Efficient spectral volumetric rendering based on astrophysical assumptions

Julien Barrès

Master of Science in Informatics at Grenoble
Master Informatique
Specialization HDWI

Efficient spectral volumetric rendering based on astrophysical assumptions

Julien Barrès

June 26th, 2023

Research project performed at Maverick
INRIA Grenoble Alpes - LJK

Under the supervision of:
Fabrice Neyret

Defended before a jury composed of:
Head of the jury: Dominique Vaufreydaz
Examiner: Céline Coutrix
External expert: Laurent Belcour

June 2023

Abstract

Physically accurate spectral rendering generally brings further computational complexity to RGB rendering pipelines which precludes real-time rates. In this report, we show that by building on several physical a-prioris for the spectral phenomena in standard astronomical scenes, we achieve to obtain a model greatly reducing the complexity of the spectral aspect for volumetric rendering - by up to 83 factor in our examples. By projecting the total local contribution on a small optimal basis, thus providing scalability for the method, our rendering scheme then makes it possible to render a volumetric scene with any number and spectra of lights in real-time and no significant visual discrepancies.

Acknowledgement

I would like to give my sincere thanks to my supervisor **Fabrice Neyret** for his help throughout the project, and the opportunity to work on such interesting research question at Maverick. His constant availability for questions, discussions, and feedback truly enriched my understanding of the subject and the quality of my work.

I also would like to thank **Antoine Richermoz** for developing and sharing his Vulkan framework *shaderV* with which the shaders of this project were run, providing an efficient tool that helped me get results without having to build a whole GPU framework from scratch during these five months.

Résumé

Le rendu spectral physique implique généralement une augmentation de la complexité de calcul par rapport aux pipelines de rendu RGB, ce qui exclut les possibilité de rendu en temps réel. Dans ce rapport, nous montrons qu'en nous appuyant sur plusieurs a-prioris physiques pour les phénomènes spectraux dans les scènes astronomiques standards, nous parvenons à obtenir un modèle réduisant considérablement la complexité de l'aspect spectral pour le rendu volumétrique - d'un facteur allant jusqu'à 83 dans nos exemples. En projetant la contribution locale totale sur une petite base optimale, assurant ainsi l'extensibilité de la méthode, notre schéma de rendu permet alors de rendre une scène volumétrique avec n'importe quel nombre et spectre de lumières en temps réel et sans défauts visuels significatifs.

Contents

— 1 —
Introduction

This work is part of a long-term project involving the ANR Galaxy/veRTIGE between INRIA, RSA-Cosmos and Observatoire de Paris-Meudon [Anr]. The goal of this project is to render complex and physically accurate astronomical scenes in real-time for entertainment and educational purposes. We focus here in particular on the accurate spectral rendering of astronomical scenes.

In astrophotography, the process of obtaining a picture often involves stacking and compositing a set of pictures taken using specific filters, as illustrated in figure 1.1. Each of these filters puts emphasis on a specific (large or narrow) range of wavelengths in the scene during acquisition (ex: ultraviolets, infrared, x-ray), which corresponds to summing the intensity

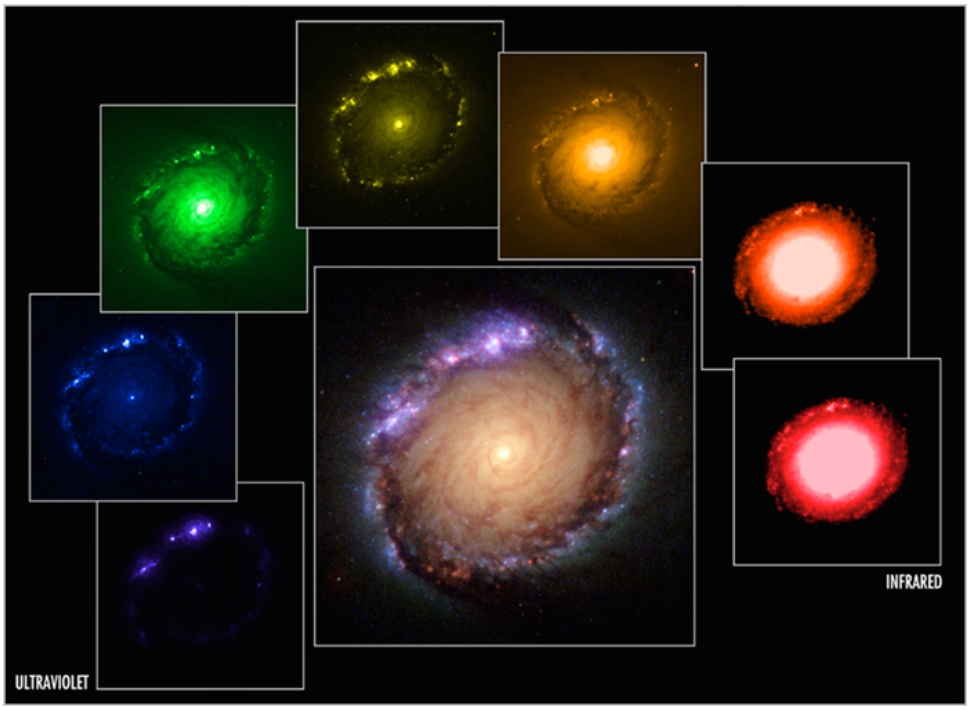


Figure 1.1: Stacking process for a picture of NGC 1512 combining the image acquisition obtained using 7 different filters
Image credit: European Space Agency, HST

contribution of all the ray lengths in the range of interest. This process is too complex to be accurately computed by traditional rendering pipelines using RGB approximation. This approximation assumes that all the light of the world and its interactions are only made of three pure colors (pure red, green and blue), while to be able to accurately render complex scenes involving non-linear light transformation (as in the case with nebulae) the range of wavelengths needs to be considered as a continuous spectrum. There is thus already a real challenge in trying to produce accurate renders of such astronomical objects without considering the real-time constraints. Moreover, we want to still be able to explore a significant variable space during render: the ability to freely move in the space around the nebulae is a prerequisite in order to get a compelling interactive exploration, but we would also want to be able to play with the filter spectra at run-time, similarly to how an operator would change a filter on the JWST in order to take a picture focusing more on an ultraviolet range, or an infrared one.

This work's purpose is thus to produce a complete model for nebulae rendering, taking into account all of these previous requirements, all of that under real-time constraints. In section 2, we introduce in more detail the spectral rendering problem and describe our requirements for a relevant solution, and look at the literature approaches to this problem. Then, we present in section 3 an extensive mathematical description to model the spectral phenomenon (reflection, emission, attenuation, filtering) taking part in an arbitrary astronomical scene, basing our models on physical observations. We use this model in section 4 to propose an efficient method for the spectrally accurate computation of these scenes under real-time constraints. We also show and evaluate the results of our implementation of the method and its performance.

— 2 —
State of the art

2.1 Spectral volumetric rendering

Nebulae are astronomical objects made of dust, which can be seen as *participating media*, meaning that the medium by itself interacts directly with the light-rays passing through it. Due to this nature, nebulae cannot be accurately rendered using traditional surface-based rendering techniques.

Unlike those methods, **volumetric rendering** considers the properties (density, absorption...) of the medium filling the space. This allows for the accurate representation of phenomena such as fog, smoke, and other participating media that interact with light, like in our case nebulae.

The **Volumetric Rendering Equation** (VRE) [Kh84; Kaj86] describes the radiance received on a point x along a viewing direction w. Computing this equation on an array of pixels[1] produces an image of the scene. In most applications, the VRE equation 2.2 is computed for each pixel over three color channels (RGB), associated with characteristic coefficients depicting the scattering, extinction, and transmission properties of the material over three pure wavelengths (red, green, blue) to approximate human vision.

$$\begin{aligned} L_{local,\lambda}(t,w) &= T(t,\lambda)[\,\sigma_s(t,\lambda)L_s(t,w,\lambda)+\sigma_a(t,\lambda)L_e(t,w,\lambda)] \\ T(t,\lambda) &= \exp\left(-\int_{u=0}^{t}\sigma_t(u,\lambda)du\right) \end{aligned} \tag{2.1}$$

T is the light extinction from the camera to point t, σ_a, σ_s and σ_t are respectively the light absorption, scattering, and extinction coefficients, L_s is the external light arriving at a point and scattered in a particular viewing direction (its expression can be more or less complex, modelling multiple bounces through the medium or not) and L_e if the light emitted at point t directly by the material.

$$L_\lambda(x,w) = \int_{t=0}^{E} L_{local,\lambda}(t,w)dt \tag{2.2}$$

The validity of the RGB model for rendering depends especially on two assumptions: the rendering pipeline should only use linear transformations on the spectra, and it should be possible

[1] For planar cameras, the position x varies on the camera plane, and w is the direction from the focal point to the pixel position.

to express any color as a sum of the three RGB base colors. However, these assumptions are often no longer true in most scenes rendered with volumetric rendering. As the generic extinction expression depends on the wavelength of the ray, results can indeed greatly differ between the RGB approximation and the expected render for the accurate simulation of arbitrary bodies. Two metamers cannot be visually differentiated without considering the spectrum as a continuous function over the wavelengths (the RGB representation not being able to differentiate "pure" yellow from a sum of red and green). Light refraction in glass or water is another example of a common physical phenomenon which cannot be simulated using a tri-ray model for the spectrum due to dispersion (visible with rainbows).

The relevance of taking the spectral aspect into account becomes even more important in the case of the study of astronomical objects, as is the case in this report. Astronomers often try to focus on specific wavelength ranges when acquiring astronomical photographs, in order to better discern the contribution of certain elements that have more impact at those wavelengths.

The **Spectral Volumetric Rendering Equation** (SVRE) written in equation 2.3 is a more general equation which takes the wavelength dependence into account, integrating the spectrum over the sensor sensibility.

$$L(x,w) = \int_{\lambda=0}^{\infty} S(\lambda) L_{\lambda}(x,w) d\lambda \tag{2.3}$$

In practice, the SVRE introduces an additional computational complexity based on the spectral integral. In real-time applications of the SVRE [Ber+02], the incoming color is often numerically integrated (sampled) along the two dimensions (wavelength and ray length), which makes the outputs rely even more on numerical resolution than regular volumetric rendering, a value linked with the number of computations done for each integration step. Moreover, this discretization of the space can introduce color artefacts like spectral aliasing.

In the case of real-time rendering of astronomical objects, the need for speed and interactivity poses additional constraints on the rendering pipeline. Real-time rendering requires that images be generated in a matter of milliseconds: this often precludes the use of computationally expensive techniques such as spectral rendering, since ray-marching to compute the spatial integral of the VRE 2.2 is already a costly process.

2.2 Basis functions for analytical integration in literature

To drastically decrease the complexity drawback of the additional spectral integral, a relevant technique is to try to compute analytically the spectral aspect. This would completely nullify the drawbacks of having to keep a spectral discretization, and hopefully enable the computation of the integral between any two bounds in constant time. The most generic form of the SVRE generally isn't integrable analytically, but a common approach in that case is to introduce basis functions to model the different terms of the SVRE. By carefully defining a set of basis functions that can be integrated analytically over a "difficult" integral (and ensuring that their integrability will remain stable over their composition), the computational load of the integral

can be significantly lowered. In the case of spectral rendering (but not necessarily volumetric), a few works have proposed several basis sets to model the involved spectra and ensure a fast and accurate integration (for example using polynomial [RF91], Fourier series [Pee93], or more generic spectrum decomposition using PCA [OYH18; Wad13]). However, most of these sets are defined for surface rendering applications and do not transfer well to a volumetric rendering, where the spectrum is composed in different ways (products and sums). Moreover, the SVRE introduces terms depending on both the spatial coordinate and the spectral coordinate (generally in a non-separable way), which requires a different model than one depending only on the wavelength.

There are several characteristics that need to be taken into account in order to build such a basis set, the main requirement being that it should be **integrable**, i.e. have a simple analytical solution to its spectral integration. It should also be **robust**, meaning that compositions of the elements of the basis set should also remain integrable. This will enable huge computational gains and decrease the time needed to compute a picture, which will be measured at the end of the report. Finally, it should be generic enough but still able to **accurately fit** and model each term of the SVRE 2.3 that are to be spectrally integrated thanks to it.

By building on a set of assumptions obtained from the study of experimental and theoretical data of nebulae and cosmic dust clouds, we will design such a basis set in section 3 that will ensure an analytical solution to the spectral integral part of the SVRE, enabling the possibility to greatly decrease the computational load of the volumetric integration while still providing accurate results in the case of astronomical bodies.

2.3 Three desirable characteristics for our optimal rendering pipeline

We list below three desirable characteristics of a good renderer:

- **Fast**: The renders should be computed in a reasonable time on reasonable hardware, with real-time rendering being the best goal
- **Precise**: Accurate computations, as close as possible to the simulated object
- **Interactive**: The parameters of the renderer should be editable (No long pre-computations should depend on these parameters)

A volumetric renderer able to reach a maximum score in the three characteristics is hard to design (as precision and speed can often be incompatible), and areas of focus can greatly vary regarding the requirements of a project (for example, a model needed to freely explore a parameter space will have a stronger focus on Speed and Interactivity).

The purpose of this work is to be able to obtain an interactive render method making it possible to explore the parameter space of most of the spectral functions of the SVRE, and still be able to compute the full integral in a fast and precise way. In effect, the idea would be to be

able to emulate the rendering process of telescope imagery, being able to change the filters and explore a complex scene in real-time.

— 3 —
Astrophysical assumptions

In the following section, the typical astrophysical scene is described, as well as the different spectra that will contribute to the SVRE 2.3 and the intensity obtained on a camera sensor (in practice, a telescope sensor and a large bank of filters). Models are presented to accurately describe those spectra and justify the functions that are used in the contribution section 4.

3.1 Scene composition and nebulae

A typical scene in astrophysical rendering contains stars as **light sources** and a **participating medium**, in the case of this work an astronomical dust cloud or nebula. Nebulae can be of three types (illustrated in figure 3.1):

(a) (b) (c)

Figure 3.1: (a) Absorption nebula (black object)
(b) Reflection nebula (blue object)
(c) Emission nebula
Image courtesy: nasa.gov

- **Dark** or **absorption nebulae** are very dense clouds that obscure the light of objects behind it.
- **Reflection nebulae** can reflect the incoming light of nearby stars.

- **Emission nebulae** are clouds of ionized gases which are also directly emitting light. The emitted light is restricted to certain wavelengths linked to the recombination of ionized atoms in the nebula cloud.

In effect, many nebulae are potentially a combination of these three optical phenomena. However, one of these phenomena often holds a greater magnitude in the final light contribution which eventually gives its type to the nebula. For example, we can simulate an **absorption nebula** by considering that the extinction term of the SVRE 2.3 is pre-dominant in front of the reflection and emission terms of the equation.

In other words, any nebula can be seen as made up of three kind of contributions that can be gradually added together to form a more complete and accurate render. The dark nebula is the simplest model by mainly being contributed by the extinction of light, the reflection nebula adds on top of that the reflection of nearby stars, and emission adds a local emission term. Each of these three terms can be seen in the VRE equation 2.2 and treated additively to account for a complete render.

3.2 Mathematical models

As the scope of this project is to efficiently compute the double spectral-spatial integral of the SVRE 2.3 by analytically integrating the spectral part, the first step is to propose a mathematical model for the terms of the integral depending on the wavelength λ and the spatial coordinate t. For each phenomenon involved, we present the classical physical equation or experimental data describing it, and then a mathematical function to model it. All the following models are summed up in table 3.1.

3.2.1 Attenuation spectrum

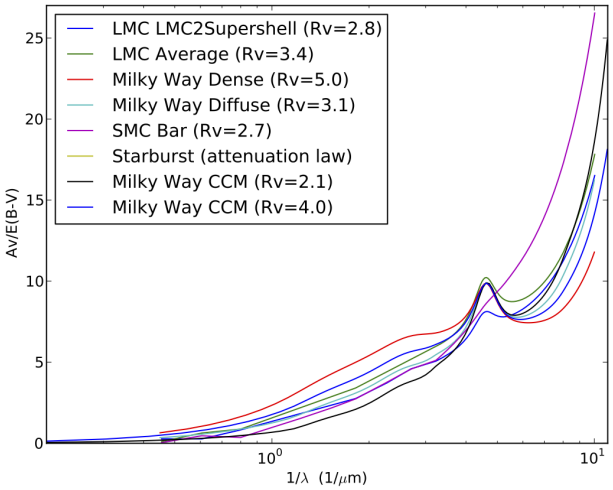


Figure 3.2: Data courtesy: Cardelli & al 1989

Figure 3.2 illustrates the attenuation spectra of various astronomical dust clouds. For high wavelengths ($\lambda > 0.25\,\mu m$), the spectra appear to be linearly correlated with the inverse of the wavelength. This range greatly coincides with the domain of interest in astrophysical imagery, as the filters on Hubble and JWST telescopes range from .1 to 30 µm.

Considering that domain of high wavelengths (low frequency), the following assumption is then made for the spectral attenuation, assuming a linear model: $\tau_{spectrum}(\lambda) \approx \frac{C_T}{\lambda}$, with C_T a constant[1]. For lower wavelength, this approximation doesn't hold any more, as the shape of the function of the inverse of λ appears to become quadratic. This isn't a problem for the scope of this project which focuses on the simulation of commonly studied wavelengths in astronomy, which coincides with the range of "high wavelengths" where the linear approximation for the attenuation still remains accurate.

Finally, the total local attenuation can be obtained by plugging the different contributions in $T_{local}(t,\lambda) \approx e^{-\tau_{space}(t)\tau_{spectrum}(\lambda)}$ with τ_{space} the formula of the spatial optical depth, in effect obtained by sampling the scene along a ray and thus let free here.

3.2.2 External light emission spectrum

In the following two sections (3.2.2, 3.2.3), we separate the modelling of *external lights* (stars and objects illuminating the scene from remote positions, implicitly contained in the L_s term of the SVRE 2.3) from *internal lights* (the light emitted by the participating medium, for example the ionized gas part of an emission nebula, implicitly contained in the L_e term of the SVRE).

We assume the standard hypothesis that any external body emits light following Planck's radiation law (3.1), thus notably depending on the temperature of the body.

$$I_{Planck,T}(\lambda) = \frac{2hc^2}{\lambda^5} \frac{1}{e^{\frac{hc}{\lambda kT}} - 1} \tag{3.1}$$

We then distinguish two different models for these emission spectra, that can both be used in the following procedure:

- For low wavelengths in the studied spectral domain, **Wien's approximation** of Plank's law [Wie97] is an accurate approximation for emitting bodies, expressed as the product of an inverse polynomial in λ and an exponential in $e^{-A/\lambda}$ (as written in equation 3.2). This approximation remains accurate for $\lambda \ll \frac{hc}{k_B T}$.

$$I_{Wien,T}(\lambda) = \frac{2hc^2}{\lambda^5} e^{-\frac{hc}{\lambda k_B T}} = \frac{A}{\lambda^5} e^{-\frac{B}{\lambda}} \tag{3.2}$$

With A and B two constants in λ

- For higher wavelengths where Wien's approximation doesn't hold true any more, we assume that Plank's law can be projected on a **piece-wise polynomial** function of λ of arbitrary degree.

[1] In section 4, C_T will be fixed to $C_T = 0.56$ to model an average of the dust clouds attenuation plotted on figure 3.2.

3.2.3 Internal light emission spectrum

Light emitted by the participating medium follows the allowed emission rays of ionization of the material making the dust cloud (most often hydrogen, helium, and oxygen [AG11; FT79]). The most accurate model for these rays is thus a set of Dirac distributions, only allowing those specific wavelengths to emit light. The integrals of these Dirac aren't always equal to 1, as they are to be pondered by the intensity of the considered rays.

3.2.4 Sensor sensibility or filter spectrum

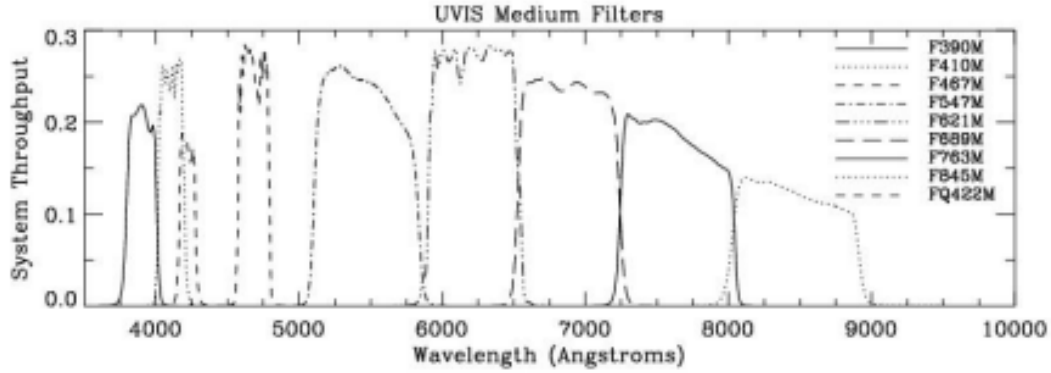


Figure 3.3: Some Hubble filters (61 in total)
Image courtesy: HST documentation

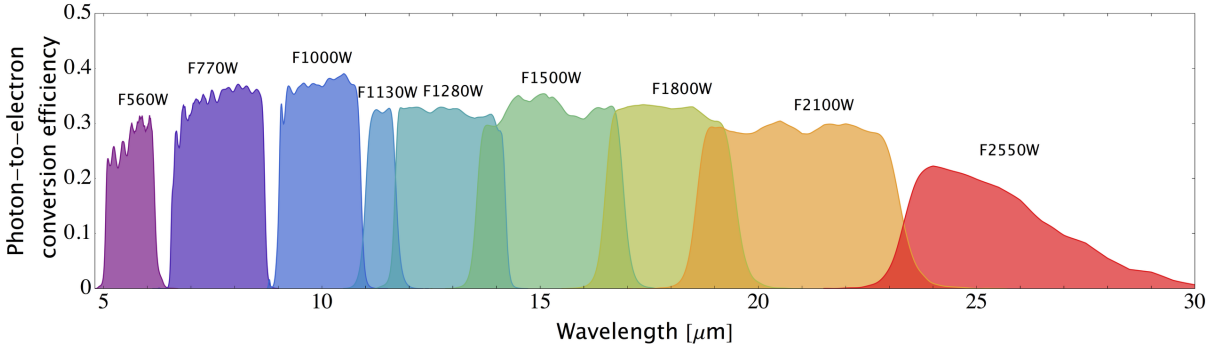


Figure 3.4: Some JWST filters (39 in total)
Image courtesy: JWST documentation

Based on observation of the sensors used in the main telescopes (3.3, 3.4), we assume that the filter spectra can be approximated as low-degree polynomials or piece-wise-defined polynomial functions. In the case of trapezoid-looking sensor functions, the model can for example be separated into three polynomials modelling the left edge, top edge, and right edge. For smoother sensor spectra, a similar approach is used with polynomials of higher degrees.

We stress here the importance of the model of the filter spectra, both as a way of getting an accurate fitting for the actual spectra function, but also with the goal to ensure an explorable

functional space in the end. The optimal renderer that is built should be able to explore the space of all the potential physically accurate filter functions so that an astronomical scene could be seen through the "eyes" of multiple filters and compared in real time. Thus, it makes sense to let freedom in the choice of the spectra models (such as using a family of polynomials of arbitrary degree) in order to explore this space.

3.2.5 Phase function

The phase function is an implicit term in the expression of the incoming light L_s in the SVRE. It describes the intensity at which incoming light will be scattered in every direction. The standard angular phase function is often modelled as Rayleigh scattering, which shape is illustrated in figure 3.5.

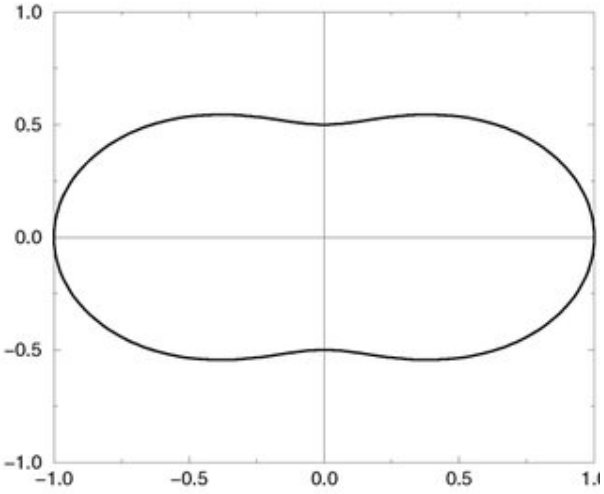


Figure 3.5: Angular aspect of Rayleigh scattering for an incoming ray coming from the left

In this work, we assume that our participating medium's incoming light phase function is **isotropic** (independent of the angle). This hypothesis is one of the strongest in the model and is forced by the proposed method in section 4, which doesn't keep information about the star positions, required for the computation of the phase function. However, this restriction could be relaxed with more work and an extended method: a way to relax this hypothesis and let the phase function free is discussed in the conclusion.

3.2.6 Participating medium coefficients

We assume that the coefficients σ_s, σ_a are separable in the form: $\sigma_i(t,\lambda) = \sigma_{i,1}(t)\sigma_{i,2}(\lambda)$, and the spectral part can be approximated as a polynomial in λ: $\sigma_i(t,\lambda) = \sigma_{i,1}(t)P_{\sigma_i}(\lambda)$.

Full freedom is left on the expression of the spatial separable terms $\sigma_{i,1}(t)$.

3.3 Assumptions summary

Table 3.1 sums up the assumptions discussed in the previous parts.

Name	Symbol	Model[2]	Domain restriction
Sensor sensibility, Filter	$S(\lambda)$	$P_s(\lambda)$	NA
Attenuation	T(t, λ)	$e^{\frac{-\tau(t)C_T}{\lambda}}$	$\lambda > 0.25\,\mu m$
Light emission (external)	$I_s(\lambda)$	$I_{Wien,T}(\lambda)$	$\lambda \ll \frac{hc}{k_B T}$
Light emission (external)	$I_s(\lambda)$	$P_L(\lambda)$	NA
Light emission (internal)	$I_e(\lambda)$	$\sum \delta(\lambda)$	NA
Phase function	$\Phi(w)$	C_Φ	NA
Medium coefficients	$\sigma_i(t,\lambda)$	$\sigma_{i,1}(t)P_{\sigma_i}(\lambda)$	NA

Table 3.1: Assumptions summary

[2] P_x are polynomials, C_x are constants

— 4 —

Real-time spectral integration

4.1 Analytical integration of the SVRE

Given the a-prioris discussed in part 3 and summed up in table 3.1, we show below that the SVRE 2.3 can be expressed as a product of polynomials in λ and exponentials of the form $e^{-a(t)/\lambda}$. By only focusing on the spectral aspect (with t a position on the ray), the general equation $F(t) = \int_\lambda f(t,\lambda)d\lambda$ can now be expressed in a more compact way[1]:

$$\begin{aligned} F(t) &= \int_\lambda P(\lambda)e^{-g(t)/\lambda}d\lambda \\ &= \sum_{n=0}^{N} P_n \int_\lambda \lambda^n e^{-g(t)/\lambda}d\lambda \end{aligned}$$

with:

$$\begin{aligned} P &= P_S P_L P_\sigma \\ N &= \deg(P) \\ g(t) &= (\tau(t) + \tau_s(t))C_T \end{aligned}$$

We prove in the appendix (6.1) that this form of the SVRE achieves to possess the **integrability** trait that was discussed in the previous section: All functions of this model can be analytically computed over the wavelength. Instead of relying on spectral discretization for the numerical integration, it is now possible to compute the exact value of the spectral integral (for the chosen model of functions). Moreover, this form for the SVRE allows the expression of the definite integral as a recurrence relation, which is relevant to the following implementation part.

4.1.1 Algorithm and implementation

Using the recurrence defined in 6.1, one bound of the spectral integral of $f(t,\lambda)$ can be computed in $\mathcal{O}(N)$ operations, with N the degree of the spectra product polynomial P. Algorithm

[1] For the sake of simplicity, we consider the case where the emission spectrum is modelled as a polynomial in λ, and not according to Wien's approximation as presented in section 3.2.2. Choosing to do so would only require extending the polynomial P to take into account the potential five inverse powers of lambdas introduced by Wien's approximation. This is completely doable (and the model is still integrable in a recurrence as presented in appendix 6.1), but it introduces over-complexification that is not needed in order to understand the gist of the model and integration scheme.

Algorithm 1 Analytical integration of the spectral integrand

function SPECTRALINTEGRATION($P, A, \lambda_0, \lambda_1$) ▷ P polynomial, A = g(t)
  return INTEGRALBOUND(P, A, λ_1) - INTEGRALBOUND(P, A, λ_0)
end function

function INTEGRALBOUND(P, A, λ_i)
  $x \leftarrow \exp(-A/\lambda_i)$
  $\alpha_n \leftarrow -E_1(A/\lambda_i)$ ▷ Exponential integral 1
  $\beta_n \leftarrow \lambda_i \cdot x$
  res $\leftarrow$ COMPUTETERM($\alpha_n, \beta_n, A, 0, P[0]$)
  for $n \leftarrow 1 ... \|P\|$ **do**
    NEXTALPHABETA($\lambda_i, n, A, \alpha_n, \beta_n$)
    res $\leftarrow$ res +COMPUTETERM($\alpha_n, \beta_n, A, n, P[n]$)
  end for
  return res
end function

function COMPUTETERM($\alpha_n, \beta_n, A, n, c$)
  return $(\frac{\alpha_n A}{n+1} + \beta_n) \cdot c$
end function

function NEXTALPHABETA($\lambda_i, n, A, \alpha_n, \beta_n$)
  $\alpha_n \leftarrow -\beta_n - \frac{A}{n} \cdot \alpha_n$
  $\beta_n \leftarrow \beta_n \cdot \lambda_i \cdot \frac{n}{n+1}$
end function

1 shows a pseudo-code for the integration in $\mathcal{O}(N)$. Thus, the full spectro-spatial integral can be computed in $\mathcal{O}(N \cdot M)$ with M the spatial resolution for a light ray in the scene. The naive rendering scheme for a nebulae scene would be to compute a spatial numerical integral for each ray shot by the camera, sampling the scene at regular intervals. For each of these samples, the spectral integral can then be computed analytically.

This nullifies the need of relying on a spectral discretization for doing a double numerical integration, but isn't by itself an efficient enough solution to the original performance problem. Indeed, efficient numerical integration methods such as Gaussian quadrature integration will be able to obtain results as precise as those obtained with our analytical integration in the same complexity, for a similar polynomial fitted function of fixed degree.

4.2 Acceleration by projection

The formula obtained in section 4.1 for the spectral integrand can be transformed in a way that will reduce the operational load on run-time for the computation of the pixel intensity. We start by defining mathematical tools and spaces that will be used in order to rearrange the spectral integral terms in a more convenient manner.

With $\mathbb{D} \subset \mathbb{R}^+$ a domain, let $\mathbb{V}$ be the vector space generated by the family $\mathbb{F} = \{ f_a \mid a \in \mathbb{R}^+ \}$, with f_a defined as:

$$\begin{aligned} f_a \colon \mathbb{D} &\longrightarrow \mathbb{R} \\ \lambda &\longmapsto e^{-a/\lambda} \end{aligned}$$

$\mathbb{F}$ is also a linearly independent family, as no exponential of arbitrary argument can be expressed as a linear combination of exponentials of different arguments. As it is also spanning over $\mathbb{V}$, $\mathbb{F}$ is a basis of $\mathbb{V}$.

There is an analogy between the spectro-spatial integral that we want to solve analytically and the vector space $\mathbb{V}$. The numerical integration of the spatial integrand for each ray shot in the scene can actually be expressed as an element of $\mathbb{V}$, and thus decomposed as a linear combination of elements of $\mathbb{F}$. This is shown in equation 4.1, where $h(t_j, w)$ can be seen as a linear coefficient for each element $f_{g(t_j)}$ of $\mathbb{F}$, the full sum giving an element of $\mathbb{V}$.

$$\begin{aligned} &\int_{\lambda \in \mathbb{D}} P(\lambda) \int_{t=0}^{E} h(t,w) e^{-g(t)/\lambda} dt d\lambda \\ &\approx \int_{\lambda \in \mathbb{D}} P(\lambda) \sum_{j=0}^{N} h(t_j,w) e^{-g(t_j)/\lambda} \Delta j d\lambda \\ &= \int_{\lambda \in \mathbb{D}} P(\lambda) \sum_{j=0}^{N} h(t_j,w) \boxed{f_{g(t_j)}(\lambda)} \Delta j d\lambda \end{aligned} \tag{4.1}$$

To compute the spectral integral of an element of $\mathbb{V}$, the naive method is to keep the decomposition on the basis of $\mathbb{F}$ in order to compute each of these spectral integrals that are known analytically. However, in the most general case, there will be as much element of $\mathbb{F}$ that will take part in the sum as the spatial resolution of the model (in the case of equation 4.1, N terms). In other words, a spectral integral must be computed for every spatial step in the scene.

In order to reduce the number of computations needed to compute the spectral integral, we explore an approach to reduce the cardinality of the basis B used to decompose the spatial integral of $\mathbb{V}$, while also keeping the cardinality of each element of the basis as low as possible. The second part is crucial in order to reduce the number of computations. The family $\mathbb{F}$ presented before has a maximal cardinality (card($\mathbb{F}$) = card($\mathbb{R}^+$)), but a minimal *element-wise* cardinality: each element of $\mathbb{F}$ is made of a single component f_a. On the other hand, the result of a Principal-Component Analysis would output the complete opposite of this basis: the PCA basis would have minimal cardinality (given the precision that is to be reached, PCA will output a few vectors containing most of the information of $\mathbb{V}$). However, the PCA basis will have maximal *element-wise* cardinality, as each element of the PCA basis will be a linear combination of all elements of $\mathbb{F}$. The idea here is to try to find a middle ground between the two bases.

Following on equation 4.1, the goal is to find the best possible basis $B = \{v_i | i \in \{0...N_v\}\}$ so that the spatial integral can be projected and resolved in the fewest operations. This generic

process is shown in equation 4.2, which shows a change of basis, from $B = \mathbb{F}$ to an arbitrary orthonormal basis that will be determined in the following part[2].

$$\begin{aligned}
&\int_{\lambda=\lambda_1}^{\lambda_2}\int_{t=0}^{E} f(\lambda)e^{g(t)/\lambda}h(t,w)dtd\lambda \\
&\approx \int_{\lambda=\lambda_1}^{\lambda_2} f(\lambda)\sum_{j=0}^{N}\boxed{f_{g(t_j)}(\lambda)}\,h(t_j,w)\Delta j d\lambda \\
&\approx \int_{\lambda=\lambda_1}^{\lambda_2} f(\lambda)\sum_{i}^{N_v} v_i(\lambda)\left[\sum_{j}^{N} h(t_j,w) < f_{g(t_j)}, v_i > \Delta j\right] d\lambda
\end{aligned} \tag{4.2}$$

By choosing a good enough basis, we can make another approximation by ordering it by the magnitude and truncating the linear combination of the basis vectors to project each vector on the approximate subspace. This is similar to the approximation of only keeping the first N_v vectors of a PCA basis in order to cover up to N_p percentage of information on the vector space.

4.2.1 Basis determination

The approach defined in the previous part is only relevant and usable if the model chosen for the basis has a good enough convergence in terms of coverage on the vector space.

In the most generic case, with an infinite family of non-co-linear functions (as in our case), the precision of the projection over a restricted basis can greatly vary, as full coverage of the information can only be assured by a basis of infinite cardinality. In the following part, we discuss a way to obtain the most precise basis possible for our specific vector space $\mathbb{V}$, and how this choice for the function models makes it possible to still get accurate results with a low cardinality (truncated) basis.

Function analysis

We quickly analyse the function shape of elements of $\mathbb{F}$ which will be projected on the basis. Knowing the range of elements it can reach is linked with the error rate between projections of elements.
The function that is considered here is, for all $\{a \in R^+\}$:

$$f_a(x) = e^{-a/x} : \{x \in R^+\} \longrightarrow [0,1]$$

Two relevant function limits are:

$$\begin{gathered}
f_a(x) \xrightarrow{a\to 0} \mathbb{1} \\
\text{and} \\
f_a(x) \xrightarrow{a\to\infty} \mathbb{0}
\end{gathered} \tag{4.3}$$

A good basis should be able to model these two limits that are easily attainable in our model: $a \to 0$ corresponds to an absence of density in the optical path, and $a \to \infty$ corresponds to an extremely high density.

[2] $< f_{g(t_j)}, v_i >$ is the scalar product of $e^{g(t_j)/\lambda}$ and v_i.

Scalar product and induced norm

In the following part, the usual scalar product between functions is used over the same spectral domain $\mathbb{D}$ as the definition domain of each $f_a \in \mathbb{F}$. The norm used to compute distances and error analysis between functions is the norm L_2. Any notation $\|a\|$ in the next sections of the report refers to the second norm $\|a\|_2$, the most usual norm for functions. Moreover, as the goal is to ensure that the integral of the projection is similar to the integral of the element projected, it makes more sense to look at the norm L_2 (in effect an integration over the spectral domain) instead of other norms (like the maximum or mean of a function). Mentions of the distance between two elements are expressed as the norm of the difference between those elements.

$$\begin{aligned} &\forall a,b \in \mathbb{V} \\ &\qquad < a,b > \;=\; \int_{\mathbb{D}} a(x)b(x)dx \\ &\qquad \|a\|_2 \;= < a,a >^{1/2} \\ &\qquad d(a,b) \;=\; \|a-b\| \end{aligned} \tag{4.4}$$

4.2.2 Obtaining a precise basis

To ensure the generation of a precise basis which can be truncated and still allow for an accurate approximation of the projected elements, an iterative approach can be used in order to choose the best next possible element to add to the basis. This constraint can be seen as choosing the element that will minimize the norm of the difference between an expected element of $\mathbb{F}$ and its projection, and this for every element of $\mathbb{F}$. This can be expressed as minimizing the following equation by choosing the best possible basis B:

$$\sum_{f \in \mathbb{F}} \|f - \text{proj}_B(f)\| \tag{4.5}$$

Application

This problem can generally be solved following a gradient descent to minimize the previous function and obtain the best possible basis. In the following application, this is approximated by a greedy approach in $\mathcal{O}(card(B) \cdot card(\mathbb{F}_c))$, with $\mathbb{F}_c$ a discretized subset of the infinite family $\mathbb{F}$. The basis B is constructed element by element, choosing the best possible element b at each step that will minimize $\sum_{f \in \mathbb{F}} \|f - \text{proj}_{B \cup b}(f)\|$.

In the following part, the spectral domain is fixed to $D = [0\,\mu\text{m}, 1.0\,\mu\text{m}]$. A typical astrophysical filter has a range length between 0.1 and 1 µm, and the minimal wavelengths observed by Hubble and JWST are around .2 µm while the maximal values are around 30 µm (see figures 3.3 and 3.4). The "hardest" part to define our projection is around low wavelengths, as it is where small changes in the optical path will have greater repercussions on the spectral function. This is why for generality, the following application will focus on this "hard" region, while more effective results for higher wavelength are discussed in the subsection 4.2.2.

With this fixed spectral domain, results are shown below with a first step where the basis is initially generated with the spectral function associated with the minimal optical path value of our model (a = 0). This corresponds to the first limit of the spectral function shown in equation

4.3, $f_a \xrightarrow{a \to 0} \mathbb{1}$. This ensures that the lowest densities can be modelled by our basis, as well as the extremely high densities (the projection of $f_a(x) \xrightarrow{a \to \infty} \mathbb{0}$ over a basis containing $\mathbb{1}$ can trivially be $\mathbb{0}$ by nullifying the projection coefficient).

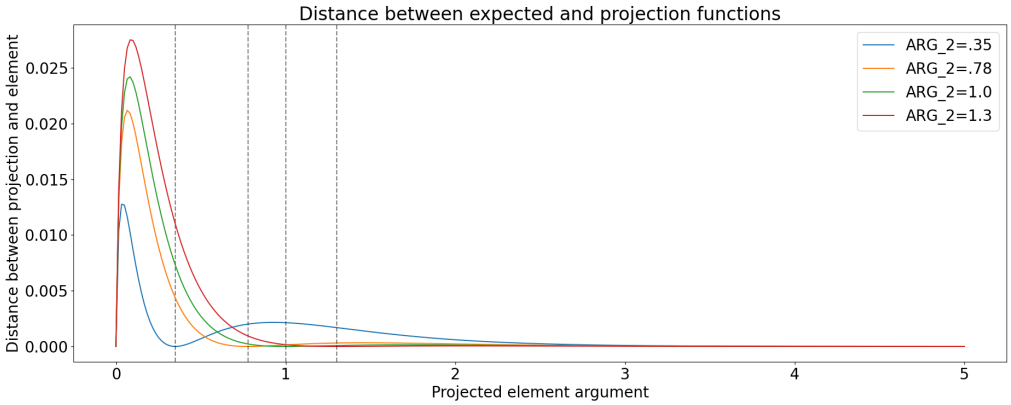


Figure 4.1: Visualization of various error functions for arbitrary choices of the second basis member
The second basis element has a form: $e^{-ARG\_2/x}$

The graph in figure 4.1 shows the error function for four different bases built on arbitrary candidates for the second element ($b_1 = \mathbb{1}$, $b_2 = \{e^{-.32/\lambda}, e^{-.78/\lambda}, e^{-1.0/\lambda}, e^{-1.3/\lambda}\}$). The y-axis ranges from a distance of 0 (meaning that an element and its projection are the same considering our distance metric) to 1 (meaning that an element and its projection are the farthest they can be (here $d(\mathbb{1}, \mathbb{0}) = \sqrt{\int_D (\mathbb{1} - \mathbb{0})^2 dx} = 1$)).

The best candidate for the basis is the one that will minimize the integral of the error function. A continuous plot of the areas of the error functions is thus shown in the first graph of 4.2. The minimization step for a **second basis element** gives a spectral function corresponding to an optical path of 0.45 ($b_2 = e^{-.45/x}$), as illustrated by the minimum in the function. The error function between all elements of $\mathbb{F}$ and their projection on this theoretically optimal basis $B = \{\mathbb{1}, e^{-.45/x}\}$ is then plotted on the second figure of 4.2. As expected, there are two local minima at 0 on the two optical path values of our basis (0 and .45), and another one at the limit $a \to \infty$ where the function converges to $\mathbb{0}$. These correspond to points that are fully expressible on this basis, while the non-zero values of the curve correspond to elements that will lose some of their information after projection. There are also two local maxima, one at low densities ($\approx .1$), and one at medium densities (≈ 1). This means that elements sampled in the 3D scene with an optical length close to these local maxima will lose part of their information when projected on the selected basis. On the other hand, elements with high optical length will lose almost no information, meaning that their spectral integral can be fully computed by their projection on B.

It can be noted that while the criterion used to decide the next best basis element 4.5 is an "objective" criterion, as it achieves to get the basis that will get a minimal distance between a plane and its projection, it is also possible to tweak it and focus on specific areas of the argument space. The second graph of figure 4.2 shows that the optimal 2-element basis has a local

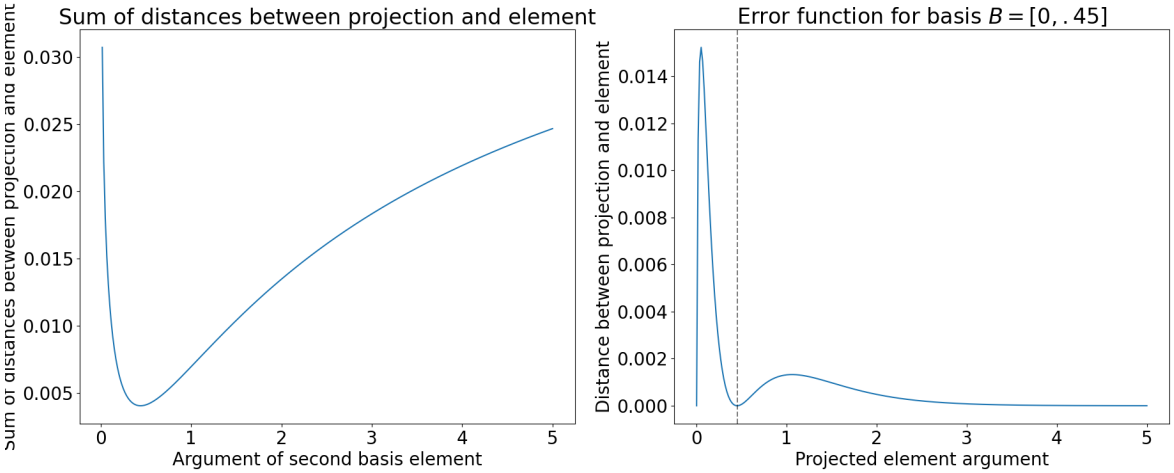


Figure 4.2: Visualization of the first minimization step
The argument for the second element of the basis being constructed that minimizes the error between all elements and their projection is obtained by minimizing the curve on graph 1, for a value of (a = .45).

maximum for elements of a very low argument. To better fit a basis in a scene containing a lot of elements with that low arguments, it would be possible to tweak the second basis element towards this local maxima. This would mean that the entire sum of the error function would increase, but that the low densities would be better modelled. We chose here to stay with an "optimal" basis without tweaking it to stay general.

At this second minimization step, we thus get a two-element basis where the maximum error between any element and its projection is at ≈ 0.015. The process can then be iterated multiple times from this *extended basis* $B = \{\mathbb{1}, e^{-.45/x}\}$ until the expected precision is obtained, and when the projections get precise enough in comparison with the actual function vector.

Figure 4.3 illustrates the decrease of the error when the basis is extended, which follows a logarithmic decrease. For each element that is added to the basis, the error is divided by 10. It is to note that this goes in pair with a linear increase in the run-time complexity of the algorithm, as it will be discussed in the implementation part. Each element of the basis indeed needs to be integrated analytically once on the sensor.

Higher spectra domain - Telescope wavelengths

Considering the wavelengths of interest for Hubble and JWST, the wavelength ranges are around $0.1\,\mu m$ to $30\,\mu m$, and the length of these ranges often does not exceed 1 micron. With these parameters, we get significantly better results than when using the previous wavelength range, as the most difficult part to integrate of the model function is located near low wavelengths. Likewise, a spectral domain spanning over a smaller range of wavelengths will enable a basis of similar cardinality to generate better results. Examples are illustrated in the figure 4.4, for a spectral domain of $D = [1\,\mu m, 10\,\mu m]$. The error is divided by 100 for each component added to the basis, instead of 10 for a range of lower wavelengths. The same precision can be attained in half the number of basis component f_a with higher wavelengths, even though the

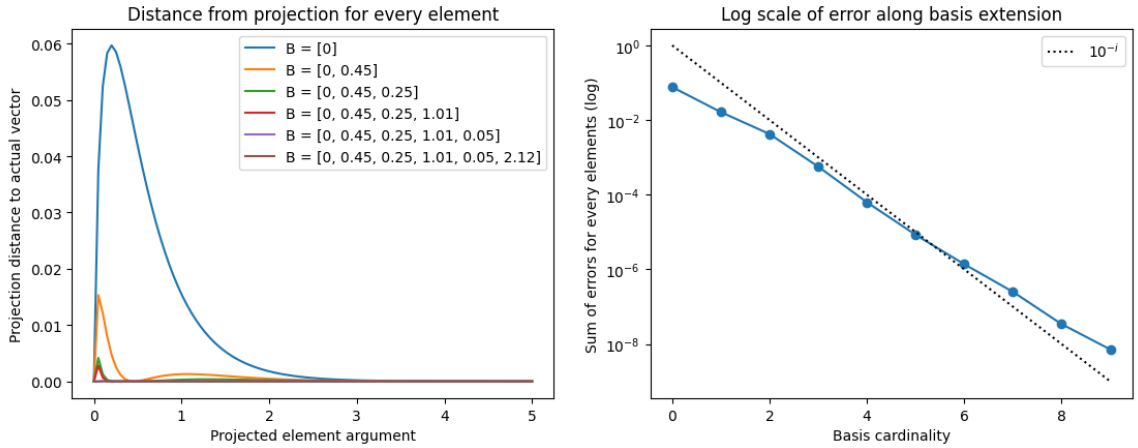


Figure 4.3: Evolution of error along basis extension
The error between all elements and their projection is divided by 10 for each element added to the basis.
For the first plot, the elements of the basis B are represented in their argument form ($B[i] = e^{-b_i/x}$)

spectral range length is 10 times bigger than the low wavelength range.

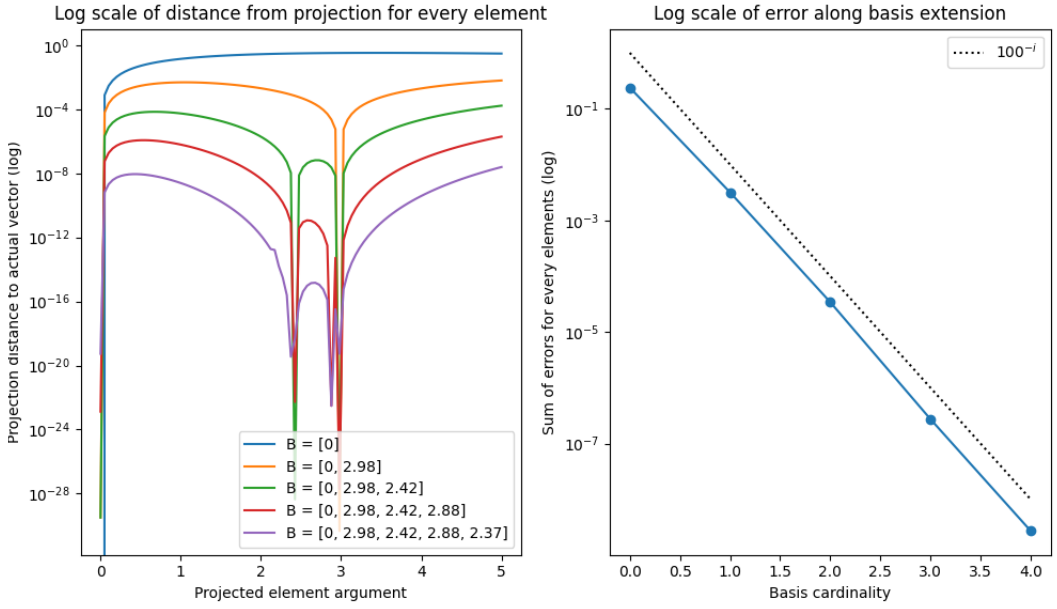


Figure 4.4: Increased projection precision when considering higher wavelength ranges
The error between all elements and their projection is divided by 100 for each element added to the basis. This is 10 times better than the results obtained for lower wavelengths.

4.2.3 Implementation and error analysis

The SVRE is computed over a scene containing spectral information (that was modelled in the previous steps), but also information about the local density at each point of the scene (used for example to compute the optical depth). The simulation of an accurate density model is outside this project's scope and has been partially covered in previous works as part of the underlying ANR [Anr]. The density structure for our comparison test scenes were built using Fractional Brownian Motion (fBm) applied to Perlin noise, but could be replaced by any arbitrary model for a nebulae density. We chose the combination of fBm and Perlin noise, as by tweaking the set of parameters (amplitude, frequency, octaves) we can obtain multiple kinds of spatial density distribution to test our model. This is important, as the previous section showed that the results can vary based on the distribution of the density in the scene, with the higher relative errors located around low densities for small-cardinality basis-based projection.

Basis generation

Algorithm 2 Basis generation

```
function GENERATEBASIS(N, a_0, a_1)
    for i ← 0...N do
        b ← FINDBESTMONOME(B, a_0, a_1) ▷ Gradient descent, find best element to add to B
        b ← GRAMSCHMIDT(B, b)           ▷ Generate orthonormal vector to B
        B ← B ∪ b
    end for
    return B
end function
```

Algorithm 2 illustrates a pseudo-code for the generation of an accurate basis, according to the steps discussed in section 4.2.1. This function is to be called only once at the start of the program, and the generated basis can then be passed to every other rendering procedure without needing to be re-computed at every frame. This step is relatively costly, as the implicit `findBestMonome` function iterates over all the possibles elements from the family $\mathbb{F}$ in order to select the element that, once added to the basis, will minimize the sum of errors for all elements in the vector family. The complexity of this part is of $\mathcal{O}(card(\mathbb{F}_c) \cdot N_B)$. N_b is the cardinality of the basis that is to be generated, and $card(\mathbb{F}_c)$ is the cardinality of a discretized subset of the infinite family $\mathbb{F}$. The greater the cardinality of B will be, the more accurate the results of the projections will be in the following steps, but the computational load will be linked with it. On the other hand, the cardinality of the subset $\mathbb{F}$ only contributes to this basis generation step. There is no run-time disadvantage to have as much spatial resolution in this discretized family for this basis generation step, as it will ensure greater precision on the basis generation.

Pre-computations

One benefit of the projection method is that we can pre-compute the shadows (external lights optical depth) and sample them in constant time during execution. Algorithm 3 illustrates the pre-computation of the shadows, while algorithm 4 shows how to use these values on run-time in order to compute the projected spatial-spectral integral. In practice, the contributions of each

Algorithm 3 Pre-computation

function PRECOMPUTE(v, b) ▷ v is a voxel of the discretized 3D space
  $\rho \leftarrow$ LOCALDENSITY(v)
  for $L_i \leftarrow 0...\|\text{lights}\|$ **do**
    $l \leftarrow l..$OPTICALPATHTOLIGHTSOURCE(v, lights$[L_i]$)
  end for
  STORE(v, l) ▷ Storing optical depth l in voxel v
end function

Algorithm 4 Run time integration

global variables
  P, Spectral polynomial
  λ_0, λ_1, spectral integration bounds
  B, basis
end global variables

function COMPUTEINTENSITY(p) ▷ p is a pixel on the camera array
  $r \leftarrow$ GETRAY(p)
  for $j \leftarrow 0...N$ **do** ▷ Spatial numerical integration
    $p \leftarrow$ GETPOINTALONGRAY(r, j)
    $l \leftarrow$ LOAD(p) ▷ Loading light optical path at position p
    $x \leftarrow$ PROJECT($\rho + l, b$) ▷ Projecting the optical path $\rho + l$ on basis b
    $e \leftarrow$ PROPAGATE(p, j, x, e) ▷ Propagate the projected contribution on the basis through the light ray
  end for
  for $i \leftarrow 0...|b|$ **do** ▷ Analytical integration of each member of b
    $a \leftarrow$ ARG($b[i]$) ▷ Get argument of basis element i
    $c \leftarrow$ PROJECTIONCOEF(i, b, p) ▷ Compute projection coefficient for the ith term
    $v \leftarrow v + c \cdot$ SPECTRALINTEGRATION($P, a, \lambda_0, \lambda_1$)
  end for
  STORE(v, p) ▷ Storing intensity value in pixel p
end function

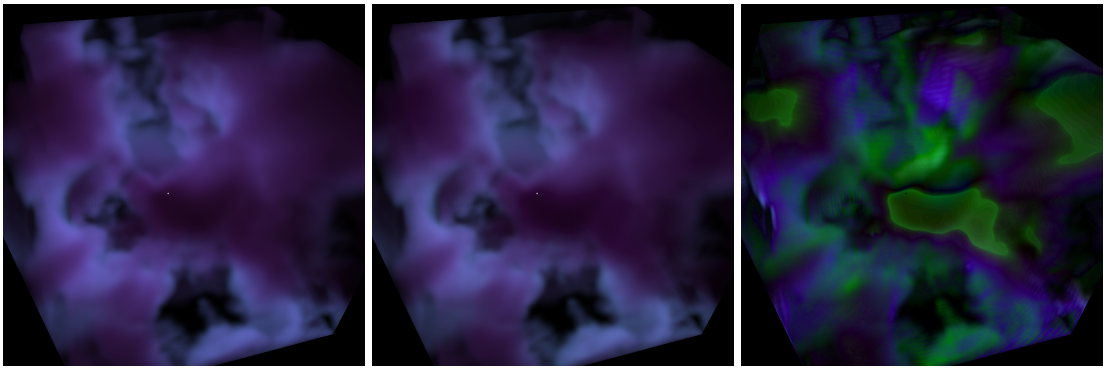
element of the discretized space (in 3D, a voxel of fixed length) are computed and stored in a data structure (in GLSL, a `texture3D`) that can be sampled at run-time. The size of this texture needs only to be the same as the input density texture, containing information about the local density number for each voxel of the discretized space.

Both algorithms presented are designed to fully take into account the benefits of GPU programming, by being completely parallelizable for each voxel in the scene. It is to note that a choice of implementation for the `load` function in algorithm 4 must be made. The approximation that the texture filled in the pre-computation step can be sampled continuously (and each value computed with trilinear interpolation) must be made in case one wants to use the full potential of most GPU implementations (the trilinear interpolation being optimized for most GPU cards). In the other case, this `load` function must be implemented to compute or approximate

the value at any point in the scene or restrain the ray to follow the voxels of the discretized scene.

Error analysis

By comparing the results with a ground truth computed with a double numerical integration (spatial and spectral), the error stays below 6% for any pixel on a computed scene as shown in figure 4.5 for a projection method with an optimal basis of cardinality only 3. This means that only three spectral integrations are done for the projection method (with the goal of retrieving most of the information of the function vectors that were projected on the basis), while the full integration method would have had to do as many spectral integrations as the spatial resolution. We achieve to model in a visually correct way both the low and high densities in our nebulae scenes, with no significant color artefacts or discontinuities.



(a) (b) (c)

Figure 4.5: Comparison between numerically integrated renders and renders obtained by a projection on a 3-element basis.
(a) Double numerical integration (b) Projection approximation (c) Absolute error $\times 16$

Complexity

We observe in effect a great increase in performance when using the projection method instead of the numerical one. The pre-computation is solved in a fixed time and helps reduce the operational load of the run-time integration. Thus, only $card(B)$ analytical integrations are done at run-time. Part of the time complexity at run-time is converted to space complexity. As discussed previously, we fill a voxel texture with the pre-computations, in effect taking N_{space} space for each ray. Table 4.1 references the time complexities for each method (separating the pre-computations and the run-time integration for the projection method). Table 4.2 illustrates the performance results when computing a particular scene and comparing the methods.

The projection scheme used in this section doesn't rely on the expression of the spectra polynomial (product of the emission and filter spectra models), which can thus remain free and modified freely at run-time. In effect, one can explore any scene in real time and slide or modify the filter functions used for the spectral integration, as it would be the case when using a real telescope.

Method	Complexity
Double Netwon integration	$\mathcal{O}(N_{space} \cdot (M_{wavelength} + N_{space}))$
Naive analytical spectral integration	$\mathcal{O}(N_{space} \cdot (Deg(P) + N_{space}))$
Pre-process	$\mathcal{O}(N_{space} \cdot N_{space})$
Run-time integration	$\mathcal{O}(N_{space} \cdot Card_{basis} + Deg(P) \cdot Card_{basis})$

Table 4.1: Comparison of algorithm complexity between numerical integration of the SVRE and projection approximation

Method	Avg Time per frame[3]
Newton	260ms
Naive analytical computation	160ms
Run-time projection	30ms

Table 4.2: Comparison of performance (time to compute each frame) between numerical integration of the SVRE and projection approximation on scene A (figure 4.5) for a similar precision level

4.3 Multiple emission spectra in the scene: Light-Density projection

A typical nebulae scene can contain a huge number of light sources, as the dust cloud can cover multiple solar systems in its span, and thus multiple stars. The default hypothesis is to state that each of these stars can have a different emission spectrum, and thus interact differently with the participating medium.

In the naive integration scheme, the full rendering pipeline needs to be computed once for each emission spectrum (and thus each star), as the polynomial in terms of lambda will differ for each light source.

We discuss here a solution to address this problem, making it possible to compute a scene with multiple light sources and multiple spectra at a reduced cost.

4.3.1 Bi-dimensional projection

Section 4.2 presents a way to generate a basis of functions for expressing the optical paths of the elements of volume.

We can extend this method to a multidimensional basis, making the basis dependent on both the optical path and the spectrum of the light. Our light spectrum model is taken from the black body equations, which reduces the dimensionality of the possible emission spectra to one (the temperature of the star directly infers the emission spectra). Thus, the basis to be generated will depend on two variables: the optical length and the temperature of the star. Function 4.6 shows analogous vector spaces to the one presented in part 4.2.

With $\mathbb{D} \subset \mathbb{R}$ a domain, let $\mathbb{V}$ be the vector space generated by the family $\mathbb{F} = \{f_{a,T} | (a,T) \in$

[3] N=100,M=20,C_b=3,Deg(P)=5,GPU=Nvidia GTX 1080 Ti, scene A

$(\mathbb{R}^+)^2\}$, with $f_{a,T}$ defined as:

$$\begin{aligned} f_{a,T} : \mathbb{D} &\longrightarrow \mathbb{R} \\ \lambda &\longmapsto P_T(\lambda)e^{-a/\lambda} \end{aligned} \tag{4.6}$$

In effect, the generation of the basis follows a similar approach as the one described in the previous section. The only difference is that the variable space for the gradient descent is two-dimensional, but all the operators remain the same.

Application

Algorithm 5 Pre-computation of projections

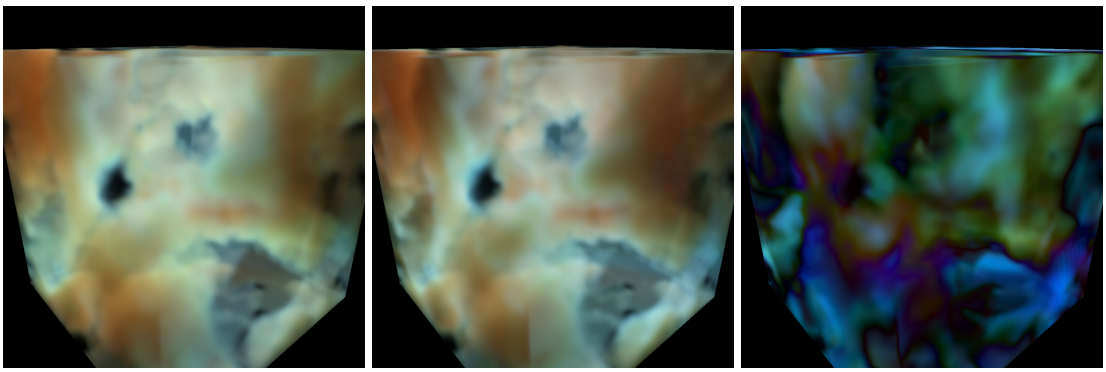
```
function PRECOMPUTE(v, b)          ▷ v is a voxel of the discretized 3D space, b a basis
    for L_i ← 0...||lights|| do
        l ← OPTICALPATHTOLIGHTSOURCE(v, lights[L_i])
        p ← p + PROJECT(l, lights[L_i], b)          ▷ Projection on basis b
    end for
    STORE(v, p)                    ▷ Storing projection in voxel v
end function
```

For verification purposes, the emission polynomials for the family $\mathbb{F}$ are generated as linear interpolations between two "edge" polynomials: $P_0(x) = \frac{(x+1)(x)}{2}$ and $P_1(x) = \frac{(x-2)(x-1)}{2}$, $P_T(x) = (1-T)P_0(x) + TP_1(x)$ with $T \in [0,1]$. This range covers most of the possible aspects for emission polynomials, with functions associated with low temperature emitting most photons in the high wavelengths, and functions associated with high temperature emitting most photons in the low wavelengths. For the rest of the parameters, we follow the same decisions as in the previous section 4.2, with the extension for the two-dimensional basis when needed.

For the application, a basis with cardinality 4 is chosen in order to describe the full scene. The same algorithm as described in the previous section 4.2.2 to generate the best possible basis is applied, and outputs four basis vectors to project each element of the scene. The complexity of this first step is significantly greater than for a basis only relying on the optical path, as the function to minimize is now a 2D surface and no longer a 1D function. Still, a gradient descent approach can be applied and will give an optimal basis according to our criteria.

Figure 4.6 illustrates the results that are obtained when computing a scene with 6 lights of different spectrums. In effect, all of these light contributions are being projected onto the four basis elements which are spectrally integrated at the end of the pipeline. The algorithm applied is extremely similar to both algorithms 3 and 4 presented in the previous section, the only difference being the nature of the basis and the actual implementation of the `project` method, which now takes into account the spectrum of the emitting light sources.

The results obtained appear decent, with low visual differences between the full double integration method and the projection one. We note that for a basis of similar length (3 and 4), we get worse results with this spatial-spectral basis in comparison with the previous one. This can be explained because the basis is now covering a two-dimensional space, and information about both the optical depth and the spectral polynomial needs to be projected. It is still possible



(a) (b) (c)

Figure 4.6: Comparison between numerically integrated renders and renders obtained by spectral-spatial projection on a 4-element basis
(a) Double numerical integral (b) Projection approximation (c) Absolute error $\times 6$

to decrease this error by augmenting the number of elements of the basis, but this also goes along with an augmentation of the run-time computational load.

Complexity

Table 4.3 illustrates the complexity of the algorithm in comparison with a double integration for a variable number of lights (L_{lights}). The space complexity is the same as in the previous scheme. In effect, while the pre-integration step takes more time as the projection of each light contribution must be computed, the run-time integration isn't impacted at all. This means that the scene can include any number of lights and spectra, and the complexity will remain the same. Table 4.4 illustrates the program performances for computations on scene B (4.6).

Method	**Complexity**
Double Netwon integration	$\mathcal{O}(N_{space} \cdot (M_{wavelength} + N_{space}) \cdot L_{lights})$
Projection (pre-process)	$\mathcal{O}(N_{space} \cdot (Deg(P) \cdot Card_{basis} + N_{space}) \cdot L_{lights})$
Run-time integration	$\mathcal{O}(N_{space} \cdot Card_{basis} + Deg(P) \cdot Card_{basis})$

Table 4.3: Comparison of algorithm complexity between numerical integration of the SVRE and spectral-spatial projection approximation
The run-time integration performance of the SVRE isn't impacted by the number of lights in the scene

Method	**Avg Time per frame**[4]
Newton	2500ms
Run-time projection	30ms

Table 4.4: Comparison of performance (time to compute each frame) between numerical integration of the SVRE and projection approximation for scene B (figure 4.6).

[4] $L_{lights} = 6$, N=100,M=20,C_b=4,Deg(P)=5, GPU= Nvidia GTX 1080 Ti, scene B

4.4 Acceleration by pre-integration

In computer rendering, *pre-integration* [EKE01] is a method used to precompute parts of the results of the rendering equation (in this case the SVRE) before runtime, and to use these pre-computed results during execution to reduce the real-time computational load by only having to sum small elements during the spatial integration part.

While the generic methods for *pre-integration* cannot be applied in the spectral case where the intervals have been inverted, it seems tempting to add a few approximations in our model to be able to apply the same pre-computations and reduce the operational load. A few approximations were made in the following part in order to explore this possibility, but no good results were reached with them in comparison with the projection 4.2 method. The previous method is thus to be preferred, as it gets better results in fewer computations and better scalability. We still chose to present this approximation below, as it seems to be the more "naive" way to approximate the literature *pre-integration*.

As in the previous part, we start from the spectral integral form of the SVRE 2.3, and we make several approximations in order to express it under a convenient form, as shown in equation 4.8:

$$\begin{aligned} &\int_{\lambda=\lambda_1}^{\lambda_2} f(g(t),\lambda)\,d\lambda \\ &= \sum_{k=0}^{N} P_k \int_{\lambda=\lambda_1}^{\lambda_2} f_k(g(t),\lambda)d\lambda \end{aligned} \tag{4.7}$$

$$\sum_{k=0}^{N} P_k \int_{\lambda=\lambda_1}^{\lambda_2} f_k(g(t),\lambda)d\lambda \tag{4.8a}$$

$$\approx \sum_{k=0}^{N} P_k C_k(\lambda_1,\lambda_2)\exp\left(-[g(t)]\cdot\varepsilon(\lambda_1,\lambda_2)\right) \tag{4.8b}$$

$$= \sum_{k=0}^{N} P_k C_k(\lambda_1,\lambda_2)\exp\left(-[\tau(t)+\tau_s(t)]\cdot\varepsilon(\lambda_1,\lambda_2)\right) \tag{4.8c}$$

$$\approx \sum_{k=0}^{N} P_k C_k(\lambda_1,\lambda_2)\cdot \tag{4.8d}$$

$$\exp\left(-[\sum_{j=0}^{t}\sigma_t(j)\Delta j+\tau_s(t)]\cdot\varepsilon(\lambda_1,\lambda_2)\right) \tag{4.8e}$$

$$= \left[\sum_{k=0}^{N} P_k C_k(\lambda_1,\lambda_2)\exp\left(-[\sigma_t(0)\Delta t]\cdot\varepsilon(\lambda_1,\lambda_2)\right)\right]_1 \cdot \tag{4.8f}$$

$$\left[\prod_{i=1}^{t}\exp\left(-\sigma_t(i)\Delta_j\cdot\varepsilon(\lambda_1,\lambda_2)\right)\right]_2 \left[\exp(-\tau_s(t)\cdot\varepsilon(\lambda_1,\lambda_2))\right]_3 \tag{4.8g}$$

with:

$$C_k(\lambda_1,\lambda_2) = \frac{\lambda_2^k-\lambda_1^k}{k} \tag{4.9}$$

We discuss below the two approximations made in demonstration 4.8:

- 4.8b : The spectral integral is approximated as an exponential in $g(t)\varepsilon(\lambda_1,\lambda_2,k)$ and pondered by a term $C_k(\lambda_1,\lambda_2)$. This approximation makes sense by looking at arbitrary analytical results of the spectral integral, assuming that the last term in $E_1(g(t)/\lambda)$ is negligible. The remaining terms can thus be put in this form, two significant results being that the two limits of the function (in $g(t) \longrightarrow 0$ and $g(t) \longrightarrow \infty$) are respected by this approximation.
- 4.8e: Discretization of the analytical spatial integration from numerical spatial integration (necessary in all cases for the methods presented in this work, in order to preserve full freedom for the spatial density function).

The final term 4.8g is expressed in a very convenient way for our problem. The first part of the term 4.8g (red) is exactly the result of the spectral integral over a small element of the domain where the spatial σ_s coefficient is uniform (i.e. the first voxel from the camera). The second term (green) is a product of all spectral integrals between the start of the ray and the current point, without external illumination, and divided by the constant term $\sum_{k=0}^{N} P_k C_k(\lambda_1,\lambda_2)$. Finally, the third term (blue) is only the external light contribution at point t.

Knowing the exact function ε isn't relevant in this case, as it will be computed by the analytical resolution of the spectral integral. The purpose of setting this function is simply to become able to separate the exponential in a product.

4.4.1 Implementation and error analysis

This way of writing the spectral integral makes it straightforward to compute in a front-to-back algorithm using two precomputed values for each voxel, as depicted in algorithms 6 and 7:

Algorithm 6 Pre-integration approximation

```
function PRECOMPUTE(v)                          ▷ v is a voxel of the discretized 3D space
    ρ ← LOCALDENSITY(v)
    p ← SPECTRALINTEGRATION(P, ρ, λ0, λ1)
    STORE(v, p)                                 ▷ Storing spectral integral in voxel v
end function
```

The results obtained with this method do not really fall into the criteria that were decided at the beginning of the report (section 2). While the renders are indeed fast and explorable to some extent (full exploration of the space is possible as the pre-integration only computes the contributions for each small voxelized element, which are to be summed up at run-time), they aren't close enough to the ground truth to appear "visually" accurate, especially in comparison with the previous projection method (4.2). Example results are shown in the appendix (6.3).

This distance between the ground truth and the actual renders from this approximation should come from the assumption that one term of the spectral integral (E_1) can be nullified in front of the other contributions. This should be an assumption that is too strong for accurate renders and doesn't retrieve all of the information that would make for a relevant render. Moreover, there is no obvious way to increase the accuracy of this scheme and get better results

Algorithm 7 Run time computation

global variables
P, Spectral polynomial
λ_0, λ_1, spectral integration bounds
end global variables

function COMPUTEINTENSITY(p) ▷ p is a pixel on the camera array
$r \leftarrow$ GETRAY(p)
$p \leftarrow$ GETPOINTALONGRAY(r, j)
$C_k \leftarrow$ COMPUTEPKCK(P, λ_1, λ_2)
$prod \leftarrow 1$
$v \leftarrow 0$
for $j \leftarrow 0...N$ **do** ▷ Spatial numerical integration
$p \leftarrow$ GETPOINTALONGRAY(r, j)
$v_{local} \leftarrow$ LOAD(p) ▷ Loading the spectral integral value at position p
if $j! = 0$ **then**
$v_{local} \leftarrow v_{local}/C_k$
end if
$v \leftarrow v + \sigma_s \cdot prod \cdot \Delta_j$
$prod \leftarrow prod \cdot v_{local}$
end for
STORE(p, v) ▷ Storing intensity value v in pixel p
end function

(whereas for the projection scheme, increasing the cardinality of the basis will significantly increase the results). Thus, the previous method should be preferred.

— 5 —
Conclusion & future works

In this report, we presented an efficient technique to compute a physically accurate spectral volumetric render for astronomical scenes. We achieve to compute in real-time complex scenes taking multiple seconds per frame with the ground truth method, without significant visual changes, reaching an improvement of 83% on the double numerical integration performances. This technique makes it possible to dynamically explore a nebulae scene in real-time and dynamically change the filters used by the simulated telescope in order to retrieve multiple kinds of spectral information, mimicking the classical imagery done with telescopes such as Hubble and JWST. This technique is also easily scalable, as increasing the spectral precision relies on augmenting the cardinality of the considered basis, and increasing the spatial resolution relies on the resolution of the underlying voxel array.

While this basis integration method is general and doesn't necessarily rely on the spectral model that was presented in section 3, further research could try to generalize the model for any kind of scene (and not only astrophysical ones). The model for the attenuation in astronomical dust clouds seems to be linearly correlated with the inverse of the wavelength, but this relationship would no longer be true when considering any arbitrary volumetric objects. Different models would have to be chosen in order to apply a similar integration method.

One of the "hard" assumptions made by our model in section 3 is also that the phase function for the participating medium must be isotropic, as in any case the information on the position of the emitting star would be lost during the projection part of the spectral pre-computations. However, it could be possible to relax this hypothesis by using another form of projection to keep the angle-dependant intensity information, for example on the basis of 3D spherical harmonics.

Furthermore, one area of improvement for our model would be to take into account multiple scattering. The integration scheme currently only models one light bounce, but the fact that the model presented in section 4.3 allows the projection of any number of light sources in the scene at constant cost appears to be a good indication for a potential extension of the model. It should be possible to compute all of the light bounces in a first pass as secondary lights and project them on the same bi-dimensional basis. While the cost for the pre-computation would greatly increase in order to compute all the possible N-bounce interactions, the cost at run-time would remain constant (and in our case, real-time).

To conclude, with precise a-priori knowledge of physics and a relevant model for complex phenomenons, it is possible to render in great detail, and in a few milliseconds, complex astronomical scenes which would otherwise be impossible to compute in real-time.

— 6 —
Addendum

6.1 Analytical integration

The following equations demonstrate a relevant recurrence relation for the computation of the analytical antiderivative of the product of a polynomial and an exponential (the applied basis function defined in section 4).

The only remaining term left to be computed is the E_1[1] function, which is only sampled twice per definite integration of the whole basis function set due to the recurrence relationship. It can be tabulated or approximated (for example with a Chebyshev approximation [CT69]).

$$\begin{aligned}
&\int \lambda^n e^{-g(t)/\lambda} d\lambda \\
&= \frac{\lambda^{n+1}}{n+1} e^{-g(t)/\lambda} + \frac{g(t)}{n+1} \int \frac{e^{-g(t)u}}{u^{n+1}} du \qquad \text{With } u = \frac{1}{\lambda} \\
&= \beta_n(g(t),\lambda) + \frac{g(t)}{n+1} \alpha_n(g(t),\lambda)
\end{aligned} \tag{6.1}$$

$$\begin{aligned}
\alpha_n(x,\lambda) &= -\beta_{n-1}(x,\lambda) - \frac{x}{n} \alpha_{n-1}(x,\lambda) \\
\alpha_0(x,\lambda) &= -E_1(x/\lambda)
\end{aligned} \tag{6.2}$$

$$\begin{aligned}
\beta_n(x,\lambda) &= \frac{\lambda n}{n+1} \beta_{n-1}(x,\lambda) \\
\beta_0(x,\lambda) &= \lambda e^{-x/\lambda}
\end{aligned} \tag{6.3}$$

6.1.1 Negative powers

For the sake of generality (allowing the numerical integral of models integrating Wien's approximation for emission spectra 3.2.2), we show below the antiderivative for negative powers of lambda.

$$\begin{aligned}
&\int \lambda^{-1} e^{-g(t)/\lambda} d\lambda \\
&= E_1(g(t)/\lambda)
\end{aligned}$$

[1] Exponential integral function E_1

$$\int \lambda^{-2} e^{-g(t)/\lambda} d\lambda$$
$$= \frac{e^{-g(t)/\lambda}}{g(t)}$$

$\forall k \geq 3$

$$\int \lambda^{-k} e^{-g(t)/\lambda} d\lambda$$
$$= -\int u^{k-2} e^{-g(t)u} du \qquad \text{With } u = \frac{1}{\lambda}$$
$$= -\frac{u^{k-2} e^{-g(t)u}}{g(t)} + \frac{k-2}{g(t)} \int u^{k-3} e^{-g(t)u} du$$

We can then use a similar recursive relationship as in the previous section 6.1 to compute the remaining antiderivative of the form $\int u^k e^{-au} du$.

6.2 Comparison of convergences for projection

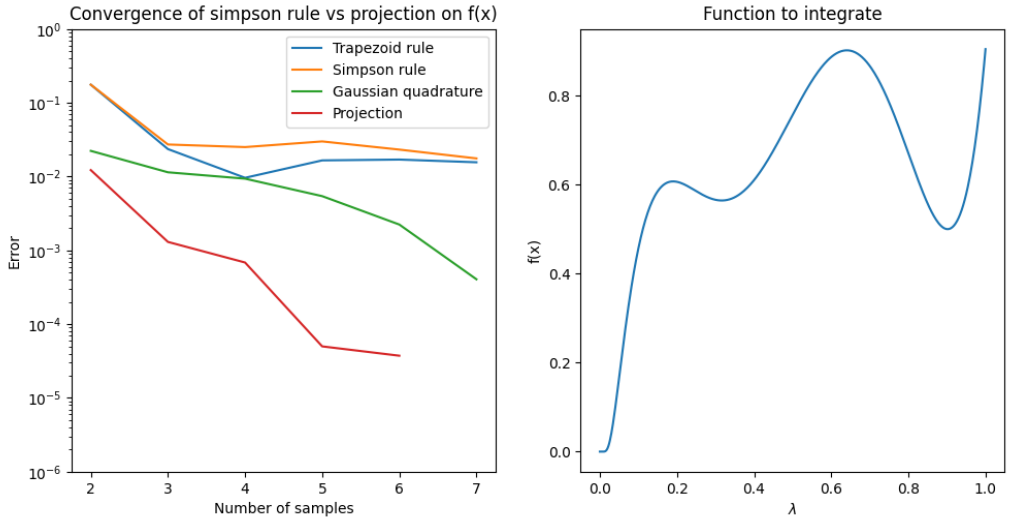
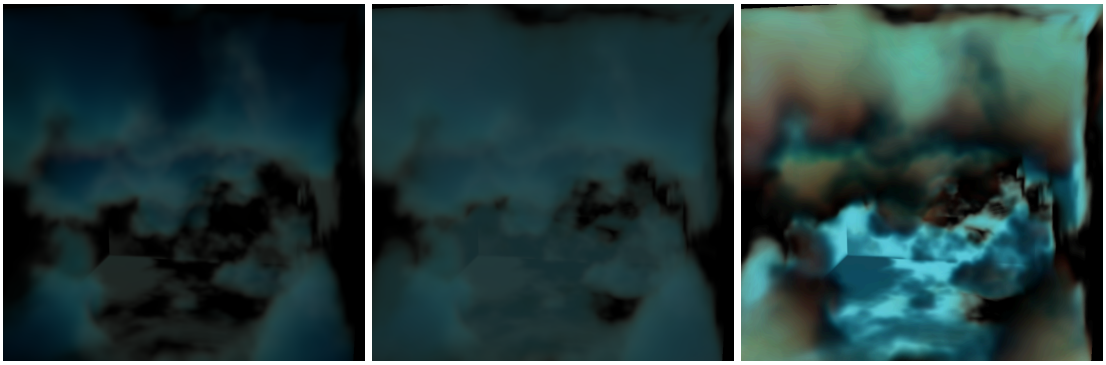


Figure 6.1: Comparison of convergences for projection and numerical methods on the integration of element $f(x) = ((x-1)(x-.2)(x-.5)(x-.7)(x)100+1)e^{-.1/x}$

Figure 6.1 illustrates how the projection method presented in section 4 compares to other numerical methods. Two Newton-Cotes methods and Gaussian quadrature converge slower than the projection method for most functions that were observed (excluding degenerate cases for the polynomial coefficients and the exponential arguments).

6.3 Results of the pre-integration method



(a) (b) (c)

Figure 6.2: Comparison between numerically integrated renders and renders obtained by pre-integration
(a) Double numerical integral (b) Pre-integrated results approximation (c) Absolute error $\times 4$

Figure 6.2 illustrates the results of the pre-integration method discussed in section 4.4. We believe these results not to be accurate enough in comparison with the double numerical integration, especially when compared with the projection method discussed in section 4.2. While the color of the main part of the nebulae appears similar for both methods, shadows, and external light attenuation doesn't appear clearly in the pre-integrated results, and multiple discrepancies appear in the low-density areas of the nebula cloud.

Bibliography

[Anr] *INRIA ANR Galaxy/veRTIGE*. https://www.inria.fr/fr/une-promenade-virtuelle-dans-notre-galaxie. Accessed: 2023-06-14.

[Kh84] J. Kajiya and B. von herzen. "Ray Tracing Volume Densities". In: *ACM SIGGRAPH Computer Graphics* 18 (July 1984). DOI: 10.1145/964965.808594.

[Kaj86] J. T. Kajiya. "The Rendering Equation". In: *SIGGRAPH Comput. Graph.* 20.4 (1986). DOI: 10.1145/15886.15902.

[Ber+02] S. Bergner et al. "Interactive Spectral Volume Rendering." In: *Proceedings of the IEEE Visualization Conference*. Jan. 2002. DOI: 10.1109/VISUAL.2002.1183763.

[RF91] M. G. Raso and A. Fournier. "A Piecewise Polynomial Approach to Shading Using Spectral Distributions". In: *Proceedings of Graphics Interface '91*. GI '91. Calgary, Alberta, Canada, 1991.

[Pee93] M. S. Peercy. "Linear Color Representations for Full Speed Spectral Rendering". In: *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '93. Anaheim, CA, 1993. DOI: 10.1145/166117.166142.

[OYH18] H. Otsu, M. Yamamoto, and T. Hachisuka. "Reproducing Spectral Reflectances From Tristimulus Colours". In: *Computer Graphics Forum* 37.6 (2018). DOI: https://doi.org/10.1111/cgf.13332.

[Wad13] C. Waddle. "Real-Time Spectral Rendering". In: *Color and Imaging Conference* 21 (Jan. 2013). DOI: 10.2352/CIC.2013.21.1.art00044.

[Wie97] W. Wien. "XXX. On the division of energy in the emission-spectrum of a black body". In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 43.262 (1897). DOI: 10.1080/14786449708620983.

[AG11] Aleman, I. and Gruenwald, R. "H2 infrared line emission from the ionized region of planetary nebulae". In: *A&A* 528 (2011). DOI: 10.1051/0004-6361/201014978.

[FT79] G. Ferland and J. Truran. "An X-Ray Model for the Nebula of Nova DQ Herculis 1934." In: *The Astrophysical Journal* 11 (Aug. 1979). DOI: 10.1086/158773.

[EKE01] K. Engel, M. Kraus, and T. Ertl. "High-Quality Pre-Integrated Volume Rendering Using Hardware-Accelerated Pixel Shading". In: *Proceedings of the ACM SIGGRAPH*. New York, NY, USA, 2001. DOI: 10.1145/383507.383515.

[CT69] W. J. Cody and H. C. Thacher. “Chebyshev Approximations for the Exponential Integral Ei(x)”. In: *Mathematics of Computation* 23.106 (1969). (Visited on 06/07/2023).