University of Amsterdam

MSc Physics and Astronomy
Track: Astronomy & Astrophysics

Master Thesis

Modelling the time lags in Black Hole X-Ray binaries

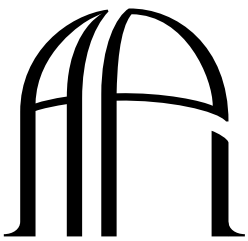
by

Dani van Enk
11823526 (UVA)

November 13, 2023
60 ECTS
June 2021 - July 2023

Supervisors:
dr. Phil Uttley

Examiners
dr. Phil Uttley
prof. dr. Sera Markoff



ANTON PANNEKOEK
INSTITUTE

Abstract

Extreme objects in the universe have been found to be in multi-object systems like binaries. When one of these components is a black hole, these binaries can emit X-rays from its accretion disk and the area closer to and around the black hole called the corona. However, due to a still debated mechanism, the X-rays from the accretion disk and the corona will lag behind each other. Recently, a new model has come out to describe these lags. Initially, the model was created in Python, and this project aimed to look into the viability of converting it to a compiled language like C++. The Reasons are quicker run time and the ability to run in a standardized fitting program. The latter, however, was not investigated in this thesis. While most of the functions part of this Python lag model could be easily converted with little to no changes, the calculations concerning the illumination fractions could be overhauled entirely by separating the radial dependent and independent calculations. The overhaul caused a speedup of this calculation of around 40-70% depending on the grid size of the geometry used. This speedup is equal to a speedup of several seconds on a function that takes several seconds. To test the converted functions, the C++ version and Python version of the model were fitted to XMM Newton data from GX 339-4 and NICER data from MAXI J1820+70. Both versions of the model fitted the data about the same. In conclusion, the conversion does speed up the model run time but less than one might expect. As such, it is still the question of whether it is worth converting the model for use in the standardized fitting program or just implementing the improvements to the Python version.

Contents

Chapter 1

Introduction

Ever since their conception, black holes have been objects filled with mystery. Them being patches of space where nothing would be able to escape makes them really hard to be seen. The only way that black holes can be detected are able to be detected is in an indirect way, in other words via matter or objects being affected by its gravitational pull. This makes them incredibly hard to detect, let alone study. Luckily in recent decades, this has become easier with different and new techniques to study these mysterious objects.

1.1 Black Hole History

A few years after Albert Einstein came up with the idea of the general theory of relativity, Karl Schwarzschild came up with the first modern solution to the general relativity theory that characterizes a black hole (Schwarzschild 1916). It would still take decades before "black holes" would be interpreted as a region of space that nothing can escape from in a paper by David Finkelstein (Finkelstein 1958). However, the idea of a "black hole" remained a mathematical curiosity. It took until the discovery of a neutron star in 1967 (Hewish et al. 1968) for interest in gravitationally collapsed compact objects to increase. Just 5 years before that proof of X-ray sources outside our solar system had been found (Giacconi et al. 1962) which prompted the launch of a mission into space to look for X-ray sources. In this mission, several X-ray sources were observed, including Cygnus X-1. However, the observations showed that Cygnus X-1 had fluctuations in the X-ray of several times a second. Just a year later two different radio observatories independently located a star (HDE 226868) that was thought to be the source of the X-ray radiation. However, since it was found to be a super-giant star so it can't generate X-rays on its own. So it needed to have a companion that could heat up the hot gas to a high enough temperature to generate those X-rays (Kristian et al. 1971; Braes & Miley 1971). Then in 1972

this companion, the first Black Hole, was detected and identified independently by multiple researchers (Webster & Murdin 1972; Bolton 1972). It would still take until the following year for there to be consensus about it being a black hole. Researchers found fluctuations in the X-ray signal that could point to mass fluctuations in an accretion disk this combined with the fact that it can't be a neutron star due to its mass made them quite certain that Cygnus X-1 is a black hole (Rothschild et al. 1974; Shipman 1975).

1.2 Types of Black Holes

The current understanding of black holes is that there are generally 3 types of black holes. The most massive of black holes are called supermassive black holes, these black holes lie in the centers of galaxies (Kormendy & Richstone 1995) and due to their mass ranging from hundreds of thousands to several billions of solar masses ($M_\odot$) are they believed to be one of the driving forces that keep a galaxy together. They are thought to have formed in the early universe. Secondly, a smaller type of black hole is the intermediate-mass black hole. These have masses from hundreds to hundreds of thousands of solar masses. They have first been proposed by Colbert & Mushotzky (1999), but it took till recently with the detection of gravitational waves for intermediate-mass black holes to be detected (Abbott et al. 2020). Finally, the black holes we know the most of are stellar-mass black holes. These black holes are formed when a big enough star collapses after going supernova (Celotti et al. 1999), creating a black hole from about five to tens of solar masses. For this thesis, the focus is on the last type.

1.3 X-ray Binaries

As mentioned in Boss & Keiser (2014) a large proportion of stars in the galactic neighborhood are believed to be multi-star systems. In the case of 2 components, these are called binaries. If one of the components in the binary goes supernova it can create a neutron star or black hole. In this case, such kind of binary systems are called X-ray binaries or XRBs in short. The other star in the system can as it lives through its life start accreting material onto the compact object, first modeled by Shakura & Sunyaev (1973). This accretion generates X-rays as the material falls in towards the compact object. XRBs can be separated into several different major categories which tell something about the type of donor star.

First are the High-mass X-ray binaries (HMXRB), where the donor is a massive star. This usually is an O or B star, a blue giant star sometimes even a red supergiant or Wolf-Rayet star. These stars have strong stellar winds (Höfner et al. 2003; Sandin & Höfner 2003, 2004; Mattsson & Höfner 2011) which get partially captured by the

compact object. This causes the material of the stellar wind to accrete onto the compact object.
The next type is a Low-mass X-ray binary (LMXRB), where the donor star is a main sequence star, red giant, or white dwarf. In these kinds of binaries, the donor star over the course of its life starts to fill its Roche lobe. As it reaches the edge of the Roche lobe it will start to enter into the Roche lobe of the compact object, accreting onto it.
Finally, there's a type between both which is called an Intermediate-mass X-ray binary (IMXRB), in which the donor star is an intermediate-mass star (Tauris et al. 2000; Podsiadlowski et al. 2002).

1.4 X-ray Detection

The X-ray that is emitted by this accretion can tell us a lot about what is happening inside the accretion disk. One way to shed more light on this is to use X-ray telescopes to study the physics behind these emissions.

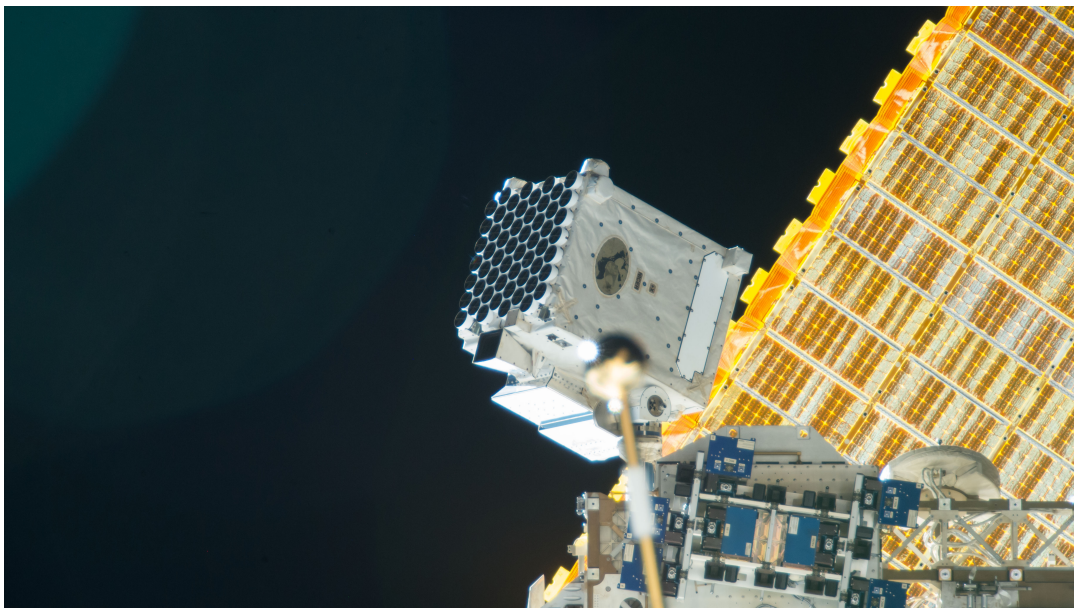


FIG. 1.1 – Picture shown of NICER as mounted on the ISS. One of the solar panels can be seen behind the telescope and on the telescope itself, its 56 X-ray detectors.
Source: https://svs.gsfc.nasa.gov/12854#media_group_9204

One such telescope is NASA's Neutron Star Interior Composition Explorer or NICER for short which is located on the International Space Station (ISS) shown in figure 1.1. It was launched and installed on the ISS in 2017 as part of the Explorer Program. With its X-ray Timing Instrument with an array of 56 X-ray detectors, it is able to

detect photons in the range of 0.2 keV - 12 keV. Using GPS, NICER is able to flag the timestamp of photons detected with a precision that is less than 300 ns as well as their energy (Gendreau et al. 2012). As its name suggests NICER was initially launched to study the interior composition of neutron stars. However, due to the very sensitive nature of its instruments, it was the perfect solution for detecting other X-ray sources as well like XRBs.

FIG. 1.2 – Picture shown of a replica of the XMM-Newton telescope in Cite de l'Espace in Toulouse. Source: https://it.wikipedia.org/wiki/File:XMM-Newton_at_Cite_de_l%27espace_4.jpg

A different but older telescope is ESA's X-ray Multi-Mirror Mission or XMM-Newton telescope. Unfortunately, unlike with NICER, no images exist of its current state. There are some computer-generated images of what it would look like and a replica exists located in Cite de l'Espace in Toulouse as shown in figure 1.2. It was launched in 1999 as part of the Horizon 2000 program. Its range of detection is 0.1 keV - 12 keV (Wilson 2005) which is fairly similar to NICER's range. However, XMM-Newton has 3 different types of instruments with fewer detectors than NICER. Its first type of instrument is the European Photon Imaging Camera (EPIC) of which it has 3. 2 of those are of the type MOS-CCD for soft photon detection (up to 5-10 keV) which have 7 detectors. The detectors have a readout cycle of around 2.9 seconds (Turner et al. 2001). The other EPIC is dedicated to hard photon detection (more than 5-10 keV) which has 12 detectors of the pn-CCD type. The hard photon EPIC is much faster than the MOS-CCD at an impressive 80 ms readout speed, which in a special mode can be brought down to 40 ms (Strüder et al. 2001). Its

second type of instrument is the Reflection Grating Spectrometer (RGS) which is a grating spectrometer consisting of 2 9 MOS-CCD arrays to be able to detect elements present in the target. Finally, it also has an Optical Monitor (OM) system which is an optical/ultra-violet imaging system to allow for simultaneous observations with the X-ray systems.

1.5 X-ray spectrum

These telescopes are then used to study the X-rays that are emitted by the XRB. This generates an X-ray spectrum of the source which is composed of several different components as can be seen in figure 1.3 which is an edited version of figure 9 shown in Gierliński et al. (1999). This is a spectrum for the black hole Cygnus-X1. This thesis will focus on such black hole XRBs. While the physics at play near the outside of the disk are fairly well known the inner hot parts of the disk close to the black hole are much less well known. Looking at the object in X-ray will uncover some of this as the inner parts of the system are the source of the X-ray emissions.

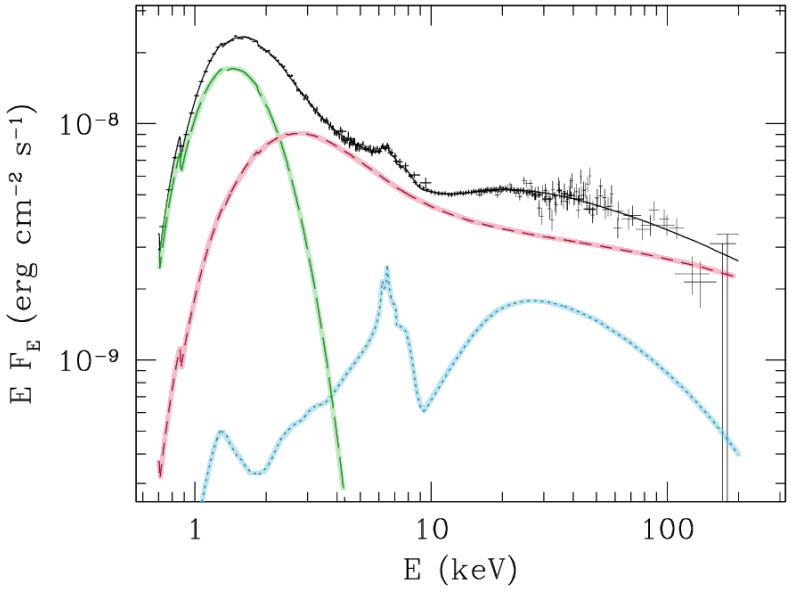


FIG. 1.3 – Shows X-ray spectrum of Cygnus-X1 as shown in Gierliński et al. (1999) figure 9. Also shown in this figure are the separate components of the X-ray spectrum. The green dashed line shows the black body radiation component of the disk, the pink dashed line is the Comptonization power law of the corona, and the blue dotted line is the reflection and reproduced photons from the corona to the disk.

The material that is caught by the gravitational field of the black hole gravitates towards a disk around the black hole. As the material is captured by the black hole

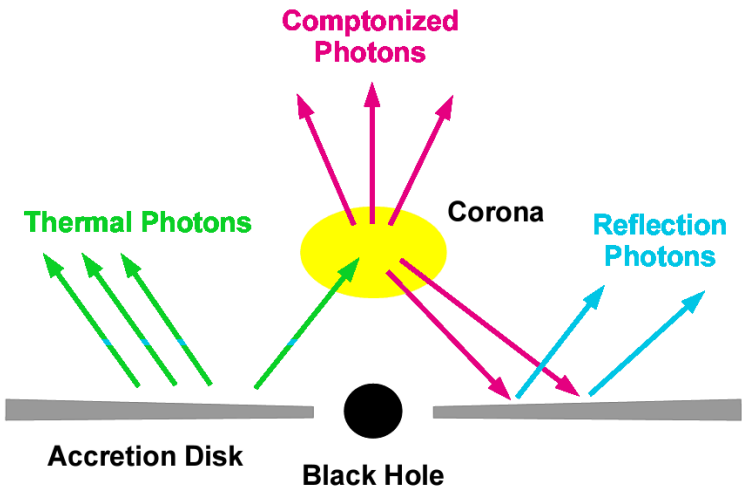


FIG. 1.4 – In this figure a schematic overview of emissions and how they flow through the system as a side-on slice. It is taken from Bambi (2021) figure 1 with altered colors to match figure 1.3. The scale nor position/shape of the corona is a true visualization of the nature of the system, it is shown this way for the sake of explanation. In green the Thermal black body emissions are shown from the accretion disk. In yellow a possible location and shape of the corona is shown. As the black body emissions get to the corona they get inversely Compton upscattered and emitted away from the corona, shown in pink. The coronal emissions are then reflected or reprocessed in the disk as shown in blue.

it all moves in the same direction around the black hole it experiences fiction which causes the speed the material has in the normal direction to the spin of the black hole to go towards zero creating a flat disk around the black hole. However, due to the conservation of angular momentum, the material ends up orbiting the black hole in its respective circular orbits according to their angular momentum as that is the type of orbit with the least energy. This friction together with the magnetic shear caused by the Magnetorotational Instability generates viscous dissipation and turbulent flow in the disk which allows for angular momentum to be transported towards the outside of the disk and material to accrete onto the black hole (Balbus & Hawley 1991; Hawley & Balbus 1991, 1992; Balbus & Hawley 1992, 1998). This inward material flow generates a lot of thermal energy in the form of black body radiation as the material heats up as it gets closer to the black hole this causes the disk to become optically thick and geometrically thin (Shakura & Sunyaev 1973; Frank et al. 2002). As black body radiation is emitted at each radius which increases as the material gets closer to the black hole, this creates a multi-temperature black body emission as can be seen in figure 1.3 as the green dashed line. With the high temperatures in the inner parts of the disk (which can reach millions or billions of degrees) the emission is mainly in the Extreme Ultra-Violet or X-rays.

The black body X-rays that are emitted by the disk go in all directions, some of these are directed toward the black hole and the area around the black hole (shown in green in figure 1.4). This area is thought to be a hot optically thin cloud of electrons close to the black hole called a corona (shown in yellow in figure 1.4). These photons are high-energy X-ray photons originating from hot plasma ($\sim 10^8$-10^9K) (Shapiro et al. 1976), these are usually referred to "seed" photons as they are the seed for the Comptonization in the corona. As these "seed" photons from the disk enter this cloud they can get Inversely Compton up-scattered to (much) higher energies generating a power-law spectrum with a cut-off at higher energies as can be seen in figure 1.3 as the pink dashed line. However, much about the corona is not known. One of these is the exact shape or geometry of the corona. Some examples of proposals for this geometry are a thin blanket over the disk close to the black as proposed in Zdziarski et al. (2021), a hot flow region between the black hole and the disk (Rapisarda et al. 2016) or even a region at the base of an astrophysical jet as described in Markoff et al. (2005). The actual geometry of the corona could be something completely different or even a combination of the proposed shapes.

As the photons get up-scattered in the corona they go in all kinds of directions again just like with the disk emission (shown in figure 1.4 in pink). This time some photons reach back to the disk (Done et al. 2007) where they either get back-scattered reflecting off the disk, which creates a hump of high-energy up-scattered photons. Some of the photons have the right energy for certain line transitions in atoms that appear in the X-ray, a famous example of this is the Kα-iron line. This reflection and reprocessing of the photons is usually classified as the reflection spectrum component of the X-ray spectrum and is shown in figure 1.3 as the blue dotted line and in blue in figure 1.4.

1.6 Accretion States

Most of the time, XRBs have been observed to be very quiet in the X-ray. However, its X-ray spectrum and brightness alter a lot over different timescales. During these moments, the disk is in quiescence and is relatively cold. On timescales of years can black holes exhibit drastic increases in X-ray emission that last weeks or months, called outbursts. It has been theorized that in the disk Hydrogen Ionisation can occur causing these outbursts (Dubus et al. 2001). At the beginning of such an outburst, the disk is about 10^4K which is very close to the ionization temperature of hydrogen. As such the disk mainly consists of neutral hydrogen. However, when the temperature rises due to for example turbulent motion. This causes the hydrogen in the disk starts ionizing, increasing the opacity of that part of the disk. As a result, the photons from that part of the disk get somewhat trapped raising the temperature, which causes more hydrogen to ionize. This creates a runaway process of ionization of the hydrogen is almost all fully ionized. This local heating causes the accretion

flow to increase which affects the neighboring radii making this heat wave propagate through the disk towards the black hole. This process makes the emission of the disk drastically increase in luminosity and is the start of the outburst (Frank et al. 2002; Done et al. 2007).

This increased accretion rate eventually gets bigger than the accretion rate of the donor star. As a result and both the temperature and pressure decrease, which consequently causes the same runaway process but in reverse. This continues until the disk returns to its quiescence state finishing the outburst. Even though this effect starts off locally, the changes in mass accretion cause different regions in the mass flow to be linked together (Frank et al. 2002; Done et al. 2007). However, unfortunately, this model is not yet fully complete and only gives part of the picture. Although in recent years it has been theorized that magnetic fields in the disk play an important role in the outburst process in the disk (Begelman et al. 2015).

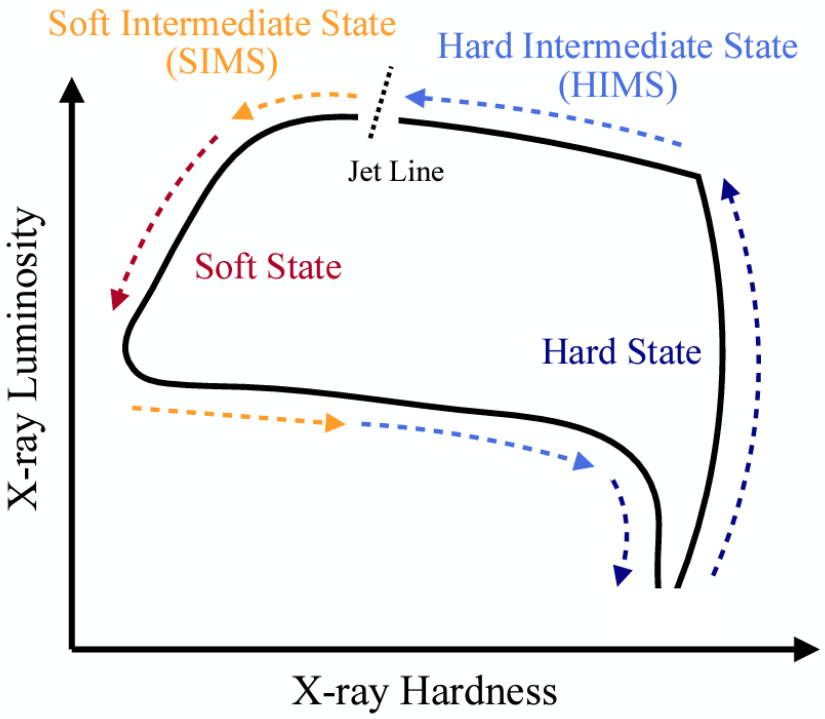


FIG. 1.5 – In this figure the evolution of an outburst is shown as it moves through the hardness (X-ray emission energy) and intensity plane (fetched from Wang et al. (2022) figure 1). The different accretion states are annotated in color as well as the corresponding direction arrow. The hard state is shown in dark blue, the hard intermediate state is shown in blue, the soft intermediate state is shown in yellow, and the soft state is shown in red.

Let's look at this process in more detail. As a black hole accretion disk goes through this outburst cycle it cycles through several different accretion states. These can be shown in a hardness-intensity diagram like shown in figure 1.5 (figure 1 in Wang et al.

(2022)), the hardness meaning the energy of the emitted photons, and the intensity relating to the total brightness luminosity of the disk. The hardness is the ratio between the earlier defined hard high energy to soft low energy photons as mentioned in section 1.4.
As mentioned before it starts out in a cold and quiescence. In this state, also called the hard state (HS), the corona gives the most emission compared to the disk. As such while the luminosity might be low the hardness of the photons that do get emitted is in the hard spectrum. In this state, the outflow is believed to be emitting photons in the radio regime, also known as the X-ray/radio correlation (Koljonen & Russell 2019). Over the course of a couple of weeks, the system starts to rise in luminosity as the hydrogen ionizes and the accretion rate increases.
The system then transitions into the next state the disk starts to heat up and generate more X-rays compared to radio emissions. In this Hard Intermediate State (HIMS) the hardness of the system starts to shift towards the "soft" regime. As this runaway process continues it continues to soften its hardness and gets into the Soft Intermediate State (SIMS). It has been found that in XRBs there's a strong coupling between ejection and accretion (Fender et al. 2004, 2009). While in the hard state, a compact and slightly relativistic jet is always present (Dhawan et al. 2000). Somewhere between the SIMS and HIMS states it quenches and transitions into a highly relativistic, discrete, and ballistic jet (Bright et al. 2020). The boundary between these two jet states is called the "jet line".
The system eventually locally depletes and the accretion rate starts to drop. As a result, the luminosity drops and the system has reached the soft state (SS). As mentioned before this causes the temperature to drop, causing the emission to harden because the disk emission moves back into the radio regime. The system goes through the SIMS and HIMS again and the highly relativistic, discrete, and ballistic jet transitions back into a compact jet but this is not well understood (Miller-Jones et al. 2012; Kalemci et al. 2013). Finally, it reaches the hard state again as the disk emissions have fully turned into radio emissions again and the luminosity drops drastically again.
It is important to distinguish between these states since, as mentioned above, they have very different accretion rates. Depending on the state either the corona (in the HS) or the disk (in the SS) are more dominant in the X-ray spectrum (Belloni & Motta 2016), or the type of jet between the two regimes on both sides of this "jet line".

1.7 Variabilities

Of course, as mentioned before the emissions of an XRB are not stationary in time. While the singular X-ray spectrum can already say quite a bit about what happens in the accretion disk and corona, to know more a time series of X-ray spectra should be observed. Analyzing these time series will give more insight into the physics

behind long and shot time-scale variabilities. One way to do this is by using Fourier transform techniques to convert the time series from the time domain to the frequency domain. This operation can bring any periodicity of the spectrum to light curves. When taking the Fourier product between two different energy bands the difference between those energy bands can be studied. The Fourier Product is taken between two time series of the same energy band and is called a power spectrum. This shows for each Fourier frequency the amplitude of the variability.
If the variability between two energy bands is to be studied Fourier techniques can also be used as is mentioned in great detail in Uttley et al. (2014). Some distinct energy bands are chosen and a cross-spectrum is calculated by using the Fourier product between the time series of both energy bands, this is also known as X-ray spectral-timing. Using this cross-spectrum the coherence between the two light curves can be calculated.
What also can be calculated from this cross-spectrum is any phase difference or lag between the Fourier frequencies of each energy band. Since the result of a Fourier transform is a series of complex amplitudes for each Fourier frequency, this phase lag can be retrieved by taking the argument of the complex amplitude. This can then be converted into a temporal lag by dividing the found phase lag by $2\pi\nu_i$ where ν_i is the mean of the Fourier frequency used.

These time lags between the light curves of different energy bands have been observed in several different sources. Several different lags can be formulated when looking at different energy bands. These lags are dependent on several different aspects of the system like which energy band but also the accretion state & geometry/scale of the disk or corona. One of the first studied lags was the hard lags between different corona energy bands, as such they are called hard PL lags. Where in the hard state where high energy corona emissions start to lag behind lower energy corona emissions. These were first studied by missions such as Ginga and later Rossi X-ray Timing Explorer (RXTE) using data above 3 keV (Miyamoto et al. 1988; Nowak et al. 1999; Grinberg et al. 2014; Altamirano & Méndez 2015), which were predecessors of XMM-Newton & NICER as mentioned in section 1.4.
Later the focus shifted towards more studies looking at softer bands of the black body disk emissions and the launches of telescopes with more sensitive equipment like XMM-Newton & Nicer (see section 1.4). Using XMM-Newton data from GX-339-4, Wilkinson & Uttley (2009) found that its disk emission variabilities can change on timescales as quickly as less than seconds to several minutes. The variabilities are the strongest for variabilities on bigger than-second timescales. Later Uttley et al. (2011) found using the same data, that for Fourier frequencies smaller than 1 Hz the soft lags between the disk (0.5-0.9 keV) and the corona (2-3 keV) were several times bigger than between the different power-law energy bands at the same frequencies. For frequencies bigger than 1 Hz this flips around with the "soft" lags becoming negative lags of around 1-2 ms. This means the mass accretion fluctuations first arrive in the corona before the power-law emissions are varied. Which in turn illuminates the disk and part emissions are reprocessed in the disk. This generates

a soft "reverberation" X-ray signal that corresponds to the light-travel delays of tens of gravitational radii (Mastroserio et al. 2018; Mahmoud et al. 2019). This "reverberation" and its role in the production of short-term soft lags along with delays in the K-α line, the broadened iron line, at similar frequencies by NICER observations (Kara et al. 2019).

1.8 Previous Models

Over the years, several models have been crafted to explain both the behavior of the hard PL lags as well as the soft disk-PL lags.
In the case of the hard PL lags 2 possible origins have been proposed. The first origination assumes the corona to be a very extended corona. In this model, the seed photons being emitted from the disk get upscattered multiple times to higher energies in the corona leading to delays. These delays are directly linked to the light-travel times and thus can cause hard PL lags to approach milliseconds on tens of seconds time scales. The corona however would need to be extended very far up from the disk on the order of thousands of gravitational radii. Which in turn needs a lot of energy to fully heat a spherical corona of that size. Due to this, it is believed to be caused by an observed hard state jet (Reig et al. 2003; Giannios et al. 2004; Kylafis et al. 2008).

A different proposed model is one where these hard PL lags are instead likely to originate from the accretion flow itself on small timescales. This may even be the origin of the broadband noise variabilities that are detected (Lyubarskii 1997; Uttley et al. 2005). The disk is taken to be a truncated disk instead of a full disk with the corona consisting of a hot geometrically thick accretion flow that increases in temperature and thus hardness as it gets closer to the black hole (Kotov et al. 2001; Arévalo & Uttley 2006). With thought to be produced by the propagation delays between radii where high variability frequencies show a decrease for both soft and hard photons emissions as the matter gets closer and closer to the black hole. Through the years this model has gotten more complex to try and explain the variability of the power spectrum as well as the dependence of the lags on both the energy and frequency. Some of these attempts were by trying to introduce discontinuities for the emission spectrum (Rapisarda et al. 2016; Mahmoud & Done 2018b), input variability signal (Mahmoud & Done 2018a) & the speed of the propagation (Kawamura et al. 2022).

Further studies focused on soft lags at different luminosities (De Marco et al. 2015, 2017) and found that for low accretion rates, the light-travel delay significantly increased to the equivalent of hundreds of gravitational radii. This could suggest the truncation of the accretion disk till these radii. Ever since the launch of NICER the studies of these soft lags have been gaining momentum. Recent studies of the

luminous black hole transient MAXI J1820+070 (Kara et al. 2019; De Marco et al. 2021; Wang et al. 2021) and compiled NICER observations of other sources (Wang et al. 2022) found a consistent pattern in the soft lags where they decrease to less than a millisecond during the hard-state rise which then sharply increases to more than ten milliseconds during the transition between the HIMS and SIMS. One explanation that could be given for the sharp rise in the soft "reverberation" lags during the HIMS, when the inner disk radius approaches close to the ISCO, would be if the corona becomes more vertically extended and jet-like (Kara et al. 2019; De Marco et al. 2021; Wang et al. 2022). This might even be connected to the relativistic jet ejections observed during the state transition according to Homan et al. (2020) and Wang et al. (2022).

As is shown in this section, a lot of progress has been made in studying the hard and soft lags. These lags seem to hold the key to important physical properties of the XRB like the extent & size of the disk and corona as well as the exact geometry of the corona. They also give insight into the accretion states and their origins. Unfortunately, more research has to be done to be able to better understand the origin of the lags between the physical components on the different time scales of variability. The individual previous models are able to explain the different kinds of lags separately. However, when trying to combine them together the models fail to properly explain the observed phenomena.
The problem with Comptonisation delay models is that they require an extremely extended (thousands of gravitational radii) coronae in the hard state. However, this is inconsistent with the reverberation delays that arise from the soft lags. Meanwhile, large PL-disk lags that are observed at low frequencies would only further increase this inconsistency if they correspond to the light-travel delays between the seed photons from the disk and the corona (Uttley et al. 2011).
The models that use the propagations of accretion fluctuations through a hot flow to explain the low-frequency PL-disk lags seem to clash with the models that explain the hard lags over the same frequency range with the hot-flow propagation.

1.9 Thesis overview

In the previous sections (chapter 1), the importance of these lags has been explained. Their link to constraining the size and extent of the disk & corona as well as its geometry. Which in turn tells us something about the physics behind the variabilities in XRBs and their emissions.
Chapter 2 will be going into the details of a new model made by Uttley & Malzac (????) which combines the previous models and tries to put restraints on the corona while explaining both the hard and soft lags at the same time. It will go into the coding details of the model as well as some of the maths behind it. It will also mention some of the improvements to the original model that have been made.

Once this has been explained chapter 3 will show the results of these improvements will also be announced along with their effects on the total run time. As well as some fitting results that were found using the improved model.
Finally, chapter 4 contains the findings of the results of this thesis. It also goes into some of the problems that happened during the thesis as well as some improvements for the future. It also gives a final conclusion to the thesis.

Chapter 2

Model

As mentioned in section 1.8, the models created in recent years to explain the variabilities and lags in the emissions of X-ray black hole binaries have somewhat lacked in their ability to explain multiple components of the hard and soft lags observed. Recently, a new model by Uttley & Malzac (????) has appeared to try and simultaneously model multiple components of the previous models while trying to put constraints on the extent and size of the disk and corona as well as the geometry of the corona. They propose a propagation fluctuations model for the lags that would give both the disk-PL hard lags at low frequencies as well as the soft lags at high frequencies. They also link the lags to the coronal geometry to explain the observed lag evolution. Following their model, they show that the observed features naturally show up while accounting for variations of the seed photons that can illuminate the corona as mass accretion fluctuations propagate through the disk to the corona. As the fluctuations move inward, the seed photons are modulated before they heat the corona. Causing the corona to cool and heat after a large viscous time-scale delay. Allowing for the production of large hard PL lags even when the corona is small, ~~as well as radii in the inner disk.~~

They believe that if the seed photon variations, the switch to negative, are accounted for, the soft lags are caused by the reverberation. As the fluctuations propagate through the inner regions of the disk, this reverberation causes a slow rise in seed photons. The negative PL-disk lags, as well as the hard PL lag, would increase as a result of a change in coronal geometry. A vertically extended corona would respond to the seed variations from further out in the disk, causing larger lags. The lags that their model predicts would link both the soft and hard lags to the propagation delays instead of the light-travel times. This prediction implies that the corona is more compact than those implied by the reverberation light-travel delay and Comptonization models. However, the change in coronal geometry, as the system changes states, remains similar.

2.1 Effects of Corona emission

To be able to model the variation of a quantity $f(t)$ as it responds to a signal perturbation $s(t)$, a so-called impulse response function $g(\tau)$ that helps describe the perturbation in $f(t)$ as a convolution between the signal perturbation and the impulse response. Where this perturbation to $f(t)$ would look as follows, $\delta f(t) = \int_{-\infty}^{\infty} s(t-\tau) * g(\tau) d\tau$, where τ is the time delay and the rest is as mentioned above. In the paper, they use something called the photon index. Which links the luminosities of the heating and cooling of the corona to the amount of photon flux density variation depending on time and energy. For this, they assume that the Comptonizing corona is in thermal equilibrium. The cooling of the corona happens through the inverse Compton scattering of seed photons from the black body emitting disk at luminosity L_s. Meanwhile, the corona is heated via an unknown heating mechanism, heating the corona at a luminosity L_h. If these luminosities can be taken as a time series, a spontaneous photon index Γ can be constructed to be related to the ratio of the seed photons to the heating photons,

$$\Gamma(t) = \Gamma_0 \left(\frac{L_s(t)}{L_h(t)} \right)^{\beta}, \tag{2.1}$$

where Γ_0 is the photon index for equal heating and cooling luminosities, and β is a parameter that describes the change in this ratio in a Comptonizing medium. It depends on the optical depth and temperature of the corona varying from 0 to 1, where at 0 it is insensitive to the heating and seed photons, while at 1 it is very sensitive to these photons (Pietrini & Krolik 1995; Beloborodov 2001).

However, the most crucial part of the model is the direction of the relation. Not the precise relation between the photon index and the ratio for the heating vs. seed luminosities, but rather what sets the sign of the lags. They believe that the positive correlation between the photon index and the ratio is natural because of the corona's conservation of total luminosity and photon number. The thermal equilibrium assumption can only be used if the corona's heating and cooling time scales are small compared to the luminosities' variability time scales. The relatively compact corona considered in their model should ensure that.

Uttley & Malzac (????) uses the following approximation for the photon flux density variation. They take the seed photons to be monochromatic with the energy E_s, and this makes the total number of photons L_s/E_s. Under the assumption that the total number of photons is conserved during the process of Comptonization and the cut-off photon energy is E_{cut} is much greater than E_s, the photon flux density variation can be constructed as follows,

$$N(E,t) = \frac{L_s(t)(\Gamma(t)-1)}{E_s^2}\left(\frac{E}{E_s}\right)^{-\Gamma(t)}, \tag{2.2}$$

where $N(E,t)$ is this photon flux density, L_s is the seed photon luminosity and Γ is the photon index.

When assuming the perturbations are small, the equations 2.1 and 2.2 can be linearized, and they find the following functions for impulse responses for the photon index and the photon flux density,

$$g_\Gamma(\tau) = \langle\Gamma\rangle\beta\left(\frac{g_s(\tau)}{\langle L_s\rangle} - \frac{g_h(\tau)}{\langle L_h\rangle}\right), \tag{2.3}$$

$$g_{pl}(E,\tau) = \langle N(E)\rangle\left([1-u(E)]\frac{g_s(\tau)}{\langle L_s\rangle} + u(E)\frac{g_h(\tau)}{\langle L_h\rangle}\right), \tag{2.4}$$

where $\langle A\rangle$ is the average of A over time, and A is anything perturbed by the seed and heating luminosities changes. In this case, these are the luminosities, photon index, and photon flux index. The impulse responses for the seed and heating g_s & g_h depend on the physical system producing the variation in the luminosities. $u(E)$ is the energy dependence of equation 2.4 and is defined as follows,

$$u(E) = \beta\langle\Gamma\rangle\left(ln\left(\frac{E}{E_s}\right) - \frac{1}{\langle\Gamma\rangle - 1}\right), \tag{2.5}$$

it gives the energy-dependent impulse response shape of the impulse response of the photon flux density. It takes into account the seed and heating impulse responses according to the energy of the PL photons in the energy band that is calculated. $u(E)$ looks at the pivoting of the power-law spectrum around a pivoting energy E_{piv} where $g_{pl}(E_{piv},\tau) = 0$ which yields,

$$ln\left(\frac{E_{piv}}{E_s}\right) = \left[\beta\langle\Gamma\rangle\left(1 - \frac{g_h(\tau)}{\langle L_h\rangle}/\frac{g_s(\tau)}{\langle L_s\rangle}\right)\right]^{-1} + \frac{1}{\langle\Gamma\rangle - 1}, \tag{2.6}$$

which introduced the fact that the impulse response of the seed photons and the heating photons are related to the pivoting energy.
When looking at the case for $\beta = 1/6$ (Beloborodov 2001) and the photon index being

the typically observed values of $\langle\Gamma\rangle \in (1.4, 2.5)$, several regimes can be defined using this equation. In the case when the seed luminosity is dominating, the minimum pivot energy can be found to be approximately 21 to 885 times the seed energy. Following the equation, one can derive that the heating impulse response is much smaller than the average heating luminosity and thus can be set to zero.
As the ratio between the seeding and heating impulse responses, the pivot energy also rapidly changes until the ratio is one ($g_h(\tau)/\langle L_h\rangle = g_s(\tau)/\langle L_s\rangle$). At this point, the pivoting energy has reached ∞, and only the normalization changes; the photon index does not change.
As the impulse response for the heating increases even more, the pivot energy goes back down, eventually limiting itself to a maximum of around 2 to 12 times the energy of the seed emissions. Once it has reached this state, the seed impulse response becomes effectively zero, so it is assumed as being 0.
This changing of the pivoting energy is the main reason that differences between the time responses between the heating and seed emissions are seen. This difference leads to a spectral pivot point that changes with the emissions, which in turn leads to significant time-delay-dependent changes where the largest and smallest variations are seen in the PL flux. It automatically shows that the lags are energy-dependent in the PL emissions as well as an energy dependence in the variability of the amplitude of the PL flux.

2.2 Effects of the accretion flow

Uttley & Malzac (????) assumes that the mass accretion rate fluctuations that propagate through the disk are the main driving signals for the lags. They simplify the propagation such that the propagation is non-diffusive, which allows for the amplitudes and power spectral shapes of the fluctuations in the mass accretion to be preserved while they propagate inwards to the center of the disk. The condition for these assumptions is that the accretion disk is "classical" and the time-scale of the fluctuations is bigger or approximately equal to the viscous time-scale at the radius it was generated at (Churazov et al. 2001; Mushtukov et al. 2018).

The radial drift velocity of the accretion variations affects the response delay τ. Because in the model mentioned in Uttley & Malzac (????) the signal from multiple radii are combined, they believe it to be useful to set $\tau = 0$ to be the moment in time when the signal propagating inward reaches the innermost radius of the accretion flow, r_{in}. Allowing them to define a negative and fixed propagation time delay for each radius r (which are in units of the gravitational radius R_g) as follows,

$$\tau(r) = \int_{r_{in}}^{r} d\tau = \int_{r_{in}}^{r} dr/v_r, \tag{2.7}$$

where v_r is the radial drift velocity for a disk with scale-height h and a dimensionless viscosity parameter α,

$$v_r = -\alpha(h/r)^2 r^{-1/2}. \tag{2.8}$$

Since the radial drift velocity is defined as a faction of the speed-of-light, c, the lags and delays are expressed naturally in units of the light-crossing time for 1 R_g.

They assume for simplicity's sake that the light-travel time lags, from the disk to the corona, are much smaller than the lags due to the propagating fluctuations considered here. Another assumption is that the inner zero-torque boundary of the accretion flow (including the corona) is r_{in}, corresponding to the innermost stable circular orbit or ISCO. The radial dependence of the propagation delays, as well as the time scales of the variability, is significantly affected by the way the disk aspect ratio h/r is radially scaled.

All the luminosities used in the model are defined as fractions of the total dissipated power from the accretion flow. The accretion flow dissipates power in an annulus of size dr at radius r defined as,

$$f_{diss}(r)dr = \frac{r^{-2}[1 - (rin/r)^{1/2}]dr}{\int_{r_{in}}^{r_{out}} r^{-2}[1 - (rin/r)^{1/2}]dr} \tag{2.9}$$

where r_{out} the outer radius of the accretion flow (Frank et al. 2002). The corona is also assumed to be powered by this accretion and may be a hot flow-like compact corona with radius r_{cor}, which is also the truncation radius of the accretion disk.

2.2.1 Heating of the corona and seed luminosity

Assuming all the power that is dissipated from within the r_{cor} goes into heating the corona, a coronal heating impulse response function can be defined as follows,

$$g_h(\tau)d\tau = \begin{cases} f_{diss}(r)dr, & if\ r \in (r_{in}, r_{cor}] \\ 0, & otherwise \end{cases}, \tag{2.10}$$

setting this impulse response to zero outside the corona. A fraction of the seed photons that are generated in the disk (as mentioned in section 1.5) at radius r reach the corona. From this, the seed photon impulse response of the disk photons produced by viscous dissipation can be defined as follows,

$$g_{s,diss}(\tau)d\tau = \begin{cases} f_{d\to c}(r)f_{diss}(r)dr, & if\ r \in (r_{cor}, r_{fluc}] \\ 0, & otherwise \end{cases}, \tag{2.11}$$

where $f_{d\to c}$ is this fraction of photons from the disk reaching the corona and r_{fluc} is the radius at which the accretion fluctuations start. For which it is assumed that the fluctuations start outside the corona or, in other words, $r_{fluc} > r_{cor}$. The dissipation-related emissions are denoted with a "diss" subscript to make a clear distinction between disk/seed emissions produced directly by dissipation in the flow and the emissions produced by reprocessing in the accretion disk of the power-law emissions. The disk-to-corona fraction calculations depend entirely on the exact assumed coronal geometry. They assume that the only source for the seed photons for the corona is the accretion disk. According to Uttley & Malzac (????) it is not h to allow for a fraction of this viscous dissipation that powers the corona to be converted into intern seed photons via, for example, synchrotron emissions if the corona is magnetized Veledina et al. (2013).

As mentioned in section 1.5, the emission of the corona, which both are the up-scattered seed photons and heating photons, are isotropically emitted. A fraction of these photons are intercepted and reprocessed by the disk and produce an X-ray reverberation signal caused by the short light-travel time delay compared to the X-ray variations in the coronal emission. A part of this reverberation signal returns to the corona, where it interacts the same way as the other seed photons, thus essentially being an extra source of seed photons. This fraction of the total coronal luminosity which returns to the corona is defined as follows,

$$f_{return} = 2\pi \int_{r_{cor}}^{r_{out}} f_{d\to c}(r)f_{c\to d}(r)rdr, \tag{2.12}$$

the fraction of the coronal photons that end up in the disk as a function of radius is $f_{c\to d}$.

These returning photons will, once reaching the corona, will be Compton-scattered. Some of these photons will be reprocessed again, yielding higher and higher orders of luminosity being returned. As such, the total returning luminosity at time t can be defined as,

$$\begin{aligned} L_{return}(t) &= f_{return}(L_h(t) + L_{s,diss}(t) + L_{return}(t)) \\ &= f_{return}\left(\frac{L_h(t) + L_{s,diss}(t)}{1 - f_{return}}\right) \end{aligned} \tag{2.13}$$

where $L_{s,diss}$ is the seed luminosity that is directly produced by dissipation in the accretion flow, as mentioned before in this section.

The other fraction of the luminosity that is returning is in the form of backscattered hard emissions, as mentioned in section 1.5. A smaller fraction of the luminosity that returns are isolated fluorescent emission lines such as the K_α line from iron. Finally, the remaining luminosity may be thermalized by the disk. This component is usually composed of softer reprocessed emissions, which can start to approximate a smooth black-body continuum of the disk thermal emissions for higher disk densities, which are common in X-ray binaries (Ross & Fabian 2007; García et al. 2016; Mastroserio et al. 2021). This "thermal reverberation" is assumed to return seed photons to the corona at energies similar to the seed photons that are emitted by viscous dissipation. It can be shown as a fraction of the returning luminosity f_{therm}. The implicit assumption made in f_{therm} is that the remaining return photons do not affect the spectral shape of the corona or heat/cool the corona. This thermal fraction acts as an efficiency factor for the amount of returning luminosity that is used for coronal cooling. Equation 2.13 can be combined with this thermal fraction to define an impulse response for the seed luminosity with respect to the thermal reverberation,

$$g_{s,rev}(\tau)d\tau = \frac{f_{therm}f_{return}}{(1-f_{return})}[g_{s,diss}(\tau) + g_h(\tau)]d\tau, \tag{2.14}$$

where $g_{s,diss}$ is the impulse response for the seed luminosity with respect to the dissipation as mentioned in equation 2.11 and g_h is the impulse response for the heat luminosity as mentioned in equation 2.10.

2.2.2 Disk emissions and thermal reverberation

For the black-body emission caused by viscous dissipation, some of the emission never gets reprocessed by the corona and is directly detected. With all the viewing angles of the observer at $r = \infty$ averaged, the impulse response of the disk with respect to the dissipation can be defined as follows,

$$g_{d,diss}(\tau)d\tau = \begin{cases} (1 - f_{d\to c}(r))f_{diss}(r)dr, & if\ r \in (r_{cor}, r_{fluc}] \\ 0, & otherwise \end{cases}, \tag{2.15}$$

where f_{diss} is the fraction of dissipation of the total dissipated power from the accretion flow as shown in equation 2.9. Equation 2.15 is only non-zero between the edge of the corona and the start of the fluctuation, just like equation 2.11.

In order to conserve luminosity, the direct disk emission needs to be reduced by the fraction of the total disk emission that is intercepted by the corona. However, due to its dependence on the geometry of the corona, the angular dependence of this component needs to be taken into account. Another correction needed is the disk reverberation signal mentioned earlier in this section, which, instead of returning to the corona, goes directly to the observer. The fraction of luminosity from the corona that is intercepted and reprocessed by the disk can be used for this correction. This relation is defined as,

$$f_{rev} = 2\pi \int_{r_{cor}}^{r_{out}} f_{c\to d}(r) r dr, \tag{2.16}$$

which sums over all the disk radii between the outer disk radius and the start of the corona. Since the returning luminosity is not directly observed, the reverberation signal needs to be corrected for the contribution of the luminosity returning to the corona. Of this reverberation signal, only the thermal part is considered in the model. This component influences the total impulse response of the thermal emissions of the disk. As such, the impulse response for the disk emissions with respect to the reverberation signals is defined as,

$$g_{d,rev}(\tau)d\tau = \frac{f_{therm}(f_{rev} - f_{return})}{(1 - f_{return})}[g_{s,diss}(\tau) + g_h(\tau)]d\tau, \tag{2.17}$$

where f_{rev} is the total fraction of luminosity that goes from the corona to the disk (see equation 2.16), f_{return} the total fraction of luminosity that goes from the disk to the corona and returns to the disk (see equation 2.12), $g_{s,diss}$ is the seed impulse response with respect to dissipation (see equation 2.11), and g_h is the heat impulse response mentioned in equation 2.10.
Then, finally, summing both the dissipation and reverberation impulse responses for their respective parts of the system. A disk impulse response can be defined as

$$g_d(\tau)d\tau = [g_{d,diss}(\tau) + g_{d,rev}(\tau)]d\tau, \tag{2.18}$$

where $g_{d,diss}$ is the disk impulse response with respect to the viscous dissipation (see equation 2.15) and $g_{d,rev}$ is the disk impulse response with respect to the reverberation as mentioned in equation 2.17. Similarly, the seed impulse response can be defined as,

$$g_s(\tau)d\tau = [g_{s,diss}(\tau) + g_{s,rev}(\tau)]d\tau, \tag{2.19}$$

where $g_{s,diss}$ is the seed impulse response with respect to the viscous dissipation (see equation 2.11) and $g_{s,rev}$ is the seed impulse response with respect to the reverberation as mentioned in equation 2.14.

By integrating these impulse responses for the heating, disk & seed components over all radii, the time-averaged luminosities for these can be found. For this, the radius at which a fluctuation occurs is assumed to be at the outer radius of the disk. For example, for a component A it would become $\langle L_A \rangle = \int_{\tau(r_{in})}^{\tau(r_{out})} g_A(\tau) d\tau$. By using the definitions for the impulse response functions described in this chapter, we can conclude that the total time-average luminosity reaching the observer is conserved as follows,

$$\langle L_{total} \rangle = \langle L_{d,diss} \rangle + \langle L_{d,rev} \rangle / f_{therm} + (1 - f_{rev})(\langle L_h \rangle + \langle L_{s,diss} \rangle + \langle L_{s,rev} \rangle / f_{therm}) = 1, \tag{2.20}$$

where f_{therm} is the correction fraction for the non-thermal part of the reflection spectrum and f_{rev} is the correction fraction of coronal luminosity reaching the observer.

Using these, the impulse response of the power-law emission (see equation 2.4) can be calculated using the derived impulse responses and time-average luminosities for the heating and seed components from this section. These time-average luminosities can also be used to finally calculate the time-averaged photon index using equation 2.1, which then is used in the energy dependence (see equation 2.5) of the power-law impulse response. The normalization factor of this impulse response can now also be calculated.

2.3 Spectral-timing products

A fluctuation signal originating at radius r_{fluc} generates a response in the spectral timing properties and can be determined using the impulse response functions derived in this chapter. However, it is thought that these propagating fluctuations originate in reality over a range of radii, summarized in Lyubarskii (1997) in several other observations and simulations. Uttley et al. (2005) mentions two of these observations: the broadband nature of the observed power spectra and the relations between the multi-time-scale & the Root-Mean-Square flux. Simulations of accretion flow also confirm that model general relativistic magnetohydrodynamic accretion models (Hogg & Reynolds 2016; Bollimpalli et al. 2020). The impulse response functions in this chapter relate the delay times to specific radii, making it fairly easy to introduce multi-location signals by using impulse responses with different starting delays.
An impulse response with respect to the fluctuations can be defined as follows,

$$g_{fluc}(\tau) = \begin{cases} g(\tau), & if\ \tau \geq \tau_{fluc} \\ 0, & otherwise \end{cases}, \tag{2.21}$$

where $g(\tau)$ is the overall impulse response and τ_{fluc} is the starting delay time at r_{fluc} which can be yielded from equation 2.7. This time delay must be negative since the innermost delay is set to zero.

The conversion from the time domain to the frequency domain needs to be made in order to find the power spectrum and cross-spectrum. This conversion can be done using the Fourier transform as mentioned in section 1.7. Now, if n_{fluc} signals would be defined, the power spectrum can be broken up into separate parts for each fluctuation signal originating from $r_{fluc,i}$ with its corresponding time delay $\tau_{fluc,i}$. This resulting Fourier power spectrum is defined as,

$$P(\nu) = \sum_{i=1}^{n_{fluc}} P_{fluc,i}(\nu)|G_{fluc,i}(\nu)|^2, \tag{2.22}$$

where $P_{fluc,i}$ is the ith Fourier power spectrum, which depends on the Fourier frequency ν. $G_{fluc,i}$ is the Fourier transform of the impulse response (see equation 2.21 at the location mentioned before. While in the power spectrum, the absolute value of the Fourier-transformed impulse response was used, for the cross-spectrum, those of different energies are used and thus are denoted with 1 and 2. The resulting Cross-spectrum is defined as follows,

$$C_{1,2}(\nu) = \sum_{i=1}^{n_{fluc}} P_{fluc,i}(\nu)G_{fluc,i,1}(\nu)G^*_{fluc,i,2}(\nu), \tag{2.23}$$

where the Fourier transform of the first impulse response and the complex conjugate of the second impulse response are multiplied by each other to get the cross-spectrum, the resulting frequency-dependent time lag is calculated from the argument of this cross-spectrum and normalized by $2\pi\nu$.

For simplicity's sake, Uttley & Malzac (????) assumes that the individual power spectra of the accretion rate fluctuations can be described by Lorentzian functions that are parameterized by a peak frequency of $\nu_{pk,i}$, a quality factor Q_i which is roughly the frequency normalized by it is full-width at half maximum, and a fractional Root-Mean-Square rms_i. Pottschmidt et al. (2003) describes a more common form of this Lorentzian function. This is done by determining the resonance frequency $\nu_{res,i}$ and normalizing factor R_i defined as follows,

$$\nu_{res,i} = \nu_{pk,i}/\sqrt{1 + 1/(4Q_i^2)};\ R_i = rms_i/\sqrt{0.5 - tan^{-1}(-2Q_i)/\pi}, \tag{2.24}$$

with which the power spectrum for each fluctuation can be defined as,

$$P_{fluc,i}(\nu) = \frac{2R_i^2 Q_i \nu_{res,i}}{\pi(\nu_{res,i}^2 + 4Q_i^2(\nu - \nu_{res,i})^2)}, \tag{2.25}$$

which together with equation 2.21 & 2.25 the cross spectrum and power spectrum can be calculated.

While the linear model described by Uttley & Malzac (????) does not take the viscous diffusion effects on mass accretion and its fluctuations as they propagate through the flow into account. It does suppress the variations significantly on time scales that exceed the time scale of the local viscous propagation (Churazov et al. 2001; Mushtukov et al. 2018). As such, they assume for simplicity that the signal peak frequencies are required to correspond only to the time scales that exceed the propagation delay flowing from that radius. However, the model calculations themselves allow for an arbitrary choice of the number of radial locations of the Lorentzian signals and their parameters.

An implicit assumption made from the model, which comes from the fact that this model builds on the propagating fluctuations model in Uttley et al. (2005), is that the mass accretion rate variations combine additively instead of multiplicatively, which is a crucial assumption to be able to combine the signals from different radii.

2.4 Numerical assumptions

While the theory described above with its simplifications and is able to describe the physics fairly well, it unfortunately still needs some more simplifications in the form of numerical assumptions to make it work with the discrete data sets to fit as well as being able to work out properly with a computer model. Uttley & Malzac (????) makes several numerical calculations to find these assumptions and puts some of them as default parameters in the computer model.

For starters, they define n_{rad} contiguous, geometrically spaced radial bins of radii r_j between a r_{in} which the model fits and a value of $r_{flux,max}$ which is the outermost radius mass-accretion functions are to be generated in the disk. Between $r_{fluc,max}$ and r_{out}, another set is defined as n_{rad} contiguous, geometrically spaced radial bins for the area of the disk where it is not variable. These are used to calculate the mean

luminosities of the disk and seed emissions. It is also used to find the contribution of the impulse response within the $r_{flux,max}$. This radius is set to be the radius for which input signals are generated with a minimum frequency of 0.03 Hz. Setting the radius at this value produces a power spectral break at low frequencies similar to those seen in the power spectra of Black Hole XRBs in the hard state.

Since these bins have a physical size, the propagation across a radial bin causes a certain amount of delay. This delay in units of R_g/c is defined as,

$$d\tau_j = s_{\tau,10} \left(\frac{r_j}{10}\right)^{n_\tau} r_j^{\frac{1}{2}} dr_j \tag{2.26}$$

where n_τ defines the way that the delay is dependent on a change in a radius just like the power-law index, $s_{\tau,10}$ is the normalizing constant for this radial drift at a radius of 10 R_g in units of the dynamical time-scale at this radius. In a standard disk, this normalization part and the part depending on the power-law index is about the same as the scale height ratio squared divided by the viscosity parameter α, $s_{\tau,10}(r_j/10)^{n_\tau} \simeq \alpha^{-1}(r/h)^2$. Assuming a constant viscosity parameter, this means that for an aspect ratio h/r that is radially constant, they assume that $n_\tau = 0$ and that if the scale height would be constant $n_\tau = 2$. These normalization and exponent values are used as fitting parameters in the model and thus can be fi[illegible]zed by using them in the fit to fit the data.

Uttley & Malzac (????) then continues to assume things about the effects of this accretion disk variability on the spectral timing properties that are being calculated. They assume that the variations in the mass accretion are continuously generated only across the disk, meaning that no variability is introduced inside the corona. In other words, the input signals are only generated for $r_j \in (r_{cor}, f_{fluc})$. In the appendix B of Uttley & Malzac (????) they show that adding an intrinsic variability for the coronal accretion is likely only to affect the spectral-timing properties at the highest Fourier frequencies.

Then, from the time-scale radial delays from equation 2.26, the peak frequencies for the signals of the Lorentzian disk variabilities can be found by doing the inverse of that equation,

$$\nu_{pk,j} = s_{\tau,10}^{-1} \left(\frac{r_j}{10}\right)^{-n_{tau}} r_j^{-3/2}, \tag{2.27}$$

which is equation 2.26 integrated over the radial bin. They then assume that the signal Root Mean square across the radial bins and the quality parameter is constant. Allowing for the power-spectral shape to be roughly $P(\nu) \propto 1/\nu$ over a broad frequency range, which is similar to what is observed in BHXRBs that are in their hard

state (Heil et al. 2015). A kind of break is introduced for the Lorentzian frequency at a radius of $f_{fluc,max}$ for low frequencies. The constant Lorentzian Root Mean Square values are used as the scales of the observed amplitudes of the power spectra but do not alter their shape since the model uses values that correspond to about 1% of the integrated accretion fluctuations. However, it does not matter much since they use a linear model, but the magnitude of these values has non-linear effects and thus is important to take into account. These are caused by variations in the power-law photon index and the multiplicative nature of the mass accretions that are implied by the Root Mean Square-flux relation, as mentioned in this section. In appendix C of Uttley & Malzac (????) they show using numerical simulations that until the accretion fluctuations with integrated Root Mean Square of 40%, these effects stay negligible, which is consistent with the observations of X-ray variability of BHXRBs in the hard state.

Since the model uses a linear binned input, the geometrically spaced impulse response bins (in units of R_g/c) need to be re-binned to a uniform grid of time bins (in units of seconds). The size of such a bin dt_{bin} becomes one-fourth of the smallest propagation delay across a radial bin,

$$dt_{bin} = \left(\frac{GM_{BH}}{4c^3}\right) d\tau_{j,min} \tag{2.28}$$

where M_{BH} is the black hole mass, c the speed of light, and $d\tau_{j,min}$ the earlier mentioned smaller propagation delay across a radial bin. To try and speed up their calculations, they take this smallest delay to be the delay across the radial bin in the innermost disk at r_{cor}. They then sum the coronal impulse responses into the single adjacent delay bin. In other words, this assumes the coronal propagation delays to be zero. In Uttley & Malzac (????) Appendix B shows that this only has a slight effect on the spectral timing properties they are calculating.

Additionally, the rebinned impulse responses are padded with zeros to accommodate for frequencies lower than 0.01 Hz. This way, they allow for a total time duration of at least 16 times more than the maximum delay time. They also assume the blackhole to have a mass of 10 $M_\odot$, which yields a time scale on the order of 10^{-5}. Finally, by keeping track of each geometrically spaced bin at which start and stop impulse response bins, they contribute to the variability signal, the frequency-dependent spectral timing properties can be calculated using equations 2.21 to 2.23.

2.5 Model and conversion

The model mentioned in Uttley & Malzac (????), and described in the previous sections, was initially made in the Python programming language (Van Rossum & Drake

2009). Using Python modules like NumPy (Harris et al. 2020) and SciPy (Virtanen et al. 2020), the model was created in Python using functions to calculate the necessary values were defined. However, making such a model in Python has its pros and cons. In Python, it is relatively easy to make the model work using the modules mentioned before. However, Python is not known to be memory efficient, which becomes especially important when using extensive and precise calculations for the setup of the model. Another problem with making it in Python is that not everyone will be able to quickly bring alterations into the model unless they are well versed in its code. As such, this project was created to look into the viability of changing the code base to a compiled language like C++ (Stroustrup 2000). Allowing for speed/run time improvements compared to Python due to C++'s memory handling.

Another benefit of converting to C++ would be the ability to alter the model so it could be fitted by a standardized fitting program like XSPEC[1] (Arnaud 1996). Allowing for easier community-driven development on the model due to the standardized nature of XSPEC models. Which makes it easier to understand what the model is doing.
A downside to using a compiled language like C++ is that fewer modules are present that can handle the calculations needed, like they are done in Python using NumPy and SciPy. As such, some of these functionalities need to be defined manually in the code. Luckily, one of the most important and challenging parts of the calculations, the Fourier Transform, has a C++ library that has a function that can do these calculations; it is called "pocketfft" and was developed by Reinecke (2022). The other functions used in Python were manually redefined in C++. While it may not have been entirely efficient, this will be discussed in Chapter 4.
To be able to compare the original Python model and the converted C++ model, a bridging module was necessary that could convert the functions and classes defined in C++ and compile them into functions and classes that were able to be used in Python. One such method is using pybind11 (Jakob et al. 2017), which allows for the creation of macros that, while compiled with Python, can then be used to call the functions from within Python. This way, the functions from the C++ model could directly be compared to the Python functions and even be used instead of the Python functions in the fitting pipeline using a Jupyter (Kluyver et al. 2016) notebook that was provided for the project.
During this conversion, some improvements have been made to the code to speed up its run time. Among these improvements are the untangling of nested code, the removal of redundant code, and the removal of unnecessary duplicate code.

[1] While this was initially the plan, this has not been done in this project

Chapter 3

Results

As mentioned in the previous chapter, the Python model has been converted to C++. However, not all functions were able to be improved in terms of run time; this is shown in detail in the next section (section 3.1). In section 3.2, the converted model was used to fit XMM-Newton data of GX399-4 in the hard state taken with XMM's EPIC-pn instrument (see section 3.2.1) as well as MAXI J820 NICER spectral-timing data (see section 3.2.2).

3.1 Converted model

Each function will now be shown in detail, as well as any changes that have been made and how those changes affect the run time. There are a few ways this has been done, as can be seen in the following sections.

3.1.1 Little to no changes

For a lot of the functions, the only necessary change was literally translating the code from Python to C++. A small amount of extra code and loops was needed in most cases because of the lack of Numpy Arrays, Numpy & SciPy functions inside C++.

A function like the find_nearest function had some of the functionality behind Numpy functions and arrays (as can be seen in figure 3.1). It takes in an array and a value, after which it finds the closest value in the array and returns the value and index. Another function where this happens is the calc_cross_psd function (see figure 3.2), which calculates the weighted cross and power spectra as well as the signal power spectra. Calculated using the frequencies, bins sizes, impulse response functions, the scale of the delay, the maximum signal radius, peak frequency, quality factor, and

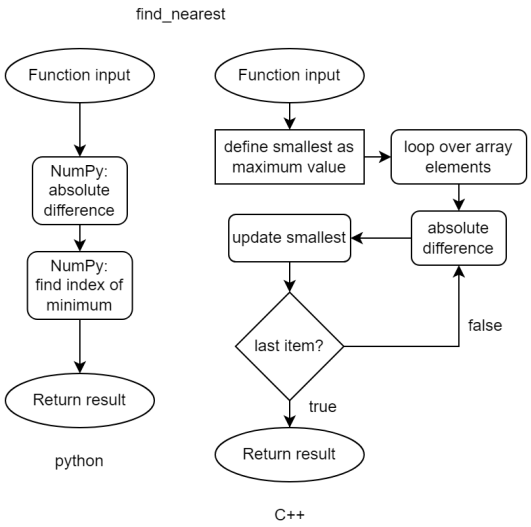


FIG. 3.1 – In this figure, the flow charts for the find_nearest function can be seen for both the Python and C++ versions of the model. It shows how, due to the use of NumPy arrays and functions in the Python model, the structure of this function is very straightforward. However, because this is done in the C++ function, it reveals the structure better.

root-mean-square.

Some functions that have a console output in the Python version have an added boolean in the C++ called "dbg" to enable/disable the output.
One of these functions is the function that calculates the timing parameters, calc_timing_params (see figures 3.3 & 3.4). ~~It calculates the variability time scale, Lorentzian frequency, quality factor, and root mean square.~~ It takes in the radial bins and information about the corona and the disk to calculate the variability time-scale, Lorentzian frequency, quality factor, and mean square.
Another such function is the calculate_stprod_mono function (see figure 3.5) which calculates the frequencies, photon lags, time lags, power spectrum densities, the number of the impulse response function bins, the number of bins to spread the impulse response function over, its edges, and the scale of the delay bins, bin size, impulse response function, mean and outer flux for the channel of interest. Calculated using a multiplication factor for the number of bins, the energy bins, energy bands, the impulse response functions for the flux, disk, photon index, propagation delays, minimum propagation delay fraction, maximum signal radius, Lorenzian frequency, quality factor, root-mean-square, and delay scale.

The biggest chunk of functions has some extra loops with respect to the Numpy arrays.
The calc_dispfrac function (see figure 3.6) calculates the viscous dissipation for each radial bin normalized over the total dissipation fractions. It also calculates the flow

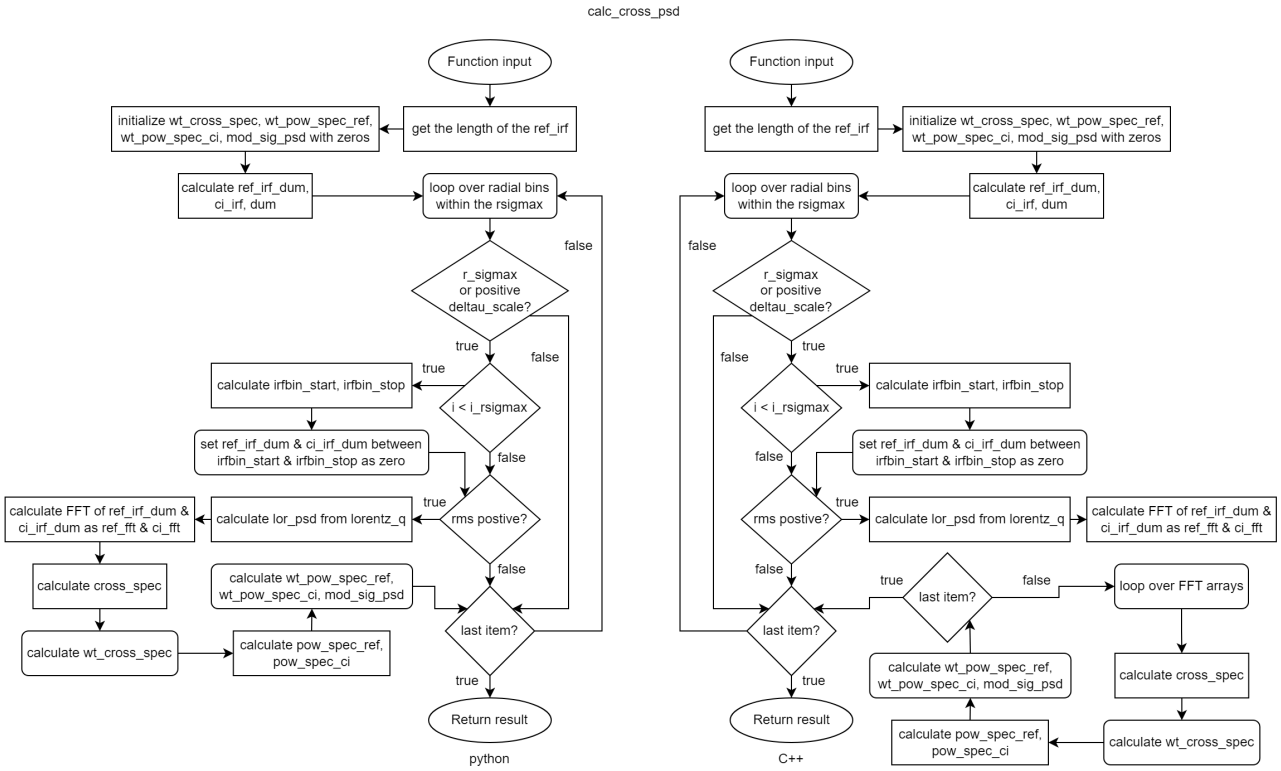


FIG. 3.2 – In this figure, the flow charts for the calc_cross_psd function can be seen for the Python and C++ versions. The only difference between the Python and C++ versions is that due to the use of Numpy arrays in the Python version, the C++ version needs extra loops.

of this dissipation going into heating and seed photons. These calculations are done with radial information and information about the heating and seed photons.
The calc_propagation_params function (see figure 3.7) calculates the propagation delay across a radial bin using the radial bins, corona & disk information.
The calc_radial_time_response function (see figure 3.8) calculates the luminosities for the disk, seed & heating photons with respect to the dissipation and reverberation in the radial bins. It does this using the radial bins, corona information, flows of heating & seed photons, as well as the fractions for the thermal photons, those going to the corona from the disk and those going back from the disk to the corona.
The calc_irfs_mono function (see figure 3.9) calculates the impulse response functions for the photon index, energy-dependent flux, disk & seed photons, as well as the mean photon index. It is calculated from the information about the photon index, energy bins, and the luminosities for the heat, disk & seed photons with respect to the dissipation and reverberation.
The calc_disk_band (see figure 3.10) calculates the fraction of flux emitted for the given energy band, the temperature, the disk & seed luminosities for the given energy band as well as the reverberation luminosities for the given energy band. Calculated

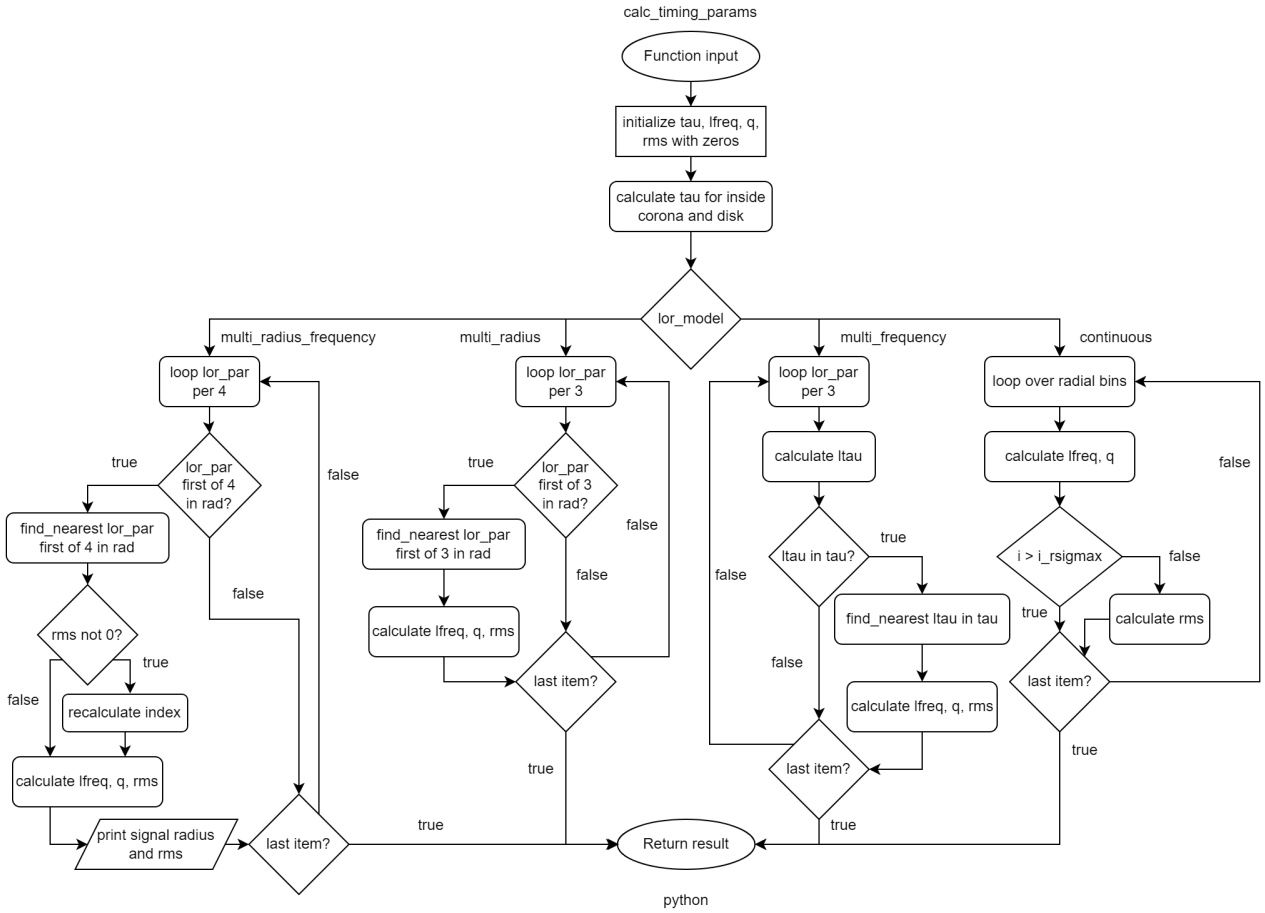


FIG. 3.3 – In this figure, the flow charts for the calc_timing_params function can be seen for the Python version.

using the dissipation, corona to disk & disk to corona, and thermal fractions, seed dissipation & heating photon luminosities, radial bins, information about the corona, and the specified energy band.

The bb_phflux function (see figure 3.11) calculates the black body photon flux for a specified energy band using the temperature and the number of bins to integrate over the energy band.

The linear_rebin_irf (see figure 3.12) rebins the impulse response function and the flux outside the maximum signal radius. Calculated with the bin size, maximum signal radius, the number of the impulse response function bins, the number of bins to spread the impulse response function over, its edges, and the scale of the delay bins.

The lorentz_q function (see figure 3.13 & figure 3.14 for inputting arrays) calculates the Lorentzian power spectrum density given the frequencies, peak frequency, quality

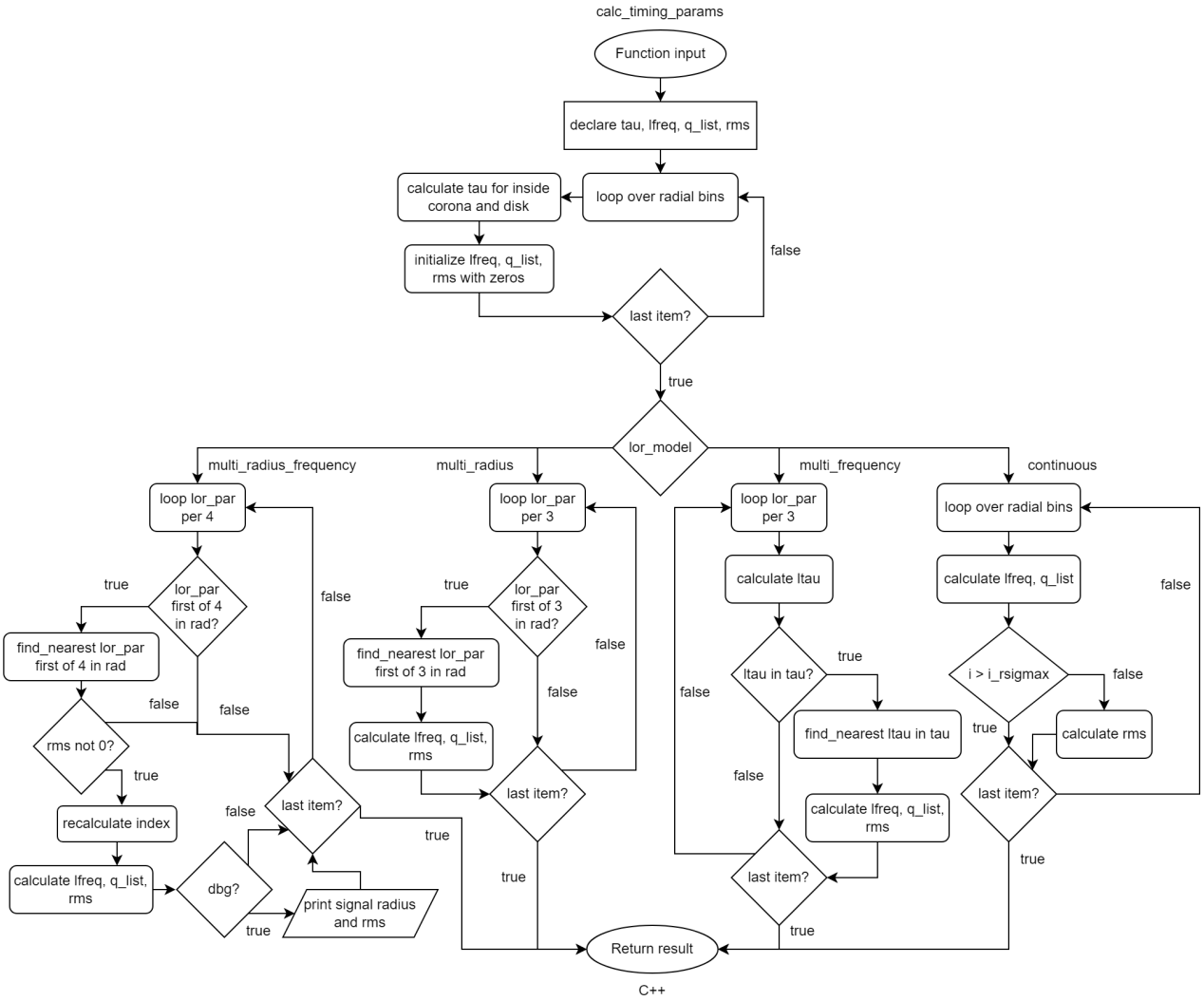


FIG. 3.4 – In this figure, the flow charts for the calc_timing_params function can be seen for the C++ version. In comparison to the Python version, as seen in figure 3.3, the print output is optional behind a dbg boolean and, as with other functions, an extra loop because of no Numpy arrays in C++.

factor, and root-mean-square.

3.1.2 Major overhaul

One of the most significant alterations between the Python and C++ versions was in the calc_illumination_fracs function, and the Geometry functions it used. The flowchart for this function can be seen in figure 3.15. It takes, for the Python version,

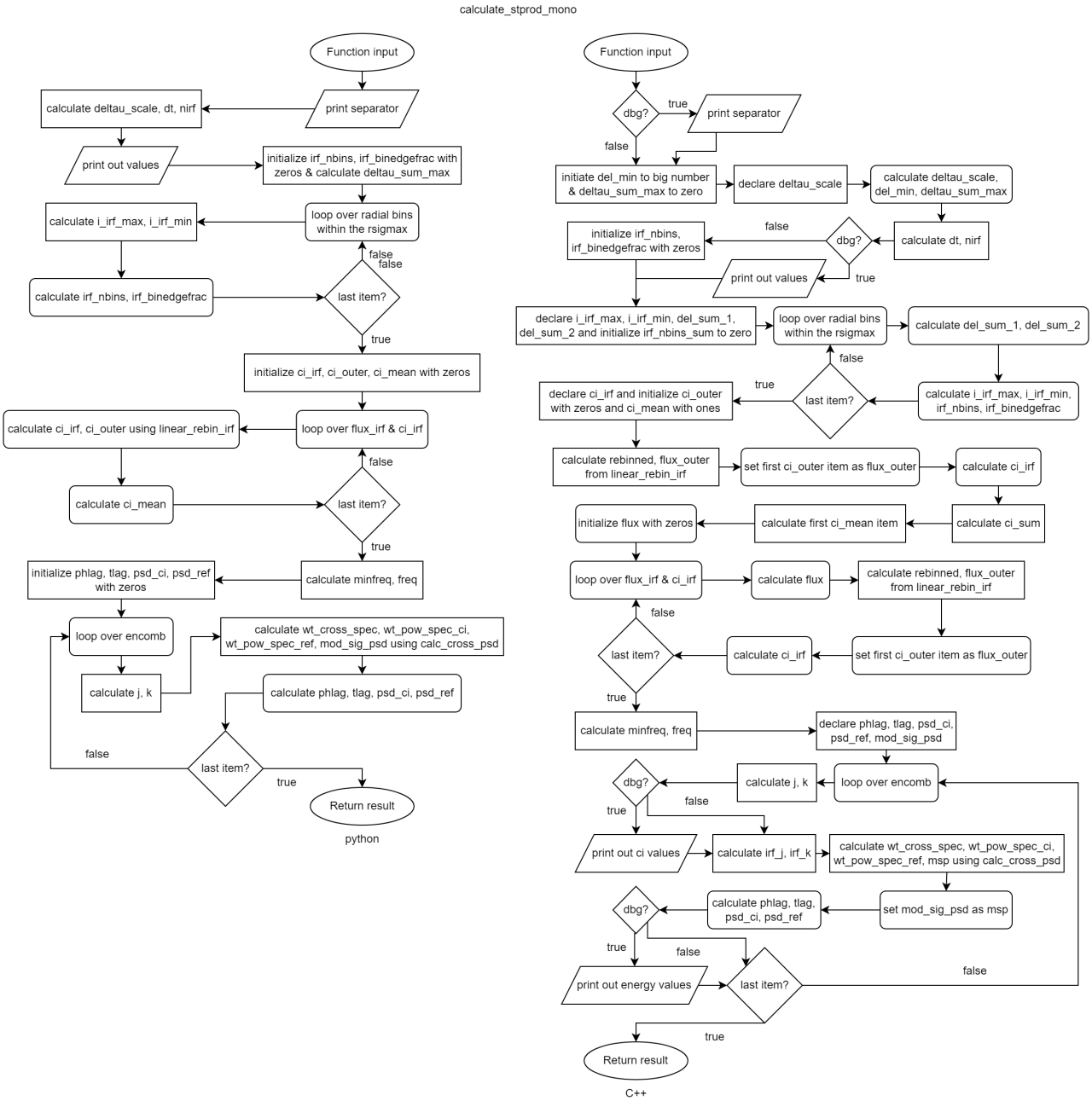


FIG. 3.5 – In this figure, the flow charts for the calculate_stprod_mono function using arrays can be seen for the Python and C++ versions. In this figure, the flow charts for the calc_timing_params function can be seen for the C++ version. In comparison to the Python version, the print output is optional behind a dbg boolean and, as with other functions, an extra loop because there are no Numpy arrays in C++.

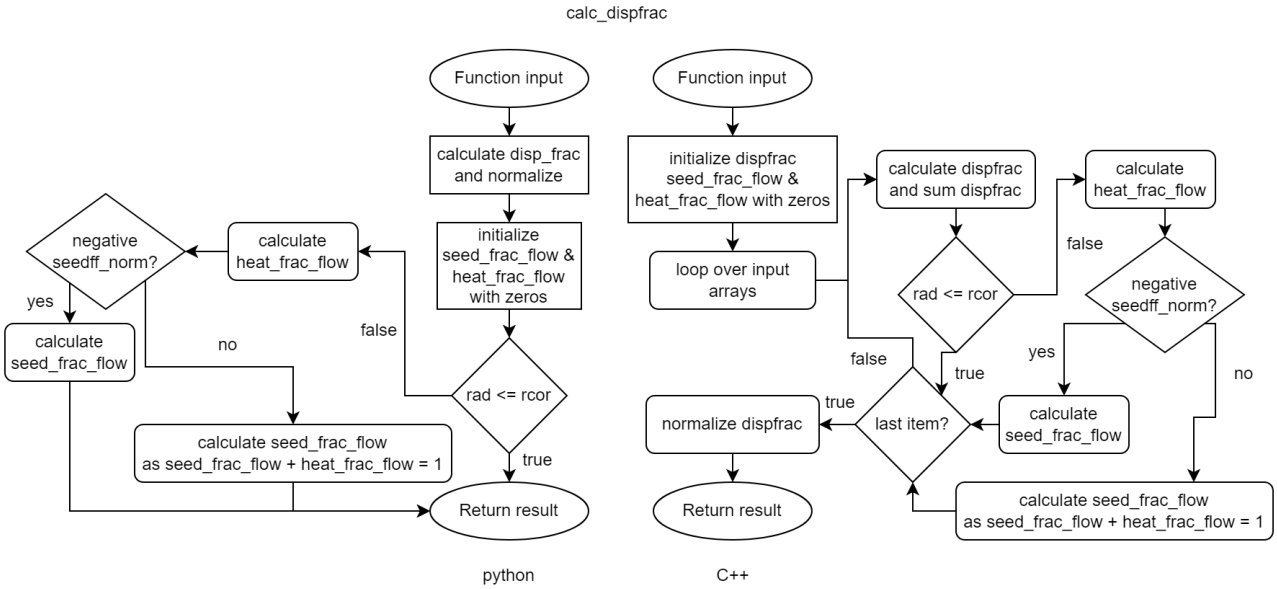


FIG. 3.6 – In this figure, the flow charts for the calc_dispfrac function can be seen for both the Python and C++ versions. Due to the use of NumPy arrays in the Python model, the loops necessary for the calculations are hidden from the structure. Meanwhile, the rest of the function follows a similar structure in both models.

in the radial bin array as well as the one multiplied by the bin width, the name of the geometry function, and the parameters for the geometry. The C++ version takes in a Geometry class, as mentioned below, or any derived classes, as also mentioned below. Both versions then return the fraction of photons going from the corona to the disk, the fraction of the photons going from the disk to the corona, and the solid angle shown by the corona. For this function, it was realized pretty early on that a significant flaw of the Python version is that the Geometry setup and calculation functions were called within each radial bin loop. However, basically, all the geometries that were defined had calculations, some of which were quite a lot of operations that were independent of the radius, which should make the calculations much quicker due to doing fewer operations in each loop. By converting the Geometry functions into classes, even the normalization calculations could be done in one go. As such, the flowchart for the C++ version is drastically different from the Python version.

Due to the way the code is defined in the calc_illumination_fracs, the geometry function can be anything as long as it takes in and returns certain variables. As such, the flow chart of the geometry functions, as can be seen in figure 3.16, can be simplified to a simple straight line. However, due to all the calculations being done for every single radial bin when this function is called, the run time goes exponentially up for more and more radial bins since the function is only calculated for each radial bin.

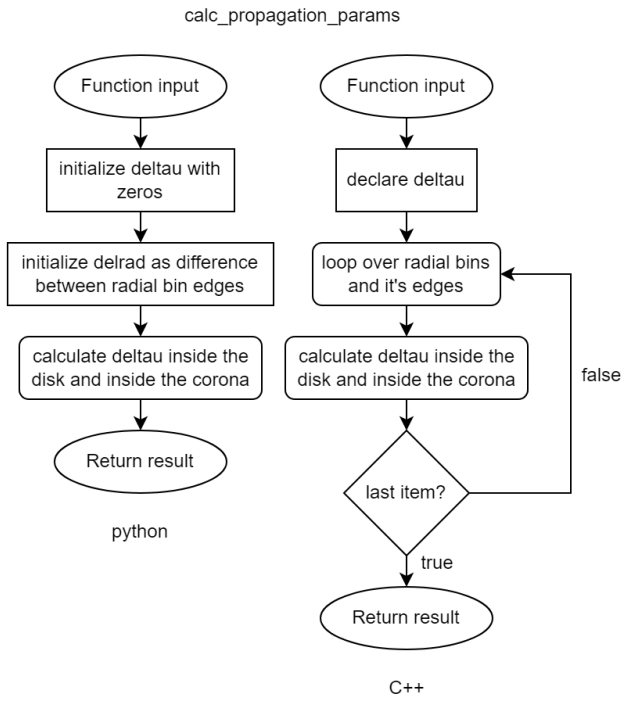


FIG. 3.7 – In this figure, the flow charts for the calc_propagation_params function can be seen for the Python and C++ versions. The only difference between the Python and C++ versions is that due to the use of Numpy arrays in the Python version, the C++ version needs an extra loop.

The function itself also takes in the geometry parameters for this specific geometry.

The change that has been made between the Python version and the C++ version is that all the calculations are now done within a Geometry class, which is read by the calc_illumination_fracs in which the appropriate arrays are returned. The Geometry class' workings are shown in the flowchart in figure 3.17. This Geometry class has a similar input to the Python version of the calc_illumination_fracs function, the radial bin array as well as the one multiplied by the bin width, and the parameters for the specific geometry. The class first initializes the internal arrays; then it calls its setup_geometry method to set up the geometry, after which, whether it is defined or not, it can normalize when necessary. This setup_geometry method sets up the defined geometry first by doing the calculations independent of the radius. Then, it loops over the radial bins and the arrays that are needed for the calculations, in which the radially dependent part of the setup is done. Since geometry is set up as a base class, any geometry can be defined as a derived class of this base Geometry class with its setup_geometry function and normalization when needed. Due to this, many different geometries can be defined, even compound geometries, when needed. As long as the geometry to be defined is defined as a derived class from the base Geometry class, as mentioned before.

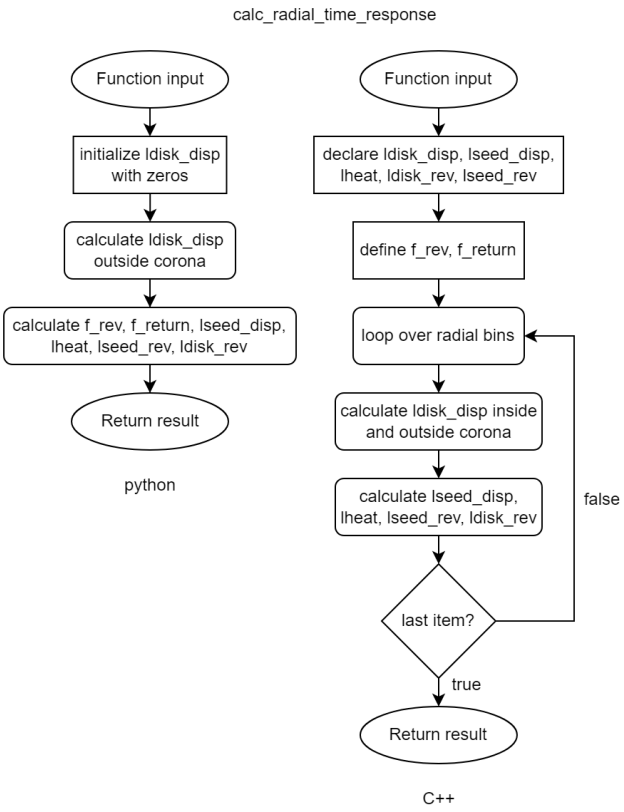


FIG. 3.8 – In this figure, the flow charts for the calc_radial_time_response function can be seen for the Python and C++ versions. The only difference between the Python and C++ versions is that due to the use of Numpy arrays in the Python version, the C++ version needs an extra loop.

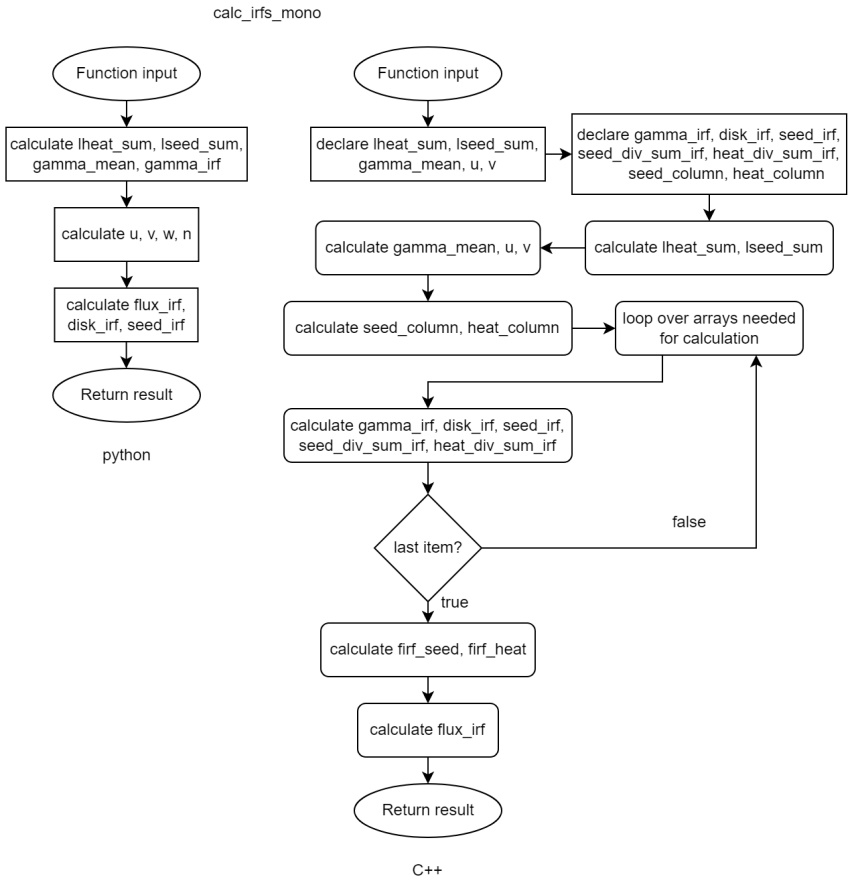


FIG. 3.9 – In this figure, the flow charts for the calc_irfs_mono function can be seen for the Python and C++ versions. The only difference between the Python and C++ versions is that due to the use of Numpy arrays in the Python version, the C++ version needs extra loops.

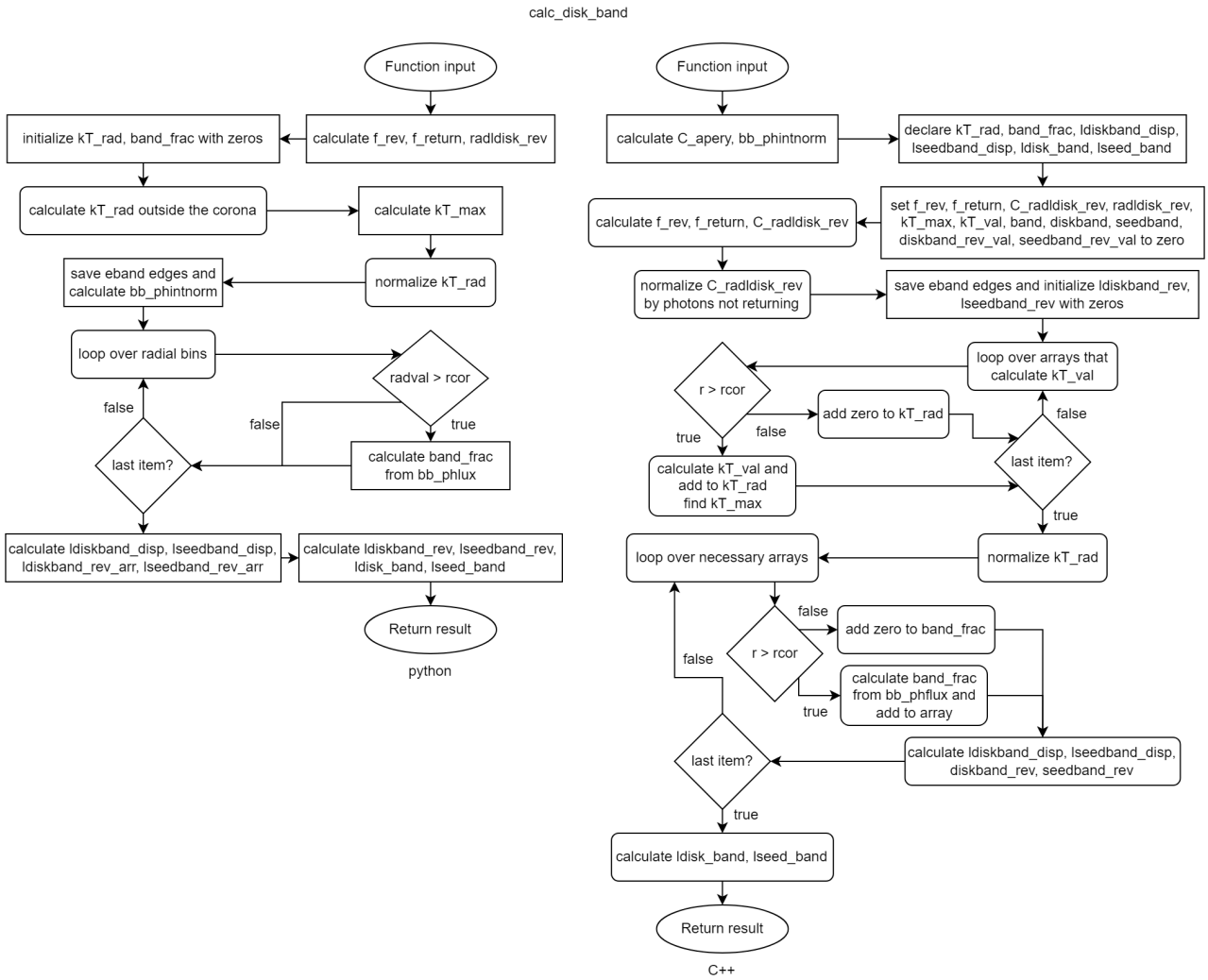


FIG. 3.10 – In this figure, the flow charts for the calc_disk_band function can be seen for the Python and C++ versions. The only difference between the Python and C++ versions is that due to the use of Numpy arrays in the Python version, the C++ version needs extra loops.

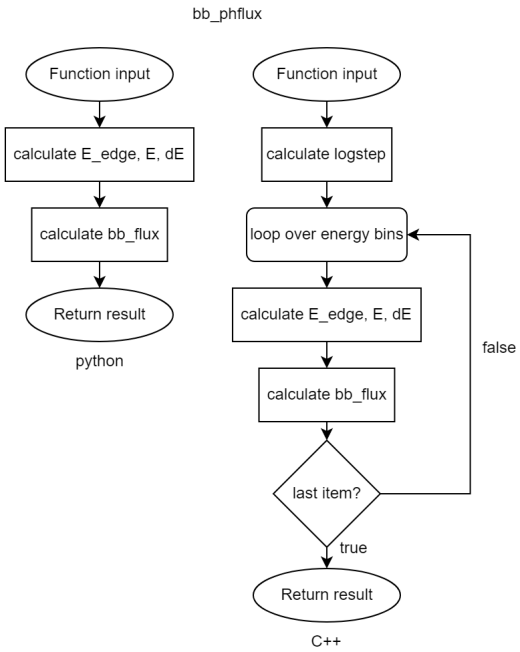


FIG. 3.11 – In this figure, the flow charts for the bb_phflux function can be seen for the Python and C++ versions. The only difference between the Python and C++ versions is that due to the use of Numpy arrays in the Python version, the C++ version needs extra loops.

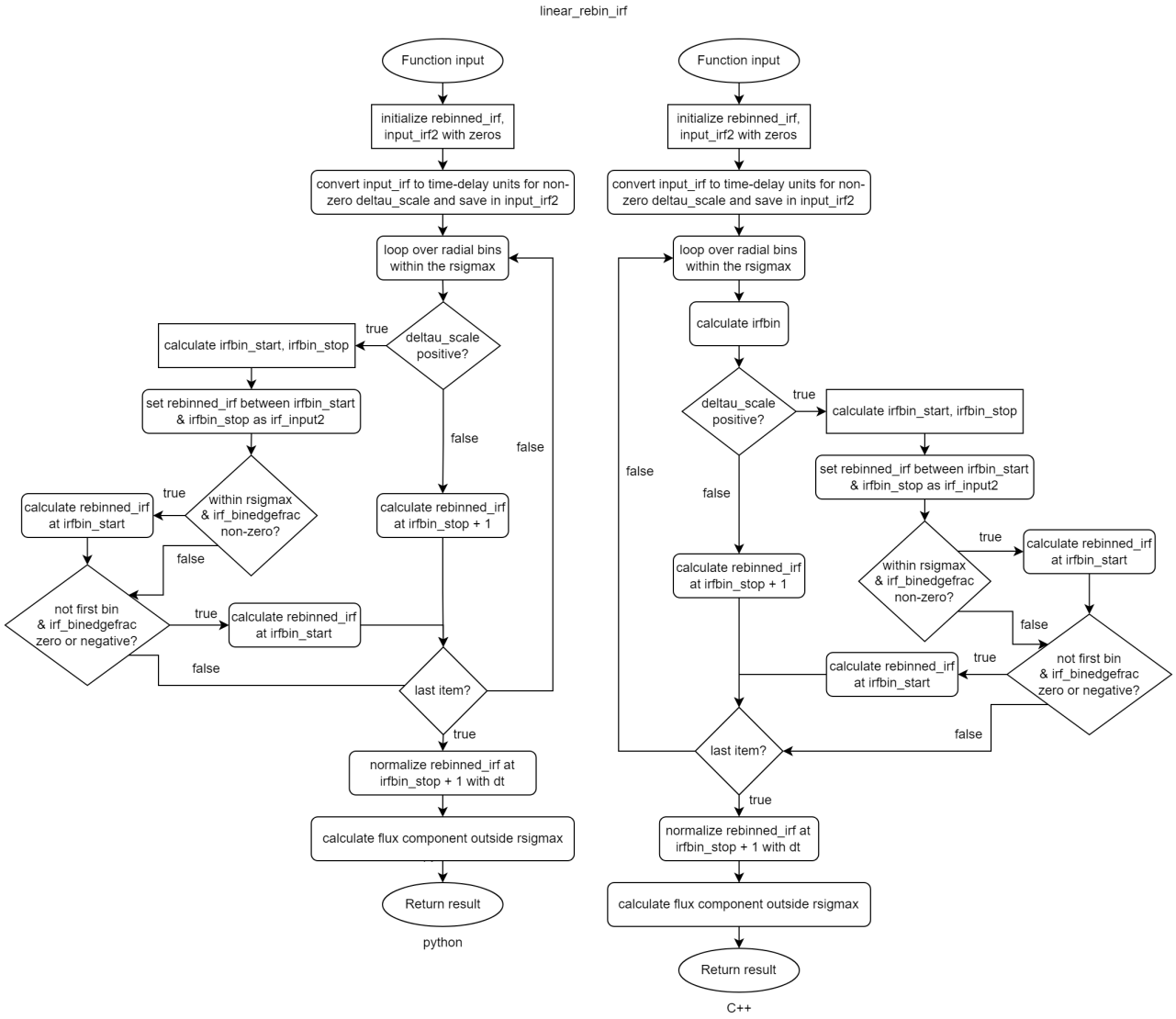


FIG. 3.12 – In this figure, the flow charts for the linear_rebin_irf function can be seen for the Python and C++ versions. There is no difference between these two versions.

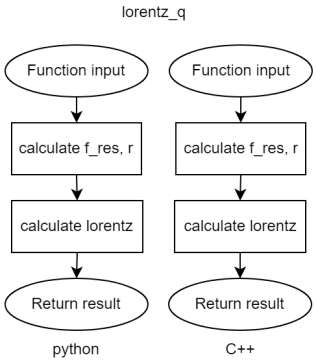


FIG. 3.13 – In this figure, the flow charts for the lorentz_q function using values can be seen for the Python and C++ versions.

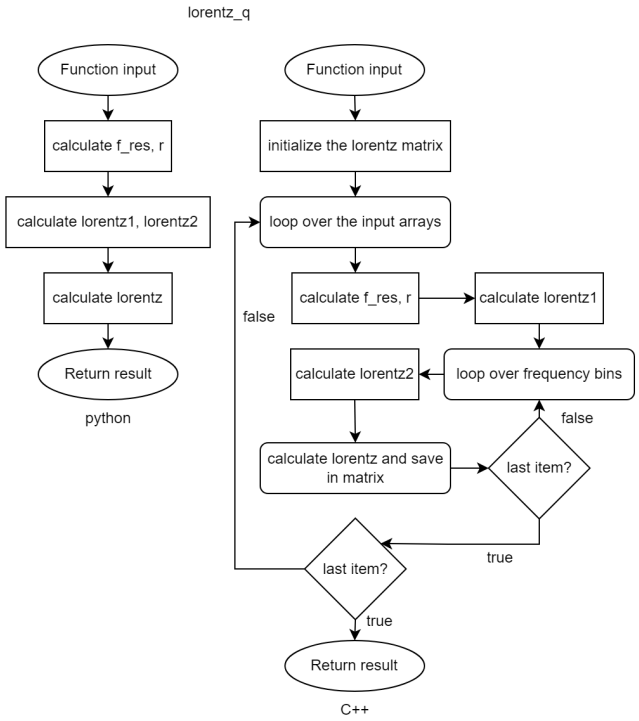


FIG. 3.14 – In this figure, the flow charts for the lorentz_q function using arrays can be seen for the Python and C++ versions.

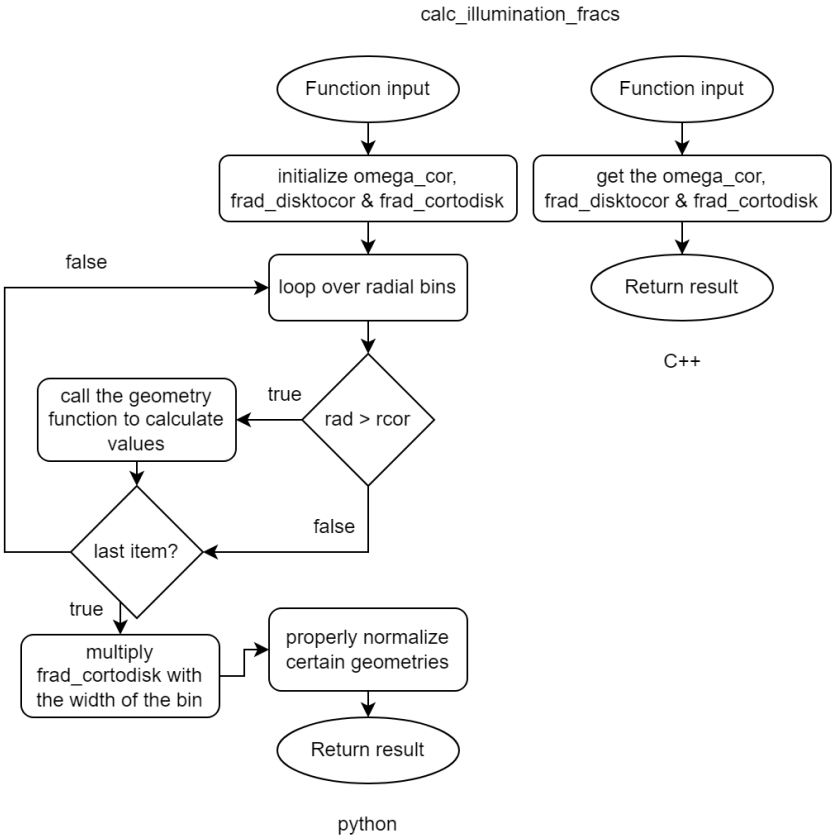


FIG. 3.15 – In this figure, the flow charts for the calc_illumination_fracs function can be seen for both the Python and C++ versions. While the Python version of the code loops over each radial bin and then calls the Geometry function in question to calculate the omega_cor and the fractions, the C++ version does these calculations in a Geometry class. As such, the flow chart is drastically different from that of the Python version.

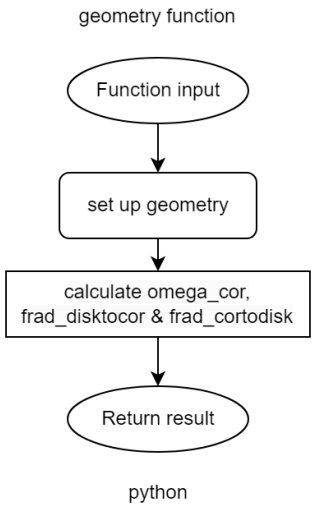


FIG. 3.16 – In this figure, the flow charts for the Python version of the geometry function for the Python version of the calc_illumination_fracs function as is shown in figure 3.15. This function is called for each radial bin in the model, and the function does the calculations for both the radially dependent and radially independent calculations of the geometry setup.

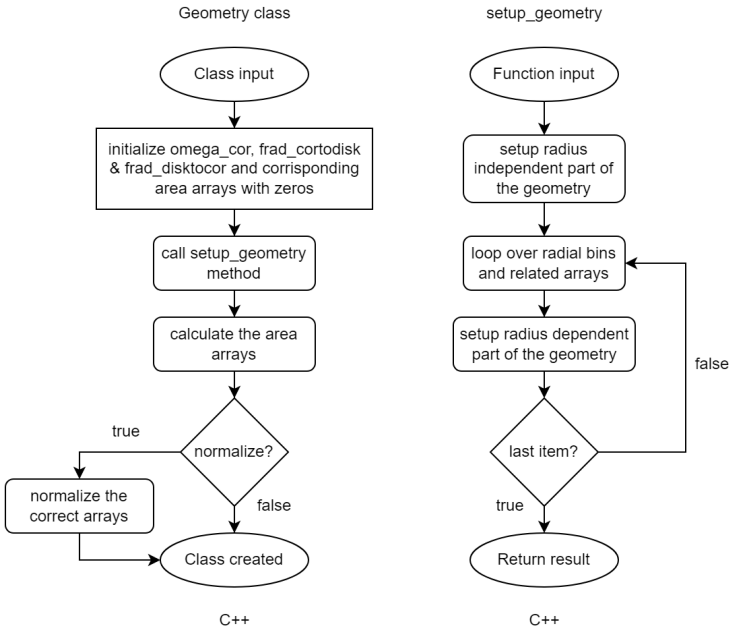


FIG. 3.17 – In this figure, the flow charts for the Python version of the geometry function for the C++ version of the calc_illumination_fracs function as is shown in figure 3.15. In C++, a geometry class (see left side) can be created using all the inputs of the calc_illumination_fracs function. Depending on the defined setup_geometry method, it will set up different kinds of geometries. In this method (see right side), the radially independent part and radially dependent parts have been separated, so the radially independent part only needs to be calculated once.

3.2 Fits

After the model had been fully converted into C++, it was fitted with some actual data to check if the converted model could fit the data correctly as the Python model does. As such, both data from the XMM Newton and NICER telescopes have been provided and fitted to see if the conversion was successful. This fitting is also used to find the run time speed of each function. Both data sets have been fitted for the time lag part of the data, using the LMfit (Newville et al. 2015) Minimizer function using the Least Squares method. For its model function, the following default values were used: $r_{out} = 1000$, $n_{rad} = 100$, an inverted cone geometry, $t_{scale} = 5 \cdot 10^{-5}$, $E_{seed} = 1.0$, energy bands $\{1, 3, 9\}$, $\Gamma_{par} = \{2.33, 0.167\}$, calculating the band fraction, indices of the channel of interest and reference band $\{\{2, 0\}, \{3, 2\}\}$, multiplication factor for the number of impulse response functions 8, minimum fraction for the propagation delay 1, maximum signal radius for the frequency 0.01, frequency range $\in [0.01, 20]$, non-variable black body emission, correcting the power spectral density and using a continuous Lorentzian model. The grid for setting up the Inverted Cone geometry has been set to 1000 bins in both the ϕ and z directions.

In the plots that show the fits (see figures 3.18-3.21), a frequency-time lag plot is shown with a log scaling for both axes. The time lag data between the soft PL energy band and the disk energy band is shown in red, and the fit for these time lags is shown as the solid blue line. The time lag data between the hard PL energy band and the soft PL energy band is shown in black, and the fit for these time lags is shown as the dashed blue line in the figure. The signal frequencies at the corona are shown as the gray dashed-dotted line. On the top right in each figure, a feature of the plot is shown where only the frequency is shown in a log scale to make sure the negative part of the plot can be seen in detail.

The found values for the fits are shown in tables 3.1 & 3.2 for the GX339-4 XMM Newton data fit and the MAXI J1820+70 NICER data fit respectively. The tables show the amount of function evaluations, the chi-squared χ^2 value, as well as the reduced chi-squared χ^2_ν value. It also shows the fitted values of the fits: innermost radius for the accretion flow r_{in}, the coronal radius r_{cor}, the height of the corona h_{cor}, cone angle (assuming inverted cone coronal geometry), the fraction of luminosity which does not turn into thermal energy thermal_frac, the disk energy disk_en, disk variability time-scale normalization disk_tau_norm.

3.2.1 XMM Newton Data Fit

Firstly, data from the XMM Newton telescope was used to be fit using both the Python and C++ models. The data comes from a popular X-ray-emitting object called GX 339-4. It is a moderately strong LMXRB situated in the constellation of Ara (Bradt & McClintock 1983). GX 339-4 was first discovered in 1973 (Markert

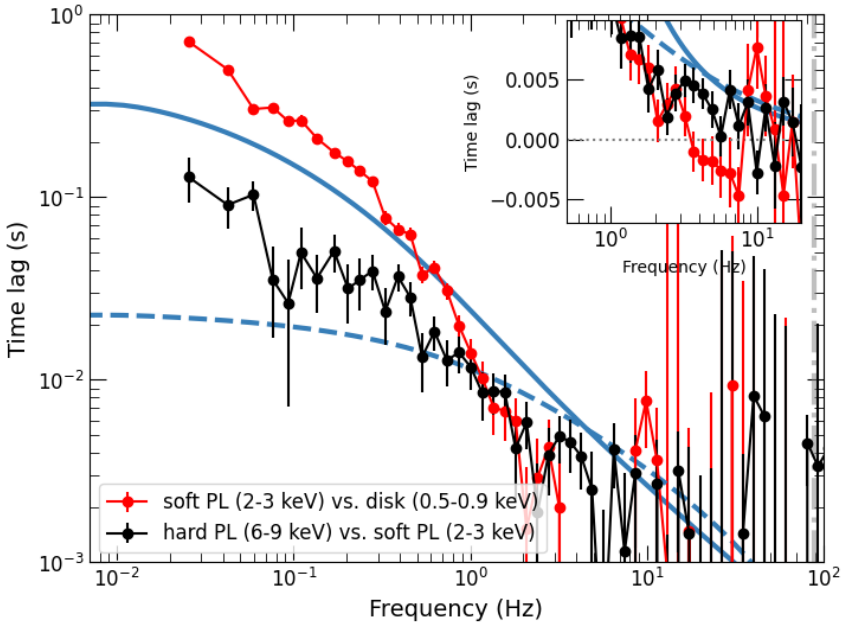


FIG. 3.18 – This figure the resulting fit from the C++ version of the model shown of the XMM-Newton data of GX339-4 in the hard state which was taken with XMM's EPIC-pn instrument (Uttley et al. 2011). It shows the soft power law $(2-3keV)$ vs. the disk power law $(0.5-0.9keV)$ energy bands data in red and the fit for this data as a solid blue line, the hard power law $(6-9keV)$ vs soft power law $(2-3keV)$ energy bands data in black and the fit for this data as a dashed blue line. The signal frequency at the corona is shown as the grey dashed-dotted line.

et al. 1973), and since its discovery, it has been observed to have an outburst every few years (Bhowmick et al. 2021). It has a mass of at least $5.8M_{\odot}$ (Hynes et al. 2003). This data has been taken of GX 339-4 in its hard state using XMM Newton's EPIC-pn instrument (Uttley et al. 2011).

The data used was the time lags for three different geometrically spaced energy bands with a factor spacing factor of ~ 3. A soft power-law energy band is in the range of $2-3keV$, a hard power-law energy band is in the range of $6-9keV$, and finally, a disk energy band is in the range of $0.5-0.9keV$. The resulting fit for this data for both the Python and the C++ versions of the model are shown in figures 3.19 & 3.18 respectively. The data was measured in uninterrupted bands of $\sim 59s$ for each cross-spectrum of the energy band combinations as mentioned above. These cross-spectra were averaged over these segments in geometrically spaced bins that have a geometrical spacing of $\nu \rightarrow 1.15\nu$.

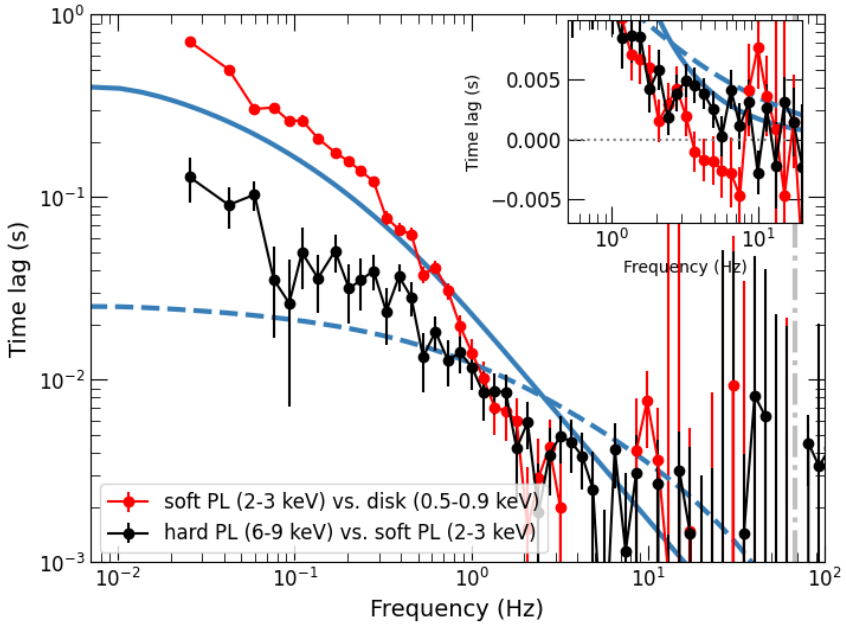


FIG. 3.19 – This figure the resulting fit from the Python version of the model shown of the XMM-Newton data of GX339-4 in the hard state which was taken with XMM's EPIC-pn instrument (Uttley et al. 2011). It shows the soft power law $(2-3keV)$ vs. the disk power law $(0.5-0.9keV)$ energy bands data in red and the fit for this data as a solid blue line, the hard power law $(6-9keV)$ vs soft power law $(2-3keV)$ energy bands data in black and the fit for this data as a dashed blue line. The signal frequency at the corona is shown as the grey dashed-dotted line.

3.2.2 NICER Data Fit

Secondly, data from the NICER telescope was used to be fit using both the Python and C++ models. MAXI J1820+70 is an LMXRB that was first detected in 2018 during its outburst (Guan et al. 2021). These outbursts of MAXI J1820+70 have been observed as highly luminous in both the visual and X-ray bands during its outburst periods (Sai et al. 2021). This XRB was confirmed to be a black hole in Torres et al. (2019), which also calculated some of its parameters. They calculated based on observations and assumptions that the black hole should be roughly $7-8M_{\odot}$. This data has been taken of MAXI J1820+70 in the hard state using the NICER telescope on the ISS (Wang et al. 2021; Bollemeijer et al. Submitted).

The data used was the time lags for three different energy bands that were geometrically spaced. A soft power-law energy band is in the range of $2-3keV$, a hard power-law energy band is in the range of $5-10keV$, and finally, a disk energy band is in the range of $0.5-0.9keV$. The resulting fit for this data for both the Python and the C++ versions of the model are shown in figures 3.21 & 3.20 respectively. The data has been measured over segments of $10s$ with bin sizes of $0.001s$ for the different combinations of cross-spectra as mentioned earlier in section 3.2. These

GX 339-4

	C++	Python
func eval	89	85
χ^2	1077.43	904.6
χ^2_ν	17.10	14.36
r_{in}	$2.45 \pm 7.6 \cdot 10^{-4}$	$2.45 \pm 6.8 \cdot 10^{-4}$
r_{cor}	7.43 ± 114.7	7.44 ± 3.04
h_{cor}	14.11 ± 48.7	13.5 ± 20.7
cone_angle	53.34 ± 142.8	48.8 ± 66.5
thermal_frac	0.37 ± 1.41	0.47 ± 0.22
disk_en	1.50 ± 3.0	1.50 ± 0.42
disk_tau_norm	10.11 ± 66.8	13.7 ± 10.26

TABLE 3.1 – In this table is shown the found values for the fitting of the GX 339-4 data of which the fits were shown in figures 3.18 & 3.19 of the C++ & Python models respectively. The fitting parameters, as well as the number of function evaluations, χ^2 & χ^2_ν are shown in this table. The first column after the parameter names shows the fitting parameter values for the C++ model fit (see figure 3.18), and the last column, the fitting parameter values for the Python model fit (see figure 3.19).

cross-spectra were averaged over these segments in geometrically spaced bins that have a geometrical spacing of $\nu \rightarrow 3.5\nu$.

MAXI J1820+70

	C++	Python
func eval	94	208
χ^2	221.8	221.9
χ^2_ν	2.9	2.9
r_{in}	2.6 ± 5064.6	2.23 ± 4169.6
r_{cor}	27.43 ± 53198.06	26.0 ± 48557.9
h_{cor}	5.22 ± 10115.44	5.06 ± 9459.0
cone_angle	56.6 ± 63.22	60.0 ± 41.5
thermal_frac	1.05 ± 1.9	0.9 ± 1.49
disk_en	1.9 ± 0.39	1.8 ± 0.38
disk_tau_norm	46.0 ± 133683.0	42.9 ± 120253.26

TABLE 3.2 – In this table is shown the found values for the fitting of the MAXI J1820+70 data of which the fits were shown in figures 3.20 & 3.21 of the C++ & Python models respectively. The fitting parameters, as well as the number of function evaluations, χ^2 & χ^2_ν are shown in this table. The first column after the parameter names shows the fitting parameter values for the C++ model fit (see figure 3.20), and the last column, the fitting parameter values for the Python model fit (see figure 3.21).

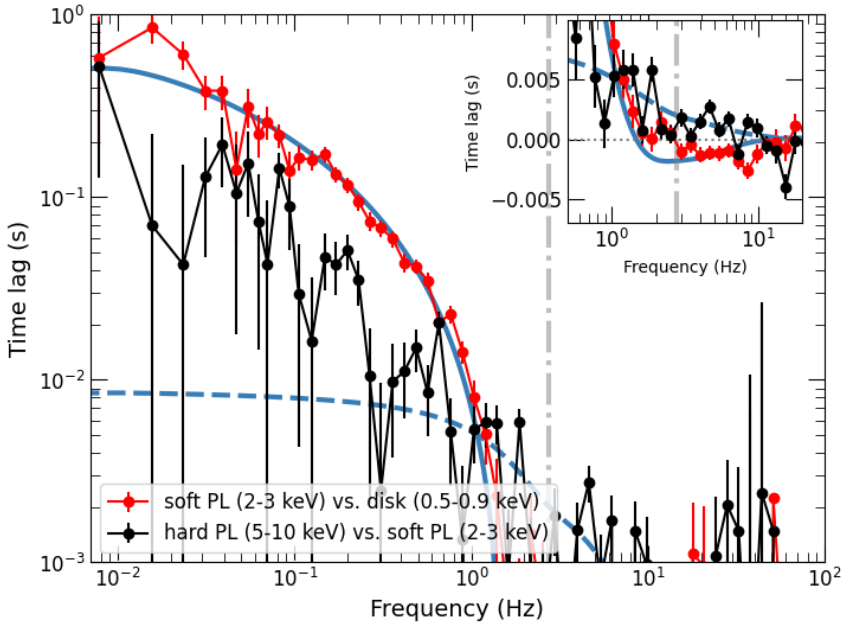


FIG. 3.20 – This figure the resulting fit from the C++ version of the model shown of the NICER data of MAXI J1820+70 in the hard state (Wang et al. 2021; Bollemeijer et al. Submitted). It shows the soft power law $(2-3keV)$ vs. the disk power law $(0.5-0.9keV)$ energy bands data in red and the fit for this data as a solid blue line, the hard power law $(5-10keV)$ vs soft power law $(2-3keV)$ energy bands data in black and the fit for this data as a dashed blue line. The signal frequency at the corona is shown as the grey dashed-dotted line.

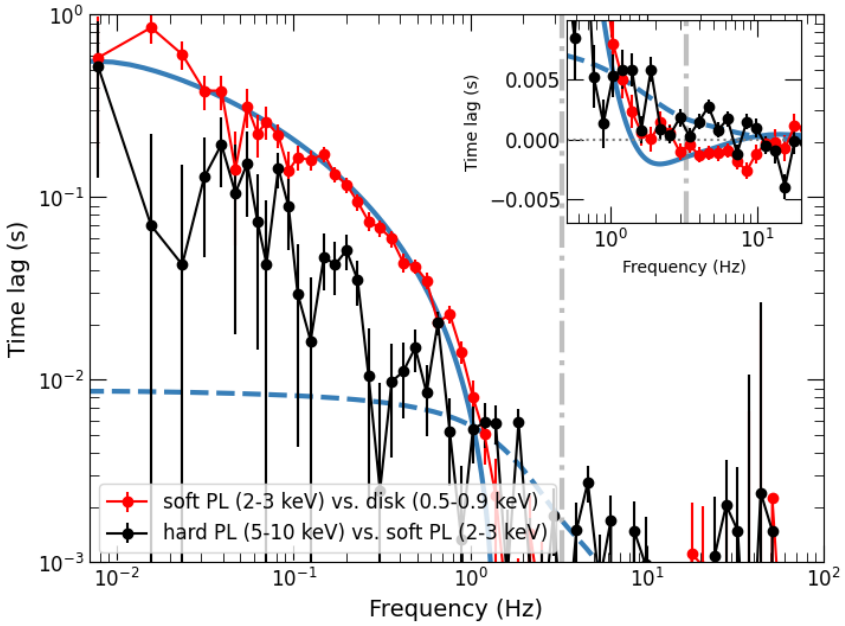


FIG. 3.21 – This figure the resulting fit from the Python version of the model shown of the NICER data of MAXI J1820+70 in the hard state (Wang et al. 2021; Bollemeijer et al. Submitted). It shows the soft power law $(2-3keV)$ vs. the disk power law $(0.5-0.9keV)$ energy bands data in red and the fit for this data as a solid blue line, the hard power law $(5-10keV)$ vs. soft power law $(2-3keV)$ energy bands data in black and the fit for this data as a dashed blue line. The signal frequency at the corona is shown as the grey dashed-dotted line.

3.3 Run time

As mentioned before, during the fitting process of the XMM & NICER data, the run time of the used functions has been measured. The found run time values are shown in table 3.3. The run-time values of both the Python and C++ versions are measured. Then, the speedup was calculated for the C++ version and compared to the Python version. A negative speedup means the Python version was faster than the C++ version, and a positive speedup means the C++ version was faster than the Python version. For the calc_illumination_frac, an inverted cone Inv_Cone geometry, three different grid sizes were used to set up the geometry. The run time of this function using the different grid sizes is shown at the bottom part of the table.

	Python (s)	C++ (s)	speedup (%)
bb_phflux	$1.10 \cdot 10^{-4}$	$1.43 \cdot 10^{-4}$	−30.15%
calc_cross_psd	$9.60 \cdot 10^{-1}$	3.58	−240.18%
calc_disk_band	$1.81 \cdot 10^{-2}$	$2.26 \cdot 10^{-2}$	−25.10%
calc_dispfrac	$9.18 \cdot 10^{-5}$	$7.86 \cdot 10^{-6}$	90.30%
calc_irfs_mono	$7.66 \cdot 10^{-5}$	$2.40 \cdot 10^{-5}$	68.59%
calc_propagation_params	$1.08 \cdot 10^{-4}$	$7.28 \cdot 10^{-6}$	93.25%
calc_radial_time_response	$6.31 \cdot 10^{-5}$	$7.87 \cdot 10^{-6}$	87.53%
calc_timing_params	$5.38 \cdot 10^{-4}$	$2.37 \cdot 10^{-5}$	95.59%
calculate_stprod_mono	1.94	7.18	−231.47%
find_nearest	$1.68 \cdot 10^{-5}$	$1.16 \cdot 10^{-7}$	98.38%
linear_rebin_irf	$1.05 \cdot 10^{-3}$	$3.84 \cdot 10^{-4}$	71.29%
lorentz_q	$5.59 \cdot 10^{-4}$	$4.34 \cdot 10^{-4}$	43.19%
calc_illumination_fracs			
Inv_Cone (1000 x 1000 grid)	$1.23 \cdot 10^{1}$	3.47	71.62%
Inv_Cone (100 x 100 grid)	$6.96 \cdot 10^{-2}$	$2.73 \cdot 10^{-2}$	60.78%
Inv_Cone (10 x 10 grid)	$2.41 \cdot 10^{-2}$	$1.35 \cdot 10^{-2}$	43.98%

TABLE 3.3 – In this table, the run time for each function that is used during the fitting process is shown. The values are averaged over the fitting process of both data sets to give a good idea of the average run time of each function in both versions of the model. In the first column, the name of the function in question is shown; in the second column, the run time values are shown for the Python version in seconds; in the third column, the run time values are shown for the C++ version in seconds, in the last column the speedup of each function is shown as a percentage speedup compared to the Python version. For the calc_illumination_fracs function, the Inv_Cone (inverted cone) geometry has been used and timed for three different sizes of grids.

Chapter 4

Findings

4.1 Data fits

From the examination of the fits and their parameters in sections 3.2.1 3.2.2, a few conclusions can be drawn. First let's look at the fits of the GX339-4 data (figures 3.18 & 3.19, table 3.1). The model fitting shows that the C++ version and Python versions of the model follow the data for both energy bands. However, the intersection point where the soft PL vs disk band lags less than the hard vs soft PL is at a different frequency. Also, the signal frequency at the coronal radius is different. Looking at the actual fitting parameters shows a similar story. The C++ version has a higher chi-squared value than the Python version, thus showing that the Python version has found a better fit. All the values that the C++ version has found during fitting fall within the error bars of the Python version, which would suggest that the C++ version is a fairly good conversion of the Python version of the model.

Next looking at the MAXI J1820+70 data (figures 3.20 & 3.21, table 3.2). Compared to the GX339-4 data, the MAXI J1820+70 data fits are much better in both versions, showing only a slight difference in the frequency of the intersection point and signal frequency at the coronal radius. Looking at the fitting parameters shows this even more, as the values are all very close to each other. However, here, a problem arises where the error bars on the fitting parameters are extremely high for most parameters. Unfortunately, this is the case for both the C++ and Python versions, which would imply that it is not the problem with the C++ version specifically but rather the model itself or the way the model is fitted. The Python version also has lower error bars for all parameter values for the MAXI J1820+70 fit, suggesting it is the best fit for both versions. Since the C++ version values all still lie within the error bars of the Python, it could be suggested that the C++ version is a fairly good conversion of the Python version of the model.

4.2 Run time

Looking at the run time results from section 3.3 (see table 3.3) and the improvements and changes explained in section 3.1, conclusions could be drawn to the success of these alterations based on the found run time improvements or lack of improvements. However, since the run time of these functions ranges anywhere from hundreds of nanoseconds to several seconds, some speedup or lack of speedup of functions matters more than others.
The first notable result is that while most functions show a slight increase in run time, a few are slowing down the run time more for the C++ version when compared to the Python version. The first of these functions is the bb_phflux function (see figure 3.11), where the C++ version is almost a third slower than the Python version. However, since this is on the order of hundreds of microseconds, it does not matter too much for the run time. Since the function uses Numpy functions to do some operations like summing and others, it is believed that the reason for the slower run time in C++ might be that they are not used in the C++ version.
A function that is slightly slower than the bb_phflux function is the calc_disk_band (see figure 3.10). For which the Python version is only 25% faster than the C++ version. However, since the run time of this function is closer to tens of milliseconds, the influence on the total run time of the model is bigger. A reason behind this could be because it calls the bb_phflux function in a loop to calculate the disk-band-related parameters.
What is an even bigger slow-down is the fact that both the calc_cross_psd (see figure 3.2) & the calculate_stprod_mono (see figure 3.5) functions are both two times slower for the C++ version when compared to the Python versions. Especially since both of these functions have a run time in the order of seconds, which means that every time the C++ version is called instead of the Python version, the model function call will take several seconds longer. In the case of the calc_cross_psd, the problem could be at several different points in the function. It could be the fact that Numpy arrays in Python are just much faster than what is used in C++. However, that would mean that the other functions would need to be slower as well, which is not the case. Another possible point of slow-down is that the order of code structure is different or the fact that the C++ version cannot use the Numpy functions or the SciPy Fast-Fourier-Transform. More testing is needed to find the exact reason for the slow-down. In the case of calculate_stprod_mono, the same problem points could be used as a reason why it is slower than the Python version. An extra point of concern is the fact that the already slowed down calc_cross_psd function is called in a loop in the calculate_stprod_mono function, making the slow-down compound even more in this function as each time the calc_cross_psd function is called, an extra few seconds is needed to run. Looking at the speedup percentage gives us insight into this relation. Since the speedup, or somewhat slow-down, is of larger magnitude for the calc_cross_psd function, it allows for the possibility that the sole reason the calculate_stprod_mono function is slowed down when compared to the Python version

is that it calls the calc_cross_psd function. However, for the exact reason behind this slow-down, more testing would have needed to be done.

The rest of the functions all have a C++ version run time, which is shorter than the Python version, ranging anywhere from a 40% speedup to almost 100% speedup. The function with the slowest speedup is the lorentz_q function (see figure 3.14) with a speedup of only around 40%. However, since its run time is on the order of hundreds of microseconds, this is not that much of a speedup and thus does not matter that much to the overall speedup. The slight speedup is probably caused by doing all the calculations inside of one loop instead of the Numpy array operations of the Python version.
Two functions where the speedup was slightly more in the range of around 70% speedup are the calc_irfs_mono function (see figure 3.9) & linear_rebin_irf (see 3.12) functions. For both functions, there is no clear reason why they are faster for the C++ version than the Python version. One possible reason is the way the array splicing in Python is done as a for loop in C++, but to confirm that, it would need to be tested. However, since the run time is just a few tens of microseconds for the calc_irfs_mono function and a few hundreds of microseconds to milliseconds for the linear_rebin_irf function, the speedup does not matter that much.
The calc_dispfrac (see figure 3.6), calc_propagation_params (see figure 3.7) & calc_radial_time_response (see figure 3.8) functions all have a speedup of around 90% for the C++ version when compared to the Python version. While this is an amazing improvement due to the functions taking anywhere from microseconds to hundreds of microseconds, this does not matter too much for the full run time of the model. One possible explanation of the speedup in these functions is the fact that while the Python version relies on the Numpy array operations, in the C++ version, fewer loops have to be done since it uses one loop for most of the calculations.
The functions that show a nearly 100% speedup for the C++ version when compared to the Python version are the calc_timing_params (see figures 3.3 & 3.4) & the find_nearest (see figure 3.1) functions. These functions have run times on the order of hundreds of nanoseconds to hundreds of microseconds, and thus, while the speedup is a lot, they do not matter much to the total run time. For the calc_timing_params, a lot of the speedup could be because of the way, in the Python version, many zeros are added to the return arrays via if/else statements even though they are already initialized as such and thus unnecessary. These extra operations were eliminated in the C++ version since it is not necessary to set values to zero if they are already zero. For the find_nearest function, the Python version uses the Numpy argmin function on the input Numpy array minus the value that needs to be found. The C++ version does the same; however, since it does the calculations in a slightly different order than the Numpy array operations, it is probably quicker.

Finally, the calc_illumination_fracs (see figure 3.15) has a different speed up depending on the geometry (see figures 3.16 & 3.17) type and the size of the grid to set up the geometry for some geometries. While in the code, more Geometries were defined for

this thesis, only the Inv_Cone geometry was used to fit the data. However, since this geometry needs a grid size to set up itself, the run-time calculations have been done for different grid sizes and, as seen in table 3.3, have different amounts of speedup. In the case of a 10 x 10 grid, the speedup is only around 40% with the total run time of the function on the order of tens of milliseconds. Similarly, in the case of a 100 x 100 grid, the total run time of the function is also on the order of tens of milliseconds. The speedup of this grid size is bigger than the smaller grid size at a speedup of around 60%. For the case of an even bigger grid of size, 1000 x 1000, the speedup is only slightly more at 70%; however, since the run time is actually on the order of several seconds, this is a much bigger increase than the smaller grid sizes.
The big reason the speedup happens is that in the Python version, the setup of the geometry has to be redone for every radial bin. With the significant change made in the C++ version, only the radially dependent parts of the calculation need to be redone for each radial bin. It shows up more for large grid sizes since the total setup time is bigger for larger grid sizes. For smaller grid sizes, the already quick calculations of the Numpy functions show up as a smaller speedup.

4.3 Choices & Improvements

During this project, several choices and compromises had to be made to make it work. As mentioned before, the initial idea was to fit the converted model using XSPEC; however, this was not possible due to timing constraints. Because of that, a different method of testing the converted model was needed. Luckily, a fitting Jupyter notebook was available. However, for this, the C++ functions need to be compiled in a way that they can be used in Python and, thus, a Jupyter Notebook. It was chosen for this project to use pybind11 since this is an easy way to compile C++ functions into a usable Python module. For convenience's sake, the function binds in pybind11 and is also defined in a way that makes the compiled functions compatible with the use of Numpy arrays. To make sure the Matrix class used in the C++ version was able to work with this, the Matrix class needed to be converted so it could be interpreted and used as a proper Numpy array. This conversion, unfortunately, took a lot more time than expected. One could argue that the functions should have been defined with Numpy arrays in mind, but as mentioned earlier, the initial goal was fitting using XSPEC, and as such, the functions had been defined with C++ templates, which would allow for multiple array classifications to work with the code as long as they have the appropriate methods and alike. While this made the function definitions more convoluted and the creation of it took more time, it ended up being well-invested time for when it had to be used in the pybind11 function binds.
When it finally came to timing the functions and fitting the data, there were still some struggles to overcome with regard to actually timing the functions. Initially, the idea was to run the functions with a set of fixed parameters several hundred times and average those results to time the functions. This timing could then be

done natively in C++ and Python. However, it came to the attention that it could have been done easier while fitting and saving the results. Unfortunately, it was not as easy as just putting the Python time module before & after each function call in the lag model function in the pipeline for both the Python version functions as well as the C++ version functions and compare. However, this created the problem that some of the functions that were defined were only called inside other functions and thus had to be timed inside the C++ version itself. Next, the idea was to do this and then dynamically save those found timing values to a JSON object in C++, which could then be read out from Python and compared to the found values in Python. However, using such a method caused an unforeseen problem: the run time of the functions increased as they were timed. Luckily, this was solved by just redefining the keys of the JSON object. A different problem arose during the fitting of the MAXI J1820+70 data where because of some problems in the data, the fitting did not work; this was fixed by using the fill_value= "extrapolate" setting in the interp1d function from the SciPy interpolation module.
As mentioned in the 3.2 & 3.3 section, the use of the Inverted Cone geometry was mentioned. While more geometries were defined along the other functions, the choice was made for the Inverted Cone because that is currently believed to be the best way to approximate the corona of the black hole. The choice to still convert all the other geometries was to make sure that the code could still work with different geometries if one wanted to use them for fitting. So if, in the future, better approximations were made from new geometries or combinations of geometries, those could still work with the code.

While the project has shown some excellent insights into possible improvements and what is less effective, some improvements can be introduced to the fitting pipeline as well as the model code. For starters, the functions that have slowed down the run time in the C++ version could be improved. Either that or instead of the C++ version, the Python version should be used. Another improvement could be to actually look into fitting the data using the C++ functions as an XSPEC model like the initial goal was. Another way to improve the model is to redo the whole lag model from the ground up to make sure that no hard-to-spot slow-downs are taken from the original model.

4.4 Conclusion

In conclusion, the results of the project are mixed. While some functions of the C++ version show an apparent speedup, it still fits the data as well as the Python version. Other functions slowed down the run time. Some functions have yet to even speed up in terms of run time, much enough to make a big difference. The most significant speedup of the model is the way the calc_illumination_fracs function was improved with a few seconds of speedup. Seeing that the speedup of the functions

was less than expected, it could be a reason to invest less time into improving the C++ version. At the same time, the arguments for converting the model to C++ still stand. Converting to C++ makes the model run faster, if slower than was expected. It also can still be implemented more efficiently as a C++ XSPEC model, which puts the model in a standardized format that allows for more accessible community-driven improvements. Of course, this would need more time and effort to figure out and implement. Even then, it is better to remake the model from the ground up to make it quicker. Even if this is not done, those changes that were made in the conversion process could also be applied to the Python version, making it quicker. Now, the dilemma becomes whether to put more effort into making it work into XSPEC and properly redo the model in C++ or to use the mentioned improvements in this thesis and apply them to the Python version. While the latter would definitely improve the run time of the model, it may be less than one might have hoped. However, making the model work in XSPEC also has both benefits and downsides. However, the most significant benefit is its potential for better community-driven improvements, which are vital in science.

Bibliography

Abbott, R., Abbott, T. D., Abraham, S., et al. 2020, Phys. Rev. Lett., 125, 101102

Altamirano, D. & Méndez, M. 2015, MNRAS, 449, 4027

Arévalo, P. & Uttley, P. 2006, MNRAS, 367, 801

Arnaud, K. A. 1996, in Astronomical Society of the Pacific Conference Series, Vol. 101, Astronomical Data Analysis Software and Systems V, ed. G. H. Jacoby & J. Barnes, 17

Balbus, S. A. & Hawley, J. F. 1991, ApJ, 376, 214

Balbus, S. A. & Hawley, J. F. 1992, ApJ, 400, 610

Balbus, S. A. & Hawley, J. F. 1998, Reviews of Modern Physics, 70, 1

Bambi, C. 2021, arXiv e-prints, arXiv:2106.04084

Begelman, M. C., Armitage, P. J., & Reynolds, C. S. 2015, ApJ, 809, 118

Belloni, T. M. & Motta, S. E. 2016, in Astrophysics and Space Science Library, Vol. 440, Astrophysics of Black Holes: From Fundamental Aspects to Latest Developments, ed. C. Bambi, 61

Beloborodov, A. M. 2001, Advances in Space Research, 28, 411

Bhowmick, R., Debnath, D., Chatterjee, K., et al. 2021, ApJ, 910, 138

Bollemeijer, N., Name1, & Name 2. Submitted

Bollimpalli, D. A., Mahmoud, R., Done, C., et al. 2020, MNRAS, 496, 3808

Bolton, C. T. 1972, Nature, 235, 271

Boss, A. P. & Keiser, S. A. 2014, ApJ, 794, 44

Bradt, H. V. D. & McClintock, J. E. 1983, ARA&A, 21, 13

Braes, L. L. E. & Miley, G. K. 1971, Nature, 232, 246

Bright, J. S., Fender, R. P., Motta, S. E., et al. 2020, Nature Astronomy, 4, 697

Celotti, A., Miller, J. C., & Sciama, D. W. 1999, Classical and Quantum Gravity, 16, A3

Churazov, E., Gilfanov, M., & Revnivtsev, M. 2001, MNRAS, 321, 759

Colbert, E. J. M. & Mushotzky, R. F. 1999, ApJ, 519, 89

De Marco, B., Ponti, G., Muñoz-Darias, T., & Nandra, K. 2015, ApJ, 814, 50

De Marco, B., Ponti, G., Petrucci, P. O., et al. 2017, MNRAS, 471, 1475

De Marco, B., Zdziarski, A. A., Ponti, G., et al. 2021, A&A, 654, A14

Dhawan, V., Mirabel, I. F., & Rodríguez, L. F. 2000, ApJ, 543, 373

Done, C., Gierliński, M., & Kubota, A. 2007, A&A Rev., 15, 1

Dubus, G., Kim, R. S. J., Menou, K., Szkody, P., & Bowen, D. V. 2001, ApJ, 553, 307

Fender, R. P., Belloni, T. M., & Gallo, E. 2004, MNRAS, 355, 1105

Fender, R. P., Homan, J., & Belloni, T. M. 2009, MNRAS, 396, 1370

Finkelstein, D. 1958, Physical Review, 110, 965

Frank, J., King, A., & Raine, D. J. 2002, Accretion Power in Astrophysics: Third Edition

García, J. A., Fabian, A. C., Kallman, T. R., et al. 2016, MNRAS, 462, 751

Gendreau, K. C., Arzoumanian, Z., & Okajima, T. 2012, in Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, Vol. 8443, Space Telescopes and Instrumentation 2012: Ultraviolet to Gamma Ray, ed. T. Takahashi, S. S. Murray, & J.-W. A. den Herder, 844313

Giacconi, R., Gursky, H., Paolini, F. R., & Rossi, B. B. 1962, Phys. Rev. Lett., 9, 439

Giannios, D., Kylafis, N. D., & Psaltis, D. 2004, A&A, 425, 163

Gierliński, M., Zdziarski, A. A., Poutanen, J., et al. 1999, MNRAS, 309, 496

Grinberg, V., Pottschmidt, K., Böck, M., et al. 2014, A&A, 565, A1

Guan, J., Tao, L., Qu, J. L., et al. 2021, MNRAS, 504, 2168

Harris, C. R., Millman, K. J., van der Walt, S. J., et al. 2020, Nature, 585, 357

Hawley, J. F. & Balbus, S. A. 1991, ApJ, 376, 223

Hawley, J. F. & Balbus, S. A. 1992, in American Astronomical Society Meeting Abstracts, Vol. 181, American Astronomical Society Meeting Abstracts, 71.05

Heil, L. M., Uttley, P., & Klein-Wolt, M. 2015, MNRAS, 448, 3339

Hewish, A., Bell, S. J., Pilkington, J. D. H., Scott, P. F., & Collins, R. A. 1968, Nature, 217, 709

Höfner, S., Gautschy-Loidl, R., Aringer, B., & Jørgensen, U. G. 2003, A&A, 399, 589

Hogg, J. D. & Reynolds, C. S. 2016, ApJ, 826, 40

Homan, J., Bright, J., Motta, S. E., et al. 2020, ApJ, 891, L29

Hynes, R. I., Steeghs, D., Casares, J., Charles, P. A., & O'Brien, K. 2003, ApJ, 583, L95

Jakob, W., Rhinelander, J., & Moldovan, D. 2017, pybind11 – Seamless operability between C++11 and Python, https://github.com/pybind/pybind11

Kalemci, E., Dinçer, T., Tomsick, J. A., et al. 2013, ApJ, 779, 95

Kara, E., Steiner, J. F., Fabian, A. C., et al. 2019, Nature, 565, 198

Kawamura, T., Axelsson, M., Done, C., & Takahashi, T. 2022, MNRAS, 511, 536

Kluyver, T., Ragan-Kelley, B., Pérez, F., et al. 2016, in Positioning and Power in Academic Publishing: Players, Agents and Agendas, ed. F. Loizides & B. Schmidt, IOS Press, 87 – 90

Koljonen, K. I. I. & Russell, D. M. 2019, ApJ, 871, 26

Kormendy, J. & Richstone, D. 1995, ARA&A, 33, 581

Kotov, O., Churazov, E., & Gilfanov, M. 2001, MNRAS, 327, 799

Kristian, J., Brucato, R., Visvanathan, N., Lanning, H., & Sandage, A. 1971, ApJ, 168, L91

Kylafis, N. D., Papadakis, I. E., Reig, P., Giannios, D., & Pooley, G. G. 2008, A&A, 489, 481

Lyubarskii, Y. E. 1997, MNRAS, 292, 679

Mahmoud, R. D. & Done, C. 2018a, MNRAS, 480, 4040

Mahmoud, R. D. & Done, C. 2018b, MNRAS, 473, 2084

Mahmoud, R. D., Done, C., & De Marco, B. 2019, MNRAS, 486, 2137

Markert, T. H., Canizares, C. R., Clark, G. W., et al. 1973, ApJ, 184, L67

Markoff, S., Nowak, M. A., & Wilms, J. 2005, ApJ, 635, 1203

Mastroserio, G., Ingram, A., & van der Klis, M. 2018, MNRAS, 475, 4027

Mastroserio, G., Ingram, A., Wang, J., et al. 2021, MNRAS, 507, 55

Mattsson, L. & Höfner, S. 2011, A&A, 533, A42

Miller-Jones, J. C. A., Sivakoff, G. R., Altamirano, D., et al. 2012, MNRAS, 421, 468

Miyamoto, S., Kitamoto, S., Mitsuda, K., & Dotani, T. 1988, Nature, 336, 450

Mushtukov, A. A., Ingram, A., & van der Klis, M. 2018, MNRAS, 474, 2259

Newville, M., Stensitzki, T., Allen, D. B., & Ingargiola, A. 2015, LMFIT: Non-Linear Least-Square Minimization and Curve-Fitting for Python

Nowak, M. A., Wilms, J., & Dove, J. B. 1999, ApJ, 517, 355

Pietrini, P. & Krolik, J. H. 1995, ApJ, 447, 526

Podsiadlowski, P., Rappaport, S., & Pfahl, E. D. 2002, ApJ, 565, 1107

Pottschmidt, K., Wilms, J., Nowak, M. A., et al. 2003, A&A, 407, 1039

Rapisarda, S., Ingram, A., Kalamkar, M., & van der Klis, M. 2016, MNRAS, 462, 4078

Reig, P., Kylafis, N. D., & Giannios, D. 2003, A&A, 403, L15

Reinecke, M. 2022, PocketFFT: FFT implementation based on FFTPack, accessed: 14/06/2022

Ross, R. R. & Fabian, A. C. 2007, MNRAS, 381, 1697

Rothschild, R. E., Boldt, E. A., Holt, S. S., & Serlemitsos, P. J. 1974, ApJ, 189, L13

Sai, H., Wang, X., Wu, J., et al. 2021, MNRAS, 504, 4226

Sandin, C. & Höfner, S. 2003, A&A, 404, 789

Sandin, C. & Höfner, S. 2004, A&A, 413, 789

Schwarzschild, K. 1916, Sitzungsberichte der Königlich Preußischen Akademie der Wissenschaften, 189

Shakura, N. I. & Sunyaev, R. A. 1973, A&A, 24, 337

Shapiro, S. L., Lightman, A. P., & Eardley, D. M. 1976, ApJ, 204, 187

Shipman, H. L. 1975, Astrophys. Lett., 16, 9

Stroustrup, B. 2000, The C++ programming language (Pearson Education India)

Strüder, L., Briel, U., Dennerl, K., et al. 2001, A&A, 365, L18

Tauris, T. M., van den Heuvel, E. P. J., & Savonije, G. J. 2000, ApJ, 530, L93

Torres, M. A. P., Casares, J., Jiménez-Ibarra, F., et al. 2019, ApJ, 882, L21

Turner, M. J. L., Abbey, A., Arnaud, M., et al. 2001, A&A, 365, L27

Uttley, P., Cackett, E. M., Fabian, A. C., Kara, E., & Wilkins, D. R. 2014, A&A Rev., 22, 72

Uttley, P. & Malzac, J. ????

Uttley, P., McHardy, I. M., & Vaughan, S. 2005, MNRAS, 359, 345

Uttley, P., Wilkinson, T., Cassatella, P., et al. 2011, MNRAS, 414, L60

Van Rossum, G. & Drake, F. L. 2009, Python 3 Reference Manual (Scotts Valley, CA: CreateSpace)

Veledina, A., Poutanen, J., & Vurm, I. 2013, MNRAS, 430, 3196

Virtanen, P., Gommers, R., Oliphant, T. E., et al. 2020, Nature Methods, 17, 261

Wang, J., Kara, E., Lucchini, M., et al. 2022, ApJ, 930, 18

Wang, J., Mastroserio, G., Kara, E., et al. 2021, ApJ, 910, L3

Webster, B. L. & Murdin, P. 1972, Nature, 235, 37

Wilkinson, T. & Uttley, P. 2009, MNRAS, 397, 666

Wilson, A. 2005, ESA achievements : more than thirty years of pioneering space activity, 3rd edn., ESA BR (ESA Publications)

Zdziarski, A. A., Dziełak, M. A., De Marco, B., Szanecki, M., & Niedźwiecki, A. 2021, ApJ, 909, L9