numerical Simulation of Optical Wave Propagation with Examples in MATLAB*. SPIE, 2010.

[33] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, Cambridge, UK, 3rd edition, 2007.

[34] G. I. Taylor. The spectrum of turbulence. *Proceedings of the Royal Society of London. Series A*, 164(919):476–490, 1938.

[35] Rod Frehlich and Larry Cornman. Estimating spatial velocity statistics with coherent doppler lidar. *Journal of Atmospheric and Oceanic Technology*, 16(12): 1967–1977, 1999.

[36] Glenn A Tyler. Bandwidth considerations for tracking through turbulence. *JOSA A*, 11(1):358–367, 1994.

[37] David L. Fried. Statistics of a geometric representation of wavefront distortion. *Journal of the Optical Society of America*, 67(3):370–375, 1977.

[38] David L Fried and Tim Clark. Extruding Kolmogorov-type phase screen ribbons. *JOSA A*, 25(2):463–468, 2008.

[39] Michael C Roggemann and Byron M Welsh. *Imaging Through Turbulence*. CRC Press, 1996.

[40] Darryl P Greenwood. Bandwidth specification for adaptive optics systems. *Journal of the Optical Society of America*, 67(3):390–393, 1977.

[41] Harald Hauschildt, Carlo Elia, Andrew Jones, Hermann Ludwig Moeller, and Josep Maria Perdigues Armengol. ESAs ScyLight Programme: activities and status of the high throughput optical network "HydRON". *Proc. of SPIE Vol. 11180, 111800G*, 2018. https://doi.org/10.1117/12.2535935.

[42] N. Védrenne, J. Conan, M. Velluet, M. Sechaud, M. Toyoshima, H. Takenaka, A. Guérin, and F. Lacoste. Turbulence effects on bi-directional ground-to-satellite laser communication systems. *Proceedings of ICSOS*, 2012.

[43] A.-R. Camboulives, M.-T. Velluet, S. Poulenard, L. Saint-Antonin, and V. Michau. Statistical and temporal irradiance fluctuations modeling for a ground-to-geostationary satellite optical link. *Applied Optics*, 57(4):709–721, 2018.

[44] R. Conan. OOMAO: an object oriented matlab adaptive optics toolbox. *GitHub (online resource)*, 2023. https://github.com/rconan/OOMAO.

[45] Aniceto Belmonte. Graph-based model for adaptive simulation of beam propagation in turbulent media. *Optics Express*, 31(11):18533–18544, 2023.

[46] Paul D. Shubert. Anisoplanatic effects in moderate aperture laser communication system uplinks. In *Free-Space Laser Communications XXXI*, volume 10910, page 109101L. SPIE, 2019.

[47] O. J. D. Farley, M. J. Townson, and J. Osborn. FAST: Fourier domain adaptive optics simulation tool for bidirectional ground-space optical links through atmospheric turbulence. *Optics Express*, 30(13):23050–23064, 2022.

[48] A. Reeves. Soapy: an adaptive optics simulation written purely in Python for rapid concept development. In *Proceedings of SPIE*, volume 9909, page 99097F. SPIE, 2016.

[49] Shane Walsh and Sascha Schediwy. Adaptive optics LEO uplink precompensation with finite spatial modes. *Optics Letters*, 48(4):880–883, 2023. doi: 10.1364/OL.482550.

[50] John D. Owens, Mike Houston, David Luebke, Simon Green, John E. Stone, and James C. Phillips. GPU computing. *Proceedings of the IEEE*, 96(5):879–899, 2008.

[51] Martín Abadi et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[52] A. N. Kolmogorov. The local structure of turbulence in incompressible viscous fluid for very large reynolds numbers. *Dokl. Akad. Nauk SSSR*, 30:299–303, 1941. In Russian; English translation in Proc. R. Soc. Lond. A 434, 9–13 (1991).

[53] A. N. Kolmogorov. Dissipation of energy in the locally isotropic turbulence. *Dokl. Akad. Nauk SSSR*, 32:16–18, 1941. In Russian; English translation in Proc. R. Soc. Lond. A 434, 15–17 (1991).

[54] S. B. Pope. *Turbulent Flows*. Cambridge University Press, Cambridge, UK, 2000.

[55] T. von Kármán and L. Howarth. On the statistical theory of isotropic turbulence. *Proceedings of the Royal Society of London. Series A*, 164(917):192–215, 1938.

[56] Max Born and Emil Wolf. *Principles of Optics*. Cambridge University Press, 7th edition, 1999.

[57] Joseph W Goodman. *Introduction to Fourier Optics*. Roberts and Company Publishers, 3rd edition, 2005.

[58] David L. Fried. Optical resolution through a randomly inhomogeneous medium for very long and very short exposures. *Journal of the Optical Society of America*, 56(10):1372–1379, 1966. doi: 10.1364/JOSA.56.001372.

[59] David L. Fried. Optical resolution through a randomly inhomogeneous medium. *Journal of the Optical Society of America*, 56(10):1372–1379, 1966.

[60] John W Hardy. *Adaptive optics for astronomical telescopes*. Oxford University Press, 1998.

[61] James H. Churnside. Aperture averaging of optical scintillations in the turbulent atmosphere. *Applied Optics*, 28(15):3304–3310, 1989. doi: 10.1364/AO.28.003304.

[62] R. E. Hufnagel. Variations of atmospheric turbulence. *Digest of Technical Papers, Topical Meeting on Optical Propagation through Turbulence*, pages Wa1–1–Wa1–4, 1974.

[63] Robert R. Beland. Propagation through atmospheric optical turbulence. In *Atmospheric Propagation of Radiation*, volume PM07, pages 1–100. SPIE Optical Engineering Press, Bellingham, WA, 1993.

[64] Jack L Bufton, Peter O Minott, Michael W Fitzmaurice, and Pierre J Titterton. Comparison of vertical profile turbulence structure with stellar observations. *Applied Optics*, 12(8):1785–1793, 1973.

[65] G. I. Taylor. The spectrum of turbulence. *Proc. R. Soc. Lond. A*, 164:476–490, 1938.

[66] F. Roddier. The effects of atmospheric turbulence in optical astronomy. *Progress in Optics*, 19:281–376, 1981.

[67] R. G. Lane, A. Glindemann, and J. C. Dainty. Simulation of a kolmogorov phase screen. *Waves in Random Media*, 2:209–224, 1992.

[68] J. R. Wertz, D. F. Everett, and J. J. Puschell. *Space Mission Engineering: The New SMAD*. Microcosm Press, 2011.

[69] John D. Owens, David Luebke, Naga K. Govindaraju, Mark Harris, Jens Kruger, Aaron E. Lefohn, and Timothy Purcell. A survey of general-purpose computation on graphics processors. *Computer Graphics Forum*, 26(1):80–113, 2007.

[70] John Nickolls, Ian Buck, Michael Garland, and Kevin Skadron. Scalable parallel programming with cuda. *ACM Queue*, 6(2):40–53, 2008.

[71] Gene H Golub and Charles F Van Loan. Matrix computations. *Johns Hopkins Studies in Mathematical Sciences*, 3, 2012.

[72] David B Kirk and Wen-mei W Hwu. *Programming Massively Parallel Processors: A Hands-on Approach*. Morgan Kaufmann, 3rd edition, 2017.

[73] TensorFlow. Introduction to graphs and tf.function. TensorFlow Documentation, 2023. URL: `https://www.tensorflow.org/guide/intro_to_graphs`.

[74] Nicholas J Higham. Accuracy and stability of numerical algorithms. *SIAM*, 2002.

[75] Lloyd N Trefethen and David Bau III. *Numerical linear algebra*. SIAM, 1997.