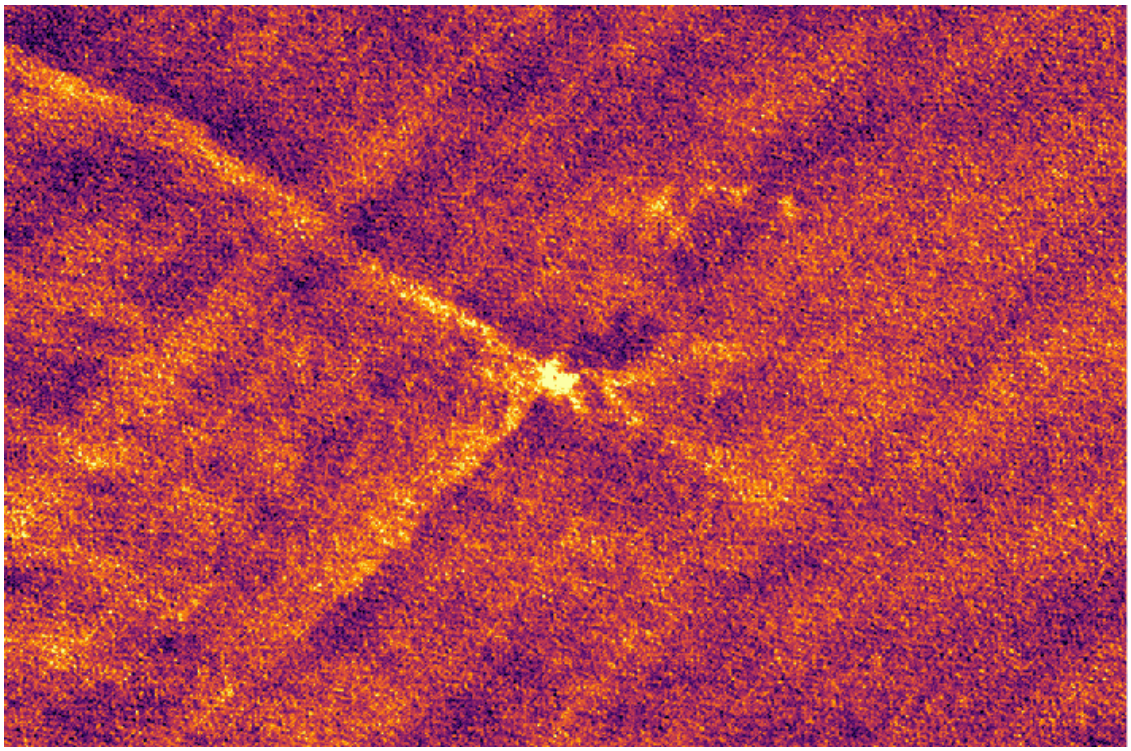
CHALMERS



Studie av molekylära utflöden från protoplanetära skivor

En systematisk arkivstudie av observationer från ALMA-teleskopet

Kandidatarbete inom Rymd-, geo- och miljövetenskap

MARKUS HJÄLT, CHRISTOPHER LARSSON,
ANTON ROSÉN, LUKAS THIM, TOMAS THURE

INSTITUTIONEN FÖR RYMD-, GEO- OCH MILJÖVETENSKAP

CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige 2022
www.chalmers.se

Kandidatarbete 2022

Studie av molekylära utflöden från protoplanetära skivor

En systematisk arkivstudie av observationer från ALMA-teleskopet

MARKUS HJÄLT
CHRISTOPHER LARSSON
ANTON ROSÉN
LUKAS THIM
TOMAS THURE

CHALMERS

Institutionen för Rymd-, geo- och miljövetenskap
Chalmers tekniska högskola
Göteborg, Sverige 2022

Studie av molekylära utflöden från protoplanetära skivor
En systematisk arkivstudie av observationer från ALMA-teleskopet

MARKUS HJÄLT
CHRISTOPHER LARSSON
ANTON ROSÉN
LUKAS THIM
TOMAS THURE

Handledare: Per Bjerkeli och Eva Wirström, Institutionen för Rymd-, geo- och miljövetenskap
Examinator: Magnus Thomasson, Institutionen för Rymd-, geo- och miljövetenskap

Kandidatarbete 2022
Institutionen för Rymd-, geo- och miljövetenskap
Chalmers tekniska högskola
412 96 Göteborg, Sverige
Telefon +46 31 772 1000

Omslag: Bilden visar ett blåförskjutet molekylärt utflöde från den protoplanetära skivan runt protostjärnan B335 studerat med ALMA-teleskopet i band 6.

Typsatt i LaTeX
Göteborg, Sverige 2022

Studie av molekylära utflöden från protoplanetära skivor
En systematisk arkivstudie av observationer från ALMA-teleskopet
Markus Hjält, Christopher Larsson, Anton Rosén, Lukas Thim och Tomas Thure
Institutionen för Rymd-, geo- och miljövetenskap
Chalmers tekniska högskola

Sammandrag

När en ny stjärna bildas, börjar processen med att moln av gas och stoft kollapsar till en central protostjärna med en omkringliggande protoplanetär skiva. Stoft och gas från skivan faller kontinuerligt in mot protostjärnan. Massa lämnar dock även systemet i form av snabba jetstrålar och långsammare molekylära utflöden från protostjärnan och skivan.

I detta arbete är målet att utveckla ett verktyg som kan underlätta för forskare att dra slutsatser om utflöden, dess sammansättning och dess utbredning. Datan som används i projektet kommer från ALMA-teleskopets (Atacama Large Millimeter/-submillimeter Array) publika arkiv. Urvalet av intressanta observationer utförs systematiskt med hjälp av ett beslutsträd. Beslutsträdet har utformats för att filtrera ut observationer baserat på till exempel vinkelupplösning, förekomst av vissa molekyler och olika nyckelord. Urval, nedladdning och analys av intressant data görs med hjälp av existerande funktioner i den digitala verktygslådan *ALminer*. Programmet börjar med att producera momentavbildningar över frekvensintervall som innehåller blå- respektive rödförskjuten emission. Utifrån dessa kan vinkeln på utflöden uppskattas. Därutöver förs statistik över, bland annat, antalet observerade objekt som innehåller vissa molekyler.

Med hjälp av dessa metoder har ett verktyg utvecklats som kan stödja forskare med insamling, filtrering och analys av data från ALMA-arkivet. Via verktyget får forskare tillgång till nedladdningsrutiner, statistiska verktyg och analytiska funktioner för att skapa momentavbildningar och vinkeluppskattningar för utflöden.

Nyckelord: protostjärna, protoplanetär skiva, molekylärt utflöde, ALMA-teleskopet, ALminer

Study of molecular outflows from protoplanetary disks
A systematic archive study of observations made by the ALMA-telescope
Markus Hjält, Christopher Larsson, Anton Rosén, Lukas Thim and Tomas Thure
Department of Space, Earth and Environment
Chalmers University of Technology

Abstract

When a new star is formed, the process begins with clouds of gas and dust that collapses into a central protostar with a surrounding protoplanetary disk. Dust and gas from the disk continuously falls in towards the protostar. Mass is also expelled from the system in the form of jets and slower moving molecular outflows from the protostar and its disk.

The aim of this project is to develop a tool that will make it easier for scientists to draw conclusions about outflows, their composition and their distribution. The data in this project is taken from the ALMA-telescope's (*Atacama Large Millimeter/submillimeter Array*) public archive. The selection of interesting observations is done systematically with the help of a decision tree. The decision tree has been designed to filter out observations based on angular resolution, presence of certain molecules and different keywords. The selection, downloading and analysis of interesting data is done with the help of existing functions in the digital toolbox *ALminer*. The program starts by producing moment maps over frequency intervals containing blue- repectively redshifted emission. From these, the angle of the outflow can be estimated. Moreover, statistics are gathered about, among others, the number of observed objects containing certain molecules.

With these methods, a tool has been developed that can support scientists with collecting, filtering and analysing data from the ALMA-archive. Through the use of this tool, scientists get access to download routines, statistical tools and analytical functions to produce moment maps and angular estimations of outflows.

Keywords: Protostar, Protoplanetary disk, Molecular outflow, ALMA-telescope, ALminer

Förord

Först och främst vill vi rikta ett stort tack till våra handledare Per Bjerkeli och Eva Wirström. Utan deras vägledning, stöd och kompetens hade inte genomförandet av detta projekt varit möjligt. Våra spännande och kreativa diskussioner vid de veckoliga mötena drev projektet framåt och inspirerade gruppen. Projektets forskningsnära karaktär gav oss stor inblick i hur modern forskning går till och gav oss en känsla av att produkten som skapats faktiskt kan komma till användning i framtiden.

Vi vill även tacka Adele Plunkett, astronom vid *National Radio Astronomy Observatory*. Hennes råd kring databehandling var insiktsfulla och hennes idéer hjälpte oss att tackla svåra problem.

Markus Hjält, Christopher Larsson, Anton Rosén, Lukas Thim och Tomas Thure, Göteborg, maj 2022

Innehåll

1
Inledning

Något vi människor alltid funderat över är vart vi kommer ifrån och hur vårt solsystem en gång bildades. Det är dessvärre svårt att undersöka eftersom solsystemet är cirka 4,5 miljarder år gammalt [1]. Som tur är bildas nya stjärnor och solsystem ständigt runt om i vår galax. Avstånden är dock för stora för att människor ska kunna ta sig dit. Det närmsta stjärnbildande området i galaxen, Orionnebulosan, är belägen ungefär 1 350 ljusår bort [2]. De nybildade stjärnorna måste istället studeras med hjälp av teleskop från jorden och rymden. Genom att studera hur nya solsystem bildas kan forskare dra slutsatser om hur vårt eget solsystem bildades en gång i tiden. Detta ökar också förståelsen för hur planeter, likt jorden, kan bildas.

För att en ny stjärna ska bildas, krävs det ett enormt moln av molekyler och stoft [3]. Molnet kommer att, på grund av gravitationskraften, ackumulera materia från omkringliggande regioner med följden att molnets massa ökar. Till slut når molnet en kritisk massa vid vilken gravitationskraften i molnet övervinner det utåt verkande trycket och molnet kollapsar. Vid kollapsen koncentreras gasen i centrum av molnet och ett klotformat objekt bildas, vilket senare kommer att bli en stjärna. Runt denna *protostjärna* bildas det även en skiva av stoft. Skivan bildas eftersom all gas inte faller in till fullo på grund av de krafter molnets rotation ger upphov till.

När massa från skivan kollapsar in mot protostjärnan ökar rörelsemängdsmomentet [3]. Detta eftersom rotationshastigheten ökar när radien minskar. Eftersom rörelsemängdsmoment är en bevarad storhet måste det totala rörelsemängdsmomentet konserveras. För att upprätthålla denna jämvikt bildas bipolära utflöden från regionen kring protostjärnan genom vilka molekyler och stoft lämnar molnet. Hur dessa utflöden uppstår och hur de beter sig är ännu till viss del oklart och det är därför av stort intresse att studera utflöden närmare. Att studera utflöden är viktigt för att förstå miljön där planetbildning sker och hur det eventuellt kan påverka planeternas sammansättning och bildning [4].

Utflöden och unga stjärnor studeras bland annat med *ALMA-teleskopet* (*Atacama Large Millimeter/submillimeter Array*) i Chile [5]. ALMA-teleskopet är lämpligt för detta ändamål eftersom dessa objekt kräver ett teleskop med god upplösning som kan observera i rätt våglängdsintervall [6][7]. Alla observationer som utförs med ALMA lagras i ett stort arkiv, ALMA-arkivet [8]. Det är dock en tidskrävande process för forskare att manuellt genomsöka detta arkiv för att leta efter olika observationer och ladda ner dem för att sedan kunna analysera datan. Därför syftar detta projekt till att utveckla ett program som hjälper forskare att på ett effektivt sätt filtrera fram relevant data från ALMA-arkivet, med fokus på utflöden, för att sedan automatiskt kunna analysera utflöden närmare.

1.1 Syfte

Det övergripande syftet med projektet är att utveckla ett program som hjälper forskare att systematiskt genomsöka ALMA-arkivet efter specifika observationer och identifiera förekomsten och distributionen av molekyler samt stoft i utflödet hos protostjärnor. En av de viktigaste delarna för att uppnå detta är skapandet av ett beslutsträd som programmet använder för att sålla ut intressanta objekt för vidare analys. Med programmet kan stora datamängder analyseras mer effektivt då denna analys i dagsläget huvudsakligen görs manuellt. Att effektivisera analysmetoden är även av intresse ur ett tidsperspektiv, då den manuella analysen är tidskrävande. Slutprodukten, i form av programmet, syftar till att vara användbart för astronomer eller andra forskare och kommer att kunna användas för att systematiskt analysera ALMA-arkivet eller vidareutveckla programmet för framtida forskning.

2

Teori

För att bättre förstå principerna bakom projektets metodik krävs en del bakgrundskunskap. Här ges bakgrund kring hur stjärnbildning sker, vad ALMA-teleskopet är och hur det fungerar, hur observationer genomförs med teleskopet och hur datan lagras. Även teorin bakom kvantmekanik, som ligger till grund för rotationsövergångar och emissionsspektrum hos atomer och molekyler, beskrivs. Vid databehandlingen används momentavbildningar och kurvanpassningar, så även det förklaras i detta kapitel.

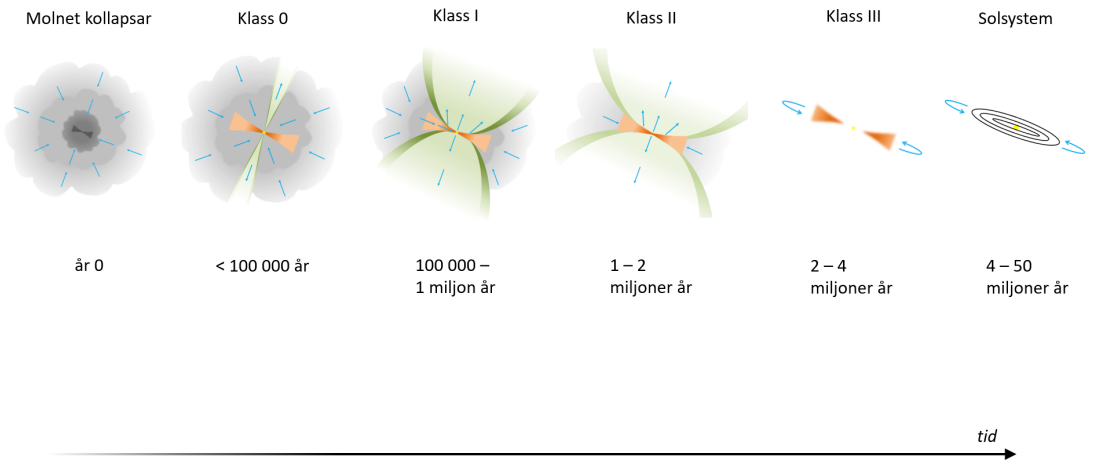
2.1 Stjärnbildning

Inuti galaxers skivor förekommer interstellära moln bestående av gas och stoft. Dessa massiva moln kan motsvara så mycket som 100 000 solmassor [3]. De består till största delen av molekylärt väte, H_2, och har en genomsnittlig densitet av 100 H_2-molekyler per kubikcentimeter. Molnen är dock inte homogena och i vissa områden kan densiteten uppgå till 10 000 H_2-molekyler per kubikcentimeter. Det är i dessa områden med hög densitet som stjärnbildning sker.

De molekylära molnen påverkas av en inåtriktad gravitationskraft. Denna kraft är i jämvikt med molnens inre termiska tryck, turbulenta gasrörelse samt ett inre magnetfält [3]. Då molnet uppnår en kritisk massa, den så kallade Jeansmassan, blir dock molnet gravitationellt instabilt och en kollaps mot den täta kärnan påbörjas. Jeansmassan ges av

$$M_J = \left(\frac{5k_bT}{Gm}\right)^{\frac{3}{2}} \left(\frac{3}{4\pi\rho}\right)^{\frac{1}{2}}, \tag{2.1}$$

där k_b är Boltzmanns konstant, T är temperaturen, G är gravitationskonstanten, m är massan av en gaspartikel och ρ är molnets densitet [9]. Ekvationen indikerar att ju kallare och tätare ett moln är, desto lägre blir Jeansmassan. Kalla och täta moln är alltså mer benägna att kollapsa jämfört med varma och tunna moln enligt Ekvation 2.1.



Figur 2.1: En schematisk bild över stjärnbildningsprocessens tidslinje (Bjerkeli, 2022).

Det är i centrum av de täta kärnorna som protostjärnor bildas, vilket är det första steget i stjärnbildningsprocessen [3]. En stjärna klassas som en protostjärna i cirka 100 000 år och förblir i detta stadie så länge som det tillförs massa från det omkringliggande höljet bestående av molekyler och stoft. Detta stadium motsvarar klass 0 i Figur 2.1, som visar stjärnbildningsprocessens olika steg längs en tidslinje och hur det bipolära utflödet förändras med tiden. Notera att tiderna i figuren är ungefärliga och endast gäller för lågmassiva stjärnor.

Tillförseln av massa från höljet sker med en hastighet av några jordmassor per år [3]. Samtidigt som tillförseln sker, förekommer även stabila utflöden av materia från området kring protostjärnans poler. Det är ofta genom dessa bipolära utflöden som nya protostjärnor upptäcks eftersom protostjärnan själv är svår att detektera.

Samtidigt som bipolära utflöden pågår, kommer protostjärnans densitet att öka eftersom materia faller in mot dess centrum och dess storlek minskar [3]. Rotationshastigheten runt protostjärnans centrum kommer också att öka då dess radie minskar. Materia med låg hastighet faller direkt in mot protostjärnan medan materia med tillräckligt hög hastighet istället hamnar i omloppsbana runt protostjärnan. På grund av materians varierande hastighet får dessa omloppsbanor olika radier, vilket medför att en skiva runt prototjärnan bildas.

När det omkringliggande stoftet och gasen har dragits in till protostjärnan eller hamnat i skivan, anses inte den centrala gasansamlingen vara en protostjärna längre utan den har nu blivit en så kallad *"pre-main-sequence star"* eller en före huvudseriestjärna [3]. Detta stadium definieras av att temperaturen i kärnan nu är tillräckligt hög, cirka 1 000 000 K, för att få deuteriumatomer, ^{2}H, att slås samman med protoner, H^+, och således bilda helium-3, ^{3}He. Det är just denna reaktion som definierar en före huvudseriestjärna [10]. Stjärnan har i och med detta nått klass III i Figur 2.1.

Stadiet innan stjärnan når huvudserien kan pågå upp till ett tiotal miljoner år [3]. Någon gång under den tidigare perioden av detta stadiet, under de första miljoner åren, börjar protostjärnan kunna observeras med optiska teleskop. Från och med denna tidpunkt kallas dessa objekt *T Tauri-stjärnor*. Dessa omges fortfarande av en protoplanetär skiva som representeras av klass II i Figur 2.1 och regionen kring stjärnan har ett fortsatt bipolärt utflöde. Efter ett antal miljoner år har gasen och stoftet i den protoplanetära skivan försvunnit genom utflödet och en ung stjärna finns kvar i centrum. I vissa fall kan relativt stora objekt bli kvar i omloppsbana runt stjärnan i så kallade fragmentskivor eller *"debris disks"*. Det är ur dessa skivor som planeter, asteroider och månar har sitt ursprung [3]. Detta kan representeras av det utvecklade solsystemet i Figur 2.1.

Efter ytterligare tiotals miljoner år kommer den inåtverkande gravitationskraften börja dominera över det utåtverkande trycket [3]. Denna kompression höjer temperaturen i stjärnans kärna till cirka 10 000 000 K. Vid denna temperatur kan fyra protoner, H^+, slås samman och bilda en helium-4 atom, ^{4}He. Det är denna fusionsreaktion som producerar den avsevärt största mängden energi i stjärnan och i samband med att denna reaktion påbörjas har stjärnan nått ett tillstånd där den är mycket stabil. Det är också när fusionsreaktionen påbörjas i stjärnans inre som den når huvudserien av sin livscykel och kommer förbli där under miljarder år.

2.2 Utflöden

En viktig aspekt av stjärnbildningsprocessen är utflöden. När massa från de molekylära molnen faller in mot den roterande skivan, kommer rörelsemängdsmomentet öka [3]. För att det totala rörelsemängdsmomentet ska bevaras måste massa lämna systemet. Det är utflöden från molnet som utgör denna naturliga massförlust, och således också motsvarande förlust av rörelsemängdsmoment [11].

Utflöden är ett fenomen som påträffas i unga stjärnor som fortfarande har ett molekylärt moln runt sig [3]. I utflöden från unga stjärnor kan två möjliga komponenter hittas, jetstrålar och långsammare vindar [12]. Båda är bipolära, alltså att de finns på båda sidor av skivan. Utöver denna likhet är utflödens karaktär ganska olika. Jetstrålar har hög hastighet och består av gas som skjuts ut i två smala, bipolära strålar. En typisk hastighet för jetstrålar är 300 km/s med en temperatur på 10 000 K. De långsammare vindarna däremot är mer konformade och består av långsammare och kallare gas. En typisk hastighet för dessa vindar är 10 km/s. Vindens hastighet tycks öka med minskande avstånd till mitten på utflödet i vad som liknar en lökstruktur [13]. Det innebär att hastigheten är segmenterad, alltså varierande för olika radier från den centrala axeln. Vindarnas temperatur är typiskt ungefär 10 K [12].

2.3 Kvantmekanik och emissionsspektrum

Följande avsnitt behandlar centrala begrepp inom kvantmekaniken som ligger till grund för uppkomsten av rotationsövergångar som kan studeras genom emissionsspektrum. Detta då emissionsspektrum är en viktig komponent i forskning kring utflöden.

2.3.1 Schrödingerekvationen

Ett centralt begrepp i kvantmekaniken är *vågfunktionen*, betecknad Ψ, som beskriver tillståndet för ett kvantmekaniskt system. Vågfunktionens utseende kan erhållas genom att lösa *Schrödingerekvationen* som är en partiell differentialekvation [14]. Schrödingerekvationen finns i två varianter, den *tidsberoende* och den *tidsoberoende*. Följande uttryck beskriver den tidsberoende Schrödingerekvationen,

$$\hat{H} \left|\Psi(\mathbf{r}, t)\right\rangle = i\hbar \frac{\partial}{\partial t} \left|\Psi(\mathbf{r}, t)\right\rangle , \tag{2.2}$$

där $\hat{H}$ är Hamiltonoperatorn som beskriver systemets energi, i är den imaginära enheten och $\hbar$ är Plancks konstant dividerad med 2π. Löses Ekvation 2.2 genom variabelseparation kommer lösningen av tidsberoendet enbart vara en periodisk fasfaktor och Ekvation 2.2 reduceras till

$$\hat{H} \left|\psi(\mathbf{r})\right\rangle = E \left|\psi(\mathbf{r})\right\rangle , \tag{2.3}$$

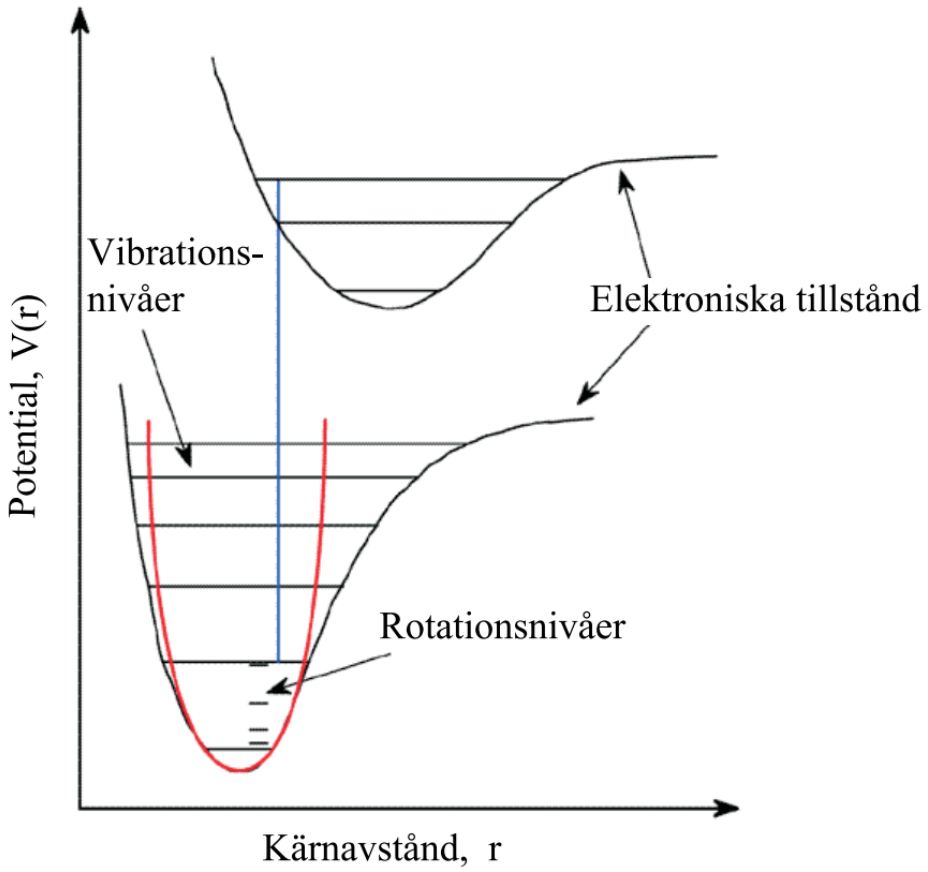
vilket är den tidsoberoende Schrödingerekvationen och är en egenvärdesekvation där E är egenenergierna till vågfunktionens egentillstånd. Denna teori ligger till grund för beskrivningen av rotationskvanttal som förklaras i Avsnitt 2.3.2.

2.3.2 Rotationskvanttal

Ljus som färdas med en viss våglängd har även en specifik energi. Atomer och molekyler kan absorbera dessa fotoner med specifika energier, för att hamna i elektroniskt exciterade tillstånd [15]. I Bohrs atommodell motsvarar detta att en elektron flyttas till ett högre skal [16]. När molekyler absorberar fotoner, uppstår dock även andra typer av excitationer [15]. Molekyler har nämligen fler frihetsgrader än atomer. Det medför att vibrations- och rotationsexcitationer kan uppstå. För molekyler finns alltså för varje elektronisk energinivå, en finare struktur bestående av vibrationstillstånd, som i sin tur innehåller rotationsnivåer. Den totala energin för molekylen kan därmed delas upp enligt

$$E = E_e + E_{vib} + E_{rot}, \tag{2.4}$$

där E är molekylens totala energi, E_e är energin från elektrontillståndet och E_{vib} och E_{rot} är energibidragen från vibrationer respektive rotationer [17]. Figur 2.2 visar schematiskt två olika elektroniska tillstånd för en diatomisk molekyl. Tillstånden beskrivs som potentialbrunnar vars minimum infaller vid jämviktsavståndet mellan de två atomerna [15]. De två elektrontillstånden delas även upp i olika vibrationsnivåer och rotationsnivåer. Notera att energin för rotationsövergångar är lägre än för både elektron- och vibrationsövergångar.



Figur 2.2: Två elektroniska tillstånd för en diatomisk molekyl. Graferna visar potentiella energin, V(r), som funktion av avståndet, r, mellan kärnorna i den diatomiska molekylen [15].

En molekyls rotationsenergi beror på molekylens rotationskvanttal J [18]. För att studera hur rotationsenergin för en molekyl beror av rotationskvanttalet, studeras en diatomisk molekyl i detalj. Om den diatomiska molekylen består av två atomer A och B med massorna m_A och m_B som har ett konstant bindningsavstånd $r_0 = r_A + r_B$ och roterar runt en punkt C blir tröghetsmomentet runt denna punkt

$$I = \frac{m_1 m_2}{m_1 + m_2} r_0^2 = \mu r_0^2, \tag{2.5}$$

där μ är den reducerade massan [19]. Används nu Schrödingerekvationen, som beskrivs i Avsnitt 2.3.1, för att lösa ekvationen för den diatomiska molekylens rotationsenergier med tröghetsmomentet från Ekvation 2.5, erhålls uttrycket

$$E_J = \frac{\hbar^2}{2I} J(J+1), \tag{2.6}$$

där E_J är rotationsenergin för kvanttalet J och $\hbar$ är Plancks konstant dividerad med 2π. Värdena J får anta är positiva heltal och således är enbart specifika diskreta energinivåer tillåtna. För varje given molekyl finns det därmed unika energinivåer specifika för den molekylen.

Om en elektron exciteras från grundtillståndet till en högre energinivå kommer en del av dess totala energi att komma från rotationsbidraget [15]. När elektronen sedan spontant återgår till ett lägre energitillstånd avges en foton med energin E som motsvarar skillnaden mellan det högre och det lägre energitillståndet. Energin E ges av $E = hf$ där h är Plancks konstant och f betecknar den emitterade fotonens frekvens [20].

2.3.3 Elektromagnetisk strålning

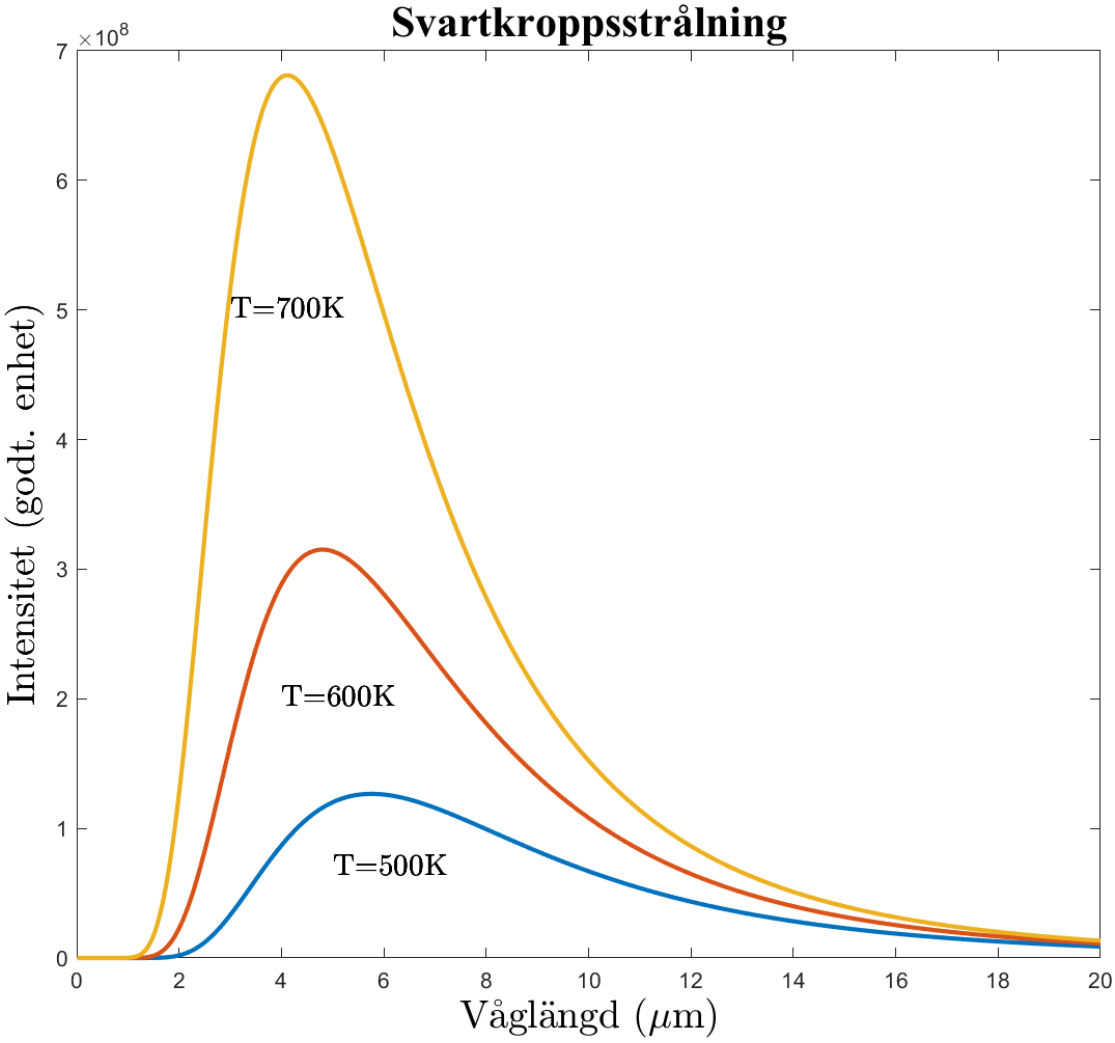
I projektet analyseras olika typer av elektromagnetiska spektrum. För att förstå rapporten krävs därmed en grundläggande förståelse för hur elektromagnetism och spektrum fungerar.

Elektriska fält uppstår kring stationära elektriska laddningar och om dessa laddningar rör på sig uppstår även ett magnetiskt fält [20]. Elektromagnetisk strålning uppkommer genom att det sker regelbundna förändringar i det elektriska och det magnetiska fältet. Denna strålning transporterar energi från en punkt till en annan och propagerar med ljusets hastighet. Energin hos strålningen beror linjärt på dess frekvens. Varje frekvens motsvarar även en specifik våglängd och för elektromagnetisk strålning ges förhållandet mellan dessa av $c = \lambda f$, där λ är våglängden och c är ljusets hastighet i vakuum.

Inom astronomin observeras det så kallade elektromagnetiska spektrumet från olika källor [21]. Hela spektrumet är uppsättningen av samtliga möjliga våglängder hos den elektromagnetiska strålningen. Strålning som människor kan uppfatta, synligt ljus, utgör endast en liten del av detta spektrum. Den absolut största delen av den elektromagnetiska strålningen är osynlig för människor. Olika delar av spektrumet har även olika namn. De vågor med kortast våglängd kallas gammastrålning. I ordning efter ökande våglängd följer sedan röntgenstrålning, ultraviolett ljus, synligt ljus, infrarött ljus, mikrovågor och radiovågor, vilka har längst våglängd.

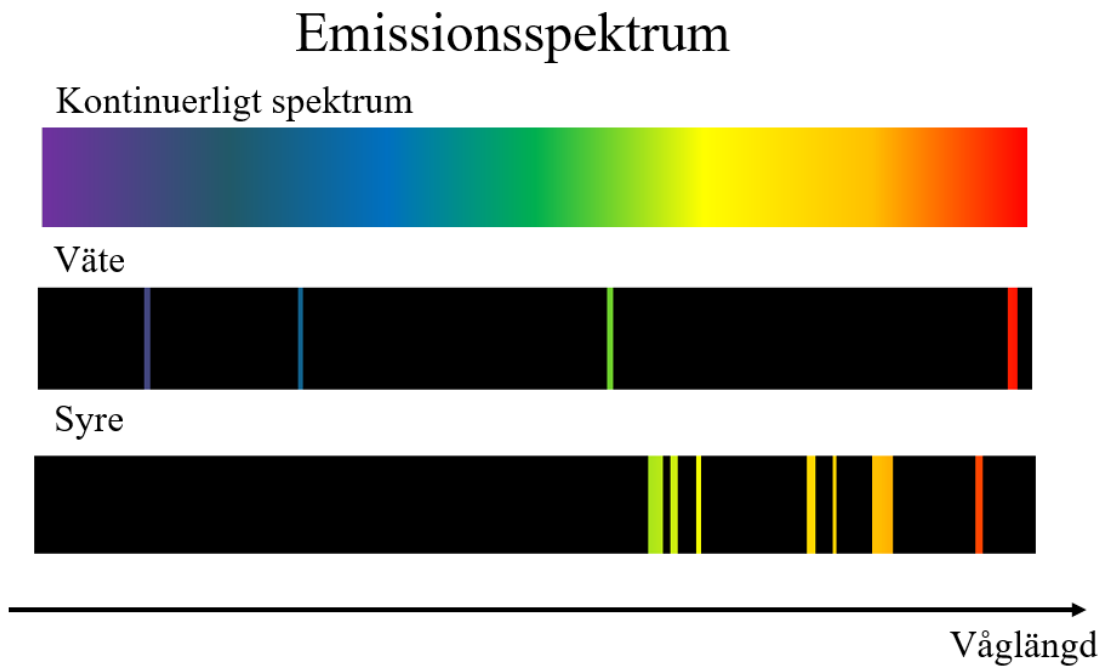
Emissionen från många astronomiska kroppar, inklusive stjärnor, kan approximeras med så kallad svartkroppsstrålning [21]. En svartkropp är ett idealt objekt som absorberar all strålning som träffar det. Objektet återemitterar sedan all energi

som den absorberat. Denna emission följer en karakteristisk fördelning som enbart beror av objektets temperatur. Dessutom beror positionen på intensitetens topp av temperaturen. Om temperaturen ökar, förskjuts toppen mot kortare våglängder och därmed blåare ljus. På samma sätt förskjuts toppen mot längre våglängder och därmed rödare ljus om temperaturen minskar. Detta förklarar varför stjärnor har olika färger. De svala stjärnorna upplevs vara röda medan de varma upplevs vara blåa. Figur 2.3 illustrerar svartkroppsstrålningen för olika temperaturer.



Figur 2.3: En illustration av svartkroppsstrålning för olika temperaturer (Hjält, 2022).

Diskreta emissionslinjer som observeras mot astronomiska objekt, som stjärnbildande områden, uppkommer när fotoner med en specifik våglängd avges då molekyler och atomer vid det astronomiska objektet övergår från ett kvantmekaniskt tillstånd till ett tillstånd med lägre energi [22]. Energin dessa fotoner har kan beräknas med Ekvation 2.6, men då dessa energier bara kan anta vissa specifika diskreta värden, förekommer emission enbart vid vissa specifika våglängder för en viss molekyl enligt Avsnitt 2.3.2. Således uppstår diskreta emissionsspektrum som är unika för varje molekyl då de övergångar som är tillåtna mellan olika energinivåer varierar mellan olika molekyler. Figur 2.4 illustrerar ett exempel på hur emissionsspektrum från olika atomer ser ut i jämförelse med det kontinuerliga spektrumet. Notera att spektrumen i figuren uppvisar det optiska våglängdsintervallet. För rotationsövergångar i molekyler är energiskillnaderna så små att emissionen istället sker i radiovågsintervallet, alltså vid längre våglängder.



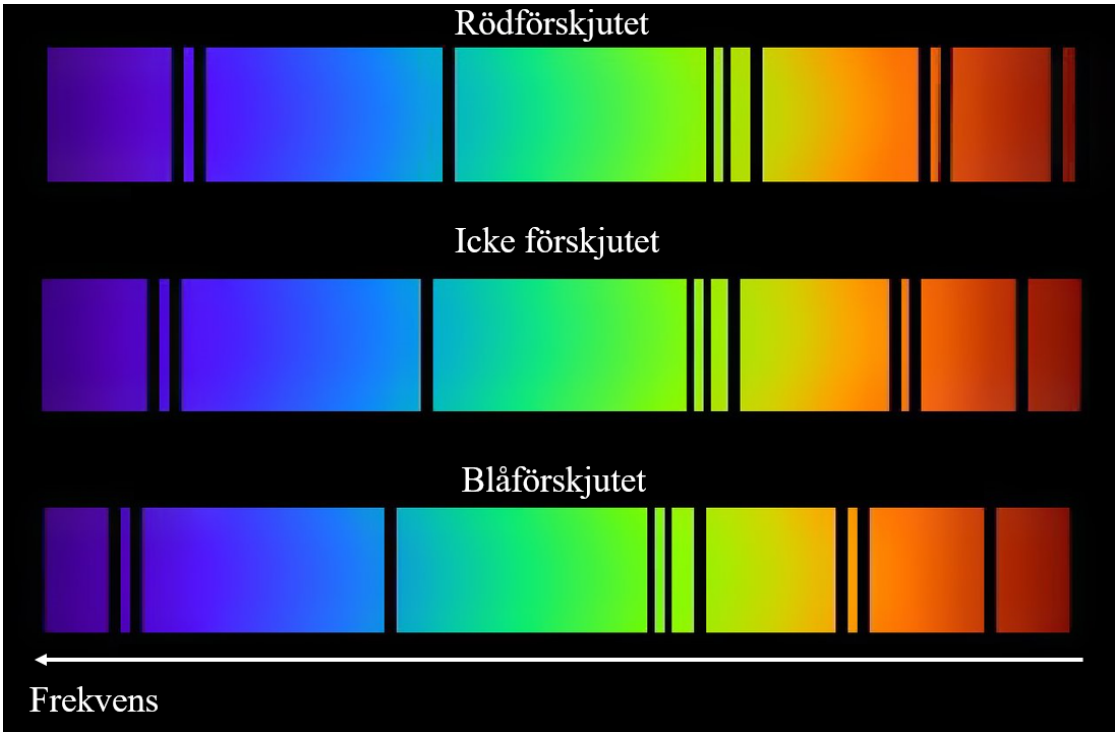
Figur 2.4: Illustration av emissionsspektrum i det optiska våglängdsintervallet. Den övre bilden visar det kontinuerliga spektrumet då ljuset inte emitteras från en specifik atom, medan de undre spektrummen visar emissionsspektrumet för väte respektive syre. För de två atomerna syns de diskreta emissionslinjerna (Hjält, 2022).

2.3.4 Dopplerförskjutning

Eftersom de astronomiska objekten som studeras med ett teleskop kan ha en rörelse i förhållande till teleskopet, kommer den emitterade frekvensen dopplerförskjutas. Formeln för dopplerförsktjutning ges av

$$\frac{f_0}{f} = 1 + \frac{\Delta v}{c}, \tag{2.7}$$

där f är den observerade frekvensen, f_0 är den utsända frekvensen, c är ljusets hastighet och $\Delta v = v_s - v_r$, där v_s och v_r är källans respektive mottagarens hastighet relativt mediet [23]. Notera att Ekvation 2.7 gäller för $v_s \ll c$ och $v_r \ll c$. Vid mycket höga hastigheter behöver relativistiska aspekter tas hänsyn till. Figur 2.5 illustrerar hur absorptionslinjer röd- eller blåförskjuts, på grund av dopplereffekten, om den observerade källan rör sig ifrån eller mot observatören.



Figur 2.5: En illustration av hur absorptionslinjer kan röd- och blåförskjutas i ett spektrum på grund av dopplereffekten (ESO).

2.4 ALMA-teleskopet

För att kunna observera molekylära utflöden behövs högupplösta observationer på långa våglängder av mycket avlägsna objekt. För detta krävs väldigt kraftfulla teleskop. Ett sådant teleskop, och det som används i detta projekt, är ALMA-teleskopet i Chile. ALMA är världens största radioteleskop och har använts för observationer sedan 2011, men invigdes först 2013 då konstruktionen var klar [24]. Det ligger i Atacamaöknen i Chile och utgör ett samarbete mellan många olika länder över hela världen [25]. Det övergripande syftet med teleskopet är att studera stjärnbildning, molekylära moln och det tidiga universum [5]. Teleskopet utför observationer i våglängdsintervallet 0,32-3,6 mm vilket motsvarar frekvenser på 31-1000 GHz [26]. Hela teleskopet består av 66 stycken individuella antenner. Av dessa har 54 antenner en diameter på 12 m och de övriga 12 har en diameter på 7 m. Antennerna är utspridda med avstånd från 150 m upp till 16 km mellan de antenner som är längst ifrån varandra [27].

Genom att använda interferometri kan observationer från de olika antennerna kombineras och därmed emulera ett större teleskop [27]. Den effektiva diametern på det större teleskopet kommer då motsvara det längsta avståndet mellan två antenner. Genom denna metod blir ALMA-teleskopets upplösning mycket god. De individuella antennerna kan även placeras på olika avstånd och i olika konfigurationer för att erhålla olika vinkelupplösningar och därmed ändra hur magnifierad observationen blir [5]. Vinkelupplösningen θ ges av

$$\theta = 1,22\frac{\lambda}{D}, \tag{2.8}$$

där λ är den observerade strålningens våglängd och D är teleskopets effektiva diameter [28]. Om den maximala effektiva diametern på ALMA-teleskopet används, det vill säga 16 km, blir vinkelupplösningen 0,004 bågsekunder enligt Ekvation 2.8 [29]. Med denna goda upplösning kan objekt som är mycket långt bort studeras. Med ALMA studeras bland annat hur de första galaxerna och stjärnorna bildades för miljarder år sedan [30]. Dessutom studeras processerna kring stjärn-, planet- och galaxbildning. Den komplexa kemin i gas- och stoftmolnen som styr bildningsprocesserna kan bättre förstås med ALMAs observationer.

När en uppsättning antenner ska kombineras till ett större teleskop genom interferometri så uppkommer ett problem. Strålning som lämnade det observerade objektet vid samma tidpunkt kommer träffa de olika antennerna vid olika tidpunkter [31]. Om signalen från de olika detektorerna kombineras rakt av, kommer det bli en röra av flera signaler vid olika tidpunkter. För att undvika detta används interferometri. När antennerna observerar objektet, noteras även tidpunkten för

observationen mycket noggrant. På detta sätt genereras, för varje antenn, en ström av data med unika tidsstämplar. Med hjälp av tidsstämplarna kan datan korreleras när den ska sammanställas från de olika antennerna. Matematiken för denna korrelering är dock komplicerad. För att interferometrin ska fungera behöver tidsskillnaderna mellan varje par av antenner vara kända. För ALMA, som har 66 antenner, blir det 2 145 par [31]. Något som komplicerar problemet ytterligare är att jorden roterar medan observationen genomförs, vilket ändrar tidsskillnaderna mellan paren av antenner. För dessa beräkningar används en superdator med stor beräkningskraft.

Figur 2.6: Några av ALMA-teleskopets 66 antenner (ESO, 2013).

Den superdator som används för att korrelera data från antennerna är *"The ALMA Correlator"*. Denna syftesspecifika superdator är en av de snabbaste superdatorerna i världen. Den kan utföra 17 biljarder (10^{15}) operationer per sekund, motsvarande 17 000 000 GHz, med dess fler än 134 miljoner processorer [32]. Denna extremt höga prestanda krävs för att kunna jämföra och lägga ihop datan från upp till 64 stycken antenner samtidigt. Vid inhämtning, korrelering och sammanställning av datan uppstår vissa oundvikliga felaktigheter [28]. Dessa kan visa sig i datan genom olika så kallade artefakter vilket försämrar kvaliteten på bilderna.

Efter processeringen med The ALMA Correlator lagras datan från observationerna i ALMAs arkiv som är baserat i Santiago, Chile och i ALMAs ARC:s *(ALMA Regional Centers)* i Europa, Östasien och Nordamerika [8]. Dessa ARC:s används sedan för att lagra datan och när forskare vill komma åt den finns den att tillgå via respektive ARC:s hemsida.

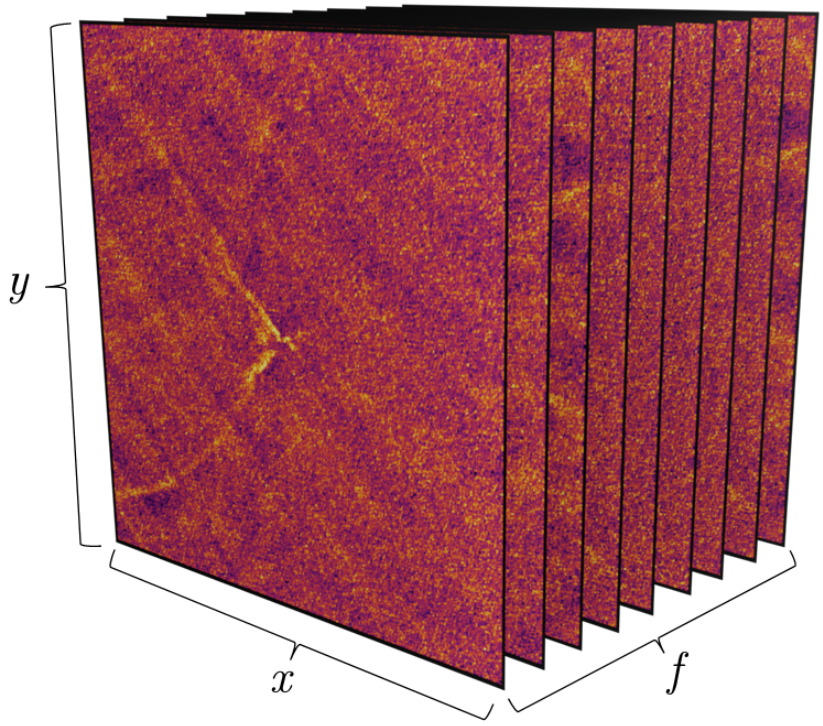
Eftersom observationstid hos ALMA är mycket efterfrågat, finns system för att tilldela den. En tiondel av observationstiden är reserverad för värdlandet, Chile [8]. Resten av tiden tilldelas övriga länder utefter deras monetära bidrag till ALMAs konstruktion. Alla länder avsätter dock en del av observationstiden till "*Open Skies*", vilket möjliggör för vilken astronom som helst att ansöka om observationstid.

En gång per år sker ansökan om observationstid [8]. Astronomer skickar då in ett förslag där de beskriver observationen, vad datan kommer vara användbar till samt hur mycket tid som observationen kräver. Förslagen granskas sedan av andra astronomer som beslutar vilka projekt som är mest intressanta, utifrån deras vetenskapliga värde. För att besluten ska vara så opartiska som möjligt, är förslagen som skickas in anonyma [33]. Då efterfrågan på observationstid är hög godkänns ungefär bara en femtedel av de förslag som skickas in [8]. Forskare från de utvalda projekten behöver inte själva resa till Chile för att utföra observationerna, utan de genomförs av anställda astronomer vid ALMA-teleskopet. Efter utförd observation kommer den eller de forskare som föreslog observationen att få exklusiv tillgång till datan under ett års tid. Därefter blir datan publik och finns tillgänglig på ALMAs öppna arkiv.

2.4.1 Dataprodukter från ALMA

Från ALMAs öppna arkiv kan många olika typer av dataprodukter hämtas, både rådata och olika former av kalibrerad och bearbetad data [34]. I detta projekt hanteras inte rådata utan endast bearbetad data i form av `.fits`-filer.

FITS (*Flexible Image Transport System*) är ett filformat som används inom astronomi. Filformatet hanterar både metadata om observationen och den producerade flerdimensionella datan [35]. Observationer inom astronomi görs över ett intervall av frekvenser, där bilder (data) inhämtas för varje frekvens. Detta resulterar i en så kallad datakub, illustrerad i Figur 2.7, med information, där de två första dimensionerna representerar spatiala koordinater och den tredje dimensionen representerar frekvens.



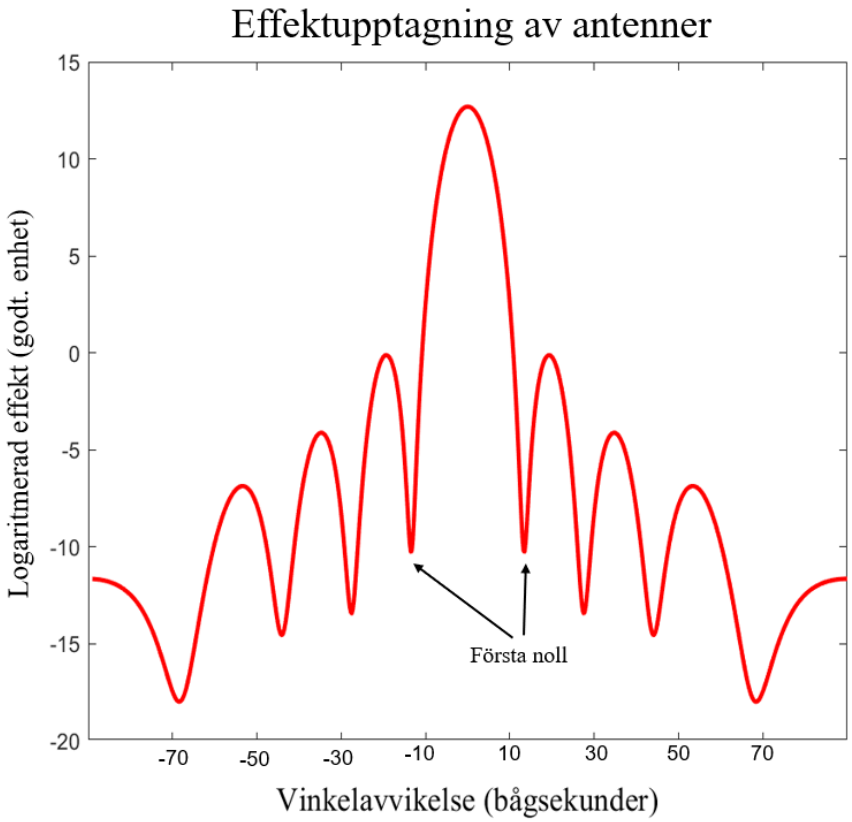
Figur 2.7: Visualisering av en datakub med emission från rotationsövergången ($2 \rightarrow 1$) för molekylen ^{12}CO från en observation av protostjärnan B335. x och y representerar spatiala axlar, medan f representerar frekvensen för varje skiva av kuben (Rosén, 2022).

Detta projekt hanterar två olika typer av data. Datakuber, som representerar molekylers emission, och kontinuumdata, vilket bland annat beskriver stoftfördelning. Denna kontinuumdata produceras av ALMA genom att observera över ett frekvensområde där emission inte förkommer, och sedan kollapsa datakuben genom en momentavbildning över hela frekvensområdet enligt Avsnitt 2.5 [34]. Att observera över ett frekvensområde utan emission är inte alltid trivialt och det kan kräva efterarbete att plocka bort frekvenser med emission för att få en klarare bild av stoftfördelningen.

2.4.1.1 Primary Beam Correction

Ett teleskop har inte en uniform upptagning över hela himlen. Det innebär att teleskopet fångar upp olika mycket strålning för olika vinklar. Figur 2.8 visar ett exempel på hur en endimensionell antenns effektupptagning varierar med vinkeln.

Effektupptagningen är högst i antennens riktning och avtar mot första noll (~14 bågsekunder i exemplet från Figur 2.8) enligt Ekvation 2.8. Den centrala parabeln kallas *primary beam* och står för majoriteten av all upptagning [28].



Figur 2.8: Effektupptagning för en idealiserad antenn, likformigt upplyst vid 150 GHz (Hjält, 2022).

Som tidigare nämnt använder ALMA flera antenner simultant för att utföra observationer. Datan från varje antenn sammanställs sedan genom interferometri [27]. Den data som ALMA producerar normalt genom interferometrin är en representation av himlen multiplicerat med *primary beam*-upptagningen av antennerna [28]. Denna data är användbar då den har en uniform brusnivå. För att få en astronomiskt korrekt bild av himlen måste dock denna *primary beam*-upptagning korrigeras för [28]. ALMA korrigerar för denna upptagning i dataprodukter benämnda `.pbcor` och detta projekt studerar enbart dessa korrigerade dataprodukter.

2.5 Momentavbildningar

Som beskrivet i Avsnitt 2.4.1 sparas observationsdata från ALMA som datakuber. Dessa datakuber kan ses som en samling tvådimensionella bilder där varje bild består av den detekterade strålningen vid en specifik frekvens. För att enklare analysera och visualisera dessa tredimensionella bilder, genereras momentavbildningar.

Momentavbildningar, eller integrerade intensitetkartor, är en metod som sammanställer information från datakuber till en tvådimensionell bild. Moment är en statistisk storhet [36], där det n-te momentet kring 0 av en funktion $f(x)$ är

$$M_n = \int_{-\infty}^{\infty} x^n f(x)\, dx.$$

Det 0-te momentet appliceras för att generera integrerade intensitetskartor över ett frekvensområde Ω genom

$$M_0 = \int_{\Omega} I_\nu \, d\nu, \tag{2.9}$$

där ν motsvarar frekvensen och I_ν intensiteten. Att applicera Ekvation 2.9 på en datakub över ett frekvensområde Ω ger en momentavbildning av ordning 0 (hädanefter bara momentavbildning).

2.6 Kurvanpassning

Data insamlad från observationer innehåller brus och artefakter till följd av bland annat begränsad upplösning och felkällor från interferometrin. För att utföra en matematisk analys av datan utnyttjas kurvanpassning. Kurvanpassning innebär att anpassa en kurva, vilken representeras av en funktion med parametrar, till data så att skillnaden mellan kurvan och datapunkterna är minimerad. Detta blir ett optimeringsproblem där parametrarna för den optimala kurvan sökes.

En av de vanligaste metoderna för att hitta dessa parametrar är *minsta kvadratmetoden* [37]. I minsta kvadratmetoden är kostnadsfunktionen summan av de kvadrerade avvikelserna i y-led från varje datapunkt till kurvan. Alltså är problemet att minimera

$$C = \sum_{i=1}^{N} \left(f(x_i) - y_i\right)^2,$$

där (x_i, y_i) är datapunkter och $f(x) = c_1 f_1(x) + \cdots + c_m f_m$ är funktionen som beskriver kurvan och där vektorn $\mathbf{c} = [c_1 \ldots c_m]^T$ är funktionens parametrar. Detta

problem kan lösas analytiskt genom att studera lösningen till normalekvationen

$$A^T A\mathbf{c} = A^T\mathbf{y},$$

där A är designmatrisen

$$A = \begin{bmatrix} f_1(x_1) & \cdots & f_m(x_1) \\ \vdots & \ddots & \vdots \\ f_1(x_n) & \cdots & f_m(x_n) \end{bmatrix},$$

och $\mathbf{y} = [y_1 \dots y_n]^T$.

Vilken funktion som anpassas beror på problemet och kräver en analys, både kring vilka typer av funktioner (polynom, exponentiella funktioner etc.) och hur detaljerade (antal parametrar, exempelvis grad på polynom). Anpassas en funktion med för många parametrar finns risk för överanpassning, det vill säga att kurvan blir komplex och opålitlig [38]. Anpassas istället en funktion med för få parametrar finns istället risk för underanpassning, att kurvan blir för simpel och missar det väsentliga.

2.6.1 Gauss-funktioner

Många fysiska system går att beskriva med hjälp av normalfördelningar [39]. En skalad *Gauss-funktion* beskriver täthetsfunktionen för en normalfördelad slumpvariabel [40]. Därmed är Gauss-funktionen en användbar funktion att anpassa till data från fysikaliska observationer. Den generella endimensionella Gauss-funktionen kan uttryckas

$$f(x) = a\exp\left(-\frac{(x-b)^2}{2c^2}\right), \tag{2.10}$$

för de reella parametrarna a, b och c.

Gauss-funktionen kan även uttryckas i två dimensioner där nivåkurvorna tar formen av ellipser. Denna beskrivs av,

$$f(x,y) = A\exp\left(-\left(a(x-x_0)^2 + 2b(x-x_0)(y-y_0) + c(y-y_0)^2\right)\right), \tag{2.11}$$

där A är amplituden och (x_0, y_0) är koordinaten för funktionens maximum, alltså

mittpunkten [41]. Sätts parametrarna a, b och c som

$$
\begin{aligned}
a &= \frac{\cos^2\theta}{2\sigma_X^2} + \frac{\sin^2\theta}{2\sigma_Y^2}, \\
b &= -\frac{\sin 2\theta}{4\sigma_X^2} + \frac{\sin 2\theta}{4\sigma_Y^2}, \\
c &= \frac{\sin^2\theta}{2\sigma_X^2} + \frac{\cos^2\theta}{2\sigma_Y^2},
\end{aligned}
$$

kan även utbredningen och rotationen av funktionen beskrivas genom σ_X och σ_Y respektive θ.

3
Metod

Nedan följer beskrivningar av metoder som används i projektet. Dessa är till största del redskap och tillvägagångssätt för att hämta, bearbeta och analysera datan från ALMA-arkivet. För koden till de metoder som utvecklats, se Bilaga C, Bilaga D och Bilaga E.

3.1 Filtrering och nedladdning av observationer

Hela ALMA-arkivet består av ~56 000 observationer och en stor mängd lagrad data som årligen ökar med ~500 TB [34]. En del av dessa observationer innehåller protoplanetära skivor med molekylära utflöden. Att analysera alla observationer för att hitta de med molekylära utflöden är både tids- och lagringsineffektivt. Dock innehåller alla observationer en mängd information, exempelvis nyckelord, som kan användas för att sortera fram de observationer som potentiellt innehåller utflöden.

För att hämta och filtrera observationer används *ALminer*, ett Python-baserat bibliotek för att hämta, visualisera och analysera data från ALMA-arkivet [42]. Det första steget i filtreringen är att genom funktionen `alminer.keysearch()` filtrera på de nyckelord som primärt har med molekylära utflöden och protostjärnor att göra. De utvalda nyckelorden är:

- Outflows, jets and ionized winds.
- Low-mass star formation.
- Intermediate-mass star formation.
- High-mass star formation.
- Disks around low-mass stars.
- Disks around high-mass stars.

Eftersom ALMA införde en ny konvention för filnamngivning i slutet av 2015, filtreras observationer som gjordes innan dess bort [43]. Detta förenklar upptäckandet av filer som är viktiga vid analys. För att öka sannolikheten att erhålla detaljrik

data för analys filtreras observationer med avseende på vinkelupplösning. En observations vinkelupplösning beskriver den minsta vinkeln mellan två särskiljbara objekt, och är alltså ett mått på förmågan att särskilja detaljer. Observationer som har en sämre vinkelupplösning än 0,4 bågsekunder filtreras bort eftersom detaljer riskeras att förloras vid sämre vinkelupplösning.

Observationerna filtreras även utifrån vilka frekvensintervall som observeras, detta för att erhålla observationer innehållande specifika molekylära rotationsövergångar. Frekvenser för rotationsövergångar av molekyler som är vanligt förekommande i observationer av utflöden valdes ut och presenteras i Tabell A.1 i Bilaga A.

Till sist, i samband med att observationerna laddas ner, filtreras de även efter filnamn. Detta genomförs med *ALminer*-funktionen `alminer.download_data()` som laddar ned observationsdata. Funktionen kan filtrera bort data vars filnamn inte innehåller vissa delsträngar. Övergripande krävs att filerna är `.fits`-filer, som beskrivet i Avsnitt 2.4.1, samt att de innehåller `_sci` vilket innebär att observationen inte är gjord i kalibreringssyfte [44]. I första delen av analysen filtreras kontinuumfiler fram, vilket säkerställs genom att filnamnet innehåller `.cont`. Identifieras en protoplanetär skiva i kontinuumfilen, som beskrivet i Avsnitt 3.2.1, filtreras även datakuber fram genom att kräva `.cube`.

Alla dessa filtreringar leder till beslutsträdet som presenteras i Figur 4.1 i Avsnitt 4.1.

3.2 Generera momentavbildningar

Från ALMA-arkivet hämtas, som beskrivet i Avsnitt 2.4.1, två olika typer av data, tvådimensionell kontinuumdata och tredimensionella datakuber. För att enklare analysera och visualisera de tredimensionella bilderna, genereras momentavbildningar beskrivna i Avsnitt 2.5.

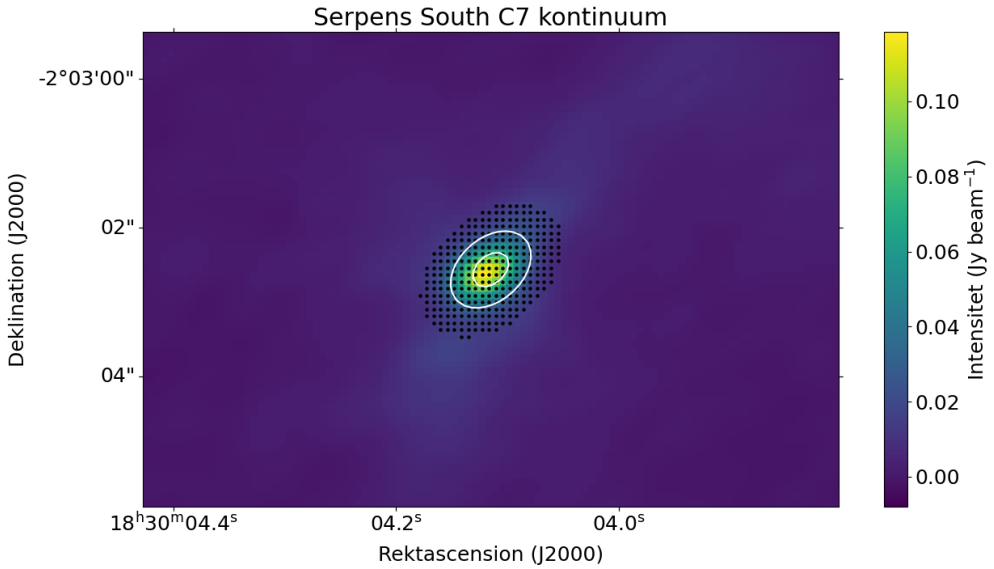
Varje datakub spänner över ett omfång av frekvenser. För att utvinna värdefull information om molekylemissionen krävs att frekvensområdena, Ω i Ekvation 2.9, väljs med omsorg. På grund av att de observerade objekten har en relativ rörelse gentemot teleskopet, dopplerförskjuts den emitterade strålningen enligt Ekvation 2.7. Denna dopplerförskjutning ger upphov till att molekylernas rotationsövergångar inträffar vid olika observerade frekvenser beroende på den relativa hastigheten. Genom att undersöka frekvensintervallen för de röd- respektive blåförskjutna frekvenserna separat framhävs respektive emission, relativt brus minskas och mer information kan utvinnas.

Utöver uppdelningen i blåförskjutna och rödförskjutna frekvensintervall förväntas emissionen från en specifik rotationsövergång vara samlad i ett visst område av det observerade frekvensintervallet. De slutliga bilderna förbättras ytterligare genom att begränsa intervallen som används i Ekvation 2.9 till dessa delar av frekvensintervallet. För att hitta dessa intervall analyseras datakubernas frekvensprofil, vilket är en graf över den genomsnittliga intensiteten för varje frekvens i ett visst område. Detta illustreras i Figur 3.2. För att studera bipolära utflöden är detta område lämpligtvis en viss area kring skivan eftersom utflödena strömmar från skivan. Genom att använda ett område som innesluter skivan kan eventuell emission detekteras i alla riktningar, därmed kommer både blå- och rödförskjuten emission återfinnas i området.

Frekvensprofiler runt skivans centrum från data med utflöden tenderar att uppvisa en viss kännetecknande form, med två lokala maximum skiljt av ett minimum. De två maximumen svarar mot intensiteten från rotationsövergången för den röd- respektive blåförskjutna emissionen medan minimumet motsvarar absorption där emission har samma centrala hastighet som objektet, relativt teleskopet. Genom att studera grafens inflektionspunkter kan lämpliga frekvensområden för momentavbildningar utvinnas. Data med frekvensprofiler som inte är på denna form, som till exempel datan presenterad i Figur 3.3a, förkastas.

3.2.1 Identifiering och lokalisering av protoplanetära skivor

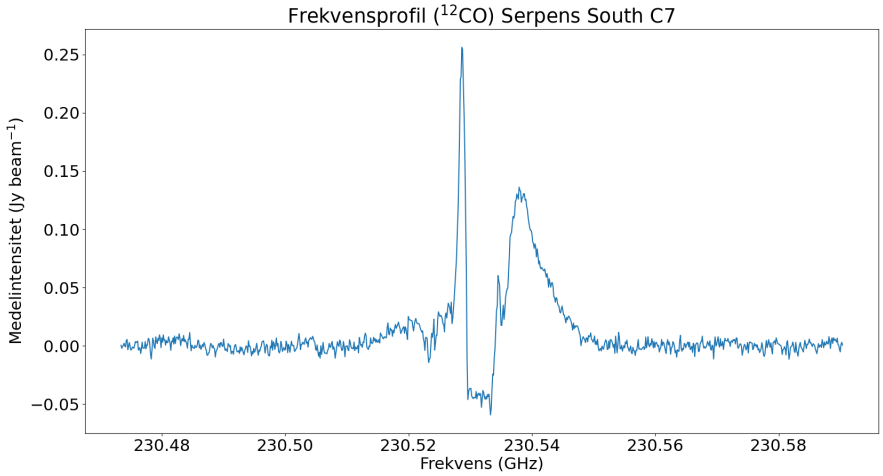
De relevanta utflödena för detta projekt är alltid associerade med protoplanetära skivor och skivans position är vital vid framtagandet av frekvensprofiler. Därmed är det första steget i analysen att identifiera och lokalisera den eventuella protoplanetära skivans spatiala position och utbredning. Den protoplanetära skivan är svår att lokalisera i datakuberna men i den tvådimensionella kontinuumdatan är skivan mer prominent. För att beskriva skivans position och utbredning anpassas en tvådimensionell Gauss-funktion till kontinuumdatan enligt Ekvation 2.11. Denna anpassning görs numeriskt med hjälp av Python-paketet *SciPy* [45]. Metoden `scipy.optimize.curve_fit()` appliceras, vilken anpassar Ekvation 2.11 till stoftdatan genom en icke-linjär minstakvadratmetod. Identifieras ingen skiva förkastas observationen i analysen. Figur 3.1 visar resultatet av denna anpassning för en observation av Klass 0-källan CARMA-7 (C7) i stjärnklustret Serpens South.



Figur 3.1: Kontinuumdata från Klass 0-källan CARMA-7 (C7) i stjärnklustret Serpens South. De vita konturerna motsvarar nivåkurvor till den anpassade Gauss-funktionen. De svarta prickarna motsvarar de spatiala koordinater som utgör området från vilket datakubernas frekvensprofil tas fram.

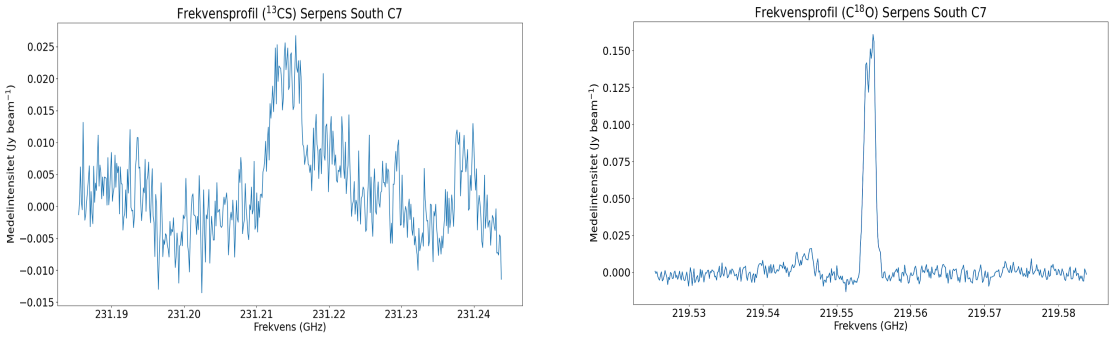
3.2.2 Analys av frekvensprofiler

Från den anpassade Gauss-funktionen erhålls skivans spatiala position, utbredning och vinkel. Med den informationen bestäms ett område från vilket datakubernas frekvensprofil tas fram. Detta genom att välja ut alla koordinater där värdet på den anpassade Gauss-funktionen är högre än 10% av amplituden. Medelintensiteten i detta område beräknas för varje frekvens vilket ger en frekvensprofil. I Figur 3.2 visas den genererade frekvensprofilen för rotationsövergången ($2 \rightarrow 1$) för molekylen ^{12}CO från en observation av Klass 0-källan CARMA-7 (C7) i stjärnklustret Serpens South. Den tidigare nämnda formen med två lokala maxima skiljt av ett minima framgår tydligt, vilket starkt tyder på ett utflöde.



Figur 3.2: Frekvensprofil för rotationsövergången $(2 \rightarrow 1)$ för molekylen ^{12}CO kring Klass 0-källan CARMA-7 (C7) i stjärnklustret Serpens South.

Utseendet på frekvensprofilen varierar från datakub till datakub, beroende på till exempel utflödets intensitet, utbredning och existens. Figur 3.3a och Figur 3.3b visar två andra frekvensprofiler genererade från andra molekyler i samma observation. Figur 3.3a visar $(5 \rightarrow 4)$-övergången för ^{13}CS. Denna profil uppvisar inte den beskrivna karaktäristiska formen som tydligt kännetecknar utflöden. För $(2 \rightarrow 1)$-övergången av C^{18}O i Figur 3.3b är det svårare att avgöra och ytterligare analys krävs.



(a) **(b)**

Figur 3.3: I (a) visas frekvensprofilen för rotationsövergången $(5 \rightarrow 4)$ i ^{13}CS. I (b) visas frekvensprofilen för övergången $(2 \rightarrow 1)$ i C^{18}O. Båda frekvensprofilerna är för Klass 0-källan CARMA-7 (C7) i stjärnklustret Serpens South.

När frekvensprofilerna är framtagna klassificeras och analyseras dessa. Målet är att filtrera bort de fall som inte uppvisar den karaktäristiska formen, samt att från den karaktäristiska formen identifiera de relevanta områdena Ω att generera momentavbildningar över.

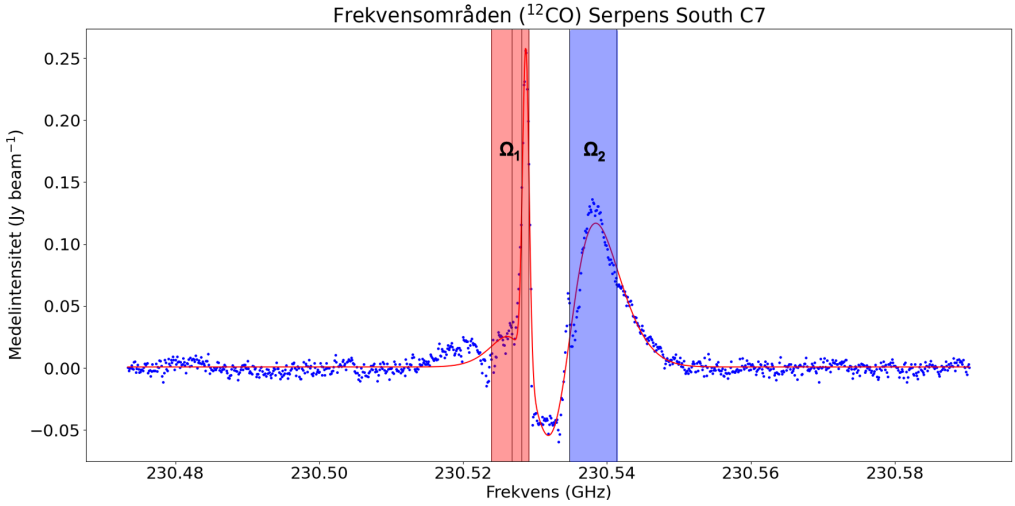
Ett första steg i denna klassifikation är att beräkna det kvadratiska medelvärdet (*root mean square*) över frekvensprofilen, vilket ges av

$$\nu_{\text{rms}} = \frac{1}{\sqrt{n}}\sqrt{\nu_1^2 + \nu_2^2 + \cdots + \nu_n^2}. \tag{3.1}$$

Det kvadratiska medelvärdet appliceras för att identifiera profiler utan tydlig emission, genom att studera andelen intensiteter större än $3\nu_{\text{rms}}$. Finns det inga eller få intensiteter större än $3\nu_{\text{rms}}$, finns troligen ingen emission.

Nästa steg är att försöka anpassa endimensionella Gauss-kurvor till profilen. Den karaktäristiska formen kan representeras som en summa av två eller tre Gauss-kurvor. Identifieras ett minimum med negativ intensitet, som i Figur 3.2, anpassas tre Gauss-kurvor, varav två med positiv amplitud, a i Ekvation 2.10, och en med negativ. Identifieras istället ett minimum med positiv intensitet, som i Figur 3.3b, anpassas två Gauss-kurvor med positiv amplitud. *SciPy*-metoden `scipy.optimize.curve_fit()` används sedan för att numeriskt anpassa summan av dessa Gauss-funktioner till frekvensprofilen [45]. Anpassningen är känslig för vilka startvärden optimeringen utgår ifrån och en metod som anpassar en enstaka Gauss-kurva används för att generera lämpliga startvärden. Lyckas inte metoden anpassa Gauss-funktionerna till datan eller om den genomsnittliga kvadratiska avvikelsen är för stor, anses den karaktäristiska formen ej vara uppfylld.

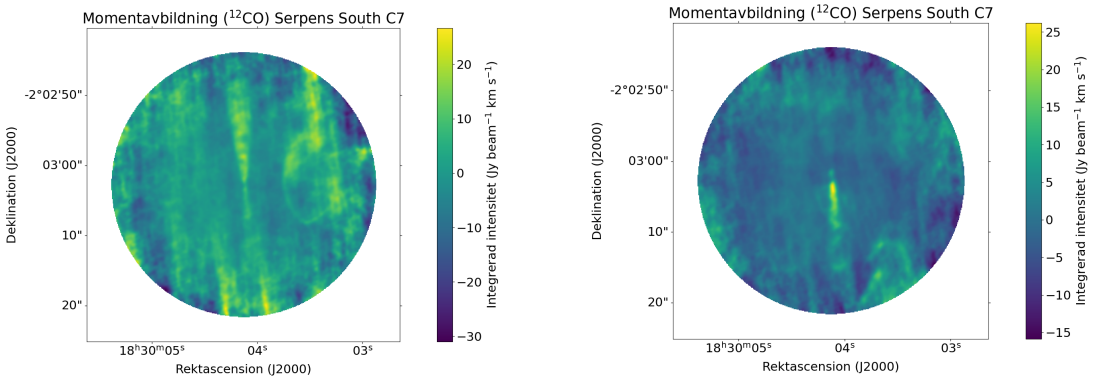
Efter att frekvensprofiler utan den karaktäristiska formen har förkastats återstår att ta fram relevanta områden Ω att generera momentavbildningar över. Genom att analysera de anpassade Gauss-kurvornas inflektionspunkter, det vill säga punkterna där andraderivatan byter tecken, kan dessa områden hittas. Med hjälp av dessa kan profilen delas upp i relevanta områden. Områdena delas upp genom att mellan de två största maximumen hitta det lägsta minimat, vilket för den karaktäristiska formen utgör absorptionen och brytpunkten mellan röd- och blåförskjuten emission. Intervallen blir sedan, för vardera sida, området mellan de inflektionspunkter belägna vid lägst respektive högst frekvens. Detta illustreras i Figur 3.4. Intervallen som framtages med denna metod är inte nödvändigtvis optimala, till exempel saknas i Figur 3.4 en del av den blåförskjutna emissionen med högre frekvens. De genererade intervallen ger dock en bra utgångspunkt för analys av utflödets morfologi.



Figur 3.4: Anpassad Gauss-funktion med utritade inflektionspunkter. De blåa punkterna är frekvensprofilen för rotationsövergången ($2 \rightarrow 1$) för molekylen ^{12}CO kring Klass 0-källan CARMA-7 (C7) i stjärnklustret Serpens South. Den röda kurvan är den anpassade Gauss-funktionen och de svarta linjerna markerar inflektionspunkter. Ω_1 och Ω_2 representerar de framtagna områdena för röd- respektive blåförskjutna frekvenser.

3.2.3 Generera momentavbildningar

Med de områden som tas fram enligt Avsnitt 3.2.2 kan intensiteten integreras upp och momentavbildningar genereras enligt Ekvation 2.9. Figur 3.5a visar momentavbildningen över det rödförskjutna frekvensområdet Ω_1 från Figur 3.4 medan Figur 3.5b visar momentavbildningen över det blåförskjutna frekvensområdet Ω_2 från samma figur. Figurerna visar ett tydligt utflöde av molekylen ^{12}CO från den protoplanetära skivan.



(a) **(b)**

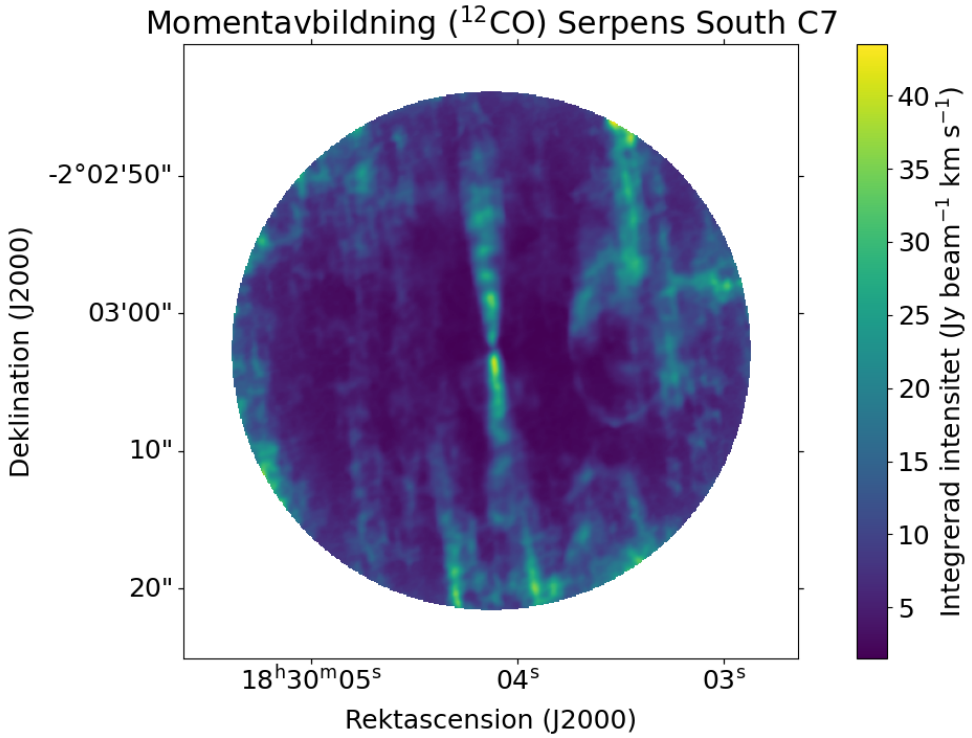
Figur 3.5: I (a) visas momentavbildningen för den rödförskjutna emissionen och i (b) visas momentavbildningen för den blåförskjutna emissionen från rotationsövergången (2 → 1) för molekylen ^{12}CO kring Klass 0-källan CARMA-7 (C7) i stjärnklustret Serpens South.

3.2.4 Alternativa metoder för att generera momentavbildningar

Den metoden för att generera momentavbildningar som beskrivs i Avsnitt 3.2 är inte kapabel att hitta alla utflöden, på grund av de gjorda antagandena. Kurvanpassningar kan misslyckas även när emission är förekommande, bland annat på grund av data med avvikande form och instabilitet till följd av dåliga initialvärden. Med detta som grund har två andra alternativa metoder för att generera momentavbildningar utforskats.

3.2.4.1 Maximummetoden

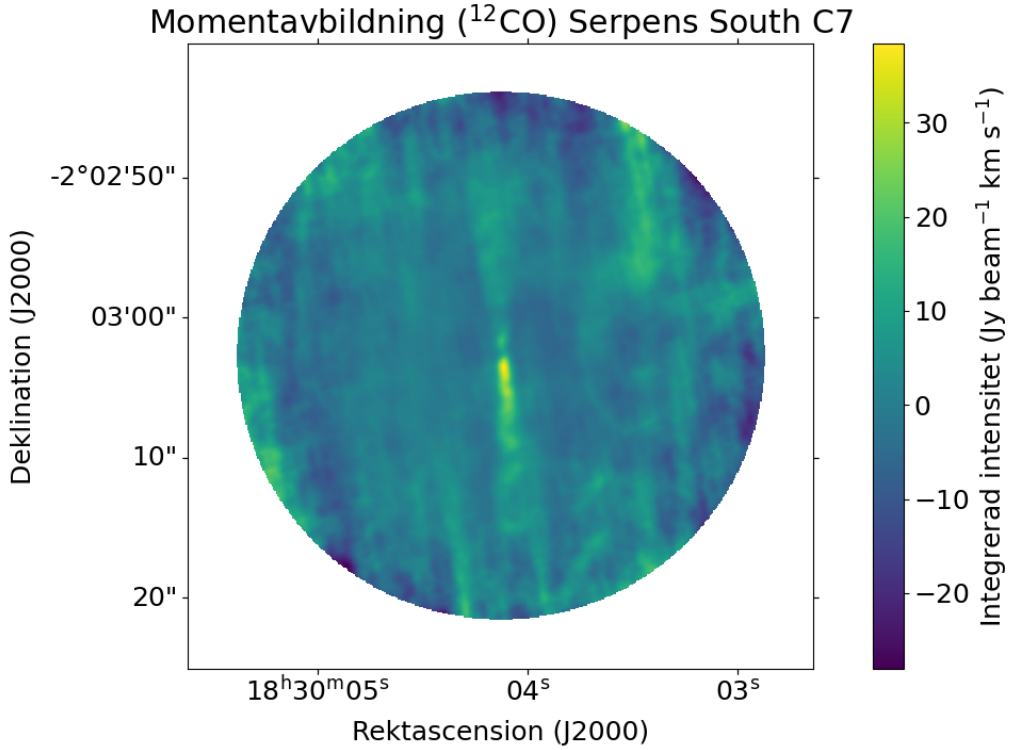
Målet med den ursprungliga metoden är att separera ut frekvenser där emission förekommer, det vill säga frekvenser med högst intensitet i utflödet. Ett alternativt tillvägagångssätt är att istället, för varje koordinat (pixel), summera ihop de till exempel 10% högsta intensiteterna. Detta under antagandet att intensiteterna från utflödet är högre än brusnivån. Tekniskt sett är detta inte en momentavbildning, då det integreras över olika frekvensområden för olika koordinater, men det fyller samma syfte för projektets ändamål. En begränsning med denna metod är att man inte separerar de blå- och rödförskjutna frekvenserna, vilket kan vara viktig information i en analys av momentavbildningar. Figur 3.6 visar den resulterande momentavbildningen från maximummetoden på rotationsövergången (2 → 1) för molekylen ^{12}CO kring Klass 0-källan CARMA-7 (C7) i stjärnklustret Serpens South.



Figur 3.6: Momentavbildning med maximummetoden för rotationsövergången $(2 \rightarrow 1)$ för molekylen ^{12}CO kring Klass 0-källan CARMA-7 (C7) i stjärnklustret Serpens South.

3.2.4.2 RMS-metoden

RMS-metoden syftar att använda den analys av kvadratiska medelvärden beskrivet i Ekvation 3.1. Istället för att använda de kvadratiska medelvärdena som ett sätt att filtrera bort data utan den karaktäristiska formen används de här för att direkt välja ut frekvenser till momentavbildningen. Detta genom att generera en momentavbildning över de frekvenser som har en högre intensitet än $C\nu_{\text{rms}}$, med lämplig konstant C. Denna metod har samma begränsning som maximummetoden i att den inte separerar röd- och blåförskjutna frekvenser. Figur 3.7 visar den resulterande momentavbildningen från RMS-metoden på rotationsövergången $(2 \rightarrow 1)$ för molekylen ^{12}CO kring Klass 0-källan CARMA-7 (C7) i stjärnklustret Serpens South.



Figur 3.7: Momentavbildning med RMS-metoden för rotationsövergången (2 → 1) för molekylen ^{12}CO kring Klass 0-källan CARMA-7 (C7) i stjärnklustret Serpens South.

3.3 Identifiering av utflöden

För identifiering och karakterisering av utflöden från de momentavbildningar som genererats har två primära metoder utforskats. Dessa är:

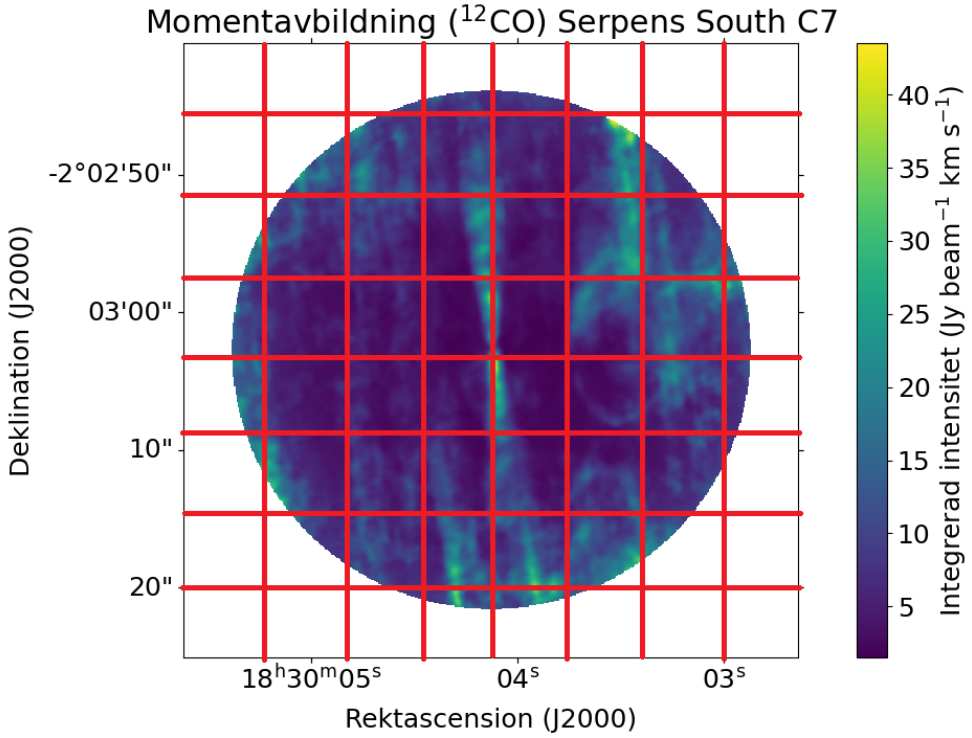
1. Identifiering av utflödets vinkel via konturkartor.
2. Identifiering av utflödets vinkel via summerad intensitet i ett vinkelomfång.

3.3.1 Beräkning av brusnivå

Något som ligger till grund för båda metoderna som används för identifiering av utflödesvinklar är ett värde på brusnivån σ. Eftersom detta inte är något som explicit anges i datan utvecklades en egen metod för att approximera denna. I detta projekt används enbart *primary beam*-korrigerad data, se Avsnitt 2.4.1.1, som inte har en spatialt uniform brusnivå. Att brusnivån inte är uniform leder till ökad komplexitet och bestämmandet av brusets fördelning blir inte trivialt. Istället be-

räknas en undre gräns på brusnivån och den icke-uniforma fördelningen hanteras från fall till fall.

Den metod som använts inom detta projekt baseras på att dela upp momentavbildningen i ett rutnät och beräkna det kvadratiska medelvärdet av intensiteten för varje ruta varav det lägsta av dessa används som brusnivån σ. För att bestämma hur rutnätet ser ut tar funktionen en parameter `partitions` som anger till vilken grad indelningen ska ske, exempelvis ger ett `partitions`-värde på 8 ett rutnät som är 8 rutor brett, och 8 rutor högt, se Figur 3.8. Notera att 8 var ett godtyckligt exempel, oftast har ett högre `Partitions`-värde använts.

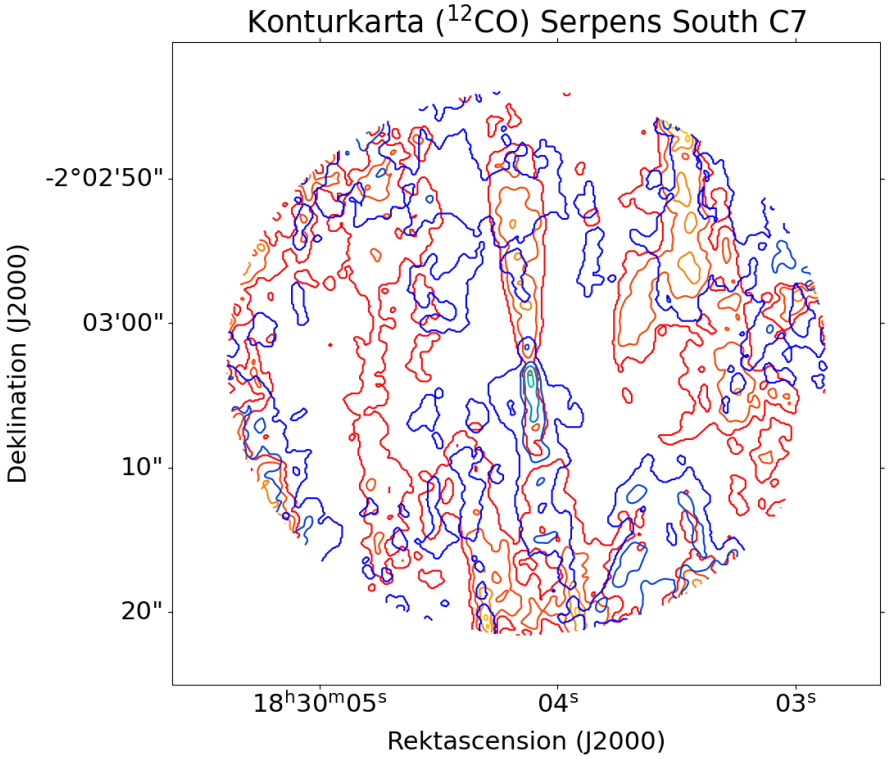


Figur 3.8: Momentavbildning från Figur 3.6 uppdelad i ett rutnät för brusberäkning med `partitions=8`.

3.3.2 Identifiering av utflödets vinkel via konturkartor

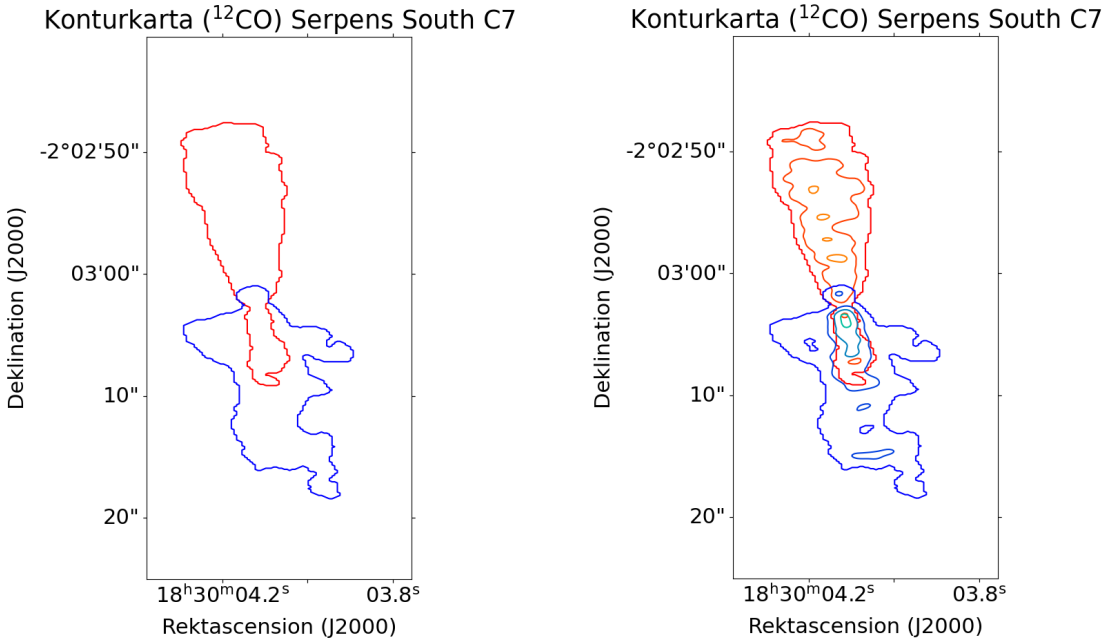
Inom astronomi används ofta konturkartor för att representera momentavbildningar. Konturer är i detta sammanhang nivåkurvor där alla punkter på kurvan har samma intensitet. Dessa ritas upp för några olika intensiteter, oftast några olika multiplar av brusnivån σ.

Brusnivån σ beräknas med metoden beskriven i Avsnitt 3.3.1. Konturkartor genereras sedan genom *Matplotlib*-funktionen `matplotlib.pyplot.contour()` för olika multiplar $C = [c_1, c_2, \ldots, c_n]$ av σ [46]. Dessa konturer innefattar inte nödvändigtvis bara utflöden utan kan även hamna runt brus och artefakter, se Figur 3.9, något som blir extra märkbart eftersom brusnivån inte är uniform. Därför bearbetas konturerna för att minska påverkan av brus och artefakter.



Figur 3.9: Sammanslagen konturkarta av momentavbildningarna från Figur 3.5. De blåaktiga konturerna motsvarar bidrag från den blåförskjutna emissionen och de rödaktiga från den rödförskjutna.

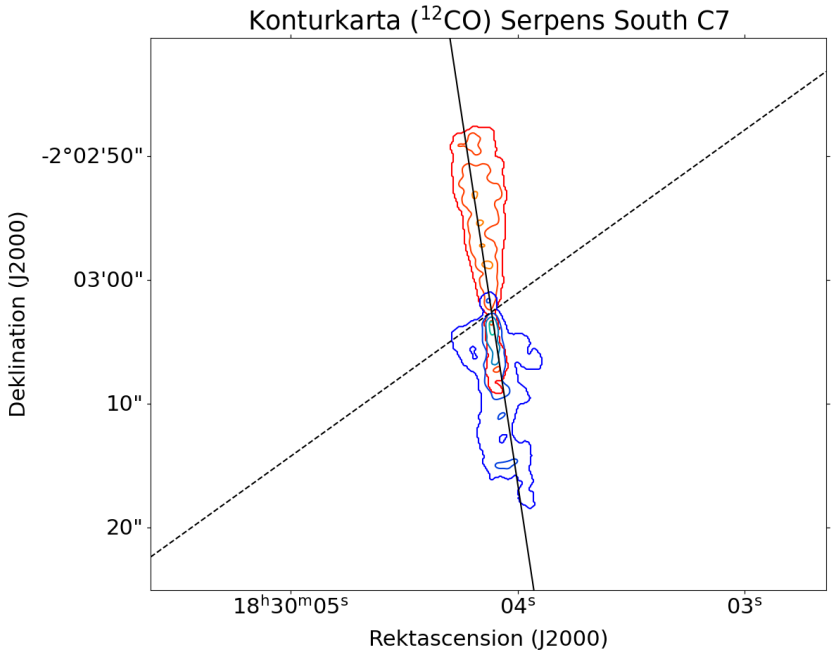
Under antagandet att någon av konturerna som representerar utflödet går runt den protoplanetära skivans centrum kan konturkartan förfinas till att enbart innehålla konturer från utflöden. Detta genom att först använda en algoritm för att identifiera konturer som går runt skivans centrum och därefter på samma sätt hitta de konturer som innesluts av de identifierade konturerna. Denna process visas i Figur 3.10.
’



Figur 3.10: I (a) visas det första steget där alla konturer i Figur 3.9 som inte går runt skivans centrum förkastats. I (b) har de inre konturer som i (a) förkastades återfåtts.

Med de förfinade konturerna anpassas sedan en rät linje genom konturerna för att bestämma vinkeln på utflödet. Detta genomförs som innan numeriskt med SciPy [45]. Här viktas datapunkterna med intensiteten, alltså med nivån på konturen. Vinkeln på denna linje jämförs sedan med den protoplanetära skivans vinkel, som bestäms i Avsnitt 3.2.1, för att få utflödets vinkel relativt skivan. Detta visas i Figur 3.11.

Nackdelen med denna metod är antagandet att konturerna går runt centrum, vilket inte alltid är fallet. Ytterligare en begränsning är valet av konturnivåer C, då detta får stor betydelse för konturernas utbredning. Om en betydlig mängd brus återfinns över konturnivåerna kan bruset återspeglas i resultatet. Detta är extra märkbart för `.pbcor`-filer då brusnivån inte är konstant, som beskrivet i Avsnitt 2.4.1.1.



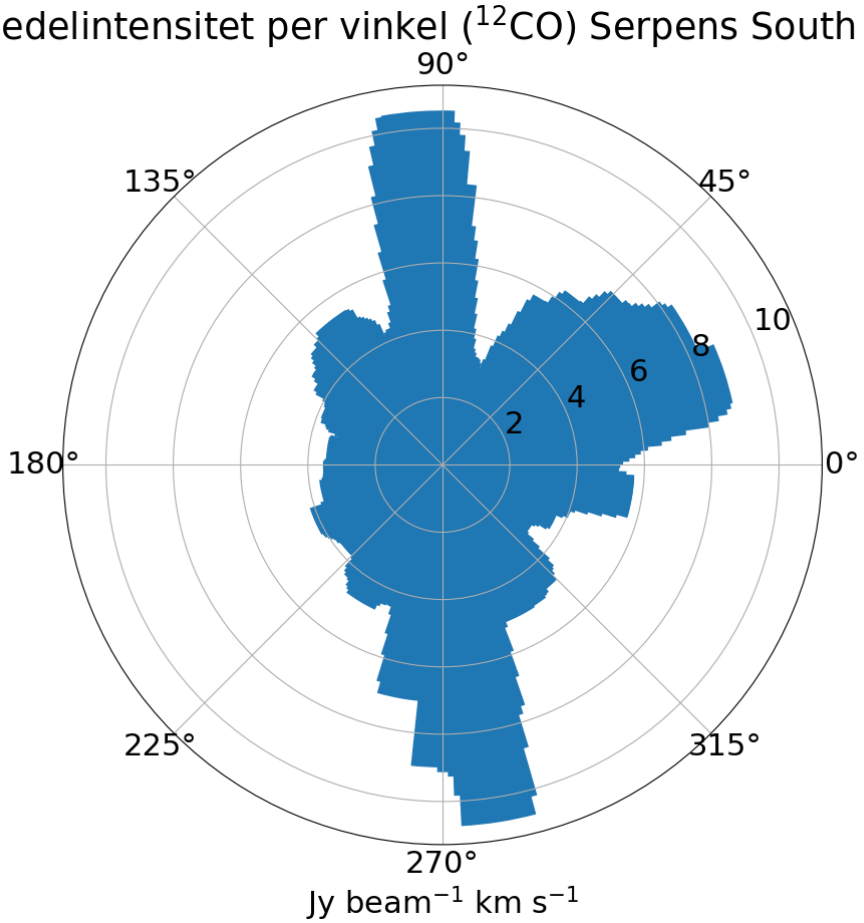
Figur 3.11: Förfinad konturkarta över momentavbildningarna i Figur 3.5. Den svarta heldragna linjen är den anpassade linjen genom konturerna och beskriver vinkeln på utflödet (~99 grader relativt x-axeln). Den svarta streckade linjen motsvarar den protoplanetära skivans vinkel (~43 grader relativt x-axeln). Utflödets vinkel relativt skivan, alltså vinkeln mellan dessa linjer, är 56 grader.

3.3.3 Identifiering av utflödets vinkel via medelintensitet i ett vinkelomfång.

En alternativ metod är att utgå från skivans centrum, dela upp momentavbildningen i vinklar, och för varje vinkel beräkna medelintensiteten. Detta bygger på antagandet att det kommer finnas mest intensitet i utflödets riktning.

Skivans centrum lokaliseras genom metoden beskriven i Avsnitt 3.2.1. Därefter beräknas varje koordinats vinkel till centrum, relativt positiva x-axeln, varpå dess intensitet adderas till den vinkelns totala intensitet. För att ta hänsyn till den icke-uniforma brusnivån som uppkommer på grund av korrigeringen för *primary beams*, beskrivet i Avsnitt 2.4.1.1, viktas denna intensitet inverst med avstånd från mitten. Eftersom skivan inte alltid ligger i mitten av bilden, med följden att olika antal intensiteter summeras för olika vinklar, divideras de summerade intensiteterna med antalet summerade intensiteter för att erhålla medelintensiteter.

För att undvika påverkan från skivan utelämnas ett område runt skivan från analysen. Resultatet förfinas ytterligare genom att förbigå intensiteter under brusnivån. Detta ger en vinkel som visas i Figur 3.12. Denna vinkel jämförs sedan med skivans vinkel, på samma sätt som beskrivet i Avsnitt 3.3.2, för att få den relativa vinkeln.



Figur 3.12: Medelintensitet per vinkel för momentavbildningen i Figur 3.6. Den beräknade vinkeln av utflödet relativt x-axeln blir 97 grader.

4
Resultat

Projektets resultat består av flera olika delar. Den första delen är beslutsträdet som filtrerar ut observationer som sannolikt innehåller diskar och utflöden baserat på de parametrar som nämns i Avsnitt 3.1.

Den andra delen av projektets resultat består av den kod som utvecklats för följande ändamål:

1. Identifiera relevanta frekvensomfång och generera momentavbildningar från dessa.
2. Identifiera tröskeln för brus och sålla bort värden under denna tröskel.
3. Identifiera skivan utifrån en kontinuum-bild.
4. Identifiera utflödenas omfång och riktning.
5. Generera konturkartor av de brusreduserade momentavbildningarna.
6. Föra statistik över observationer från ALMAs arkiv.

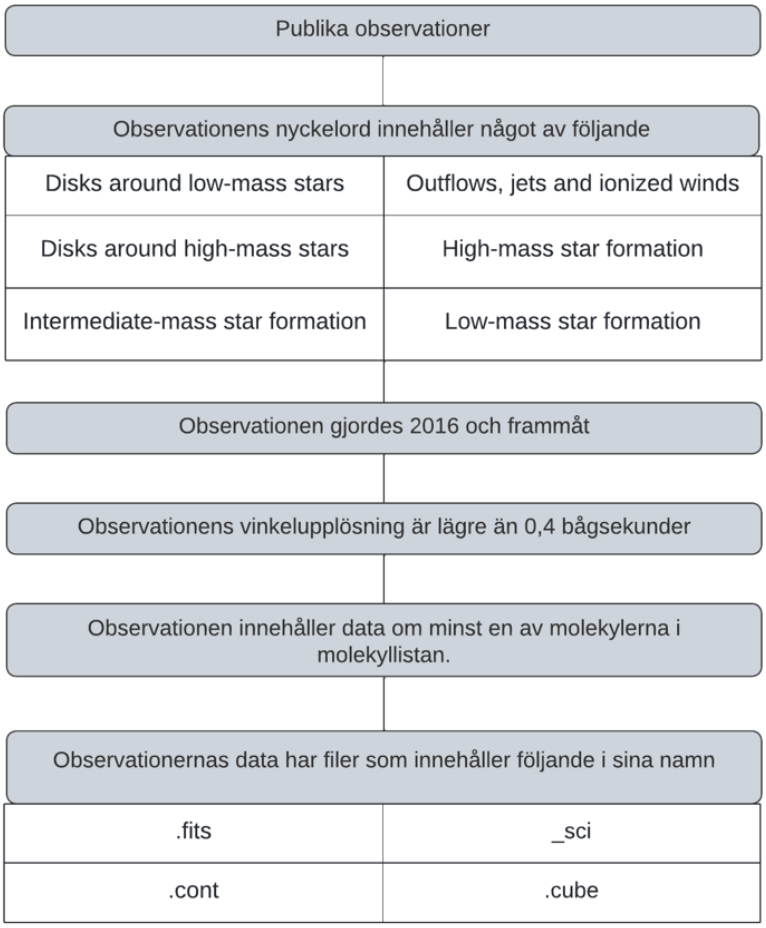
I ovanstående lista är punkt 1 och 2 nära knutna eftersom identifiering av relevanta frekvensomfång till viss del kan sägas vara identifiering av vilka frekvenser som inte innehåller brus. Således kan bortfiltreringen av brus ha direkt anknytning till punkt 1 beroende på val av metod.

Ett gemensamt drag för ovanstående punkter är att, med undantag för punkt 3, flera lösningar utvecklades för dess ändamål. Detta på grund av att en specifik lösning var mer eller mindre passande för olika observationer. Det vill säga, medan en lösning kunde ge lovande resultat för en observation och observationer lika denna, gav ofta samma metod vaga eller till synes oanvändbara resultat för andra observationer.

4.1 Beslutsträd

Genom restriktioner på nedladdningen av data blir alla observationer filtrerade enligt beslutsträdet i Figur 4.1. Filtrering på nyckelord, publiceringsår, vinkelupp-

lösning och molekyler från molekyllistan utförs enligt Avsnitt 3.1. Hela ALMA-arkivet innehåller ~56 000 observationer och efter denna filtrering återstår 3496 observationer. Efter denna första filtrering görs ett försök att ladda ner potentiella `.cont`-filer. Om det lyckas kommer även `.cube`-filer laddas ned varpå analys av den nedladdade datan följer.



Figur 4.1: Beslutsträdet som används för att sortera ut relevanta observationer från ALMAs arkiv för analys.

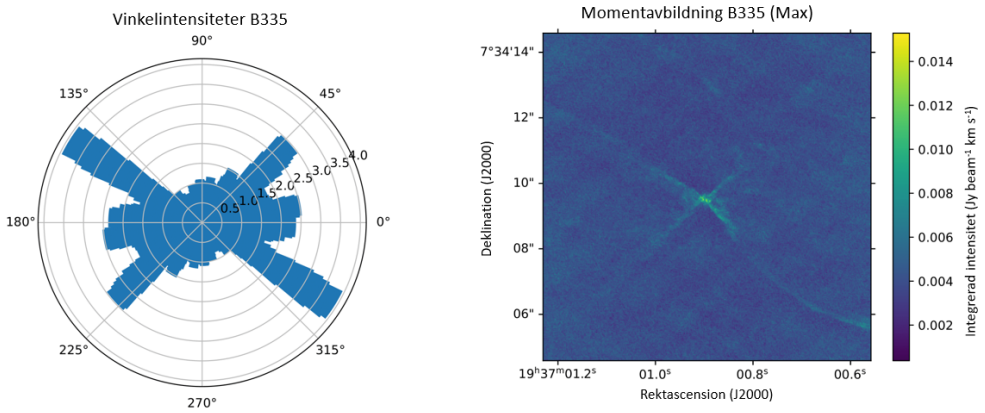
4.2 Momentavbildningar

Framställning av momentavbildningar har, som beskrivet i Avsnitt 3.2, uppnåtts genom tre olika metoder. Dessa består av en metod som filtrerar bort frekvenser med en medelintensitet under den fastställda tröskeln för brus, en metod som fastställer intressanta frekvensintervall genom att anpassa två eller tre endimensionella

Gauss-kurvor och hitta inflektionspunkter och en metod som för varje koordinat summerar de 10% högsta intensiteterna.

4.3 Identifiering av utflöde

Inom projektet har två lösningar producerats för vinkeluppskattning på utflöde. Den ena lösningen, beskriven i Avsnitt 3.3, summerar intensiteter i olika vinkelomfång och lämpas därför för smala utflöden som i observationerna är synliga som en solid stråle. Detta då hela eller en majoritet av utflödet hamnar inom ett vinkelomfång och därför ger större och mer definitivt utslag. För bredare utflöden blir resultaten vagare. Detta kan exemplifieras av observationer där det huvudsakligen är kanterna av utflödet som har betydande intensitet vilket gör att vinkeluppskattning försvåras. Detta på grund av att flera vinkelomfång kan få en högre summerad intensitet alternativt att endast en del av utflödet får en högre summerad intensitet. Se Figur 4.2a och Figur 4.2b för ett exempel där den ena kanten av utflödet får betydligt högre summerad intensitet än resten.

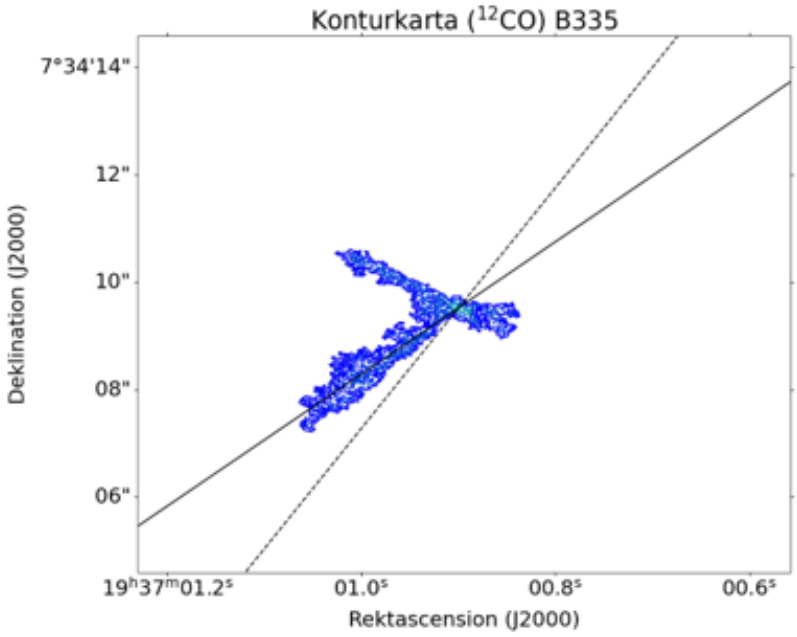


(a) **(b)**

Figur 4.2: I (a) visas summerade intensiteter för olika vinklar hos protostjärnan B335. I (b) visas momentavbildningen av utflödet från protostjärnan B335. Bilderna är baserade på $(2 \rightarrow 1)$ övergången hos ^{12}CO.

Den andra lösningen, beskriven i Avsnitt 3.3.2, använder sig av konturer som omringar skivan. Denna metod är, när applicerbar, mer generell eftersom den kan ge rimliga resultat för både smalare och bredare utflöden. Huvudsakliga förutsättningen för denna lösning, utöver att konturerna måste gå runt skivans mitt, är en relativt tydlig distinktion mellan utflöde och brus. Om bruset har en till-

räckligt hög intensitet vid utflödena kommer konturkartorna att till en mindre grad överensstämma med utflödena. Figur 4.3 visar ett exempel där brusnivån har gett upphov till en sämre konturkarta, där endast konturer från den blåförskjutna emissionen detekteras.



Figur 4.3: Exempel på identifiering av utflöde med konturkarta från en observation av B335. Se Figur 4.2b för momentavbildning från samma observation.

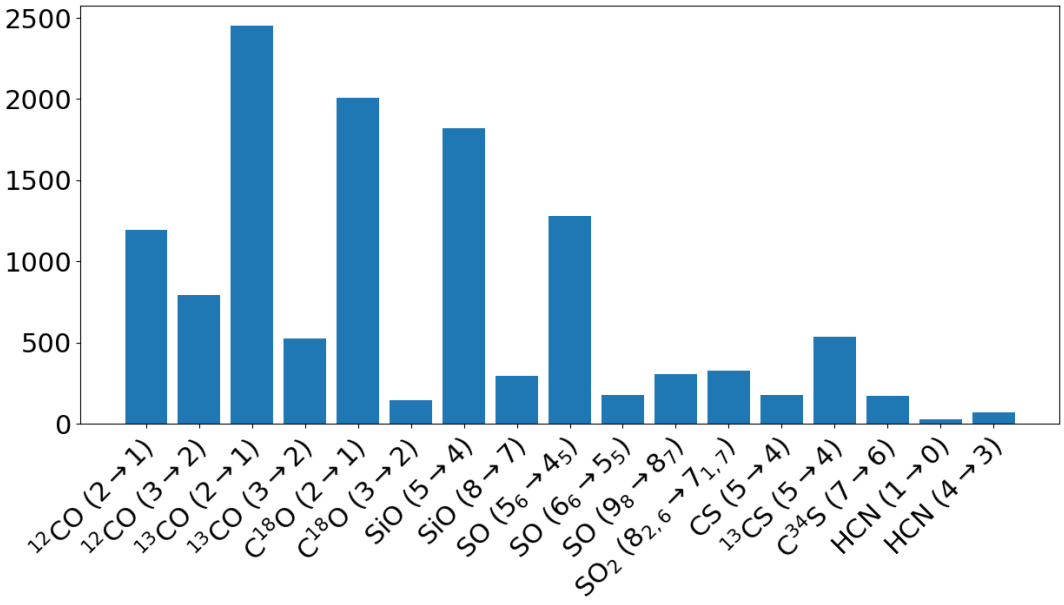
4.4 Statistik

En del av koden kan användas för att beräkna statistik genom att analysera observationer som filtrerats fram. Genom användning av *ALminer*, se Bilaga B för tillhörande kod, kan en *skymap*, stjärnkarta, genereras. Ett exempel av detta syns i Figur 4.6. Kartan visar stjärnhimlen där röda prickar indikerar varje observation som uppfyller den filtrering som användaren har definierat. Detta möjliggör för användaren att få en överblick över var på himlen observationer har genomförts.

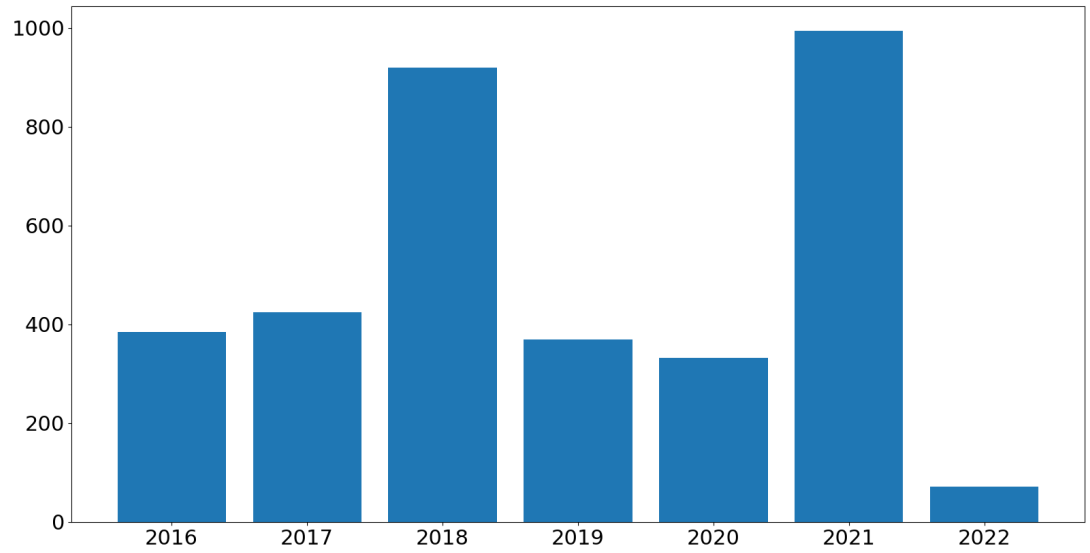
Användaren kan också gå igenom framfiltrerade observationer och se vilka som saknar `.cont`-filer vilket kan vara användbart för de andra funktionerna i programmet. Dessa visas i konsolen genom utskriven text som beskriver den totala mängden observationer som kontrollerats, mängden observationer som innehåller `.cont`-filer tillsammans med procentandelen observationer som innehåller `.cont`-filer. Det finns sedan flera funktioner som genererar stapeldiagram för att visua-

lisera statistik. Bland dessa finns det en funktion som visar vilka årtal de valda observationerna laddades upp till arkivet, visat i Figur 4.5.

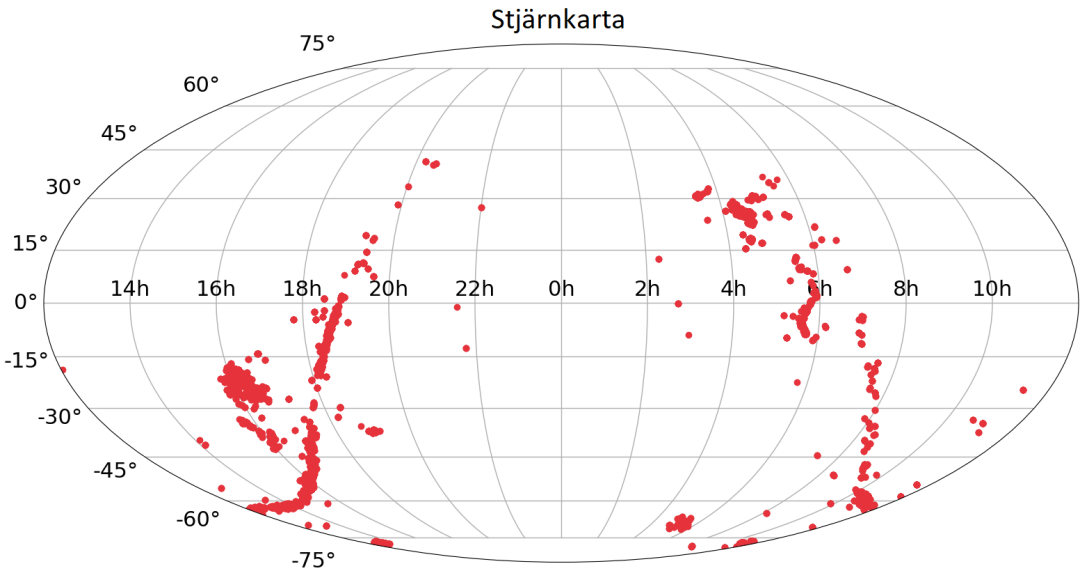
Det finns också en funktion som extraherar hur många observationer från sökningen som innehåller de olika molekylerna med tillhörande rotationsövergångar som är vanliga i observationer innehållande utflöden, beskrivna i Tabell A.1, som filtreras för. Resultatet visas i stapeldiagrammet i Figur 4.4. Ur figuren framgår det att de vanligaste rotationsövergångarna i observationer som filtrerats fram är ^{13}CO (2 $\rightarrow$ 1), C^{18}O (2 $\rightarrow$ 1), SiO (5 $\rightarrow$ 4), SO ($5_6 \rightarrow 4_5$) och ^{12}CO (2 $\rightarrow$ 1). Övriga molekyler förekommer inte lika frekvent i de valda observationerna.



Figur 4.4: Stapeldiagram som visar fördelningen av de olika molekylerna från Tabell A.1 i de framfiltrerade observationerna.



Figur 4.5: Stapeldiagram som visar årsfördelningen av antalet observationer framfiltrerade med beslutsträdet.



Figur 4.6: Stjärnkarta som indikerar var på stjärnhimlen observationer framfiltrerade med beslutsträdet är observerade.

4.5 Systematisk analys

I projektet har de metoder som utvecklats kombinerats för att bilda ett program som systematiskt går igenom de observationer som filtreras ut. Efter filtrering laddar programmet ner data för en observation i taget. På denna data utförs de analyser som utvecklats, varpå resultaten av dessa sparas. Därefter raderas observationsdatan och programmet går vidare till nästa observation för att ladda ner dess data och, på även den datan, genomföra analys. Denna programslinga fortsätter tills samtliga av de framfiltrerade observationer behandlats.

5
Diskussion och Slutsatser

Här diskuteras projektets resultat relaterat till syftet i Avsnitt 1.1, varpå slutsatser dras. Diskussionen berör även tillvägagångssättet som använts under arbetets gång och förslag på förbättringsområden för både metod och produkt tillsammans med vidareutveckling av produkten.

5.1 Diskussion

Projektets syfte skulle kunna delas upp i delmål som består av filtrering och inhämtning av observationsdata, framtagning av momentavbildningar, identifiering av utflöde samt analys av den spatiala relationen som molekyler och stoft har med utflöden. Vissa av delmålen visade sig vara mer tidskrävande och svårare att fullfölja än andra. Därmed skiljer det sig åt hur väl projektets syfte uppnåddes, beroende på vilken aspekt av syftet som beaktas.

Filtrering och inhämtning av observationsdata har, inom ramen för detta projekt, i stort uppfyllts. Som nämnt i Avsnitt 3.1 filtreras i detta projekt observationerna enligt beslutsträdet i Figur 4.1. Parametrarna i beslutsträdet valdes för ändamålet att studera utflöden i stjärnbildning. För andra forskningsområden inom astronomin är därmed inte det producerade beslutsträdet lika användbart. Däremot kan beslutsträdet modifieras med exempelvis andra nyckelord och molekyler. På så vis kan beslutsträdet få fler användningsområden än enbart studier av molekylära utflöden.

Framtagning av momentavbildningar har genomförts med varierande framgång. De metoder som utvecklats ger en tydligare bild av datan, men kan ge bättre eller sämre resultat beroende på bland annat brusnivå och utformning av frekvensprofilen. Medan framtagning av momentavbildningar inte nödvändigtvis är en förutsättning för framgången av projektet, har det bidragit till ett bättre uppfyllande av syftet. Dock är de användbara för den utflödesidentifiering som genomförs, då en lägre andel brus och oönskad emission ger bättre förutsättningar för denna.

Viss identifiering av utflöden har uppnåtts, men stor potential för förbättring kvarstår. Som nämnts i Avsnitt 3.3 och Avsnitt 4.3 har metoder utvecklats för identifiering av utflöden. Dessa lösningar har definitivt viss användning eftersom de kan ge en uppfattning om utflödets utformning. Användbarheten begränsas dock av att resultaten som genereras inte alltid är helt korrekta. Dessutom är detaljnivån på även de mer lyckade resultaten oftast ganska låg, det vill säga att den form som fås är en relativt grov uppskattning. Trots detta kan metoderna ofta vara användbara för att relativt precist utvinna vinkeln av utflöden, vilket ger en viss ökad nytta.

Något som ursprungligen var tänkt att utföras var att låta programmet gå igenom samtliga framfiltrerade observationer och utföra analyserna. Detta har dock inte gjorts, huvudsakligen av tidsskäl. En observation som har gjorts är att den största bidragande faktorn till programmets exekveringstid är just nedladdningen av datan, då den totala datamängden även efter filtrering är hundratals TB. Dessutom varierar den tillgängliga bandbredden från ARCerna vilket leder till att nedladdningshastigheten stundtals kan vara kraftigt begränsad, vid ett tillfälle var den endast cirka 12 kB/s.

Analys av den spatiala relationen mellan molekyler och stoft med utflöden har inte uppnåtts. För att detta delmål skulle uppfyllas skulle det, i sin mest grundläggande form, behövas en identifiering av utflödets utformning. I och med att detta inte uppnåtts till en tillfredsställande grad, har inte heller någon form av analys av den spatiala relationen kunnat genomföras.

5.2 Förbättringsområden

Medan genererandet av momentavbildningar har varit framgångsrik för många observationer finns det fortfarande en andel data som på olika sätt hindrar genererandet av momentavbildningar. Detta kan bero på många olika anledningar varav två diskuteras i Avsnitt 3.2.2. Metoden för att generera momentavbildningar kan därför troligen förbättras på något sätt som inte utforskats hittills under projektets gång.

Som diskuterats i Avsnitt 5.1 har inte automatiserad jämförelse mellan observationer uppfyllts. Varierande kvalitet av ALMA data och utflödens varierande morfologi gör det svårt att automatiskt, utan mänsklig interaktion, jämföra observationer med varandra. Dock, om en sådan automatisk jämförelse åstadkommits skulle den dels kunna användas för att automatiskt hitta utflöden och skivor i observationer, men också för att komma fram till slutsatser om egenskaper hos utflöden.

5.3 Vidareutveckling av projektet

Under arbetets gång har en del idéer framförts som antingen har sträckt sig utanför ramen för detta arbete eller som inte har passat in med syftet som definierats för projektet. Dessa idéer har då skrivits ned och diskuterats med handledare. Projektet kan med hjälp av vissa av dessa idéer utvecklas vidare för att producera en bättre och mer användbar produkt.

Något som utforskats under projektets senare delar är möjligheten att anpassa antingen parabler eller en paraboloid för att erhålla formen av utflöden. Detta har dock, på grund av tidsbrist, inte uppnåtts och skulle därför vara en potentiell vidareutveckling.

En av idéerna för att producera en bättre produkt är att inkludera en större mängd molekyler i analysen för att minimera risken att missa potentiella upptäckter genom att ignorera molekyler. En framtida utveckling kan därför vara att låta användaren specificera vilka molekyler som är intressanta för dem. Detta kan också inkludera att användaren också får välja vilka rotationsövergångar hos molekylerna som ska analyseras. Denna funktionalitet är redan möjlig då molekylerna som analyseras kommer från en `.csv`-fil som är framställd via *Splatalogue*, men kan göras mer åtkomlig för användaren [47]. Detta kan åstadkommas genom att låta användaren specificera vilka molekyler de vill analysera och sedan generera *Splatalogue*-filen baserat på inmatningen.

En annan utveckling till projektet är att använda ett konvolutionerande neuralt nätverk för att utföra fler funktioner än att bildanalysera och jämföra momentkartor hos utflöden. De kan användas för att till exempel välja ut intressanta observationer som är relevanta till utflöden genom att hitta gemensamma egenskaper hos de observationer som tidigare ansetts intressanta.

Användningen av neurala nätverk kan även vara intressant för de områden som projektet redan har berört, exempelvis identifiering av utflöden, då det givet bra träningsdata bör kunna resultera i en mer generell lösning än det som åstadkommits i detta projekt.

En viktig vidareutveckling är även att utveckla metoder för analys av stoft och molekyldistribution av observerade objekt. I detta syfte skulle metoder krävas som, med hög precision, kan bestämma formen av utflödet. Som en vidareutveckling av detta skulle en formbestämning av utflödet över flera observationer av samma objekt ge en vidare möjlighet för analys, särskilt för distributionen av olika molekyler givet att någon av observationerna använder olika frekvensband.

5.4 Slutsatser

Sammanfattningsvis har projektet lyckats med sitt syfte att utveckla ett verktyg som kan vara till hjälp för forskare som systematiskt vill analysera ALMAs arkiv. Genom det utvecklade programmet får forskarna tillgång till nedladdningsrutiner, statistiska verktyg och analytiska funktioner som genererar momentavbildningar, konturbilder och vinkeluppskattningar. För att vidareutveckla projektet mot att automatiskt jämföra olika observationer hade det behövts en automatisk identifiering av utflödesområden, då detta ligger till grund för att kunna jämföra olika observationsdata för samma objekt. Alltså hade det varit ett betydelsefullt nästa steg för projektet. Avslutningsvis har en användbar produkt framtagits som kan hjälpa forskare att systematiskt studera ALMAs arkiv och protostjärnors utflöden.

Referenser

[1] NASA, ”Our Solar System.” nasa.gov. https://solarsystem.nasa.gov/solar-system/our-solar-system/in-depth/ (hämtad maj. 5, 2022).

[2] D. Dobrijevic, “Orion Nebula: Facts about Earth’s nearest stellar nursery,” *space.com*, nov. 2021. [Online]. Tillgänglig: https://www.space.com/orion-nebula (hämtad maj. 11, 2022).

[3] T. P. Greene, “Protostars ”Stellar embryology” takes a step forward with the first detailed look at the youngest Sun-like stars,” *American Scientist*, årg. 89, 2001. [Online]. Tillgänglig: https://www.americanscientist.org/sites/americanscientist.org/files/2005223144527_306.pdf.

[4] I. Pascucci, S. Cabrit, S. Edwards m. fl., “The Role of Disk Winds in the Evolution and Dispersal of Protoplanetary Disks,” 2022. [Online]. Tillgänglig: https://arxiv.org/pdf/2203.10068.pdf.

[5] ALMA Observatory, ”About ALMA, at first glance.” almaobservatory.org. https://www.almaobservatory.org/en/about-alma/ (hämtad febr. 16, 2022).

[6] ALMA Observatory, ”Star and planet formation.” almaobservatory.org. https://www.almaobservatory.org/en/about-alma/how-alma-works/capabilities/star-and-planet-formation/ (hämtad maj. 10, 2022).

[7] ALMA Observatory, ”Detecting extrasolar planets under formation.” almaobservatory.org. https://www.almaobservatory.org/en/about-alma/how-alma-works/capabilities/detecting-extrasolar-planets-under-formation-with-alma/ (hämtad maj. 10, 2022).

[8] ALMA Observatory, ”How ALMA Observations are carried out.” almaobservatory.org. https://www.almaobservatory.org/en/about-alma/how-alma-works/how-alma-observations-are-carried-out/ (hämtad april. 28, 2022).

[9] S. Aalto, Föreläsning, Ämne: ”Exoplanets-Modern Astrophysics”, Institutionen för rymd-, geo- och miljövetenskap, Chalmers tekniska högskola, Göteborg, nov., 2022.

[10] Harvard, ”Pre-Main Sequence (PMS) Stars.” harvard.edu. https://lweb.cfa.harvard.edu/~pberlind/atlas/htmls/pmsstars.html (hämtad maj. 11, 2022).

[11] J. A. Nuth och N. M. Johnson, “Complex Protostellar Chemistry,” *Science*, s. 424–425, april 2012. [Online]. Tillgänglig: https://www.science.org/doi/full/10.1126/science.1219709.

[12] S. Stahler, “Bipolar Flow,” *Encyclopedia of Astrobiology*, Tillgänglig: https://link.springer.com/referenceworkentry/10.1007/978-3-642-11274-4_195 (hämtad april. 5, 2022).

[13] P. Bjerkeli, M. H. van der Wiel, D. Harsono m. fl., “Resolved images of a protostellar outflow launched by an extended disk wind,” *Nature*, 2016. [Online]. Tillgänglig: https://arxiv.org/ftp/arxiv/papers/1612/1612.05148.pdf. DOI: 10.1038/nature20600.

[14] D. J. Griffiths och D. F. Schroeter, “Time-independent Schrödinger Equation,” i *Introduction to Quantum Mechanics*, 3. utg., Cambridge, England: Cambridge University Press, 2018.

[15] M. Sveningsson och M. Andersson, ”Laserinducerad Fluorescens från Jodmolekyler” https://www.chalmers.se/sv/centrum/fysikcentrum/utbildning/fol/laborationer/Documents/A-labbar/a12.pdf (hämtad april. 28, 2022).

[16] E. G. Blackman, ”The Bohr Model ” https://www.pas.rochester.edu/~blackman/ast104/bohr.html (hämtad maj. 11, 2022).

[17] B. Bayram och M. Freamat, “A Spectral Analysis of Laser Induced Fluorescence of Iodine,” juli 2015. DOI: 10.1119/bfy.2015.pr.002.

[18] University of Oxford, ”Molecular energy levels and spectroscopy.” vallance.chem.ox.ac.uk. http://vallance.chem.ox.ac.uk/pdfs/MolecularEnergyLevelsNotes.pdf (hämtad maj. 11, 2022).

[19] Indian Institute of Science Education and Research, ”Rotation and Vibration of Diatomic Molecules.” sites.iiserpune.ac.in. http://sites.iiserpune.ac.in/~bhasbapat/phy420_files/Demtroeder_rotovibrazioni.pdf (hämtad maj. 11, 2022).

[20] D. F. Miller, “The Properties of Electromagnetic Radiation,” i *Basics of Radio Astronomy for the Goldstone-Apple Valley Radio Telescope*, Tillgänglig: https://www2.jpl.nasa.gov/radioastronomy/radioastronomy_all.pdf (hämtad mars. 28, 2022), Pasadena, California, USA: California Institute of Technology, 1998. [Online].

[21] ESA, “The electromagnetic spectrum,” *CESAR's Booklet*, Tillgänglig: https://cesar.esa.int/upload/201711/electromagnetic_spectrum_booklet_wboxes.pdf (hämtad mars. 28, 2022).

[22] R. G. Grainger, “Electromagnetic Absorption and Emission by Atoms and Molecules,” i *An Atmospheric Radiative Transfer Primer*, Tillgänglig: http://eodg.atm.ox.ac.uk/user/grainger/research/book/ (hämtad mars. 23, 2022), Boca Raton, Florida, USA: CRC Press, [Online].

[23] C. Nordling och J. Österman, “Physical Formulas and Diagrams,” i *Physics Handbook*, 4. utg., Lund, Sverige: Studentlitteratur, 1980.

[24] ESO, ”ALMA - In search of our cosmic origins.” ESO.org. https://www.eso.org/public/teles-instr/alma/ (hämtad maj. 6, 2022).

[25] ALMA Observatory, ”How ALMA Works.” almaobservatory.org. https://www.almaobservatory.org/en/about-alma/how-alma-works/ (hämtad febr. 16, 2022).

[26] ESO, ”ALMA’s Antennas.” ESO.org. https://www.eso.org/public/teles-instr/alma/antennas/ (hämtad febr. 16, 2022).

[27] ALMA Observatory, ”Interferometry.” almaobservatory.org. https://www.almaobservatory.org/en/about-alma/how-alma-works/technologies/interferometry/ (hämtad febr. 16, 2022).

[28] R. Warmels m.fl., ”ALMA Technical Handbook” arc.iram.fr https://arc.iram.fr/documents/cycle6/ALMA_Cycle6_Technical_Handbook.pdf (hämtad maj. 5, 2022).

[29] ALMA Observatory, “ALMA In-depth,” *ALMA Newsletter*, nr 5, april 2010. [Online]. Tillgänglig: https://www.almaobservatory.org/wp-content/uploads/2017/06/05_how_will_alma_make_images.pdf.

[30] ALMA Observatory, ”Capabilities.” almaobservatory.org. https://www.almaobservatory.org/en/about-alma/how-alma-works/capabilities/ (hämtad febr. 18, 2022).

[31] B. Koberlein, ”How interferometry works, and why it’s so powerful for astronomy” Phys.org. https://phys.org/news/2020-02-interferometry-powerful-astronomy.html (hämtad april. 19, 2022).

[32] ALMA Observatory, “Supercomputer Ready to make ALMA a Powerful Telescope,” dec. 2012. [Online]. Tillgänglig: https://www.almaobservatory.org/en/press-releases/supercomputer-ready-to-make-alma-a-powerful-telescope/.

[33] ALMA Observatory, ”ALMA Proposal Review.” almascience.eso.org. https://almascience.eso.org/proposing/alma-proposal-review (hämtad maj. 11, 2022).

[34] F. Stoehr m.fl., ”ALMA Science Archive Manual” almascience.eso.org https://almascience.eso.org/documents-and-tools/cycle9/science-archive-manual (hämtad maj. 5, 2022).

[35] D. C. Wells och E. W. Greisen, “Introduction,” i *FITS - a Flexible Image Transport System*, Trieste, Italien: Osservatorio Astronomico di Trieste, 1979.

[36] A. Papoulis och S. U. Pillai, “Moments,” i *Probability, Random Variables and Stochastic Processes*, 4. utg., Boston, Massachusetts, USA: McGraw-Hill, 2002.

[37] I. Gustavsson och K. Holmåker, “Linjära minskakvadratproblem,” i *Numerisk Analys*, 1. utg., Stockholm, Sweden: Liber, 2016.

[38] K. P. Burnham och D. R. Anderson, “Background Material,” i *Model Selection and Multimodel Inference*, 2. utg., New York, USA: Springer-Verlag, 2002.

[39] E. W. Weisstein, ”Normal Distribution ” mathworld.wolfram.com `https://mathworld.wolfram.com/NormalDistribution.html` (hämtad maj. 9, 2022).

[40] A. Papoulis och S. U. Pillai, “Continuous-Type Random Variables,” i *Probability, Random Variables and Stochastic Processes*, 4. utg., Boston, Massachusetts, USA: McGraw-Hill, 2002.

[41] J. J. Condon, “Errors in Elliptical Gaussian Fits,” *Publications of the Astronomical Society of the Pacific*, årg. 109, s. 166–172, febr. 1997. DOI: `10.1086/133871`.

[42] A. Ahmadi, ”ALminer: ALMA Archive Mining & Visualization Toolkit.” alminer.readthedocs.io. `https://alminer.readthedocs.io/en/latest/` (hämtad febr. 18, 2022).

[43] D. Petry m.fl., ”ALMA QA2 Data Products for Cycle 3 ” help.almascience.org `https://almascience.eso.org/processing/documents-and-tools/cycle3/ALMAQA2Products3.0.pdf` (hämtad maj. 10, 2022).

[44] C. Ubach, ”What Calibration and Imaging products will be delivered to me? ” help.almascience.org `https://help.almascience.org/kb/articles/what-calibration-and-imaging-products-will-be-delivered-to-me` (hämtad maj. 9, 2022).

[45] P. Virtanen m. fl., “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python,” *Nature Methods*, årg. 17, s. 261–272, 2020. DOI: `10.1038/s41592-019-0686-2`.

[46] J. D. Hunter, “Matplotlib: A 2D graphics environment,” *Computing in Science & Engineering*, årg. 9, nr 3, s. 90–95, 2007. DOI: `10.1109/MCSE.2007.55`.

[47] Splatalogue, `https://splatalogue.online/advanced1.php` (hämtad maj. 9, 2022).

[48] H.-W. Yen, S. Takakuwa, N. Ohashi m. fl., "ALMA observations of infalling flows toward the Keplerian disk around the Class I protostar L1489 IRS," *The Astrophysical Journal*, årg. 793, nr 1, 2014.

[49] Ł. Tychoniec, C. L. Hull, L. E. Kristensen m. fl., "Chemical and kinematic structure of extremely high-velocity molecular jets in the Serpens Main star-forming region," *Astronomy & Astrophysics*, årg. 632, 2019.

[50] F. Louvet, C. Dougados, S. Cabrit m. fl., "ALMA observations of the Th 28 protostellar disk-A new example of counter-rotation between disk and optical jet," *Astronomy & Astrophysics*, årg. 596, 2016.

[51] L. Busch, A. Belloche, S. Cabrit m. fl., "The dynamically young outflow of the Class 0 protostar Cha-MMS1," *Astronomy & Astrophysics*, årg. 633, 2020.

[52] S. Jin, A. Isella, P. Huang m. fl., "New Constraints on the Dust and Gas Distribution in the LkCa 15 Disk from ALMA," *The Astrophysical Journal*, årg. 881, nr 2, aug. 2019. DOI: 10.3847/1538-4357/ab2dfe. URL: https://doi.org/10.3847/1538-4357/ab2dfe.

[53] F. Yusef-Zadeh, M. Royster, M. Wardle m. fl., "ALMA observations of the Galactic center: SiO outflows and high-mass star formation near Sgr A," *The Astrophysical Journal Letters*, årg. 767, nr 2, 2013.

[54] B. Tabone, S. Cabrit, E. Bianchi m. fl., "ALMA discovery of a rotating SO/SO2 flow in HH212-A possible MHD disk wind?" *Astronomy & Astrophysics*, årg. 607, 2017.

[55] L. Dewangan, I. Zinchenko, P. Zemlyanukha m. fl., "The Disk–Outflow System around the Rare Young O-type Protostar W42-MME," *The Astrophysical Journal*, årg. 925, nr 1, 2022.

[56] Y. Zhang, A. E. Higuchi, N. Sakai m. fl., "Rotation in the NGC 1333 IRAS 4C Outflow," *The Astrophysical Journal*, årg. 864, nr 1, 2018.

[57] J. Bally, R. K. Mann, J. Eisner m. fl., "ALMA observations of the largest proto-planetary disk in the Orion Nebula, 114–426: A CO silhouette," *The Astrophysical Journal*, årg. 808, nr 1, 2015.

[58] T. Baug, K. Wang, T. Liu m. fl., "An ALMA study of outflow parameters of protoclusters: outflow feedback to maintain the turbulence," *Monthly Notices of the Royal Astronomical Society*, årg. 507, nr 3, juli 2021, ISSN: 0035-8711. DOI: 10.1093/mnras/stab1902. URL: https://doi.org/10.1093/mnras/stab1902.

A

Utvalda molekyler och tillhörande rotationsövergångar

Tabell A.1: Lista över valda molekylära rotationsövergångar. Samtliga övergångar är vanligt förekommande i observationer av utflöden med ALMA. Referenskolumnen uppger artiklar som observerat utflöden med respektive rotationsövergång med ALMA.

Molekyl	Rotationsövergång	Frekvens (GHz)	Referens
^{12}CO	$2 \rightarrow 1$	230,538	[13] [48] [49]
^{12}CO	$3 \rightarrow 2$	345,796	[50] [51] [52]
^{13}CO	$2 \rightarrow 1$	220,399	[13] [48] [50]
^{13}CO	$3 \rightarrow 2$	330,588	[51] [52]
C^{18}O	$2 \rightarrow 1$	219,560	[13] [48]
C^{18}O	$3 \rightarrow 2$	329,331	[52]
SiO	$5 \rightarrow 4$	217,105	[53] [49]
SiO	$8 \rightarrow 7$	347,331	[54] [55]
SO	$5_6 \rightarrow 4_5$	219,949	[48]
SO	$6_6 \rightarrow 5_5$	244,936	[56]
SO	$9_8 \rightarrow 8_7$	346,528	[54]
SO_2	$8_{2,6} \rightarrow 7_{1,7}$	334,673	[54]
CS	$5 \rightarrow 4$	244,936	[56]
^{13}CS	$5 \rightarrow 4$	231,221	[55]
C^{34}S	$7 \rightarrow 6$	337,396	[54]
HCN	$1 \rightarrow 0$	88,631	[49]
HCN	$4 \rightarrow 3$	354,505	[57] [58]

B

ALMA_statistics.py

```
from datetime import date
from tabnanny import filename_only
import alminer
import alminer_extensions
import pandas
import alminer_extensions
import matplotlib
import numpy
import matplotlib.pyplot as plt
import os
import FittingData



# this changes the default date converter for better interactive plotting of
    dates:
plt.rcParams['date.converter'] = 'concise'
query = alminer.keysearch({'science_keyword': ['Outflows, jets and ionized
    winds']})

def observations_by_year(query):

    #the years that are being checked
    categories = ["2011", "2012", "2013", "2014", "2015", "2016", "2017", "
    2018", "2019", "2020", "2021", "2022"]
    data = []

    #going through the query and counting amount of observations each year
    for i in categories:
        count = 0
        queryData = (query[query.obs_release_date > i])
        queryData = (queryData[queryData.obs_release_date < str(int(i)+1)])

        for i in range(len(queryData)):
            count += 1
        data.append(count)

```

```
35      #prints figure showing the data
36      fig1, ax = plt.subplots(figsize=(8, 4), layout='constrained')
37      fig1.canvas.manager.set_window_title('Publikationsår för observationerna
        i sökningen')
38      ax.bar(categories, data)
39
40
41
42  #statistics about what electon transitions are covered by observations
43  def electron_transitions(query, frequencies, z=0., only_relevant=True):
44      line_names = (frequencies['Species'] + ' ' + frequencies['Resolved QNs'
        ]).tolist()
45      line_freqs = frequencies['Ordered Freq (GHz)'].tolist()
46      minfreq = min(query['min_freq_GHz'].tolist())
47      maxfreq = max(query['max_freq_GHz'].tolist())
48      categories = []
49      data = []
50      for t, line in enumerate(line_names):
51          if not (minfreq <= line_freqs[t] <= maxfreq) and only_relevant:
52              continue
53          line_df = alminer.line_coverage(query, line_freq=line_freqs[t], z=z,
        line_name=line_names[t], print_summary=False, print_targets=False)
54          if not line_df.empty:
55              categories.append(line_names[t])
56              data.append(len(line_df))
57
58      fig2, bx = plt.subplots(figsize=(12, 8))
59      fig2.canvas.manager.set_window_title('Molekyler och elektronövergångar i
        sökningen')
60      y_pos = range(len(categories))
61      plt.xticks(y_pos, shorten_line_names(categories), rotation=90)
62      bx.bar(shorten_line_names(categories), data)
63
64  # skymap för vart alla observationer från arkivet är
65  def fig_skymap(query):
66      alminer.plot_sky(query)
67
68  def shorten_line_names(names):
69      newNames = []
70      for name in names:
71          newNames.append(name.replace("J=","").replace("v=0","").split(",F=")
        [0])
72      return newNames
73
74  def missing_cont_file(query):
75      total_obs = 0
76      has_cont = 0
77      for i in range(len(query)):
78          line = query.take([i])
```

```
79          total_obs += 1
80          data = alminer.download_data(line, fitsonly=False, dryrun=True,
    print_urls=False, filename_must_include=".cont")
81          print(data)
82          if data != None:
83              has_cont += 1
84          percentage = has_cont/total_obs * 100
85          print(str(has_cont) + " observation(er) inehåller en .cont fil utav
    totalt: " + str(total_obs) + " observation(er)." + " (" + str(percentage)
     + "%)")
86
87 ## fix this function, use functions in fitting data that is actually used to
     analyse data
88 def analasys_success(total_files_analysed, total_files_analysed_2,
    analasys_failed, analasys_failed_2):
89     analasys_fig, dx = plt.subplots(figsize=(9, 5), layout='constrained')
90     analasys_fig.canvas.manager.set_window_title('Mängden framgångsrika
    analyser i sökningen')
91     categories_analasys_success = ["Analyser", "Misslyckade analyser","
    Analyser 2", "Misslyckade analyser 2"]
92     data_analasys_success = [total_files_analysed, analasys_failed,
    total_files_analysed_2, analasys_failed_2]
93     dx.bar(categories_analasys_success, data_analasys_success)
```

C

alminer_extensions.py

```
from os import system
import alminer
import pandas as pd
from astroquery.alma import Alma
from astropy.io import fits
import numpy as np
import os

# Below license is for ALminer since we have modified some code from there
"""
MIT License

Copyright (c) 2021 Aida Ahmadi, Alvaro Hacar

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
"""

```

```
34
35 def get_freq(freqs):
36     res = freqs.split(',')
37     return float(res[0])
38
39
40 # Takes in frequencies from a splatalogue CSV file and returns a formated
    pandas data frame
41 def get_frequencies(dir):
42     csv_file = pd.read_csv(dir, sep=':')
43     csv_file = csv_file.drop(['Chemical Name', 'CDMS/JPL Intensity', 'Lovas/
    AST Intensity',
44                               'E_L (cm^-1)', 'E_L (K)', 'Linelist'], axis=1)
45     csv_file = csv_file.rename(columns={'Ordered Freq (GHz) (rest frame,
    redshifted)': 'Ordered Freq (GHz)'})
46     csv_file['Ordered Freq (GHz)'] = csv_file['Ordered Freq (GHz)'].apply(
    get_freq)
47     return csv_file[csv_file['Resolved QNs'].str.contains('F') == False]
48
49 # Takes dataframe returned by get_frequencies and a given frequency range
    and matches to a
50 # molecule, if no direct match gives closest match
51 def get_molecule(ref_freqs, min_freq, max_freq):
52     smallest_diff = 1000000000000 # 1000 GHz
53     closest_match = None
54     mid_freq = (max_freq + min_freq) / 2
55
56     for index, row in ref_freqs.iterrows():
57         tmp = min(abs(row['Ordered Freq (GHz)']*1000000000 - min_freq), abs(
    row['Ordered Freq (GHz)']*1000000000 - max_freq))
58         if tmp < smallest_diff:
59             smallest_diff = tmp
60             closest_match = row['Species'] + ' ' + row['Resolved QNs']
61         if min_freq <= row['Ordered Freq (GHz)']*1000000000 <= max_freq:
62             return row['Species'] + ' ' + row['Resolved QNs']
63     if closest_match is None:
64         closest_match = 'No match found'
65     return 'Closest match: ' + closest_match
66
67
68 # Generalized lines function. frequencies takes the outputted dataframe from
     get_frequencies
69 def get_lines(observations, frequencies, z=0., only_relevant=True,
    print_summary=True, print_targets=True):
70     df_list = []
71     line_names = (frequencies['Species'] + ' ' + frequencies['Resolved QNs'
    ]).tolist()
72     line_freqs = frequencies['Ordered Freq (GHz)'].tolist()
73     minfreq = min(observations['min_freq_GHz'].tolist())
```

```
74      maxfreq = max(observations['max_freq_GHz'].tolist())
75
76      for t, line in enumerate(line_names):
77          if not (minfreq <= line_freqs[t] <= maxfreq) and only_relevant:
78              continue
79          line_df = alminer.line_coverage(observations, line_freq=line_freqs[t
    ], z=z,
80                                          line_name=line_names[t],
    print_summary=print_summary,
81                                          print_targets=print_targets)
82          if not line_df.empty:
83              df_list.append(line_df)
84      if df_list:
85          df = pd.concat(df_list)
86          # need to reset the index of DataFrame so the indices in the final
    DataFrame are consecutive
87          df = df.drop_duplicates().reset_index(drop=True)
88          return df
89      else:
90          print("Found no ALMA observations covering transitions of given
    molecules.")
91          print("---------------------------------")
92
93
94  SiO_line_names = ["SiO (1-0)", "SiO (2-1)", "SiO (3-2)", "SiO (4-3)", "SiO
    (5-4)", "SiO (6-5)",
95                    "SiO (7-6)", "SiO (8-7)", "SiO (9-8)", "SiO (10-9)", "SiO
    (11-10)", "SiO (12-11)"]
96  SiO_line_freq = {"SiO (1-0)": 43.42376000, "SiO (2-1)": 86.84696000, "SiO
    (3-2)": 130.26861000,
97                   "SiO (4-3)": 173.68831000, "SiO (5-4)": 217.10498000, "SiO
    (6-5)": 260.51802000,
98                   "SiO (7-6)": 303.92696000, "SiO (8-7)": 347.33063100, "SiO
    (9-8)": 390.72844830,
99                   "SiO (10-9)": 434.11955210, "SiO (11-10)": 477.50309650, "
    SiO (12-11)": 520.87820390}
100
101 # removes all projects which do not include any of the rotational
    transitions we want to study.
102 def removeAllProjectsWithoutMolecules(dataframe, frequencies):
103     trueFalse = [moleculesInRange(minFreq, maxFreq, frequencies) for minFreq
    , maxFreq in
104                  zip(dataframe['min_freq_GHz'], dataframe['max_freq_GHz'])]
105     dataframe = dataframe.reset_index()
106     dataframe = dataframe.drop(trueFalseToIndex(trueFalse)) # Think we can
    just do something like dataframe = dataframe[trueFalse]
107     return dataframe
108
109 # checks if there are any transitions of interest in the range.
```

```
110 def moleculesInRange(min, max, frequencies):
111     for freq in frequencies["Ordered Freq (GHz)"].tolist():
112         if min < freq < max:
113             return True
114     return False
115 
116 # turns an array of booleans to an array of indices to remove (False =
    Remove)
117 def trueFalseToIndex(TrueFalse):
118     indicies = []
119     for i in range(len(TrueFalse)):
120         if not TrueFalse[i]:
121             indicies.append(i)
122     return indicies
123 
124 
125 # Same as alminer.download_data except it ignores individual files larger
    than {maxSize} GB
126 def download_data2(observations, fitsonly=False, dryrun=False, print_urls=
    False, filename_must_include='',
127                    location='./data', frequencies=[], maxSize=20):
128     """
129     Download ALMA data from the archive to a location on the local machine.
130     Parameters
131     ----------
132     observations : pandas.DataFrame
133         This is likely the output of e.g. 'conesearch', 'target', 'catalog
    ', & 'keysearch' functions.
134     fitsonly : bool, optional
135         (Default value = False)
136         Download individual fits files only (fitsonly=True). This option
    will not download the raw data
137         (e.g. 'asdm' files), weblogs, or README files.
138     dryrun : bool, optional
139         (Default value = False)
140         Allow the user to do a test run to check the size and number of
    files to download without actually
141         downloading the data (dryrun=True). To download the data, set
    dryrun=False.
142     print_urls : bool, optional
143         (Default value = False)
144         Write the list of urls to be downloaded from the archive to the
    terminal.
145     filename_must_include : list of str, optional
146         (Default value = '')
147         A list of strings the user wants to be contained in the url
    filename. This is useful to restrict the
148         download further, for example, to data that have been primary beam
    corrected ('.pbcor') or that have
```

```
149         the science target or calibrators (by including their names). The
    choice is largely dependent on the
150         cycle and type of reduction that was performed and data products
    that exist on the archive as a result.
151         In most recent cycles, the science target can be filtered out with
    the flag '_sci' or its ALMA target name.
152     location : str, optional
153         (Default value = ./data)
154         directory where the downloaded data should be placed.
155     frequencies: dataframe
156         Dataframe of frequencies from splatalogue as given by
    alminer_extentions.get_frequencies()
157     maxSize : float
158         (Default value = 20)
159         The maximum file size of a single file in GB.
160 
161     """
162     print("================================")
163     # we use astroquery to download data
164     myAlma = Alma()
165     default_location = './data'
166     myAlma.cache_location = default_location
167     # catch the case where the DataFrame is empty.
168     try:
169         if any(observations['data_rights'] == 'Proprietary'):
170             print("Warning: some of the data you are trying to download are
    still in the proprietary period and are "
171                   "not publicly available yet.")
172             observations = observations[observations['data_rights'] == '
    Public']
173         uids_list = observations['member_ous_uid'].unique()
174         # when len(uids_list) == 0, it's because the DataFrame included only
     proprietary data and we removed them in
175         # the above if statement, so the DataFrame is now empty
176         if len(uids_list) == 0:
177             print("No data to download. Check the input DataFrame. It is
    likely that your query results include only "
178                   "proprietary data which cannot be freely downloaded.")
179             return
180     # this is the case where the query had no results to begin with.
181     except TypeError:
182         print("No data to download. Check the input DataFrame.")
183         return
184     # change download location if specified by user, else the location will
    be the astrquery cache location
185     if location != default_location:
186         if os.path.isdir(location):
187             myAlma.cache_location = location
188         else:
```

```
189             print("{} is not a directory. The download location will be set
    to {}".format(location, default_location))
190             myAlma.cache_location = default_location
191      if fitsonly:
192          data_table = Alma.get_data_info(uids_list, expand_tarfiles=True)
193          # filter the data_table and keep only rows with "fits" in '
    access_url' and the strings provided by user
194          # in 'filename_must_include' parameter
195          dl_table = data_table[[i for i, v in enumerate(data_table['
    access_url']) if v.endswith(".fits") and
196                                 all(i in v for i in filename_must_include)]]
197          #dl_table.pprint_all()
198
199          # General idea is to check the mfs file to be able to map spw to
    frequencies so that we can filter out and only
200          # download cube files of the transitions we are interested in.
201          oldLength = len(dl_table)
202          if ".cube" in filename_must_include and len(frequencies) > 0:
203              UIDquery = alminer.keysearch({'member_ous_uid': [uids_list[0]]})
204              alminer.download_data(UIDquery, fitsonly=True,
    filename_must_include=["_sci", ".pbcor", ".mfs"],
205                                    location=location)
206              i=0
207              spwToRestFreq = []
208              for filename in os.listdir(location):
209                  if filename.__contains__(".mfs"):
210                      with fits.open(location + "/" + filename) as hdul:
211                          h = hdul[0].header
212                          freq = h.get("RESTFRQ")
213                          spw = h.get("SPW")
214                          spwToRestFreq.append([spw, freq,i])
215                          i = i+1
216                          del hdul[0].data
217
218              spwToRestFreq = np.array(sorted(spwToRestFreq, key = lambda x: x
    [1])) # sort according to frequencies
219
220              newIndex = spwToRestFreq[:,2] # spw order
221
222              UIDquery = UIDquery.sort_values(by=['min_freq_GHz'])
223              UIDquery = UIDquery.reset_index()
224
225              UIDquery["newIndex"] = newIndex
226              UIDquery = UIDquery.set_index("newIndex")
227              UIDquery = UIDquery.sort_values(by=["newIndex"]) # sort the
    query in "spw order".
228
229              #UIDquery.to_csv("query2.csv")
230              trueFalse = [moleculesInRange(minFreq, maxFreq, frequencies) for
```

```
    minFreq, maxFreq in
231                         zip(UIDquery['min_freq_GHz'], UIDquery['
    max_freq_GHz'])]
232             dl_table = dl_table[trueFalse]
233 
234         dl_table = dl_table[dl_table['content_length'] < maxSize * 1e9] #
    filter on file size (hard to handle too large files)
235         dl_link_list = dl_table['access_url'].tolist()
236         # keep track of the download size and number of files to download
237         dl_size = dl_table['content_length'].sum() / 1E9
238         dl_files = len(dl_table)
239         print("Original number of files:" +oldLength + ". Reduced number of
    files: " + dl_files+".")
240         if dryrun:
241             print("This is a dryrun. To begin download, set dryrun=False.")
242             print("================================")
243         else:
244             print("Starting download. Please wait...")
245             print("================================")
246             myAlma.download_files(dl_link_list, cache=True)
247     else:
248         data_table = Alma.get_data_info(uids_list, expand_tarfiles=False)
249         dl_link_list = data_table['access_url'].tolist()
250         # keep track of the download size and number of files to download
251         dl_size = data_table['content_length'].sum() / 1E9
252         dl_files = len(data_table)
253         if dryrun:
254             print("This is a dryrun. To begin download, set dryrun=False.")
255             print("================================")
256         else:
257             print("Starting download. Please wait...")
258             print("================================")
259             myAlma.retrieve_data_from_uid(uids_list, cache=True)
260     print("Download location = {}".format(myAlma.cache_location))
261     print("Total number of Member OUSs to download = {}".format(len(
    uids_list)))
262     print("Selected Member OUSs: {}".format(uids_list.tolist()))
263     print("Number of files to download = {}".format(dl_files))
264     if dl_size > 1000.:
265         print("Needed disk space = {:.1f} TB".format(dl_size / 1000.))
266     elif dl_size < 1.:
267         print("Needed disk space = {:.1f} MB".format(dl_size * 1000.))
268     else:
269         print("Needed disk space = {:.1f} GB".format(dl_size))
270     if print_urls:
271         print("File URLs to download = {}".format("\n".join(dl_link_list)))
272     print("--------------------------------")
```

D

FittingData.py

```
from fileinput import filename
import numpy as np
import matplotlib.pyplot as plt
from astropy.io import fits
from alminer_extensions import get_molecule, get_frequencies
import scipy.optimize as opt
from astropy.wcs import WCS
import warnings
import os
from heapq import heappop, heappush, heapify

warnings.filterwarnings("ignore")

freqs = get_frequencies('molecules.csv')

################################################
# Math and helper functions
################################################

def twoDimensionalEllipticalGauss(xDataTuple, amplitude, xCenter, yCenter, sigmaX, sigmaY, theta, offset):
    """
    Evaluates the two-dimensional Gaussian function at some coordinates given the parameters.
    ----------
    xDataTuple : tuple of (list of) floats
         The coordinates where the function is evaluated
    amplitude : float
         The amplitude of the Gaussian
    xCenter : float
         The x-coordinate for the center point of the Gaussian
    yCenter : float
         The y-coordinate for the center point of the Gaussian
    sigmaX : float
         The standard deviation of the Gaussian in the x-direction
    sigmaY : float
         The standard deviation of the Gaussian in the y-direction
```

```
36      theta : float
37          The counterclockwise rotation of the Gaussian
38      offset : float
39          The offset of the Gaussian (shift in the z-direction)
40      Returns
41      -------
42      A 1-dimensional array containing the function evaluated at every given
        coordinate, in row-major order.
43      """
44      x, y = xDataTuple
45      xCenter = float(xCenter)
46      yCenter = float(yCenter)
47      a = np.cos(theta) ** 2 / (2 * sigmaX ** 2) + np.sin(theta) ** 2 / (2 *
        sigmaY ** 2)
48      b = -np.sin(2 * theta) / (4 * sigmaX ** 2) + np.sin(2 * theta) / (4 *
        sigmaY ** 2)
49      c = np.sin(theta) ** 2 / (2 * sigmaX ** 2) + np.cos(theta) ** 2 / (2 *
        sigmaY ** 2)
50      z = offset + amplitude * np.exp(- (a * ((x - xCenter) ** 2) + 2 * b * (x
        - xCenter) * (y - yCenter)
51                                          + c * ((y - yCenter) ** 2)))
52      return z.ravel()
53
54
55  def oneDimensionalGaussian(x, amplitude, center, sigma, offset=0):
56      """
57      Evaluates the one dimensional Gaussian function at a value x given
        parameters.
58      ----------
59      x : float
60          The location to compute the function.
61      amplitude : float
62          The amplitude of the gaussian(max value)
63      center : float
64          The center of the gaussian(location of max)
65      sigma : float
66          The standard deviation of the gaussian(in "space")
67      offset : float
68          The offset of the gaussian(shift in y-direction)
69      Returns
70      -------
71      The value of the specified gaussian at point x.
72      """
73      return offset + abs(amplitude) * np.exp(-(x - center) ** 2 / (2 * sigma
        ** 2))
74
75
76  def sumOfTwoGauss(x, amplitude1, center1, sigma1, offset1, amplitude2,
        center2, sigma2, offset2):
```

```
77      """
78      Evaluates the sum of two one dimensional Gaussians at a point x given
        parameters
79      ----------
80      x : float
81          The location to compute the function.
82      amplitude : float
83          The amplitude of the gaussian(max value)
84      center : float
85          The center of the gaussian(location of max)
86      sigma : float
87          The standard deviation of the gaussian(in "space")
88      offset : float
89          The offset of the gaussian(shift in y-direction)
90      Returns
91      -------
92      The value of the specified gaussian at point x.
93      """
94      return oneDimensionalGaussian(x, amplitude1, center1, sigma1, offset1) +
        oneDimensionalGaussian(x, amplitude2,
95
                               center2, sigma2,
96
                               offset2)
97
98
99  def sumOfThreeGauss(x, a1, c1, s1, o1, a2, c2, s2, o2, a3, c3, s3, o3):
100     """
101     Evaluates the sum of three one dimensional Gaussians (where the third
        has negative amplitude) at a point x given parameters
102     ----------
103     x : float
104         The location to compute the function.
105     a : float
106         The amplitude of the gaussian(max value)
107     c : float
108         The center of the gaussian(location of max)
109     s : float
110         The standard deviation of the gaussian(in "space")
111     o : float
112         The offset of the gaussian(shift in y-direction)
113     Returns
114     -------
115     The value of the specified gaussian at point x.
116     """
117     return oneDimensionalGaussian(x, a1, c1, s1, o1) +
        oneDimensionalGaussian(x, a2, c2, s2,
118
          o2) - oneDimensionalGaussian(x, a3, c3,
```

```
                                    s3, o3)


def linFunc(x, k, m):
    """Computes value of the line function y = kx+m at x value"""
    return k * x + m


def fitWrapper(coeffs, *args):
    """Wrapper function that allows us to weight a line function"""
    xdata, ydata, prio = args
    return prio * (linFunc(xdata, *coeffs) - ydata)


def clearPlots(plotIndicies):
    """Clears and closes all plots"""
    for i in plotIndicies:
        plt.figure(i)
        plt.clf()
        plt.cla()
        plt.close()


def rms(matrix):
    """Computes the quadratic mean"""
    vals = np.ravel(matrix)
    rootmeansquared = np.sqrt(np.nanmean(vals ** 2))
    return rootmeansquared


def computeNoise(moment,partitions=8):
    """
    Computes the noise in a moment map by subdividing the map and computing the average noise in each submap.
    ----------
    moment : matrix
        The moment map matrix
    partitions : integer
        The side length for the grid, i.e. we get a (partitions x partitions) grid
    Returns
    -------
    """
    means = []
    imageWidth = moment.shape[0]
    for submatrix in split(moment, imageWidth // partitions, imageWidth // partitions):
        submatrix = submatrix[~np.isnan(submatrix)]
```

```
164         if len(submatrix) < 5:
165             continue
166         means.append(rms(submatrix))
167     return np.min(means)
168 
169 
170 def split(array, nrows, ncols):
171     """Helper method that splits a matrix into sub-matrices."""
172 
173     r, h = array.shape
174     return (array.reshape(h // nrows, nrows, -1, ncols)
175             .swapaxes(1, 2)
176             .reshape(-1, nrows, ncols))
177 
178 ##############################################
179 # Data fitting
180 ##############################################
181 
182 def fit2DGaussianToContData(filename, createPlot=False,
    plotDistanceFromCenter=10):
183     """
184     Fits a two-dimensional Gaussian function to continuum data.
185     ----------
186     filename : String
187          The location of the continuum data fits file.
188     createPlot : bool, optional
189          (Default value = False)
190          Plots the gaussian fit to the continuum data.
191     plotDistanceFromCenter : float, optional
192          (Default value = 10)
193          Determines how far out from the center the bounds of the plot are.
194     Returns
195     -------
196     The fitted parameters to the Gaussian function.
197     """
198     with fits.open(filename) as hdul:
199         fitsData = hdul[0].data
200         contMatrix = np.squeeze(fitsData)  # Transforms matrix into correct
    shape
201         imageWidth = contMatrix.shape[0]
202         contMatrix = np.nan_to_num(contMatrix)
203         x = np.linspace(0, imageWidth, imageWidth)
204         y = np.linspace(0, imageWidth, imageWidth)
205         x, y = np.meshgrid(x, y)
206         initialGuess = [np.max(contMatrix), imageWidth / 2, imageWidth / 2,
    1, 1, 0, 1]
207         fittedValues, _ = opt.curve_fit(twoDimensionalEllipticalGauss, (x, y
    ), contMatrix.ravel(), p0=initialGuess)
208         if createPlot:
```

```
209             fittedData = twoDimensionalEllipticalGauss((x, y), *fittedValues
    )
210             plt.figure(1)
211             wcs = WCS(filename)
212             if wcs.naxis > 2:
213                 wcs = wcs.sub(2)
214             plt.subplot(projection=wcs)
215             plt.imshow(contMatrix, origin='lower')
216             plt.colorbar(label=r"Intensity (Jy beam$^{-1}$)")
217             plt.contour(x, y, fittedData.reshape(imageWidth, imageWidth), 2,
     colors="w")
218             plt.xlabel("Right Ascension (J2000)")
219             plt.ylabel("Declination (J2000)")
220             plt.title(hdul[0].header.get("OBJECT").replace("_", " ") + "
    Continuum")
221             plt.axis([fittedValues[1] - plotDistanceFromCenter *
    fittedValues[3],  # Show plot in region around max
222                       fittedValues[1] + plotDistanceFromCenter *
    fittedValues[3],
223                       fittedValues[2] - plotDistanceFromCenter *
    fittedValues[4],
224                       fittedValues[2] + plotDistanceFromCenter *
    fittedValues[4]])
225             plt.savefig(filename.split(".cont")[0] + "_contFit" + ".pdf")
226         del hdul[0].data
227     return fittedValues
228
229 def getPointsWithinGaussian(fittedValues, proportionOfMaximum=1 / 2,
    distanceFromCenter=10, createPlot=False):
230     """
231     Finds the points which are in an ellipse of "given size"
232     ----------
233     fittedValues : list of floats
234         List of fitted parameters to the Gaussian function.
235     proportionOfMaximum : float, optional
236         (Default value = 1/2 {FWHM} )
237         Cutoff for size of ellipse
238     distanceFromCenter: float, optional
239         (Default value = 10)
240         How far from the center to search, (to skip iterating over all 1000
    x1000 pixels)
241     createPlot : bool, optional
242         (Default value = False)
243         Plots the points.
244     Returns
245     -------
246     A list of all coordinates contained within an ellipse of given size.
247     """
248     fMax = fittedValues[0] + fittedValues[6]
```

```
249     xCenter = fittedValues[1]
250     yCenter = fittedValues[2]
251     xSigma = np.abs(fittedValues[3])
252     ySigma = np.abs(fittedValues[4])
253     coordinatesInEllipse = []
254     for i in range(int(xCenter - distanceFromCenter * xSigma), int(xCenter +
        distanceFromCenter * xSigma)):
255         for j in range(int(yCenter - distanceFromCenter * ySigma), int(
       yCenter + distanceFromCenter * ySigma)):
256             if twoDimensionalEllipticalGauss((i, j), *fittedValues) > fMax *
        proportionOfMaximum:
257                 coordinatesInEllipse.append([i, j])
258     if createPlot:
259         x, y = np.array(coordinatesInEllipse).T
260         plt.scatter(x, y, c='black')
261     return coordinatesInEllipse
262
263
264 def oneDGaussianMeanFit(means, createPlot=False):
265     """
266     Fits a one dimensional Gaussian to a list of values
267     ----------
268     means : list of floats
269         The values to fit the Gaussian to.
270     createPlot : bool, optional
271          (Default value = False)
272          Decides if the fit is plotted
273     Returns
274     -------
275     The parameters of the fitted Gaussian.
276     """
277     n = len(means)
278     x = np.linspace(1, n, n)
279     sigma = len(means) / 30
280     initialGuess = [np.max(means), np.argmax(means), sigma]
281     parameters, _ = opt.curve_fit(oneDimensionalGaussian, x, means, p0=
       initialGuess)
282     if createPlot:
283         plt.figure(8)
284         plt.plot(x, means, 'b+:', label='data')
285         plt.plot(x, oneDimensionalGaussian(x, *parameters), 'ro:', label='
       fit')
286     return parameters
287
288
289 def bimodalGaussianMeanFit(means, createPlot=False, createSubPlot=False):
290     """
291     Fits the sum of two one dimensional Gaussians to a list of values
292     ----------
```

```
293     means : list of floats
294         The values to fit the Gaussian to.
295     createPlot : bool, optional
296          (Default value = False)
297          Decides if the fit is plotted
298     createSubPlot : bool, optional
299          (Default value = False)
300          Decides if the subfit is plotted
301     Returns
302     -------
303     The parameters of the fitted Gaussians.
304     """
305     n = len(means)
306     x = np.linspace(1, n, n)
307     sigma = oneDGaussianMeanFit(means, createSubPlot)[2]
308     initialGuess = [np.max(means), np.argmax(means), sigma, 0.001, np.max(
    means), np.argmax(means), sigma,
309                     0.001]
310     lowerBounds = [-2 * np.max(means), 0, -len(means), -1, -2 * np.max(means
    ), 0, -len(means), -1]
311     upperBounds = [2 * np.max(means), len(means), len(means), 1, 2 * np.max(
    means), len(means), len(means), 1]
312     parameters, _ = opt.curve_fit(sumOfTwoGauss, x, means, p0=initialGuess,
    bounds=(lowerBounds, upperBounds))
313     if createPlot:
314         plt.figure(9)
315         plt.plot(x, means, 'b+:', label='data')
316         plt.plot(x, sumOfTwoGauss(x, *parameters), 'ro:', label='fit')
317     return parameters
318 
319 
320 def trimodalGaussianMeanFit(means, createPlot=False, createSubPlot=False):
321     """
322     Fits the sum of three one dimensional Gaussians to a list of values
323     ----------
324     means : list of floats
325         The values to fit the Gaussian to.
326     createPlot : bool, optional
327          (Default value = False)
328          Decides if the fit is plotted
329     createSubPlot : bool, optional
330          (Default value = False)
331          Decides if the subfit is plotted
332     Returns
333     -------
334     The parameters of the fitted Gaussians.
335     """
336     n = len(means)
337     x = np.linspace(1, n, n)
```

```
338     sigma = oneDGaussianMeanFit(means, createSubPlot)[2]
339     initialGuess = [np.max(means), np.argmax(means), sigma, 0.001, np.max(
    means), np.argmin(means) + sigma, sigma,
340                     0.001, np.min(means), np.argmin(means), sigma, 0.001]
341     lowerBounds = [-2 * np.max(means), 0, -len(means), -1, -2 * np.max(means
    ), 0, -len(means), -1, -2 * np.max(means),
342                    0, -len(means), -1]
343     upperBounds = [2 * np.max(means), len(means), len(means), 1, 2 * np.max(
    means), len(means), len(means), 1,
344                    2 * np.max(means), len(means), len(means), 1]
345     parameters, _ = opt.curve_fit(sumOfThreeGauss, x, means, p0=initialGuess
    , bounds=(lowerBounds, upperBounds))
346     if createPlot:
347         plt.figure(10)
348         plt.plot(x, means, 'b+:', label='data')
349         plt.plot(x, sumOfThreeGauss(x, *parameters), 'ro:', label='fit')
350     return parameters
351 
352 
353 ##############################################
354 # Intensity Profiles and Ranges
355 ##############################################
356 
357 def meanSpectralProfile(filename, coordinates, createPlot=False):
358     """
359     Computes the mean intensity in the fitted region for each frequency
360     ----------
361     filename : String
362          The location of the cube data fits file.
363     coordinates : ints
364          List of coordinates within the given region.
365     createPlot : bool, optional
366          (Default value = False)
367          Plots the spectral profile.
368     Returns
369     -------
370     A list of the mean values for each frequency.
371     """
372     with fits.open(filename) as hdul:
373         cube = hdul[0].data
374         cubeData = np.squeeze(cube)
375         means = []
376         for i in range(0, cubeData.shape[0]):
377             mean = 0
378             for x, y in coordinates:
379                 mean += cubeData[i, x, y]
380             mean = mean / len(coordinates)
381             means.append(mean)
382 
```

```
383         rootMean = rms(means)
384         print(rootMean)
385 
386         if createPlot:
387             header = hdul[0].header
388             x = np.linspace(header.get("CRVAL3"), header.get("CRVAL3") +
    header.get("CDELT3") * cubeData.shape[0],
389                             cubeData.shape[0])
390             if header.get("CDELT3") < 0:
391                 x = np.flip(x)
392             plt.figure(3)
393             plt.plot(x, means)
394             plt.axhline(rootMean)
395             col = "k"
396             plt.axvline(x=header.get("RESTFRQ"), color=col)
397             plt.xlabel("Frequency (Hz)")
398             plt.ylabel(r"Mean intensity (Jy beam$^{-1}$)")
399             plt.title(hdul[0].header.get("OBJECT").replace("_", " ") + "
    Spectral Profile")
400             plt.savefig(filename.split(".cube")[0] + "_spectralProfile" + ".
    pdf")
401         del hdul[0].data
402     return means
403 
404 
405 def findRangesByInflection(means, filename, createPlot=False):
406     """
407     Computes inflection points of the spectral profile and extracts ranges.
    Needs some work.
408     ----------
409     means : list of floats
410         The mean values for each frequency
411     createPlot : bool, optional
412         (Default value = False)
413         Plots the inflection points
414     Returns
415     -------
416     A list of start and end values for use as bounds.
417     """
418     interpolatedMeans = sumOfThreeGauss(np.linspace(0, len(means), len(means
    )), *trimodalGaussianMeanFit(means))
419     normalisedSquaredError = np.mean(((interpolatedMeans - means) / np.max(
    means)) ** 2)  # rms kanske istället
420     print("error: ", normalisedSquaredError)
421     if normalisedSquaredError > 0.05:
422         return "break"
423 
424     interpolatedMeansDerivative = np.gradient(interpolatedMeans)
425     interpolatedMeans2ndDerivative = np.gradient(interpolatedMeansDerivative
```

```
    )
    # add small number to avoid float precision errors when approaching zero
    inflectionPoints = np.where(np.diff(np.sign(
interpolatedMeans2ndDerivative + 1e-18)))[0]
    extremumPoints = np.where(np.diff(np.sign(interpolatedMeansDerivative +
1e-18)))[0]

    # Finds smallest local minimum in the region between the two largest
local maximums
    minPoints = {}
    maxPoints = {}
    for extremum in extremumPoints:
        if interpolatedMeans2ndDerivative[extremum] > 0:
            minPoints[extremum] = interpolatedMeans[extremum]
        elif interpolatedMeans2ndDerivative[extremum] < 0:
            maxPoints[extremum] = interpolatedMeans[extremum]

    sortedMax = dict(sorted(maxPoints.items(), key=lambda item: item[1]))
    twoLargestMaxima = sorted(list(sortedMax)[-2:])
    sortedMin = dict(sorted(minPoints.items(), key=lambda item: item[1]))
    for point in sortedMin:
        if twoLargestMaxima[1] >= point >= twoLargestMaxima[0]:
            minPoint = point
            break

    lower = inflectionPoints[inflectionPoints <= minPoint]
    upper = inflectionPoints[inflectionPoints >= minPoint]
    if len(upper) == 1 or len(lower) == 1:
        raise Exception("Not enough inflection points")

    if createPlot:
        plt.figure(4)
        x = np.linspace(0, len(means) - 1, len(means))
        plt.plot(x, means, 'ro:')
        plt.plot(x, interpolatedMeans)
        for inflectionPoint in inflectionPoints:
            1 + 1
        for extremum in extremumPoints:
            plt.axvline(x=extremum, color='k')
        for minPoint in minPoints:
            plt.axvline(x=minPoint, color='g')
        with fits.open(filename) as hdul:
            plt.title(hdul[0].header.get("OBJECT").replace("_", " ") + "
Ranges")
            plt.savefig(filename.split(".cube")[0] + "_ranges" + ".pdf")
            del hdul[0].data
    return [lower[0], lower[-1], upper[0], upper[-1]]


```

```
470 def findRangesByGaussianFit(means, createPlot=False, createSubPlot=False,
    modality=2, sigmaMult=1):
471     """
472     Finds the ranges to compute moment maps from by fitting Gaussians. (Does
     not work well for modality = 3)
473     ----------
474     means : list of floats
475         The values to fit the Gaussian to.
476     createPlot : bool, optional
477          (Default value = False)
478          Decides if the fit is plotted
479     createSubPlot : bool, optional
480          (Default value = False)
481          Decides if the subfit is plotted
482     modality : integer
483          (Default value = 2)
484          How many Gaussians to fit
485     sigmaMult : float
486          (Default value = 1)
487          How many standard deviations from the peaks to include in the range
    .
488     Returns
489     -------
490     The ranges to compute momentmaps from.
491     """
492     if modality == 2:
493         parameters = bimodalGaussianMeanFit(means, createPlot, createSubPlot
    )
494     elif modality == 3:
495         parameters = trimodalGaussianMeanFit(means, createPlot,
    createSubPlot)
496     else:
497         raise Exception("Unsupported modality of Gaussian")
498     center1 = parameters[1]
499     center2 = parameters[5]
500     sigma1 = abs(parameters[2])
501     sigma2 = abs(parameters[6])
502     ranges = [int(center1 - sigmaMult * sigma1), int(center1 + sigmaMult *
    sigma1), int(center2 - sigmaMult * sigma2),
503               int(center2 + sigmaMult * sigma2)]
504     return ranges
505
506
507 def findRangesByRMS(means):
508     """Gets the indicies of all intensities larger than the rms in the
    spectral profile"""
509     rootMean = rms(means)
510     indicies = [i for i in range(len(means)) if means[i] > rootMean]
511     return indicies
```

```
512
513
514 ##############################################
515 # Moment maps
516 ##############################################
517
518 def computeMoments(filename, ranges, createPlot=False):
519     """
520     Computes red- and blueshifted moments given a datacube and ranges and
    joins blue- and rightshifted sides.
521     ----------
522     filename : String
523          The location of the cube data fits file.
524     ranges : list of floats
525          The start and endpoints of the ranges where moments are to be
    computed.
526     createPlot : bool, optional
527          (Default value = False)
528          "Plots" the moment map
529     Returns
530     -------
531     Two matricies with the "intensities" making up the blue- and redshifted
    moment maps.
532     """
533     with fits.open(filename) as hdul:
534         cube = hdul[0].data
535         cubeData = np.squeeze(cube)
536         cubeSlab1 = cubeData[ranges[0]:ranges[1], :, :]
537         cubeSlab2 = cubeData[ranges[2]:ranges[3], :, :]
538         moment1 = np.sum(cubeSlab1, axis=0)
539         moment2 = np.sum(cubeSlab2, axis=0)
540         if createPlot:
541             wcs = WCS(filename)
542             if wcs.naxis > 2:
543                 wcs = wcs.sub(2)
544             plt.figure(5)
545             plt.subplot(projection=wcs)
546             plt.imshow(moment1, origin='lower')
547             plt.colorbar(label=r"Integrated Intensity (Jy beam$^{-1}$ km s$
    ^{-1}$)")
548             plt.xlabel("Right Ascension (J2000)")
549             plt.ylabel("Declination (J2000)")
550             plt.title(hdul[0].header.get("OBJECT").replace("_", " ") + "
    Moment 0 Map")
551             plt.savefig(filename.split(".cube")[0] + "_moment1" + ".pdf")
552             plt.figure(6)
553             plt.subplot(projection=wcs)
554             plt.imshow(moment2, origin='lower')
555             plt.colorbar(label=r"Integrated Intensity (Jy beam$^{-1}$ km s$
```

```
    ^{-1}$)")
556             plt.xlabel("Right Ascension (J2000)")
557             plt.ylabel("Declination (J2000)")
558             plt.title(hdul[0].header.get("OBJECT").replace("_", " ") + "
    Moment 0 Map")
559             plt.savefig(filename.split(".cube")[0] + "_moment2" + ".pdf")
560         del hdul[0].data
561     return moment1, moment2
562
563
564 def maskedMoment(moment, factor=1):
565     """Sets all values lower than factor*noise to 0"""
566     moment[moment < factor * computeNoise(moment)] = 0
567     plt.figure()
568     plt.imshow(moment, origin='lower')
569     return moment
570
571
572 def computeMomentsByMax(filename, indicies=[]):
573     """Computes a "moment" by for each pixel summing the 10% largest pixels
    """
574     with fits.open(filename) as hdul:
575         cube = hdul[0].data
576         cubeData = np.squeeze(cube)
577
578         imageWidth = cubeData.shape[1]
579         moment = np.zeros((imageWidth, imageWidth))
580
581         for x in range(cubeData.shape[1]):
582             for y in range(cubeData.shape[1]):
583                 pixelVals = list(-1*cubeData[:, x, y])
584                 heapify(pixelVals)
585                 val = 0
586                 for i in range(cubeData.shape[0] // 10):
587                     val += -1*heappop(pixelVals)
588                 moment[x, y] = val
589
590         wcs = WCS(filename)
591         if wcs.naxis > 2:
592             wcs = wcs.sub(2)
593         plt.figure()
594         plt.subplot(projection=wcs)
595         plt.imshow(moment, origin='lower')
596         plt.colorbar(label=r"Integrated Intensity (Jy beam$^{-1}$ km s$^{-1}
    $)")
597         plt.xlabel("Right Ascension (J2000)")
598         plt.ylabel("Declination (J2000)")
599         plt.title(hdul[0].header.get("OBJECT").replace("_", " ") + " Moment
    0 Map (Max)")
```

```
600         plt.savefig(filename.split(".cube")[0] + "_maxmoment" + ".pdf")
601         del hdul[0].data
602     return moment
603 
604 
605 def computeMomentByIndex(filename, indicies, createPlot=True):
606     """Computes a moment by summing all frequency indicies"""
607     with fits.open(filename) as hdul:
608         cube = hdul[0].data
609         cubeData = np.squeeze(cube)
610         cubeSlab = cubeData[indicies, :, :]
611         moment1 = np.sum(cubeSlab, axis=0)
612         if createPlot:
613             wcs = WCS(filename)
614             if wcs.naxis > 2:
615                 wcs = wcs.sub(2)
616             plt.figure()
617             plt.subplot(projection=wcs)
618             plt.imshow(moment1, origin='lower')
619             plt.colorbar(label=r"Integrated Intensity (Jy beam$^{-1}$ km s$
    ^{-1}$)")
620             plt.xlabel("Right Ascension (J2000)")
621             plt.ylabel("Declination (J2000)")
622             plt.title(hdul[0].header.get("OBJECT").replace("_", " ") + "
    Moment 0 Map (Index)")
623             plt.savefig(filename.split(".cube")[0] + "_indexmoment" + ".pdf"
    )
624         del hdul[0].data
625     return moment1
626 
627 
628 ################################################
629 # Angle finding and contours
630 ################################################
631 
632 def refineContours(contours, xCenter, yCenter):
633     """
634     "Refines" contours by deleting all contours that do not surrond center
635     ----------
636     contours : contour object from plt.contour
637         The contour object from plt.contour
638     xCenter : float
639         The x-coordinate for the center point of the disc
640     xCenter : float
641         The y-coordinate for the center point of the disc
642     Returns
643     -------
644     """
645 
```

```
646     # First iteration to save all contours that surround center of disc
647     contoursAroundCenter = []
648     for level in contours.collections:
649         for kp, path in reversed(list(enumerate(level.get_paths()))):  #
    loop in reverse since deletions
650             if path.contains_point((xCenter, yCenter)):
651                 contoursAroundCenter.append([level, path])
652 
653     # Second iteration to remove all contours that are not within the above
    contours
654     for level in contours.collections:
655         for kp, path in reversed(list(enumerate(level.get_paths()))):
656             if not path.contains_point((xCenter, yCenter)):
657                 isWithin = False
658                 for _, bigPath in contoursAroundCenter:
659                     if bigPath.contains_path(path):
660                         isWithin = True
661                 if not isWithin:
662                     del (level.get_paths()[kp])
663 
664     plt.gcf().canvas.draw()  # uppdatera plotten
665     return contours
666 
667 
668 def plotContours(moment1, moment2, fittedValues, filename, combinedPlot=True
    ):
669     """
670     Plots the wanted contours by first removing noise, computing the
    contours and then refining them.
671     ----------
672     moment1 : matrix
673         The first moment map matrix
674     moment2 : matrix
675         The second moment map matrix
676     fittedValues : list of floats
677         List of fitted parameters to the Gaussian function.
678     combinedPlot : bool, optional
679          (Default value = True)
680          Plot both contours in same plot
681     Returns
682     -------
683     """
684     mmom1 = maskedMoment(moment1)
685     mmom2 = maskedMoment(moment2)
686     x, y = fittedValues[1:3]
687     plt.figure(1337)
688     contours1 = plt.contour(mmom1)
689     refineContours(contours1, x, y)
690     if not combinedPlot:
```

```
691         plt.figure()
692     contours2 = plt.contour(mmom2)
693     refineContours(contours2, x, y)
694     findAngleFromContour(contours1, contours2, fittedValues, filename)
695 
696 
697 def findAngleOfOutflow(moment, fittedValues,filename, extra="", coordinates=[], useDistance=True):
698     """
699     Find and plots the directions where intensities are present.
700     ----------
701     moment : matrix
702         The moment map matrix
703     fittedValues : list of floats
704         List of fitted parameters to the Gaussian function.
705     coordinates : list of list of ints
706         (Default value: [] (i.e. none))
707         Coordinates to ignore when calculating.
708     useDistance : boolean
709         (Default value = False)
710         Whether or not to weight by distance (in the sense that intensities closer to the disc have more weight)
711 
712     Returns
713     -------
714     """
715     xCenter, yCenter = fittedValues[1:3]
716     imageWidth = moment.shape[0]
717     angularIntensities = {}
718     noise = computeNoise(moment, partitions=20)
719     for i in range(-180, 180):
720         angularIntensities[i] = 0
721     newMoment = np.zeros((imageWidth, imageWidth))
722     for x in range(0, imageWidth):
723         for y in range(0, imageWidth):
724             if len(coordinates) == 0 or [x, y] not in coordinates:
725                 if moment[x, y] > noise:
726                     distanceFactor = 0
727                     if useDistance:
728                         distanceFactor = ((x - xCenter) ** 2 + (y - yCenter) ** 2) / ((imageWidth * 0.5) ** 2)
729                         if distanceFactor > 1:
730                             continue
731                     index = np.floor(np.arctan2(x - xCenter, y - yCenter) * 180 / np.pi)
732                     angularIntensities[index] += moment[x, y] * (1 - distanceFactor)
733                     angularIntensities[index + 180 if index < 0 else -180 + ((180 + index) % 180)] += moment[x, y] * (
```

```
                                1 - distanceFactor)
                    newMoment[x, y] = moment[x, y]*(1-distanceFactor)

    with fits.open(filename) as hdul:
        plt.figure()
        plt.imshow(newMoment, origin="lower")
        plt.scatter(xCenter, yCenter)
        plt.title(hdul[0].header.get("OBJECT").replace("_", " ") + " Reduced Moment")
        plt.savefig(filename.split(".cube")[0] + "_redmoment" + extra + ".pdf")
        plt.figure()
        ax = plt.subplot(111, polar=True)
        bars = ax.bar((np.array(list(angularIntensities.keys()))) * np.pi / 180, angularIntensities.values(),
                      width=12 * np.pi / 180)
        plt.title(hdul[0].header.get("OBJECT").replace("_", " ") + " Angular intensities")
        plt.savefig(filename.split(".cube")[0] + "_angles" + extra + ".pdf")
        del hdul[0].data

# Finds the angle given contours
def findAngleFromContour(contour1, contour2, fittedValues, filename):
    """From two contours find an angle."""
    xCenter, yCenter = fittedValues[1:3]
    coordinates = []
    for level in contour1.collections: #adds all points of the contours to an array.
        for path in level.get_paths():
            verts = np.array(path.vertices)
            n = len(verts)
            verts = verts[1::int(np.ceil(n / 8))]
            for x, y in verts:
                coordinates.append([x, y])
    for level in contour2.collections:
        for path in level.get_paths():
            verts = np.array(path.vertices)
            n = len(verts)
            verts = verts[1::int(np.ceil(n / 8))]
            for x, y in verts:
                coordinates.append([x, y])

    if len(coordinates) < 10:
        return

    coordinates = np.array(coordinates)
    distanceFromCenter = (coordinates[:, 0] - xCenter) ** 2 + (coordinates[:, 1] - yCenter) ** 2
    prio = np.ceil(100 * distanceFromCenter / np.max(distanceFromCenter))
```

```
777     coordinates[0, :] = [xCenter, yCenter]
778     prio[0] = 100 * 100 # prioritize center
779     res = opt.least_squares(fitWrapper, x0=[1, 100], args=(coordinates[:,
    0], coordinates[:, 1], prio))
780     coeff = res.x
781 
782     # angle of contours
783     angle = np.arctan(coeff[0])
784     if angle < 0:
785         angle += np.pi
786 
787     #angle of cont fit
788     clockWiseContRotation = fittedValues[5] * 180 / np.pi % 360
789     if fittedValues[4] > fittedValues[3]:
790         clockWiseContRotation = (clockWiseContRotation + 90) % 360
791     k = np.tan(-clockWiseContRotation * np.pi / 180) # finds slope of cont
    line
792     m = yCenter - k * xCenter
793     print(fittedValues[3], fittedValues[4])
794 
795     angleBetweenLines = np.arctan((coeff[0] - k) / (1 + coeff[0] * k)) #
    finds the angle between the lines
796     print(np.abs(angleBetweenLines * 180 / np.pi))
797     with fits.open(filename) as hdul:
798         imageWidth = np.squeeze(hdul[0].data).shape[1]
799         plt.figure(1337)
800         x = np.linspace(0, imageWidth, 500)
801         y1 = np.polyval(coeff, x)
802         plt.plot(x, y1, 'k')
803         plt.plot(x, linFunc(x, k, m))
804         plt.xlim([0, imageWidth])
805         plt.ylim([0, imageWidth])
806         plt.title(hdul[0].header.get("OBJECT").replace("_", " ") + "
    Contours")
807         plt.savefig(filename.split(".cube")[0] + "_contours" + ".pdf")
808         plt.figure(1)
809         plt.plot(x, linFunc(x, k, m))
810         del hdul[0].data
811 
812 
813 ##############################################
814 # Complete analysis from fits files
815 ##############################################
816 
817 def findMoment(contFile, cubeFile, oneMillionPlots=False):
818     """
819     Finds and plots the momentmap and contours
820     ----------
821     contFile : file
```

```
822         The continuum file of the observation
823     cubeFile : file
824         Datacube file from the observation
825     oneMillionPlots : bool, optional
826          (Default value = False)
827          Whether to plot all the plots or not
828     Returns
829     -------
830     """
831     clearPlots([1, 2, 3, 4, 5, 6, 1337])
832     fittedValues = fit2DGaussianToContData(contFile, oneMillionPlots)
833     coordinates = getPointsWithinGaussian(fittedValues, 1 / 15, 20, False)
834     means = meanSpectralProfile(cubeFile, coordinates, oneMillionPlots)
835     ranges = findRangesByInflection(means, cubeFile, oneMillionPlots)
836     moment1, moment2 = computeMoments(cubeFile, ranges, oneMillionPlots)
837     plt.figure()
838     plt.contour(moment1)
839     plt.figure()
840     plotContours(moment1, moment2, fittedValues, cubeFile)
841 
842     plt.show()
843 
844 
845 def findMoment2(fittedValues, coordinates, cubeFile, oneMillionPlots=True):
846     """Helper function to analyse cube files given cont files"""
847     means = meanSpectralProfile(cubeFile, coordinates, oneMillionPlots)
848     ranges = findRangesByInflection(means, cubeFile, oneMillionPlots)
849     if ranges == "break":
850         return
851     moment1, moment2 = computeMoments(cubeFile, ranges, oneMillionPlots)
852     plotContours(moment1, moment2, fittedValues, cubeFile)
853 
854 def analyseDir(dir, allPlots=True):
855     """Analyses an entire directory of fits files"""
856     cubeFiles = []
857     contFiles = []
858     for filename in os.listdir(dir):
859         if filename.__contains__(".cont") and filename.__contains__(".fits")
    :
860             contFiles.append(dir + "/" + filename)
861         elif filename.__contains__(".cube") and filename.__contains__(".fits
    "):
862             cubeFiles.append(dir + "/" + filename)
863 
864     if len(contFiles) == 0:
865         return
866 
867     for contFile in contFiles:
868         fittedValues = fit2DGaussianToContData(contFile, allPlots)
```

```
869         coordinates = getPointsWithinGaussian(fittedValues, 1 / 15, 20,
    False)
870         for cubeFile in cubeFiles:
871             if cubeFile.__contains__(contFile.split("_sci")[0]):
872                 clearPlots([1, 2, 3, 4, 5, 6, 1337])
873                 try:
874                     findMoment2(fittedValues, coordinates, cubeFile,
    allPlots)
875                 except:
876                     analasys_failed += 1
877                     print("exception occured :(")
878     print("done")
879 
880 def analyseDir2(dir):
881     cubeFiles = []
882     contFiles = []
883     for filename in os.listdir(dir):
884         if filename.__contains__(".cont") and filename.__contains__(".fits")
    :
885             contFiles.append(dir + "/" + filename)
886         elif filename.__contains__(".cube") and filename.__contains__(".fits
    "):
887             cubeFiles.append(dir + "/" + filename)
888 
889     if len(contFiles) == 0:
890         return
891 
892     for contFile in contFiles:
893         fittedValues = fit2DGaussianToContData(contFile, True)
894         coordinates = getPointsWithinGaussian(fittedValues, 1 / 15, 20,
    False)
895         for cubeFile in cubeFiles:
896             if cubeFile.__contains__(contFile.split("_sci")[0]):
897                 clearPlots([1, 2, 3, 4, 5, 6,7,8,9,10,11,1337])
898                 try:
899                     means = meanSpectralProfile(cubeFile, coordinates, True)
900                     indicies = findRangesByRMS(means)
901                     indexMoment = computeMomentByIndex(cubeFile, indicies)
902                     maxMoment = computeMomentsByMax(cubeFile, indicies)
903                     findAngleOfOutflow(indexMoment, fittedValues,cubeFile,"
    _index")
904                     findAngleOfOutflow(maxMoment, fittedValues,cubeFile,"
    _max")
905                 except:
906                     alanasys_failed_2 += 1
907                     print("exception occured :(")
908     print("done")
```

E

main.py

```
from gc import get_freeze_count
import alminer
import alminer_extensions as almext
from keysearchmod import keysearch_mod
import pandas
import sys
import os
from astropy.io import ascii
from soupsieve import select
from FittingData import analyseDir, analyseDir2
import time

frequencies = almext.get_frequencies('./molecules.csv')

def download_routine(datadir, dryrun, keywords):
    for i in range(len(keywords)):
            # Query and filtering
            print("Querying with keyword: " + keywords[i])
            my_query = alminer.keysearch({'science_keyword':[keywords[i]]}, print_targets=False)
            selected = my_query[my_query.obs_release_date > '2016']
            selected = selected[selected.ang_res_arcsec < 0.4]
            selected = almext.removeAllProjectsWithoutMolecules(selected, frequencies)
            selected = selected.drop_duplicates(subset='obs_id').reset_index(drop=True)
            selected = selected.sort_values(by=['obs_release_date'], ascending=False)
            print(len(selected))

            # Iterates over the rows
            for i in range(len(selected)):
                tmp = selected.take([i])

                obsdir = datadir + "/" + selected.iloc[i].at['obs_id'].replace('uid://','').replace('/','-')

```

```
33                  if os.path.isdir(obsdir):
34                      print("Already analysed, skipping")
35                      continue
36
37                  if dryrun == 'True':
38                      almext.download_data2(tmp, fitsonly=True, dryrun=True,
    location=datadir, filename_must_include=[".pbcor", "_sci"], maxSize=30)
39                  while(True):
40                      inp = input("Would you like to proceed with the download
    ? [y/n]: ")
41                      #inp = "y"
42                      if inp.lower() == 'y':
43                          os.mkdir(obsdir)
44                          almext.download_data2(tmp, fitsonly=True, dryrun=
    False, location=obsdir, filename_must_include=[".pbcor", "_sci", ".cont"
    ], maxSize=30)
45                          hasCont = False
46                          for filename in os.listdir(obsdir):
47                              if filename.__contains__(".cont"):
48                                  hasCont = True
49
50                          if hasCont:
51                              almext.download_data2(tmp, fitsonly=True, dryrun
    =False, location=obsdir, filename_must_include=[".pbcor", "_sci", ".cube"
    ], maxSize=30,frequencies=frequencies)
52                              analyseDir2(obsdir)
53                              time.sleep(20) #idk
54                              deleteAllFits(obsdir)
55                          break
56                      elif inp.lower() == 'n':
57                          print("Ok, skipping.")
58                          break
59                      else:
60                          print('Incorrect input, try again.')
61
62  def deleteAllFits(dir):
63      for filename in os.listdir(dir):
64          if filename.__contains__(".fits"):
65              os.remove(dir + "/" + filename)
66
67
68  # main program.
69  # sys.argv[1] == 'all' for all keywords, or a single keyword
70  # sys.argv[2] == True for dryrun before download, otherwise it can be
    anything
71  def main():
72      # Directory for observation data
73      datadir = './data'
74      # Terminal inputs, given default values in case none are given
```

```
75      arg1 = 'all'
76      arg2 = 'True'
77
78      # All our chosen keywords
79      keywords = ['Disks around low-mass stars', 'Disks around high-mass stars
        ', 'High-mass star formation',
80                  'Intermediate-mass star formation', 'Low-mass star formation
        ', 'Outflows, jets and ionized winds']
81
82      #Checks amount of terminal arguments
83      if len(sys.argv) >= 2:
84          arg1 = sys.argv[1]
85      if len(sys.argv) >= 3:
86          arg2 = sys.argv[2]
87      # Checks that the terminal input is correct
88      if arg1 not in keywords and arg1.lower() != 'all':
89          print("Incorrect input, shutting down.")
90          quit()
91
92      # If sys.argv[1] == 'all' then keywords consists of all keywords,
        otherwise takes the terminal input (a single keyword)
93      keywords = keywords if arg1.lower() == 'all' else [arg1]
94      # Makes a folder for data downloads if there is none
95      if not os.path.isdir(datadir):
96          os.mkdir(datadir)
97
98      download_routine(datadir, arg2, keywords)
99
100     print("Program finished, shutting down.")
101
102 if __name__ == "__main__":
103     main()
```

INSTITUTIONEN FÖR RYMD-, GEO- OCH MILJÖVETENSKAP
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige
www.chalmers.se

CHALMERS