Analyzing HST Fomalhaut b data using Non-negative Matrix Factorization

Ivan A. Abreu Paniagua
Department of Physics, University of Rhode Island
Department of Physics, University of California Santa Barbara

Dr. Maxwell Millar-Blanchaer
Assistant Professor, Department of Physics, University of California Santa Barbara

Connor Vancil
Graduate Student Mentor, Department of Physics, University of California Santa Barbara

October 2, 2023

Abstract

Since its discovery in 2004, Fomalhaut and its companion, Fomalhaut b, have undergone much investigation. Originally believed to be an exoplanetary system, the prevailing theory supports the idea that a collision between two planetesimals resulted in a dust cloud now observed to be Fomalhaut b. In this paper we further investigate the nature of this object. Unlike previous examinations where point source-based post processing methods were used, we decide to utilize nonnegative matrix factorization (NMF). NMF works best for detecting broad and spread sources such as disks and dust clouds. By using NMF, we work to examine Fomalhaut b and compare results from previous works. Due to its iterative nature, NMF uses significant computing resources and compile times. To work around this, we implement Google's JAX NumPy and Just-In-Time (JIT) compilation to increase efficiency and speeds. By running code on accelerators (graphics processing unit) and JIT's ability to compile given functions all at once, this allows us to improve on computing times greatly. The data we use consists of different years of direct imaging of Fomalhaut through the Hubble Space Telescope coronagraph, that we then use NMF to process. After locating Fomalhaut b in these images, we then examine its make-up pixels for differences in relative size and brightness across different years and works.

1 Introduction

Despite being statistically one of the most common objects in the universe, exoplanets are poorly sampled. Despite there being an estimated 100 billion exoplanets in the Milky Way alone, as of August 2023 only 5496 have been discovered across the universe [8]. One of the many methods astronomers use to detect and analyze data from planetary systems is Direct Imaging (DI). Using DI, astronomers attempt to take a direct snapshot of the space and material around a star, in hopes of capturing the light from circumstellar objects [3]. Unlike other methods, DI allows astronomers to gather large amounts of data on the traits of these objects. Composition, atmospheric data, and trajectory are some of the characteristics of exoplanets that can be effectively extracted via DI. Despite its promise, DI is extremely challenging compared to other methods. Stars are billions of times brighter than their planets, thus the planet light gets drowned out. Additionally, one of the consequences of taking images of these far away systems is the point spread function (PSF). The point spread function occurs when taking data from sources far enough that they appear to be a point of light. As one transfers the image into data, the information gets spread out across multiple positions, turning these points into blurs in the resultant image. Stellar PSFs, being many times brighter than the objects orbiting them, overlap the light of any planets or disks around them.

Due to DI relying solely being able to distinguish the light from circumstellar objects, this makes the process significantly more difficult. Astronomers thus have to come up with different methods to subtract the excess noise and bring out the planets.

1.1 Methods of image processing

Reference differential imaging (RDI), angluar differential imaging (ADI), and spectral differential imaging (SDI), often accompanied by more post-processing methods such as locally optimized combination of images (LOCI) and Karhunen–Loève image projection (KLIP) are some of the primary ways to subtract much of the noise and PSF from stellar images. All of these methods are suited for calculating and subtracting the PSFs and noise of point sources, allowing astronomers to study stars and their surrounding planets. Although these methods are useful these ideal situations, they often suffer from overfitting and subtracting errors. These errors are especially prominent in any other sources that are not point based; *i.e.* large and spread structures such as circumstellar disks and clouds. To account for these errors, methods such as forward modelling are used. These methods come with their share of complications, as they are often used for point sources and not well suited for others [10]. Here we decide to investigate other methods of image processing that are able to accurately subtract out stellar PSFs and construct circumstellar disks.

1.2 Non-negative matrix factorization

Another proposed method of post processing stellar system data is through the utilization of non-negative matrix factorization (NMF). First proposed in [9], NMF separates the circumstellar disk from an image by taking reference images and interatively builds the components off of them. When building these components, NMF avoids making any of these components' values negative, which helps minimize over-subtraction. From these reference images and built components, the NMF method then creates a model of the given target image. This new model can then be deconstructed to separate the disk from the stellar signal. This is done through a forward modeling method distinct to NMF, which allows the algorithm to circumvent the drawbacks of standard forward modeling [14][10]. Due to the current consensus on the nature of our case study Fomalhaut b, we decided to use NMF in hopes that it can give us new insight as compared to other processing methods .

1.3 Fomalhaut b

First observed in 2004 by the Hubble Space Telescope Advance Camera for Survey (HST ACS), Fomalhaut b is a circumstellar object orbiting the star Fomalhaut A about 25.11 ly away [5]. Originally believed to be an exoplanet, Fomalhaut b has undergone extensive investigation. In recent observations, Fomalhaut b seems to have grown larger and dimmer. The current prevailing theory is that of a circumstellar collision between two bodies resulted in a dust cloud perceived to be an exoplanet [4]. This dust cloud has thus been expanding over the years. For that reason, we decided to investigate Fomalhaut b using NMF. NMF has not been applied to Fomalhaut system before, which opens the possibility for new insight on past Fomalhaut b data. If the cloud model is accurate, then we hope to see improvements in the processed data where Fomalhaut b should be, further solidifying the cloud model as the primary theory.

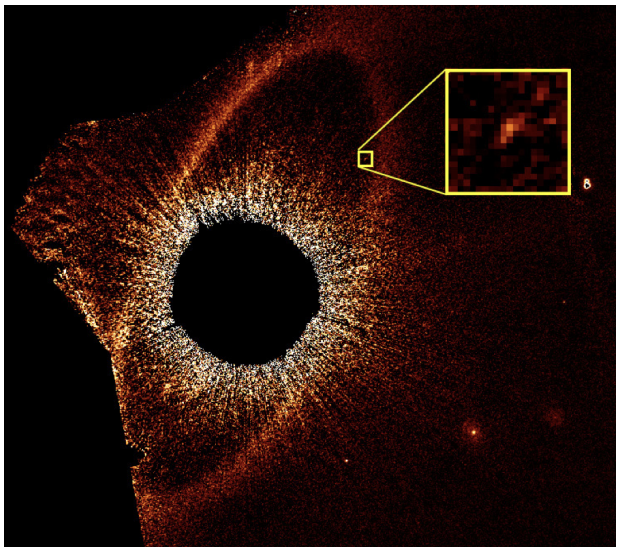

Figure 1: False color 2012 HST images of Fomalhaut b processed by [6].

This paper is split into multiple sections detailing this research process. Section 2 explains the programs used for implementing NMF on target images as well as some complications with these implementations and how we can improve on them, as well as detailing how we use NMF on Fomalhaut data. In section 3 go over the results of our computations and compare our results between years. Finally, we summarize our results and improvements on NMF in section 5. Additionally, we also go over

Run times (sec)				
Environment	2010	2012	2013	2014
Mac	265.617 ± 23.393	1250.604 ± 359.580	671.398 ± 43.686	303.906 ± 41.927
Mueller	172.706 ± 44.683	1228.121 ± 159.156	701.357 ± 132.444	281.781 ± 46.948
Mueller + JAX JIT	51.725 ± 11.104	62.464 ± 7.016	44.299 ± 1.151	31.108 ± 6.360

Table 1: Run time comparisons of nmf_imaging on different environments. Measured times are for the entire nmf_imaging code.

Run times (sec)				
Environment	2010	2012	2013	2014
Mac	265.606 ± 23.394	1250.585 ± 359.575	671.385 ± 43.685	303.893 ± 41.930
Mueller	172.698 ± 44.681	1228.113 ± 159.156	701.349 ± 132.444	281.775 ± 46.947
Mueller + JAX JIT	51.718 ± 11.103	62.456 ± 7.016	$44.291 + 1.151$	31.103 ± 6.360

Table 2: Run time comparisons of nmf_imaging on different environments. Measured times are for the functions in the nmf_imaging code that are effected by implementing JAX; when building the components and model.

2 Methods

Most of our work is done in Python, utilizing a variety of programs to access Fomalhaut data and then process them. External programs such as SAOImageDS9 will be used to visualize data before and after processing.

2.1 Fomalhaut Data

The data we will use is a compilation of 2010 to 2013 direct images of Fomalhaut. These images were taken by HST Space Telescope Imaging Spectrograph (STIS) in the **blank** range. These images were accessed through [6]. Each year consists of separate visits done at different roll angles, with each visit having multiple images. This gives us the advantage of allowing us to use the different roll angles as references for NMF in a similar manner to ADI. The Kalas *et al.* data set also includes images of the star Vega, which also allows us to build more references out of this much like in RDI. Each of these data sets are accompanied by masks that cover the given star, coronagraphs, and diffraction spikes. These masks thus allow our algorithms to disregard those pixel values and favor everything else.

2.2 Processing Methods

To analyze the Fomalhaut system data through NMF, we will be using the NonnegMFPy and nmf_imaging Python packages respectively [14] [10]. The nmf_imaging package contains the main code needed to do a NMF subtraction on exostellar images, namely functions that calculate the components, make the models, find the best factor, and subtract from the target image by utilizing target and references as well as their errors, and an arbitrary mask to identify the information we are interested in. To calculate the components and models, nmf_imaging uses the NonnegMFPy package to do the background NMF work. Rin *et al.* implemented NMF in an iterative nature, where each iteration converges to an end condition. For NonnegMFPy this end condition is a small enough χ^2 value, although one can set a maximum to the number of iterations. This results in large computation times for data sets of considerable sizes, thus we have to account for this by adjusting the algorithm.

2.3 Implementing JAX

To speed up the computational process, we can either improve our hardware or speed up the software. Since NMF still holds long computing times even with top hardware, improving the software is the only option. To do so, we use Google's JAX python package [11]. JAX works to replace NumPy while improving on run times. This is done by allowing JAX NumPy code to run on accelerators such as graphics processing units (GPUs) and tensor processing units (TPUs). Specifically, our utilization of JAX works by incorporating TensorFlow's XLA (Accelerated Linear Algebra). XLA allows code and matrix operations to execute using GPU kernels. XLA then fuses different matrix operations to run on one kernel, freeing up other kernels for other operations. This, combined XLA's ability to stream the operations' results off of memory, saves substantial amounts of memory bandwidth, in turn significantly improving per-

formance [7]. These features are translated directly to JAX, where JAX now applies them to extensive matrix operations, *i.e.* NumPy operations. Since NonnegMFPy's matrix computations are heavily dependent on NumPy operations, this is the best way to improve efficiency. Additionally, the JAX package also includes its Just-In-Time (JIT) functionality. JIT treats a desired function like a library, working on them immediately when compiling. It converts the function into a separate intermediate language that can then be traced. Variables in this function have all of the compatible operations on them traced throughout and recorded. This traced function is then feed to the XLA compiler all at once, as XLA is optimized to work with large amounts of code. In the end, our JITed function and however many times it is called are run simultaneously at compile time instead of waiting for each function call. JAX JIT then results in even faster computing times. We replace the NonnegMFPy NumPy code with their JAX NumPy alternatives to make the package compatible with JAX and JIT. After determining which function takes up the most run time, we convert it to being JAX JIT compatible while maintaining its original functionality. This involves rewriting many of the conditionals and loops, as JAX JIT is generally incompatible with their Python built in counterparts.

Implementing JAX and JIT helps runtime most when building the components and forming the model. These two processes both depend on the matrix operations of NonnegMFPy, thus benefit from JAX. Due to how scalable the component building process is, this often takes up the longest computing time out of all. The component building process is detailed in 5, and our results from applying JAX are shown in 3.

To be able to use nmf_imaging, NonnegMFPy, and JAX together, we made a front-end Python file that extracts the Fomalhaut data and prepares it for processing. This is done by first opening the data in the form of .FITS files using Astropy [1]. Here we also use the same package to open and extract the mask data. We then collapse all of the images in each visit down to one image. These collapsed arrays are then re-centered around Fomalhaut A. We use radonCenter [2] to find the star center and pyKLIP [13] to re-center.

3 Results

3.1 Code improvements

As a big part of our research, improving the efficiency of the software used to execute NMF is essential to testing the quality of NMF on the Fomalhaut system. After implementing JAX NumPy to the NonnegMFPy code, we determined that the SolveNMF function in this .py file took the longest computing time. This is a result of a while_loop that performed considerable matrix operations with every iteration. This loop calls another function that recalculates the χ^2 values. This is done through some more matrix operations, allowing for the loop to meet its end condition when the difference in sequential χ^2 values is small enough. This work helps us in determining which function to run JAX JIT on, as JIT has very specific behaviors that need to be met for it to run effectively [12]. JIT can also result in diminishing returns if it is used on an entire file that is not catered towards it. After altering the code to be JIT compatible while still keeping its functionality, some of our preliminary results can be seen in Table 1. These tests are done for each of the years in our given Fomalhaut database. We utilize all of the maximum number of references for each test and kept the modes to 5 components unless not enough components could be used. Such is the case for the 2010 data where only 3 modes were used. Each test is also conducted without data imputation and only uses the corresponding masks.

These results show that implementing JAX NumPy and JIT results in significantly lower run times for the iterative processes of NMF. The uncertainties were extracted by doing multiple timing runs to account for the random initial conditions of NMF. These measurements were done by timing the entire NMF process, from building the components to extracting the results from subtraction. Since best factor finding (if utilized) and subtraction do not utilize any of the JAX impacted code, these portions see no change in computing time. They still do maintain a constant computing time throughout our tests, as they were all done using consistent samples. Table 2 includes the timing comparisons done on just the functions that are affected when we implement JAX; component and model building. These results are consistent with Table 1, as both show significant improvements in efficiency. JAX also shows an improvement in consistency for run times, as the uncertainties for our JAX JIT results are lower relative to their respective values. The difference in times between Table 1 and Table 2 demonstrates how much computing time NMF calculations take. An overwhelming majority of the computing resources are allocated for the component and model building functions as compared to any other operations.

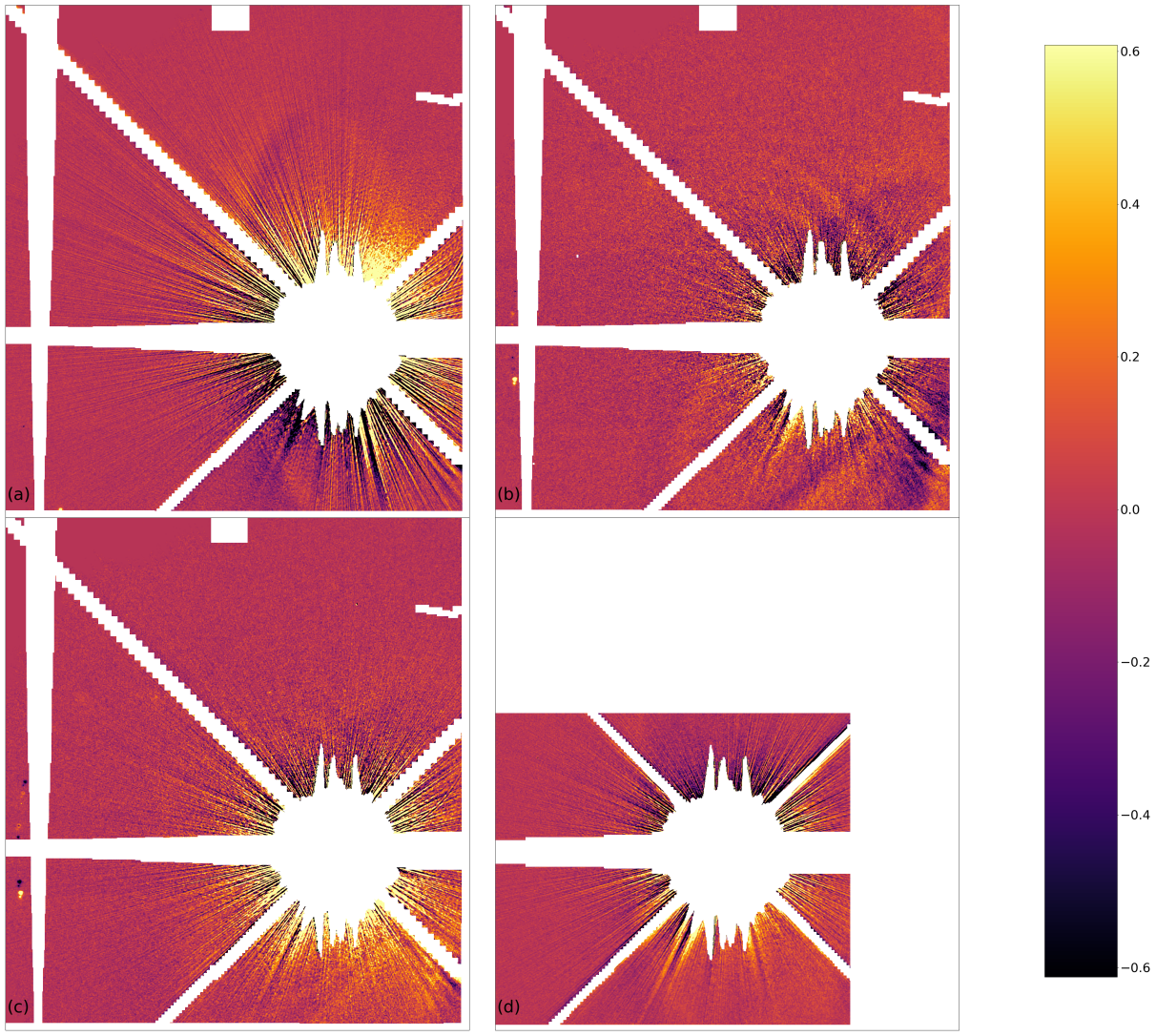

Figure 2: Single target image plots of Fomalhaut. This includes 2010 (a), 2012 (b), 2013 (c), and 2014 (d) data.

3.2 NMF implementation on Fomalhaut b

After incorporating JAX and speeding up run times for our code, we can now efficiently test and compare the results of NMF. Fig. 2 shows the single NMF processed images of the Fomalhaut system throughout our different HST data sets. (a), (b), (c), (d) show the processed data from 2010, 2012, 2013, and 2014 respectively. Here we have the PSF reduction on one target image with the number of references and components being dependent on the number of roll angles supplied in each data set. (b), (c), and (d) were processed using 5 components and their appropriate masks. (a) was processed using 3 components as this is the maximum we can do for the number of images given in the 2010 data. These low component numbers results in a worse PSF reduction for the 2010 data as compared to the other three. Additionally, due to the smaller aspect ratio of the 2014 data that was provided, the resultant image is smaller than the rest. This is further amplified by the re-centering step which cuts off parts of all four images. We cannot locate any objects that resemble Fomalhaut b in any of these images, but there is a faint residue of Fomalhaut's circumstellar disk. To extract both of these objects, we perform the post processing stacking method detailed in 2.

Once we rearrange and stack the images together, we get the results seen in Fig. 3. Similarly to Fig. 2, (a), (b), (c), (d) correspond to 2010, 2012, 2013, and 2014 data. The PSF for the 2010 data is prominent

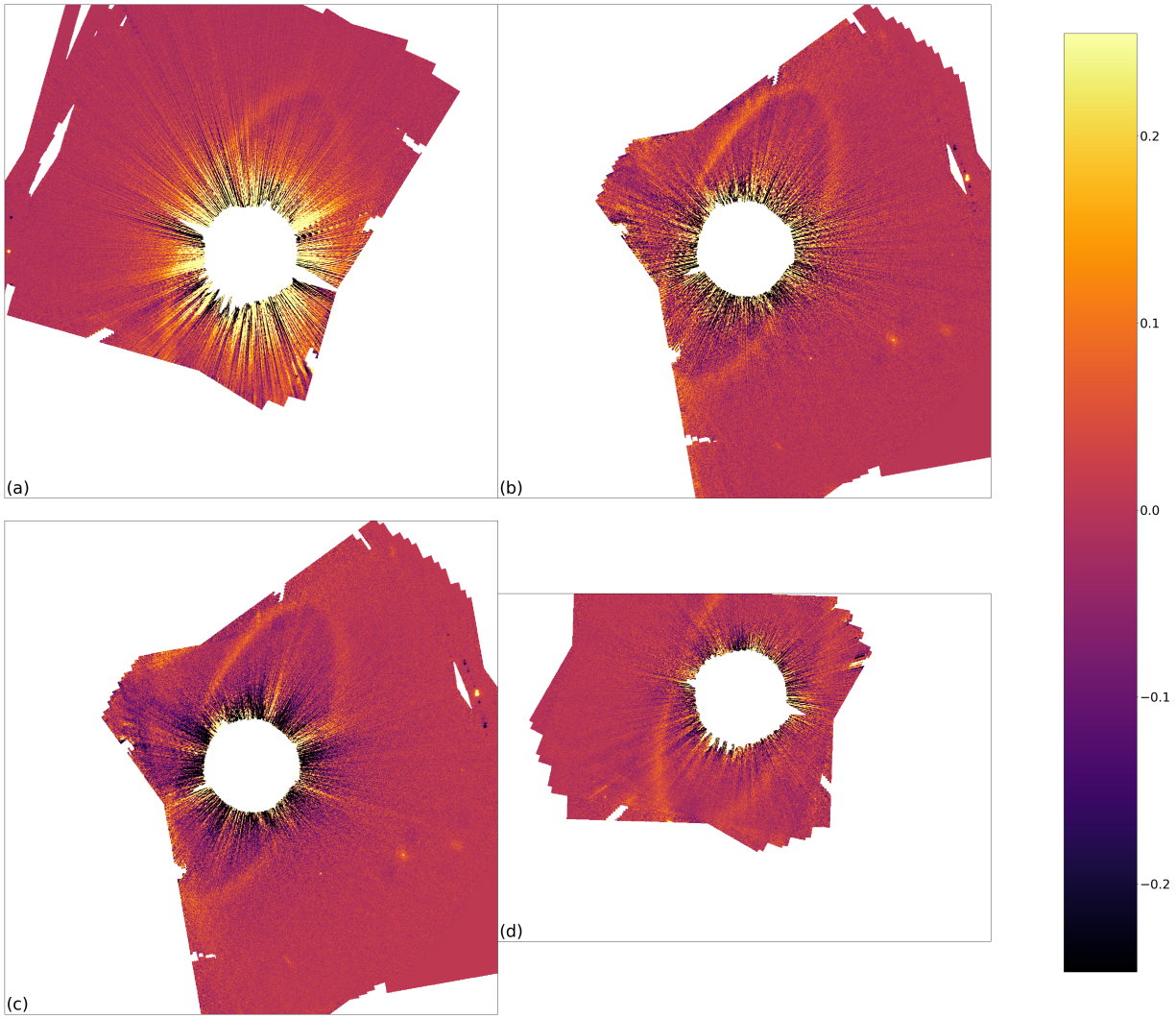

Figure 3: Stacked target images of Fomalhaut. Similarly to 3, this consists of 2010 (a), 2012 (b), 2013 (c), and 2014 (d) data.

compared to the other three images, still a result of the low component numbers as seen in Fig. 2. Additionally, due to the smaller image size of the 2014 data, the resultant image in (d) is also smaller than the rest. These graphs are comprised of the images shown in Fig. 2, as well as the remaining target images. These results show a very prominent circumstellar disk around Fomalhaut A. This falls within our expectations, as NMF has been proven to work well for these purposes [10].

When we zoom in to where we expect Fomalhaut b to be [4], we can see objects consistent with Fomalhaut b in Fig. 4. When analyzing these areas, we see that Fomalhaut b is not located in every image. For the 2012 (b) and 2013 (c) data, we notice bright spots in these areas. We determine that the centermost object in (b) is Fomalhaut b. In the case of (c), the object slightly off center is Fomalhaut b. The movement of this spot across time allows us to be confident that this is Fomalhaut b we are observing instead of PSF residue. Due to the limited sizes of the 2014 data, not all of the constituent processed images overlap in this search region. The resultant image (d) lacks any prominent spots we can confidently point out to be Fomalhaut b. Many of the objects seen in this area are likely PSF residue or the circumstellar disk. We observe similar behaviour when examining the 2010 data. Due to the lack of sufficient data to make references, we can only process the 2010 data with a maximum of 3 components. The consequence of this is lower PSF subtraction in our results as compared to other years. When we observe our search region,

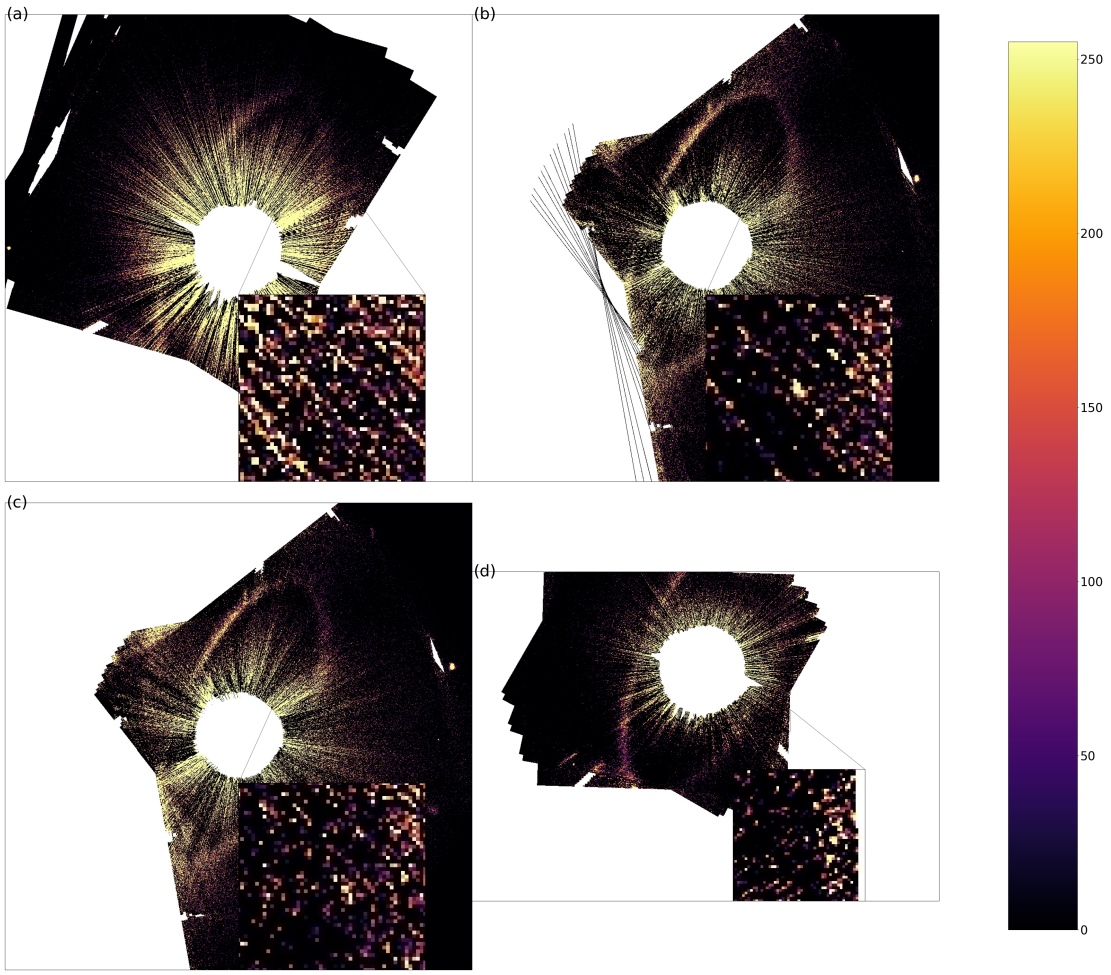

Figure 4: Same as Figure 3 using a different contrast and color bias. Zoomed in on the location of where Fomalhaut b is expected to be. (a) shows the expected location of Fomalhaut b in 2010 with relatively poor PSF subtraction. (b) shows the 2012 location of Fomalhaut b, with it being very visible. In the 2013 (c) and 2014 (d) locations, Fomalhaut b is more difficult to locate in their respective regions.

no objects comparable to Fomalhaut b can be seen. To account for this and other complications, we have to do further pre and post processing adjustments.

3.3 Component number comparisons

To test the importance of modes number on our NMF results, we can run our NMF program using different component numbers and compare the results. Fig. 5 shows this comparison, with the component number labeled on the bottom left of each result. We note improvements on the PSF reduction for our results as the number of modes increase starting at 2. This improvement starts to plateau as the modes increase past 5. For any number of modes past 8, there is no visible improvement in PSF reduction.

In Figure 6, we have the results of the region of Fomalhaut b after varying the component numbers for our 2013 data. Despite Fomalhaut b being less prominent as compared to Figure 5, the 2013 data shows a similar behavior to it. There is a large improvement in the PSF reduction as the component numbers increase before stagnating. Our 2013 data also shows a comparably worse PSF reduction than the processed 2012 data.

Figure 7 shows the variations in component numbers for the 2010 data set. The PSF reduction for this dataset is notably worse than the other datasets, which can primarily be attributed to the low reference numbers of this dataset. Improvements in the PSF subtraction are shown as the component numbers increase from 2 to 3. For NMF, a correlation between component number and quality of results is something we will investigate further 4.

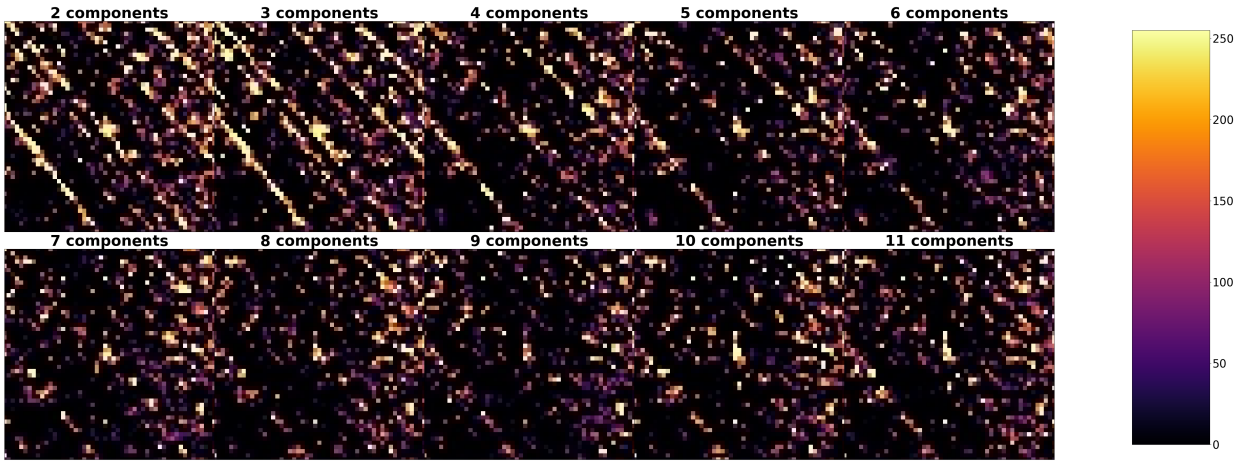

Figure 5: 2012 data processed with different component numbers. Images shown are of the location of Fomalhaut b with the object in the center.

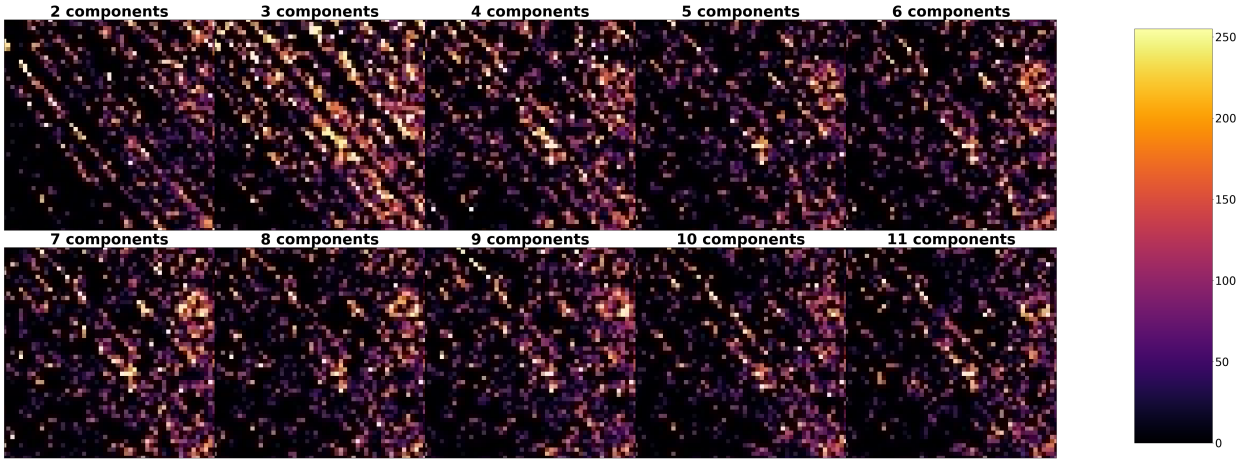

Figure 6: 2013 data processed with different component numbers. Images shown are of the location of Fomalhaut b.

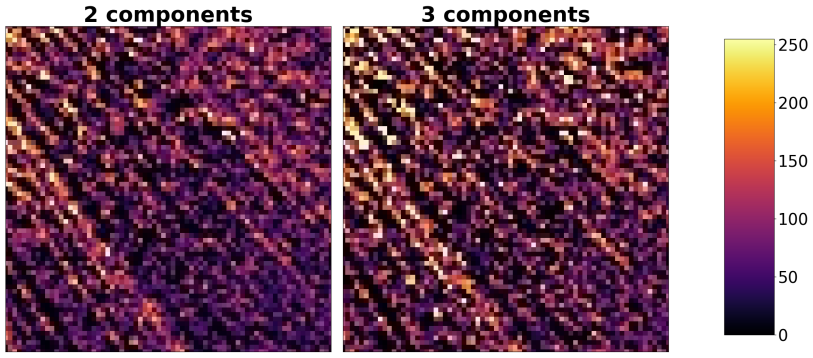

Figure 7: 2010 data processed with different component numbers. Images shown are of the location of Fomalhaut b.

4 Future Work

Now that we have improved the efficiency of NMF implementation for exoplanetary data, we hope further apply NMF to Fomalhaut b. With its lower utilization of computing resources, we plan to work with JAXed NMF by changing some of the pre and post NMF parameters. Our hopes are that we can further improve on the accuracy of the resultant images and reduce as much noise as possible.

Once we have consistent reductions and results, we will do a full investigation on Fomalhaut b. In this investigation will consist of a comparison of the size and brightness of Fomalhaut b across the successive years of our given database. Additionally, we hope to compare our results with those from point source methods (KLIP, RDI, etc.). Due to the difference in nature between NMF and point source processing methods, if Fomalhaut b is a debris cloud [4] then our NMF results should be more accurate.

5 Conclusions

Our JAX comprised NMF code shows considerable improvements in run time as compared to the base NMF operations. This helps in the analysis of any DI source as it speeds up the computing process. One of the behaviors that we observed with JAX NMF is that the comparative run times grow with input data size. NMF and nmf_imaging tend to slow down significantly when using more references and higher component numbers. While our JAX NMF does trend to higher computing times under the same circumstances, the difference in computing times between JAX and non-JAX NMF grows considerably. The reasons behind this are due for further investigation in our future work.

Our analysis of Fomalhaut b proved successful in locating the object after processing with NMF. Fomalhaut b is prominent in our 2012 and 2013 data samples, yet it fails to show definitively in the 2010 and 2014 images. We believe this is a result of the nature of NMF processing. When making the components and model, a higher number of references trends to better building of PSF models that can then be subtracted away. Additionally, the number of modes is tied to the reference numbers, thus more references allows for more modes. This then further supports our image quality, as we examined how lower component numbers are correlated with poor PSF subtractions. To account for these errors, we hope to improve our pre and post NMF methods to collect better results. See section 4 for more on our future work.

The code used for our research can be found on GitHub, where it will be kept updated with our improvements in future work.

Acknowledgements

I would like to give my dearest thanks to my project advisor and primary investigator, Dr. Max Millar-Blanchaer. His guidance and mentorship were crucial for the development of this project and my personal growth. I must also thank my graduate student mentor, Connor Vancil, for the consistent support and advising throughout this entire process. I would like to show my appreciation to everyone in the Max Millar-Blanchaer research group for their kindness and aid, as well as additional thanks to Dr. Millar-Blanchaer for allowing me to be part of this group. My dearest gratitude to Dr. Sathya Guruswamy for organizing the UCSB REU program and providing me with such a great opportunity, as well as her continued career guidance and encouragement. Lastly, I would like to thank the National Science Foundation for making all of this possible and funding this opportunity with the grant PHY-1852574.

Appendix

Our JAX version of nmf_imaging and NonnegMFPy is posted to `https://github.com/ivanabreu0/NMF_imagingJAX`.

References

[1] Astropy Collaboration et al. "The Astropy Project: Sustaining and Growing a Community-oriented Open-source Project and the Latest Major Release (v5.0) of the Core Package". In: 935.2, 167 (Aug. 2022), p. 167. DOI: 10.3847/1538-4357/ac7c74. arXiv: 2206.14220 [astro-ph.IM].

[2] Rin Ben and Jason Wang. *Determination of Star Centers based on Radon Transform*. 2018. URL: https://github.com/seawander/centerRadon (visited on 08/01/2023).

[3] Thayne Currie et al. "Direct Imaging and Spectroscopy of Extrasolar Planets". In: (2023).

[4] Andras Gaspar and George Rieke. "New HST data and modeling reveal a massive planetesimal collision around Fomalhaut". In: *Proceedings of the National Academy of Sciences* (2020).

[5] Paul Kalas et al. “Optical Images of an Exosolar Planet 25 Light Years from Earth”. In: *Science* (2008).

[6] Paul Kalas et al. “STIS Coronagraphic Imaging of Fomalhaut: Main Belt Structure and the Orbit of Fomalhaut b”. In: *The Astrophysical Journal* (2013).

[7] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[8] NASA Exoplanet Archive. *Exoplanet and Candidate Statistics*. 2021. URL: https://exoplanetarchive.ipac.caltech.edu/docs/counts_detail.html (visited on 08/10/2023).

[9] Pentti Paatero and Unto Tapper. “Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values”. In: *Environmetrics* (1994).

[10] Bin Ren et al. “NON-NEGATIVE MATRIX FACTORIZATION: ROBUST EXTRACTION OF EXTENDED STRUCTURES”. In: (2018).

[11] The JAX Authors. *JAX: High-Performance Array Computing*. 2023. URL: https://jax.readthedocs.io/en/latest/index.html (visited on 08/01/2023).

[12] The JAX Authors. *Just In Time Compilation with JAX*. 2023. URL: https://jax.readthedocs.io/en/latest/jax-101/02-jitting.html (visited on 08/01/2023).

[13] Jason J Wang et al. “pyKLIP: PSF Subtraction for Exoplanets and Disks”. In: *Astrophysics Source Code Library* (2015). URL: https://bitbucket.org/pyKLIP/pyklip.

[14] Guangtun Ben Zhu. “NONNEGATIVE MATRIX FACTORIZATION (NMF) WITH HETEROSCEDASTIC UNCERTAINTIES AND MISSING DATA”. In: (2016).