MATEMATICKO-FYZIKÁLNÍ FAKULTA
Univerzita Karlova

BAKALÁŘSKÁ PRÁCE

Ondřej Janoška

Přenos záření v protoplanetárních discích versus pozorování interferometrem ALMA

Astronomický ústav MFF UK

Vedoucí bakalářské práce: doc. Mgr. Miroslav Brož, Ph.D
Studijní program: Fyzika
Studijní obor: Obecná fyzika

Praha 2022

Prohlašuji, že jsem tuto bakalářskou práci vypracoval(a) samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů. Tato práce nebyla využita k získání jiného nebo stejného titulu.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V ..... dne ....

Podpis autora

V první řadě chci poděkovat mému vedoucímu docentu Brožovi za konzultace, rady a příjemný přístup tváří v tvář i těm nejelementárnějším dotazům. Dále patří mé vřelé díky své přítelkyni, rodině a kamarádům, kteří mi byli v průběhu psaní inspirací, oporou i rozptýlením.

Název práce: Přenos záření v protoplanetárních discích versus pozorování interferometrem ALMA

Autor: Ondřej Janoška

Ústav: Astronomický ústav MFF UK

Vedoucí bakalářské práce: doc. Mgr. Miroslav Brož, Ph.D, Astronomický ústav MFF UK

Abstrakt: V této práci se zabýváme pozorovatelností struktur v protoplanetárních discích přístrojem ALMA. Strukturami, které vytvářejí vnořené planety, bývají spirální ramena, prstence nebo mezery. Cílem je vypočítat syntetické obrazy takových disků. Za tímto účelem jsme naprogramovali program Faradit, kterým jsme dvourozměrnou hydrodynamickou simulaci protoplanetárního disku rozšířili do třetího, vertikálního směru. Zkoumali jsme pět různých případů: vertikálně izotermální disk, opticky tlustý disk, opticky tenký disk, disk s rozdělením velikostí prachových zrn a disk s viskózním ohřevem a vypařováním zrn. Ve všech jsme pomocí programu RADMC-3D vypočítali přenos záření a získali ideální obrazy pro šest různých vlnových délek. Nakonec jsme pomocí simulátoru interferometrického pozorování (ALMA OST) posoudili, jak jsou struktury pozorovatelné. Ačkoli jsme v ideálních obrazech projevy přítomnosti planet o hmotnosti $3\,M_\oplus$ viděli, nejsou přístrojem ALMA pozorovatelné v kontinuu. Muselo by se jednat o pozdější vývojovou fázi, s otevřenou mezerou.

Klíčová slova: přenos záření, protoplanetární disk, ALMA, interferometrie

Title: Radiation transfer in protoplanetary disks vs ALMA interferometric observations

Author: Ondřej Janoška

Institute: Astronomical Institute MFF UK

Supervisor: doc. Mgr. Miroslav Brož, Ph.D, Astronomical Institute MFF UK

Abstract: We study observability of protoplanetary disc structures by the ALMA interferometer. Structures, which planets create in discs have a form of spiral arms, rings or gaps. Our goal is to calculate synthetic images of such discs. For this purpose, we created Faradit, a program for extending a two-dimensional hydrodynamical simulation of a protoplanetary disc to the third, vertical dimension. We studied 5 different cases: a vertically isothermal disc, an optically thick disk, an optically thin disc, a disc with a distribution of grain sizes and a disc with viscous heating and dust grain evaporation. In all of them, we calculated radiation transfer using RADMC-3D program and created ideal images for six different wavelengths. Eventually, we assessed whether these structures are observable, using a simulator of interferometric observations (ALMA OST). Despite the structures induced by protoplanets with masses $3\,M_{\oplus}$ were present, we concluded they are not observable with a realistic signal and noise of ALMA in continuum. The respective disc would have to be more evolved, with an opened gap.

Keywords: radiation transfer, protoplanetary disk, ALMA, interferometry

Obsah

1 Úvod **2**

2 Pozorování disků **3**
2.1 Princip interferometru ALMA 3
2.2 ALMA a protoplanetární disky v odborné literatuře 7

3 Fyzika disků **10**
3.1 Plyn 10
3.2 Prach a balvany 11
3.3 Viskozita 13
3.4 Opacita 16

4 Simulace toků plynu a přenosu záření **19**
4.1 Princip Fargo2d a konkretní disky 19
4.2 Princip Radmc3d 22
4.3 Faradit 23

5 Syntetické a simulované obrazy **34**
5.1 Základní trojrozměrný disk 34
5.2 Zkoumané disky 35
5.2.1 Vertikálně izotermální disk 37
5.2.2 Opticky tlustý disk s radiačním ohřevem 39
5.2.3 Opticky tenký disk s radiačním ohřevem 42
5.2.4 Vícezrnný disk s radiačním ohřevem 45
5.2.5 Opticky tlustý disk s viskózním ohřevem a vypařenými zrny 49
5.2.6 Některé zajímavé detaily zkoumaných modelů 53
5.3 Syntetické inreferometrické pozorování 55

6 Závěr **58**

Literatura **59**

A Přílohy **64**
A.1 Výpis programu disk.py 64
A.2 Výpis programu opacities.py 86

1. Úvod

Teorie původu Sluneční soustavy i jiných planetárních systémů je bezmála 400 let stará. První zmínky o „vírech hmoty“ lze najít u Reného Descarta (viz překlad Descartes 1677), rigorózněji však byl disk kolem mladé hvězdy popsán Emanuelem Swedenborgem (Swedenborg 1734), Immanuelem Kantem (Kant 1755) a Pierrem-Simonem de Laplacem (de Laplace 1796) v 18. století. Dnes se pozoruje, že planetární systémy vznikají z protoplanetárního disk prachu a plynu, který vzniká spolu s centrální hvězdou při kolapsu mezihvězdné hmoty. Planety samotné se formují uvnitř disku postupnou akrecí pevné látky a plynu. Samy také disk perturbují, tvoříc spirální ramena , mezery a prstence (Armitage 2010).

Protoplanetární disky jsou hojně pozorovány přímo (například instrumentem VLT/SPHERE, de Boer a kol. 2021) i interferometricky (mimo jiné interferometrem ALMA, viz třeba Andrews a kol. 2018). Přímo pozorovat v disku vnořené planety nebo dokonce planetární embrya je obtížné. Jednodušší je pozorovat rozlehlejší struktury, které planety v disku tvoří. Nyní je standardem porovnávat pozorování přímo s numerickými modely. Pokud chceme například identifikovat planetu podobnou Jupiteru podle mezery otevřené v disku, musíme provést hydrodynamickou simulaci a ukázat, že planeta o hmotnosti Jupiteru by skutečně v disku mezeru otevřela (viz Fedele a kol. 2018). Pak musíme provést simulované pozorování této struktury a ukázat, že by v rámci nejistot daných fotonovým šumem a jinými přístrojovými jevy vypadala stejně, jako ta pozorovaná.

Hydrodynamické simulace jsou ovšem náročné na výpočetní čas (např. Flock a kol. 2016). Protoplanetární disky jsou obvykle geometricky tenké, poměr jejich výšky ku průměru bývá řádově menší než jedna. Pro zjednodušení se proto hydrodynamické simulace často provádějí pouze na dvourozměrné doméně, ve vertikálním směru jsou všechny veličiny integrované. Tím se výpočetní náročnost značně sníží (např. Chrenko a kol. 2017).

Při interakci disku se zářením hvězdy a při pozorování je ale třetí rozměr disku důležitý. Materiál poblíž hvězdy může v některých místech zastiňovat vzdálenější části disku, které jsou pak chladnější a méně září vlastním tepelným zářením. Navíc pokud se na disk nedíváme z kolmého směru, může disk samotný hvězdu zcela zastínit. Tyto efekty také závisí na vlnové délce — na některých λ může být disk opticky tlustý a zastínění bude vypadat jinak, než na λ, na kterých je naopak opticky tenký. Dvourozměrný model tedy nemusí být pro vyhodnocení pozorování dostatečný. Obecně tedy bývá potřeba dvourozměrný disk převést do tří rozměrů (např. Dullemond a kol. 2020).

V této práci jsme pro určitou dvourozměrnou hydrodynamickou simulaci protoplanetárního disku s vnořenými planetami o hmotnosti $3\,M_\oplus$ — převzatou z práce Chrenko a kol. (2017) — provedli výše uvedený převod. Použili jsme přitom vlastní program pro rozšíření plochého disku do tří rozměrů, provedli jsme výpočet přenosu záření open-source programem RADMC-3D (Dullemond a kol. 2012) a též simulovali jeho pozorování interferometrem ALMA (`https://almaost.jb.man.ac.uk/`).

2. Pozorování disků

Protoplanetární disky nejsou pouze předpovězené teorií, ale jsou přímo i nepřímo pozorované. Interferometr ALMA je nejen svým rozlišením ale i oborem, na kterém pozoruje, pro pozorování disků ideální. Interferometrické pozorování se ale od toho přímého principiálně liší natolik, že je vhodné, abychom mu věnovali tuto kapitolu.

2.1 Princip interferometru ALMA

Detektor ALMA (Atacama Large Milimeter-submilimeter Array, česky Atacamské velké milimetrové-podmilimetrové souantění) v Jižní Americe je soustava 54 12metrových antén a 12 menších, 7metrových. Obsáhlé informace o přístroji jsou dostupné z webových stránek ESO `https://www.eso.org/public/teles-instr/alma`. ALMA má na kontě celou řadu významných měření objektů, které jinak nebylo možné měřit. Kromě protoplanetárních disků je vhodné zmínit například příspěvek k celosvětovému projektu Event Horizon Telescope jehož výsledkem byl první obraz horizontu událostí černé díry (viz Goddi a kol. 2019). Jak je tedy ALMA schopná vyššího rozlišení, než konvenční teleskopy?

Cesta k většímu úhlovému rozlišení vede přes interferenci vln. Představme si dva bodové zdroje — například dvojhvězdu — v dalekém poli. Úhlové rozlišení v radiánech $\Delta\theta$ teleskopů lze odhadnout vztahem (dle Brož a Wolf 2017)

$$\Delta\theta \approx 1{,}22\frac{\lambda}{D}, \tag{2.1}$$

kde λ je vlnová délka, na které snímáme, a D je průměr teleskopu. Konvenční radiový teleskop o průměru řekněme $D = 12\,\mathrm{m}$ pracující na vlnové délce $\lambda = 870\,\mu\mathrm{m}$ bude mít dle (2.1) úhlové rozlišení nepřevyšující 18 úhlových vteřin. Pozorovat dvojhvězdu HD 171779 v souhvězdí Draka s úhlovou separací složek zhruba $0{,}25''$ (Lane a Muterspaugh 2004) tímto teleskopem se tedy zdá být nemožné. Klíčové slovo: *zdá*.

Nechme světlo z dvojhvězdy projít dvouštěrbinou, resp. dvěma dalekohledy (pro lepší zisk), a interferovat na stínítku, dle situace na obrázku 2.1.

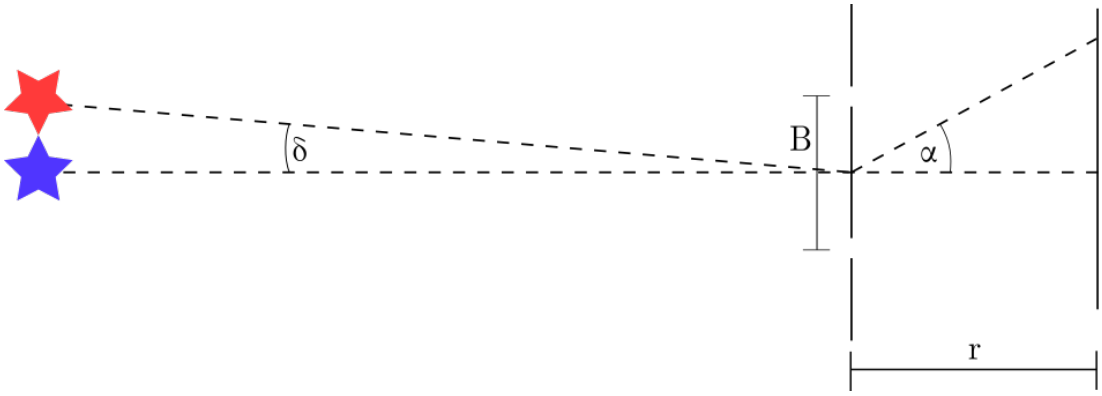
Pozorujeme-li jednu jen hvězdu v dalekém poli, přichází rovinná vlna s vlnovou délkou λ do dalekohledů vzdálených B, na vzdáleném detekujícím stínítku máme proměnnou intenzitu

$$I(\alpha) = I_* \left[1 + \cos\left(\frac{2\pi B}{\lambda}\alpha\right)\right], \tag{2.2}$$

kde úhel α určuje polohu na stínítku. Vzdáleným stínítkem myslíme takové, které je dál od dalekohledů, než jsou dalekohledy od sebe: $B \ll r$. Na čipu vidíme *proužky*.

Kdybychom radioteleskopy od sebe oddálili, dle rovnice (2.2) bychom spatřili jinou frekvenci proužků na čipu, ale jejich kontrast, resp. viditelnost, definovaná vztahem

$$\mathcal{V} := \frac{I_{\max} - I_{\min}}{I_{\max} + I_{\min}}, \tag{2.3}$$



Obrázek 2.1: Základ interferometru je vlastně dvouštěrbina. Záření hvězd bývá většinou sbíráno dalekohledy a ne štěrbinami, stejně tak na stínítko bývá záření dovedeno složitou optikou, avšak princip je stejný. Zde pozorujeme dvojhvězdu s úhlovou separací složek δ dvěma dalekohledy od sebe vzdálenými B, za kterými je ve vzdálenosti $r \gg B$ stínítko, čip nebo nějaký jiný detektor.

by zůstala stejná, proto, že amplituda kosinu v rovnici (2.2) na délce základny nezávisí. $\mathcal{V} = 1$.

Pozorování dvojhvězdy se ukazuje být mnohem zajímavější. Výpočet je stejný jako u klasické dvouštěrbiny, pouze musíme na čipu provést superpozici obou vln. Vlna od jedné z hvězd bude vykazovat posuv fáze daný její úhlovou vzdáleností δ od hvězdy druhé. Výsledná intenzita, pokud má každá ze složek stejnou intenzitu I_* a celková intenzita dvojhvězdy je I_{**}, bude

$$I(\alpha) = \frac{I_{**}}{8}\left\{3 + \cos\left[\frac{2\pi B}{\lambda}\delta\right] + 2\cos\left[\frac{2\pi B}{\lambda}\alpha\right] + 2\cos\left[\frac{2\pi B}{\lambda}(\alpha+\delta)\right]\right\}. \tag{2.4}$$

Pro dané B opět pozorujeme na čipu proužky, ale se změnou B *není* $\mathcal{V}$ konstantní, nýbrž

$$\mathcal{V}_{**} = \frac{1}{\sqrt{2}}\sqrt{1 + \cos\left(\frac{2\pi B}{\lambda}\delta\right)}. \tag{2.5}$$

Tento výsledek *není* omezený průměrem teleskopů. Pro libovolně velké teleskopy pracující na libovolných vlnových délkách najdeme takovou vzdálenost B, pro kterou už $\mathcal{V}$ bude měřitelně rozdílné od jedné. Dva radioteleskopy zmiňované výše (které byly samostatně zcela neschopné rozlišit dvojhvězdu HD 171779) postavené 360 m od sebe naměří dle rovnice (2.4) $\mathcal{V} \approx 0$ a s jistotou tak vyvrátí, že pozorují pouze jednu hvězdu. Rozlišení se stalo funkcí základny B. ALMA využívá možnosti soustavy se základnami B mezi 150 metry a 16 kilometry.

Pokud předem víme, jakou viditelnost interferenčních proužku $\mathcal{V}(B)$ má pozorovaný objekt mít, je jasné, jak správně provést měření. Stačí si zvolit některé základny B a naměřenou viditelnost porovnávat s vypočtenou.

Co když chceme pozorovat objekt, o kterém apriori nevíme, jak vypadá? Řešení nabízí teorém, který svazuje viditelnost $\mathcal{V}$ s rozložením intenzity zdroje na nebeské sféře.

Tvrzení 1 (Teorém van Citterta a Zernikeho v kontextu astronomické interferometrie)**.** *Nechť intenzita $I(\alpha, \delta) : Z \to \mathbb{R}_0^+$ je zobrazení ze zorného pole Z do kladných reálných čísel. Předpokládejme, že zdroj je mnohem menší, než je jeho vzdálenost od pozorovatele a že světlo z něj vycházející není koherentní, avšak*

je monochromatické s vlnovou délkou λ. Pak komplexní viditelnost μ definujeme jako Fourierovu transformaci $I(\alpha, \delta)$

$$\mu(u,v) := \mathcal{F}\left[I(\alpha,\delta)\right](u,v) \equiv \int_Z I(\alpha,\delta) e^{-2\pi i \ (\alpha,\delta)\cdot\frac{\vec{B}}{\lambda}(u,v)} \, \mathrm{d}\alpha \, \mathrm{d}\delta, \tag{2.6}$$

kde $\vec{B}(u,v)$ je projekce základny na nebeskou sféru a u, v jsou prostorové frekvence. Komplexní μ je taková veličina, že platí

$$|\mu| = \mathcal{V}, \tag{2.7}$$

kde $\mathcal{V}$ je viditelnost interferenčních proužků pozorovaných ideálním interferometrem, a

$$\arg \mu = \phi, \tag{2.8}$$

kde fáze ϕ odpovídá poloze proužku nultého řádu (též „bílého proužku“).

Důkaz. Viz Max Born (1999), Thompson a kol. (2017).

□

Základny interferometru tedy odpovídají prostorovému rozlišení obrazu. Čím delší základna, tím vyšší prostorová frekvence (ve stejném směru, jako je orientace základy vůči obrazu) a tedy i vyšší rozlišení. Na měření ideálního interferometru tedy stačí aplikovat inverzní Fourierovu transformaci a získáme obraz.

Praktická aplikace není tak přímočará. Za prvé, abychom opravdu získali obraz, potřebujeme μ vyčíslené ve všech bodech roviny (u, v). To částečně řeší metoda supersyntézy, která využívá rotaci Země pro změnu projekce základen na nebeskou sféru. Supersyntéza ale přináší další problém: udržení koherence. Interferometry musí být vybaveny sofistikovaným systémem zpožďovací dráhy, která zajišťuje, že fáze vln dopadajících do každé ze štěrbin, resp. dalekohledů není posunutá jen zakřivením Země. Počet měření ale musí být vždy konečný. Místo plné Fourierovy transformace stejně můžeme maximálně provést diskrétní rychlou Fourierovu transformaci.

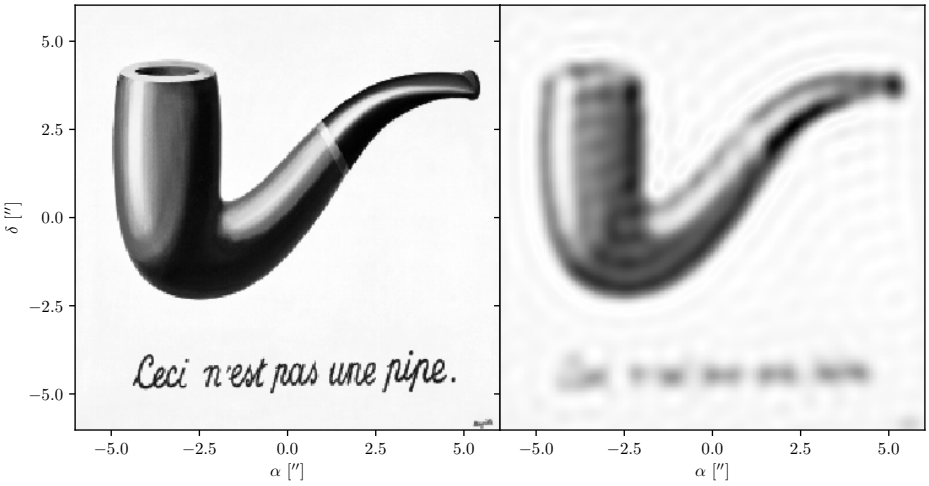
Některé body roviny (u, v) jsou prakticky neměřitelné. Nejdelší možná základna pozemského interferometru bude vždy kratší, než průměr Země, nejkratší pak nebude kratší, než průměr jedné antény, neboť blíž k sobě je umístit nelze. Chybějící body roviny (u, v) výrazně ovlivní obraz získaný inverzní transformací. Důležité je proto zmínit, že výsledkem měření interferometrem nikdy *není* obraz, ale pouze *některé* body roviny (u, v) s obrazem spjaté.

Jak získat obraz vyčištěný od artefaktů interferometrie? Obvyklý postup je algoritmus CLEAN. Jde o relativně jednoduchý a efektivní algoritmus. Jeho implementace ve zkratce spočívá v iteraci následujícího postupu:

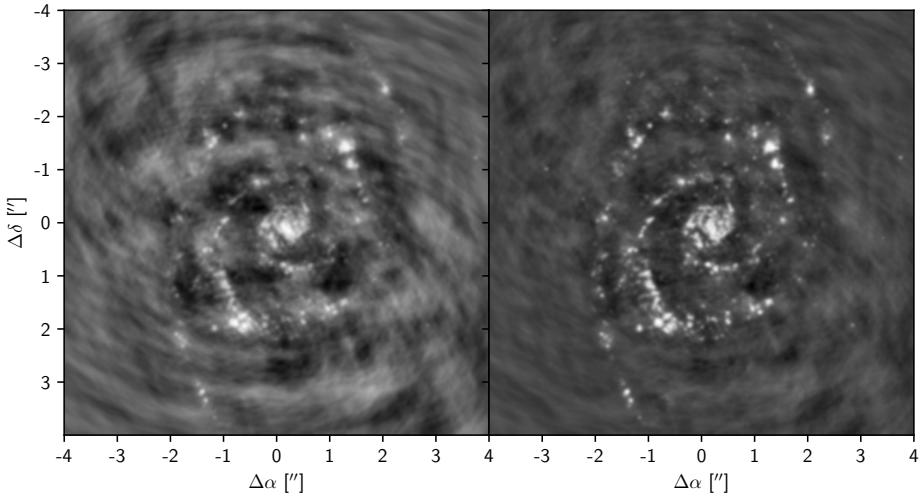
1. nalezení nejvýraznějšího zdroje v obraze,

2. odečtení bodového zdroje se stejnou intenzitou a polohou.

Po dosažení zvoleného počtu iterací (obvykle po odstranění všech výrazných zdrojů) je vytvořen čistý obraz složený pouze ze zdrojů o stejných polohách a intenzitách, jaké měly zdroje odečtené při iteraci.



Obrázek 2.2: Příklad vnesení sinových artefaktů do obrazu získaného inverzní fourierovou transformací. Obraz nalevo (Zrádnost obrazů, Magritte 1929) má rozlišení 256×256 pixelů. Vpravo jsme nasimulovali měření tohoto obrazu ideálním diskrétním interferometrem s takovými základnami, že $|(u,v)| \leq 300\,\mathrm{k}\lambda$. Tato délka základen není pro interferometr ALMA vůbec nereálná, ale úplné pokrytí všech základen až po $(u,v) = (0,0)$ je v praxi nemožné. Ani idealizované měření nestačí pro rekonstrukci původního obrazu; detaily textu úplně chybí, zatímco na těle dýmky se objevují sinové artefakty. Výsledek měření interferometru není dýmka, je to neúplný inverzní fourierovský obraz dýmky.



Obrázek 2.3: Simulované pozorování galaxie M51 pomocí ALMA OST. Vlevo neošetřený obraz získaný inverzní Fourierovou transformací, vpravo vyčištěný algoritmem CLEAN. Zatímco původní obraz je zanesený artefakty a PSF zdrojů, ve vyčištěném jsou zřetelně vidět struktury galaxie. ALMA OST bylo nastaveno na konfiguraci antén C–10 a vlnovou délku pozorování 3,2 mm.

2.2 ALMA a protoplanetární disky v odborné literatuře

Jak vypadá měření pomocí ALMA v praxi? V rámci této práce jsme podrobněji studovali některé publikace zabývající se protoplanetárními disky. Zde prezentujeme výběr z nich:

1. Práce Cox a kol. (2017) se zabývá analýzou 49 systémů s protoplanetárními disky v mračnu ρ Ophiuchi pomocí ALMA v kontinuu, na vlnové délce 870 μm. Autoři využili konfigurace antén C34-7/(6) se základnami o délkách $B \in (42, 1547)$ m. Na získaná data aplikovali algoritmus CLEAN a získali tak rozlišení až 0,18″. Následně odvodili rozměry plynoprachových disků. Disky s jednou centrální hvězdou mají větší rozměry ($r < 90$ au) oproti těm, jejichž centrální objektem je dvoj- či trojhvězda ($r < 40$ au).

2. Ve článku Fedele a kol. (2018) prezentují údaje o měření systému AS 209 v souhvězdí Hadonoše. Obraz získaný algoritmem CLEAN odhalil prstencovou strukturu disku. Autoři porovnali tuto strukturu s výsledky hydrodynamických simulací kódem Fargo3D. Simulovali přitom vliv obřích planet na profil hustoty disku. Ukazuje se, že jedna nebo dvě planety podobné Saturnu na 95 au by mohly vytvořit obdobné mezery v disku, jaké jsou v systému AS 209 pozorované.

3. Rozsáhlou studii 20 protoplanetárních disků nazvanou DSHARP (Disk Substructures at High Angular Resolution Project, či Projekt měření podstruktur disků ve vysokém úhlovém rozlišení) prezentovali autoři v práci Andrews a kol. (2018). Měření pomocí ALMA v kontinuu na 1,25 mm (s úhlovým rozlišením ≈ 0,035″) bylo zpracováno algoritmem na bázi CLEAN a jsou ve zpracované i nezpracované formě dostupné z `https://almascience.eso.org/almadata/lp/DSHARP/`. Zkoumané objekty jsou zapsané v tabulce 2.1.

4. V práci Pinte a kol. (2019) byl podroben analýze další disk, tentokrát kolem mladé hvězdy HD 97048 v souhvězdí Chameleona. Měření opět získaná přístrojem ALMA, tentokrát však ve spektrální čáře ^{13}CO 3-2 ($\lambda = 908$ μm) a ^{12}CO 2-1 ($\lambda = 231$ μm), odhalují rychlostní kličku na 130 au. Touto rychlostní kličkou autoři myslí lokální odchylku oběžné rychlosti plynu (menší, než 0,1 km/s), kterou pravděpodobně vytváří svou gravitací vznikající planeta o hmotnosti zhruba $2\,M_\mathrm{J}$. Jde o projev Lindbladových rezonancí v disku, které vytvářejí v disku spirální vlny hustoty. Zdroj těchto oscilací je navíc přímo u mezery v disku, o nichž se předpokládá, že mohou vznikat právě vlivem planet.

5. V následující práci (Pinte a kol. 2020) hledají obdobné rychlostí kličky v jiných systémech. V datech DSHARP zkoumali 18 objektů a ve spektrální čáře ^{12}CO 2-1 u 8 z nich nalezli celkem 9 rychlostních kliček podobných těm, jaké naměřili u hvězdy HD 97048.

6. Pozorovatelnost struktur, které v disku planety vytvářejí, zkoumá například práce Ruge a kol. (2013). Její autoři nejdřív pomocí kódu PLUTO simulovali magnetohydrodynamický vývoj disku a následně kódem MC3D simulovali

přenos záření. Do simulovaného disku vnořili planetu o hmotnosti $M_\mathrm{p} = 10^{-3}\,M_*$, která otevřela v disku mezeru o šířce 2 au a vytvořila spirální vlny hustoty v plynu a prachu. Syntetické pozorování takového disku ze vzdálenosti 140 pc pomocí ALMA ukázalo, že mezera v disku byla zřetelná, Lindbladova rezonance však nebyla měřitelná. Autoři práce nicméně věří, že v hustších discích by spirální vlny mohly být měřitelné.

7. V článku Riols a Latter (2019) prezentují autoři výsledky magentohydrodynamické simulace ve třech rozměrech pomocí kódu Pluto. Jejich simulace probíhala na kartézské doméně s rozlišením $512 \times 512 \times 128$ buněk a snažili se pomocí ní studovat životnost magnetických polí v protoplanetárních discích a jejich zdroj — argumentují, že gravitační nestabilita může být klíčová v nastartování procesů (jako například magnetorotační nestability), které dál přispívají k tvorbě magnetických polí.

8. S použitím Fargo2D1D, rozšířením dvourozměrného kódu Fargo, simulovali autoři práce Chametla a kol. (2020) migraci Jupiteru a Saturnu v plynném disku. Zkoumali celkem 15 modelů s rozdílnými počátečními polohami a hmotnostmi planet. Hydrodynamická simulace probíhala na polární síti o 512×384 buňkách a odhalila, že planety obvykle migrují tak, aby dosáhly rezonance 2:1 nebo 3:2.

Jméno (1)	d [pc] (2)	$\overline{\nu}$ [GHz] (3)	$I_{\overline{\nu},\,\mathrm{max}}$ [mJy/beam] (4)	$F_{\overline{\nu}}$ [mJy] (5)	$\log(T_{\mathrm{eff}}/1\mathrm{K})$ (6)	$\log(L_*/L_\odot)$ (7)
HT Lup	154±2	239,0	8,25	77	3,69±0,02	0,74±0,20
GW Lup	155±3	239,0	3,35	89	3,56±0,02	-0,48±0,20
IM Lup	158±3	239,0	7,11	253	3,63±0,03	0,41±0,20
RU Lup	159±3	239,0	3,45	203	3,61±0,02	0,16±0,20
Sz 114	162±3	239,0	3,36	49	3,50±0,01	-0,69±0,20
Sz 129	161±3	239,0	0,96	86	3,61±0,02	-0,36±0,20
MY Lup	156±3	239,0	1,78	79	3,71±0,02	-0,06±0,20
HD 142666	148±2	231,9	1,28	130	3,88±0,02	0,96±0,21
HD 143006	165±5	239,0	0,67	59	3,75±0,02	0,58±0,15
AS 205	128±2	233,7	6,15	358	3,63±0,03	0,33±0,15
SR 4	134±2	239,0	3,40	69	3,61±0,02	0,07±0,20
Elias 20	138±5	239,0	2,59	104	3,59±0,03	0,35±0,20
DoAr 25	138±3	239,0	1,35	246	3,63±0,03	-0,02±0,20
Elias 24	136±3	231,9	4,63	352	3,63±0,03	0,78±0,20
Elias 27	116^{+19}_{-10}	231,9	4,83	330		
DoAr 33	139±2	239,0	1,89	35	3,65±0,03	0,18±0,20
WSB 52	136±3	239,0	2,60	67	3,57±0,03	-0,15±0,20
WaOph 6	123±2	239,0	8,67	161	3,62±0,03	0,46±0,20
AS 209	121±2	239,0	1,83	288	3,63±0,03	0,15±0,20
HD 163296	101±2	239,0	4,26	715	3,97±0,03	1,23±0,30

Tabulka 2.1: Protoplanetární disky zkoumané v DSHARP (Andrews a kol. 2018). Sloupce obsahují následující data: (1) jméno objektu, (2) vzdálenost objektu, (3) pozorovaná střední frekvence záření, (4) nejvyšší naměřená monochromatická intenzita, (5) celková hustota zářivého toku, (6) logaritmus efektivní teploty centrální hvězdy a (7) logaritmus zářivosti centrální hvězdy.

3. Fyzika disků

Abychom mohli vysvětlit struktury, které lze v protoplanetárních discích pozorovat a abychom mohli disky modelovat, je nutné pochopit fyzikální principy, které se v discích projevují. V této kapitole krátce shrneme základní fyzikální jevy v discích. Pro hlubší analýzu doporučujeme například Brož a Šolc (2013), Armitage (2010) nebo Bertotti a kol. (2003).

3.1 Plyn

Je-li v prostoru oblak plynu, resp. plynu a prachu, může za určitých podmínek gravitačním kolapsem vzniknout hvězda. Základní podmínka kolapsu je Jeansovo kritérium, tj. odhad hmotnosti kolabujícího oblaku na základě bilance gravitační a vnitřní tepelné energie. Dle Brož a kol. (2018) jej můžeme zapsat ve tvaru

$$M_\mathrm{c} \gtrsim \left(\frac{5kT}{G\overline{\mu}m_\mathrm{H}}\right)^{\frac{3}{2}} \left(\frac{4}{3}\pi\rho\right)^{-\frac{1}{2}}, \tag{3.1}$$

kde k označuje Boltzmannovu konstantu, G gravitační konstantu a m_H hmotnost atomu vodíku, M_c je celková hmotnost oblaku, T teplota, $\overline{\mu}$ střední molekulová hmotnost plynu a ρ objemová hustota. Jde pouze o přibližné kritérium, odvozené za předpokladu homogenního oblaku s konstantní hustotou a teplotou.

Pokud je kritérium splněné, většina materiálu akretuje na centrální objekt, ve kterém se může zažehnout termonukleární reakce, čímž vznikne hvězda. Ze zbylého materiálu vzniká kvůli zachování momentu hybnosti rotující protoplanetární disk, ze kterého vznikají planetární systémy. Rozměry disku mohou dosahovat řádově stovek astronomických jednotek, v případě disků kolem hvězd typu T Tauri jde dle Kitamura a kol. (2002) zhruba o rozsah 100–500 AU. Hmotnost disků bývá řádově kolem 1 % hmotnosti Slunce, u hvězd typu T Tauri opět dle Kitamura a kol. (2002) se pohybujeme na rozsahu 0.6–63 % $M_\odot$, typicky však ~ 1 % $M_\odot$.

Obvykle se radiální (tj. se změnou vzdálenosti r od hvězdy) profil plošné hustoty plynu popisuje mocninnou závislostí, například pro model sluneční mlhoviny minimální hmotnosti

$$\Sigma = \Sigma_1 \sqrt{\frac{1\,\mathrm{au}}{r}}^{\,3}, \tag{3.2}$$

nebo pro určité části disků

$$\Sigma = \Sigma_1 \sqrt{\frac{1\,\mathrm{au}}{r}}, \tag{3.3}$$

například v práci Chrenko a kol. (2017) a jiných. Σ_1 zde je povrchová hustota na jedné astronomické jednotce. U sluneční mlhoviny minimální hmotnosti se dle hmotnosti planet například odhaduje $\Sigma_1 \approx 1{,}7 \cdot 10^3\,\mathrm{g/cm^2}$, viz Hayashi (1981).

Přítomnost plynu přirozeně implikuje přítomnost nenulového tlaku a radiálně nekonstantní potenciál φ pak také vede na nenulový gradient a tlaku p.

Dle Eulerovy rovnice

$$\frac{\mathrm{D}\vec{v}}{\mathrm{D}t} = -\frac{\nabla p}{\rho} - \frac{\nabla \varphi}{\rho} \tag{3.4}$$

je plyn kvůli radiálně klesajícímu tlaku nadlehčován proti gravitačnímu zrychlení. Vektorový součet účinků gravitace (centrální hvězdy i disku samotného) s gradientem tlaku je dostředivým zrychlením a jako takový diktuje úhlovou rychlost oběhu. Ta musí být nutně *nižší*, než keplerovská úhlová rychlost

$$\Omega_{\mathrm{K}} = \sqrt{\frac{GM_*}{r^3}}. \tag{3.5}$$

Radiální směr není jediný, ve kterém jsou nenulové gradienty. Kritické pro tuto práci je rozložení objemové hustoty ρ ve vertikálním směru. Ve zjednodušeném popisu (Brož a Šolc 2013) můžeme považovat plyn v disku za ideální a vertikálně adiabatický. Z těchto předpokladů se dostaneme k diferenciální rovnici popisující rovnováhu vertikálních složek sil tlakových a gravitačních

$$\frac{\mathrm{d}\rho}{\mathrm{d}z} + \frac{\overline{\mu} m_{\mathrm{P}}}{kT} \frac{GM_*}{r^3} z\rho = 0. \tag{3.6}$$

Její řešení má tvar

$$\rho(z) = \rho(0) \exp\left(-\frac{z^2}{2H^2}\right), \tag{3.7}$$

kde $\rho(0)$ je objemová hustota v rovníkové rovině, parametr H nazýváme výšková škála, která je zřejmě rovna koeficientu v diferenciální rovnici výše. Explicitně ji lze zapsat

$$H = \sqrt{\frac{kT}{\overline{\mu} m_{\mathrm{P}}} \frac{r^3}{GM_*}}. \tag{3.8}$$

Výše zmíněný předpis teploty $T \propto \frac{1}{\sqrt{r}}$ implikuje $H \propto r^{\frac{5}{4}}$. Takovýto jednoduchý disk se tedy se vzdáleností od hvězdy rozšiřuje. Dosazením hodnot pro sluneční mlhovinu minimální hmotnosti zjišťujeme, že disk je relativně tenký; v oblasti $r = 5\,\mathrm{au}$ je $\frac{H}{r} \approx 0.1$.

3.2 Prach a balvany

Kromě plynu se v discích nachází prach a balvany. Prach je pouze jednou z více ingrediencí v disku a bývá v hustotě řádově méně zastoupen než plyn, hraje však zásadní roli. Vliv prachu na přenos záření je natolik významný, jeho přítomnost zcela mění vzhled i samotnou strukturu disku.

Prachem nazýváme libovolnou substanci tvořenou *pevnými* částečkami, jejichž charakteristický rozměr nepřesahuje řádově centimetr. Důležité je, že ač nejde o plyn, prach je na plyn vázaný. Má stejnou teplotu, rozložení v prostoru i rychlost jako plyn. Typicky jde o vločky, krystalky ledu, uhlíků, silikátů nebo sloučenin.

Atomy a molekuly na sebe navzájem působí přitažlivými elektromagnetickými silami a slepují se do větších a větších struktur. Mikrometrová zrnka prachu mohou narůstat do makroskopických velikostí.

Představíme-li si jednu prachovou částici letící prachoplynným oblakem, její hmotnost se bude navyšovat, pokud se na ni nalepí každá jiná částice, se kterou se setká. Nárůst musí být úměrný součinu hustoty prachu v oblaku a objemu,

který proletí studovaná zárodečná částice. Dle Bertotti a kol. (2003) je nárůst hmotnosti daný vztahem

$$\frac{\mathrm{d}m_\mathrm{z}}{\mathrm{d}t} = 4\pi R_\mathrm{P}^2 \frac{v_T}{\sqrt{A}} \rho_\mathrm{prach}, \tag{3.9}$$

kde m_P označuje hmotnost částice, R_P její poloměr, v_T termální rychlost plynu, A relativní atomovou hmotnost pevné látky a ρ_prach prostorovou hustotu prachu. Slovem prostorová se snažíme naznačit, že jde o hustotu danou jako součet hmotností všech zrnek v metru krychlovém, nikoli hmotnost jednotkové krychle ze stejného materiálu, který tvoří prachová zrna.

Vztah popsaný výše je ovšem zjednodušením, protože předpokládá kulová zrnka v celém průběhu akrece. Numerické simulace však poukazují na fraktální strukturu s Hausdorfovou dimenzí od ~1,5 ze začátku formace zrna až k dimenzi kolem 2,5 až 3 ke konci procesu (Testi a kol. 2014). Povrch zrnka, na který se mohou částice nalepovat, je ve skutečnosti větší než povrch koule a faktor $4\pi R_\mathrm{P}^2$ v rovnici (3.9) proto neodpovídá reálnému nárůstu.

Plyn, ve kterém se prachová zrna nachází, obíhá kolem centrální hvězdy rychlostí nižší než keplerovskou. Stojí za tím gradient tlaku, za kterým zase stojí odpudivé elektromagnetické síly mezi molekulami. Prachové částice však takové síly (mimo srážky) nevykazují. Samy mohou být dle Archimédova zákona nadlehčovány, ale s rostoucím rozměrem zrn tento efekt klesá a na prach proto působí větší dostředivá síla než na plyn na daném r. Důsledkem toho musí prach obíhat rychleji než plyn, aby se udržel na kruhové dráze, ale pak se zákonitě musí o plyn třít. Třením prach přichází o mechanickou energii, nemůže se udržet na kruhové dráze a spiráluje směrem ke hvězdě.

Kromě spirálování budou prachová zrna rozptýlená v disku sesedat k jeho rovníkové rovině. Jednoduchý rozbor sil působících na částici prachu vede na diferenciální rovnici podobnou tlumenému harmonickému oscilátoru,

$$\frac{\mathrm{d}^2 z}{\mathrm{d}t^2} = -\frac{GM_*}{r^2}\frac{z}{r} - \frac{S\ \rho_\mathrm{plyn} v_T}{m_\mathrm{P}}\frac{\mathrm{d}z}{\mathrm{d}t}, \tag{3.10}$$

kde první člen na pravé straně je průmět gravitační síly do směru $\hat{z}$ a druhý je třecí člen daný Epsteinovým zákonem. Zde S je průřez částice, v_T termální rychlost plynu, ρ_plyn jeho objemová hustota a m_P hmotnost částice.

I kdybychom se odvážili aproximovat r konstantou, $\frac{S}{m_\mathrm{P}}$ je nutně funkcí času, neboť se prachová zrna na sebe nabalují. Dle Bertotti a kol. (2003) si ale můžeme dovolit zrychlení $\frac{\mathrm{d}^2 z}{\mathrm{d}t^2}$ zanedbat oproti ostatním členům a studovat tak obálku funkcí, které řeší (3.10). Získáme tak diferenciální rovnici 1. řádu, ve které navíc použijeme rovnici (3.9). Řešením je funkce

$$z\ = z_0 \exp\left(-\frac{GM_*}{r^3}\frac{2\alpha}{3\sqrt{A}}\,t^2\right). \tag{3.11}$$

Částice se tedy budou soustřeďovat ve středové rovině.

Jak prachová zrna narůstají, klesá vliv hydrostatického tlaku na jejich pohyb a přestávají být na plyn vázány. Když narostou na velikost mnohem větší, než je střední volná dráha molekul plynu, přestane být Epsteinův vztah přesný. Tření o plyn začne lépe popisovat Stokesův vztah

$$\vec{a}_\mathrm{tření} = -\frac{C}{2}\rho_\mathrm{plyn} S|\vec{v}|\vec{v}, \tag{3.12}$$

kde C označuje koeficient tření, S průřez částice, v její rychlost relativně k plynu o hustotě ρ_{plyn}. Kvalitativně se ale chování příliš nezmění. Při monotónních gradientech částice dál narůstají, sesedají k rovníkové rovině a spirálují směrem ke hvězdě.

Dle Armitage (2010) na 1 au do 10^3 let částečky narostou na velikost 1 mm a sesednou až k rovníkové rovině. Jiné práce (třeba Brož a Šolc 2013) odhadují ještě rychlejší nárůst. Nárůst zrn v každém případě probíhá na kratší časové škále než je obvyklá životnost disku (řádově 10^6 let).

Balvany mohou kolizemi dál narůstat, strhávat s sebou okolní plyn a začít ho perturbovat. Vznikne-li v radiálním profilu tlaku plynu lokální maximum, budou se v jeho okolí soustřeďovat balvany, protože při obíhání v gravitačním potenciálu jsou nuceny spirálovat ve směru gradientu tlaku. Koncentrace balvanů v těchto místech vede na častější srážky a ty zase na rychlejší růst. Mezi 10^5 a 10^7 lety tak mohla vzniknout planetární embrya, jejichž celková hmotnost mohla být dostatečně velká pro vznik takových planet, jaké dnes pozorujeme (viz Johansen a kol. 2007).

3.3 Viskozita

Jelikož protoplanetární disk je hydrodynamický systém, určitou roli v jeho struktuře hraje viskozita. Fyzikální definice dynamické viskozity μ je založená na tečném napětí τ působícím na vrstvy tekutiny, když po sobě tečou rozdílnými rychlostmi v:

$$\tau_{ij} = \mu \frac{\partial v_i}{\partial x_j}. \tag{3.13}$$

Bývá praktické dynamickou viskozitu normovat hustotou tekutiny a pracovat s viskozitou kinematickou

$$\nu \coloneqq \frac{\mu}{\rho}. \tag{3.14}$$

Pohyb tekutiny v úplnosti popisují Navierovy–Stokesovy rovnice. Pokud pomocí nich chceme popisovat pohyb elementu tekutiny, stačí vzít divergenci tenzoru napětí (3.13). Dostaneme tak člen $\nu\nabla^2 v$, resp. pokud provedeme úplnou analýzu pohybů, které jsou pro tekutiny možné, člen $\frac{1}{\rho}[\nabla\cdot(\mu\nabla v) + \nabla(\mu_V + \frac{1}{3}\mu)\nabla\cdot v]$. V druhém případě kromě dynamické viskozity vystupuje ještě obdobný člen μ_V — objemová dynamická viskozita, která popisuje odpor tekutiny proti její expanzi či kontrakci.

Interpretací viskozity je vícero. Můžeme o ní přemýšlet jako o vnitřním tření, které přetváří pohyb tekutiny v teplo. Pokud jde o princip viskozity, je vhodná představa náhodných odchylek rychlostí částic od makroskopických proudnic. Představme si dvě vrstvy tekutiny, které po sobě tečou, vrchní b.ú.n.o. doleva, spodní doprava. Všechny částice jistě nemají přesně rovnoběžné trajektorie. Elektromagnetické interakce či tepelný šum nutně způsobí, že alespoň některé z částic budou vychýleny ze své (například vrchní) vrstvy do té protisměrné (spodní). Tam se nutně srazí s částicemi tekoucími v protisměru, zpomalujíc tak tok tekutiny.

Vzhledem k jednotkám tečného napětí (Pa), které jsou identické i s jednotkami toku hybnosti, se dá o viskozitě též mluvit jako o difuzi. Nikoli však tepla, ale

hybnosti. V rovnici (3.13) hraje koeficient viskozity obdobnou roli, jako tepelná vodivost v rovnici přenosu tepla.

Zdrojem viskozity bývají typicky mezimolekulární síly, mluvíme pak o molekulární viskozitě. Molekulová viskozita plynů závisí na teplotě. Navzdory intuici je pro plyny charakteristický nárůst viskozity s nárůstem teploty. Statistický přístup s přiblížením pevných elastických koulí jakožto částic plynu vede na $\mu \propto \sqrt{T}$, viz třeba učebnice Chapman a kol. (1995). Přesnější závislost zahrnující též molekulové interakce uváděná například v práci Sutherland (1893) vede na

$$\mu \propto \frac{\sqrt{T}}{1 + \frac{C}{T}}, \tag{3.15}$$

kde C je konstanta.

Mezimolekulární síly v protoplanetárních discích bývají relativně malé a na pohyb planet má proto molekulární viskozita malý vliv (viz Armitage 2010). Tyto síly ovšem nejsou to jediné, co může v tekutině způsobovat přenos hybnosti či disipaci energie. Na stopu dalšího zdroje nás přivede Reynoldsovo číslo.

Reynoldsovo číslo je definované vztahem

$$\mathrm{Re} := \frac{UL}{\nu}, \tag{3.16}$$

kde U je charakteristická rychlostní škála a L charakteristická rozměrová škála v systému. Re charakterizuje mód toku tekutiny; pro $\mathrm{Re} \ll 1$ hraje molekulová viskozita tak významnou roli, že proudění je laminární. Naopak u velkých Re, tj. $\mathrm{Re} \gg 1$, se dá předpokládat turbulentní tok. Pro protoplanetární disky se pohybujeme v režimech, kde Re může být alespoň o tři řády (Lynden-Bell a Pringle 1974) nebo mnohem větší.

Není proto nerozumné předpokládat, že protoplanetární disky jsou turbulentní[1]. Víry ovšem makroskopicky míchají tekutinu, transportujíce plyn z jedné oblasti do oblastí sousedních. Efektivně jde o podobný proces jako u molekulární viskozity. Tato vírová viskozita byla popsána v práci Shakura a Sunyaev (1973) pomocí parametru α:

$$\nu = \alpha c_s H, \tag{3.17}$$

kde c_s označuje rychlost zvuku a H vertikální škálu disku. Vírová viskozita je vztažená k rozměrům samotných vírů, které nemohou být větší, než nejmenší rozměr disku, což je vertikální škála H a rychlostmi, které se v disku mohou vyskytovat čili rychlostí zvuku (protože nadzvukový pohyb by vzhledem k disipaci způsobené rázovými vlnami brzy zanikl). Jelikož $c_s \propto \sqrt{T}$, i tento vztah implikuje rostoucí viskozitu pro rostoucí teploty. Reálný parametr α je obvykle číslo menší nebo rovno jedné, dle Schobert a kol. (2019) řádově 10^{-2}.

Důsledky viskozity jsou víceré. V první řadě disipuje kinetickou energii. Teplo vzniklé diferenciálním tokem je úměrné kvadrátu střihu tekutiny. V protoplanetárním disku je hlavním zdrojem střihu gradient gravitačního potenciálu. Kvůli němu musí tekutiny dál od hvězdy téct pomaleji než ty blíž. Jak ukazuje např.

[1] Vysoké Reynoldsovo číslo bez přítomnosti hydrodynamické nestability nestačí jako důkaz přítomnosti turbulencí, viz například diskuzi v práci Pringle (1981). Dle Brož (2020) však disky nejsou na nestability chudé

Dullemond (2013), ohřev na jednotku plochy disku je

$$Q_+ = \Sigma\nu \left(r \frac{\mathrm{d}\Omega_\mathrm{K}}{\mathrm{d}r} \right)^2 , \tag{3.18}$$

přičemž Ω_K je keplerovská úhlová frekvence. Teplo se v disku samozřejmě nehromadí, ale je vyzařováno jako tepelné záření pryč. Viskozita tedy zprostředkovaně nutí plyn akretovat na hvězdu.

Za druhé ale viskozita stojí za transportem momentu hybnosti makroskopických objektů (balvanů, planetesimál...) na plyn. Ty pak ochuzeny o moment hybnosti migrují k centrální hvězdě, zatímco plyn o moment hybnosti obohacený se od hvězdy vzdaluje.

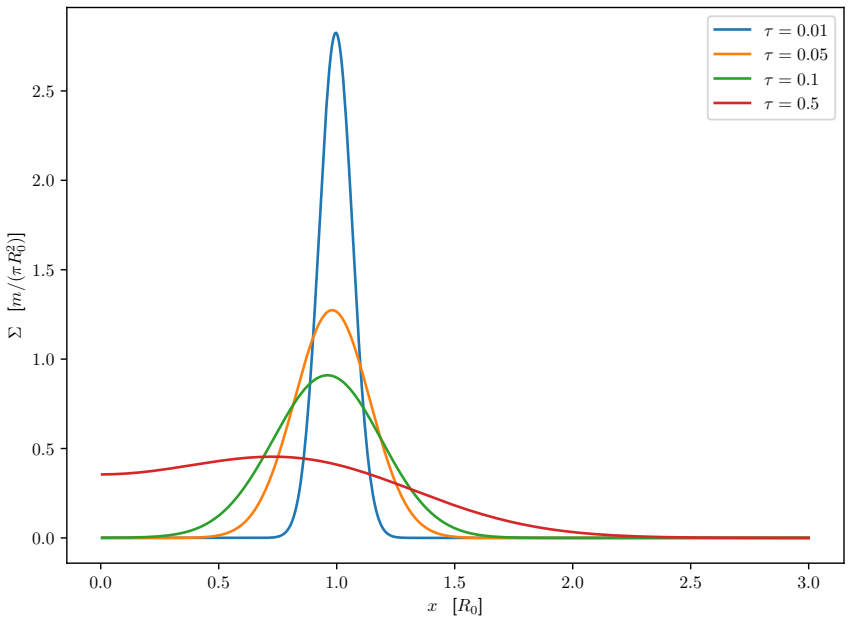
Třetí vliv viskozity popisuje práce Pringle (1981). Špičatá maxima rozdělení hustoty plynu $\Sigma(r)$ se vlivem viskozity musejí vyhlazovat. Jde o řešení diferenciální rovnice kontinuity spolu s rovnicí pohybovou

$$\frac{\partial \Sigma}{\partial t} = \frac{1}{r} \frac{\partial}{\partial r} \left[\frac{1}{r^2 \Omega'} \frac{\partial}{\partial r} \left(-\Omega' \nu \Sigma r^3 \right) \right] , \tag{3.19}$$

Čárka u obecně nekeplerovské úhlové rychlosti Ω zde značí derivaci podle r. Začínáme-li například s pouhým prstencem plynu na poloměru R_0, s funkcí podobnou $\Sigma(r) = \frac{m_\mathrm{prstence}}{2\pi R_0} \delta(r - R_0)$, dosazením této počáteční podmínky a $\Omega = \Omega_\mathrm{K}$ do rovnice (3.19) dostaneme řešení tvaru

$$\Sigma(x, \tau) = \frac{m}{\pi R_0^2} \frac{1}{\tau \sqrt[4]{x}} \exp \left(-\frac{1 + x^2}{\tau} \right) I_{\frac{1}{4}} \left(\frac{2x}{\tau} \right) . \tag{3.20}$$

Proměnné x a τ jsou normované r a t; $x = \frac{r}{R_0}$, $\tau = \frac{12\nu t}{R_0^2}$. Funkce $I_{\frac{1}{4}}$ zde značí modifikovanou Besselovu funkci.



Obrázek 3.1: Vývoj povrchové hustoty plynu v důsledku viskozity dle rovnice (3.20). Graf zachycuje plyn v různých normovaných časech τ $[t = \tau R_0^2/(12\nu)]$, jak se vyvíjí od limitně nekonečně tenkého prstence s poloměrem R_0. Na osách jsou normovaný poloměr x v jednotkách R_0, a povrchová hustota normovaná na $m/(\pi R_0^2)$. Pozorujeme postupné zahlazování původního profilu hustoty důsledkem přerozdělování hmotnosti (a momentu hybnosti) zejména do vnitřní ale i vnější části disku.

3.4 Opacita

Zabývejme se nyní zářením. Prochází-li diskem záření, interaguje s atomy a molekulami plynu či pevnými částicemi. Nehledě na to, o jaký proces se jedná, můžeme po průchodu plynem a prachem zaznamenat extinkci, tj. pokles intenzity záření. Tento pokles motivuje zavedení optické hloubky τ diferenciálním vztahem

$$\tau := -\frac{1}{I_\nu}\frac{\mathrm{d}I_\nu}{\mathrm{d}x}, \tag{3.21}$$

kde I_ν označuje monochromatickou intenzitu záření o frekvenci ν. Intuitivně zaznamenává bezrozměrná veličina τ počet středních volných drah, které by foton urazil v prostředí, kterým by prolétal.

Díváme-li se na libovolný objekt, vidíme typicky do optické hloubky 1. Fotony z větší hloubky nedoletí bez interakce až k pozorovateli. Objekty s optickou hloubkou $\tau \ll 1$ jsou průhledné (neboť fotony v průměru interagují až po uražení jedné střední volné dráhy) a nazýváme je opticky tenkými. Naopak objekty s $\tau \gg 1$ jsou neprůhledné a nazýváme je opticky tlustými.

Interakce s látkou jistě musí probíhat častěji, pokud je látka hustší; v hustší látce je pravděpodobnější, že se v dráze fotonů budou vyskytovat částice látky a tedy budou i častěji nastávat interakce. Je proto vhodné odlišit hustotu látky od její vlastní, na hustotě nezávislé schopnosti filtrovat procházející světlo. Definujeme tedy i opacitu κ_ν

$$\kappa_\nu := -\frac{1}{\rho I_\nu}\frac{\mathrm{d}I_\nu}{\mathrm{d}x}. \tag{3.22}$$

Opacita závisí na stavu látky, neboť právě ten určuje, jaké procesy za extinkcí stojí. To není nic překvapivého. Vezměme si jako příklad naší atmosféru. Při letních teplotách a bezoblačné obloze je v optickém oboru průhledná, vodní pára se jeví jako by byla neviditelná. Jen co teplota v poklesne dostatečně nízko, vytvoří se mlha a průhlednost okamžitě poklesne.

Co se vlastně děje se světlem, které prochází látkou? O jaké interakce jde? Jde o absorpci nebo rozptyl. Oba procesy mohou způsobit, že fotony, které k pozorovateli látkou letěly, už k němu neletí. Nicméně různé typy interakcí jsou různě efektivní a různě efektní. Stručně si alespoň ty nejdůležitější projdeme.

Při interakci s atomy a molekulami plynu jde o mikroskopické procesy:

1. Rayleighův rozptyl na neutrálních částicích,
2. Comptonův rozptyl na nabytých částicích, který mění vlnovou délku fotonu,
3. Thompsonův rozptyl, nízkoenergetická limita Comptonova rozptylu zachovávající vlnovou délku
4. excitace elektronu v atomu a následná reemise, při které se vyzáří stejná vlnová délka, jako byla pohlcena
5. ionizace atomu, což způsobí úplnou absorpci fotonu.

Typickým výsledkem interakce záření s plynem jsou spektra s kontinuem a diskrétními čárami. Díky nim lze využít spektroskopie pro analýzu disků. Například pozorováním spektrální čáry ^{12}CO můžeme získat rychlostní profil plynu v disku.

Tato metoda byla použita pro hledání planet, které můžou být sice na první pohled neviditelné, avšak jejich rušení okolního toku plynu je díky Dopplerově efektu nepřehlédnutelné (Pinte a kol. 2020).

Plyn přispívá k opacitě jen málo. Hlavním zdrojem neprůhlednosti je prach. Jeho interakce se světlem se odvíjí od velikosti zrn. Zrna, která jsou malá oproti vlnové délce záření jsou při průběhu vlny vystavena homogennímu elektromagnetickému poli. Reagují tedy polarizací jako každé jiné dielektrikum a vytváří vlastní elektrické pole. V této Rayleighově limitě není záření vůbec citlivé na průřez zrn, ale na celý objem (Dullemond 2012).

Opacita v Rayleighově režimu vykazuje strmou mocninnou závislost na vlnové délce světla. Pro rozptyl platí

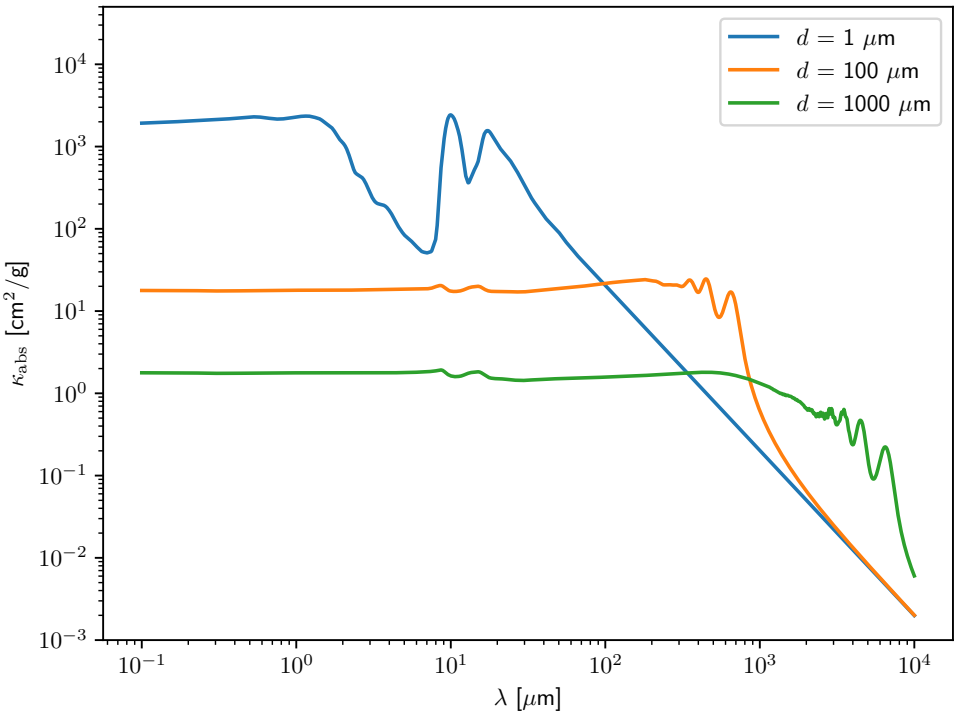
$$\kappa_{\text{Rayleigh}} \propto \frac{1}{\lambda^4}, \tag{3.23}$$

což je ostatně příčinou barvy naší atmosféry. V logaritmicko-logaritmickém grafu opacity v závislosti na vlnové délce očekáváme proto klesající úsečku.

U kratších vlnových délek se začne projevovat absorpce. Přicházející elektromagnetické vlna excituje atomovou mřížku zrna, foton se přeměňuje na fonon a světlo je absorbováno. S rostoucím rozměrem zrn se i zvětšuje množství frekvencí, kterými může být mřížka excitovaná a tudíž se i rozšiřuje mezera ve spektru, kterou zrno zastíní. Výsledkem jsou široké vrcholy na grafu $\kappa(\lambda)$. Fonony samozřejmě v zrnu nezůstanou navěky a jsou přeměněny opět na fotony v tepelném oboru.

Pro makroskopická zrna, resp. taková, jejichž rozměry jsou výrazně větší než vlnová délka světla, musíme vzít v potaz i makroskopické opacitní jevy, tj. geometrický režim. V něm se na fotony můžeme dívat jako na částice, čímž se ztratí závislost na jejich vlnové délce.

Zbývá zdůraznit důležitý fakt: na velikosti zrn opravdu záleží! Velká zrna totiž reprezentují velkou hmotnost. Například jeden kilogram silikátového prachu bude mít očividně jiné optické vlastnosti, půjde-li o extrémně jemný prach zakrývající celé zorné pole, než pokud bude týž materiál v jediném velkém kameni. Větší zrna mívají podstatně *menší* opacitu.



Obrázek 3.2: Závislost absorpční monochromatické opacity κ prachových zrn na vlnové délce λ záření. Zde jde konkrétně o silikátová zrna z amorfního olivínu s 50 % hořčíku a 50% železa o rozměrech $d = 0{,}1\,\mu$m, $d = 10\,\mu$m a $d = 1000\,\mu$m. V levé části grafu ($\lambda < d$) vidíme u všech zrn téměř konstantní průběh, což odpovídá geometrickému režimu. Na pravé straně grafu ($\lambda \gg d$) pozorujeme mocninný pokles, odpovídající Rayleighově limitě. Mezi tím je oblast charakteristická svými širokými vrcholky, kde dochází k excitaci mřížky zrn. Vlnka ve spektru na $\lambda = 10\,\mu$m odpovídá vibračnímu spektru SiO_4, viz práci Dorschner a kol. (1977). Opacity vypočteny kódem na základě Jaeger a kol. (1994), Dorschner a kol. (1995).

4. Simulace toků plynu a přenosu záření

Budeme se zabývat syntetickými protoplanetárními disky. Využijeme k tomu již existující programy Fargo a Radmc3d. Základní, dvourozměrné disky jsou výsledky hydrodynamické simulace programem Fargo a vystupují v práci Chrenko a kol. (2017). Radmc3d, jak název napovídá, pracuje s trojrozměrnými objekty. Ústřední pro tuto práci byla tedy tvorba kódu Faradit, pomocí nějž jsme mohli dvourozměrné výsledky z Farga převést pro Radmc do tří rozměrů.

4.1 Princip Fargo2d a konkretní disky

Fargo je program pro numerické řešení Navierovy–Stokesovy rovnice a rovnice kontinuity pro disk. Započítává vliv gravitace centrálního objektu a protoplanet vnořených v disku. Je na něm založený jeho nástupce Fargo3D. Oba kódy jsou dostupné z webu `http://fargo.in2p3.fr/` a byly použity v mnoha publikacích.

V této práci se zabýváme hlavně výsledky dvourozměrnou verzí. Veličiny, kterými se disk popisuje, jsou vertikálně integrované. Například místo objemové hustoty plynu ρ se používá povrchová hustota $\Sigma = \int_{-\infty}^{\infty} \rho \, \mathrm{d}z$, obdobně $P = \int_{-\infty}^{\infty} p \, \mathrm{d}z$ atd.

S rozšířením Thorin byl kód v práci Chrenko a kol. (2017) použit pro pokročilý popis migrace protoplanet v disku. Rozšíření Thorin (dostupné z `https://sirrah.troja.mff.cuni.cz/~chrenko/`) obohacuje původní model o popis balvanů jako neviskózní tekutiny a zpřesňuje vzájemnou interakci mezi diskem a protoplanetami.

Fargo Thorin (dle Chrenko a kol. 2017) řeší na dvourozměrné polární síti následující rovnice:

1. rovnici kontinuity pro plyn

$$\frac{\partial \Sigma}{\partial t} + \nabla \cdot (\Sigma \vec{v}) = 0, \tag{4.1}$$

2. obdobnou rovnici kontinuity pro balvany

$$\frac{\partial \Sigma_\mathrm{b}}{\partial t} + \nabla \cdot \left(\Sigma_\mathrm{b} \vec{V} \right) = - \left(\frac{\partial \Sigma_\mathrm{b}}{\partial t} \right)_\mathrm{acc}, \tag{4.2}$$

   kde velké $\vec{V}$ odlišuje rychlost balvanů od rychlosti plynu $\vec{v}$ a člen na pravé straně odpovídá akreci balvanů na protoplanety,

3. Navierovu–Stokesovu rovnici pro plyn

$$\frac{\partial \vec{v}}{\partial t} + \vec{v} \cdot \nabla \vec{v} = -\frac{1}{\Sigma} \nabla P + \frac{1}{\Sigma} \nabla \cdot \overset{\leftrightarrow}{T} - \frac{1}{\Sigma} \int_{-\infty}^{\infty} \rho \nabla \varphi \, \mathrm{d}z + \frac{\Sigma_\mathrm{b}}{\Sigma} \frac{\Omega_\mathrm{K}}{\tau} \left(\vec{V} - \vec{v} \right), \tag{4.3}$$

   kde P je vertikálně integrovaný tlak, $\overset{\leftrightarrow}{T}$ tenzor viskózního napětí popsaný v práci Masset (2002), φ gravitační potenciál, Ω_K keplerovská úhlová rychlost a τ Stokesovo číslo charakterizující vazbu balvanů na plyn,

4. Navierovu–Stokesovu rovnici pro balvany

$$\frac{\partial \vec{V}}{\partial t} + \vec{V} \cdot \nabla \vec{V} = -\frac{1}{\Sigma_\mathrm{b}} \int_{-\infty}^{\infty} \rho_\mathrm{b} \nabla \varphi \, \mathrm{d}z - \frac{\Omega_\mathrm{K}}{\tau} \left(\vec{V} - \vec{v} \right) \tag{4.4}$$

a

5. rovnici energetické bilance

$$\frac{\partial E}{\partial t} + \nabla \cdot (E\vec{v}) = -P \nabla \cdot \vec{v} + Q_\mathrm{visc} + Q_\mathrm{irr} + Q_\mathrm{acc} - Q_\mathrm{rad}, \tag{4.5}$$

kde na pravé straně vystupují v příspěvky k energii buňky díky viskóznímu ohřevu, ozáření hvězdou a akrecí mínus ztrátu energie vyzářením.

Ve Fargu se navíc předpokládá stavová rovnice pro plyn

$$P = \Sigma \frac{RT}{\mu}, \tag{4.6}$$

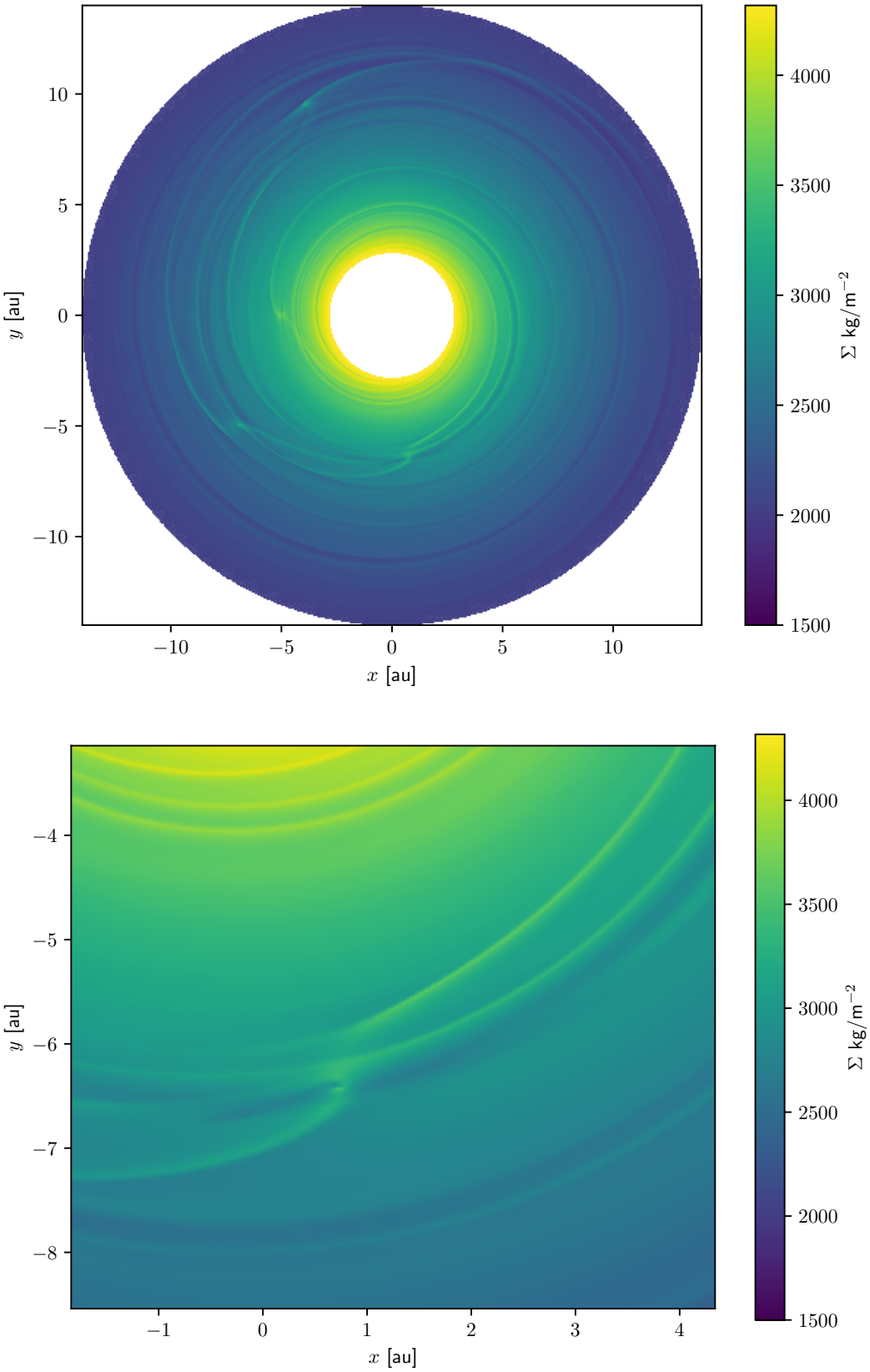
kde R je univerzální plynová konstanta, T teplota plynu a μ je střední molekulová hmotnost.

Simulace disků které v této práci diskutujeme, probíhaly na polární síti v doméně (3,6 au; 14 au) $\times$ $(0, 2\pi)$, která byla diskretizována na 1024 pravidelných radiálních a 1536 pravidelných azimutálních sektorů.

V této práci se budeme zabývat konkrétním protoplanetárním diskem z práce Chrenko a kol. (2017). Jeho počáteční podmínky byly voleny tak, aby se podobaly podmínkám, které nejspíš nastávaly ve Sluneční soustavě během jejího vzniku. Konkrétně jde o předepsaný profil povrchové hustoty plynu

$$\Sigma = 750\,\mathrm{g/cm^2} \sqrt{\frac{1\,\mathrm{au}}{r}}. \tag{4.7}$$

Do disku poté autoři přidali balvany o povrchové hustotě předepsané dle počátečního toku hmotnosti systémem $\dot{M} = 2 \cdot 10^{-4}\, M_\oplus$, 4 planetární embrya s hmotnostmi $M = 3M_\oplus$ a nechali disk vyvíjet. Tento konkrétní disk je označen jako Case III: embrya mohla interagovat spolu navzájem i s diskem, mohla akretovat balvany a tím sebe i okolí zahřívat. Po $5 \cdot 10^4$ letech byla v disku výrazná spirální ramena a embrya prošla z důvodu tzv. hot–trail efektu značnou změnou excentricity a hlavní poloosy. Disk pro čas $t = 5 \cdot 10^4$ let je ukázán na obrázku 4.1.



Obrázek 4.1: Diskutovaný protoplanetární disk Case III z práce Chrenko a kol. (2017). Jde o disk získaný hydrodynamickou simulací kódem Fargo Thorin. V disku jsou vnořená čtyři planetární embrya o počáteční hmotnosti tří Zemí na pozicích s velkými poloosami $a_1 = 5\,\mathrm{au}$, $a_2 = 6{,}7\,\mathrm{au}$, $a_3 = 8{,}4\,\mathrm{au}$ a $a_4 = 10{,}1\,\mathrm{au}$. Embrya obíhají mladé Slunce o hmotnosti $M_* = M_\odot$ a poloměru $R_* = 1{,}5\,R_\odot$, akretují okolní materiál, gravitačně a tepelně interagují s diskem. Simulace probíhala na škále $5 \cdot 10^4$ let a její prostorové rozlišení bylo 1024×1536 radiálních a azimutálních sektorů.

4.2 Princip Radmc3d

Radmc3D (Dullemond a kol. 2012) je program napsaný ve Fortranu90, pomocí kterého se simuluje přenos záření v prachu a plynu. Hlavním autorem je profesor Cornelis P. Dullemond, který se na univerzitě v Hiedelbergu zabývá formováním planet a protoplanetárními disky. Na programu pracují i další dobrovolníci a je ve formě open-source přístupný na GitHubu (`https://github.com/dullemond/radmc3d-2.0`).

I když je rozdíl mezi Radmc3d a jeho dvourozměrným předchůdcem Radmc, v této práci se budeme zabývat výhradně trojrozměrnou verzí a pro jednoduchost ji budeme nazývat pouze Radmc.

Radmc je primárně určen pro astrofyzikální výpočty a, jak je pro astrofyziku typické, používá soustavu jednotek CGS. Veškeré vstupní soubory musí obsahovat veličiny právě v těchto jednotkách. Jediná výjimka je vlnová délka, která se zapisuje v mikrometrech.

Soubory mohou být textové nebo binární, údaje zapisujeme výhradně ve formátu 8-bitového `int` pro celočíselné hodnoty. Pro reálné hodnoty lze specifikovat přesnost, my zapisujeme převážně 8-bitové `double` .

Radmc používá při výpočtu přenosu záření buňky. Volíme souřadnou síť, v této práci konkrétně sférickou, a pomocí ní diskretizujeme prostor na konečné buňky. Do vstupního souboru `amr_grid.inp` zapisujeme hranice těchto buněk.

V případě tzv. regulární mříže o $i \times j \times k$ buňkách zapisujeme pouze $(i+1)+$ $+(j+1)+(k+1)$ hranic; v jednom směru po druhém. Například pro sférickou síť nejdříve zapíšeme všechny hranice ve směru $\hat{r}$ v centimetrech, poté hranice ve směru $\hat{\vartheta}$ v radiánech a nakonec hranice ve $\hat{\varphi}$ též v radiánech.

Regulární kartézská mříž *musí* mít ve všech směrech buňky stejně dlouhé. Sférická má naopak výhodu v tom, že takovou podmínku nemusí splňovat. Podporuje tedy jemnou diskretizaci kolem $r = 0$, kde se typicky nachází hvězda, gradienty veličin bývají vysoké a jemné vzorkování je důležité, a naopak hrubou diskretizaci pro velká r, kde jsou typicky teploty, hustoty i jejich gradienty nižší a vzorkování nemusí být jemné. Podobně ve směru $\hat{\vartheta}$ můžeme volit hrubou diskretizaci v opticky tlustém prostředí, typicky v rovníkové rovině disku, jemnější v atmosféře a opět hrubou v téměř prázdném prostoru.

Jak už zkratka v názvu vstupního souboru mříže naznačuje[1], Radmc podporuje i zjemnění sítě, konkrétně pomocí osmistromé sítě (Oct-tree). Princip Oct-Tree je v dělení už existujících buněk. Libovolnou buňku lze rozdělit na 2^n podbuněk (kde n označuje dimenzi prostoru), pro trojrozměrné simulace tedy na 8. Odtud pochází první polovina názvu této metody. „Mateřská“, velká buňka se nazývá větev, malé jsou listy, odtud Tree v názvu. Každý list se opět může stát větví rozdělením na dalších 2^n listů. Fraktálně tedy můžeme získat vysoké rozlišení pro takové části prostoru, které jsou zajímavé nebo problematické. V této práci nebyla použita, avšak pro další zkoumání oblastí poblíž planet je taková síť nanejvýš vhodná.

Když už je připravená mříž hustot plynu a prachu, přichází řada na zahřívání. Radmc zahřívá disk převážně centrální hvězdou. Vysílá z ní fotony pseudonáhodně dle předem zadaného spektra a do pseudonáhodných směrů; jde tedy o metodu Monte Carlo. Foton letící diskem může prolétávat bezúhonně, nebo, v závislosti

[1] AMR je zkratka pro Adaptive Mesh Refinement, tedy adaptivní zjemnění mříže.

na opacitě κ_ν se může rozptýlit do jiného směru, či být absorbován a ihned reemitován do jiného směru s odlišnou vlnovou délkou dle Bjorkman a Wood (2001). Fotony se takhle odráží tam a zpět v mříži dokud nedorazí na její hranici a neopustí model. Teprve tehdy je foton „zničen“ a ze zdroje vyslán nový. V této části simulace se tedy žádný foton nedostane k pozorovateli, neboť je ihned po opuštění disku odstraněn. *Netvoří* se obraz, pouze se disk zahřívá.

Pokud pracujeme s opticky tlustým prostředím, může se v buňkách foton pohybovat překvapivě dlouho, protože je v této fázi simulace uvnitř modelu nezničitelný. Unikne-li konečně z problematické buňky, čeká jej buňka sousední, o které lze předpokládat, že nebude výrazně průhlednější a dlouhý výpočet úniku fotonu z buňky začíná nanovo. Když pak foton z této buňky unikne zpět do buňky předchozí (což u izotropního prostředí představuje asi $\sim 13\,\%$ případů), není obtížné pochopit, že se výpočet výrazně protáhne.

Simulace zářiveho přenosu v disku může kvůli optické hloubce trvat dny. V rámci zachování „upočitatelnosti“, tj. rozumné délky výpočtu na dostupných zařízeních, je vhodné použít tzv. modifikovanou náhodnou procházku (anglicky Modified Random Walk nebo jen krátce MRW). V módu MRW Radmc nepodrobí foton v opticky tlustých buňkách desítkám či stovkám rozptylů, nýbrž pomocí analytického řešení difúzní rovnice (dle Robitaille 2010) předpoví, kde a jak unikne foton z buňky. Jsou-li opticky tlusté buňky rozlehlé, může MRW výpočet zkrátit řádově stokrát.

Víme-li, jakou má disk teplotu, nutně také víme, jak září. Pomocí Radmc poté můžeme vytvořit i syntetický obraz disku na zvolené vlnové délce pomocí metody známé z počítačové grafiky — sledování paprsku (ray tracing).

Sledování paprsku je algoritmus pomocí kterého lze generovat realistické fotografie. Spočívá ve vysílání paprsků ze zkoumaného objektu na pozorovatele[2] (například z pixelů detektoru). Hodnota jasu pixelu bude dána příspěvky paprsků, které na něj dopadají. Radmc se pak podívá, kterými buňkami paprsek prochází, jak silnými jsou zdroji záření a jejich příspěvky sečte. Výsledkem je jas jednoho pixelu. Pixel po pixelu získáme celý obraz.

Takovouto vygenerovanou fotografii budeme nazývat ideální syntetický obraz, neboť, jak jméno naznačuje, jde sice pouze o simulaci (tj. nejde o fotografii oblohy), ale teoreticky by jej za ideálních podmínek (jako jsou nekonečná aparatura, nulový šum, neexistující atmosféra...) sledovat šlo.

4.3 Faradit

V rámci této práce jsme vyvinuli kód Faradit. Název je zkratka anglického Fargo to Radmc Disk Thickener neboli rozšiřovač disků z Farga do Radmc. Jak název napovídá, jeho hlavním účelem je na základě dvourozměrných hydrodynamických simulací Fargo připravit soubory pro trojrozměrnou simulaci přenosu záření kódem Radmc.

V první řadě musí být do paměti načteny parametry hydrodynamické simulace. Konkrétně jde o hranice simulační domény (R_{min} a R_{max}), počet buněk ve směru $\hat{r}$ a $\hat{\varphi}$ (N_{rad} a N_{sec}), efektivní teplotu hvězdy (T_*), její hmotnost (M_*),

[2]Prakticky se většinou vysílají paprsky od pozorovatele na objekt. Můžeme si pak být jistí, že nepočítáme s paprsky, které k pozorovateli vůbec nedoletí.

mimo jiné.

Pole povrchových hustot a teplot plynu v buňkách simulace jsou typicky uložena v souborech `gasdens*.dat` , resp. `gastemper*.dat` . V nich jsou v binárním formátu zapsána v double precision prstenec po prstenci od vnitřního okraje ke vnějšímu. Pro načtení využíváme knihovnu numpy a její proceduru `fromfile` .

Simulace Fargo běžela v doméně 2.8 au $< r <$ 14 au, avšak to není pro naše účely dostatečné a disk musíme extrapolovat. Extrapolací v našem případě nazýváme vytvoření dat pro body v prostoru mimo původní doménu. Konkrétně se zde zabýváme hlavně povrchovou hustotou a teplotou plynu dvourozměrného disku na $r \notin (R_{\min}, R_{\max})$, kde tyto hranice vychází z hydrodynamické simulace Fargo. Tato data nejsou výsledkem simulace, spíše vzdělaného odhadu.

Mají-li být extrapolovaná data fyzikální, očekáváme, že přechod mezi hydrodynamickou simulací a extrapolací bude hladký. Pro tyto účely využíváme nástroj hladkého minima:

Definice 1. *Pro libovolné funkce $f, g : \mathbb{R} \to \mathbb{R}^+, n \in \mathbb{R}^+$ definujeme hladké minimum $S[f,g,n]$*

$$S\ [f,g,n] = \sqrt[n]{\frac{f^n g^n}{f^n + g^n}}. \tag{4.8}$$

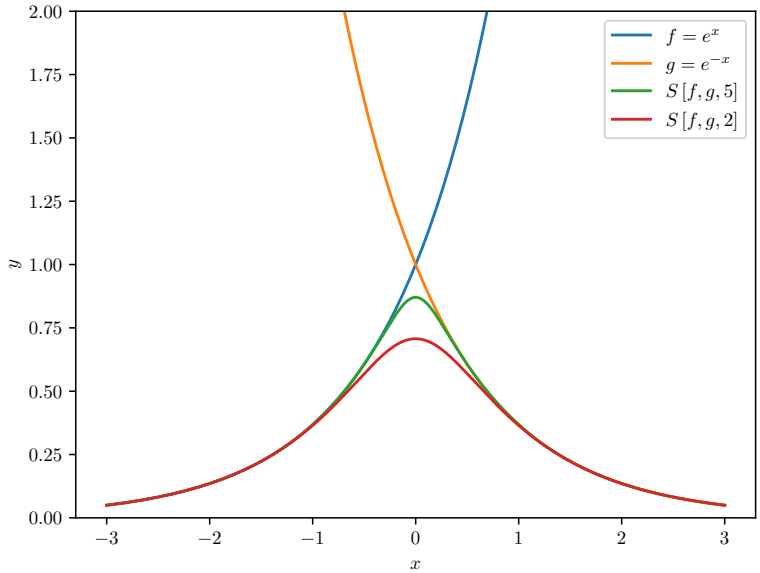
Pro $S[f,g,n]$ očividně platí

$$S \leq \min\{f, g\}\ \forall x \in \mathbb{R}, \tag{4.9}$$

$$f \ll g \Rightarrow S\ \to f, \tag{4.10}$$

$$f = g \Rightarrow S\ = \frac{f}{\sqrt[n]{2}}, \tag{4.11}$$

navíc pokud máme m takové, že $f, g \in \mathcal{C}^m$, tj. mají m spojitých derivací, pak nejhůře $\mathrm{S(x)} \in \mathcal{C}^{\mathrm{m}-1}$. *Reálné číslo n nazveme zhlazovacím parametrem.*



Obrázek 4.2: Příklad zhlazeného přechodu od jedné funkce k druhé. Zde bereme dvě exponenciály $f = e^x$ a $g = e^{-x}$ a přechod mezi nimi zhlazujeme je hladkým minimem, jednou s parametrem hladkosti 5, podruhé s parametrem 2. Čím větší tento parametr je, tím „ostřejší“ je přechod mezi nimi; tím blíž je zhlazená funkce blíže původním f a g.

Předpokládáme, že průběh povrchové hustoty a teploty plynu lze dobře aproximovat některými mocninnými závislostmi. V literatuře se typicky objevuje například $\Sigma \approx \frac{C}{\sqrt{r}}$, viz třeba Andrews a kol. (2009). V hydrodynamických simulacích našich disků jsou profily veličin lépe odhadnuty vztahy s konstantami C, D:

$$\Sigma \approx \frac{C}{\sqrt{r}^3}, \tag{4.12}$$

$$T \approx \frac{D}{\sqrt{r}}. \tag{4.13}$$

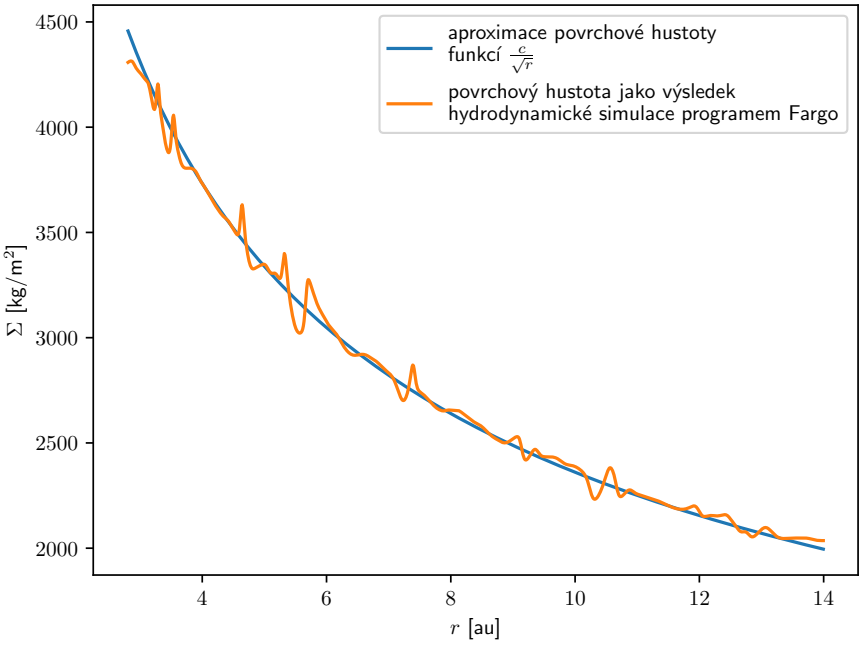
Ve vnitřní části disku, tj. pro $r < R_{\mathrm{min}}$, kde $R_{\mathrm{min}} = 2.8\,\mathrm{au}$ je hranice domény hydrodynamické simulace, přirozeně nemáme žádné simulované hodnoty hustot a teplot. Pro přenos záření však není přípustné, aby disk nespojitě začínal až na 2.8 au. Volíme tedy analytický přepis povrchové hustoty a teploty plynu. Předpokládáme

$$\Sigma \approx C \exp\left[-\left(\frac{r_1}{r}\right)^2\right], \tag{4.14}$$

$$T \approx T_{\mathrm{eff}} \sqrt{\frac{R_*}{r}}, \tag{4.15}$$

kde C a r_1 jsou konstanty. V našich discích volíme $r_1 = 0.1$ au, C volíme vhodně pro hladké navázání.

Přechod mezi hodnotami z hydrodynamického modelu a těmito prodlouženími zhlazujeme pomocí hladkého minima. Pro povrchovou hustotu plynu je vhodné volit multiplikativní konstantu C tak, aby nebylo navázání nespojité (viz vlastnost



Obrázek 4.3: Povrchová hustota hydrodynamických modelů disků a jejich fit funkcí (4.12). Původní Σ je zde vyčíslena ve všech 1024 bodech. Je vidět, že průběh je velmi podobný předepsané funkci $\frac{c}{\sqrt{r}}$, odchylky od ní jsou způsobeny perturbacemi planetami v disku; pozorujeme řez spirálními vlnami v disku.

hladkého minima pro funkce s podobnými a rozdílnými hodnotami). Multiplikativní konstanta nezmění kvalitativně „kvadraticky exponenciální“ pokles hustoty pro $r \to 0$.

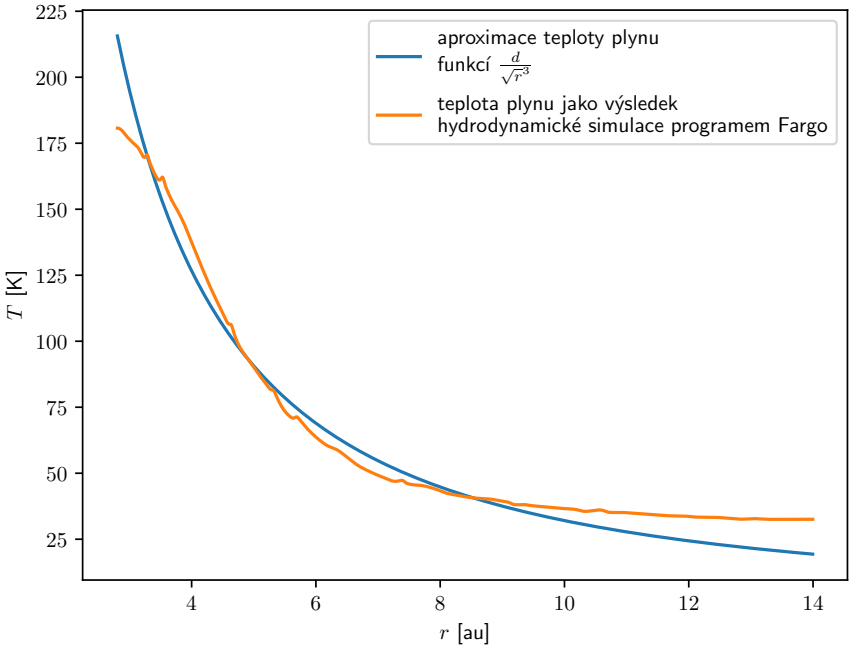
Pro vnější část disku jsme se inspirovali prací Kwon a kol. (2011); pro povrchovou hustotu plynu ve vzdálených částech disku předpokládáme

$$\Sigma \approx c \exp\left[\left(-\frac{r - R_{\max}}{r_2}\right)^2\right], \tag{4.16}$$

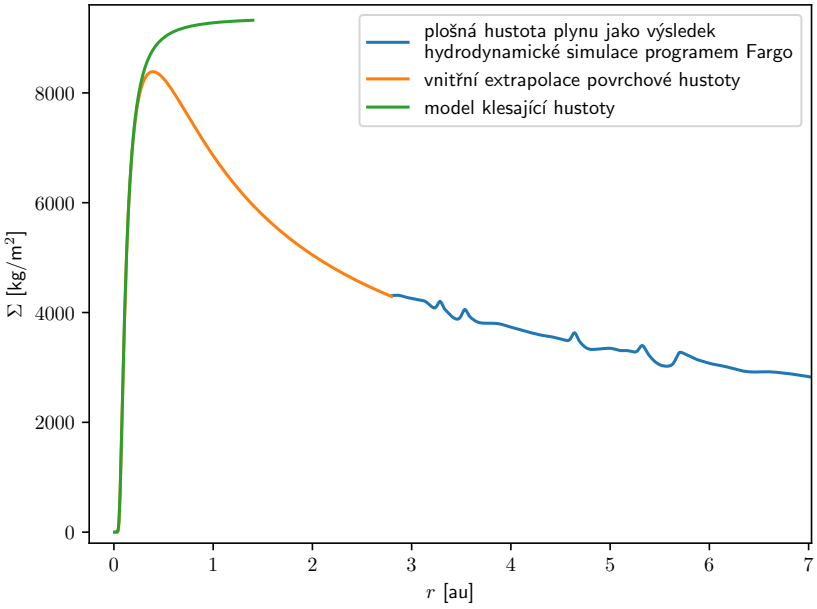
kde c, r_2 jsou konstanty. Pro naše disky volíme $r_2 = 3\,\mathrm{au}$. Pro profil teploty plynu ve vnějších částech disku volíme

$$T \approx d \exp\left[\left(-\frac{r - R_{\max}}{r_2}\right)^2\right], \tag{4.17}$$

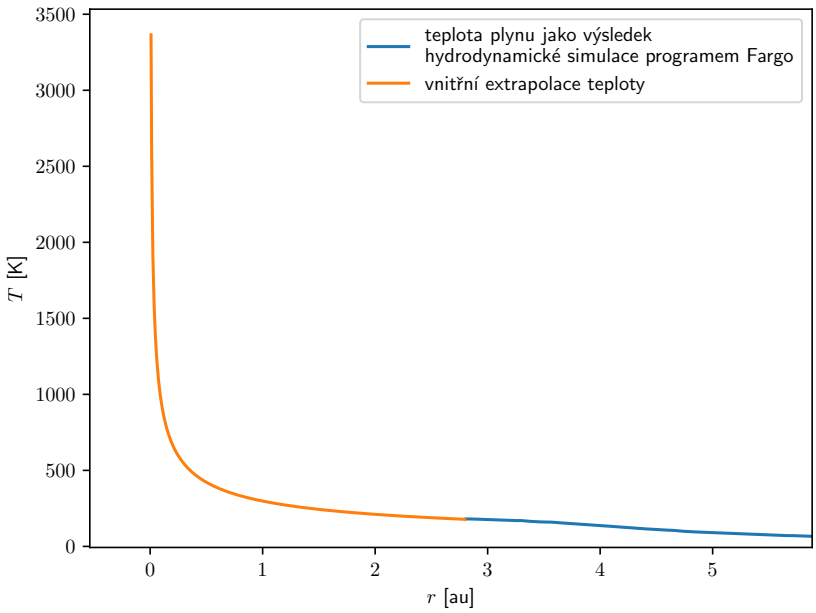
r_2 je jako výše a d je konstanta. Přechod mezi výstupem hydrodynamického modelu a tímto analytickým předpisem opět zhalzujeme hladkým minimem. Podobně, jako u vnitřní části disku kvůli vlastnostem hladkého minima musíme pro spojitost volit takové multiplikativní konstanty c, d, abychom neměli na přechodu měřitelnou nespojitost. Kvalitativně se tím nezmění „kvadraticky exponenciální“ průběh v r.



Obrázek 4.4: Teplota plynu hydrodynamických modelů disků a jejich fit funkcí (4.13). Oproti povrchové hustotě Σ zde pozorujeme výraznější nesoulad s aproximací. Citlivost průběhu teploty na změnu opacity způsobuje odchýlení od mocninného profilu, který je přeci jen velmi jednoduchý.



Obrázek 4.5: Povrchová hustota plynu jakožto výsledek hydrodynamické simulace kódem Fargo (v modrém) je extrapolována směrem ke hvězdě (v oranžovém). Metoda spočívá ve fitování původních modrých dat funkcí $\frac{c}{\sqrt{r}}$, kterou prodloužíme až ke hvězdě a poté ji spojíme s požadovaným kvadraticky exponenciálním poklesem (v zeleném) dle rovnice (4.14) a spojení zhladíme hladkým minimem. Výsledkem jsou data, která respektují jak původní mocninou závislost, tak klesající povahu hustoty poblíž centrální hvězdy



Obrázek 4.6: Teplota plynu jako výsledek dvourozměrné hydrodynamické simulace kódem Fargo (v modrém) je extrapolována směrem ke hvězdě mocninnou funkcí (v oranžovém) dle rovnice (4.15). Tato volba vede na správnou teplotu na povrchu hvězdy, má průběh odpovídající odhadům dle práce Flock a kol. (2016) a zároveň díky okrajových podmínek simulace Fargo navazuje na původní disk.

Pro zmírnění výpočetní náročnosti simulace Radmc a zmenšení jeho vstupních souborů provádíme zmenšení počtu sektorů v radiálním i azimutálním směru faktorem $2^n, n \in \mathbb{N}$. Hodnoty veličin v nových buňkách jsou získány metodou zmenšení lokálními průměry. Každou oblast o $n \times m$ buňkách nahrazujeme jednou buňkou o průměrné hodnotě povrchové hustoty/teploty plynu. Navíc součástí této redukce volíme novou diskretizaci v radiálním směru. Volíme síť v r logaritmickou, tj. získáváme jemnou diskretizaci poblíž hvězdy a hrubší na okraji domény.

Při přenosu záření je výrazně významnější hustota prachových zrn než hustota plynu. Převod mezi nimi provádíme přenásobením faktorem f_{G2D}[3], tzv. poměrem prachu ku plynu. Platí

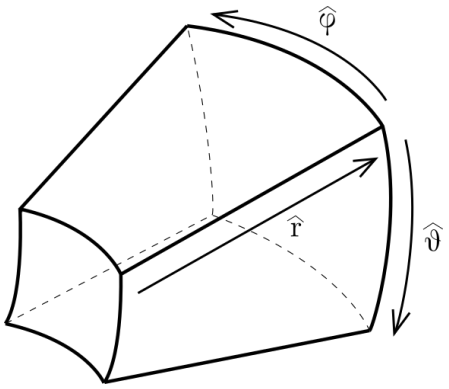
$$\rho_{\mathrm{prach}} = f_{\mathrm{G2D}} \rho_{\mathrm{plyn}}. \tag{4.18}$$

Dá se očekávat, že ve vertikálním směru *nebude* f_{G2D} konstantní; ve středové rovině se typicky usazují zrna a balvany (viz například Armitage (2010)). V této práci používáme aproximaci: naše f_{G2D} není funkcí z, vztah (4.18) tedy používáme i pro Σ.

Je obvyklé používat konstantní $f_{\mathrm{G2D}} = 0.01$, viz například Andrews a kol. (2009). Prostorové rozložení f_{G2D} však může být velmi netriviální například kvůli vzniku makroskopických struktur (prstence, spirální ramena...) nebo tření prachu o plyn. Profilem f_{G2D} v plochém disku se zabývá například Birnstiel a kol. (2012). I z této práce však vyplývá, že $f_{\mathrm{G2D}} = 0.01$ je přijatelné zjednodušení až po $r = 100$ au.

[3]Index „G2D“ je odkaz na anglické gas-to-dust ratio, kterýmžto názvem se tento faktor typicky v literatuře objevuje

Výsledek hydrodynamické simulace programu Fargo je dvourozměrný, všechny extenzivní veličiny jsou vertikálně naintegrované. Pro rozšíření disku do třetího rozměru vytváříme pomocí Faraditu trojrozměrnou mříž ve sférických souřadnicích pro Radmc. Délky hran buněk ve směrech $\hat{r}$ a $\hat{\varphi}$ jsou zděděny po plochém disku, diskretizaci ve směru $\hat{\vartheta}$ lze volit manuálně. Výchozí diskretizace je 100 buněk v $\hat{\vartheta}$ směru. Volíme ji navíc logaritmickou — ve středové rovině (kde předpokládáme velkou optickou tloušťku a tedy i nízkou teplotu) hrubou, v atmosféře jemnou.



Obrázek 4.7: Nákres buňky ve sférických souřadnicích, se kterými pracuje Radmc a do kterých se zapisují objemové hustoty a teploty prachu. Naznačeny jsou směry $\hat{r}$, $\hat{\vartheta}$ a $\hat{\varphi}$. Z takových buněk se skládá celá síť pro Radmc, typicky jsou její rozměry malé a rozměry délky stěn na její v r spodní a horní hranici jsou velmi podobné. Tyto spodní a horní hranice v různých souřadnicích označujeme indexem inf, resp. sup.

Faradit při zápisu souborů pro kontrolu vypisuje šířky, výšky a hloubky největších a nejmenších buněk, aby uživatel mohl posoudit, zda nejsou velikosti buněk v některém ze směrů řádově odlišné. Jelikož Radmc podporuje zrcadlovou symetrii podle rovníkové roviny definované vektory $\hat{x}$ a $\hat{y}$, generují se pouze buňky nad touto rovinou. Hranici úhlu ϑ, tj. nejmenší úhel braný v potaz, lze Faraditu zadat, výchozí hodnotu $\vartheta_{\min}$ volíme podle optické hloubky τ. Požadujeme, aby

$$\max_{\varphi,r}\left\{\int_{z=r\sin\vartheta_{\min}}^{\infty}\rho\kappa\,\mathrm{d}z\right\}<10^{-3},$$

tedy tak, abychom nezapisovali až takové buňky, které by přispěly nejvýše 10^{-3} do optické hloubky. Nezahlcujeme tedy paměť relativně prázdným prostorem vysoko nad diskem.

Pro výpočet optické hloubky je potřeba znát objemovou hustotu prachu ρ v prostoru. Kód Faradit má pro tyto účely volný parametr: rozšiřovací funkci $\rho = \rho\left(z, r_{\mathrm{eq}}, \Sigma(r_{\mathrm{eq}}), T(r_{\mathrm{eq}}), M_*\right)$. Veličina r_{eq} označuje projekci r do rovníkové roviny. Standardně předpokládáme vertikálně adiabatický disk, tj.

$$\rho = \frac{\Sigma}{\sqrt{2\pi}H}\exp\left(-\frac{z^2}{2H^2}\right), \tag{4.19}$$

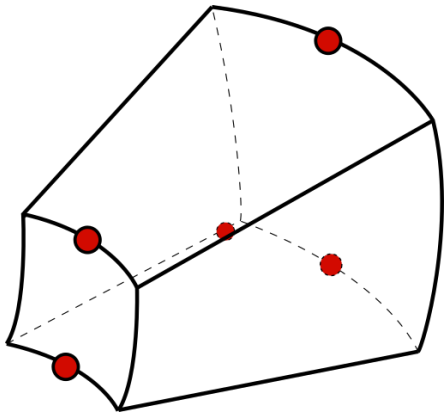
kde výšková škála $H = \frac{c_{\mathrm{c,iso}}}{\Omega_{\mathrm{K}}}$. Zde $c_{\mathrm{c,iso}}$ je isothermální rychlost zvuku a Ω_{K} Keplerovská úhlová rychlost. Za předpokladu ideálního plynu můžeme $c_{\mathrm{c,iso}}$ odhadnout vztahem $c_{\mathrm{c,iso}} = \sqrt{\frac{\mathcal{R}T}{\mu}}$, kde $\mathcal{R}$ je plynová konstanta, T je teplota plynu a μ střední molekulová hmotnost.

Hodnoty $\Sigma(r_{\mathrm{eq}})$ a $T(r_{\mathrm{eq}})$ jsou pro obecnou souřadnou soustavu získány dvourozměrnou lineární interpolací. Zde se ale projevuje efektivita volby sférické souřadné sítě. Projekce středů trojrozměrných buněk do rovníkové roviny jsou oproti středům původních, dvourozměrných buněk posunuty pouze ve směru $\hat{r}$. Stačí tedy provádět pouze jednorozměrnou lineární interpolaci.

Vytvořenou mříž poté vyplňujeme objemovou hustotou prachu. Do vstupních souborů by bylo nejpřesnější zapisovat průměrnou hustotu každé buňky, tedy

$$\rho_{\text{pro zápis}} = \frac{1}{V} \iiint\limits_{\text{buňka}} \rho \, \mathrm{d}V. \tag{4.20}$$

Tento integrál je ale zbytečně výpočetně náročný. V rámci optimalizace počítáme pouze průměr objemové hustoty ze středu každé buňky a ze středů čtyř jejích hran. Volíme pouze ty hrany, které jsou vždy na spodní a horní radiální hranici buňky. Právě v těchto směrech se buňka vychyluje od dvourozměrné sítě disku z Farga. Naopak ve směru $\hat{\varphi}$ jsou i trojrozměrné buňky stejně diskretizované, jako v původní simulaci, předpokládáme tedy, že počítání průměru v tomto směru není nutné.



Obrázek 4.8: Nákres jedné buňky mříže pro Radmc. Červené kruhy znázorňují pět bodů, ze kterých se provádí průměr objemových hustot prachu pro zápis: čtyři na hranách buňky (tedy na $\vartheta_{\mathrm{sup}}, \vartheta_{\mathrm{inf}}, r_{\mathrm{inf}}$ a r_{sup}), jeden v jejím středu (tedy na souřadnicích s indexy med).

Tato pole — mříž a objemovou hustotu prachu — poté zapisujeme do souborů `dust_density.inp` a `amr_grid.inp`. Pomocí Faraditu je můžeme zapisovat v textovém (vhodné pro ladění), nebo binárním, kompaktnějším formátu. Oba tyto soubory jdou i pomocí procedur Faraditu `radmc_read` pro vizualizaci načíst do paměti.

Pro použití Radmc je ještě nutné zapsat tři další soubory:

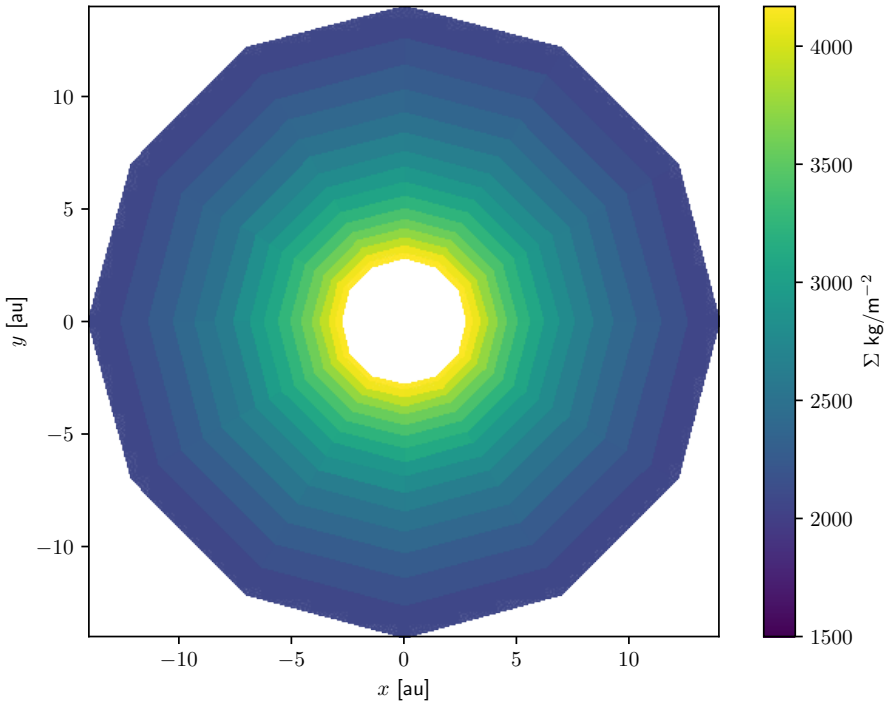
1. Soubor `stars.inp` s polohou hvězdy, hmotností a spektrem. Spektrum typicky volíme jako spektrum černého tělesa.

2. Soubor `wavelength_micron.inp` s výpisem vlnových délek v mikronech, se kterými má Radmc počítat při přenosu záření. Používáme předepsaný seznam vlnových délek.

3. Opacitní soubor `dustkappa_*.inp`, nebo `dustkapscatmat_*.inp` s opacitami prachu na vlnových délkách ze souboru `wavelength_micron.inp`.

Faradit disponuje také řadou vizualizačních procedur, které umožňují detailně v jednom i více rozměrech vykreslovat relevantní vlastnosti v disku.

Ihned po načtení dat z Farga je vhodné vykreslit si obě načtená pole, teplotní i hustotní, pomocí procedur `plot_fargo_*` . Pomocí procedury knihovny `matplotlib` — `pcolormesh` se vykreslí dvourozměrné teplotní mapy povrchové hustoty a teploty plynu. U radiálního průběhu teploty jsou často jemnější struktury utopeny v teplotním gradientu. Pomocí parametru Faraditu lze od celé teploty odečíst teplotní gradient, což zvýrazní menší změny v teplotě.

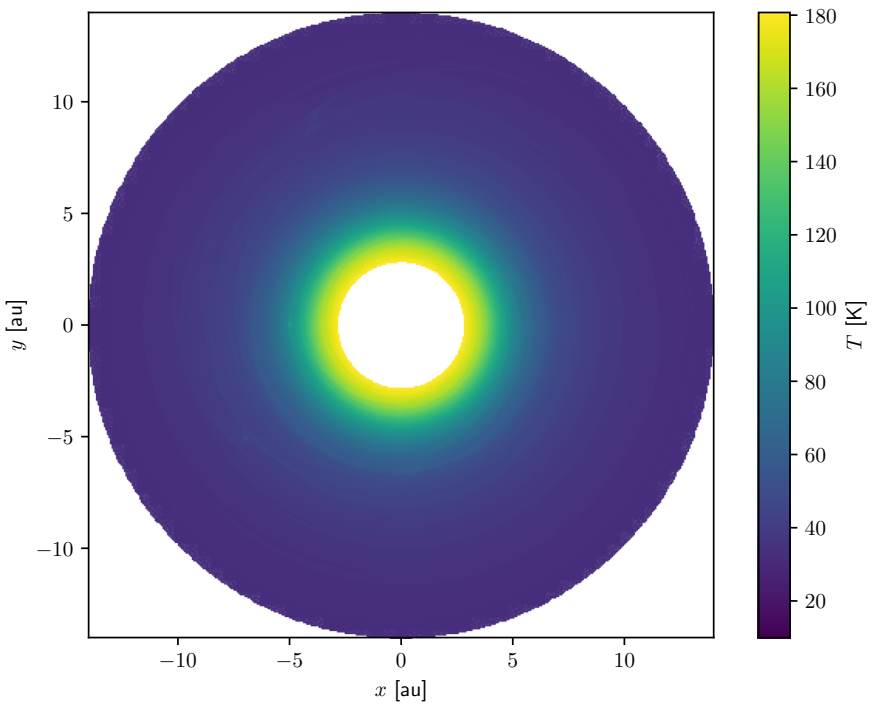


Obrázek 4.9: Příklad vykreslení povrchové hustoty plynu pomocí procedury `plot_fargo_gasdens` zkoumaného disku. Relaxací bylo dosaženo rozlišení 16 radiálních na 12 azimutálních buněk. Takto hrubě diskretizovaný disk je vhodný pro ladění modelů, neboť simulace přenosu záření jím je relativně rychlá.

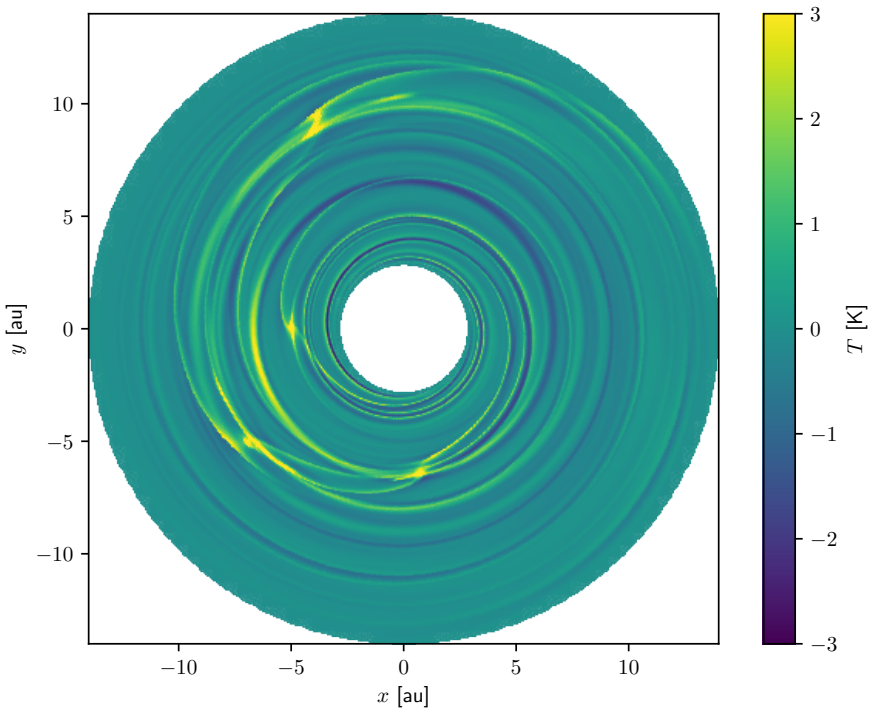
Obdobné procedury, `plot_density_slice` a `plot_T_slice` , zkoumají veličiny ve vertikálním průřezu diskem na zvoleném azimutálním úhlu φ. Procedura `plot_T_slice` je obzvlášť vhodná pro prohlížení disků *po* proběhnutí simulace přenosu záření pomocí Radmc. Obě procedury lze spouštět s parametrem `logarithmic_scale = True` , aby vykreslovaly v log. škále.

Pro přesnější, kvantitativní rozbor disku jsou vhodné procedury Faraditu `plot_density_profile` a `plot_T_profile` . Obě mají mimo jiné argument `along` , který určuje podél které přímky mají být profily vykresleny. Často užívaná jsou klíčová slova `ray` a `perpendicular` . Klíčové slovo `ray` tedy paprsek je pro vykreslení podél polopřímky procházející středovou rovinou s počátkem v centrální hvězdě. Další možnost, `perpendicular` neboli kolmý, způsobí vykreslení podél přímky procházející kolmo středovou rovinou ve vybraném bodě.

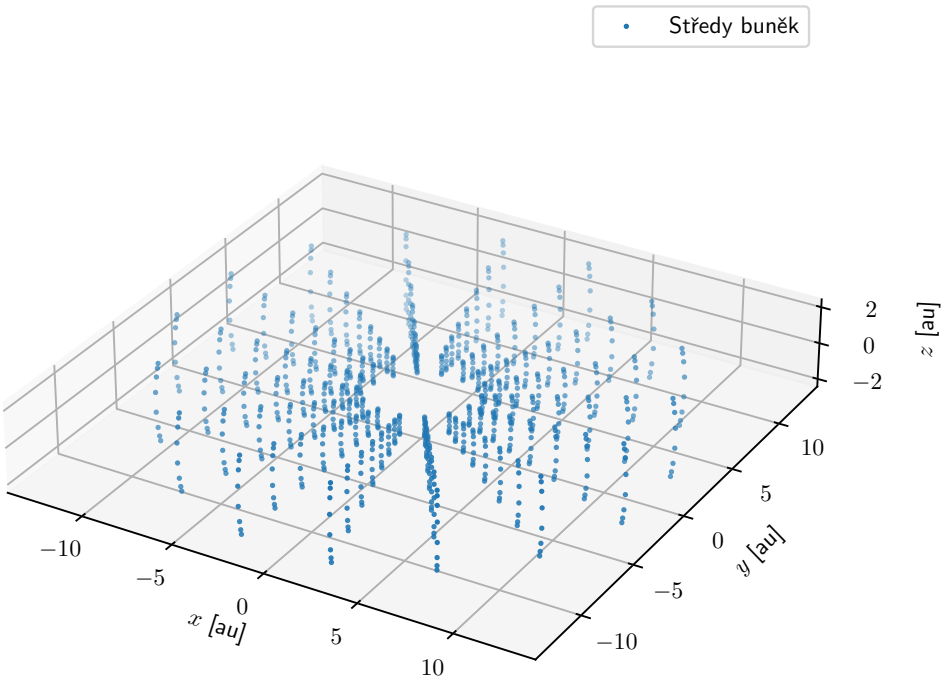
Poslední praktický vizualizační nástroj je procedura `plot_grid` , která vykreslí ve dvou- či trojrozměrném grafu středy buněk trojrozměrné sítě.



Obrázek 4.10: Příklad vykreslení teploty plynu pomocí procedury `plot_fargo_gastemper` jednoho ze zkoumaných disků. Relaxací bylo dosaženo rozlišení 512 radiálních na 512 azimutálních buněk.



Obrázek 4.11: Příklad vykreslení teploty plynu s odečteným radiálním gradientem pomocí procedury `plot_fargo_gastemper` jednoho ze zkoumaných disků. Relaxací bylo dosaženo rozlišení 512 radiálních na 512 azimutálních buněk.



Obrázek 4.12: Středy buněk trojrozměrné mříže po zhrubnutí disku na 16 $\hat{\varphi}$, 8 $\hat{r}$ a $\hat{\vartheta}$ sektorů. V radiálním směru je navíc mříž logaritmická, podobně jako ve směru $\hat{\vartheta}$ — v opticky tlusté rovníkové rovině je řidší, než v atmosféře. Takováto mříž je samozřejmě nevhodná pro výpočty; typicky používaná síť má řádově více buněk v každém směru. Taková síť je ale tak hustá, že by na grafu nebyla téměř vidět její struktura. Další výrazný problém této diskretizace je v její nesouměřitelnosti. Je očividné, že ve směru $\hat{r}$ a $\hat{\varphi}$ je síť výrazně řidší, než ve směru $\hat{\vartheta}$

5. Syntetické a simulované obrazy

Programy Fargo a Faradit byly vytvořeny trojrozměrné modely protoplanetárních disků. V této kapitole se budeme zabývat následujícími kroky: výpočtem přenosu záření, tvorbou ideálních obrazů těchto disků pomocí Radmc a simulací interferometrického pozorování nástrojem ALMA OST dostupným z `https://almaost.jb.man.ac.uk/`.

5.1 Základní trojrozměrný disk

V této sekci budeme, nebude-li řečeno jinak, pracovat s diskem založeném na dvourozměrném disku z práce Chrenko a kol. (2017), označeném jako „Case III“. Jsou v něm vnořeny 4 planety o hmotnosti $M = 3M_\oplus$. Při hydrodynamickém vývoji tohoto disku byla kromě hydrodynamiky plynu započtena i hydrodynamika balvanů, které mohly na planety akretovat a tím je ohřívat.

Původní dvourozměrný disk o 1024 radiálních sektorech jsme rozšířili směrem k hvězdě 256 a směrem od hvězdy (až k hranici naší simulační domény $R_{\max} = 30\,\mathrm{au}$) 512 dalšími radiálními sektory. Poté jsme disk v radiálním směru logaritmicky relaxovali, tj. původní pravidelnou mříž jsme převedli na logaritmickou, aby lépe reflektovala gradienty v modelu vystupující. Povrchovou hustotu a teplotu jsme extrapolovali metodou `inner_outer_finish` , která používá funkce (4.14), (4.15), (4.16) a (4.17). Vertikální hranice disku jsme získali metodou `set_theta_boundary` s požadovanou optickou hloubkou 0,001 s parametrem `extra_margin` $= 5°$. Výsledné hranice domény ve směru $\hat{\vartheta}$ jsou 72,95° a 90°, přičemž pro $\vartheta > 90°$ jsme využili předpoklad symetrie disku kolem rovníkové roviny. Rozložení buněk ve směru $\hat{\vartheta}$ jsme rovněž volili logaritmicky, aby byla dobře rozlišena atmosféra disku, zatímco opticky tlustý střed mohl být hruběji diskretizován. Pro rozumnou výpočetní náročnost vzhledem k dostupným přístrojům jsme omezili počet buněk na výsledných $512 \times 32 \times 386 = 6324224$ buněk ve směrech $\hat{r}$, $\hat{\vartheta}$ a $\hat{\varphi}$. Nejkratší hrana trojrozměrné buňky v tomto disku je $1{,}9{\cdot}10^{-5}\,\mathrm{au}$, nejdelší $6{,}8 \cdot 10^{-1}\,\mathrm{au}$.

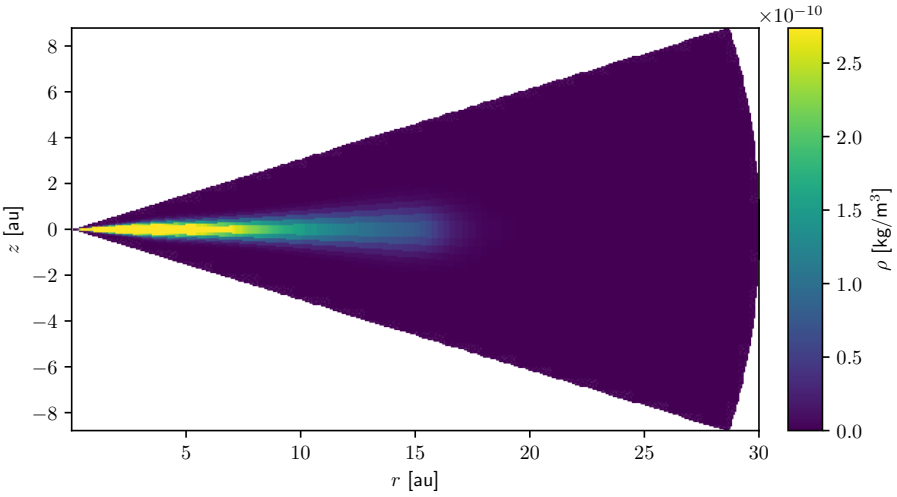
Hustotu plynu ve vertikálním směru jsme získali dle rovnice (4.19). Každá buňka byla zaplněna hustotou získanou jako průměr z pěti bodů na hranách buňky dle obrázku 4.8. Pro převod mezi hustotou plynu a prachu byl dle Birnstiel a kol. (2012) použit faktor $f_{\mathrm{G2D}} = 0{,}01$, výsledný prachový disk disk má hmotnost $2 \cdot 10^{-4}\,M_\odot$ a optická tloušťka jeho rovníkovou rovinou pro $\lambda = 870\,\mu\mathrm{m}$ je řádově 10^{13}. Kvůli velké optické tloušťce byla pro simulaci přenosu záření vždy volena metoda MRV dle Robitaille (2010).

Prach je tvořen silikátovými zrny z amorfního olivínu s 50 % Mg a 50 % Fe (viz Jaeger a kol. 1994, Dorschner a kol. 1995).

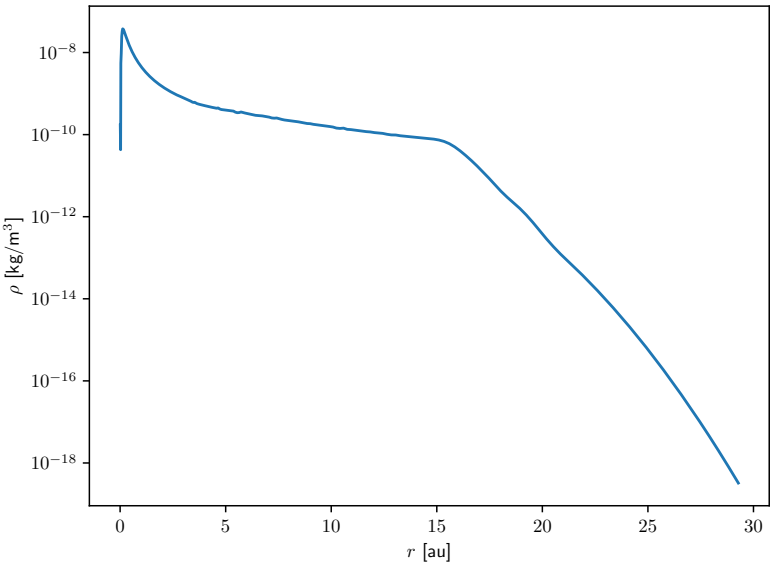
Centrální hvězda má parametry stejné, jako hvězda v práci Chrenko a kol. (2017), z které disk vychází: hmotnost $M_* = M_\odot$, poloměr $R_* = 1{,}5\,R_\odot$ a efektivní teplotu $T_{\mathrm{eff}} = 4370\,\mathrm{K}$. Hvězda je brána jako koule (nikoli jako bodový zdroj) a předpokládáme, že její spektrum je totožné se spektrem absolutně černého tělesa.

5.2 Zkoumané disky

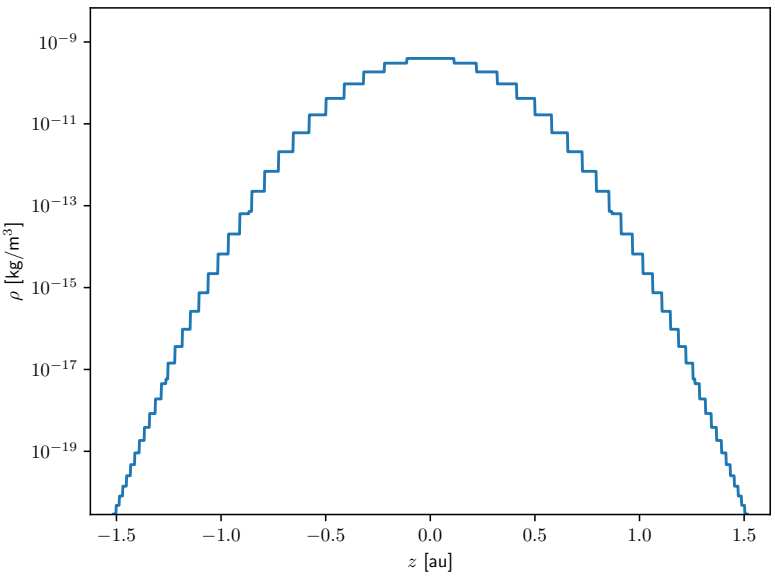
Na základě jednoduchého trojrozměrného disku zmíněného výše jsme sestavili 5 postupně složitějších modelů. U většiny jsme nejdříve provedli výpočet teploty prachu procedurou Radmc `mctherm`. Následně jsme pomocí Radmc měřili spektrum procedurou `sed` a tvořili ideální obrazy ze vzdálenosti 130 pc pomocí procedury `image`. Obrazy byly vždy v kontinuu na vlnových délkách 0,1 µm, 1 µm, 10 µm, 100 µm, 500 µm a 870 µm.



Obrázek 5.1: Objemová hustota prachu ve vertikálním řezu rozšířeným diskem Case III z práce Chrenko a kol. (2017). Disk byl rozšířen funkcí (4.19) a řez byl brán na $\varphi = 0°$. Na tomto grafu je dobře patrné, že absolutní většina hmoty se koncentruje v těsné blízkosti hvězdy. Saturovaná levá část grafu ukrývá maximum hustoty zhruba $3{,}8 \cdot 10^{-8}$ kg/m^3 na jedné desetině astronomické jednotky. Hranice modelu ve směru $\hat{\vartheta}$ byly voleny pomocí `set_theta_boundary` s parametrem `wanted_depth` = 0,001 a `extra_margin` = 5°.



Obrázek 5.2: Objemová hustota prachu v trojrozměrném disku po extrapolaci ke hvězdě a od hvězdy. V oblasti mezi $r = 2{,}8\,\mathrm{au}$ a $r = 14\,\mathrm{au}$ pozorujeme průběh hustoty prachu kopírující hustotu plynu z práce Chrenko a kol. (2017). Na vyšších r je na hustotu hladce navázaný kvadraticky exponenciálně klesající průběh dle funkce (4.16). Směrem ke hvězdě hustota prachu dále roste a vytváří vnitřní okraj disku. Maxima $3{,}78 \cdot 10^{-8}\,\mathrm{kg/m^3}$ dosahuje hustota prachu na $r = 0{,}12\,\mathrm{au}$ před rychlým poklesem směrem k hvězdě.



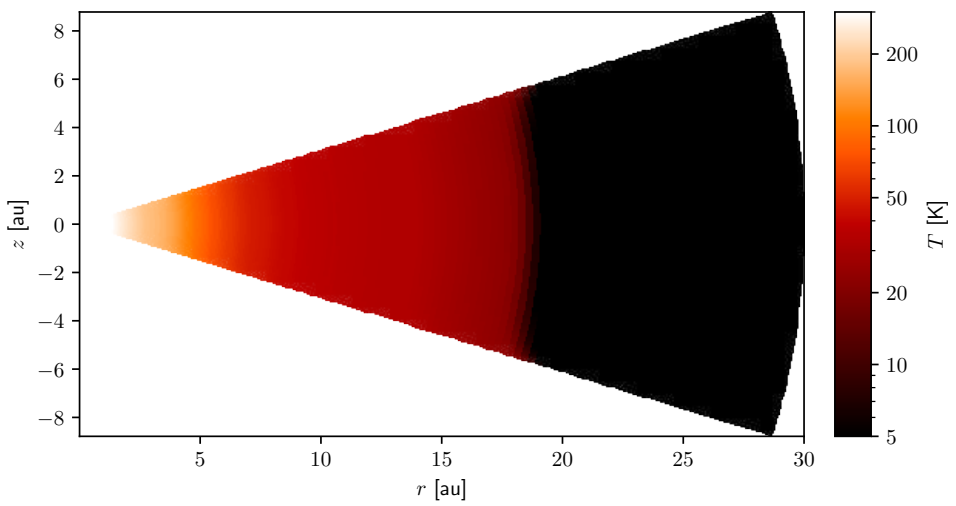
Obrázek 5.3: Detail objemové hustoty prachu na $r = 5\,\mathrm{au}$ v logaritmickém měřítku. Je zde patrný kvadraticky exponenciální průběh hustoty a diskretizace modelu. Výrazné je zde logaritmické rozložení buněk v $\hat{\vartheta}$ směru; buňky se nahušťují v oblasti atmosféry disku.

5.2.1 Vertikálně izotermální disk

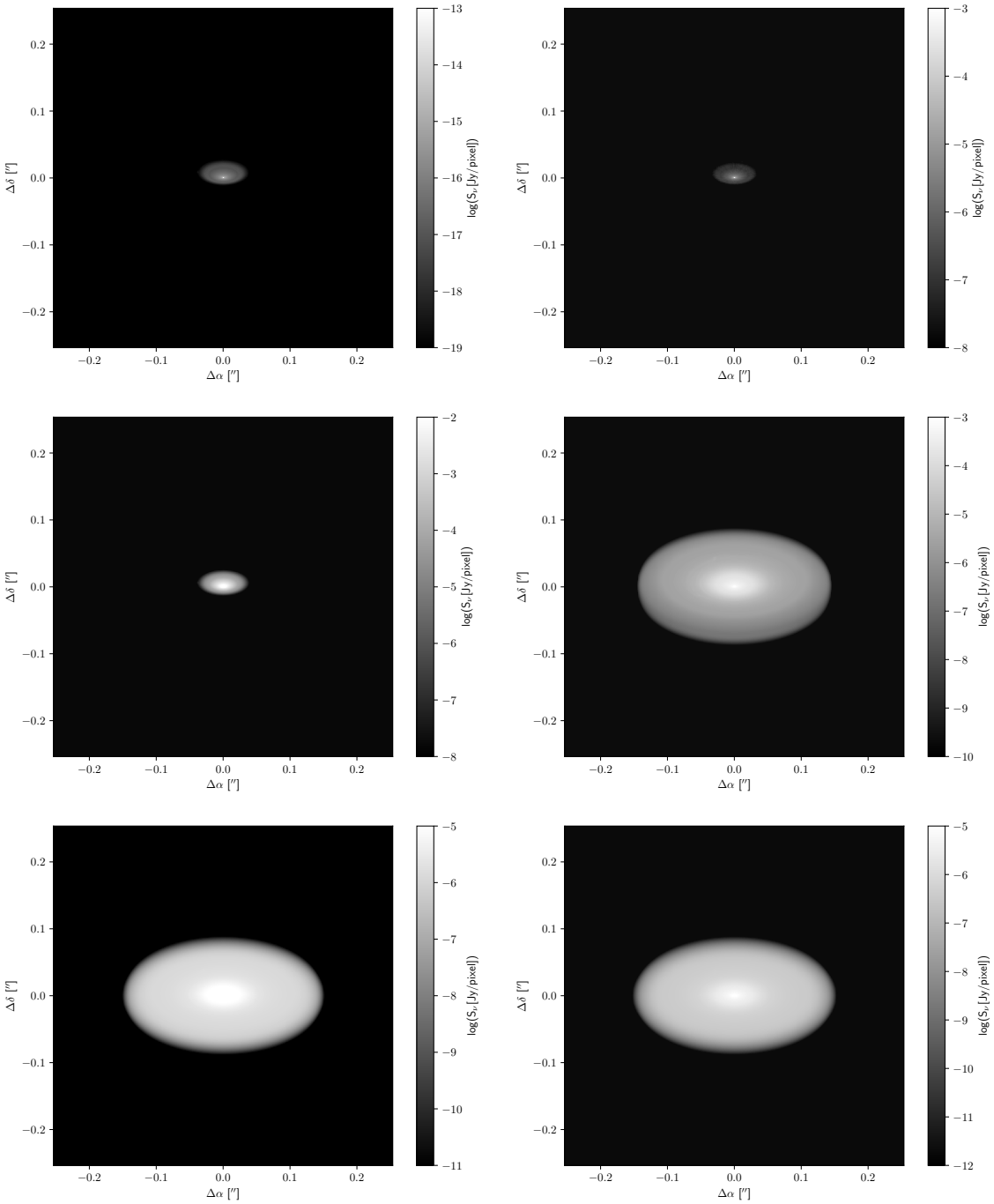
V prvním přiblížení jsme zkoumali vertikálně izotermální disk. Předpokládáme, že teplota prachu je rovna rovníkové teplotě z dvourozměrné hydrodynamické simulace, tudíž jsme nepočítali teplotu procedurou `mcterm`. Vertikální struktura disku je v rovnováze, přesně jak předpokládá dvourozměrný hydrodynamický model. Teplota prachu je vynesena na obrázku 5.4.

Na obrázku 5.5 jsou vyneseny pořízené ideální obrazy v kontinuu. Vytvořili jsme je pro disk natočený pod úhlem 60°. Obrazy jsou v logaritmické škále jednotek Jansky; platí $1\,\mathrm{Jy} := 10^{-26}\,\mathrm{W}\cdot\mathrm{m}^{-2}\cdot\mathrm{Hz}^{-1}$. Na vlnových délkách $\lambda \leq 10\,\mu\mathrm{m}$ je obraz dominovaný zářením centrální hvězdy; je vidět pouze vnitřní okraj disku. Obraz na $\lambda = 100\,\mu\mathrm{m}$ odhaluje teplou středovou rovinu, která, jak bychom z vertikální izotermie předpokládali, září stejně, jako atmosféra. Na této vlnové délce jsou nejlépe patrné struktury vytvořené planetami. Blíže se tomuto obrazu věnujeme v sekci 5.2.6. Na delších vlnových délkách se disk jeví více homogenní a struktury úplně mizí.

Prachová zrna mají v této simulaci všechna velikost 0,1 µm, disk je proto opticky tlustý. Optická tloušťka pro $\lambda > 100\,\mu\mathrm{m}$ τ podél přímky kolmé na disk je řádově 10^2 až 10^3 pro vnitřní části disku ($r \leq 10\,\mathrm{au}$) až po $\tau \approx 1$ na vnějším okraji disku. Na krátkých vlnových délkách $\lambda \leq 100\,\mu\mathrm{m}$ je τ ještě o 3 až 4 řády vyšší.



Obrázek 5.4: Teplota prachu ve vertikálně izotermálním disku v řezu na $\varphi = 0°$. Předpokládáme teplotu prachu identickou s teplotou plynu plochého disku z hydrodynamické simulace práce Chrenko a kol. (2017), extrapolovanou k hvězdě a od hvězdy funkcemi (4.15) a (4.17).



Obrázek 5.5: Ideální obrazy vertikálně izotermálního disku vzdáleného 130 pc. Zleva doprava, řádek po řádku jsou obrazy v kontinuu na vlnových délkách 0,1 µm, 1 µm, 10 µm, 100 µm, 500 µm a 870 µm. Úhel sklonu disku je 60°. První tři vlnové délky dominuje světlo hvězdy, na zbylých už výrazně září disk samotný. Monochromatická intenzita je zde vynesena v jednotkách Jansky ($1\,\mathrm{Jy} := 10^{-26}\,\mathrm{W}\cdot\mathrm{m}^{-2}\cdot\mathrm{Hz}^{-1}$) a v logaritmické škále.

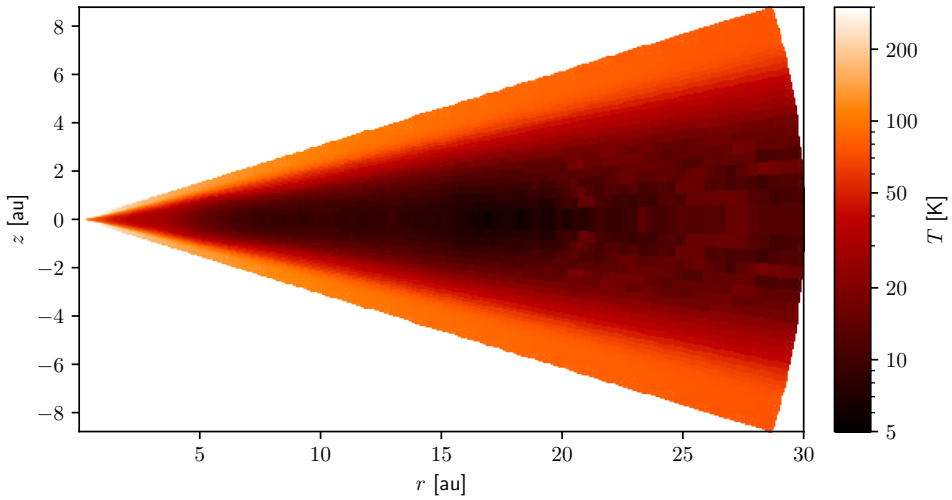
5.2.2 Opticky tlustý disk s radiačním ohřevem

Tento disk je podobně jako ten předchozí zcela tvořen prachem se zrny o velikosti 0,1 μm, optická tloušťka je tedy stejná, jako u minulého modelu. Tento však byl ohřátý procedurou `mcterm`. Na obrázku 5.6 je v nejzazších částech středové roviny vidět numerický šum. Jelikož je disk opticky tlustý, dostane se do této oblasti relativně málo fotonů. Pro lepší prohřátí by byl potřeba řádově vyšší počet simulovaných fotonů.

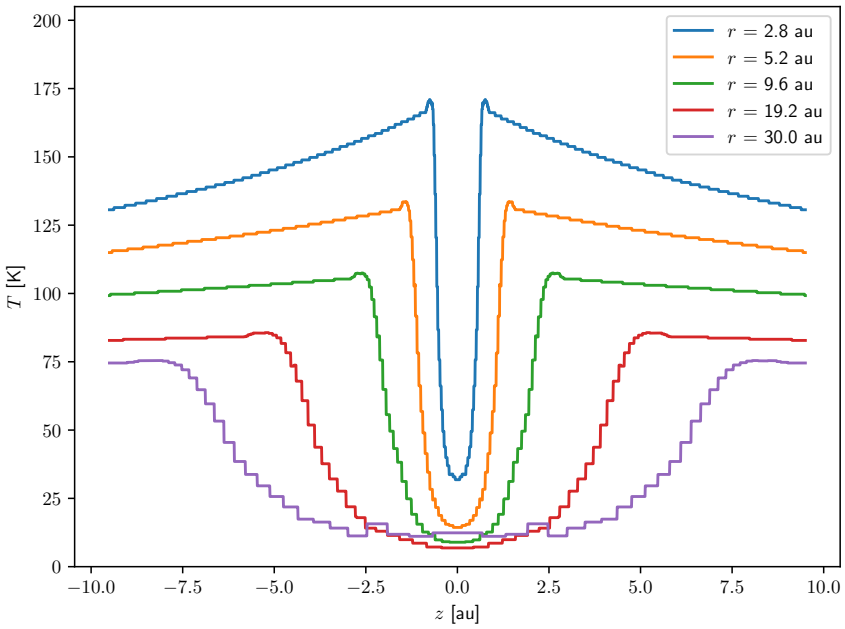
Jelikož je disk zahřívaný centrální hvězdou, stíní jeho vnitřní části rovníkovou oblast. Ta je proto výrazně chladnější než atmosféra, která je přímo vystavená záření hvězdy. Vertikální teplotní profily na vybraných poloměrech r jsou vyneseny na obrázku 5.7. Teplota ve středové rovině se drží v nízkých desítkách Kelvinů, zatímco atmosféra dosahuje obzvláště blízko hvězdě až stovek Kelvinů. Na vnitřním okraji se teplota prachu blíží efektivní teplotě hvězdy.

Na hranici mezi chladnou středovou rovinou a horkou atmosférou se objevují maxima teploty, které převyšují teplotu atmosféry. Jde o oblast s ideální hustotou prachu, který není příliš řídký, aby jím většina energie procházela, ale ani příliš hustý, aby byl proti záření stíněn, jako střed disku.

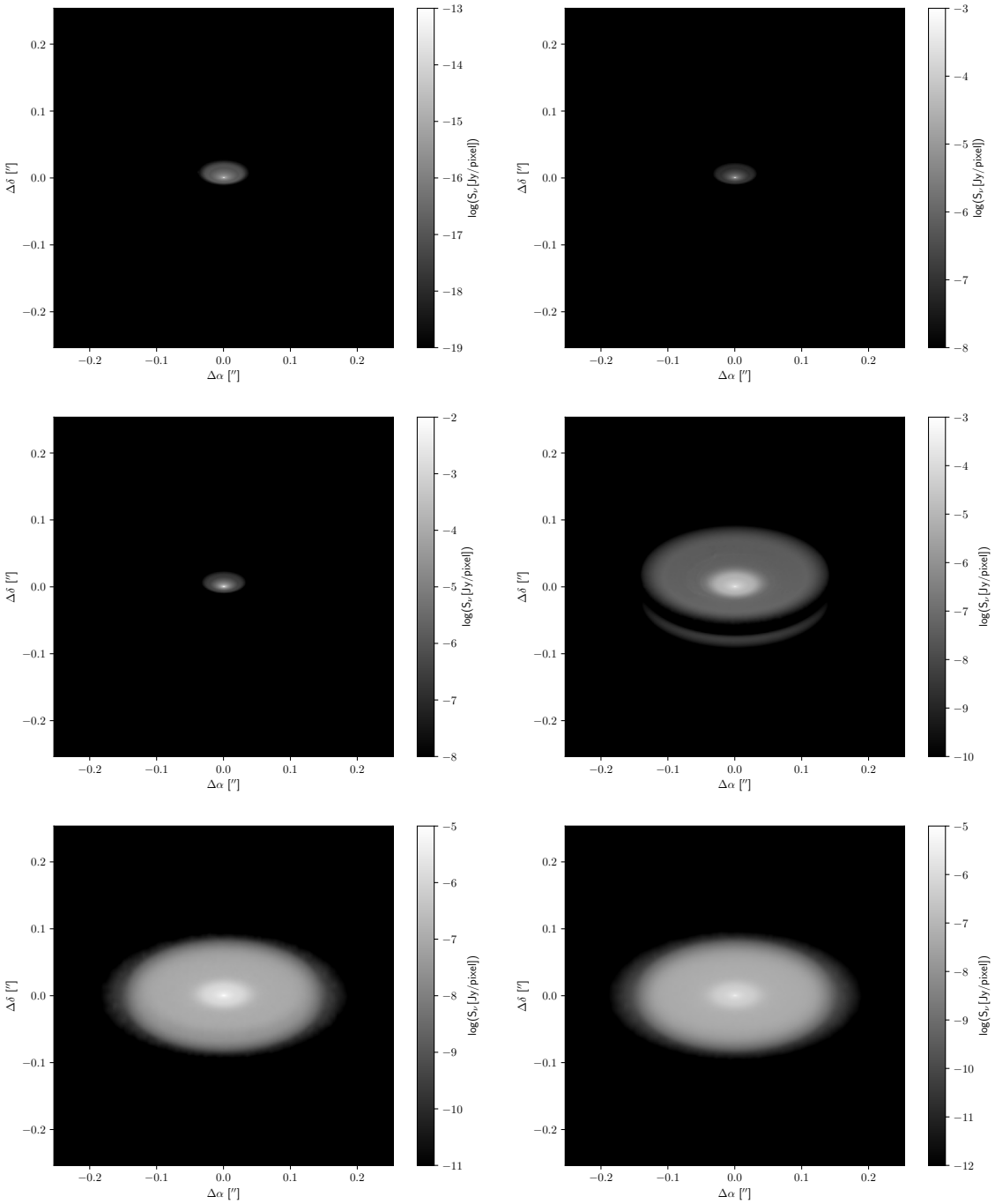
Ideální obrazy na obrázku 5.8 se od předchozího disku výrazně liší. Disk je méně jasný a na $\lambda = 100\,\mu\mathrm{m}$ je výrazná chladná středová rovina. Na takto krátké vlnové délce kvůli své teplotě nezáří. Další dva obrazy ukazují halo řídkého žhnoucího prachu, které ve vertikálně izotermálním modelu úplně chybí. Teplota je zde kvůli radiačnímu ohřevu vyšší, než předchozí disk předpokládá.



Obrázek 5.6: Teplota prachu jakožto výsledek simulace radiačního ohřevu v opticky tlustém disku s velikostí zrn $d = 0{,}1$ µm. Řez je opět na $\varphi = 0°$. Pozorujeme výraznou horkou atmosféru, která je přímo vystavená záření hvězdy a chladnou středovou rovinu. Ve vzdálených částech disku se začíná projevovat šum daný pseudonáhodností metody Monte Carlo. Takto opticky tlustý disk vyžaduje enormní množství fotonů pro ideální prohřátí všech svých částí. Zároveň v této části pozorujeme teplotu *vyšší*, než v rovníkové rovině blíže disku. Neboť je zde hustota prachu malá, je tato část ohřívána z vrchu atmosférou.



Obrázek 5.7: Teplota prachu jakožto výsledek simulace radiačního ohřevu v opticky tlustém disku. Mezi teplotou ve středové rovině a atmosférou je vždy velký rozdíl. Na okraji chladného vnitřku disku teplota krátce převyšuje teplotu atmosféry. Jde o oblast, kde není prach řídký, aby pohlcoval velmi málo záření, ale není ani zastíněný příliš hustým prachem.

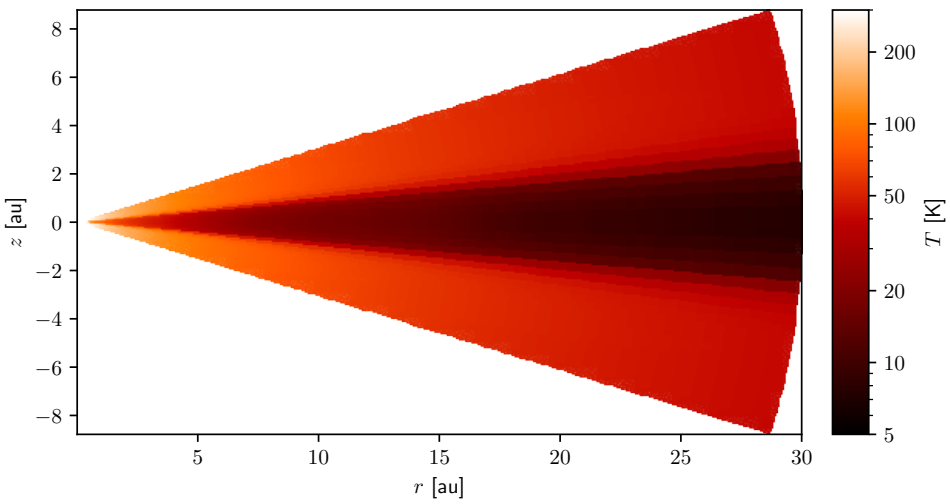


Obrázek 5.8: Ideální obrazy opticky tlustého disku se silikátovým prachem o velikosti zrn $d =$ 0,1 µm. Vlnové délky obrazů v kontinuu jsou stejné, jako u obr. 5.5. Tento disk je výrazně tmavší, než vertikálně izotermální model. Na obrazu ve 100 µm je výrazná chladná, zastíněná část disku v jeho středové rovině. Velká optická hloubka propouští velmi malé množství energie hvězdy hluboko do disku. Na spodních obrázcích, které odpovídají vlnovým délkám 500 µm a 870 µm, pozorujeme halo žhnoucího řídkého prachu. Teplota prachu v této oblasti je z důvodu radiačního ohřevu vyšší, než předpokládá vertikálně izotermální model.

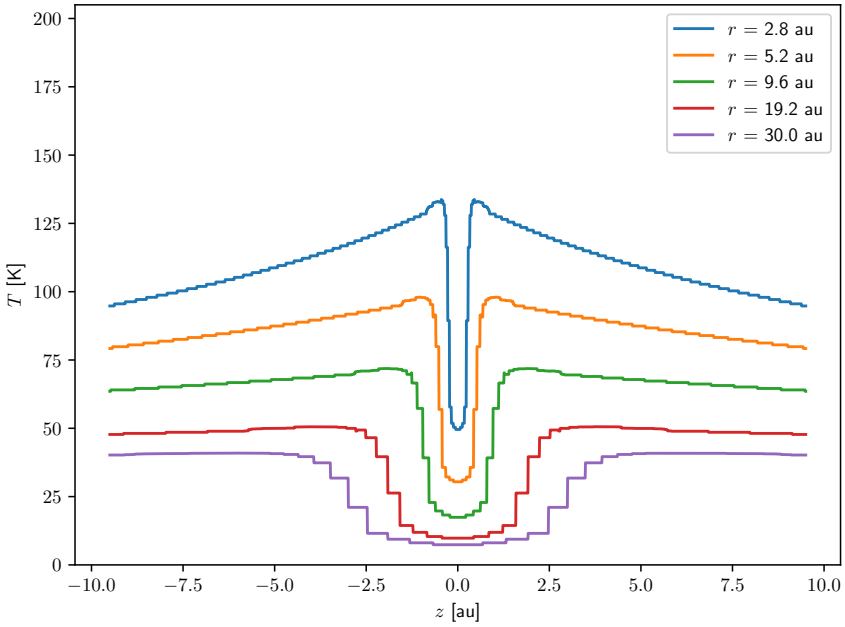
5.2.3 Opticky tenký disk s radiačním ohřevem

Tento disk byl taktéž ohřátý pomocí `mcterm` , avšak zrna mají velikost 10 cm, díky čemuž má prach na většině spektra zhruba o tři řády nižší opacitu než disky předchozí, v některých oblastech a na některých λ je optická tloušťka podél kolmice na disk menší až o 6 řádů. Chladní rovníková oblast, jak ukazuje obrázek 5.9, je proto užší a disk jako celek je kvůli neschopnosti absorbovat světlo chladnější. Zároveň lokální teplotní maxima na hranici atmosféry jsou širší a méně výrazná, jak ukazuje graf na obrázku 5.10.

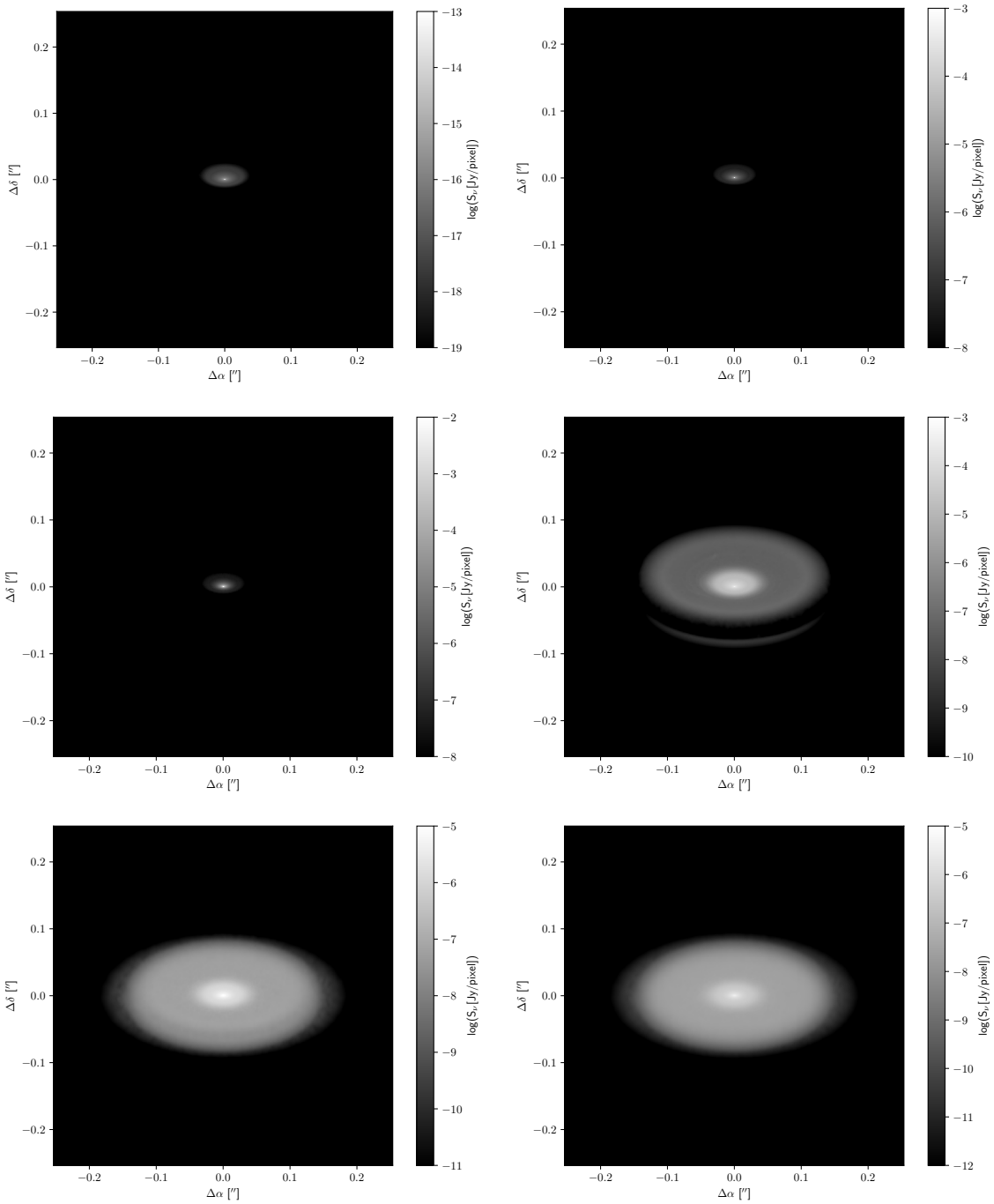
Na obrázku 5.11 jsou opět ideální obrazy. Je na nich výrazné, že disk je na krátkých vlnových délkách méně jasný, neboť kvůli malé opacitě méně rozptyluje světlo centrální hvězdy, než předchozí modely. Na $\lambda = 500\,\mu\text{m}$ jde lépe rozeznat skrz halo chladnou středovou rovinu, neboť je prach průhlednější, než v opticky tlustém disku.



Obrázek 5.9: Teplota prachu jakožto výsledek simulace radiačního ohřevu v opticky tlustém disku s velikostí zrn $d = 10\,\text{cm}$. Chladná středová rovina je výrazně užší, než u opticky tenkého modelu, avšak disk je celkově chladnější, neboť kvůli své nižší opacitě nemůže absorbovat tolik zářivé energie, jako opticky tlustý disk.



Obrázek 5.10: Teplota prachu jakožto výsledek simulace radiačního ohřevu v opticky tenkém disku. V porovnání s opticky tlustým diskem je zde vidět výrazně užší chladná střední oblast a obecně nižší teplota na každém r. Vrcholky po okrajích chladné oblasti jsou širší a méně výrazné. S nižší opacitou tohoto disku je oblast ideální absorpce záření širší, než u opticky tlustého modelu.

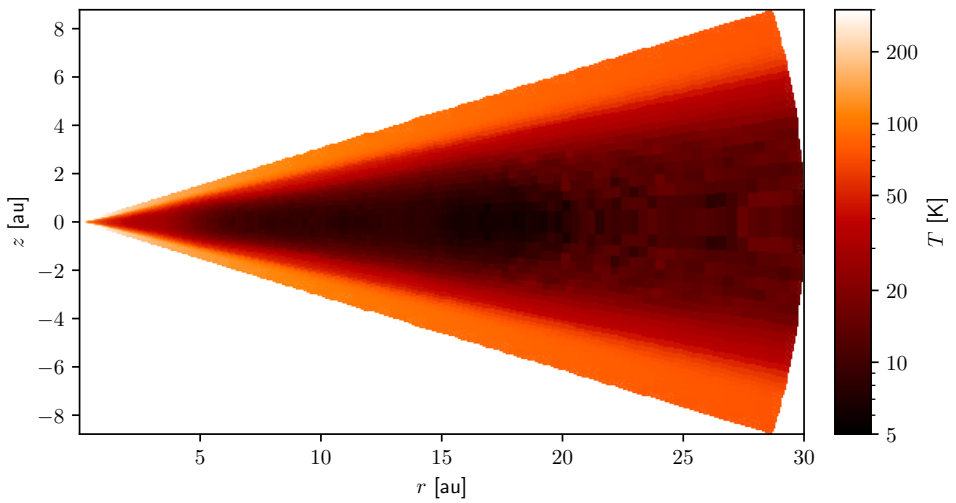


Obrázek 5.11: Ideální obrazy opticky tenkého disku. Disk je na krátkých vlnových délkách výrazně méně jasný, neboť nízká opacita velkých zrn způsobuje, že méně světla centrální hvězdy je od prachu odražena k pozorovateli. Vlevo dole na obrazu na vlnové délce 500 μm je oproti opticky tlustému disku zřetelná průhlednost prachu; skrz zářící halo je vidět náznak studenější středové roviny.

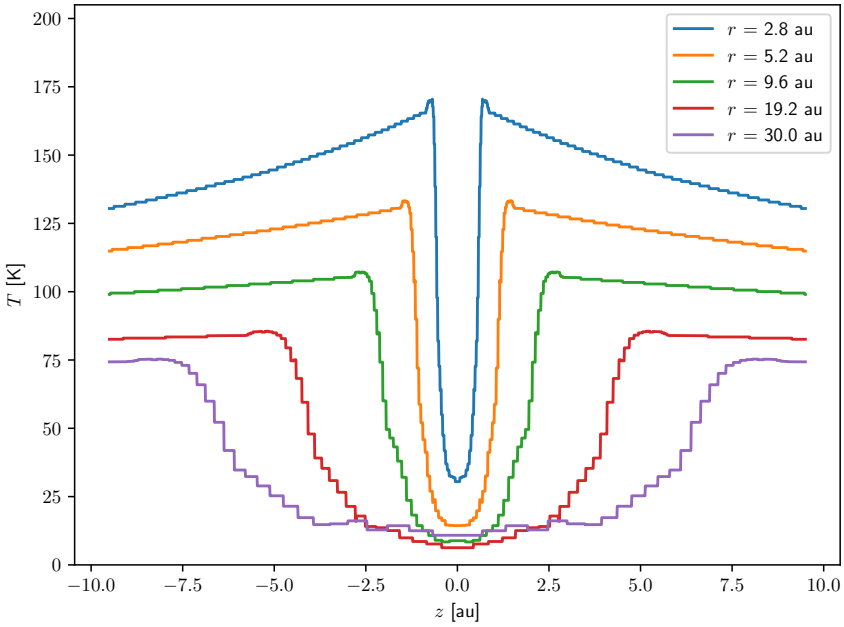
5.2.4 Vícezrnný disk s radiačním ohřevem

V tomto modelu jsme zahrnuli rozdělení velikostí zrn. Brali jsme v potaz zrna o velikostech 0,1 µm až 10 cm. Předpokládali počet zrn úměrný jejich velikosti $n \propto d^{-3,5}$ (dle rovnovážného kolizního modelu viz Dohnanyi 1969). Jelikož hmotnost zrnka prachu $m \propto d^3$, pro celkovou hmotnost prachu o velikosti d v každé buňce platí $\rho(d) \propto d^{-\frac{1}{2}}$, jak ukazuje graf na obrázku 5.16. Celkovou hmotnost prachu všech velikostí jsme normovali tak, aby v součtu stále platilo $f_{\mathrm{G2D}} = 0{,}01$.

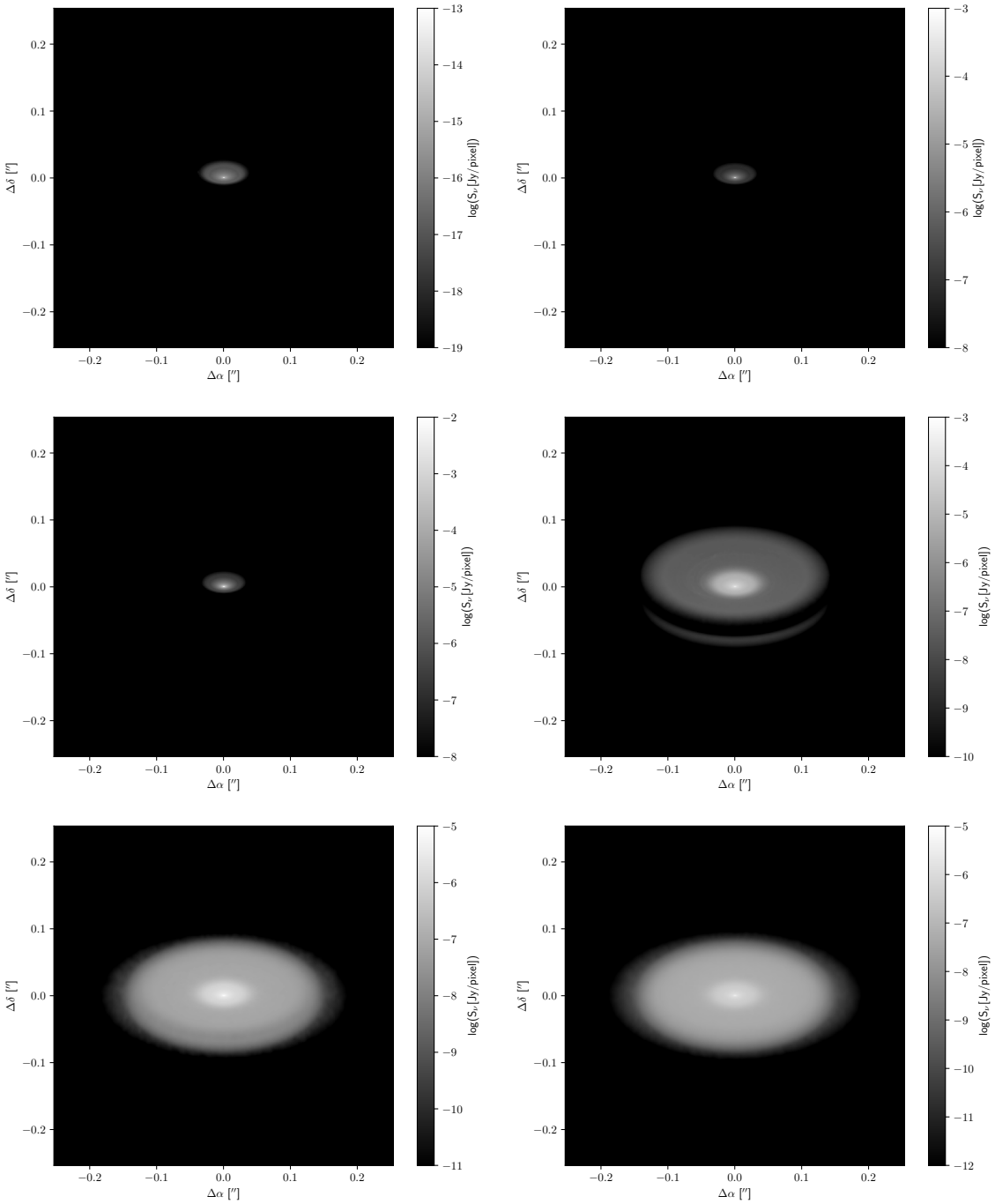
Na obrázcích 5.12, 5.13 a 5.14 je vidět, že se tento disk velmi podobá opticky tlustému modelu; všechny jevy, které jsme u něj pozorovali (numerický šum, teplotní maxima kolem okrajů atmosféry...) se tu také projevují. Disk je sice téměř všude opticky tlustý, kolmá optická tloušťka je ale výrazně nižší (téměř o dva řády), než u základního, opticky tlustého a vertikálně izotermálního modelu. Například na $r = 10\,\mathrm{au}$ pro $\lambda = 500\,\mu\mathrm{m}$ je $\tau \approx 7$, pro $\lambda = 870\,\mu\mathrm{m}$ už dokonce $\tau \approx 1{,}5$.



Obrázek 5.12: Teplota prachu jakožto výsledek simulace radiačního ohřevu ve vícezrnném disku. Vyobrazený je řez na $\varphi = 0°$. Prohřátí disku je velmi podobné opticky tlustému modelu včetně atmosférického ohřívání ve vnějších částech.

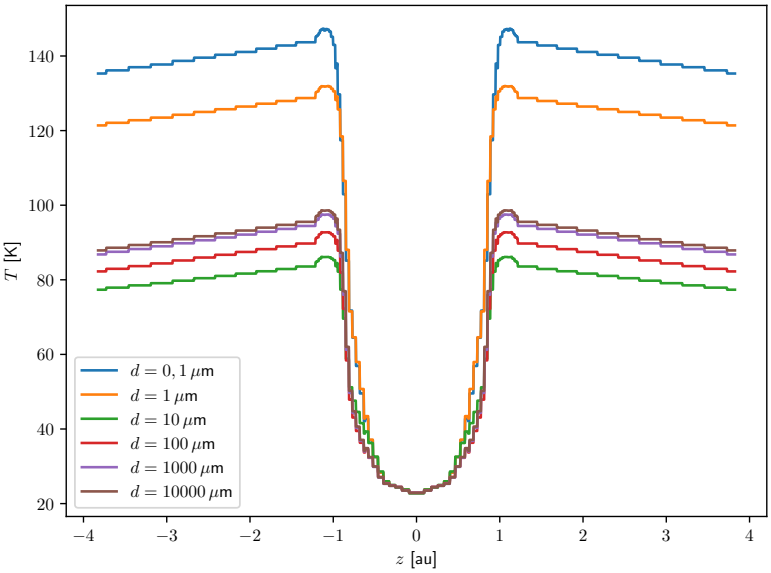


Obrázek 5.13: Teplota prachu jakožto výsledek simulace radiačního ohřevu ve vícezrnném disku. Ani ve vertikálním profilu teplot se tento disk téměř neliší od modelu s pouze jedním typem prachových částic.

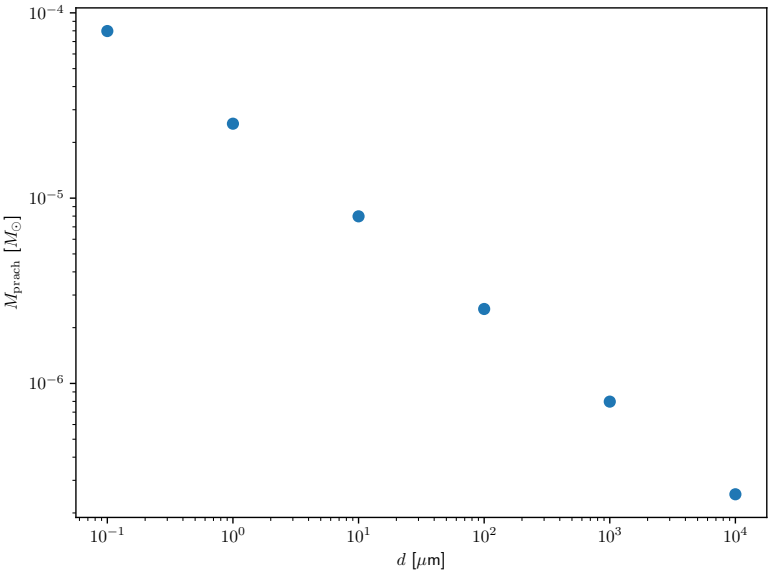


Obrázek 5.14: Ideální obrazy vícezrnného disku. Příspěvek od malých zrn se výrazně projevuje v jeho opacitě, proto jsou obrazy podobné obrazům opticky tlustých disků, avšak například na vlnové délce 100 μm je tento disk nepatrně tmavší.

Pokud počítáme v Radmc s vícero různými druhy prachových zrn, je teplota každého druhu počítána zvlášť. Nejmenší zrna s nejvyšší opacitou absorbují nejvíce energie, přezařujíc tak ostatní prach. Naopak ve středové rovině mají všechny druhy prachu stejnou teplotu, protože záření ve středové rovině má hlavně dlouhé vlnové délky, na kterých jsou optické vlastnosti zrn všec velikostí podobné.



Obrázek 5.15: Teploty prachových zrn jednotlivých velikostí na $r = 5\,\mathrm{au}$. Pozorujeme monotónní závislost teploty na rozměru. Jelikož je na rozměru monotónně závislá i opacita na většině vlnových délkách, je jasné, že velká zrna nebudou mít vysokou teplotu, jelikož nemohou absorbovat mnoho energie. Ve středové rovině je teplota všech zrn stejná. Pokud se do této oblasti nějaký foton dostane, dá se předpokládat, že kvůli velké optické tloušťce zde stráví hodně času a může tedy rovnoměrně ohřát všechen prach.



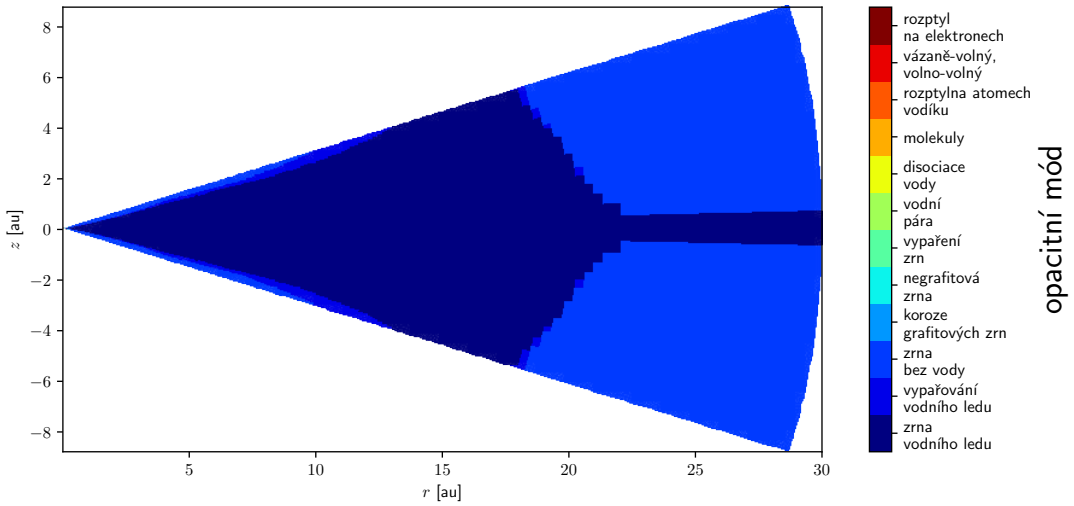
Obrázek 5.16: Celková hmotnost prachu se zrny velikosti d v disku. Dle Dohnanyi (1969) jsme pracovali s počtem zrn o velikosti d: $n(d) \propto d^{-3,5}$ a $\rho_{\mathrm{prach}} \propto nd^3$, tedy $\rho_{\mathrm{prach}} \propto d^{-0,5}$. V logaritmicko–logaritmickém grafu proto pozorujeme přímku. Výsledek jsme normovali na f_{G2D}, součet hmotností všech zrn je tedy identický s původním modelem.

5.2.5 Opticky tlustý disk s viskózním ohřevem a vypařenými zrny

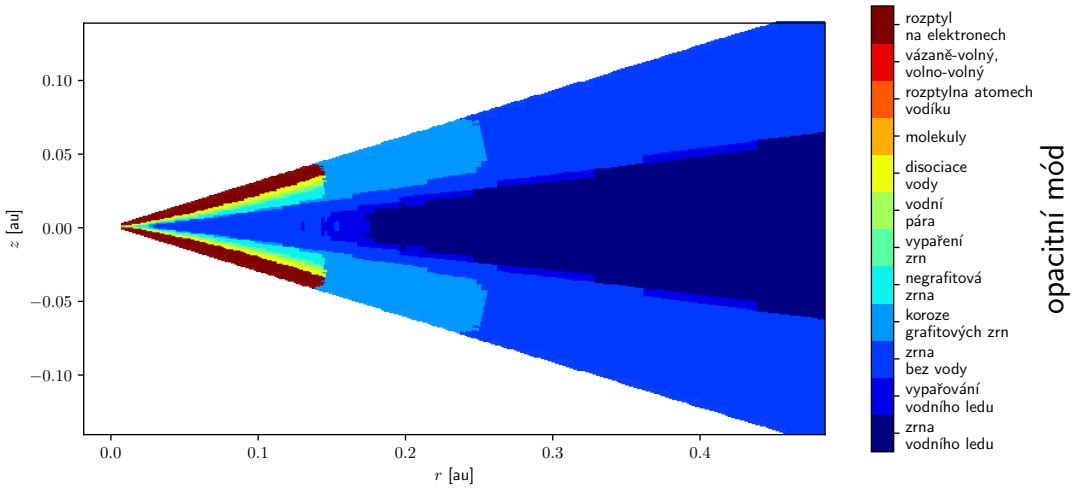
Tento poslední disk je založený na opticky tlustém disku s radiačním ohřevem (viz sekci 5.2.2). Disk jsme jednou nechali prohřát procedurou `mcterm`, poté jsme pomocí Zhu a kol. (2012) vypočetli, v jakém režimu opacity se prach nachází, viz obrázky 5.17 a 5.18. Předpokládali jsme $f_{\mathrm{G2D}} = 0{,}01$ a identickou teplotu prachu a plynu. Všude tam, kde byla teplota a hustota plynu odpovídající vypařování zrn, jsme hustotu prachu zmenšili řádem 10^{10}. Tím jsme hrubě aproximovali vypařování prachu. Navíc jako vstupní soubor do Radmc jsme nyní přidali `heatsource.inp` pro simulaci viskózního ohřevu. Každá buňka se stala zdrojem tepla podle rovnice (3.18).

V tomto disku je mnohem výraznější problém numerického šumu. Jak je vidět na řezu na obrázku 5.19, ani navýšení počtu simulovaných fotonů faktorem 10^2 oproti opticky tlustému modelu nestačilo pro úplné prohřátí disku.

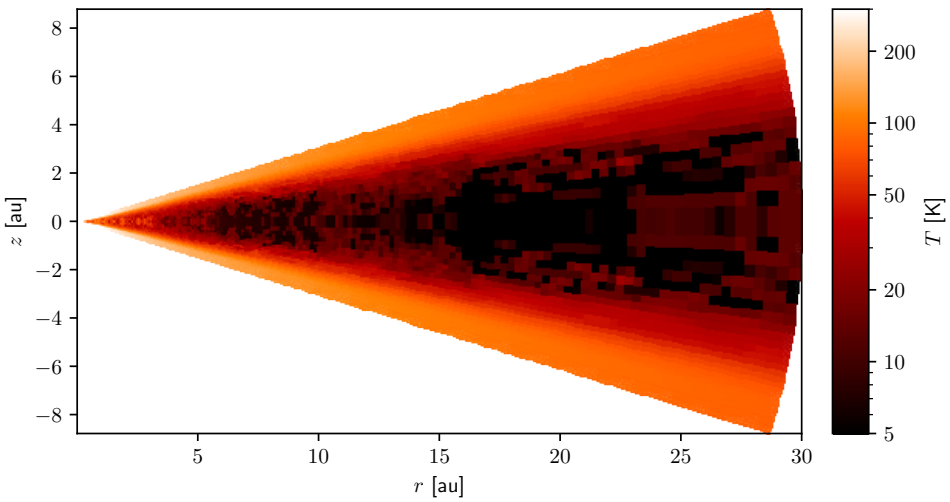
Na vertikálním profilu teploty, který je vynesený na obrázku 5.20, je vidět, že disk je obecně teplejší, než původní model. Navíc uprostřed rovníkové roviny nacházíme lokální maximum teploty, neboť právě zde je viskózní ohřev nejsilnější.



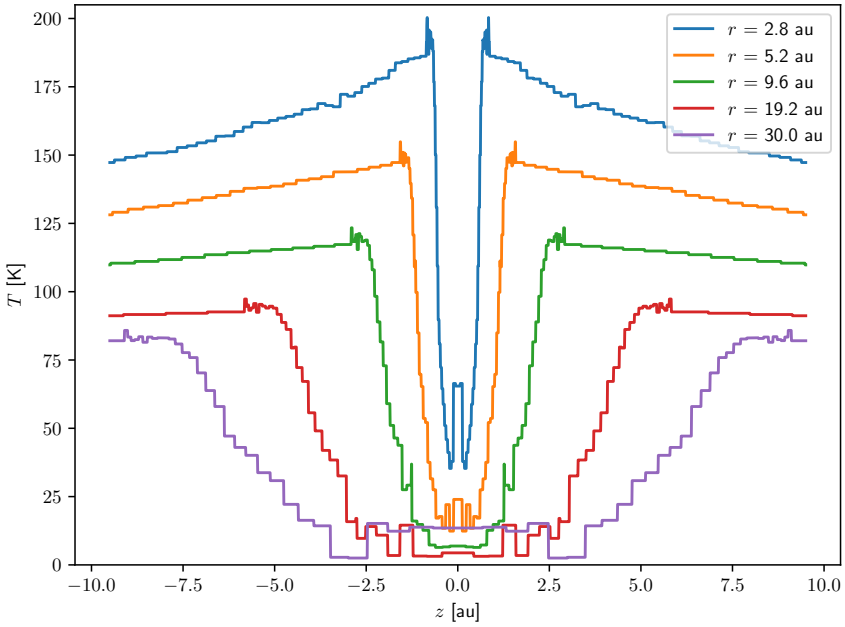
Obrázek 5.17: Opacitní mód, ve kterém dle Zhu a kol. (2012) operuje materiál disku. Jelikož pracujeme se silikátovým diskem, všechny oblasti, ve kterých by disk měl být v režimu vypařování zrn a vyšším, nutně nemohou obsahovat prachová zrna. Hustotu prachu v nich proto zmenšujeme řádem 10^{10}, abychom vypaření prachu simulovali. Absolutní většina disku však existenci silikátových zrn podporuje. V nejzazší části disku se vypařuje vodní led, neboť hustota plynu a prachu zde strmě klesá a s ní i tlak.



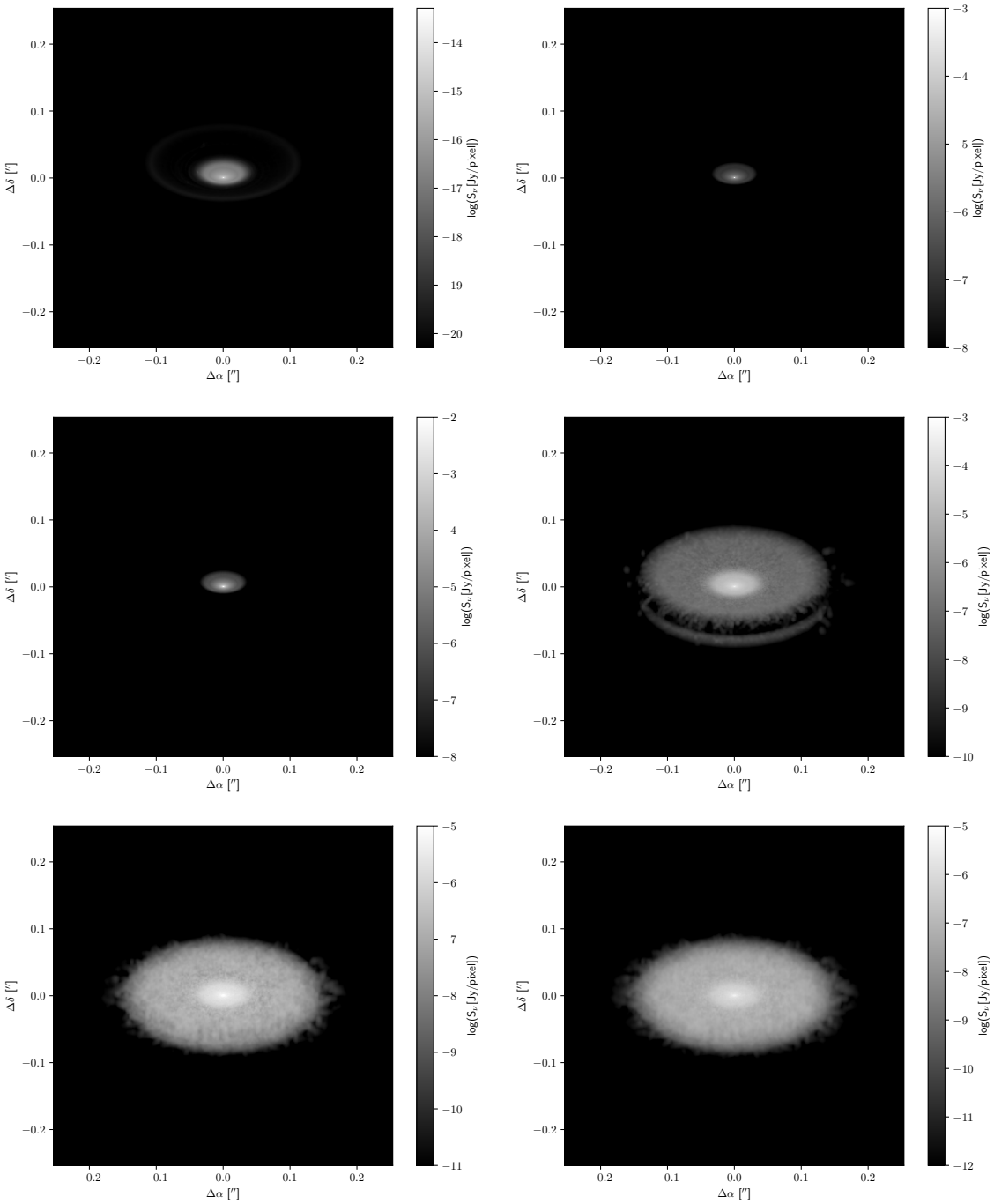
Obrázek 5.18: Detail vnitřního okraje silikátového disku a opacitních režimů, které zde pozorujeme. Velmi blízko centrální hvězdě nastávají vyskoé teploty a materiál může být nejen vypařený, ale i ionizovaný. Vnitřní části disku jsou ovšem efektivně odstíněny a vypařování probíhá i blízko hvězdě převážně v atmosféře.



Obrázek 5.19: Teplota prachu jakožto výsledek simulace radiačního a viskózního ohřevu ve vícezrnném disku. Vyobrazený je řez na $\varphi = 0°$. V teplotě se zde velmi výrazně projevuje pseudonáhodný šum. Pravděpodobně jde o problém optimalizace procedury v Radmc, která pracuje s `heatsource.inp`. Jelikož jde o stejný disk, jako v opticky tlustém modelu, jen s dalším zdrojem ohřevu který je navíc hojně zastoupený i v opticky tlusté středové rovině, očekávali bychom vyšší teplotu, než je v opticky tlustém modelu. V tomto disku se viditelně projevuje poblíž hvězdy kolem $r = 3$ au a blíže. Efektivně prohřát se nám tento model nepodařilo ani s řádově většími počty simulovaných fotonů.



Obrázek 5.20: Teplota prachu jakožto výsledek simulace radiačního a viskózního ohřevu. Na profilu je výrazný pseudonáhodný šum nedostatečného počtu fotonů. Teplota je všude výrazně vyšší a hlavně uprostřed rovníkové roviny, kde u ostatních modelů bývala teplota nejnižší, se nyní nachází teplotní vrchol.



Obrázek 5.21: Ideální obrazy opticky tlustého disku vzdáleného 130 pc se simulací viskózního ohřevu a vypařování zrn. Zleva doprava, řádek po řádku jsou obrazy v kontinuu na vlnových délkách 0,1 µm, 1 µm, 10 µm, 100 µm, 500 µm a 870 µm. Úhel sklonu disku je 60°. Výrazný šum obrazů je způsobený nedostatečným prohřáním disků.

5.2.6 Některé zajímavé detaily zkoumaných modelů

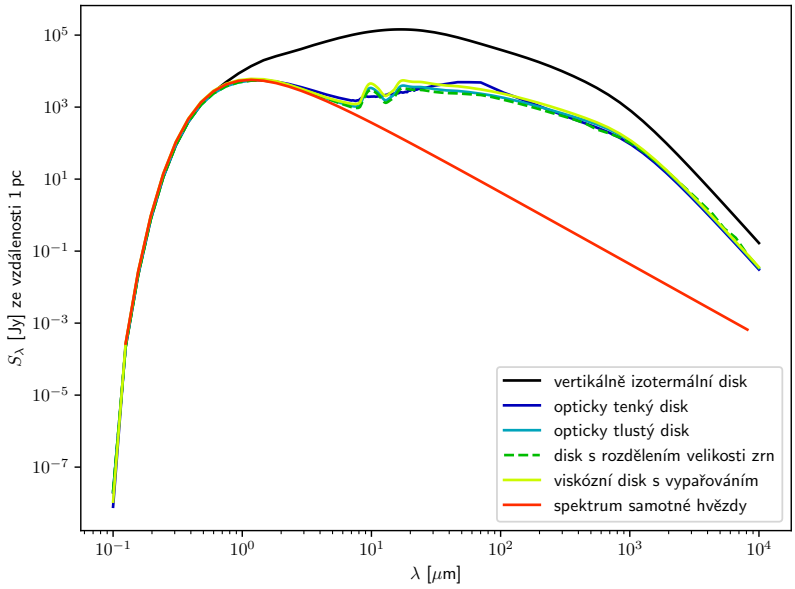
Věnujme nyní pozornost některým detailům zkoumaných modelů.

Pomocí příkazu `radmc3d sed` jsme vygenerovali syntetická spektra pozorovaných disků a porovnali je na obr. 5.22. Na krátkých vlnových délkách ($\lambda \leq 10\,\mu\mathrm{m}$) disky samotné téměř vůbec nezáří a spektra vypadají jako spektra absolutně černého tělesa — centrální hvězdy. Na dlouhých vlnových délkách ($\lambda \gg 1\,\mu\mathrm{m}$) však chybí charakteristický mocninný pokles S_λ, protože ke spektru přispívají různé části disku, které mají různé teploty. Výsledná křivka je tedy součtem mnoha spekter černých těles s různými teplotami.

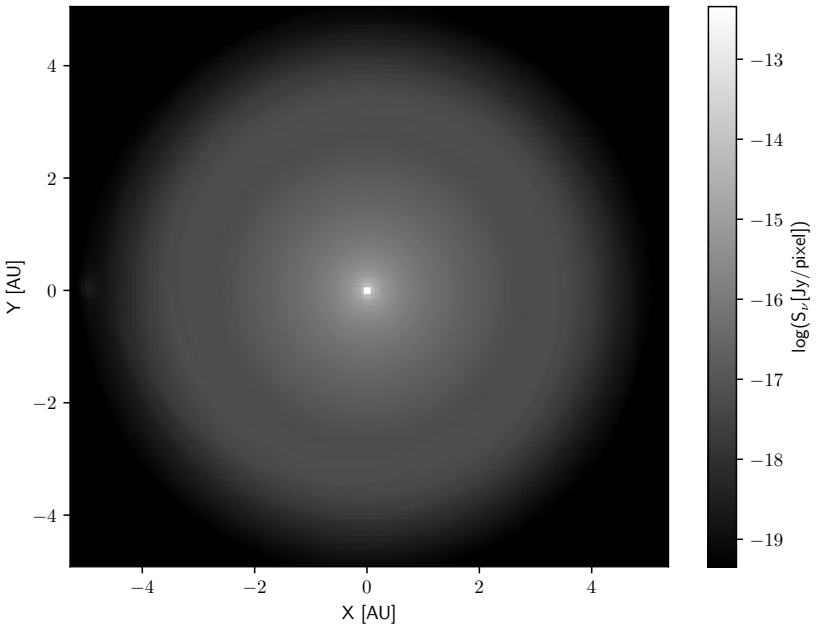
Vzhledem k rozdělení teplot mezi různými zrny u vícezrnného disku, jak ukazuje obrázek 5.15, bychom očekávali výše zmíněné rozšíření spektra u tohoto modelu výraznější. Očividně je ale efekt malý, neboť spektrum vícezrnného disku je tvarem prakticky nerozeznatelné od spektra disku opticky tlustého, jen je disk nepatrně méně jasný.

Na krátkých vlnových délkách ($\lambda \leq 10\,\mu\mathrm{m}$), kde obrazu dominuje rozptýlené světlo centrální hvězdy, se na vnitřním okraji disku objevuje tmavý prstenec. Toto nejspíš není struktura způsobená fyzikálním jevem, spíše jde o artefakt napojení původních dat hydrodynamické simulace na vnitřní extrapolaci hustoty disku.

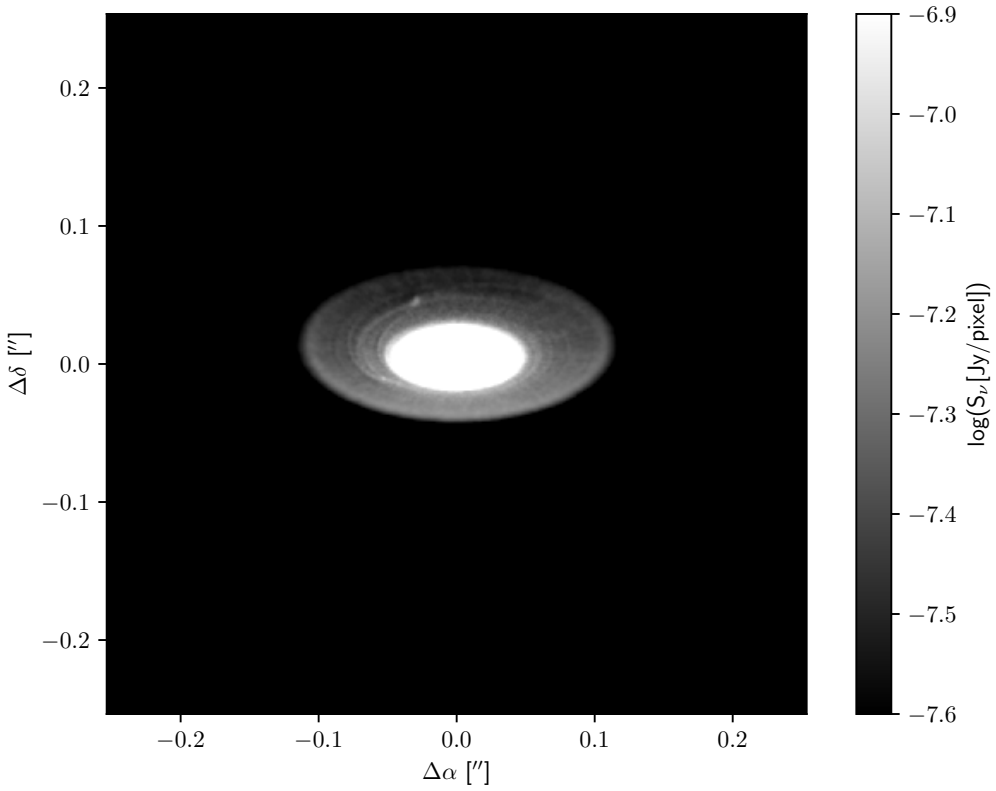
Na některých vlnových délkách (například $\lambda = 100\,\mu\mathrm{m}$) jsou dobře rozeznatelné perturbace disku způsobené planetami. Jejich kontrast oproti okolí nepřesahuje půl řádu Jy a jejich rozměry jsou velmi malé. Jasné oblasti kolem planet nemají rozměr vyšší, než $(6 \cdot 10^{-3})''$, méně výrazná spirální ramena jsou dlouhá zhruba $(5 \cdot 10^{-2})''$.



Obrázek 5.22: Spektra všech modelovaných disků a centrální hvězdy o teplotě $T_{\mathrm{eff}} = 4370\,\mathrm{K}$. U většiny disků pozorujeme kolem $\lambda = 10\,\mu\mathrm{m}$ vrcholky, které odpovídají vibračnímu přechodu v molekule SiO_4 (viz Dorschner a kol. 1977). Tento přechod není vůbec vidět u izotermálního disku. To je zcela očekávané, neboť tato teplota je výsledkem hydrodynamického modelu práce Chrenko a kol. (2017), který počítal s teplotou *plynu*, nikoli silikátových zrn prachu. Spektra disku se od spektra černého tělesa výrazně liší. Pozorujeme maximum odpovídající záření hvězdy na $\lambda \approx 2\,\mu\mathrm{m}$, avšak poté chybí u disků charakteristický mocninný pokles. Tento tvar spektra je výsledkem superpozice spekter černých těles na různých teplotách, které mají různé části disku. Spektrum opticky tlustého disku není vidět, neboť je téměř totožné se spektrem viskózně ohřátého disku s vypařováním zrn.



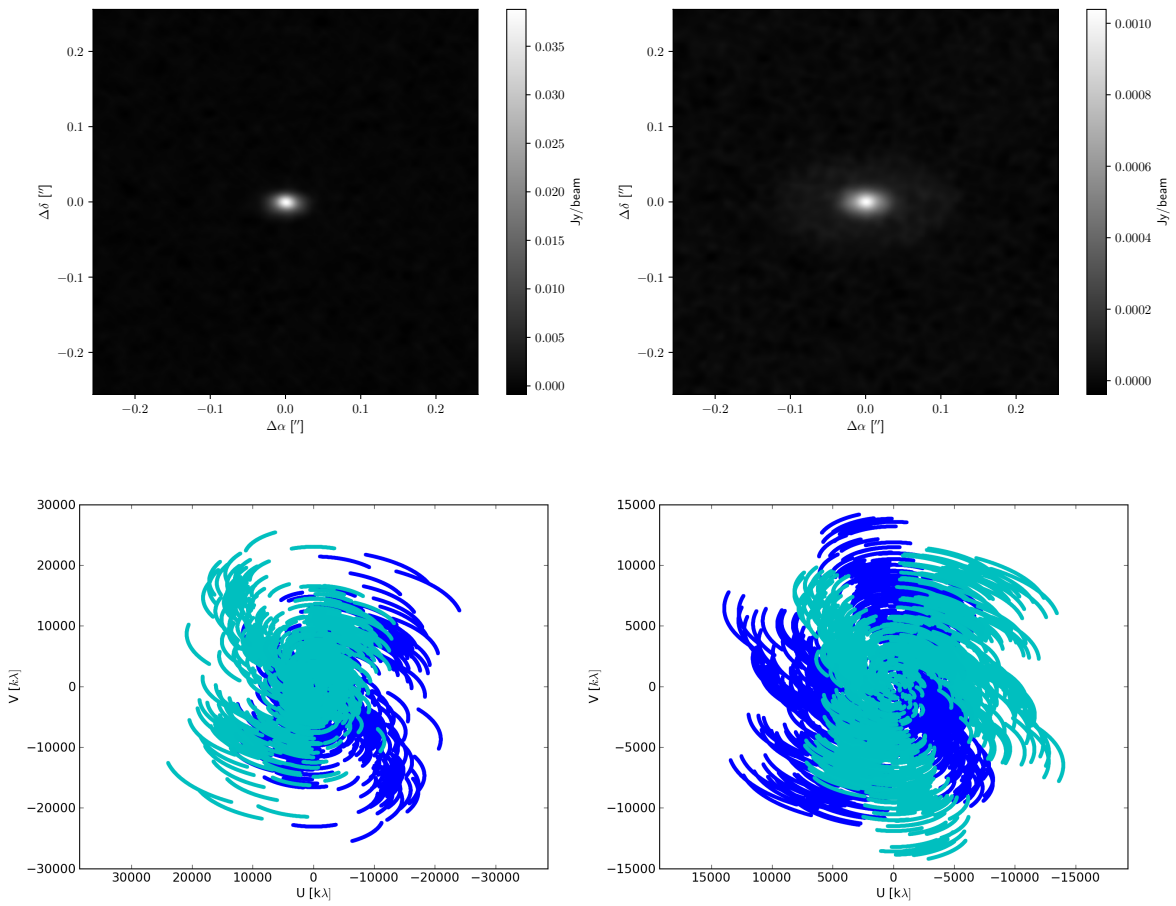
Obrázek 5.23: Na obrazech většiny modelů na krátkých vlnových délkách je pozorovatelný tenký prstenec nižší intenzity na $r = 2{,}8$ au. Řádově jde o poklese menší, než čtvrt řádu Jy. Jak ukazuje tento obraz na $\lambda = 0{,}1\,\mu\mathrm{m}$ opticky tlustého modelu, focený kolmo na rovinu disku, odpovídá prstenec místu, kde se napojuje vnitřní extrapolace hustoty a teploty na hydrodynamický model.



Obrázek 5.24: Saturovaný ideální obraz vícezrnného disku na $\lambda = 100\,\mu$m. Na ideálních obrazech všech modelovaných disků jsou pozorovatelné vnořené planety, resp. spirální ramena způsobená Lindbaldovými rezonancemi. Rozdíl v toku mezi jasnými strukturami a tmavším okolním diskem je zhruba půl řádu Jy. Pozorovatelnost je velmi závislá na vlnové délce. Pro $\lambda = 100\,\mu$m jsou struktury zdaleka nejvýraznější, na obvyklé vlnové délce pro pozorování ALMA $\lambda = 870\,\mu$m jsou prakticky neviditelné.

5.3 Syntetické inreferometrické pozorování

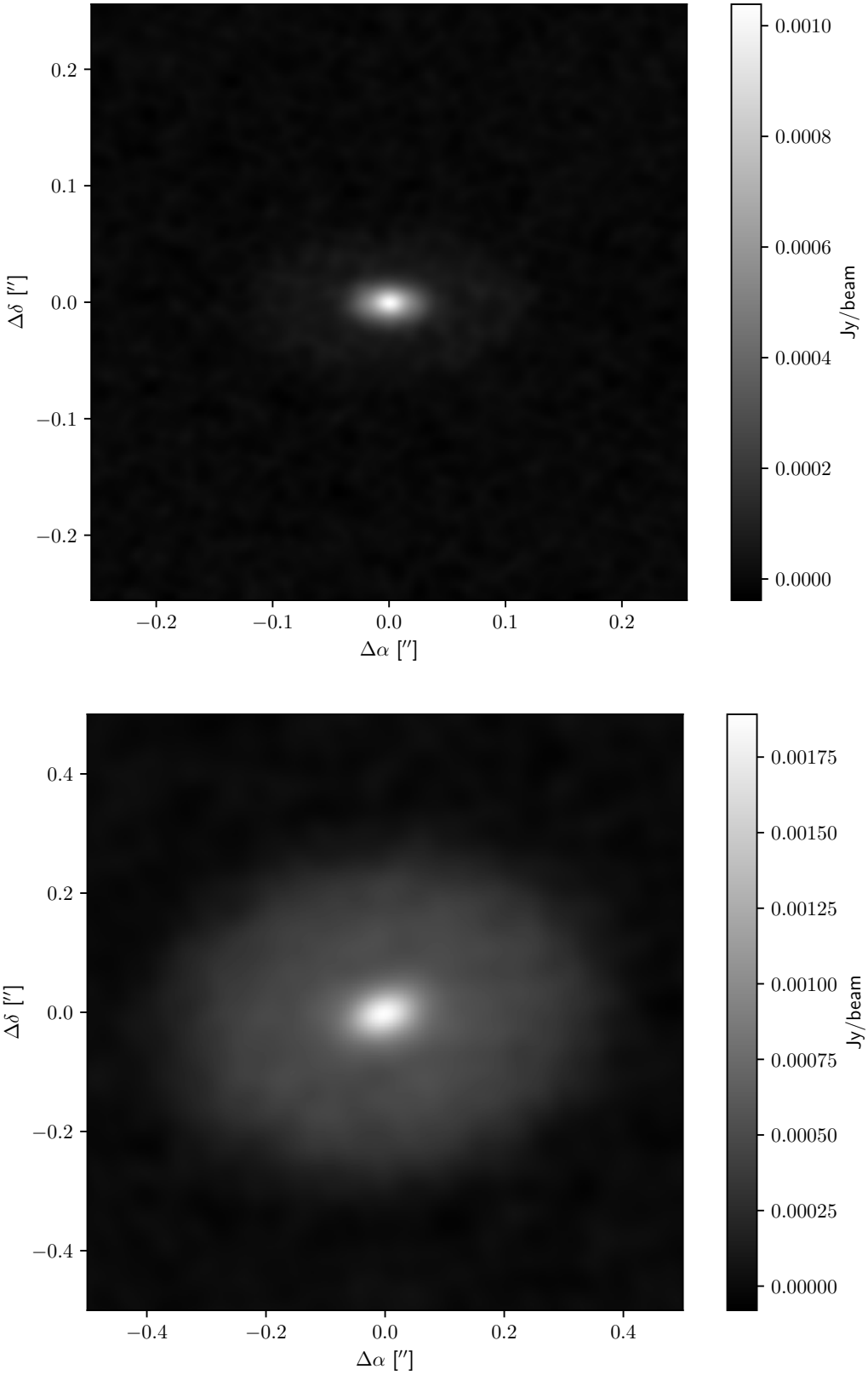
Nakonec jsme se zabývali pozorovatelností planet, respektive struktur které vytváří, interferometrem ALMA. Pomocí ALMA OST jsme simulovali interferometrické pozorování v kontinuu. ALMA podporuje pozorování v pásmech číslovaných 1–10 na vlnových délkách mezi 8500 μm a 300 μm, resp. 35 Ghz až 950 GHz (viz například přehled na `https://www.eso.org/public/teles-instr/alma/receiver-bands/`). Vzhledem ke špatné pozorovatelnosti struktur v ideálních obrazech jsme volili co nejkratší vlnovou délku simulovaného pozorování a co nejdelší možné základny. Nejkratším vlnovým délkám odpovídá pásmo ALMA 9 a 10. Deklinaci simulovaného disku na nebeské sféře jsme volili δ =-24d43m14,1s, což odpovídá systému DoAr 25 ve vzdálenosti 130 pc (viz Andrews a kol. 2009). Paralelně jsme též provedli simulaci pozorování v pásmu 7, ve kterém jsou všechny disky jasné. Při simulaci jsme předpokládali nejlepší možné atmosférické podmínky: šum odpovídající vodní páře v atmosféře (PWV) v prvním oktilu pro Atacamskou poušť — PWV= 0,47 mm. Výsledky jsou na obr. 5.25.



Obrázek 5.25: Simulované interferometrické měření vícezrnného disku pomocí ALMA OST. Vlevo je vyobrazený výsledek pozorování v pásmu 10 na nejkratší vlnové délce, na které ALMA operuje a odpovídající pokrytí roviny (u,v), vpravo totéž v pásmu 7. Obrazy jsme vyčistili algoritmem CLEAN. Na obou obrazech vidíme jasný střed protoplanetárního disku, pouze na pozorování v pásmu 7 je vidět tmavší okolní disk. Struktury vyvolané planetami podobné těm na obrázku 5.24 však nejsou pozorovány.

Na krátkých vlnových délkách se všechny disky ukázaly příliš málo jasné a i při využití algoritmu CLEAN jsme nebyli schopni identifikovat žádné ze struktur, které planety v disku vytvořily. Na delších vlnových délkách je ALMA schopná rozlišit disk, avšak na těchto λ nejsou struktury pozorovatelné ani v ideálních obrazech.

Základní disk — Case III z práce Chrenko a kol. (2017) — není pro pozorování planet dostatečně vyvinutý. Pro pozorovatelnost planet by musela být v disku například otevřená mezera, či bychom se museli obrátit k pozorování v čáře a hledat rychlostí kličky jako v práci Pinte a kol. (2019). Naše modely jsou porovnatelné s diskem DoAr 33 ze souhvězdí Hadonoše (viz tabulka 2.1). Tento disk se nachází 139 pc daleko. Součástí projektu DSHARP (Andrews a kol. 2018) byl měřen v kontinuu na vlnové délce $\lambda = 1250\,\mu$m. Obraz převzatý přímo z práce Andrews a kol. (2018), resp. z `https://almascience.eso.org/almadata/lp/DSHARP/` je vyobrazený na obrázku 5.26. Jas i rozměry našeho disku se řádově podobají DoAr33. U obou pozoruje velmi jasný vnitřní okraj a slabší okolní disk. DoAr33 je jasnější, avšak byl měřen na delší vlnové délce. Oba disky se zdají být v podobné fázi vývoje, neboť nepozorujeme zatím žádné projevy přítomnosti planet.



Obrázek 5.26: Porovnání námi simulovaného pozorování ALMA v kontinuu v pásmu 7 (nahoře) a disku DoAr 33 z projektu DSHARP (Andrews a kol. 2018) pozorovaného v kontinuu v pásmu 6. Disky jsou si řádově podobné veličinami (DoAr 33 je dvakrát větší a má jasnější oblasti dál od hvězdy, od Země je vzdálený 139 pc, námi simulovaný disk 130 pc) i strukturou (oba obrazy jsou dominované jasným vnitřním okrajem, zatímco okolní disk je méně jasný, i když v našem disku je okolí výrazně slabší). Ani v jednom nejsou pozorovatelné žádné projevy přítomnosti planet.

6. Závěr

Výsledek této práce je negativní — struktury, které v málo vyvinutém disku „CASE III“ tvoří planety *nejsou* interferometrem ALMA pro své malé rozměry a špatný kontrast pozorovatelné.

Vytvořili jsme program Faradit (dostupný z `https://sirrah.troja.mff.cuni.cz/~janoska/bakalarka.html`), pomocí něhož lze převádět dvourozměrné modely protoplanetárních disků do tří rozměrů pro výpočet přenosu záření. Programem Radmc jsme přenos záření vypočetli pro pět různých modelů disků, založených na disku označeném jako „Case III“ z práce Chrenko a kol. (2017). Získali jsme ideální obrazy pro vlnové délky 0,1 μm, 1 μm, 10 μm, 100 μm, 500 μm a 870 μm a ukázali, že struktury vytvořené v disku protoplanetami o hmotnosti $3\,M_\oplus$ jsou v principu měřitelné; nejsou zcela skryté v opticky tlustém prostředí. Nástrojem ALMA OST jsme nakonec simulovali interferometrické měření našich disků a ukázali, že při realistickém nastavení interferometru, daném rozlišení, dané úrovni signálu a šumu nejsou tyto struktury přístrojem ALMA v kontinuu pozorovatelné. Došli jsme tedy k podobnému výsledku jako v analogické práci (Ruge a kol., 2013). Předpokládáme, že by disk, respektive planetarní systém, musel být více vyvinutý, aby se otevřela mezera, která by už byla pozorovatelná. Jinou možností by bylo pozorování ve spektrální čáře, které může ukázat poruchu rychlostního pole způsobenou planetou (Pinte a kol., 2019).

Pozorování v kontinuu nemusí být ideální způsob hledání vznikajících planet v protoplanetárních discích. V nedávné práci Öberg a kol. (2021) autoři provedli studii systémů z DSHARP a podobně jako v práci Pinte a kol. (2019) hledali v čarách rychlostní kličky. Pro přidání tohoto efektu do našich simulací bychom museli vyřešit rozšíření rychlostního pole původní hydrodynamické simulace do třetího rozměru. Budoucí prací tedy může být ověření měřitelnosti takových kliček v nevyvinutých discích.

Literatura

Andrews, S. M. a kol. (2009). Protoplanetary disk structures in ophiuchus. *The Astrophysical Journal*, **700**(2), 1502–1523. ISSN 0004-637X. doi: 10.1088/0004-637X/700/2/1502. URL https://doi.org/10.1088/0004-637x/700/2/1502.

Andrews, S. M. a kol. (2018). The Disk Substructures at High Angular Resolution Project (DSHARP). I. Motivation, Sample, Calibration, and Overview. *The Astrophysical Journal*, **869**(2), L41. ISSN 2041-8205. doi: 10.3847/2041-8213/aaf741. URL https://doi.org/10.3847/2041-8213/aaf741.

Armitage, P. J. (2010). *Astrophysics of Planet Formation*. Cambridge University Press, 1 edition. ISBN 0521887453,9780521887458.

Bertotti, B. a kol. (2003). *Physics of the Solar System: Dynamics and Evolution, Space Physics, and Spacetime Structure*. Kulwer Academic Publishers. ISBN 1402014287.

Birnstiel, T. a kol. (2012). A simple model for the evolution of the dust population in protoplanetary disks. *Astronomy & Astrophysics*, **539**, A148. ISSN 0004-6361, 1432-0746. doi: 10.1051/0004-6361/201118136. URL https://www.aanda.org/articles/aa/abs/2012/03/aa18136-11/aa18136-11.html.

Bjorkman, J. E. a Wood, K. (2001). Radiative Equilibrium and Temperature Correction in Monte Carlo Radiation Transfer. *The Astrophysical Journal*, **554**, 615–623. ISSN 0004-637X. doi: 10.1086/321336. URL https://ui.adsabs.harvard.edu/abs/2001ApJ...554..615B. ADS Bibcode: 2001ApJ...554..615B.

Brož, M. (2020). Hydrodynamka v astronomii. V době psaní této práce nevydaná učebnice. URL https://sirrah.troja.mff.cuni.cz/~mira/hydrodynamika/.

Brož, M. a Wolf, M. (2017). *Astronomická měření*. MatfyzPress, 1 edition. ISBN 9788073782368.

Brož, M. a kol. (2018). Dynamics of multiple protoplanets embedded in gas and pebble discs and its dependence on Σ and ν parameters. *Astronomy & Astrophysics*, **620**, A157. ISSN 0004-6361, 1432-0746. doi: 10.1051/0004-6361/201833855. URL https://www.aanda.org/articles/aa/abs/2018/12/aa33855-18/aa33855-18.html.

Brož, M. a Šolc, M. (2013). *Fyzika sluneční soustavy*. MatfyzPress, 1 edition. ISBN 9788073782368.

Chametla, R. O. a kol. (2020). Capture and migration of Jupiter and Saturn in mean motion resonance in a gaseous protoplanetary disc. *Monthly Notices of the Royal Astronomical Society*, **492**(4), 6007–6018. ISSN 0035-8711. doi: 10.1093/mnras/staa260. URL https://doi.org/10.1093/mnras/staa260.

Chapman, S. a kol. (1995). *The mathematical theory of non-uniform gases: an account of the kinetic theory of viscosity, thermal conduction, and diffusion in gases*. Cambridge Mathematical Library. Cambridge University Press, 3 edition. ISBN 052140844X; 9780521408448.

Chrenko, O. a kol. (2017). Eccentricity excitation and merging of planetary embryos heated by pebble accretion. *Astronomy & Astrophysics*, **606**, A114. ISSN 0004-6361, 1432-0746. doi: 10.1051/0004-6361/201731033. URL `https://www.aanda.org/articles/aa/abs/2017/10/aa31033-17/aa31033-17.html`.

Cox, E. G. a kol. (2017). Protoplanetary Disks in ρ Ophiuchus as Seen from ALMA. *The Astrophysical Journal*, **851**(2), 83. ISSN 0004-637X. doi: 10.3847/1538-4357/aa97e2. URL `https://doi.org/10.3847/1538-4357/aa97e2`.

de Boer, J. a kol. (2021). A single-armed spiral in the protoplanetary disk around HD34282 ? *Astronomy & Astrophysics*, **649**, A25. ISSN 0004-6361, 1432-0746. doi: 10.1051/0004-6361/201936787. URL `http://arxiv.org/abs/2010.12202`. arXiv: 2010.12202.

de Laplace, P. (1796). *The System of the World*. Number sv. 1 in The System of the World. University Press for Longmans, Rees, Orme, Brown, and Green. URL `https://books.google.cz/books?id=aygzAQAAMAAJ`. Originál v roce 1830 přeložil H.H. Harte.

Descartes, R. (1677). *The World and Other Writings*. Cambridge Texts in the History of Philosophy. Cambridge University Press. ISBN 9780511001871. Originál v roce 1998 přeložil Stephen Gaukroger.

Dohnanyi, J. S. (1969). Collisional model of asteroids and their debris. *Journal of Geophysical Research (1896-1977)*, **74**(10), 2531–2554. ISSN 2156-2202. doi: 10.1029/JB074i010p02531. URL `https://onlinelibrary.wiley.com/doi/abs/10.1029/JB074i010p02531`.

Dorschner, J. a kol. (1977). The 10-Micron Interstellar Absorption Band: Curves of Growth for Terrestrial and Meteoritic Olivine Particles. *Astrophysics and Space Science*, **48**(2), 305–312. doi: 10.1007/BF00648119. URL `https://ui.adsabs.harvard.edu/abs/1977Ap&SS..48..305D`. Provided by the SAO/NASA Astrophysics Data System.

Dorschner, J. a kol. (1995). Steps toward interstellar silicate mineralogy. II. Study of Mg-Fe-silicate glasses of variable composition. *Astronomy & Astrophysics*, **300**, 503. URL `https://ui.adsabs.harvard.edu/abs/1995A&A...300..503D`. Provided by the SAO/NASA Astrophysics Data System.

Dullemond, C. P. (2012). Radiative Transfer in Astrophysics: Theory, Numerical Methods and Applications. Poznámky z přednášek na Ruprecht-Karlově univerzitě v Heidelbergu. URL `https://www.ita.uni-heidelberg.de/~dullemond/lectures/radtrans_2012/index.shtml`.

Dullemond, C. P. (2013). Theoretical Models of the Structure of Protoplanetary Disks. Poznámky z přednášek Les Houches 2013. URL https://www.ita.uni-heidelberg.de/~dullemond/lectures/leshouches2013.pdf.

Dullemond, C. P. a kol. (2012). RADMC-3D: A multi-purpose radiative transfer tool. *Astrophysics Source Code Library*, page ascl:1202.015. URL https://ui.adsabs.harvard.edu/abs/2012ascl.soft02015D. ADS Bibcode: 2012ascl.soft02015D.

Dullemond, C. P. a kol. (2020). Midplane temperature and outer edge of the protoplanetary disk around HD 163296. *Astronomy & Astrophysics*, **633**, A137. ISSN 0004-6361, 1432-0746. doi: 10.1051/0004-6361/201936438. URL https://www.aanda.org/articles/aa/abs/2020/01/aa36438-19/aa36438-19.html.

Fedele, D. a kol. (2018). ALMA continuum observations of the protoplanetary disk AS 209 - Evidence of multiple gaps opened by a single planet. *Astronomy & Astrophysics*, **610**, A24. ISSN 0004-6361, 1432-0746. doi: 10.1051/0004-6361/201731978. URL https://www.aanda.org/articles/aa/abs/2018/02/aa31978-17/aa31978-17.html.

Flock, M. a kol. (2016). Radiation hydrodynamics models of the inner rim in protoplanetary disks. *The Astrophysical Journal*, **827**(2), 144. ISSN 0004-637X. doi: 10.3847/0004-637X/827/2/144. URL https://doi.org/10.3847/0004-637x/827/2/144.

Goddi, C. a kol. (2019). First M87 Event Horizon Telescope Results and the Role of ALMA. *The Messenger*, **177**, 25–35. ISSN 0722-6691. doi: 10.18727/0722-6691/5150. URL https://ui.adsabs.harvard.edu/abs/2019Msngr.177...24G. ADS Bibcode: 2019Msngr.177...24G.

Hayashi, C. (1981). Structure of the Solar Nebula, Growth and Decay of Magnetic Fields and Effects of Magnetic and Turbulent Viscosities on the Nebula. *Progress of Theoretical Physics Supplement*, **70**, 35–53. ISSN 0375-9687. doi: 10.1143/PTPS.70.35. URL https://doi.org/10.1143/PTPS.70.35.

Jaeger, C. a kol. (1994). Steps toward interstellar silicate mineralogy. I. Laboratory results of a silicate glass of mean cosmic composition. *Astronomy & Astrophysics*, **292**, 641–655. URL https://ui.adsabs.harvard.edu/abs/1994A&A...292..641J. Provided by the SAO/NASA Astrophysics Data System.

Johansen, A. a kol. (2007). Rapid planetesimal formation in turbulent circumstellar disks. *Nature*, **448**, 1022–1025. ISSN 0028-0836. doi: 10.1038/nature06086. URL https://ui.adsabs.harvard.edu/abs/2007Natur.448.1022J. ADS Bibcode: 2007Natur.448.1022J.

Kant, I. (1755). *Universal Natural History and Theory of the Heavens* . ISBN 9781935238911. Originál v roce 2009 přeložil Ian Johnston.

Kitamura, Y. a kol. (2002). Investigation of the Physical Properties of Protoplanetary Disks around T Tauri Stars by a 1 Arcsecond Imaging Survey: Evolution and Diversity of the Disks in Their Accretion Stage*. *The Astrophysical Journal*, **581**(1), 357. ISSN 0004-637X. doi: 10.1086/344223. URL https://iopscience.iop.org/article/10.1086/344223/meta.

Kwon, W. a kol. (2011). Resolving the circumstellar disk of HL Tauri at millimeter wavelengths. *The Astrophysical Journal*, **741**(1), 3. ISSN 0004-637X. doi: 10.1088/0004-637X/741/1/3. URL https://doi.org/10.1088/0004-637x/741/1/3.

Lane, B. F. a Muterspaugh, M. W. (2004). Differential Astrometry of Sub-arcsecond Scale Binaries at the Palomar Testbed Interferometer. *The Astrophysical Journal*, **601**(2), 1129–1135. ISSN 0004-637X, 1538-4357. doi: 10.1086/380760. URL http://arxiv.org/abs/astro-ph/0308381. arXiv: astro-ph/0308381.

Lynden-Bell, D. a Pringle, J. E. (1974). The evolution of viscous discs and the origin of the nebular variables. *Monthly Notices of the Royal Astronomical Society*, **168**, 603–637. ISSN 0035-8711. doi: 10.1093/mnras/168.3.603. URL https://ui.adsabs.harvard.edu/abs/1974MNRAS.168..603L. ADS Bibcode: 1974MNRAS.168..603L.

Magritte, R. (1929). La trahison des images.

Masset, F. S. (2002). The co-orbital corotation torque in a viscous disk: numerical simulations. *Astronomy & Astrophysics*, **387**(2), 605–623. ISSN 0004-6361, 1432-0746. doi: 10.1051/0004-6361:20020240. URL http://arxiv.org/abs/astro-ph/0205211. arXiv: astro-ph/0205211.

Max Born, E. W. (1999). *Principles of optics*. Cambridge University Press, 7th edition. ISBN 9780521642224,0521642221.

Pinte, C. a kol. (2019). Kinematic detection of a planet carving a gap in a protoplanetary disk. *Nature Astronomy*, **3**(12), 1109–1114. ISSN 2397-3366. doi: 10.1038/s41550-019-0852-6. URL https://www.nature.com/articles/s41550-019-0852-6.

Pinte, C. a kol. (2020). Nine Localized Deviations from Keplerian Rotation in the DSHARP Circumstellar Disks: Kinematic Evidence for Protoplanets Carving the Gaps. *The Astrophysical Journal Letters*, **890**(1), L9. ISSN 2041-8205. doi: 10.3847/2041-8213/ab6dda. URL https://doi.org/10.3847/2041-8213/ab6dda.

Pringle, J. E. (1981). Accretion Discs in Astrophysics. *Annual Review of Astronomy and Astrophysics*, **19**(1), 137–160. doi: 10.1146/annurev.aa.19.090181.001033. URL https://doi.org/10.1146/annurev.aa.19.090181.001033.

Riols, A. a Latter, H. (2019). Gravitoturbulent dynamos in astrophysical discs. *Monthly Notices of the Royal Astronomical Society*, **482**(3), 3989–4008. ISSN 0035-8711. doi: 10.1093/mnras/sty2804. URL https://doi.org/10.1093/mnras/sty2804.

Robitaille, T. P. (2010). On the modified random walk algorithm for Monte-Carlo radiation transfer. *Astronomy & Astrophysics*, **520**, A70. ISSN 0004-6361, 1432-0746. doi: 10.1051/0004-6361/201015025. URL https://www.aanda.org/articles/aa/abs/2010/12/aa15025-10/aa15025-10.html.

Ruge, J. P. a kol. (2013). Tracing large-scale structures in circumstellar disks with ALMA. *Astronomy & Astrophysics*, **549**, A97. ISSN 0004-6361, 1432-0746. doi: 10.1051/0004-6361/201220390. URL https://www.aanda.org/articles/aa/abs/2013/01/aa20390-12/aa20390-12.html.

Schobert, B. N. a kol. (2019). The Impact of Accretion Heating and Thermal Conduction on the Dead Zone of Protoplanetary Disks. *The Astrophysical Journal*, **881**(1), 56. ISSN 0004-637X. doi: 10.3847/1538-4357/ab2df6. URL https://doi.org/10.3847/1538-4357/ab2df6.

Shakura, N. I. a Sunyaev, R. A. (1973). Black holes in binary systems. observational appearance. *Astronomy and Astrophysics*, **24**, 337–355. doi: 10.1007/978-94-010-2585-0_13.

Sutherland, W. (1893). LII. The viscosity of gases and molecular force. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, **36**(223), 507–531. ISSN 1941-5982. doi: 10.1080/14786449308620508. URL https://doi.org/10.1080/14786449308620508.

Swedenborg, E. (1734). *The Principia*. Swedenborg Scientific Association. ISBN 0-91522120-G. Originál v roce 1988 přeložil Augustus Clissold.

Testi, L. a kol. (2014). Dust Evolution in Protoplanetary Disks. *arXiv:1402.1354 [astro-ph]*. doi: 10.2458/azu_uapress_9780816531240-ch015. URL http://arxiv.org/abs/1402.1354. arXiv: 1402.1354.

Thompson, A. R. a kol. (2017). *Interferometry and Synthesis in Radio Astronomy*. Astronomy and Astrophysics Library. Springer International Publishing, 3 edition. ISBN 978-3-319-44429-1,978-3-319-44431-4,978-0-471-25492-8.

Zhu, Z. a kol. (2012). Dust filtration by planet-induced gap edges: implications for transitional disks. *The Astrophysical Journal*, **755**(1), 6. doi: 10.1088/0004-637x/755/1/6. URL https://doi.org/10.1088/0004-637x/755/1/6.

Öberg, K. I. a kol. (2021). Molecules with ALMA at Planet-forming Scales (MAPS). I. Program Overview and Highlights. *The Astrophysical Journal Supplement Series*, **257**, 1. ISSN 0067-0049. doi: 10.3847/1538-4365/ac1432. URL https://ui.adsabs.harvard.edu/abs/2021ApJS..257....1O. ADS Bibcode: 2021ApJS..257....1O.

A. Přílohy

A.1 Výpis programu disk.py

Program použitý v této práci. Dostupný online z `https://sirrah.troja.mff.cuni.cz/~janoska/bakalarka.html`.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
"""
FaRaDiT.py
Disk module to handle Fargo a Radmc programs.
"""

import sys
import numpy as np
import matplotlib.pyplot as plt
from numpy.core.numeric import identity
try:
    from radmc3dPy import image
except ImportError: print("radmc3dPy package not found! Image function via python won't work, you may need to call directly radmc3d image")
from scipy.optimize import curve_fit
from scipy import integrate
import os
import time

G = 6.67430e-11     #[SI]  CODATA 2018
au = 1.495978707e11 #[m]  IAU 2012
M_sun = 1.9884e30   #[kg]  ASA 2018
cm = 1e-2
micron = 1e-6
gram = 1e-3
R_gas = 8.314462618     #[J*K^-1*mol^-1] CODATA 2018
mu = 0.0024             #Mean molecualr weight in kg*mol^-1
planck = 6.62607015e-34 #[J*s] CODATA 2018
k_B = 1.3806416e-23      #[J*K^-1] CODATA 2018
light_speed = 299792458 #[m*s^-1] CODATA 2018
steff = 5.6703744116e-8  #[W*m^-2*K^-4] CODATA 2018
sol_rad = 6.957e8       #[m] IAU 2015 https://arxiv.org/pdf/1510.07674.pdf
erg = 1e-7              #[J]

__author__ = "Ondrej Janoska"
__version__ = "April 26th 2022"

def is_number(s):
    """Returns True if input is a number"""
    try:
        float(s)
        return True
    except ValueError:
        return False

def polynomial_smooth_min(a, b, smoothing = 1):
    """
    Returns the smaller value between a and b in such a way that if a = f(x) and b = g(x), smooth_min(f, g)(x) will be smooth.
    """
    h = np.clip( 0.5+0.5*(b-a)/smoothing, 0.0, 1.0 )
    return a*h+b*(1-h)-smoothing*h*(1-h)

def exponential_smooth_min(a, b, smoothing = 1):
    res = np.exp2(-smoothing*a)+np.exp2(-smoothing*b)
    return -(np.log2(res))/smoothing

def smooth_min(a, b, smoothing = 1):
    """
    For two positive functions a,b in C^n, returns a function f that's at worst C^n-1 such that f<=min{a,b}

    (wolfram input:
    d/dx((f[x]^n*g[x]^n/(f[x]^n+g[x]^n))^(1/n))//FullSimplify
    )
    """
    a, b = np.power(a, smoothing), np.power(b, smoothing)
    return np.power((a*b)/(a+b), 1.0/smoothing)

def f_z(h=0, sigma=0, T = 0, equatorial_r=0, M_star=M_sun, getH = False): #[SI]
    """
    Returns volumetric density. Vertically isothermal model according to Chrenko etal. (2017), A&A, 606, A114.
    --------------------
    Parameters:
    --------------------
    h [m]: float, height of the point of interest above the disk equatorial plane

    sigma [kg/m^2]: float, surface density of the dust at the point in equatorial plane under the point of interest

    T [K]: float, temperature of the dust at the point in equatorial plane under the point of interest

    equatorial_r [m]: float, distance from the star to the point in equatorial plane under the point of interest

    M_star [kg]: float, mass of the star at the center

```

```
    getH: bool, if True, returns the scale height in [m] instead of density
    """

    H = np.sqrt(R_gas*T*equatorial_r**3/(G*M_star*mu))
    if getH: return H
    return sigma/np.sqrt(2*np.pi)/H*np.exp(-0.5*h*h/H/H)

def identity_T(h=0, sigma=0, T = 0, equatorial_r=0, M_star=M_sun, getH = False):
    return T

def index_before(array = None, value = 0):
    """Returns the last index "i" at which monotically ascending "array" is smaller than "value" """
    if array is None:
        raise ValueError("index_around error: No array was passed!")
    i = 0
    while (i<len(array)) and (array[i] <= value):
        i+=1
    return i

def is_file_binary(filename):
    """
    Slow binary checker --- goes character by character till it finds one, that it can't convert to utf-8, which it interprets as it being a binary file symbol
    """
    try:
        with open(filename, "r") as f:
            for l in f:
                l.encode("utf-8")
        return False
    except UnicodeDecodeError:
        return True

def black_body(wavelength, T = 4000):
    """
    Idealized blackbody spectrum a.k.a. Planck's law.
    reference: Harmanec, Broz, Stavba a vyvoj hvezd
    """
    return 2*planck*light_speed**2/(wavelength**5*(np.exp(planck*light_speed/(wavelength*k_B*T))-1))


class Disk():
    """Disk class to handle Fargo input/Radmc output."""

    def __init__(self):
        """
        Initialize disk.
        """

        self.dust_density = np.array([])
        self.dust_species = []
        self.gas_density = np.array([])
        self.T = np.array([])
        self.par = {
        'Nrad' : 1024,  # Number of radial sectors compatible with fargo input
        'Nsec' : 1536,  # Number of angular sectors
        'Nthet': 0,     # Number of sectors in the theta direction
        'Thetamin' : 0,
        'Thetamax' : 0.5*np.pi,
        'grid_type' : 100, # grid_type: 0 => cartesian, 100 => spherical
        'M_star' : np.array([1*M_sun]), # Mass of the star(s) in kg
        'EffectiveTemperature': 4370, # effective surface temperature of the star in K
        'StellarRadius': 1, # radius of the star in Solar radii
        }

        self.Rmin = 2.8   # Boundary of the disk in au
        self.Rmax = 14    # *ditto*
        self.Phimin = 0
        self.Phimax = 2*np.pi

        self.dtheta = 0
        self.dphi = (self.Phimax - self.Phimin)/self.par["Nsec"]
        self.dr = (self.Rmax-self.Rmin)/self.par["Nrad"]

        self.r_inf = np.linspace(self.Rmin, self.Rmax-self.dr, self.par["Nrad"])
        self.r_med = np.linspace(self.Rmin+0.5*self.dr, self.Rmax-0.5*self.dr, self.par["Nrad"])
        self.r_sup = np.linspace(self.Rmin+self.dr, self.Rmax, self.par["Nrad"])


        self.phi_inf = np.linspace(self.Phimin, self.Phimax-self.dphi, self.par["Nsec"])
        self.phi_med = np.linspace(self.Phimin+0.5*self.dphi, self.Phimax-0.5*self.dphi, self.par["Nsec"])
        self.phi_sup = np.linspace(self.Phimin+self.dphi, self.Phimax, self.par["Nsec"])

        self.theta_inf = None   # None => disk is flat
        self.theta_med = None
        self.theta_sup = None

    ## ---------------------------------------------------
    ## Debugging procedures
    ## ---------------------------------------------------

    def print_params(self):
        """
        Prints out self.par dictionary containing basic parameters of disk
        """
        for name, value in self.par.items():
            if isinstance(value, list) or isinstance(value, np.ndarray):
                print(name, '\tIs present as a list of ', len(value), ' values')
            else:
                if is_number(str(value)):
```

```
                print(name, '\t', '{:.2e}'.format(value))
            else:
                print(name, '\t', value)

    def plot_density_slice(self, phi = 0, thickness = f_z, fine = 100, maxheight = None, logarithmic_scale=False, note = "", show = True, fig = None):
        """
        Creates 2D colormesh vertical plot of the disk's volumetric density along a ray coming from the star
        --------------------
        Parameters:
        --------------------
        phi [deg]: float, angular coordinate of the ray along which the plot is to be drawn

        thickness: function, by which the volumetric density around the disk is determined. It is expected to take parameters of the point above which the density is needed:
            h [m]: float, height of the point of interest above the disk
            equatorial_r [m]: float, distance from the star to the point in equatorial plane under the point of interest
            T [K]: float, temperature of the dust at the point in equatorial plane under the point of interest
            sigma [kg/m^2]: float, surface density of the dust at the point in equatorial plane under the point of interest
            M_star [kg]: float, mass of the star at the center

        fine: int, resolution in the vertical direction

        maxheight [au]: total height of the plot

        logarithmic_scale: bool, if True, log scale will be used on the color (z) axis

        r_range: array-like, list containing boundaries of desired graph,

        note: string, note to be written under the title,

        show: bool, if set to True the plot will be immediatelly shown. Otherwise plt.show() or other method with show = True has to be used later

        fig: matplotlib figure, figure to which draw the plot

        """
        if fig is None: fig = plt.figure()

        index = index_before(array = self.phi_med, value = phi*np.pi/180)
        if self.dust_density.ndim == 2:
            if maxheight is None: maxheight = 2
            sigma_seed = self.dust_density[:,index]
            T_seed = self.T[:,index]
            density_slice = np.array([[]])
            dz = maxheight/(2*fine+1)
            for i in range(-fine,fine+1):
                stack = [thickness(sigma = sigma_seed[j], T = T_seed[j], h = i*dz*au, equatorial_r = self.r_med[j]*au) for j in range(len(sigma_seed))]
                if (i == -fine): density_slice = stack
                else: density_slice = np.vstack((density_slice, stack))
            equatorial_r = np.insert(self.r_sup, 0, self.r_inf[0])
            z = np.linspace(-fine*dz, fine*dz, 2*fine+2)
            r_mesh, z_mesh = np.meshgrid(equatorial_r,z)
            ax = fig.add_subplot(111)
            if logarithmic_scale:
                import matplotlib.colors
                ap = ax.pcolormesh(r_mesh,z_mesh,density_slice, norm=matplotlib.colors.LogNorm(), cmap = "hot")
            else: ap = ax.pcolormesh(r_mesh ,z_mesh, density_slice, cmap = "hot")

        elif self.dust_density.ndim == 3:
            if maxheight is None: maxheight = 0
            else: maxheight = (len(self.theta_med)-maxheight//2)
            density = np.append(self.dust_density[:,maxheight:,index], self.dust_density[:,maxheight:,index][:,::-1], axis = 1)
            theta_bound = np.append(self.theta_inf[maxheight:], self.theta_sup[-1])
            theta_bound = np.append(theta_bound, np.pi-self.theta_inf[maxheight:][::-1])

            r_mesh, theta_mesh = np.meshgrid(np.append(self.r_inf, self.r_sup[-1]), theta_bound)
            x_mesh = r_mesh*np.sin(theta_mesh)
            y_mesh = r_mesh*np.cos(theta_mesh)

            ax = fig.add_subplot(111)
            ax.set_aspect('equal')
            if logarithmic_scale:
                import matplotlib.colors
                ap = ax.pcolormesh(x_mesh, y_mesh, density.T, norm=matplotlib.colors.LogNorm(), cmap = "hot")
            else: ap = ax.pcolormesh(x_mesh, y_mesh, density.T, cmap = "hot")
        else: raise ValueError("Strange dimension of density array")

        plt.xlabel("r [au]")
        plt.ylabel("z [au]")
        plt.title(f"Gas density at phi = {phi}  extdegree\n{note}")
        plt.colorbar(ap, label = "kg*m^-3")
        if show: plt.show()

    def plot_T_slice(self, phi = 0, fine = 100, maxheight = None, logarithmic_scale=False, note = "", show = True, fig = None, species = 0):
        """
        Creates 2D colormesh vertical plot of the disk's temperature along a ray coming from the star
        --------------------
        Parameters:
        --------------------
        phi [deg]: float, angular coordinate of the ray along which the plot is to be drawn

        fine: int, resolution in the vertical direction

```

```
        maxheight [au]: total height of the plot

        logarithmic_scale: bool, if True, log scale will be used on the color (z) axis

        r_range: array-like, list containing boundaries of desired graph,

        note: string, note to be written under the title,

        show: bool, if set to True the plot will be immediatelly shown. Otherwise plt.show() or other method with show = True has to be used later

        fig: matplotlib figure, figure to which draw the plot

        species: int, which dust species to plot
        """

        if fig is None: fig = plt.figure()

        index = index_before(array = self.phi_med, value = phi*np.pi/180)

        if self.T.ndim == 3 or self.T.ndim == 4:
            if maxheight is None: maxheight = 0
            else: maxheight = (len(self.theta_med)-maxheight//2)
            if self.T.ndim == 4:
                Temp = np.append(self.T[species,:,maxheight:,index], self.T[species,:,maxheight:,index][:,::-1], axis = 1)
            else:
                Temp = np.append(self.T[:,maxheight:,index], self.T[:,maxheight:,index][:,::-1], axis = 1)
            theta_bound = np.append(self.theta_inf[maxheight:], self.theta_sup[-1])
            theta_bound = np.append(theta_bound, np.pi-self.theta_inf[maxheight:][::-1])

            r_mesh, theta_mesh = np.meshgrid(np.append(self.r_inf, self.r_sup[-1]), theta_bound)
            x_mesh = r_mesh*np.sin(theta_mesh)
            y_mesh = r_mesh*np.cos(theta_mesh)

            ax = fig.add_subplot(111)
            ax.set_aspect('equal')
            if logarithmic_scale:
                import matplotlib.colors
                ap = ax.pcolormesh(x_mesh, y_mesh, Temp.T, norm=matplotlib.colors.LogNorm(), cmap = "hot")
            else: ap = ax.pcolormesh(x_mesh, y_mesh, Temp.T, cmap = "hot")
        else: raise ValueError("Strange dimension of temperature array")

        plt.xlabel("r [au]")
        plt.ylabel("z [au]")
        plt.title(f"Gas temperature at phi = {phi}  extdegree\n{note}")
        plt.colorbar(ap, label = "K")
        if show: plt.show()

    def plot_fargo_gastemper(self, Mean = True, note = "", show = True):
        """
        Creates 2D colormesh plot of the disk's equatorial temperature
        --------------------
        Parameters:
        --------------------
        mean: bool, if True, the radial gradient is substracted from the temperature array so fine structure can be observed

        note: string, note to be written under the title,

        show: bool, if set to True the plot will be immediatelly shown. Otherwise plt.show() or other method with show = True has to be used later
        """

        phi = np.insert(self.phi_sup, 0, self.phi_inf[0])
        equatorial_r = np.insert(self.r_sup, 0, self.r_inf[0])
        phi_mesh, r_mesh = np.meshgrid(phi, equatorial_r)
        x = r_mesh*np.cos(phi_mesh)
        y = r_mesh*np.sin(phi_mesh)
        fig = plt.figure()
        ax = fig.add_subplot(111)
        ax.set_aspect('equal')
        mean = 0
        if Mean:
            mean = np.zeros((self.par["Nsec"]+1, self.par["Nrad"]+1))
            mean = [np.average(self.T, axis=1) for i in range(self.par["Nsec"])]
            mean = np.transpose(mean,(1,0))

        ap = ax.pcolormesh(x,y,self.T-mean)
        plt.title("Gas temperature" + Mean*" with subtracted radial gradient" + "\n"+note)
        plt.colorbar(ap, label = "K")
        if show: plt.show()

    def plot_fargo_gasdens(self, logarithmic_scale=False, note="", show = True):
        """
        Creates 2D colormesh plot of the disk's surface density
        --------------------
        Parameters:
        --------------------
        logarithmic_scale: bool, if True, log scale will be used on the color (z) axis

        note: string, note to be written under the title,

        show: bool, if set to True the plot will be immediatelly shown. Otherwise plt.show() or other method with show = True has to be used later
        """

        phi = np.insert(self.phi_sup, 0, self.phi_inf[0])
        equatorial_r = np.insert(self.r_sup, 0, self.r_inf[0])
```

```
        phi_mesh, r_mesh = np.meshgrid(phi, equatorial_r)
        x = r_mesh*np.cos(phi_mesh)
        y = r_mesh*np.sin(phi_mesh)
        fig = plt.figure()
        ax = fig.add_subplot(111)
        ax.set_aspect('equal')
        if self.gas_density.ndim == 2: sigma = self.gas_density
        elif self.dust_density.ndim == 3: sigma = self.dust_density[:,(len(self.theta_med)-1),:]
        if logarithmic_scale:
            import matplotlib.colors
            ap = ax.pcolormesh(x,y,sigma, norm=matplotlib.colors.LogNorm(), cmap = "hot")
        else: ap = ax.pcolormesh(x,y,sigma, cmap = "hot")
        plt.title(f"Gas density\n{note}")
        plt.xlabel("au")
        plt.ylabel("au")
        plt.colorbar(ap, label = "kg*m^-2")
        if show: plt.show()

    def plot_T_profile(self, along = "ray", r = 4, phi = 0, logarithmic_scale = False, r_range = None, theta_range = None, z_range = None, note = "", show = True, fig = None, label = None, species = 0):
        """
        Plots temperature along a set curve
        --------------------
        Parameters:
        --------------------
        along: string, determines along which curve to plot the profile. Possibilities are:
            "ray" - line from star through the equatorial plane
            "perpendicular" - line perpendicular to the equatorial plane
            "arc" - curve in 3D spherical coordinates with theta being the variable

        phi [deg]: float, angular coordinate of the ray along which the plot is to be drawn

        r [au]: float, radial coordinate of the ray along which the plot is to be drawn (if arc == "arc" or "perpendicular")

        logarithmic_scale: bool, if True, log scale will be used on the y axis

        r_range [au]: array-like, list containing boundaries of desired graph (if arc == "arc"),

        theta_range [deg]: array-like, list containing boundaries of desired graph (if arc == "arc"),

        z_range [au]: array-like, list containing boundaries of desired graph (if along == "perpendicular")

        note: string, note to be written under the title,

        show: bool, if set to True the plot will be immediatelly shown. Otherwise plt.show() or other method with show = True has to be used later

        fig: pyplot figure, which fig the plot is supposed to be added to

        species: int, which dust species to plot
        """

        if fig is None: plt.figure()
        else: plt.figure(fig.number)
        plt.subplot(111)

        phi_index = index_before(self.phi_med, phi/180*np.pi)

        if along.casefold() == "arc":
            if self.T.ndim == 2:
                raise ValueError("T plot error: Temperature field is not 3D! Can't plot the third dimension!")
            if theta_range is None:
                start = 0
                end = len(self.theta_med)-1
            else:
                theta_range = np.deg2rad(theta_range)
                start = index_before(array = np.append(self.theta_med, np.pi-self.theta_med[::-1]), value = theta_range[0]) #all this strange reflected append is cuz the data is assumed symmetric around theta = pi/2
                end = index_before(array = np.append(self.theta_med, np.pi-self.theta_med[::-1]), value = theta_range[1])
            r_index = index_before(array = self.r_med, value = r)
            x = np.rad2deg(self.theta_med[start:end])
            if self.T.ndim == 4:
                data = np.append(self.T[species, :,:,:], self.T[species, :,::-1,:], axis = 1)[r_index, start:end, phi_index]
            else:
                data = np.append(self.T[:,:,:], self.T[:,::-1,:], axis = 1)[r_index, start:end, phi_index]

            plt.title(f"Temperature of the disk along a theta arc\n at phi = {phi}  extdegree and r = {r} au"+f"\n{note}"*(len(note)>0))
            plt.xlabel("theta [deg]")

        elif along.casefold() == "perpendicular":
            from scipy.interpolate import griddata

            r_mesh, theta_mesh = np.meshgrid(self.r_med, np.append(self.theta_med, np.pi-self.theta_med[::-1]))
            souradnice_sph = np.column_stack((r_mesh.ravel(), theta_mesh.ravel()))

            x = souradnice_sph[:,0]*np.sin(souradnice_sph[:,1])
            y = souradnice_sph[:,0]*np.cos(souradnice_sph[:,1])

            points = np.column_stack((x,y))
            if self.T.ndim == 3:
                temps =  np.append(self.T[:, :, phi_index],self.T[:, ::-1, phi_index], axis = 1).T.ravel()
            elif self.T.ndim == 4:
                temps =  np.append(self.T[species,:, :, phi_index],self.T[species,:, ::-1, phi_index], axis = 1).T.ravel()

```

```
            if z_range is None: z_range = [-r*np.sin(self.theta_inf[0]), r*np.sin(self.theta_inf[0])]
            ys = np.linspace(z_range[0], z_range[1], num = 10000)
            xs = r*np.ones(shape = 10000)

            interp = griddata(points, temps, (xs,ys), method = "nearest")

            plt.title(f"Temperature of the disk along a line perpendcular to the disk\n at phi = {phi}  extdegree and r = {r} au"+f"\n{note}"*(len(note)>0))
            x = ys
            data = interp
            plt.xlabel("z [au]")

        elif along.casefold() == "ray":
            if r_range is None:
                start = 0
                end = len(self.T[:,phi_index])-1
            else:
                start = index_before(array = self.r_med, value = r_range[0])
                end = index_before(array = self.r_med, value = r_range[1])
            x = self.r_med[start:end]
            if self.T.ndim == 2:
                data = self.T[start:end,phi_index]
            elif self.T.ndim == 3:
                theta_midplane = (len(self.theta_med)-1)
                data = self.T[start:end, theta_midplane, phi_index]
            elif self.T.ndim == 4:
                theta_midplane = (len(self.theta_med)-1)
                data = self.T[species,start:end, theta_midplane, phi_index]

            plt.xlabel("r [au]")
            plt.title(f"Equatorial temperature of the disk at phi = {phi} extdegree"+f"\n{note}"*(len(note)>0))
        else: raise ValueError(f"T plot error: Unknow option '{along}'! Only accepted options are 'ray', 'perpendicular' and 'arc'")


        plt.plot(x, data, label = label)
        if logarithmic_scale: plt.yscale('log')
        plt.ylabel("T [K]")

        if show: plt.show()

    def plot_density_profile(self, along = "ray", r = 4, phi = 0, logarithmic_scale = False, r_range = None, theta_range = None, z_range = None, note = "", show = True, fig = None, label = None, species = 0):
        """
        Plots gas density along a set curve
        --------------------
        Parameters:
        --------------------
        along: string, determines along which curve to plot the profile. Possibilities are:
            "ray" - line from star through the equatorial plane
            "perpendicular" - line perpendicular to the equatorial plane
            "arc" - curve in 3D spherical coordinates with theta being the variable

        phi [deg]: float, angular coordinate of the ray along which the plot is to be drawn

        r [au]: float, radial coordinate of the ray along which the plot is to be drawn (if arc == "arc" or "perpendicular")

        logarithmic_scale: bool, if True, log scale will be used on the y axis

        r_range [au]: array-like, list containing boundaries of desired graph (if arc == "arc"),

        theta_range [deg]: array-like, list containing boundaries of desired graph (if arc == "perpendicular"),

        z_range [au]: array-like, list

        note: string, note to be written under the title,

        show: bool, if set to True the plot will be immediatelly shown. Otherwise plt.show() or other method with show = True has to be used later

        fig: pyplot figure, which fig the plot is supposed to be added to

        species: int, which dust species to plot
        """

        if fig is None: plt.figure()
        else: plt.figure(fig.number)
        plt.subplot(111)

        phi_index = index_before(self.phi_med, phi/180*np.pi)

        if along.casefold() == "arc":
            if self.dust_density.ndim == 2:
                raise ValueError("density plot error: Density field is not 3D! Can't plot the third dimension!")
            if theta_range is None:
                start = 0
                end = len(self.theta_med)-1
            else:
                theta_range = np.deg2rad(theta_range)
                start = index_before(array = np.append(self.theta_med, np.pi-self.theta_med[::-1]), value = theta_range[0]) #all this strange reflected append is cuz the data is assumed symmetric around theta = pi/2
                end = index_before(array = np.append(self.theta_med, np.pi-self.theta_med[::-1]), value = theta_range[1])
            r_index = index_before(array = self.r_med, value = r)
            x = np.rad2deg(self.theta_med[start:end])
            if self.dust_density.ndim == 4:
                data = np.append(self.dust_density[species, :,:,:], self.dust_density[species,:,::-1,:], axis = 1)[r_index, start:end, phi_index]
```

```
            elif self.dust_density.ndim == 3:
                data = np.append(self.dust_density[:,:,:], self.dust_density[:,::-1,:], axis = 1)[r_index, start:end, phi_index]

            plt.title(f"Density of the disk's dust along a theta arc\n at phi = {phi} extdegree and r = {r} au"+f"\n{note}"*(len(note)>0))
            plt.xlabel("theta [deg]")

        elif along.casefold() == "perpendicular":
            from scipy.interpolate import griddata

            r_mesh, theta_mesh = np.meshgrid(self.r_med, np.append(self.theta_med, np.pi-self.theta_med[::-1]))
            souradnice_sph = np.column_stack((r_mesh.ravel(), theta_mesh.ravel()))

            x = souradnice_sph[:,0]*np.sin(souradnice_sph[:,1])
            y = souradnice_sph[:,0]*np.cos(souradnice_sph[:,1])

            points = np.column_stack((x,y))
            if self.dust_density.ndim == 3:
                rhos =  np.append(self.dust_density[:, :, phi_index], self.dust_density[:, ::-1, phi_index], axis = 1).T.ravel()
            if self.dust_density.ndim == 4:
                rhos =  np.append(self.dust_density[species,:, :, phi_index], self.dust_density[species:, ::-1, phi_index], axis = 1).T.ravel()


            if z_range is None: z_range = [-r*np.sin(self.theta_inf[0]), r*np.sin(self.theta_inf[0])]
            ys = np.linspace(z_range[0], z_range[1], num = 10000)
            xs = r*np.ones(shape = 10000)

            interp = griddata(points, rhos, (xs,ys), method = "nearest")

            plt.title(f"Density of the disk's dust along a line perpendcular to the disk\n at phi = {phi} extdegree and r = {r} au"+f"\n{note}"*(len(note)>0))
            x = ys
            data = interp
            plt.xlabel("z [au]")

        elif along.casefold() == "ray":
            if r_range is None:
                start = 0
                end = len(self.dust_density[:,phi_index])-1
            else:
                start = index_before(array = self.r_med, value = r_range[0])
                end = index_before(array = self.r_med, value = r_range[1])
            x = self.r_med[start:end]
            if self.dust_density.ndim == 2:
                data = self.dust_density[start:end,phi_index]
            elif self.dust_density.ndim == 3:
                theta_midplane = (len(self.theta_med)-1)
                data = self.dust_density[start:end, theta_midplane, phi_index]
            elif self.dust_density.ndim == 4:
                theta_midplane = (len(self.theta_med)-1)
                data = self.dust_density[species, start:end, theta_midplane, phi_index]

            plt.xlabel("r [au]")
            plt.title(f"Equatorial density of the disk's dust at phi = {phi}  extdegree"+f"\n{note}"*(len(note)>0))
        else: raise ValueError(f"density plot error: Unknow option '{along}'! Only accepted options are 'ray', 'perpendicular' and 'arc'")


        plt.plot(x, data, label = label)
        if logarithmic_scale: plt.yscale('log')

        if self.dust_density.ndim == 3 or self.dust_density.ndim == 4: plt.ylabel(r"\rho [kg*m^-3]")
        elif self.dust_density.ndim == 2: plt.ylabel(r"\rho [kg*m^-2]")
        else: raise ValueError(f"Strange dimension of density {self.dust_density.ndim}, what's going on?")

        if show: plt.show()

    def plot_grid(self, fig = None, show = True, flat = True):
        """
        Plots points of cell centers or boundaries
        --------------------
        Parameters:
        --------------------
        fig: matplotlib figure into which to draw

        show: bool, if True, will immediately show the figure

        flat: bool, if True will only show a flat slice, if False, will show a 3D plot
        """
        if fig is None: fig = plt.figure()
        else: plt.figure(fig.number)

        theta_points = np.append(self.theta_med, np.pi-self.theta_med[::-1])

        if not flat:
            r_mesh, theta_mesh, phi_mesh = np.meshgrid(self.r_med, theta_points, self.phi_med)

            souradnice_sph = np.column_stack((r_mesh.ravel(), theta_mesh.ravel(), phi_mesh.ravel()))

            x = souradnice_sph[:,0]*np.sin(souradnice_sph[:,2])*np.sin(souradnice_sph[:,1])
            y = souradnice_sph[:,0]*np.cos(souradnice_sph[:,2])*np.sin(souradnice_sph[:,1])
            z = souradnice_sph[:,0]*np.cos(souradnice_sph[:,1])

            from mpl_toolkits.mplot3d import Axes3D
            ax = Axes3D(fig, auto_add_to_figure = False)
            fig.add_axes(ax)
```

```
            ax.scatter(x,y,z, label = "Grid centres")
            ranges = (np.max(x)-np.min(x), np.max(y)-np.min(y), np.max(z)-np.min(z)) # used to set equal aspect ratio
            ax.set_box_aspect(ranges)
            ax.set_xlabel("x [au]")
            ax.set_ylabel("y [au]")
            ax.set_zlabel("z [au]")
            plt.legend()


        else:
            r_mesh, theta_mesh = np.meshgrid(self.r_med, theta_points)
            souradnice_sph = np.column_stack((r_mesh.ravel(), theta_mesh.ravel()))
            x = souradnice_sph[:,0]*np.sin(souradnice_sph[:,1])
            y = souradnice_sph[:,0]*np.cos(souradnice_sph[:,1])

            plt.scatter(x,y, s = 0.5, label = "Gridpoint centers")
            plt.legend()
            plt.ylabel("z [au]")
            plt.xlabel("r [au]")
        if show: plt.show()

    def optical_depth(self, how_close_to_equat = None, phi = 0, from_star = False, equatorial_r = 4, wav = 870, thickness = f_z, filename = None, message = True):
        """
        Calculates (and returns) optical depth along a line perpendicular to the equatorial plane of the disk at a chosen point or a line from the star through the midplane
        --------------------
        Parameters:
        --------------------
        how_close_to_equat [m]: float, if from_star == False, how close to the equatorial plane does the ray origin from. If None will return depth along a ray coming all the way through the disc i.e. -10 au to 10 au (expected for those to be far enough to be "infinite" --- might be a problem at some point but rn I couldn't be bothered; it's 3 AM and I have too much work left to do. Good luck future user! :^)  )

        phi [deg]: float, phi coordinate of the point of intersection of the disk and the line

        from_star: bool, if False, the line will be perpendicular to the midplane and cross it at 'equatorial_r' [au]. If True, the line of sight will go from the star through the midplane of the disk and 'equatorial_r' is ignored

        equatorial_r [m]: float, r coordinate of the point of intersection of the disk and the line

        wav [micron]: float, wavelength of the light the optical depth of which is to be calculated

        thickness: function, by which the volumetric density around the disk is determined. It is expected to take parameters of the point above which the density is needed:
            h [m]: float, height of the point of interest above the disk
            equatorial_r [m]: float, distance from the star to the point in equatorial plane under the point of interest
            T [K]: float, temperature of the dust at the point in equatorial plane under the point of interest
            sigma [kg/m^2]: float, surface density of the dust at the point in equatorial plane under the point of interest
            M_star [kg]: float, mass of the star at the center

        filename: string, name of dustkappa* file to be loaded to get the opacity of the disk's dust

        message: bool, if True, will also write the value in terminal
        """
        if filename is None:
            import glob
            files = glob.glob("dustkappa*")
            if len(files) == 0: raise IOError("Optical depth error: Found no candidate for dustkappa file in current directory!")
            elif len(files) > 1:
                print("Optical depth notice: Found multiple candidates for dustkappa file in current directory, using 'silicate'")
                filename = "dustkappa_silicate.inp"
            else: filename = files[0]
        wavs_kappas = np.loadtxt(filename, skiprows = 2, usecols = (0,1,2))
        kappa = np.interp(wav, wavs_kappas[:,0], (wavs_kappas[:,1]+wavs_kappas[:,2])*cm*cm/gram)
        depth = 0
        phi_index = index_before(array = self.phi_med[:], value = phi/180*np.pi)

        if from_star:
            from scipy.interpolate import UnivariateSpline
            spl = UnivariateSpline(self.r_med, thickness(h = 0, equatorial_r=self.r_med, T = self.T[:,phi_index], sigma = self.dust_density[:,phi_index], M_star = self.par["M_star"]))
            depth = kappa*spl.integral(self.Rmin, self.Rmax)
            if message: print(f"Optical depth along a line in the midplane at phi = {phi} deg for wavelength {wav} microns is {depth:.5e}")

        else:
            interpolated_T = np.interp(equatorial_r, au*self.r_med[:], self.T[:, phi_index])
            interpolated_sigma = np.interp(equatorial_r, au*self.r_med[:], self.dust_density[:, phi_index])

            H = thickness(sigma = interpolated_sigma, T = interpolated_T, equatorial_r = equatorial_r, M_star = self.par["M_star"][0], getH=True)

            def integrand(z):
                return kappa*thickness(h = z, sigma = interpolated_sigma, T = interpolated_T, equatorial_r = equatorial_r, M_star = self.par["M_star"][0])

            if how_close_to_equat is None: boundary = -10*au    #that's more or less infinity, right? :^)
            else: boundary = how_close_to_equat

            depth, err = integrate.quad(integrand, boundary, 10*au, points = [-3*H, 0, 3*H])

            if message: print(f"Optical depth for wavelength {wav} microns along a line intersecting midplane at point r = {equatorial_r:.2f} au, phi = {phi} deg is {depth:.5e}")
```

```
        return depth

    def integrate_mass(self, just_dust = True):
        """
        Returns the mass of the disk in kg
        """
        if self.dust_density.ndim == 2 or self.dust_density.ndim==1:
            areas = np.matmul(0.5*np.array([self.phi_sup-self.phi_inf]).T, [(self.r_sup*self.r_sup-self.r_inf*self.r_inf)*au*au])
            if just_dust: mass = np.sum(self.dust_density*areas.T)
            else: mass = np.sum(self.gas_density*areas.T)
            return mass
        else:
            volumes = np.tensordot((np.power(self.r_sup,3)-np.power(self.r_inf, 3))*au*au*au/3 , -np.cos(self.theta_sup)+np.cos(self.theta_inf), axes = 0)
            volumes = np.tensordot( volumes, self.phi_sup-self.phi_inf, axes = 0)
            if just_dust: mass = np.sum(self.dust_density*volumes)
            else: mass = np.sum(self.gas_density*volumes)
            return mass

    ## ------------------------------------------------
    ## Input procedures
    ## ------------------------------------------------

    def fargo_input(self, filename = None):
        """Read Fargo input file in.par"""
        if filename is None:
            import glob
            filename = glob.glob("in.par*")[0]
            print(f"Found parameter file {filename}, I'm loading it...")
        with open(filename, 'r') as file:
            for line in file:
                if '#' not in line:
                    #'''
                    words = line.split()
                    name =''
                    value = ''
                    try:
                        name = words[0]
                    except Exception: pass
                    try:
                        value = words[1]
                    except Exception: pass
                    if is_number(value): self.par.update({f'{name}' : float(value)})
                    else: self.par.update({f'{name}' : value})
        try:
            self.Rmin = self.par["Rmin"]
        except IndexError:
            pass
        try:
            self.Rmax = self.par["Rmax"]
        except IndexError:
            pass
        try:
            self.par["Nrad"] = int(self.par["Nrad"])
        except ValueError:
            pass
        try:
            self.par["Nsec"] = int(self.par["Nsec"])
        except ValueError:
            pass

    def fargo_read_field(self, filename):
        """Read 1 Fargo binary file == field;
        makes the array 2D in order processed[r,phi]"""

        loaded = np.fromfile(filename, dtype=np.double)
        processed = np.reshape(loaded, (len(self.r_med), len(self.phi_med)))
        return(processed)

    def fargo_read_fields(self, no=0, Nrad = 1024, Nsec = 1536, Rmin = 2.8, Rmax = 14):
        """
        Reads Fargo file with "no" number, updates internal parameters acoordingly. This procedure expects polar coordinates in the read file with cells written from the inside out, ring after ring.
        --------------------
        Parameters:
        --------------------
        no: int, number in fargou output filename

        Nrad: int, number of cells in the radial direction in the fargo file

        Nsec: int, number of cells in the angular direction in the fargo file

        Rmin [au]: float, inner radius of the disk

        Rmax [au]: float, outer radius of the disk
        """

        # conversion factors from O. Chrenko and his fargo thorin
        GM_SI = 1.32712440018e20
        AU_SI = 149597870700.0
        R_STANDARD = 8.3144598
        MMW = 2.4
        R_SI = R_STANDARD/(MMW*0.001)
        T2SI = GM_SI/R_SI/AU_SI

        self.par["Nrad"] = Nrad
        self.par["Nsec"] = Nsec
        self.Rmin = Rmin
```

```
        self.Rmax = Rmax

        self.r_inf = np.linspace(self.Rmin, self.Rmax-self.dr, self.par["Nrad"])
        self.r_med = np.linspace(self.Rmin+0.5*self.dr, self.Rmax-0.5*self.dr, self.par["Nrad"])
        self.r_sup = np.linspace(self.Rmin+self.dr, self.Rmax, self.par["Nrad"])

        self.phi_inf = np.linspace(self.Phimin, self.Phimax-self.dphi, self.par["Nsec"])
        self.phi_med = np.linspace(self.Phimin+0.5*self.dphi, self.Phimax-0.5*self.dphi, self.par["Nsec"])
        self.phi_sup = np.linspace(self.Phimin+self.dphi, self.Phimax, self.par["Nsec"])

        self.gas_density = self.fargo_read_field("gasdens%d.dat" % (no))
        self.T = self.fargo_read_field("gastemper%d.dat" % (no))

        self.gas_density = self.gas_density/au/au*M_sun
        self.T = self.T*T2SI
        self.total_mass = self.integrate_mass(just_dust = False)/M_sun

        print(f"Loaded files gasdens{no}.dat and gastemper{no}.dat. Total mass of the disk is {self.total_mass:.4e} M_sun")

    def flat_relax(self, angular = 4, radial = 512, log = True):
        r"""
        Reduces the amount of angular and radial sectors in the disk.
        --------------------
        Parameters:
        --------------------
        angular: int, desired number of angular sections in the desired disk. If 0, the angular dimension stays unchanged

        radial: int, desired number of radial sections in the desired disk. If 0, the radial dimension stays unchanged

        log: bool, if true, the new disk will have logarithmic grid in the radial direction

         ~ ~ ~ - ,                    ~ ~ ~ - ,
        |     -    ' ,               |     -    ' ,
        |    /     _. ,              |    /        ,
        |   -   _.      ,     ->     |   - averaged ,
        |  / _-    _  - ,            |  /       _  - ,
        | -- _  -       ,            | -  _  -       ,
        |/-____________,             |/-____________,

        ~ ~ ~ - ,                    ~ ~ ~ - ,
        ------       ' ,             ------       ' ,
              --__     ,                  --__     ,
        ----__    \     ,    ->             \     ,
        ----   \    \    ,           ----averaged\    ,
        --  \   \    \   ,               \        \   ,
          |  |   |   |   ,                |        |   ,

        """
        print(f"Relaxing disk...")

        if radial not in [2**i for i in range(13)]:
            raise ValueError(f"Relax error: only powers of 2 up to 1024 are allowed for radial relax! {radial} not allowed")
        if angular not in [2**i for i in range(10)] and angular//3 not in [2**i for i in range(10)]:
            raise ValueError(f"Relax error: only powers of 2 up to 512 and their multiples of 3 are allowed for radial relax! {angular} not allowed")

        if len(self.dust_density) == 0:
            newsigma = self.gas_density[:]
        else:
            newsigma = self.dust_density[:]
        newT = self.T[:]
        Nrad = self.par["Nrad"]
        Nsec = self.par["Nsec"]

        from skimage.transform import downscale_local_mean as dlm

        newsigma = dlm(newsigma, (Nrad//radial, Nsec//angular), cval = np.NaN)
        newT = dlm(newT, (Nrad//radial, Nsec//angular), cval = np.NaN)

        self.phi_inf = self.phi_inf[:-(Nsec//angular-1):Nsec//angular]
        self.phi_med = dlm(self.phi_med, Nsec//angular, cval = np.NaN)
        self.phi_sup = self.phi_sup[Nsec//angular-1::Nsec//angular]

        self.r_inf = self.r_inf[:-(Nrad//radial-1):Nrad//radial]
        self.r_med = dlm(self.r_med, Nrad//radial, cval = np.NaN)
        self.r_sup = self.r_sup[Nrad//radial-1::Nrad//radial]


        if log:
            new_rs = np.logspace(np.log10(self.Rmin), np.log10(self.Rmax), num = 2*radial+1)

            new_r_inf = new_rs[0:-1:2]
            new_r_med = new_rs[1::2]
            new_r_sup = new_rs[2::2]

            newsigma = np.array([np.interp(new_r_med, self.r_med, newsigma[:,i]) for i in range(len(self.phi_med))]).T
            newT = np.array([np.interp(new_r_med, self.r_med, newT[:,i]) for i in range(len(self.phi_med))]).T

            self.r_inf = new_r_inf
            self.r_med = new_r_med
            self.r_sup = new_r_sup

        if len(self.dust_density) == 0:
            self.gas_density = newsigma
        else:
            self.dust_density = newsigma
```

```
        # If you're reading this: Wow. I really didn't expect anyone to study my thesis THAT hard. Email me if you have any questions
        self.T = newT

        self.par["Nrad"] = Nrad = radial
        self.par["Nsec"] = Nsec = angular

        phi_distances = np.outer(self.r_med, 2*np.sin(self.phi_sup-self.phi_med))

        r_minmax = [np.min(self.r_sup - self.r_inf), np.max(self.r_sup - self.r_inf)]
        phi_minmax = [np.min(phi_distances), np.max(phi_distances)]

        print(f"Flat disk relaxed to "+ f"a log grid "*log + f"with a total of {Nrad} radial and {Nsec} angular sectors. Cells are now between \n{r_minmax[0]:.1e} and {r_minmax[1]:.1e} au wide in radial and between \n{phi_minmax[0]:.1e} and {phi_minmax[1]:.1e} au in angular directions")

    def inner_outer_finish(self, outer_steps = 512, inner_steps = 512, r_newmax = 40, r_newmin = None, smoothing = 5, r1 = 0.1, r2 = 3, r3 = 0.1):
        """
        Finishes inner and outer parts of the flat disk using funcions
        sigma(r) = g(r) = exp(-(r1/r)**2.0) and
        T(r) = u(r) = 500
        for the inner part,
        sigma(r) = h(r) = exp(-((r-self.Rmax)/r2)**2.0)
        T(r) = v(r) = exp(-((r-self.Rmax)/r2)**2.0)
        for the outer part. Blends original disk with these functions using the power_smooth_min function.
        --------------------
        Parameters:
        --------------------
            outer_steps: int, number of new cells outside of the outer rim of the disk. If set to 0 no new cells will be added

            inner_steps: int, number of new cells inside of the inner rim of the disk. If set to 0 no new cells will be added

            r_newmax: float [au], new outer radius of the disk

            r_newmin: float [au], new inner radius of the disk. Must not be smaller than the radius of the star. By default set to be the radius of the Sun.

            smoothing: float, parameter for power_smooth_min function. Determines how quickly the disk transitions to the finishing functions mentioned above

            r1, r2, r3: float, parameters for finishing functions, see above
        """
        if (len(self.r_med) + inner_steps + outer_steps) not in [2**i for i in range(13)]:
            print("WARNING: The new disc doesn't have a power of 2 of radial sectors. You won't be able to relax it!")

        if r_newmin is None: r_newmin = 1.1*self.par["StellarRadius"]*sol_rad/au

        def f(r):   #funciton mimicing the disk's sigma
            return r**(-.5)
        def g(r):   #function taking care of sigma's fading as it gets to the center
            return np.exp(-(r1/r)**2.0)
        def h(r):   #function taking care of sigma's fading as it gets towards the outer rim
            return np.exp(-((r-0.85*self.Rmax)/r2)**2.0) #inspired by https://iopscience.iop.org/article/10.1088/0004-637X/741/1/3

        def t(r):   #function mimicing the disk's T
            return r**(-1.5)
        def u(r):   #function taking care of temperature's profile as it gets to the center
            return np.sqrt(self.par["StellarRadius"]*sol_rad/au/r)*self.par["EffectiveTemperature"]
        def v(r):   #function taking care of temperature's fading as it gets towards the outer rim
            return np.exp(-((r-0.85*self.Rmax)/r2)**2.0)

        #Here starts the finishing of the inner part of the disk
        if inner_steps != 0:
            dr = (self.Rmin-r_newmin)/inner_steps
            self.r_inf = np.insert(self.r_inf, 0, np.linspace(r_newmin,self.Rmin+dr, inner_steps+1)[:-1])
            r_med_new =  np.linspace(r_newmin+dr/2, self.Rmin-dr/2, inner_steps)
            self.r_med = np.insert(self.r_med, 0, r_med_new)
            self.r_sup = np.insert(self.r_sup, 0, np.linspace(r_newmin+dr,self.Rmin, inner_steps+1)[1:])
            self.par["Nrad"] += inner_steps

            if len(self.dust_density) == 0:
                inner_Csigma = self.gas_density[0,:]/f(self.Rmin)
            else:
                inner_Csigma = self.dust_density[0,:]/f(self.Rmin)

            inner_CT =     self.T[0,:]/t(self.Rmin)

            inner_sigma_ring = np.outer(f(r_med_new), np.transpose(inner_Csigma))
            inner_T_ring =     np.outer(t(r_med_new), np.transpose(inner_CT)    )

            inner_sigma_fading = np.outer(g(r_med_new), 1.3*np.mean(np.transpose(inner_Csigma)))
            inner_T_fading =     np.outer(u(r_med_new), np.ones((1,self.par["Nsec"]))    )

            new_inner_sigma = smooth_min(inner_sigma_ring, inner_sigma_fading, smoothing = smoothing)
            new_inner_T =     smooth_min(inner_T_fading, 1.5*inner_T_ring,         smoothing = 2.45)

            if len(self.dust_density) == 0:
                self.gas_density = np.insert(self.gas_density, [0], new_inner_sigma, axis = 0)
            else:
                self.dust_density = np.insert(self.dust_density, [0], new_inner_sigma, axis = 0)

            self.T =     np.insert(self.T, [0], new_inner_T, axis = 0)

            self.par["Rmin"] = r_newmin
            self.Rmin = r_newmin
```

```python
        # Here starts the finishing of the outer part of the disk
        if outer_steps != 0:
            dr = (r_newmax-self.Rmax)/outer_steps
            r_med_new = np.linspace(self.Rmax+dr/2, r_newmax-dr/2, outer_steps)
            self.r_inf = np.append(self.r_inf, np.linspace(self.Rmax, r_newmax+dr, outer_steps+1)[1:] )
            self.r_med = np.append(self.r_med, r_med_new)
            self.r_sup = np.append(self.r_sup, np.linspace(self.Rmax+dr, r_newmax, outer_steps+1)[1:] )
            self.par["Nrad"] += outer_steps

            if len(self.dust_density) == 0:
                outer_Csigma = self.gas_density[-1,:]/f(self.Rmax)
            else:
                outer_Csigma = self.dust_density[-1,:]/f(self.Rmax)

            outer_CT =     self.T[-1,:]/t(self.Rmax)

            outer_sigma_ring = np.outer(f(r_med_new), np.transpose(outer_Csigma))
            outer_T_ring =     np.outer(t(r_med_new), np.transpose(outer_CT)    )

            outer_sigma_fading = np.outer(h(r_med_new), np.mean(outer_Csigma) )
            outer_T_fading =     np.outer(v(r_med_new), np.mean(outer_CT)*np.ones((1,self.par["Nsec"])))

            new_outer_sigma = smooth_min(outer_sigma_ring, outer_sigma_fading, smoothing = smoothing)
            new_outer_T     = smooth_min(outer_T_ring,     outer_T_fading,     smoothing = smoothing)

            if len(self.dust_density) == 0:
                self.gas_density = np.append(self.gas_density, new_outer_sigma, axis = 0)
            else:
                self.dust_density = np.append(self.dust_density, new_outer_sigma, axis = 0)

            self.T =     np.append(self.T,     new_outer_T,     axis = 0)

            self.Rmax = r_newmax
            self.par["Rmax"] = r_newmax

        print(f"The disk was finished with {inner_steps} new inner and {outer_steps} new outer radial sectors")

    def gas_to_dust(self):
        """
        Modifies the disk's density to go from gas to dust, which dominates optical properities (later also according to its temperature to better reflect dust density with its typical evaporations and stuffs)
        Brinstiel et. al. 2012
        """

        simple_modifier = .01
        self.dust_density = self.gas_density*simple_modifier

    ## ---------------------------------------------------
    ##radmc3D procedures
    ##ALL RADMC3D FILES NEED TO BE IN CGS, conversion happens at the end of each procedure just before writing
    ## ---------------------------------------------------

    def set_theta_boundary(self, wanted_depth = .001, extra_margin = 0, thickness = f_z):
        """
        Sets the theta boundary such that only the disk with at least sufficeint "wanted_depth" is included. Uses peak of stars spectrum
        Should be run BEFORE write density (obviously)
        --------------------
            Parameters:
        --------------------
        wanted_depth: double, smallest wanted depth such that this part of the disk is included

        extra_margin [deg]: margin on top of the wanted crop
        """

        # Find the wav at which to calculate tau. A good guess might be to look at mean temp of the flat disk

        extra_margin = np.deg2rad(extra_margin)

        b = 2897.771955 # microns per Kelvin via CODATA 2018 Wien's displacement law
        wav = b/np.mean(self.T) # in microns

        # I only look at the points with the highest scale theta atan(H/r); expecting those to have the largest tau

        tiled_rs = au*np.tile(self.r_med, ( len(self.T.T) ,1)).T
        scale_heights = thickness(T=self.T, equatorial_r = tiled_rs, getH=True)
        scale_thets = np.arctan(scale_heights/tiled_rs)
        suspect = np.unravel_index(scale_thets.argmax(), scale_thets.shape)
        suspect_r = self.r_med[suspect[0]]*au
        suspect_phi = self.phi_med[suspect[1]]

        # Now I find the theta at which tau gets small enogh with sime quick interval halving

        highest_theta = 0
        max_iterations = 10000
        max_z = 10*au   #is this infinite enough?
        min_z = 0*au
        tolerance = 1e-3    #is this too tolerant?
        last_guess = 1e10
        initial_guess = 0.1*au
        for i in range(max_iterations):
            last_guess = initial_guess
            if self.optical_depth(how_close_to_equat=initial_guess, phi = suspect_phi, equatorial_r = suspect_r, wav = wav, message = False, thickness = thickness) > wanted_depth:
                min_z = initial_guess
                initial_guess = (min_z+max_z)*0.5
            else:
                max_z = initial_guess
```

```
                initial_guess = (min_z+max_z)*0.5
            if abs(last_guess-initial_guess)<tolerance:
                break
        found_z=initial_guess
        found_theta = 0.5*np.pi-np.arctan(found_z/suspect_r)
        if found_theta > highest_theta: highest_theta = found_theta

        highest_theta -= extra_margin

        self.par["Thetamin"] = highest_theta

        print(f"Lower theta boundary cropped to {np.rad2deg(highest_theta):.2f} deg")

    def radmc_write_dust_density(self, binary = True, thickness = f_z, nthet = 100, buffer_size = 4096*32*40, note = "
    ):
        """
        Write dust_density.(b)inp file for radmc3D by thickening flat disk. The procedure will print progress
    percentage every 5 minutes starting after the first minute.
        --------------------
        Parameters:
        --------------------
        binary: bool, if set True, binary file with .binp file extension. Otherwise creates a text file with .inp file
     extension

        thickness: function, by which the volumetric density around the disk is determined. It is expected to take
    parameters of the point above which the density is needed:
            h [m]: float, height of the point of interest above the disk
            equatorial_r [m]: float, distance from the star to the point in equatorial plane under the point of
    interest
            T [K]: float, temperature of the dust at the point in equatorial plane under the point of interest
            sigma [kg/m^2]: float, surface density of the dust at the point in equatorial plane under the point of
    interest
            M_star [kg]: float, mass of the star at the center

        nthet: int, number of cells in theta direction to be made above the disk. The same amount is generated beneath
     the disk as well, mirrored by radmc itself

        buffer_size: int, max number of floats to be stored in RAM at any given point

        note: string, added as a second line of fig title
        """

        if len(self.dust_density) == 0:
            raise ValueError("Write dust error: list of densities is empty!")
        start_time = time.time()


        if self.dust_density.ndim == 3:  # just writing down 3D grid
            print(f"Density notice: density is in 3D, writing a dust_density."+binary*f"b"+f"inp for radmc3d, a total
    of {len(self.r_med)}*{len(self.theta_med)}*{len(self.phi_med)} = {np.size(self.dust_density)} cells...")
            nrspecs = 1  # number of dust species
            nrcells = np.size(self.dust_density)

            if binary:
                outfile = open("dust_density"+note+".binp","wb")
                np.array([1,8,nrcells,nrspecs]).tofile(outfile)
                np.transpose(self.dust_density*cm*cm*cm/gram, (2,1,0)).tofile(outfile)
            else:
                outfile = open("dust_density"+note+".inp","w")
                np.array([1,nrcells,nrspecs]).tofile(outfile, sep = "\n")
                outfile.write("\n")
                np.transpose(self.dust_density*cm*cm*cm/gram, (2,1,0)).tofile(outfile, sep = "\n", format="%16e") #
    transposition is required, cuz radmc want its files in such an order: [0,0,0], ... [n_rad, 0, 0], [0,1,0], ...  [
    n_rad, 1, 0]. Numpy tofile goes the other way around, first 'depleting' the last index: [0,0,0], ... [0, 0, n_phi
    ], [0, 1 ,0], ...  [0, 1, n_phi]... I suppose anyway
            outfile.close()
            return 0

        if self.dust_density.ndim == 4:  # case multiple grainsizes or species are present, write down 3D grid
            print(f"Density notice: density is in 3D, writing a dust_density."+binary*f"b"+f"inp for radmc3d, a total
    of {len(self.r_med)}*{nthet}*{len(self.phi_med)}*{len(self.dust_density[:,0,0,0])} = {np.size(self.dust_density)}
     cells...")
            nrspecs = len(self.dust_density[:,0,0,0])  # number of dust species
            nrcells = np.size(self.dust_density[0,:,:,:])

            if binary:
                outfile = open("dust_density"+note+".binp","wb")
                np.array([1,8,nrcells,nrspecs]).tofile(outfile)
                np.transpose(self.dust_density*cm*cm*cm/gram, (0,3,2,1)).tofile(outfile)
            else:
                outfile = open("dust_density"+note+".inp","w")
                np.array([1,nrcells,nrspecs]).tofile(outfile, sep = "\n")
                outfile.write("\n")
                np.transpose(self.dust_density*cm*cm*cm/gram, (0,3,2,1)).tofile(outfile, sep = "\n", format="%16e") #
    transposition is required, cuz radmc want its files in such an order: [0,0,0], ... [n_rad, 0, 0], [0,1,0], ...  [
    n_rad, 1, 0]. Numpy tofile goes the other way around, first 'depleting' the last index: [0,0,0], ... [0, 0, n_phi
    ], [0, 1 ,0], ...  [0, 1, n_phi]... I suppose anyway
            outfile.close()
            return 0

        # Creating 3D grid out of 2D disk seed
        print(f"Density notice: writing a dust_density."+binary*f"b"+f"inp for radmc3d, a total of {len(self.r_med)}*{
    nthet}*{len(self.phi_med)} = {np.size(self.dust_density)*nthet} cells...")
        nrspecs = 1  # number of dust species
        self.dust_species.append("silicate")
        nrcells = np.size(self.dust_density)*nthet

        # Update to theta boundaries is necessary
        self.par["Nthet"] = nthet
```

```
        self.par["Thetamax"] = np.pi/2  #makes the eq plane a plane of symmetry
        thetas = (np.logspace(0, 1, num = nthet+1)-1)/9*(self.par["Thetamax"] - self.par["Thetamin"]) + self.par["Thetamin"]

        self.theta_inf = thetas[:-1]
        self.theta_sup = thetas[1:]
        self.theta_med = 0.5*(self.theta_inf+self.theta_sup)

        total_mass = 0
        buffer_size = int(buffer_size/40)    # there are total of 40 arrays of size "buffer_size" used at once later, so the number has to be adjusted to correctly represent the amount of mermory the arrays take up
        if binary:
            outfile = open("dust_density.binp"+note,"wb")
            np.array([1,8,nrcells,nrspecs]).tofile(outfile) # Necessary parameters for radmc3d: 1 (always present), 8(# of bytes),...
        if not binary:
            outfile = open("dust_density.inp" + note,"w")
            np.array([1,nrcells,nrspecs]).tofile(outfile, sep = "\n") # Necessary parameters for radmc3d: 1 (always present)...
            outfile.write("\n")

        timer = time.time()+60
        for phi_index in range(len(self.phi_med)):
            for theta_index, theta in enumerate(self.theta_med):
                cont = False
                first_index = 0
                last_index = buffer_size
                while not cont:
                    if last_index > len(self.r_med)-1:                                                  #.*z3
                        last_index = len(self.r_med)                                                    #           .*
                        cont = True                                                                     #       .*
                    rs = self.r_med[first_index:last_index]                                             #   .*
                    equatorial_rs = np.sin(theta)*rs                                                    #z4*       o<-z       . <- elementary volume from the side
                    equatorial_r1s = np.sin(self.theta_sup[theta_index])*rs                             #  *
                    equatorial_r2s = np.sin(self.theta_sup[theta_index])*rs                             #    *
                    equatorial_r3s = np.sin(self.theta_inf[theta_index])*rs                             #     .
                    equatorial_r4s = np.sin(self.theta_inf[theta_index])*rs                             #       z1
                    zs = np.cos(theta)*rs                                                               #Average density from points z through z4 is saved
                    z1s = self.r_inf[first_index:last_index]*np.cos(self.theta_sup[theta_index])
                    z2s = self.r_sup[first_index:last_index]*np.cos(self.theta_sup[theta_index])
                    z3s = self.r_sup[first_index:last_index]*np.cos(self.theta_inf[theta_index])
                    z4s = self.r_inf[first_index:last_index]*np.cos(self.theta_inf[theta_index])
                    interpolated_Ts =  np.interp(equatorial_rs,  self.r_med, self.T[:,phi_index])
                    interpolated_T1s = np.interp(equatorial_r1s, self.r_med, self.T[:,phi_index])
                    interpolated_T2s = np.interp(equatorial_r2s, self.r_med, self.T[:,phi_index])
                    interpolated_T3s = np.interp(equatorial_r3s, self.r_med, self.T[:,phi_index])
                    interpolated_T4s = np.interp(equatorial_r4s, self.r_med, self.T[:,phi_index])
                    interpolated_sigmas =  np.interp(equatorial_rs,  self.r_med, self.dust_density[:,phi_index])
                    interpolated_sigma1s = np.interp(equatorial_r1s, self.r_med, self.dust_density[:,phi_index])
                    interpolated_sigma2s = np.interp(equatorial_r2s, self.r_med, self.dust_density[:,phi_index])
                    interpolated_sigma3s = np.interp(equatorial_r3s, self.r_med, self.dust_density[:,phi_index])
                    interpolated_sigma4s = np.interp(equatorial_r4s, self.r_med, self.dust_density[:,phi_index])
                    rhos  = thickness(h =  zs*au, sigma = interpolated_sigmas,  T = interpolated_Ts,  equatorial_r=equatorial_rs*au,  M_star= self.par["M_star"][0])
                    rho1s = thickness(h = z1s*au, sigma = interpolated_sigma1s, T = interpolated_T1s, equatorial_r=equatorial_r1s*au, M_star= self.par["M_star"][0])
                    rho2s = thickness(h = z2s*au, sigma = interpolated_sigma2s, T = interpolated_T2s, equatorial_r=equatorial_r2s*au, M_star= self.par["M_star"][0])
                    if theta_index != 0:
                        rho3s = thickness(h = z3s*au, sigma = interpolated_sigma3s, T = interpolated_T3s, equatorial_r=equatorial_r3s*au, M_star= self.par["M_star"][0])
                        rho4s = thickness(h = z4s*au, sigma = interpolated_sigma4s, T = interpolated_T4s, equatorial_r=equatorial_r4s*au, M_star= self.par["M_star"][0])
                        average_rhos = (rhos+rho1s+rho2s+rho3s+rho4s)/5
                    else: average_rhos = (rhos+rho1s+rho2s)/3
                    elementary_volumes = (self.phi_sup[phi_index]-self.phi_inf[phi_index])*(self.r_sup[first_index:last_index]**3-self.r_inf[first_index:last_index]**3)/3*(np.cos(self.theta_inf[theta_index])-np.cos(self.theta_sup[theta_index]))*au*au*au
                    total_mass += np.sum(elementary_volumes*average_rhos)
                    if binary:
                        (average_rhos*cm*cm*cm/gram).tofile(outfile)
                    if not binary:
                        (average_rhos*cm*cm*cm/gram).tofile(outfile, sep = "\n", format="%.16e")
                        outfile.write("\n")
                    if cont: break
                    first_index, last_index = last_index, last_index + buffer_size

                if time.time()>timer:
                    print(f"{phi_index/(len(self.phi_med)-1)*100:.2f} % done")
                    timer += 300


        outfile.close()
        flat_mass = self.integrate_mass()
        print(f"Density notice: dust_density."+binary*f"b"+f"inp file was created using '{thickness.__name__}' function to make the disk thick. The total mass of the disk is now {2*total_mass/M_sun:.4e} M_sun (including the part below eq. plane) and it took {(time.time()-start_time)/60:.2f} min")
        if total_mass/flat_mass > 4: print(f"Your new disk is {total_mass/flat_mass:.1f} more massive than its flat version. This could be caused by too rough of a discretisation in the theta direction. Consider using a higher '
```

```
steps' parameter when calling radmc_write_dust_density.")

        phi_distances = np.multiply.outer(self.r_med, 2*np.sin(self.phi_sup-self.phi_med))
        phi_distances = np.multiply.outer(phi_distances, np.sin(self.theta_med))
        theta_distances = np.multiply.outer(self.r_med, 2*np.sin(self.theta_med-self.theta_inf))
        r_distances = self.r_sup-self.r_inf

        r_minmax = [np.min(r_distances), np.max(r_distances)]
        phi_minmax = [np.min(phi_distances), np.max(phi_distances)]
        theta_minmax = [np.min(theta_distances), np.max(theta_distances)]

        print(f"The mesh now has side lengths between\n{r_minmax[0]:.1e} au and {r_minmax[1]:.1e} au in the r direction\n{theta_minmax[0]:.1e} au and {theta_minmax[1]:.1e} au in the theta direction\n{phi_minmax[0]:.1e} au and {phi_minmax[1]:.1e} au in the phi direction")

    def radmc_write_grid(self, style = 0): #, grid_type = 100, style = 0, x1min = 2.8, x1max=14, x2min = 0, x2max = np.pi, x3min = 0, x3max = 2*np.pi):
        """
        Creates amr_grid.inp file for the radmc3d program.
        --------------------
        Parameters:
        --------------------
        style: int, 0, 1 or 2, determining the style of the grid: 0 for regular, 1 for layered, 2 for oct-tree. SO FAR ONLY REGULAR GRID SUPPORTED
        """
        grid_type=100                   # makes the grid spherical as opposed to cartesian (grid type 0)
        if style != 0: raise ValueError("Grid error: Non-uniform grid style not yet supported!")

        nx = [len(self.r_med),len(self.theta_med), len(self.phi_med)] # number of cells in each direction (x, y, z, or r, theta, phi, or r, phi, z) gets imported
        if 0 in nx:
            raise ValueError("Grid error: wrong dimension of density array. Perhaps it wasn't thickened yet?")

        on_off = [0 if n == 1 else 1 for n in nx]   # looks at which dimension is "turned off", i.e. which has only 1 cell in it. Radmc3d cares

        with open("amr_grid.inp", "w") as outfile:
            outfile.write(f"1\n{style}\n{grid_type}\n0\n{on_off[0]} {on_off[1]} {on_off[2]}\n{nx[0]} {nx[1]} {nx[2]}\n")

            outfile.write(f"{self.r_inf[0]*au/cm:.16e}\n")
            for rboundary in self.r_sup:
                outfile.write(f"{rboundary*au/cm:.16e}\n")

            outfile.write(f"{self.theta_inf[0]:.16e}\n")
            for thetaboundary in self.theta_sup:
                outfile.write(f"{thetaboundary:.16e}\n")

            outfile.write(f"{self.phi_inf[0]:.16e}\n")
            for phiboundary in self.phi_sup:
                outfile.write(f"{phiboundary:.16e}\n")

        print("Grid notice: amr_grid.inp file was created.")

    def radmc_stars(self, coordinates=[0,0,0], radii=None, fluxes=None):
        """Creates stars.inp file for radmc3d program
        --------------------
        Parameters:
        --------------------
        coordinates [cm]: array-like, positions of stars, expected in [x_1,y_1,z_1,x_2,y_2,z_2,...] format, that is cartesian coordinates

        masses [g]: array-like, masses of stars, expected in [m_1, m_2, ...] format

        radii [cm]: array-like, masses of stars, expected in [r_1, r_2, ...] format

        fluxes [erg*cm^-2*s^-1*Hz^-1] OR [-K]: array-like, fluxes of stars (as seen from 1 parasec = 3.08572e18 cm) in wavelengths lambda, expected in [F_1(lambda_1), F_1(lambda_2), ..., F_1(lambda_n), F_2(lambda_1), ...] format. If a blackbody radiation is to be assumed, then a negative value of star's surface temperature is to be entered. For instance if star 2 has surface temperature of 5000 K, then fluxes will look like [F_1(lambda_1), F_1(lambda_2), ... ,F_1(lambda_n), -5000, F_3(lambda_1), ...]
        """

        # https://www.aanda.org/articles/aa/full_html/2017/11/aa29786-16/aa29786-16.html r star
        if radii is None: radii = [self.par["StellarRadius"]*sol_rad/cm]
        masses = self.par["M_star"]/gram

        if fluxes is None: fluxes = [-self.par["EffectiveTemperature"]]

        if len(coordinates)%3 != 0: raise ValueError("Stars error: Number of coordinates isn't a multiple of 3!")
        if len(coordinates) != 3*len(masses): raise ValueError("Stars error: List of star masses isn't the same length as that of their coordinates!")
        if len(coordinates) != 3*len(radii): raise ValueError("Stars error: List of star radii isn't the same length as that of their coordinates!")
        if len(masses) != len(radii): raise ValueError("Stars error: List of star radii isn't the same length as that of their masses!")

        nlambda = 0
        try:
            with open("wavelength_micron.inp","r") as wavelength:
                nlambda = wavelength.readline()
        except FileNotFoundError:
            raise Exception("Stars error: wavelength_micron.inp file not found. It is essential for generating stars.inp!")

        with open("stars.inp", "w") as outfile:
            outfile.write(f"2\n{len(coordinates)//3} {nlambda}")
            for i in range(len(masses)):
```

```
                outfile.write(f"{radii[i]:.16e} {masses[i]:.16e} {coordinates[i]:.16e} {coordinates[i+1]:.16e} {coordinates[i+2]:.16e}\n")
            outfile.write("\n")
            done_with_lambda = False
            with open("wavelength_micron.inp","r") as wavelength:
                for line in wavelength:
                    if done_with_lambda:
                        if is_number(line): outfile.write(f"{float(line):.16e}\n")
                    else: done_with_lambda = True
            outfile.write("\n")
            for value in fluxes:
                outfile.write(f"{value:.16e}\n")

        print("Stars notice: stars.inp file was created.")

    def radmc_wavelength(self,array=None, smallest = None, largest = None, step = None):
        """Creates wavelength_micron.inp file for radmc3d program"""
        # Supports input array of wavelengths, uniform distribution or, if no arguments are put in, will use predetermined wavelengths
        # Wavelength is in microns

        if ((smallest is None) ^ (step is None)) or ((step is None) ^ (largest is None)):
            raise ValueError("Wavelength error: Smallest, largest AND step wavelength have to be passed!")
        if (array != None and smallest != None):
            raise ValueError("Wavelength error: Choose either the array method or uniform method, not both!")

        with open("wavelength_micron.inp","w") as outfile:
            if (array != None):
                outfile.write(f"{len(array)}")
                for i in range (0, len(array)):
                    outfile.write(f"\n{array[i]:.16e}")
            if (step != None):
                outfile.write(np.floor((largest-smallest)/step))
                lambd = smallest
                while lambd < largest:
                    outfile.write(f"\n{lambd:.16e}")
                    lambd+=step
            else:
                predetirmed = """                    99
                    1.000000000e-01
                    1.250576960e-01
                    1.563942733e-01
                    1.955830748e-01
                    2.445916871e-01
                    3.058807285e-01
                    3.825273915e-01
                    4.783799423e-01
                    5.9825093316e-01
                    7.481588342e-01
                    9.356302004e-01
                    1.170077572e+00
                    1.463272052e+00
                    1.829934315e+00
                    2.288473692e+00
                    2.861912473e+00
                    3.579041800e+00
                    4.475867213e+00
                    5.597416412e+00
                    7.000000000e+00
                    7.1805031816e+00
                    7.365660863e+00
                    7.555593045e+00
                    7.750422850e+00
                    7.9502765616e+00
                    8.155283751e+00
                    8.365577282e+00
                    8.581293478e+00
                    8.8025721616e+00
                    9.0295567816e+00
                    9.262394474e+00
                    9.501236151e+00
                    9.7462366316e+00
                    9.997554752e+00
                    1.025535340e+01
                    1.051979968e+01
                    1.079106502e+01
                    1.106932526e+01
                    1.135476076e+01
                    1.164755654e+01
                    1.194790242e+01
                    1.225599306e+01
                    1.257202817e+01
                    1.289621263e+01
                    1.322875656e+01
                    1.356987552e+01
                    1.391979063e+01
                    1.427872872e+01
                    1.464692244e+01
                    1.502461047e+01
                    1.541203763e+01
                    1.580945504e+01
                    1.621712034e+01
                    1.663529775e+01
                    1.706425837e+01
                    1.750428023e+01
                    1.795564857e+01
                    1.841865598e+01
                    1.889360257e+01
                    1.938079621e+01
```

```
                    1.988055271e+01
                    2.039319602e+01
                    2.091905843e+01
                    2.145848083e+01
                    2.201181286e+01
                    2.257941320e+01
                    2.316164978e+01
                    2.375890002e+01
                    2.437155105e+01
                    2.500000000e+01
                    3.073733534e+01
                    3.779135136e+01
                    4.6464217516e+01
                    5.712744950e+01
                    7.023782290e+01
                    8.635694065e+01
                    1.061752898e+02
                    1.305418194e+02
                    1.605003072e+02
                    1.973340706e+02
                    2.426209401e+02
                    2.9830084716e+02
                    3.667589278e+02
                    4.509276862e+02
                    5.544126202e+02
                    6.816466650e+02
                    8.380800851e+02
                    1.030413945e+03
                    1.266887158e+03
                    1.557629417e+03
                    1.9150951016e+03
                    2.354596824e+03
                    2.894961287e+03
                    3.559335835e+03
                    4.376179966e+03
                    5.380484445e+03
                    6.615270188e+03
                    8.133431126e+03
                    1.000000000e+04
                    """
                print(predetirmed.replace("                    ", ""), file = outfile)

    def radmc_write_opacity(self):
        """
        Creates dustopac.inp file that tracks which opacity files should be used for which dust species in the radiative transfer
        """

        nspec = len(self.dust_species)  # number of dust species, here only silicate
        opacity_header = [2, nspec]
        dashes = "--------------------------------"
        spiecies_line = [1, 0, "name"]  # 1/10 for dustkappa/dustkapscatmat; 0/1 for thermal/quantu heating (quantum apparently doesn't even work yet); name used in dustkappa filename
        with open("dustopac.inp", "w") as outfile:
            print(f"{opacity_header[0]}\n{opacity_header[1]}\n{dashes}", file = outfile)
            for spiec in self.dust_species:
                spiecies_line[2] = spiec    # <-- HERE! I declare the species name
                for thing in spiecies_line:
                    print(thing, file = outfile)
                print(dashes, file = outfile)

    def radmc_write_inputs(self, mrw = None, thickness = f_z, nthet = 100, binary = True, buffer_size = 4096*32, gas_to_dust = True, wanted_depth = 0.001, extra_margin = 0, **kwargs):
        """
        Creates all the necessary files for running radmc3d script
        --------------------
        Parameters:
        --------------------
        mrw: bool, determines whether modified random walk should be used. If unspecified, will calculate the optical thickness of the disk and sets the parameter acoordingly

        thickness: function, by which the volumetric density around the disk is determined. It is expected to take parameters of the point above which the density is needed:
            h [m]: float, height of the point of interest above the disk
            equatorial_r [m]: float, distance from the star to the point in equatorial plane under the point of interest
            T [K]: float, temperature of the dust at the point in equatorial plane under the point of interest
            sigma [kg/m^2]: float, surface density of the dust at the point in equatorial plane under the point of interest
            M_star [kg]: float, mass of the star at the center

        nthet: float, number of cells in theta direction to be made above the disk. The same amount is generated beneath the disk as well

        buffer_size: int, max number of floats to be stored in RAM at any given point

        gas_to_dust: bool, if True, will assume only gas density was provided and will perform gas_to_dust function

        wanted_depth [m^-1]: float, the optical depth at which "the disk ends"; used for setting boundary of the theta direction

        extra_margin [deg]: float, extra margin to be put on top of the new theta boundary

        **kwargs: see radmc3d manual, chapter MAIN INPUT AND OUTPUT FILES OF RADMC-3D, section INPUT: radmc3d.inp. There's like a brazillion of inputs, you can't demand from me to write them all here
        """

        self.radmc_write_opacity()
        if gas_to_dust: self.gas_to_dust()
```

```
        if mrw is None:
            depth = self.optical_depth(from_star=True, wav = 10, message = False, thickness=thickness)
            if depth > 1e7:
                print(f"Optical depth is quite high ({depth:.3e}), setting modified_random_walk to true in radmc3d.inp file...")
                mrw = 1
            else: mrw = 0

        radmc_inp_params = {
            "istar_sphere": 1,  #whether to take star as a sphere (1) or a point (0)
            "itempdecoup": 1,   #all dust species are thermally independet
            "lines_mode": -1,   #default, needs extra files to work properly
            "modified_random_walk": int(mrw),   #if 1 and if photon is stuck for too long in a cell, the probability of where it will exit is calculated and the photon is transported there
            "nphot": 1e9,    #number of photons for thermal siulation. Careful, radmc needs this to be int, not float! I convert it below
            "nphot_scat": 3000000000,   #number of photons for image-making
            "nphot_spec": 100000,   #number of photons for spectra-making
            "rto_style": 3,     #1 for output files to be ascii, 3 for binary, 2 for pain (old and as of now obsolete fortran format)
            "scattering_mode_max": 1,   #if 1, radmc3d ignore anisotropic scattering (it's not like I provided it in the opacity files anyway)
            "tgas_eq_tdust": 1, #if 1, gas temperature will be the same as the first dust species' temp
        }

        for kw in kwargs:
            if kw == "nphot": kwargs[kw] = int(kwargs[kw])
            radmc_inp_params[kw] = kwargs[kw]

        with open("radmc3d.inp", "w") as radmcinp:
            for name, value in radmc_inp_params.items():
                print(f"{name} = {value}", file = radmcinp)
        self.set_theta_boundary(wanted_depth=wanted_depth, extra_margin=extra_margin)
        self.radmc_write_dust_density(binary = binary, thickness = thickness, nthet=nthet, buffer_size=buffer_size)
        self.radmc_write_grid()
        self.radmc_wavelength()
        self.radmc_stars()

    def radmc_write_temperature(self, binary=True, note = "", nthet = 100, buffer_size = 4096*32*40, thickness = identity_T):
        """
        Creates dust_temperature.inp from fargo temperature data for image making. Works just like write_dust_density()
        """
        if self.T.ndim == 3:
            nrspecs = 1
            nrcells = np.size(self.T)
            if binary:
                file = open(note+"dust_temperature.bdat", "wb")
                header = np.array([1, 8, nrcells, nrspecs])
                header.tofile(file)
                np.transpose(self.T, (2,1,0)).tofile(file)
            else:
                file = open(note+"dust_temperature.dat", "w")
                header = np.array([1, nrcells, nrspecs])
                header.tofile(file, sep = "\n")
                file.write("\n")
                np.transpose(self.T, (2,1,0)).tofile(file, sep = "\n")
            a = np.fromfile("dust_temperature.bdat", dtype = np.double, offset = 4*8)

        elif self.T.ndim == 2:
            #Creating 3D grid out of 2D disk seed
            print(f"Density notice: writing a dust_temperature."+binary*f"b"+f"inp for radmc3d, a total of {len(self.r_med)}*{nthet}*{len(self.phi_med)} = {np.size(self.T)*nthet} cells...")
            nrspecs = 1  # number of dust species
            nrcells = np.size(self.T)*nthet

            #Update to theta boundaries is necessary
            self.par["Nthet"] = nthet
            self.par["Thetamax"] = np.pi/2  #makes the eq plane a plane of symmetry
            thetas = np.power(np.linspace(self.par["Thetamin"]**3, self.par["Thetamax"]**3, num = nthet+1),1/3)

            self.theta_inf = thetas[:-1]
            self.theta_sup = thetas[1:]
            self.theta_med = 0.5*(self.theta_inf+self.theta_sup)

            total_mass = 0

            buffer_size = int(buffer_size/40)    # there are total of 40 arrays of size "buffer_size" used at once later, so the number has to be adjusted to correctly represent the amount of mermory the arrays take up
            if binary:
                outfile = open(f"dust_temperature.bdat"+note,"wb")
                np.array([1,8,nrcells,nrspecs]).tofile(outfile) # Necessary parameters for radmc3d: 1 (always present), 8(# of bytes per datapoint),...
            if not binary:
                outfile = open("dust_temperature.dat" + note,"w")
                np.array([1,nrcells,nrspecs]).tofile(outfile, sep = "\n") #Necessary parameters for radmc3d: 1 (always present)...
                outfile.write("\n")

            timer = time.time()+60
            for phi_index in range(len(self.phi_med)):
                for theta_index, theta in enumerate(self.theta_med):
                    cont = False
                    first_index = 0
                    last_index = buffer_size
                    while not cont:
                        if last_index > len(self.r_med)-1:                                                    #
.*z3
```

```
                        last_index = len(self.r_med)                                                    #
 .*      *
                        cont = True                                                                     #       .*
            .
                    rs = self.r_med[first_index:last_index]                                             #   .*
             .
                    equatorial_rs = np.sin(theta)*rs                                                    #z4*
 o<-z       . <- elementary volume from the side
                    equatorial_r1s = np.sin(self.theta_sup[theta_index])*rs                             #  *
              *
                    equatorial_r2s = np.sin(self.theta_sup[theta_index])*rs                             #    *
              .
                    equatorial_r3s = np.sin(self.theta_inf[theta_index])*rs                             #     .
        ____.z2
                    equatorial_r4s = np.sin(self.theta_inf[theta_index])*rs                             #       z1
..----
                    zs = np.cos(theta)*rs                                                               #Average T
 from points z through z4 is saved
                    z1s = self.r_inf[first_index:last_index]*np.cos(self.theta_sup[theta_index])
                    z2s = self.r_sup[first_index:last_index]*np.cos(self.theta_sup[theta_index])
                    z3s = self.r_sup[first_index:last_index]*np.cos(self.theta_inf[theta_index])
                    z4s = self.r_inf[first_index:last_index]*np.cos(self.theta_inf[theta_index])
                    interpolated_Ts =  np.interp(equatorial_rs,  self.r_med, self.T[:,phi_index])
                    interpolated_T1s = np.interp(equatorial_r1s, self.r_med, self.T[:,phi_index])
                    interpolated_T2s = np.interp(equatorial_r2s, self.r_med, self.T[:,phi_index])
                    interpolated_T3s = np.interp(equatorial_r3s, self.r_med, self.T[:,phi_index])
                    interpolated_T4s = np.interp(equatorial_r4s, self.r_med, self.T[:,phi_index])
                    interpolated_sigmas =  np.interp(equatorial_rs,  self.r_med, self.dust_density[:,phi_index])
                    interpolated_sigma1s = np.interp(equatorial_r1s, self.r_med, self.dust_density[:,phi_index])
                    interpolated_sigma2s = np.interp(equatorial_r2s, self.r_med, self.dust_density[:,phi_index])
                    interpolated_sigma3s = np.interp(equatorial_r3s, self.r_med, self.dust_density[:,phi_index])
                    interpolated_sigma4s = np.interp(equatorial_r4s, self.r_med, self.dust_density[:,phi_index])
                    temps  = thickness(h =  zs*au, sigma = interpolated_sigmas,  T = interpolated_Ts,
 equatorial_r=equatorial_rs*au,  M_star= self.par["M_star"][0])
                    temp1s = thickness(h = z1s*au, sigma = interpolated_sigma1s, T = interpolated_T1s,
 equatorial_r=equatorial_r1s*au, M_star= self.par["M_star"][0])
                    temp2s = thickness(h = z2s*au, sigma = interpolated_sigma2s, T = interpolated_T2s,
 equatorial_r=equatorial_r2s*au, M_star= self.par["M_star"][0])
                    if theta_index != 0:
                        temp3s = thickness(h = z3s*au, sigma = interpolated_sigma3s, T = interpolated_T3s,
 equatorial_r=equatorial_r3s*au, M_star= self.par["M_star"][0])
                        temp4s = thickness(h = z4s*au, sigma = interpolated_sigma4s, T = interpolated_T4s,
 equatorial_r=equatorial_r4s*au, M_star= self.par["M_star"][0])
                        average_temps = (temps+temp1s+temp2s+temp3s+temp4s)/5
                    else: average_temps = (temps+temp1s+temp2s)/3
                    if binary:
                        average_temps.tofile(outfile)
                    if not binary:
                        average_temps.tofile(outfile, sep = "\n", format="%.16e")
                        outfile.write("\n")
                    if cont: break
                    first_index, last_index = last_index, last_index + buffer_size

            if time.time()>timer:
                print(f"{phi_index/(len(self.phi_med)-1)*100:.2f} % done")
                timer += 300

        outfile.close()

    def radmc_read_temperature(self, isbinary = None, filename = None):
        """
        Reads radmc3d dust_temperature.inp file. It's usually quite large so watch out
        --------------------
        Parameters:
        --------------------
        isbinary: bool, if True, will try and load the file as binary, if False, will try and load as text, if None,
 will first check each line of the file to check whether it's binary

        filename: string, name of the file to be loaded. If None will look for typical filename: 'dust_temperature*'
        """

        if self.theta_med is None: raise ValueError("Temperature error: Disk hasn't been thickened yet, no knowledge
 of what shape the temperature is! Load the grid first.")

        if filename is None:
            import glob
            radm_files = glob.glob("dust_temperature*")
            if len(radm_files) > 1: raise IOError("Found multiple radmc3d temperature files")
            #far_files = glob.glob("gastemper*.dat")
            #if len(far_files) > 1: raise IOError("Found multiple fargo temperature files")
            files = radm_files#+far_files
            if len(files) > 1: raise IOError("Found both fargo and radmc3d temperature files. Specify file name")
            filename = files[0]

        if isbinary is None:
            print("Read temperature notice: unknown file type, checking for binary...")
            isbinary = is_file_binary(filename)

        if isbinary: self.T = np.fromfile(filename, dtype = np.double, offset = 4*8)
        else: self.T = np.loadtxt(filename, skiprows = 3)

        self.T = np.reshape(self.T, (len(self.phi_med), len(self.theta_med), len(self.r_med)))
        self.T = np.transpose(self.T, (2,1,0))
        self.T = np.reshape(self.T, (len(self.r_med), len(self.theta_med), len(self.phi_med)))

    def radmc_read_density(self, isbinary = None, filename = None):
        """
        Reads radmc3d dust_density.inp file. It's usually quite large so watch out
        """
```

```
        if self.theta_med is None: raise ValueError("Density error: Disk hasn't been thickened yet, no knowledge of what shape the density is! Load the grid first.")

        if filename is None:
            import glob
            radm_files = glob.glob("dust_density*")
            if len(radm_files) > 1: raise IOError("Found multiple radmc3d density files")
            filename = radm_files[0]

        if isbinary is None:
            print("Read density notice: unknown file type, checking for binary...")
            isbinary = is_file_binary(filename)

        if isbinary: self.dust_density = np.fromfile(filename, dtype = np.double, offset = 4*8)
        else: self.dust_density = np.loadtxt(filename, skiprows = 3)

        self.dust_density = np.reshape(self.dust_density, (len(self.phi_med), len(self.theta_med), len(self.r_med)))
        self.dust_density = np.transpose(self.dust_density, (2,1,0))
        self.dust_density = np.reshape(self.dust_density, (len(self.r_med), len(self.theta_med), len(self.phi_med)))*gram/cm/cm/cm

    def radmc_read_grid(self, filename = None):

        if filename is None:
            import glob
            radm_files = glob.glob("amr_grid*")
            if len(radm_files) > 1: raise IOError("Found multiple radmc3d grid files")
            filename = radm_files[0]

        loaded = np.fromfile(filename, sep = " ")
        n_r =    int(loaded[7])
        n_thet = int(loaded[8])
        n_phi =  int(loaded[9])

        self.r_inf = loaded[10:10+n_r]*cm/au
        self.r_sup = loaded[10+1:10+n_r+1]*cm/au
        self.r_med = 0.5*(self.r_inf+self.r_sup)
        self.par["Nrad"] = n_r

        self.theta_inf = loaded[10+n_r+1:  10+n_r+n_thet+1]
        self.theta_sup = loaded[10+n_r+2:10+n_r+n_thet+2]
        self.theta_med = 0.5*(self.theta_inf+self.theta_sup)
        self.par["Nthet"] = n_thet

        self.phi_inf = loaded[10+n_r+n_thet+2:  10+n_r+n_thet+n_phi+2]
        self.phi_sup = loaded[10+n_r+n_thet+3:10+n_r+n_thet+n_phi+3]
        self.phi_med = 0.5*(self.phi_inf+self.phi_sup)
        self.par["Nsec"] = n_phi

        print(f"Updated internal grid, which now consists of {n_r}*{n_thet}*{n_phi} r*theta*phi cells")

    def interpolate_temp(self, cutoff = 0.95):
        """
        Fills in holes in the temperature grid with the interpolation of cells neighboring it in the phi direction
        """
        print("Correcting disk's temperature...")
        total = 0
        for r_index in range(len(self.r_med)):
            for theta_index in range(len(self.theta_med)):
                shifted_left = np.roll(self.T[r_index, theta_index,:], -1)
                shifted_right = np.roll(self.T[r_index, theta_index,:], 1)
                average_of_neighbours = 0.5*(shifted_left+shifted_right)
                to_fix = np.where(self.T[r_index, theta_index,:]<average_of_neighbours*cutoff)[0]   #this [0] is there cuz np.where returns tuple of arrays, but the second element is nothing in this case
                total += len(to_fix)
                for bad_index in to_fix:
                    self.T[r_index, theta_index, bad_index]=average_of_neighbours[bad_index]
        print(f"Altered {total} out of {np.size(self.T)} cells.")

    def add_heatsource(self, binary = True, thickness = f_z):
        """
        Adds heatsource file for radmc3d, which imitates basic viscous heating

        according to https://www.ita.uni-heidelberg.de/~dullemond/lectures/leshouches2013.pdf,
        https://www.epj-conferences.org/articles/epjconf/pdf/2015/21/epjconf_ppd2014_00011.pdf

        Using the equation Q_plus = honu[r d r Omega]^2 kde nu = alpha*c_s^2/Omega kde alpha, see eq 20 https://arxiv.org/pdf/2110.11341.pdf

        heat transfer not included!
        """

        if len(self.dust_density) == 0: self.radmc_read_density()
        if self.dust_density.ndim == 2: raise ValueError("Heatsource error: disk is flat!")

        simple_modifier = 0.01
        self.gas_density = self.dust_density/simple_modifier # dust to gas

        if not binary:
            outfile = open("heatsource.inp", "w")
            outfile.write(f"1\n{len(self.phi_med)*len(self.theta_med)*len(self.r_med)}\n")
        if binary:
            outfile = open("heatsource.binp", "wb")
            np.array([1, 8, len(self.phi_med)*len(self.theta_med)*len(self.r_med)]).tofile(outfile)    #header containting 1, num of bits (8) and number of cells

        alpha = 1e-2    # estimate according to Schobert et al 2019
        for phi_index, phi in enumerate(self.phi_med):  #double for loop, might be too slow!
```

```
            for theta_index, theta in enumerate(self.theta_med):

                    nu = R_gas*self.T[:,theta_index,phi_index]*alpha/mu*np.sqrt((self.r_med*au)**3/G/self.par["M_star"]) # turbulent viscostiy in SI # nu_t = alpha*c^2/Omega; Schobert et. al. 2019 eq. 20

                    Q_plus = 9/4*self.gas_density[:,theta_index,phi_index]*nu*(G*self.par["M_star"]/(self.r_med*au)**3)

                    if not binary:
                        (Q_plus/erg*cm*cm*cm).tofile(outfile, sep = "\n", format = '%.16e')
                        outfile.write("\n")
                    if binary:
                        (Q_plus/erg*cm*cm*cm).tofile(outfile, dtype = np.double)

        outfile.close()
        print("heatsource file was created")


    def evaporate_densities(self, show_scattering_modes=False):
        """
        Evaporates dust if its temperature and density corresponds to 'mode 6 or higher' of opacity (i.e. grain evaporation)
        according to Zhu etal. (2012), http://www.physics.unlv.edu/~zhzhu/Opacity.html

        show_scattering_modes: bool, if true plots slice of disc that shows where in the disc is which mode
        """
        print("Evaporating densities...")
        from opacities import ZHU
        simple_modifier = 0.01
        gas_density = self.dust_density[:]/simple_modifier   # since for the complex mode one needs 3D temp, we're assuming you already did radmc3d sim to get that temp, which means this density is of the dust, so lets turn it back to gas for this
        if show_scattering_modes:
            density = np.append(gas_density[:,:,0], gas_density[:,:,0][:,::-1], axis = 1)
            temperature = np.append(self.T[:,:,0], self.T[:,:,0][:,::-1], axis = 1)
            data = ZHU(rho = density/simple_modifier, T = temperature, return_mode = True)
            theta_bound = np.append(self.theta_inf[:], self.theta_sup[-1])
            theta_bound = np.append(theta_bound, np.pi-self.theta_inf[:][::-1])
            r_mesh, theta_mesh = np.meshgrid(np.append(self.r_inf, self.r_sup[-1]), theta_bound)
            x_mesh = r_mesh*np.sin(theta_mesh)
            y_mesh = r_mesh*np.cos(theta_mesh)
            from matplotlib import colors as c
            mode_names = ["water ice grains", "water ice evaporation", "non-water grains", "graphite corrosion", "non-graphite grains", "grain evaporation", "water vapour", "water dissociation", "molecules", "H scattering", "bound-free, free-free", "e- scattering", "molecules and H scattering"]
            cMap = plt.cm.get_cmap("jet", len(mode_names)-1)
            fig = plt.figure()
            ax = fig.add_subplot(111)
            ax.set_aspect('equal')
            ap = ax.pcolormesh(x_mesh, y_mesh, data.T, cmap = cMap, vmin=np.min(data) - 0.5, vmax=np.max(data) - 0.5)
            plt.xlabel("r [au]")
            plt.ylabel("z [au]")
            plt.title(f"Opacity mode")
            cbar = plt.colorbar(ap)
            cbar.ax.set_yticks(range(13))
            cbar.ax.set_yticklabels(mode_names)
            plt.show()

        modes = ZHU(rho = gas_density, T = self.T, return_mode = True)
        evaporation_coefficient = 1e-10  #by how much to decrease the density as the grains evaporate
        evaporation = np.where(modes>5, evaporation_coefficient, 1)
        self.dust_density *= evaporation

    def create_grain_distribution(self, grainsizes = np.array([0.1, 1, 10, 100, 1000, 10000])*micron):
        """
        Turns single dust density field into several that correspond to different grainsizes
        """
        self.dust_species = []  # clear previous species name

        power_of_distribution = -.5     # n_d ~ d^-3.5   m_d ~ d^3   m_all_d = n_d * m_d ~ d^-.5

        if len(self.dust_density) == 0: raise ValueError("Dust density not yet created! First get that sorted out before you bother me with dustsizes")
        new_multidust_array = []

        original_density = self.dust_density

        for grainsize in grainsizes:
            self.dust_species.append(f"{grainsize/micron:.1f}") # convention to write size in dustkappa file in microns
            this_species = self.dust_density*np.power(grainsize, power_of_distribution)
            new_multidust_array.append(this_species)

        new_multidust_array = np.array(new_multidust_array)
        new_multidust_array /= np.sum(new_multidust_array, axis = 0)    # now is the new array normalized in the sense that the sum over all species in each 3D cell is 1
        new_multidust_array = np.nan_to_num(new_multidust_array, nan = 0.0) #in normalization if the orig array had zeros somewhere, so will the sum over all species and therefore nans will arise

        new_multidust_array *= original_density # and now each 3D cell if summed over all species should have same value as before

        self.dust_density = new_multidust_array
        print(f"Exchanged original dust for {len(grainsizes)} different sized species of dust!")


def main():
    """
    Test program for the Disk class
```

```
    writes radmc input files similar to my bc thesis' ones
    """
    disk = Disk()
    disk.fargo_read_fields(no = 4)
    disk.fargo_input()
    disk.inner_outer_finish(r_newmax=30, inner_steps=256, outer_steps = 512)
    disk.plot_T_profile()
    disk.flat_relax(radial=512, angular=24*16)
    disk.radmc_write_inputs(nthet = 32, extra_margin=5, nphot = 1e10)



if __name__ == "__main__":
    main()
```

A.2 Výpis programu opacities.py

Skript na základě práce Zhu a kol. 2012, dostupný online z `https://sirrah.troja.mff.cuni.cz/~janoska/bakalarka.html`.

```python
#!/usr/bin/env python
# -*- coding: utf-8 -*-
"""
Set of functions that return opacities for a bunch of situations
"""
import matplotlib
from matplotlib.colorbar import ColorbarBase
import numpy as np

@np.vectorize
def ZHU(rho, T, return_mode = False):
    """
    Calculates opacity based on Zhu etal. (2012), http://www.physics.unlv.edu/~zhzhu/Opacity.html
    --------------------
        Parameters:
    --------------------
    rho [kg/m^3]: double, density of the cloud

    T [K]: double, its temperature

    return_mode: bool, if True, will return the mode of scattering
    """

    xlp = np.log10(rho*T*8.314472*1.e7/2.4)
    xlt = np.log10(T)

    def kappa_ZHU1(rho,T):  return 1.5*(xlt -1.16331)-0.736364          # water ice grains
    def kappa_ZHU2(rho,T):  return -3.53154212*xlt +8.767726-(7.24786-8.767726)*(xlp  -5.)/16.  # water ice evaporation
    def kappa_ZHU3(rho,T):  return 1.5*(xlt -2.30713)+0.62            # non-water grains
    def kappa_ZHU4(rho,T):  return -5.832*xlt +17.7             # graphite corrosion
    def kappa_ZHU5(rho,T):  return 2.129*xlt -5.9398              # no-graphite grains
    def kappa_ZHU6(rho,T):  return 129.88071-42.98075*xlt +(142.996475-129.88071)*0.1*(xlp  +4) # grain evaporation
    def kappa_ZHU7(rho,T):  return -15.0125+4.0625*xlt            # water vapour
    def kappa_ZHU8(rho,T):  return 58.9294-18.4808*xlt +(61.6346-58.9294)*xlp  /4.    # water dissociation
    def kappa_ZHU9(rho,T):  return -12.002+2.90477*xlt +(xlp  -4)/4.*(13.9953-12.002)   # molecules
    def kappa_ZHU10(rho,T): return  -39.4077+10.1935*xlt +(xlp  -4)/2.*(40.1719-39.4077)  # H scattering
    def kappa_ZHU11(rho,T): return  17.5935-3.3647*xlt +(xlp  -6)/2.*(17.5935-15.7376)    # bound-free, free-free
    def kappa_ZHU12(rho,T): return  -0.48               # e- scattering
    def kappa_ZHU13(rho,T): return  3.586*xlt -16.85              # molecules and H scattering

    xlop = None
    if xlt  < 2.23567+0.01899*(xlp-5.):
        mode = 1
        xlop =  kappa_ZHU1(rho,T)
    elif xlt  < 2.30713+0.01899*(xlp-5.):
        mode = 2
        xlop =  kappa_ZHU2(rho,T)
    elif xlt  < (17.7-0.62+1.5*2.30713)/(1.5+5.832):
        mode = 3
        xlop =  kappa_ZHU3(rho,T)
    elif xlt  < (5.9398+17.7)/(5.832+2.129):
        mode = 4
        xlop =  kappa_ZHU4(rho,T)
    elif xlt  < (129.88071+5.9398 + (142.996475-129.88071)*0.1*(xlp+4))/(2.129+42.98075):
        mode = 5
        xlop =  kappa_ZHU5(rho,T)
    elif xlt  < (129.88071+15.0125 + (142.996475-129.88071)*0.1*(xlp+4))/(4.0625+42.98075):
        mode = 6
        xlop =  kappa_ZHU6(rho,T)
    elif xlt  < 3.28+xlp/4.*0.12:
        mode = 7
        xlop =  kappa_ZHU7(rho,T)
    elif xlt  < 3.41+0.03328*xlp/4.:
        mode = 8
        xlop =  kappa_ZHU8(rho,T)
    elif xlt  < 3.76+(xlp-4)/2.*0.03:
        mode = 9
        xlop =  kappa_ZHU9(rho,T)
    elif xlt  < 4.07+(xlp-4)/2.*0.08:
        mode = 10
        xlop =  kappa_ZHU10(rho,T)
    elif xlt  < 5.3715+(xlp-6)/2.*0.5594:
        xlop =  kappa_ZHU11(rho,T)
        mode = 11
    else:
        xlop =  kappa_ZHU12(rho,T)
        mode = 12

    xlop_ = xlop

    if xlop < 3.586*xlt-16.85 and xlt <4.:
        xlop_ = kappa_ZHU13(rho,T)
        mode = 13

    if return_mode: return mode
    return np.power(10, xlop_)

if __name__ == "__main__":
    main()
```