University of Groningen

Bachelor thesis

Astronomy

Measuring the radio spectra of compact radio sources

Author:
Jesper N.K.Y. Tjoa

Supervisor:
Dr. John P. McKean

July 6, 2016

Abstract

A study is performed into the angular extensions and differential number densities of gigahertz-peaked spectrum (GPS) compact radio sources (CRS) at low flux densities. An introduction is given into the topics of active galactic nuclei (AGN), CRSs and GPS, followed by a discussion of physical mechanisms abound in these sources. The selected sample of 45 CRSs is discussed and spectrally fitted using least squares and Markov-chain Monte Carlo algorithms, finding about half of the sources to indeed be GPS. Use is made of the K-z relation for radio sources, relating K-band magnitude to redshift, which is used to obtain redshifts for 14 sources without spectroscopic data, leading to a redshift completeness level in the sample of 66%. Number counts of redshift, peak flux density and peak frequency are made, showing no signs of bimodal behaviour. Obtained results are synthesised with data from Snellen et al. concerning angular size, which is shown to indeed decrease as expected at low flux densities and high frequencies. An unexpected offset is discovered in the small angular size-regime, which might hint at unusual behaviour of extremely young CRSs. Differential number counts of the number density per peak flux are performed, leading to the conclusion that a new population of unexplored, faint GPS sources may exist at low radio frequencies. Concluding remarks are provided as to how to proceed on this exploratory path, suggesting additional observations at differing frequencies, a redo of the Snellen et al. sample and exploration of the extremely high and low frequency ends of the known CRS range.

Acknowledgements

Writing your scientific début - for as far as a bachelor thesis can be regarded as such - is not something one simply does in an afternoon. Over the past two months, I have passed through both valleys of despair, when python returned errors deep within the *emcee* source code, and mountains of elated joy, when the data lined up beautifully with literature values and all was well. There are, however, a few people I would like to thank for both dragging me out of the abyss and keeping me down whenever I started cheering too early. Or just too loudly. Mainly the latter.

I would first of all like to thank my supervisor, Dr. John McKean, for his swift responses to my numerous SOS'es, his optimism and praise, and his ability to explain the matter with clarity, dissolving the cloud of confusion whenever the spectral indices had me confounded. Furthermore, a big thank you is due to Ajinkya Patil, who almost single-handedly made me understand the nuances of Markov-chain Monte Carlo, something which I regard as the turning point in this project. I say almost, for Matthijs Dries and Anke Arentsen also helped me out of a tight spot with regards to MCMC, so I would like to thank them too for their sage advice. I would also like to thank Folkert Nobels for 1) being a walking scientific thesaurus, 2) helping me comprehend the impenetrable conflagration that is LaTeXand 3) lending me his book on radiative processes. In that regard I would also like to thank Sophie van Mierlo for lending me back the book on statistical methods that I sold to her - an excellent deal, in hindsight. Additionally, I would like to thank Nick Oberg for teaching me some great feats of python I would not have thought up myself, without which the fitting procedure would have been considerably more painful.

And lastly, I would like to thank the entire student body of Kapteyn and, in some cases, those from beyond, from propaedeutic students to masters, for having a great time both the past three years and specifically two months, for the numerous lunch breaks and the conversations in the Virgo and Hydra clusters which never tired and made some particularly gruelling phases of the data reduction considerably more bearable.

Quite a mouthful of acknowledgements, but I think that covers most of it.

Contents

1 Introduction

Before one delves into the subject of compact radio sources, it is imperative to gain a basic understanding of the physics behind astronomical radio sources in general, and the active galactic nuclei (AGN) that lie at their roots. To this end, this chapter will discuss the physical and observational backgrounds regarding cosmic radio sources, gigahertz-peaked spectra and previous research into this topic. At the end of this chapter a short summary of this thesis will be given.

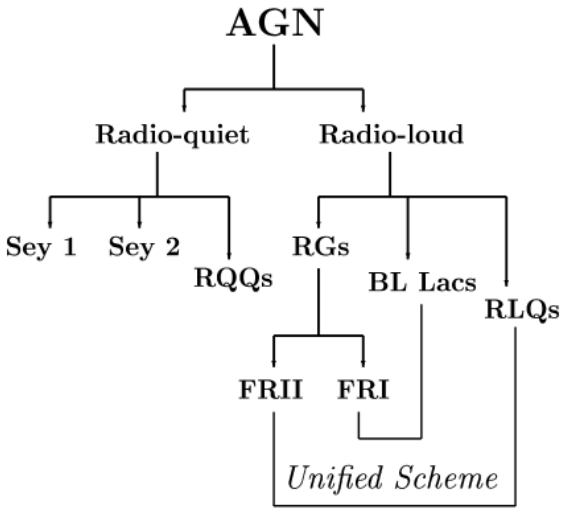


Figure 1: The classification scheme for AGN as presented by P. Kharb in her 2004 thesis. Sey stands for Seyfert galaxy, RQQs for radio-quiet quasars, RGs for radio galaxies, RLQs for radio loud quasars, BL Lacs for BL Lacertae objects and FRI and II for Fanaroff-Riley type I and II radio galaxies. Save for the last two types, none of the above objects will be discussed in much detail. The Unified Scheme refers to the hypothesis put forth by Barthel in 1989 that unifies radio galaxies with quasars [Kharb, 2004].

1.1 Radio sources & active galactic nuclei

The radio band of the electromagnetic spectrum is generally taken to range from ∼100 GHz down to the MHz regime or lower [Wilson et al., 2012]. It is an important scientific window into the cosmos, as the atmosphere is transparent to radio waves, thus enabling us to make radio observations from the Earth's surface, in contrast to for instance infrared or X-rays. That radio waves are a relevant source of astronomical knowledge is in part because such a wide array of astrophysical processes emit them, or emit radiation that by its arrival to Earth has redshifted into the radio regime.

That radio waves had an extraterrestrial origin other than the Sun was first shown by Karl Jansky in 1931 [Jansky, 1933], who detected Sagittarius A*, the galactic centre, and since then radio astronomy has taken a prominent place among the ranks of the astrophysical sciences. It has become one of the premier methods by which astronomers study young, distant and often luminous galaxies and their active galactic nuclei (termed AGN). These are nuclei of exceptional brightness (luminosities can exceed 10^{46} erg/s) which can outshine the stellar light of their host galaxies. These sources are highly variable, on the order of days has been observed, and generally extend no further than several astronomical units (AU). They exhibit continuum emission over a large range of frequencies (low radio regime to X-ray), and their emission lines are strongly Doppler broadened, indicating rotational velocities of the gas in these AGN of up to 10^4 km/s, which orbit a central and highly active supermassive black hole (SMBH). This black hole powers the AGN by accretion of matter onto it, which by conservation of angular momentum is then released into perpendicular jets [Kharb, 2004].

There are multiple types of galaxies that house an AGN. Kharb presents a clear-cut scheme in her 2004 thesis which shows the dichotomy between these species (see Figure 1). As our thesis treats radio-loud AGN, and in particular radio galaxies, radio-quiet AGN will not be discussed in-depth here. The division between these two families of active galaxies is due to the difference in radio band output, with

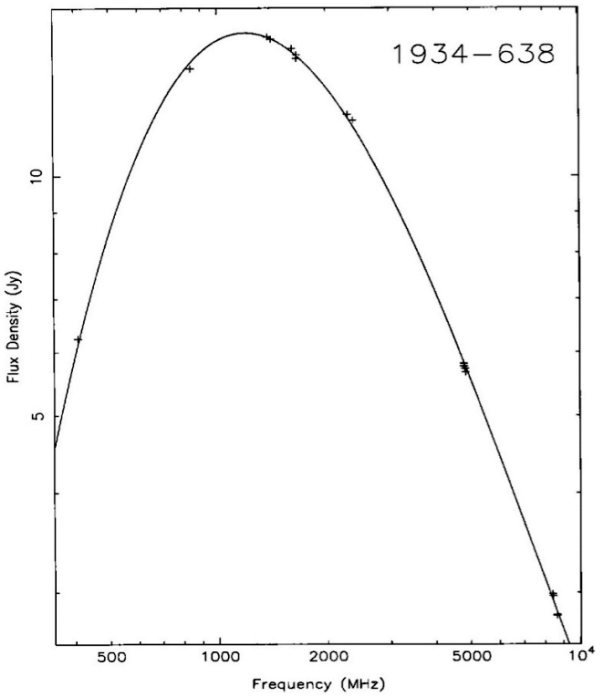


Figure 2: A quintessential GPS-shaped spectrum, belonging to PKS 1934–638, observed between 300 MHz and 8 GHz. This object is positioned at redshift 0.18, peaking just above 1 GHz. Taken from [Sadler, 2016].

radio-loud galaxies, as the name implies, showing strong emission in radio. AGN are typified by the presence of a bright, star-like core with prominent emission lines, which are absent in normal galaxies. Radio-loud galaxies, however, have additional radio emission that can be an order of magnitude larger than the optical, and possess jets of highly energetic particles and ionized material which can extend up to hundreds of kiloparsecs perpendicular to the accretion disk around the central SMBH. This radio emission is commonly interpreted as synchrotron radiation originating from a non-thermal distribution of relativistic ($v \sim c$) electrons [Kharb, 2004].

1.2 Evolution of compact radio sources

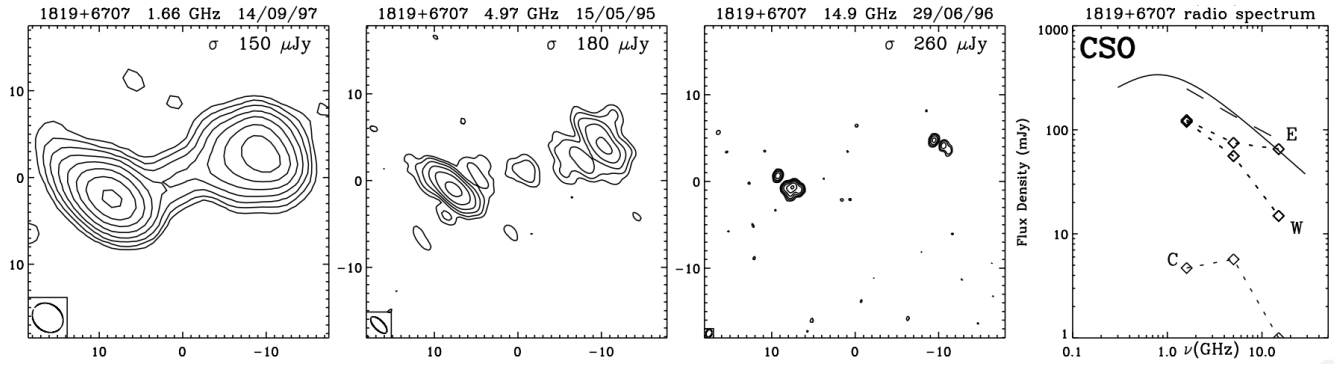
A specific subset of astronomical radio sources is made up by the so-called compact radio sources (CRS). These sources are often characterised by a small angular size (no larger than 1-2 arcsec), convex spectra that powerfully peak in the radio band, with a steep thick spectral index below the turnover peak and a thin spectral index of $\alpha \sim 0.7$ above it [Orienti, 2016]. We will henceforth refer to sources as 'pure' if their spectra fit the above description fully, and as such have a form which is most similar to the theoretical spectrum of synchrotron emission by a homogeneous object showing signs of self-absorption at the low frequency-end (also known as a 'canonical' or 'classical' spectrum) [Mingaliev et al., 2013].

Compact radio sources can be divided into multiple categories, with the main divisions being spectral and morphological. Over the spectrum range, we find, from high to low frequency [Orienti, 2016]:

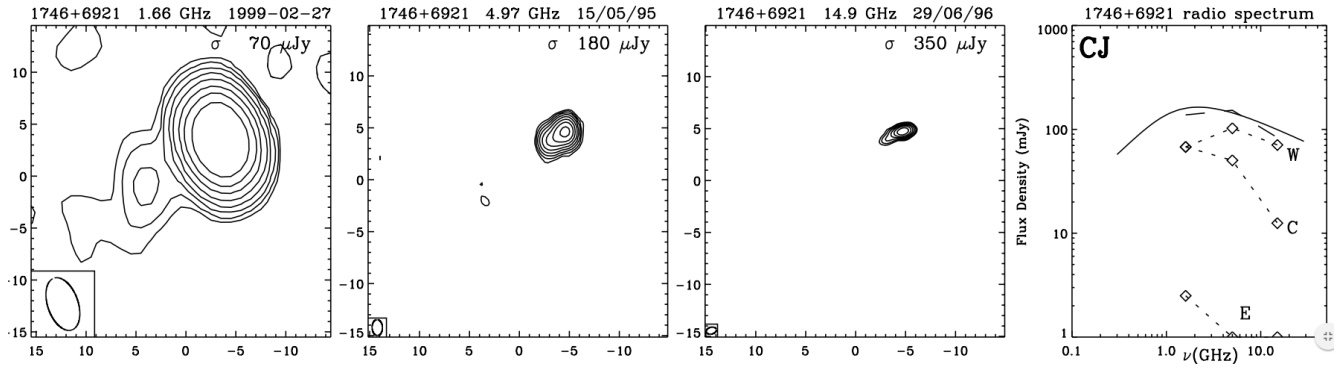
1. *High-frequency peakers (HFP):* these are radio sources with their spectral turnover at the high end of the GHz-regime. They are considered a sub-population of the gigahertz-peaked spectra.

2. *Gigahertz-peaked spectra (GPS):* these sources have a spectral turnover, as the name implies, in the GHz-regime, generally of the order of unity although it can vary from 0.5 to several GHz, as our sample will show. This thesis will look primarily at objects exhibiting these kind of spectra (see Figure 2 for a typical example).

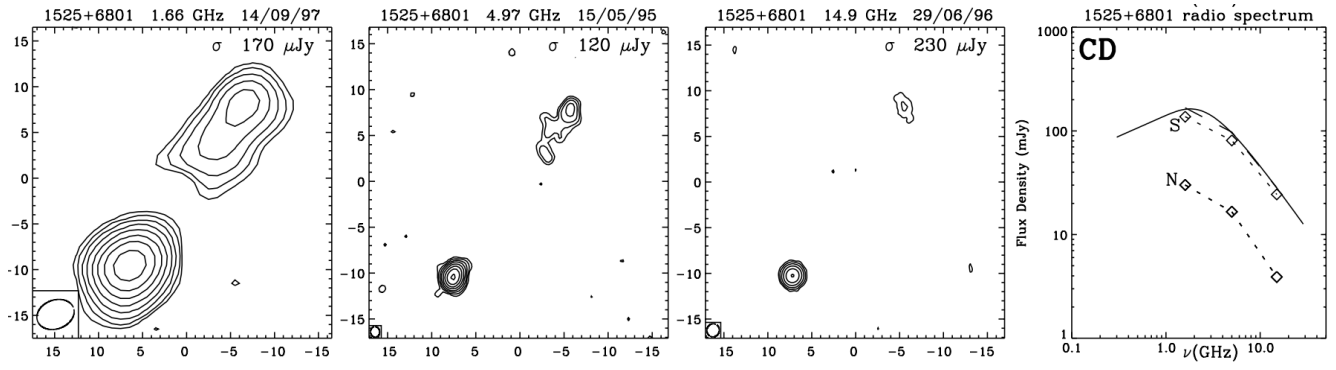
3. *Compact steep spectra (CSS):* this low-end type of radio spectra is typified by a spectral turnover in the high MHz-regime, typically of the order 100 MHz. As with HFP's and GPS, CSS show the same shapes of spectra, but at lower frequencies.



(a) A typical compact symmetric object (CSO).

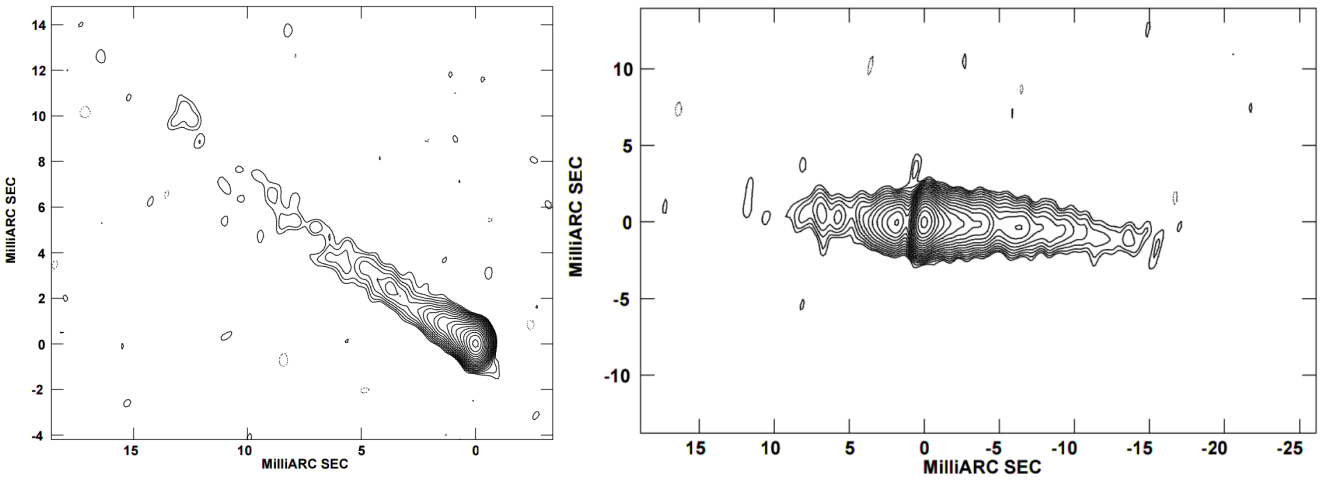


(b) A typical core-jet source (CJ).

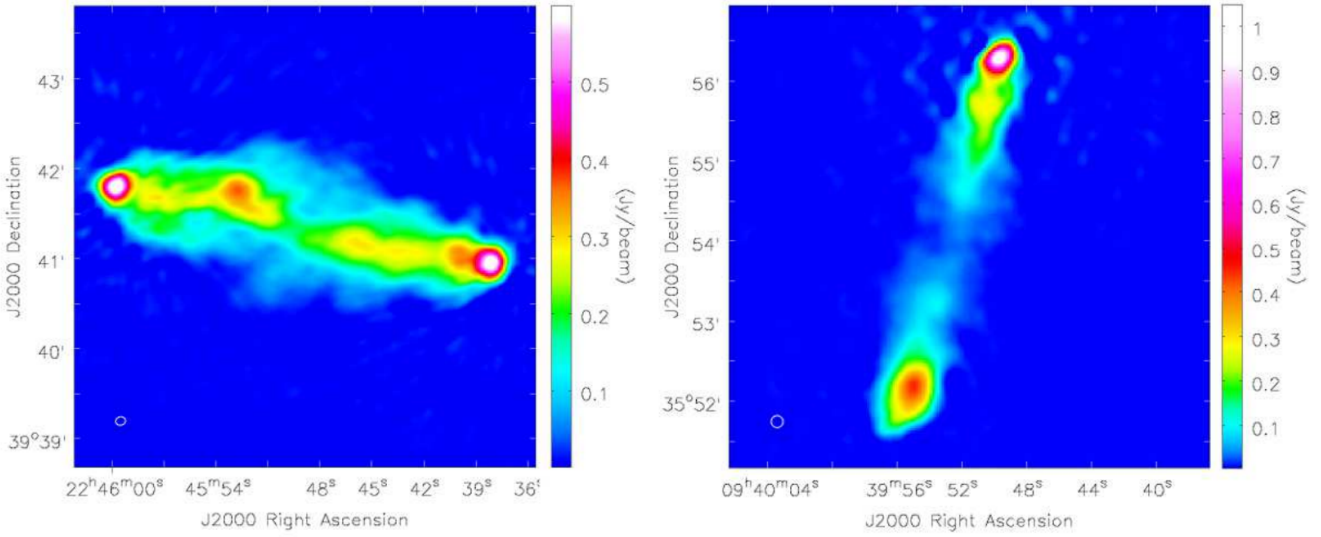


(c) A typical compact double (CD).

Figure 3: Examples of morphological classes of CRSs. Taken from [Snellen et al., 2000a].



(a) Two examples of FRI radio galaxies. Left: 3C 66B, imaged by combining 1.4 GHz VLA and 5 & 8 GHz VLBI observations, right: 3C 270, imaged by combining 5 GHz VLA and 8.4 & 8 GHz VLBI observations. Notice the lack of hotspots and the fuzzy appearance of the jets. Taken from [Kharb, 2004].



(b) Two examples of FRII radio galaxies. Left: 3C452 at 138 MHz, right: 3C223 at 147 MHz. Notice the prominent hotspots and well defined appearance of the jets. Taken from [Harwood et al., 2016].

Figure 4: Examples of the Fanaroff-Riley dichotomy in radio galaxies.

This division proves useful when one is interested in one particular stage of the evolutionary process of CRS's, as it has been implied that all three spectral classes are evolutionary stages a CRS passes through in the above order as it ages until it has grown enough to no longer be considered compact [Orienti, 2016]. In addition to this separation of types, there is also a morphological classification scheme (based on [Snellen et al., 2000b] and [Orienti, 2016]) - in particular order:

1. *Compact symmetric objects (CSO):* these are objects with a milli-arcsecond scale symmetrical structure, often characterised by two-sided lobes (although in many cases one of the lobes is far brighter than the other), extending no larger than 1 kpc. They contain a flat spectrum component and have steep spectral slopes on either side (see Figure 3a).

2. *Medium-sized symmetric objects (MSO):* if the source extends up to 10-15 kpc, they are termed medium-sized. Morphologically they look similar to CSOs.

3. *Core-jet sources (CJ):* these sources are typified by a compact flat spectral component with a steeper component on only one side (see Figure 3b).

4. *Compact double (CD):* sources showing two approximately equal components with similar spectra, but no indications of a central flat spectrum component (see Figure 3c).

5. *Complex sources (CX):* sources possessing spectra that do not fall in any of the above categories.

Because the spectral and morphological divisions are so different in their criteria, selection of sources judging on the characteristics of one such category results in widely differing samples. However, Snellen points out that there is significant overlap between particularly GPS and CSOs: most of the GPS that are identified with galaxies have a mas-scale symmetric structure, and a large majority of CSOs possess a GHz-peaked spectrum [Snellen et al., 2000b].

A third division is the Fanaroff-Riley classification (FR), which divides radio galaxies into FRI and FRII types. The realization that radio galaxies came in two distinct species gave rise to this classification, which separates sources on the basis of the relation between core and jets: FRI sources show extended, diffuse tails which do not have clear termination points (see Figure 4a), while the jets of FRII sources are narrow, well collimated and terminate in hot spots (see Figure 4b) [Kharb, 2004]. FRII sources show bright lobes; FRI, conversely, are centre brightened [Wilson et al., 2012]. An additional category, FR0, was introduced to indicate a candidate radio source [Sadler, 2016].

Multiple hypotheses were put forth by O'Dea, Baum and Stanghellini in 1991 in one of the first papers on the topic of GPS ("What are the gigahertz peaked-spectrum radio sources?",[O'Dea et al., 1991]) about the nature of the sources that produced these spectra. They postulated that they could either signify that they were ordinary radio sources which were being frustrated by dense ISM, which limited the growth of the source's jets, or that they were young radio sources, and that GPS were the progenitors of type FRI/II radio galaxies, into which they would over time evolve [Orienti, 2016], [Snellen et al., 2000a]. The general consensus is in favour of the latter hypothesis, deduced from dust spectral ageing and observed expansion speeds, although the frustration theory cannot be ruled out.

It is suspected that the often sharp peak of all these compact sources is due to the high density of synchrotron emitting electrons in the source, which participate in synchrotron self-absorption. Another striking feature is their - cosmically speaking - extremely young age, sometimes less than 10 kyr, further adding to the suggestion that these objects are in fact young radio sources akin to Active Galactic Nuclei (AGN) which in turn evolve into larger sources [Snellen et al., 1998b]. GPS sources are estimated to comprise approximately 10% of the bright radio source population, with the CSS sources accounting for another 30% [Sadler, 2016].

While CSS and GPS sources were long considered the least variable radio sources in the night sky, but this hypothesis was strongly adjusted when later research found that especially HFP and GPS type CRS vary strongly over their lifetimes, only displaying the traditional convex steep spectrum during flaring events. Only $\sim$30% of the HFP/GPS galaxies maintain their convex spectra, while CSS radio galaxies are rarely, if ever variable. HFP, GPS and CSS quasars, however, do display large variability in both overall flux density as in their spectrum.

1.3 Previous research

Earlier research into the topic of GPS was carried out by primarily Cristopher O'Dea [O'Dea et al., 1991], [O'Dea, 1998] and Ignas Snellen, who performed research into the radio properties [Snellen et al., 1998b], infrared and optical spectrum [Snellen et al., 1998a], classification [Snellen et al., 2000b] and evolution [Snellen et al., 2000a] of these sources. Compact radio sources however were already known to the scientific community for more than thirty years [Allen et al., 1962], [Conway et al., 1963], [Blake, 1970], Blake being the first to suggest that CRS's were young, and a portion of them had been catalogued by Peacock and Wall in 1981 and 1982 [Peacock and Wall, 1981] [Peacock and Wall, 1982]. The study of these objects began in earnest when Rudnick and Jones subjected a sample of CRS's to a polarization analysis in 1982 and found that the degree of polarization showed little correlation with the observed wavelength [Rudnick and Jones, 1982]. The first sample of GPS (in that they had been selected with a peak flux around 1 GHz) was documented in 1983 as candidates for high redshift objects and radio sources with a symmetry at the mas scale [Gopal-Krishna et al., 1983] and a second sample was introduced two years later [Spoelstra et al., 1985].

The first real attempts at explaining the spectra of these sources were made by O'Dea, Baum and Stanghellini in 1991, who were the first to assume synchrotron self-absorption to be the cause of the GPS's unusual shape, and used this inference to draw conclusions about them, further adding that in at least some sources free-free absorption too plays a role. They also found that these objects lack prominent jets, instead finding small, parsec scale steep-spectrum emission (dubbed "micro-lobes"), and that the sources are tightly confined. Another feature was the nature of the GPS, which were found to comprise both galaxies and quasars, with the galaxies having a more symmetric mas scale structure and also often participating in mergers. Further conclusions were made concerning the formation (the young source versus frustration scenario which is mentioned in the previous section) and the redshift, which was found to be mainly high (half with a z larger than 3) [O'Dea et al., 1991].

Snellen et al., in their 1998 paper ("A New Sample of Faint Gigahertz Peaked Spectrum Radio Sources"), were the first to describe the observed spectra using a synchrotron self-absorbed function of the frequency. This succeeded in describing typical GPS in a sample ranging in peak frequency from 500 MHz to >15 GHz, but failed to fit power law spectral distributions or combined spectra (different types of synchrotron spectra will be discussed in the next chapter). They also concluded that GPS sources have lifetimes of approximately 250 times shorter than a typical large scale radio source, and that source luminosity must decrease by a factor 10 during its transition from GPS- to radio source. Redshift distribution in the sample hinders a direct interpretation of the source counts, and it is concluded that more modelling of radio source evolution is required [Snellen et al., 1998b].

O'Dea makes several additional arguments for the evolutionary hypothesis in his 1998 paper, including an inverse relation between linear size and radiated radio power ($P \propto l^{-0.5}$), and the fact that the relation between number density and linear size is consistent with what one would expect for young, evolving radio sources. He structures the arguments in favour of the young hypothesis into four parts: that their morphologies are similar to large-scale sources, that there seems not to be enough gas in their environments to justify the frustration hypothesis, that in the absence of cold gas these sources can indeed expand at velocities some ∼10% of the speed of light, and that neither GPS nor CSS sources show a diffusely emitting halo, which is what one would expect were they confined for their lifetimes. He goes on to conclude that, combining his own data with other literature results, confines can be placed on the age of these sources, which he estimates to be intermittent on timescales of $10^4 - 10^5$ yr.

1.4 Motivation for & summary of this thesis

A critical question that should be asked before scientific research is carried out is, quite simply, why? Why is the to be performed research relevant, and why is it necessary? In the case of this thesis, the answer is similarly straightforward. The earliest stages of AGN are poorly understood, and the mechanism that leads to their triggering forms an enticing mystery. Although this thesis might not resolve this issue single-handedly, it remains important, critical even, that thorough examinations of CRSs are carried out, given that these sources possibly represent the youngest spawns of the cosmic AGN population. Analysis of these compact sources might point out major flaws in our theoretical framework regarding the formation of AGN, or instead provide convincing support for the prevailing theories. In the end, it is the author's humble hope that this thesis may contribute, even if ever so slightly, to mankind's

indomitable pursuit of knowledge.

This thesis will first discuss the dominant physical mechanisms in the sources we are studying: synchrotron radiation, free-free absorption and synchrotron self-absorption. The different spectra detected among our sources will then be explained and analysed. A short discussion of the selected sample will then be given, specifying locations and criteria, after which a brief introduction to the numerical methods used - least squares and Markov-chain Monte Carlo - to obtain the spectral fits will be given. This thesis will proceed by discussing each fit in short and providing the tables with the fitting parameters. The following chapter will first give an introduction to the K-z relation to obtain redshift from K-band magnitude, which will then be applied to find the redshifts of 14 members of the sample which do not have spectroscopic redshifts. Combining spectroscopic and K-z redshifts, number counts will be made per redshift. Results will then be synthesised to obtain values for luminosity distance, angular size differential number count per peak flux band. This thesis will conclude with a discussion of the obtained results, after which ideas will be opted as to how to continue in this scientific direction and some concluding remarks will be given. Throughout this thesis, a standard cosmological model is assumed, with $\Omega_M = 0.286$, $\Omega_V = 0.7139$ and $H_0 = 67.80$ km s^{-1} Mpc^{-1}, the most precise value from PLANCK data.

2 Radio source emission mechanisms

Radio-band radiation is emitted by a wide array of astrophysical processes, both thermal (blackbody spectrum) and non-thermal. This chapter will focus on the specific non-thermal mechanism of synchrotron emission and synchrotron self-absorption. In all formulas treated for the rest of this thesis, the following symbols denote the following quantities unless stated otherwise:

c : velocity of light
e : charge of the electron
m_e : mass of the electron
E : energy
v : velocity of the electron
a : acceleration of the electron
B : magnetic field
S : flux density
ν : frequency
γ : Lorentz factor, given by $\gamma = \dfrac{E}{m_e c^2} = \left(1 - \dfrac{v^2}{c^2}\right)^{-1/2}$

2.1 Synchrotron radiation

Synchrotron radiation is non-thermal radiation emitted by electrons in a magnetic field. Electrons gyrate around the magnetic field lines of the jets and accretion disk. At velocities of $v < c$, this is called cyclotron radiation, but if the particles are accelerated to relativistic velocities this is known as synchrotron radiation. According to the Larmor formula,

$$P = \frac{2e^2}{3c^3} \cdot \gamma^4 a'^2, \tag{1}$$

wherein P is the power of the electron and a' the acceleration of the particle in the comoving electron frame. Ergo: a particle that moves with some acceleration a' in its own comoving frame, radiates with some power P in the rest frame. Following from the Einstein-Planck equations of relativistic motion of an electron in a (homogeneous) magnetic field,

$$\frac{d}{dt}(\gamma m_e \mathbf{v}) = \frac{e}{c}(\mathbf{v} \times \mathbf{B}), \tag{2}$$

with $\mathbf{v}$ the vector velocity and $\mathbf{B}$ the vector magnetic field, we obtain the formula for the acceleration as a function of the velocity perpendicular to the field lines and the field's strength for a single electron,

$$a_\perp = \frac{eB}{\gamma m_e c} v_\perp \quad \text{with} \quad \omega_B = \frac{eB}{\gamma m_e c}. \tag{3}$$

Wherein $a_\perp$ is the acceleration perpendicular to the magnetic field, $v_\perp$ the velocity perpendicular to the direction of B and ω_B the non-relativistic gyration angular frequency. Therefore, over all emitted frequencies,

$$\langle P \rangle = \frac{2e^4 B^2 v_\perp^2}{3c^5 m_e^2} \gamma^2 = \frac{2e^4}{3c^9 m_e^4} \cdot B^2 E^2 v_\perp^2. \tag{4}$$

Wherein all symbols take their aforementioned meanings. So the radiated power of a single synchrotron electron is proportional to the squares of the magnetic field strength, the velocity and the energy of the particle. Hence the total power emitted by the source is dependent on its magnetic field and the velocity and energy of the electrons, which in turn are related to their frequency.

The frequency dependence of the radiated electron power is given by,

$$P(\nu) = \sqrt{3}\frac{e^3 B \sin(\alpha)}{m_e c^2} F\left(\frac{\nu}{\nu_c}\right). \tag{5}$$

Wherein α is the pitch angle (the angle between the magnetic field and the velocity), F the dimensionless synchrotron spectral function, ν the frequency of the radiation and ν_c the critical frequency. F and ν_c are respectively given by,

$$F\left(\frac{\nu}{\nu_c}\right) = \frac{\nu}{\nu_c}\int_{\nu/\nu_c}^{\infty} K_{5/3}(\eta)d\eta, \tag{6}$$

$$\nu_c = \frac{3\gamma^2 eB\sin(\alpha)}{4\pi m_e c^2} = \frac{3}{2}\gamma^2 \nu_G \sin\alpha \quad \text{with} \quad \nu_G = \frac{eB}{2\pi m_e c^2} \tag{7}$$

Here η is some arbitrary integration variable, and ν_G is the relativistic gyration frequency. From empirical evidence - cosmic ray measurements - we know that the number density of electrons $N(E)$ behaves like a power law,

$$N(E)dE = KE^{-\delta}dE, \tag{8}$$

in which K is some constant with units of energy to the power $-\delta$ per volume, δ some power law index and dE the energy range under investigation, while we know very little about the value of the pitch angle α's distribution. However, we also know that,

$$\epsilon(\nu)d\nu = \frac{dE}{dt}N(E)dE = \langle P\rangle N(E)dE \quad \text{hence} \quad \epsilon(\nu) = \langle P\rangle N(E)\frac{dE}{d\nu}, \tag{9}$$

where ϵ is the volume emissivity and the other symbols take their aforementioned meanings. Substituting equations 8 and 4 into the above relation, and using the relation given by,

$$E = \gamma m_e c^2 \approx \left(\frac{\nu}{\nu_G}\right)^{1/2} \cdot m_e c^2 \quad \text{hence} \quad \frac{dE}{d\nu} \approx \frac{m_e c^2 \nu^{-1/2}}{2\nu_G^{1/2}}, \tag{10}$$

where ν_G is again the relativistic gyration frequency, we obtain as a final result for $\epsilon(\nu)$ for a power law energy distributed electron population,

$$\epsilon(\nu) = \frac{2e^4}{3c^9 m_e^4} \cdot B^2 E^2 v_\perp^2 K E^{-\delta}\frac{m_e c^2 \nu^{-1/2}}{2\nu_G^{1/2}}. \tag{11}$$

This is a messy relation which can be more transparently rewritten to the proportionality of frequency, particle energy and magnetic field strength, given by,

$$\epsilon(\nu) \propto B^2 E^{2-\delta}(\nu \cdot \nu_G)^{-1/2}, \tag{12}$$

which, because of the relations between energy and frequency (equation 10) and relativistic gyration frequency and magnetic field strength (equation 7), then becomes a proportionality of frequency and magnetic field strength alone given by,

$$\epsilon(\nu) \propto B^2 \left(\frac{\nu}{B}\right)^{1-\delta/2} (\nu \cdot B)^{-1/2} \propto B^{(1+\delta)/2} \cdot \nu^{(1-\delta)/2}. \tag{13}$$

If we then define a spectral index $\alpha = (1-\delta)/2$ and realize that flux density is proportional to intensity, which is proportional to volume emissivity, i.e. $S(\nu) \propto I(\nu) \propto \epsilon(\nu)$, we obtain the proportionality which forms the cornerstone of this thesis, namely the relation between synchrotron frequency and flux density of a power law energy distribution for a synchrotron emitting electron population,

$$S \propto \nu^{\alpha}, \tag{14}$$

wherein α is some spectral index, defined separately in different regions of the synchrotron spectrum [Wilson et al., 2012]. For a purely synchrotron emitting population, the spectrum follows such a power law distribution, leading to the remarkable conclusion that the spectrum of a power law energy distribution in an electron population is itself a power law.

Synchrotron radiation is also a beamed phenomenon. This is because the electrons, which gyrate around and experience a net movement in the direction of the magnetic field lines, emit in a cone towards the propagation direction [Rybicki and Lightman, 1986]. As such, synchrotron emission from an AGN, which possesses extremely strong magnetic fields, is stronger in the direction parallel to the magnetic field's orientation.

2.2 Synchrotron self-absorption

In regions where the synchrotron radiation intensity and density are high, the emitted photons interact with the emitting electrons via synchrotron self-absorption: they are reabsorbed by the electrons, which then reach a more excited energy state. This happens in CRS's and as such in GPS sources. This phenomenon is called synchrotron self-absorption (SSA) as the synchrotron emitting electrons absorb their own emission, and no external absorber is in play [Torniainen, 2008]. This behaviour is inversely proportional to the frequency, meaning that below some frequency ν_{peak}, SSA dominates over synchrotron emission and the spectrum decreases sharply with a theoretical spectral index (called the thick spectral index) of 5/2. This can be proven using the concept of effective temperature of the electron plasma, the temperature of a theoretical black body source that would emit the same amount of radiation as the plasma,

$$T_e = \frac{E}{3k}. \tag{15}$$

Wherein T_e is the effective temperature and k is the Boltzmann constant. Using the relation between energy and frequency and magnetic field strength, this can be rewritten to,

$$T_e = \frac{\gamma m_e c^2}{3k} \propto \sqrt{\frac{\nu}{B}}. \tag{16}$$

Wherein all symbols take their usual meaning. The specific intensity, which is the intensity of a theoretical black body source that would emit the same amount of radiation as the plasma, is defined when the effective temperature equals the brightness temperature T_b (the temperature the electron plasma would have if it were a blackbody) of the plasma as,

$$I(\nu) = \frac{2kT_b}{\lambda^2}, \tag{17}$$

and as the wavelength λ is proportional to one over the frequency ν, we obtain,

$$I(\nu) \propto \nu^2 \sqrt{\frac{\nu}{B}} = \nu^{5/2} B^{-1/2}. \tag{18}$$

When we then once more recall that $S(\nu) \propto I(\nu)$, we obtain our final result for the flux density of a synchrotron self absorbed source,

$$S(\nu) \propto \nu^{5/2}. \tag{19}$$

[Wilson et al., 2012] In thermal processes, this thick index has the value 2, known as the Rayleigh-Jeans value, and thus showing that SSA is not a thermal process [Rybicki and Lightman, 1986]. This value is in theory constant regardless of the electron population's spectral index above the turnover peak, although in practice such a steep value has not been observed in previous research[1] [Torniainen, 2008].

As the source ages, the highest energy electrons radiate away their energy first, leading to the overall flux of the source to drop at high frequencies. This leads to a high-frequency 'ageing cutoff' where the spectrum drops sharply. When the energy of the electron plasma drops, so does its frequency threshold for when SSA starts to dominate, so the peak shifts towards the lower end of the spectrum. This is one of the main pieces of evidence that supports the hypothesis that HFP's, GPS and CSS are, in that order, stages of the evolution of a young radio galaxy, as the peak frequency decreases with each type [Orienti, 2016]. Additionally, at very low frequencies, below the SSA region, a cutoff has been theorized because the spectra of individual electrons start to interfere with the synchrotron radiation. This has not yet been physically observed [Wilson et al., 2012].

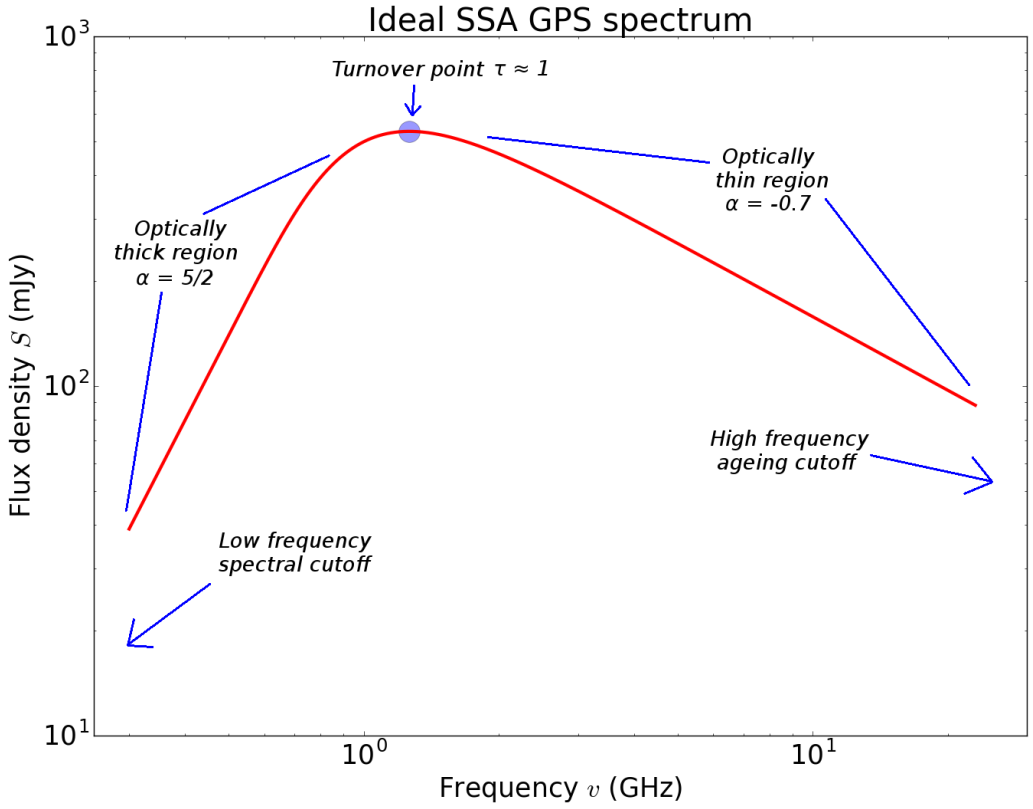
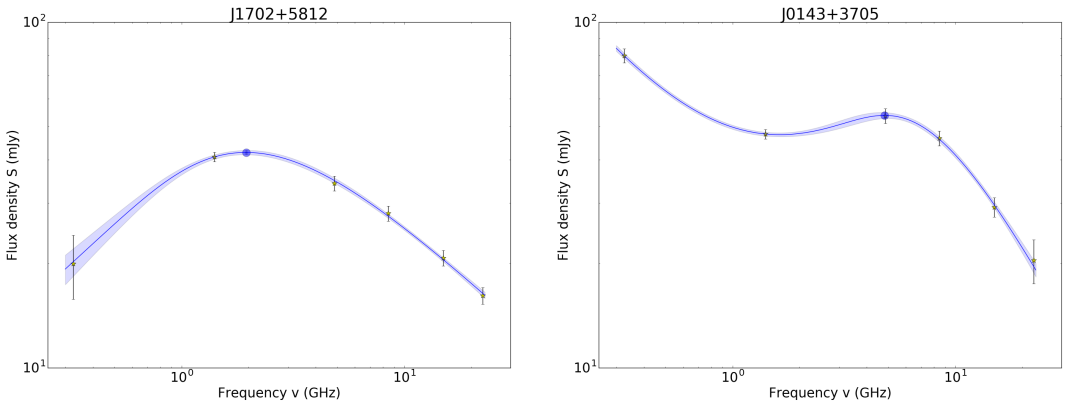


Figure 5: An annotated idealized synchrotron self absorbed GPS spectrum. Peak location at 1.3 GHz, peak flux 535 mJy, which exceeds the sources used in this thesis.

[1] In our sample, two sources (J0103+4322 and J0104+3840) possessed thick spectral indices of approximately 5/2. However, the author would like to point out that both spectra were notoriously hard to fit with a synchrotron peak, as the first of these two was almost a flat spectrum and the second one had a very poorly defined optically thick region, which was entirely described by just two data points, thus giving rise to large uncertainties.



(a) An archetypal GPS. Peak is located at 1.95 GHz, peak flux is 42 mJy. Spectral indices from left to right: 0.6, -0.6 respectively.

(b) A typical GPS+power law combined spectrum. Peak is located at 4.81 GHz, peak flux is 54 mJy. Indices: 0.5, -1.2, -0.8 respectively.

Figure 6: Examples of the two main types of peaked spectra.

2.3 Free-free absorption

Another process by which a strong turnover in the low GHz regime can be facilitated is free-free absorption (FFA). While free-free radiation (otherwise known as bremsstrahlung, after the German term for braking and radiation) is emitted when an electron in a plasma is slowed down and deflected by another charged particle, thereby losing energy in the form of a photon, during FFA the opposite occurs: the electron gains energy in its encounter with for instance an ion, absorbing a photon, causing it to jump to a higher energy state [Wilson et al., 2012]. The equation for FFA is given by,

$$S_{FFA}(\nu) = S_0 \cdot v^{\alpha} \cdot e^{\tau_f \cdot \nu^{-2.1}}. \tag{20}$$

Where S_{FFA} is the flux at at some frequency, S_0 the scale flux density, ν the frequency, α the spectral index and τ_f the optical depth at the peak. Suggestions have been made that in fact FFA by a thick gas- and dust disk around the AGN is responsible for the steep turnover in the lower frequency regions of a CRS spectrum, describing a "cold dense FFA plasma around the lobes of GPS sources" which "could be a cocoon which smothers expansion of jets and lobes" [Kameno et al., 2003], rather than SSA. If this were the case, then the degree of FFA, measured by the opacity τ_ν in equation 20, should vary with the viewing angle. Although Orienti comments that FFA cannot be completely discarded as an additional mechanism, because SSA spectra seem to fit the data accurately, more so than FFA, the scientific consensus leans towards SSA [Orienti, 2016], and as such the fits made in this thesis have been performed assuming SSA to be the dominant, if not only mechanism in play.

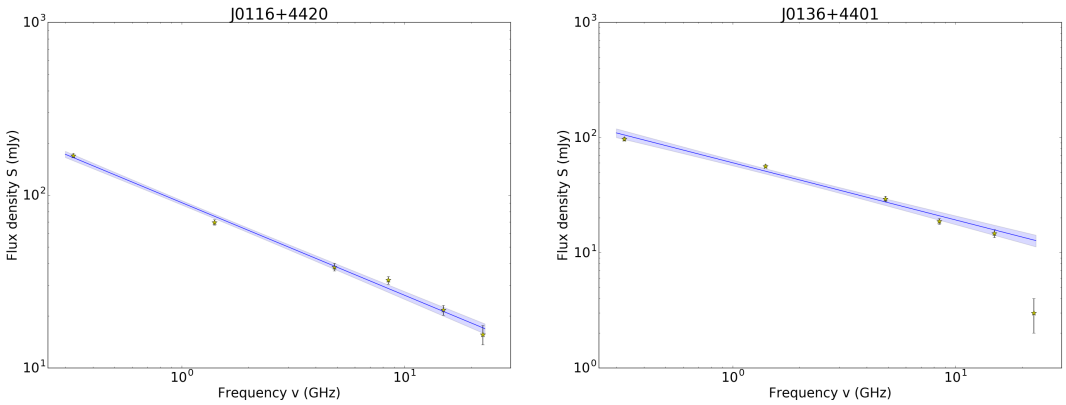
2.4 Synchrotron spectra

Although the quintessential SSA spectrum is given by the spectrum in Figure 5, multiple shapes of spectra appeared in the data, for which different fitting equations were used. These are discussed below.

2.4.1 Pure GPS spectra

Figure 6a shows a typical SSA GPS spectrum from our sample, with the shaded region denoting the one sigma regions around the fit. The fitting formula that we used in case of a typical GPS is given by,

$$S_{GPS}(\nu) = \frac{S_0}{1 - e^{-1}} \cdot \left(\frac{\nu}{\nu_0}\right)^k \cdot \left(1 - e^{-\left(\frac{\nu}{\nu_0}\right)^{l-k}}\right). \tag{21}$$



(a) A typical regular power law spectrum. Spectral index m is -0.54.

(b) A typical aged power law spectrum. Spectral index m is -0.49.

Figure 7: Examples of the two main types of power law spectra.

This formula is taken from [Snellen et al., 1998b]. In optically thin regions (above the turnover peak), the spectral index, denoted by l, is negative, generally between -0.5 and -1. This is because in these regions the radiation intensity and density are low enough that SSA by the plasma becomes negligible, and synchrotron emission can escape the source. Below the peak, the equation is dominated by the thick spectral index k, which as mentioned previously has a theoretical value of 5/2, which represents a homogeneous, synchrotron self-absorbed radio source, but in practice generally varies between 0.5 and 2. In consultation with Dr. McKean, this is explained by the spectrum being composed of multiple overlapping SSA components, some with a lower peak frequency and peak flux than the primary source, thus changing the spectral index k and making the increase more gradual in the low frequency regime (as is done in for instance [Kellermann, 1972]) [Orienti, 2016]. S_0 and ν_0 are scaling constants which influence the height and frequency of the peak, but have no further physical meaning.

Spectra with such a thick spectral index (i.e. when it is not exactly 2.5) are interpreted as multiple synchrotron emitting populations in different regions of the source, like nascent lobes, hotspots or jets. However, as only six data points were present across the observed band, a model with as many free parameters as a superposition of multiple synchrotron sources could not be fitted.

2.4.2 Power law spectra

A second subtype of data sets was formed by the power law-fitted sets (Figure 7). The spectra of these sources were formed by power laws - or in some cases combinations of power laws - of varying spectral indices, generally in the same range as the thin spectral index of a synchrotron spectrum. The fit used for these kind of spectra is given by,

$$S_{pl}(\nu) = \sum_{i=1}^{n=1,2} S_{0,i} \cdot \left(\frac{\nu}{\nu_{0,i}}\right)^{m_i}, \tag{22}$$

in which m_i is the spectral index of the i-th power law population, and $S_{0,i}$ and $\nu_{0,i}$ the scaling coefficients of the i-th power law which determine the overall height and thus the size and intensity of that population - more strongly emitting populations have a larger height. The summation is used because linear combinations of power law spectra did occur, n being the number of power laws involved, but in only three occurrences where $n = 2$: in all other cases n was taken as unity. This is interpreted physically as an aged population of electrons which has passed through a synchrotron phase before but which has now radiated away most of its energy. Therefore, the turnover has shifted to such low frequencies that it is no longer within the observed range, and hence in the frequency window we use only a power law spectrum with $n = 1$ is observed. SSA no longer plays an important role as synchrotron emission is no longer dense enough to enable it.

More than one type of spectrum necessitated a power law fit: The two main types are regular power laws (Figure 7a) and aged spectra (Figure 7b). These have a strong falloff at the high frequency end, with the data point at 22.49 GHz being offset negatively from the power law trend. Fitting an aged spectrum would be highly complicated and not required for the results to be obtained, so the decision was made to fit the remaining five points with a normal power law spectrum, effectively dropping the last point from the fitting process. These are physically interpreted as highly aged electron populations that are experiencing a high-frequency falloff as its most energetic electrons (and thus those with the highest energy) have radiated away.

2.4.3 GPS/Power law combined spectra

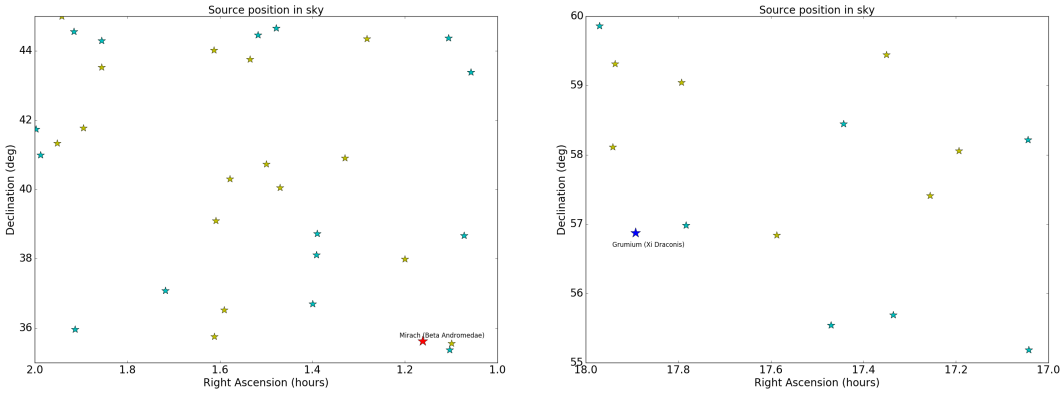
Spectra were also encountered wherein the first data point had a flux density offset from the optically thick part of the spectrum predicted by a simple SSA GPS fit (Figure 6b). These were fitted by assuming another, older population of electrons that follow a power law-distribution being superposed with the SSA spectrum, which gives rise to the combined formula,

$$S_{GPS+pl}(\nu) = S_{GPS}(\nu) + S_{pl}(\nu). \tag{23}$$

The physical interpretation of this combined population is a highly aged population of electrons in the source that have attained a power law distribution with a younger synchrotron component, implying that it has passed through a period of synchrotron emission before and that the present-day synchrotron component is not indicative of a young radio galaxy but an older, re-energized radio-loud source.

3 Fitting the radio spectra

The data sets obtained by McKean, Browne and Jackson showed clear synchrotron or power law behaviour but required fitting before any further conclusions could be drawn. This chapter discusses the methods used to obtain spectral fits to the used datasets and analyse the fit uncertainties. The entire procedure was carried out in the python programming language.



(a) The low field ($1^h \leq \alpha \leq 2^h$, $35° \leq \delta \leq 45°$). Mirach, a bright star, is provided for reference.

(b) The high field ($17^h \leq \alpha \leq 18^h$, $55° \leq \delta \leq 60°$). Grumium, a bright star, is provided for reference.

Figure 8: The two observed fields. Cyan stars are sources who were proven to possess peaked spectra. Created using the script in Appendix B.4

3.1 Sample screening and selection

The sample that has been used is drawn from data from observations done with the Very Large Array (VLA) at 4.86, 8.46, 14.94, 22.46 GHz. Measurements at 325 MHz and 1.4 GHz have been taken from the Westerbork Northern Sky Survey (WENSS) [Rengelink et al., 1997] and the NRAO VLA Sky Survey (NVSS) [Condon et al., 1998], respectively. The quasi-simultaneous VLA observations for each source were performed in 2000 and 2001 in C and D configuration while the WENSS and NVSS data were taken years before, resulting in in some cases large discrepancies between those data points and the VLA data and large uncertainty bars.

The sources were selected according to the following criteria:

- The spectral index in equation 14 between 1.4 (NVSS) and 4.85 (GB6) GHz satisfies $\alpha \geq -0.5$;
- The flux density at 4.85 GHz according to the GB6 survey [Gregory et al., 1996] satisfies $25 < S_{4.85} < 50$ mJy;
- The NVSS flux density at 1.4 GHz is measured by summing the radio emission within 70" of the GB6 position;
- The flux density at 8.46 GHz as measured by CLASS [Myers et al., 2003] satisfies $S_{8.46} \geq 16.7$ mJy.

The 45 sources of the complete sample are distributed between two fields: these regions are given by $17^h \leq \alpha \leq 18^h$, $55° \leq \delta \leq 60°$ and $1^h \leq \alpha \leq 2^h$, $35° \leq \delta \leq 45°$, with α and δ the right ascension and declination respectively (see Figure 8). These two fields were chosen as to coincide with the fields used by Marlow et al. in his 2000 paper on spectroscopic redshifts [Marlow et al., 2000]: this becomes relevant in the determination of the K-z relation for this sample. The observation results are listed in Appendix A, with the VLA measurements in the second and the others in the first table. Sky areas have also been computed using the relation,

$$A_{sky} = (\alpha_{high} - \alpha_{low}) \cdot (\sin(\delta_{high}) - \sin(\delta_{low}).) \tag{24}$$

3.2 Least squares fitting and Markov chain Monte-Carlo algorithms

Two numerical methods were used to compute the model parameters and estimate their respective uncertainties: least squares (LSq) and Markov chain Monte-Carlo (MCMC). These types of algorithm are shortly discussed in this section, but as most of the mathematical groundwork for these subroutines was already included in the modules from which the used functions were drawn, we will not go into great depths.

3.2.1 Least squares

Least squares fitting algorithms are a frequentist approach to model fitting and work by minimizing the sum of the squares of the differences between data and model over the uncertainty in the data (which can vary from point to point). Consider a model given by,

$$f(x) = a + bx. \tag{25}$$

Then the best fit to some data set y, x of size N is found by minimizing the quantity,

$$\chi^2 = \sum_{i=1}^{N} \frac{(y_i - f(x_i))^2}{\sigma_{y_i}^2}. \tag{26}$$

With σ_{y_i} the (Gaussian) uncertainty of the i-th data point. This quantity is called chi-squared, and the model is optimized for the data set when chi-squared is minimal. The advantage of this method is that numerically it is straightforward and reliable (especially if one is dealing with only six data points), but a pitfall rests in the fact that the minimum chi-squared obtained is in fact a false minimum: that is, somewhere else in the parameter space that is being sampled, there is a position with a smaller chi-squared [Hogg et al., 2010]. The covariances (and the variances, which essentially are simply a special case of the covariance where the two co-varying parameters are the same) are computed by,

$$Cov(X, Y) = \frac{\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})}{N}. \tag{27}$$

Wherein X and Y take the place of the parameters. This then constructs the covariance matrix, which in turn is used to compute the uncertainty in the actual fit. If more free parameters exist than data points, however, the system is said to be underdefined and no covariance estimates can be made (This stems from experience with the fitting algorithm, which will be discussed in the following sections).

3.2.2 Markov chain Monte-Carlo

Markov chain Monte-Carlo algorithms follow a more Bayesian approach to model fitting. It employs random sampling (the Monte-Carlo part) and in that random walking through the parameter space by a multitude of walkers (Markov chains). As taken from "Handbook of Markov Chain Monte Carlo": suppose we have a program of the form,

```
Initialize x
repeat {
    Generate pseudorandom change to x
    Output x
}
```

This is the essence of MCMC: placing a walker in a phase space, performing a pseudo-random change[2] on that walker, and returning the new value. More exactly, when MCMC is used to fit a model to data, sampling is performed from an n-dimensional phase space of parameter values, where n is the number of parameters in the model (in the case of this thesis, this will vary from 2 to 6). To each point in this

[2]Computer generated data is never truly random.

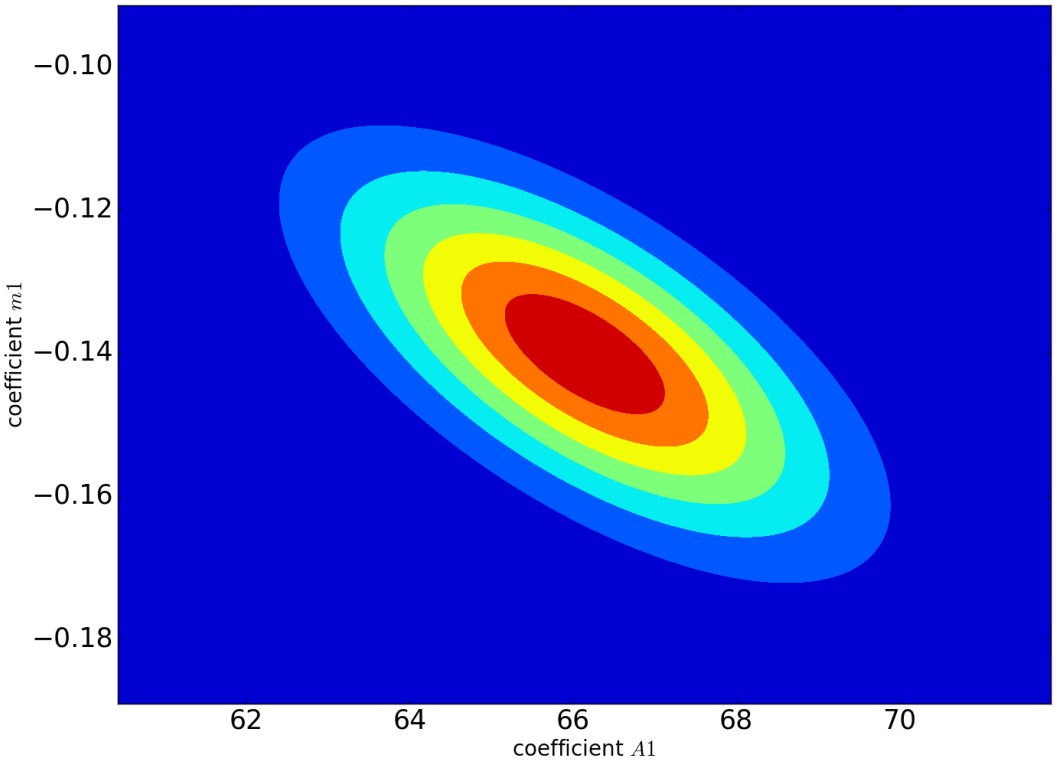


Figure 9: An example contour plot for the coefficients of a power law spectrum, in this case J0136+3905.

parameter space is assigned a probability value, and the optimal values for the model's parameters are found by maximizing that probability: this maximization is performed by having the walkers return the probability value from the phase space after every random step. Is the probability value higher than the one found in the step before, then it is accepted as the new initial position for the next random iteration. If this is not the case, the walker falls back to its original starting position and attempts a different change. This continues until the walker finds a maximum from which it cannot escape. As we use a collective of multiple hundreds of walkers from which we take only the absolute highest value, the chances of collapsing into a false maximum are reduced. When one takes for instance the model in equation 25, the phase space of the parameters might look like the contour plot in Figure 9, where the region within the central isophote delineates the most probable values of the parameters within one sigma [Brooks et al., 2011].

In our code the *emcee* and *corner* python packages are utilized to perform the actual sampling. The *emcee* package takes a likelihood (the model) and prior (an indication of prior knowledge about the model, this can be used to shape the phase space to be biased towards certain values) function, distributes m walkers in n dimensions, then makes x iterations to their phase space position, eventually returning the minimum probability value found, which then provides the optimal parameters for the model. *corner* in turn uses the accrued walker values to create contour plots from which variances and covariances can be deduced by determining the full-width half-maximum (FWHM) values of the contours parallel to the parameter axes and the diagonals.

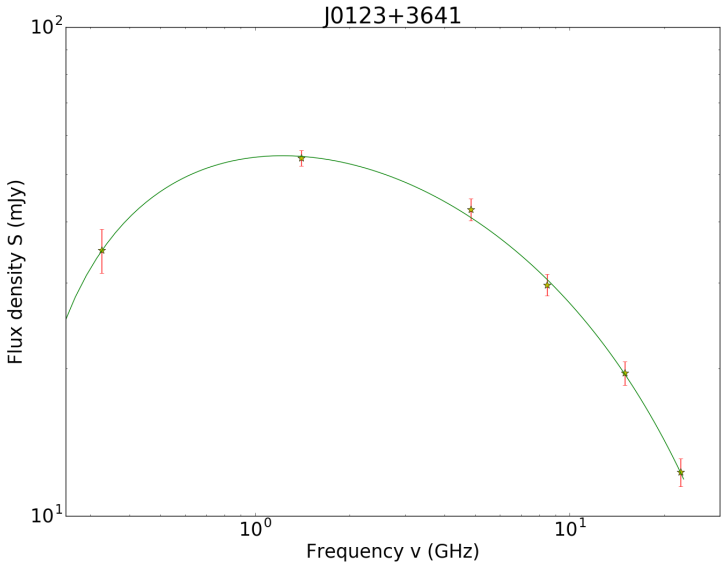


Figure 10: The polynomial fit to the spectrum of J0123+3641. This source was chosen to test the fitting algorithms as it was smooth, continuous and therefore easy to fit.

3.3 Methodology

3.3.1 Polynomial fitting

A preliminary analysis of the spectra was performed by fitting them with a logarithmic polynomial (see Figure 10), given by the equation,

$$S(\nu) = \sum_{i=0}^{n=2,3} a_i \cdot (\log_{10}(\nu))^i \,. \tag{28}$$

This function has no physical significance and does not describe the synchrotron behaviour of the source's electron population according to the model for our sources, but can be used as a preliminary analysis of the data and test the robustness of our code.

The function was described in python and subjected to an LSq algorithm from the *scipy.optimize* package to obtain the coefficients a_i for the polynomial by minimizing the sum of the squares of the uncertainties. Dependent on whether or not the data points showed signs of multiple turnovers, a quadratic ($n = 2$) polynomial was fitted normally and a cubic ($n = 3$) polynomial in case of two stationary points. As a polynomial proved substantially more intuitive and easy to fit than an SSA spectrum, this method was used to obtain a general impression of the sample (i.e. determine which spectra behaved like typical GPS, power law spectra or combinations of these) and find rough estimates for the peak frequency and peak flux. Extensive attention was not devoted to the uncertainties in either the coefficients or the peak, as this was not the final result and only served to give a crude impression of the spectral shape. This is also why an LSq algorithm was used instead of MCMC: LSq is substantially lighter and faster in terms of processing power and as the output served only as an estimate the risk of collapsing into a false minimum chi squared was accepted.

3.3.2 SSA fitting

As a polynomial fit holds no physical meaning, the code was rewritten to fit an SSA spectrum to the data instead, as would physically be expected in the case of our sources. Although FFA fits to the data were also an option, the decision was made to focus on SSA. This spectrum is given by equation 21. In case a second turnover had been registered in the polynomial fit or was abundantly clear from the data first-hand, a GPS plus power law spectrum was fitted, given by equation 23. In case no turnover

had been registered, an ageing falloff was detected at the high frequency end or a negative turnover was detected, either one or multiple power laws were fitted, given by equation 22.

Flux density, flux uncertainty and observation frequency data are read from the table by the algorithm. These are then plotted to give the user an impression of the set, after which a user prompt enables picking one of the possible fitting procedures:

- *Power law spectrum:* This fit is selected if the spectrum does not show signs of curvature in log-log space. It includes all data points, which are then fitted with equation 22 for $n = 1$.
- *Aged power law spectrum:* This fit is used if the final data point shows a strong falloff with respect to the general trend through the other five data points. It excludes the final data point. The remaining points are then fitted with equation 22 for $n = 1$.
- *Double power law spectrum:* This fit can be selected if the spectrum shows signs of curvature, but concave instead of convex (which is what one would expect for a GPS). It includes all data points, which are then fitted with equation 22 for $n = 2$.
- *SSA GPS:* This is the type of fit used for stereotypical GPS spectra: convex, with a steep thick spectral index in the low frequency band and a more gently sloping thin spectral index in the higher frequency region. It includes all data points, which are then fitted with equation 21.
- *SSA+power law GPS:* This fit is applied if the spectrum shows a turnover peak, but curves back upwards in the lower frequency regions, i.e. it has two turnovers. It includes all data points, which are then fitted with equation 23.

The functions were then fitted using the *curve_fit* function from the *scipy.optimize* package, which applies an LSq algorithm to find the parameter values and estimates the covariances between them (see Appendix B.1). In case of either a power law or the SSA GPS fitting type, this succeeded. The function for a power law had to be rewritten for ease of coding to,

$$S(\nu) = \sum_{i=1}^{2} A_i \cdot \nu^{m_i} \quad \text{with} \quad A_i = \frac{S_{0,i}}{\nu_{0,i}^{m_i}}. \tag{29}$$

As the degeneracies between $S_{0,i}$ and $\nu_{0,i}$ simply proved too great, leading to the fitting algorithm to give us results with massive uncertainties. The fitting of a function with simply an amplitude fudge factor $A_{0,i}$ proved far more effective, and as the scaling coefficients had no real physical meaning to begin with this did not lead to non-physical results. The SSA+power law GPS is however deserving of some extra attention.

As only six data points were given per source, this frustrated efforts to determine the uncertainties of the parameters in a model with six degrees of freedom. LSq methods failed to account for this, so attempts were made to resolve this issue by instead performing the fitting procedure with an MCMC. To this end, the *emcee* and *corner* modules were acquired, which perform an MCMC algorithm and create covariance contour-plots, respectively. An MCMC fitting algorithm was coded (see Appendix B.2) using the values yielded by the LSq method as starting points for the walkers, assuming normal (Gaussian) prior distributions with small FWHM around these values in case of the spectral indices. 200 Walkers were used which were distributed randomly around the values estimated by the LSq method and then iterated over 1000 steps, returning values, uncertainties and a covariance matrix. The uncertainties in the parameters themselves and the covariances between them were determined by a function from the *numpy* package, but which utilizes the same equation as the LSq method (equation 27). To ascertain that the returned values were correct this was compared to the output of an algorithm that computes the uncertainties by determining the FWHM of the histogram of the relevant parameter as outputted by the MCMC. The *numpy.cov* function proved correct.

The peak and uncertainties in that peak were then found by using a bisection algorithm of a finite step size on the derivative of the fitting function - if the spectrum was fitted by a power law, then no peak was computed. The uncertainty in the peak flux was found by classical propagation of uncertainty (taking partial derivatives, multiplying by relevant covariance, square and sum for all parameters) while for the uncertainty in the peak frequency the step size of the bisection array was used. The spectrum

(with a shaded region around the best fit line indicating one sigma), contour plots and, in the MCMC's case, parameter histograms were then plotted and saved, as were the source name and index, parameter values for the best fits, their uncertainties and the peak coordinates, which were saved to a file.

3.3.3 MCMC uncertainty analysis (issues)

Although the MCMC was successful in finding the optimal values for the parameters to obtain the best fit line, the uncertainty analysis proved problematic. Uncertainties in the parameters for the GPS+power law fit were on average an order of magnitude in excess of what the LSq method returned, and when propagated through these uncertainties showed one sigma regions around the fit which exceeded the uncertainties in the data points themselves, something which was deemed non-physical. Suspicions were raised that this was due to high degeneracy between the spectral indices m (first power law, low frequency regime), k (second power law, mid regime) and l (third power law or exponential falloff, high end). m Often proved poorly defined by only the two data points at 325 MHz and 1.4 GHz, thus prompting considerable uncertainties. Additionally, around 1.4 GHz the spectrum should transition from power law dominance, i.e. the m-regime, to dominance of the SSA component, and specifically the thick spectral region (the k-regime). This was assumed the culprit.

Attempts were thus made to combat this issue by strongly constraining m, first by taking a Gaussian prior with an extremely small FWHM, then by fixing it completely in the hopes that by allowing the program only five free parameters, this would push back uncertainties below the physical threshold of the data points' uncertainties. The formula used for fixing m is given by,

$$m = \frac{d[\log_{10}(S)]}{d[\log_{10}(\nu)]} = \frac{\log_{10}(0.5 \cdot S_{1.4GHz}) - \log_{10}(S_{0.325GHz})}{\log_{10}(1.4GHz) - \log_{10}(0.325GHz)}. \tag{30}$$

The factor 0.5 by which the flux at 1.4 GHz is multiplied stems from the fact that at in that frequency band the main contribution to the flux transitions from the power law component to the SSA component: therefore, at 1.4 GHz, half of the flux is contributed by the power law and half by the SSA spectrum. This fix proved ineffective. For reasons not discerned, uncertainties in the remaining five parameters remained an order of magnitude larger than outputted by the LSq method. When the MCMC was attempted on power law and SSA GPS spectra, with 2 and 4 free parameters respectively, success *was* achieved in obtaining uncertainties below the physically allowed values. Analysis of the problem showed that the mistake was not in the uncertainty propagation but stemmed directly from the MCMC output: the FWHM of the histograms confirmed the excessive uncertainties in the coefficients. This means that even after restraining one of the six free parameters, degeneracies between the remaining five remain too high to provide a reliable result. As parameter uncertainties of the LSq method were of the same scale as for the MCMC in case of GPS and power law spectra, the LSq algorithm was used for these for time's sake and ease of computation. As in the GPS+power law case the MCMC returned non-physical uncertainties, another LSq algorithm was written to output the results for a GPS+power law fit with fixed m.

3.4 Results

The sample consisted of 45 sources. 44 Of these sources have flux density data at all six frequencies (0.325, 1.4, 4.86, 8.46, 14.94 and 22.46 GHz) and could thus be used. 23 spectra were peaked: 13 spectra were typical GPS and 10 behaved like GPS+power law combined spectra. The remaining 21 sources showed power law spectra, 5 of which showed signs of spectral ageing, 3 of which were double power laws[3] ($n = 2$) while the remaining 13 behaved like regular power laws. Their coefficients and, in the peaked cases, peak coordinates are listed in the tables below the source-by-source discussion (Tables 1 to 6).

A full list of all the spectrum outputs is provided in Figures 11 and 12. A short discussion of each separate source will now be given in the same order as the given Figures. Some example corner plots of the covariance of the fit parameters will be provided to show what constitutes a good or a bad fit.

[3]The spectra fitted with double power laws were fitted as such as they clearly did not show a straight, power law like trend, but also lacked a clearly distinguishable peak, instead showing signs of a knee halfway through the frequency range. The fits here are somewhat arguable, as will become clear from the excessive uncertainties. However, all is not lost as these sources, being without peaks, are not used in the final results of this thesis.

1. *J0103+4322* shows signs of peaking in the high frequency regime, yet has an exceptionally flat background spectrum which can possibly be attributed to multiple overlapping SSA components. It took careful manipulation of the MCMC code to fit a GPS+power law spectrum to it as the thick spectral index k is extremely steep (~ 2.5, the maximum physical value).

2. *J0104+3840* shows clear signs of a peak, but as the thick spectral index k was quite poorly defined by only the two non-VLA data points, the uncertainties there are rampant. This can be combated by tightly constraining k, but that would essentially be tampering with the results.

3. *J0105+3533* is an ordinary power law spectrum.

4. *J0106+3522* is an ordinary GPS.

5. *J0106+4422* shows GPS behaviour, but suffers similar issues as J0104+3840 does: a poorly constrained k, in addition to an offset final data point, which might be indicative of a second, younger synchrotron component in the HFP regime.

6. *J0116+4420* is an ordinary power law spectrum.

7. *J0119+4054* is an ordinary power law spectrum.

8. *J0123+3842* is an ordinary GPS+power law spectrum.

9. *J0123+3806* is an ordinary GPS+power law spectrum.

10. *J0123+3641* is an ordinary GPS. This source was used to test the LSq and MCMC algorithms because of its typical shape.

11. *J0128+4003* is an ordinary aged power law spectrum. It was the first in our sample to be identified as such and thus sparked efforts to adapt the code to this kind of behaviour.

12. *J0128+4439* is an ordinary GPS.

13. *J0129+4044* is a double power law. Originally suspected a GPS+power law spectrum, after repeated attempts to find a peak failed the double power law solution was applied instead to much greater success.

14. *J0131+4428* shows GPS+power law like behaviour, but in this case the thin spectral index l proved to be under-defined. After adaptation of the code, this spectrum was successfully fitted, although the thick spectral index was outputted as ~ 2.5, an exceptionally high value.

15. *J0132+4345* is an ordinary power law.

16. *J0134+4018* is an ordinary aged power law spectrum.

17. *J0135+3631* is an ordinary aged power law spectrum.

18. *J0136+3905* is an ordinary power law.

19. *J0136+3545* is an ordinary aged power law spectrum.

20. *J0136+4401* is an ordinary aged power law spectrum.

21. *J0143+3705* is an ordinary GPS+power law spectrum.

22. *J0151+4332* is an ordinary power law, albeit with large scatter around the power law trend.

23. *J0151+4417* is an ordinary GPS+power law spectrum.

24. *J0103+4322* is a double power law. Attempts have been made to fit it with a GPS+power law spectrum but as l is barely defined this was unsuccessful, leading to a double power law with large uncertainties.

25. *J0154+3558* shows a GPS peak, but there are signs of an exceptional falloff in the high frequency regime. This could possibly be due to a highly aged background electron population overlaid onto the GPS.

26. *J0154+4433* is an ordinary GPS.

27. *J0156+4459* is an ordinary power law.

28. *J0157+4120* is an ordinary power law.

29. *J0159+4059* is an ordinary GPS+power law.

30. *J0159+4144* is a GPS which shows similar issues to J0154+3558.

31. *J1702+5511* is a GPS which shows similar issues to J0104+3840.

32. *J1702+5812* is an ordinary GPS.

33. *J1711+5803* is an aged power law, with signs of falloff at lower frequencies than usual.

34. *J1715+5724* is an ordinary power law.

35. *J1720+5541* is an ordinary GPS+power law.

36. *J1720+5926* is an ordinary power law.

37. *J1726+5826* is a GPS with small spectral indices. It suffers extreme degeneracies in its parameters (see Figure 17), presumably due to its small indices.

38. *J1728+5532* is a GPS with an unusually gently sloping k.

39. *J1735+5650* is an ordinary double power law.

40. *J1746+5659* is an ordinary GPS+power law.

41. *J1747+5902* shows signs of GPS+power law behaviour. However, l proved under-defined to such an extent that the spectral peak falls off the observed frequency band. Hence, although fitted with a GPS+power law model, it does not have a peak.

42. *J1756+5918* is a power law, but with a slightly offset 325 MHz measurement. This can be due to a small degree of SSA or FFA, but it was not enough for the LSq or MCMC of the SSA GPS model to pick it up.

43. *J1756+5806* is an ordinary power law.

44. *J1758+5951* is a GPS, but with an unusual amount of scatter in the data around the model's trend. As a result uncertainties are considerable.

In total some 10 sources have uncertain spectra, but these uncertainties do not affect later sections of the results. Therefore we take these spectra as final.

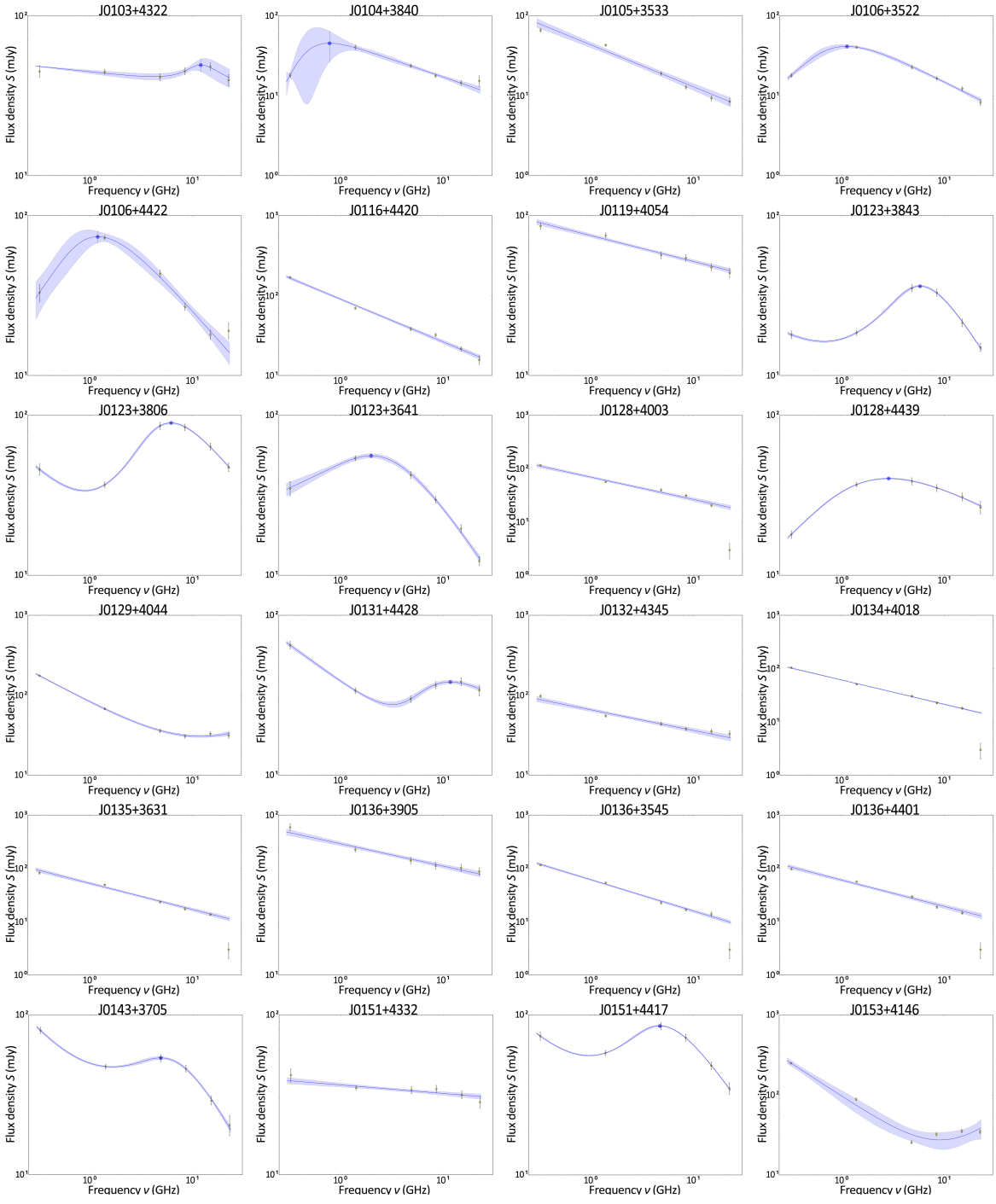


Figure 11: The first portion of the observed sources. List continues on the next page.

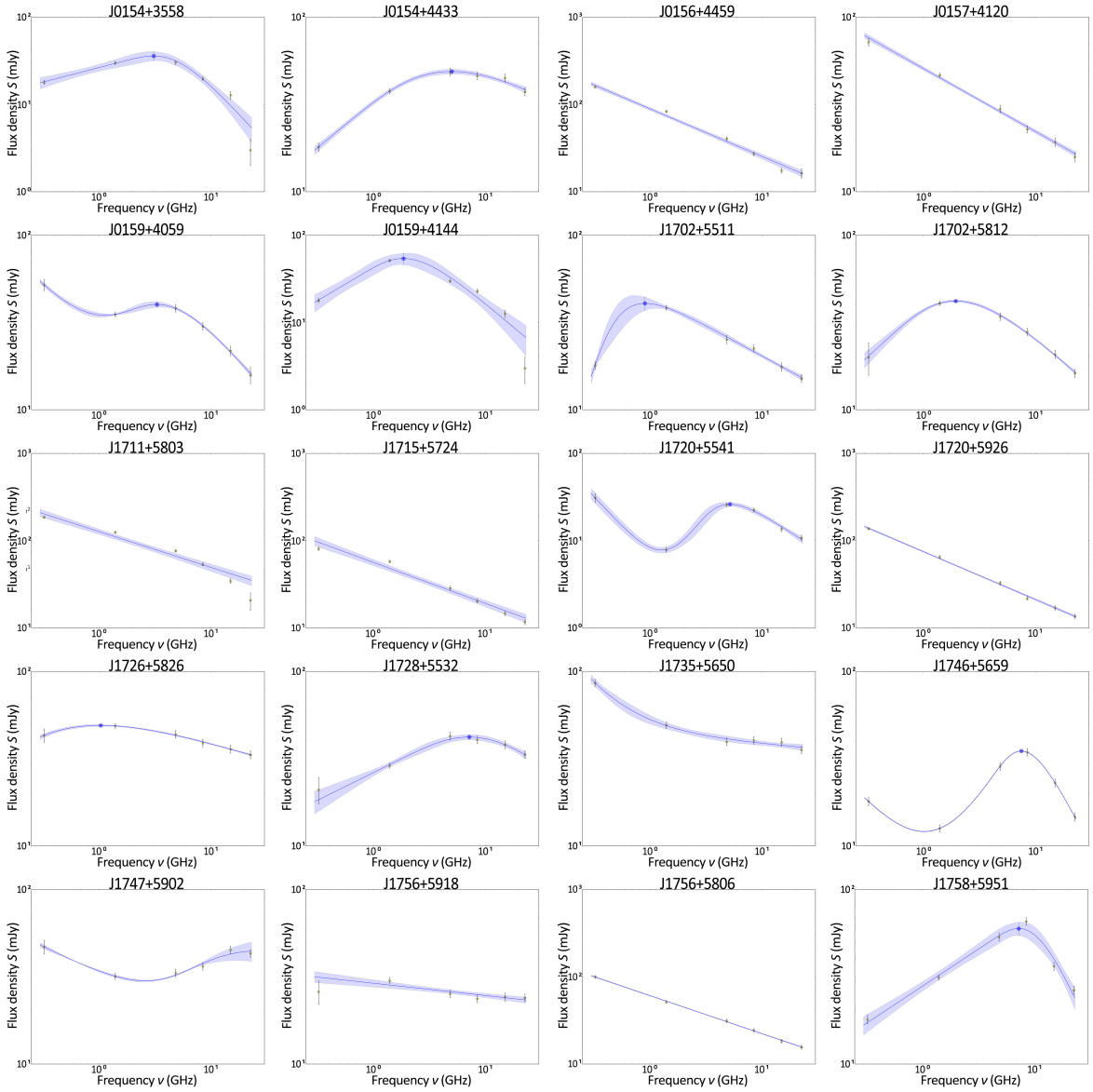


Figure 12: The second portion of the observed sources.

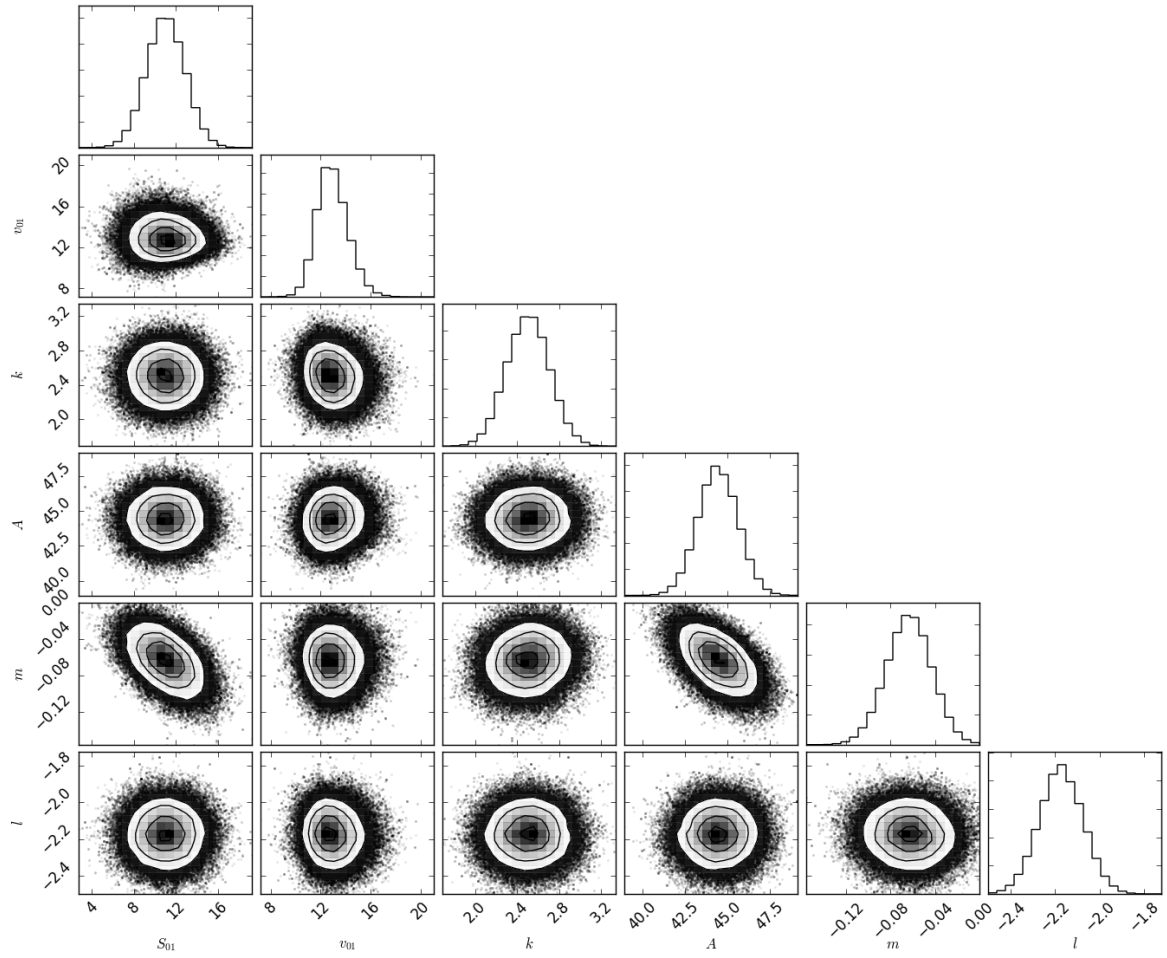


Figure 13: The contour plots of J0103+4322. Notice the relatively large uncertainties in k and l, contributing to the large final uncertainty in the fit. These contour plots were constructed from the MCMC algorithm (see Appendix B.2).

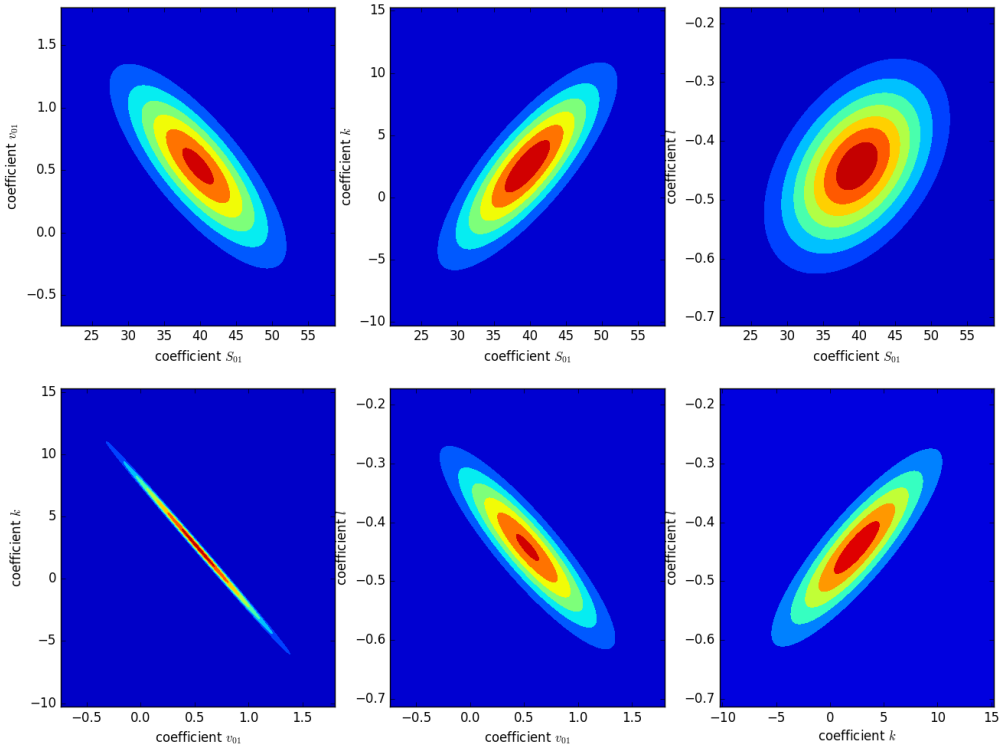


Figure 14: The contour plots of J0104+3840. Notice the major uncertainty in k and exceptionally high degeneracy between k and ν_{01}, contributing to the large final uncertainty in the thick spectral region of the fit. These contour plots were constructed from the LSq algorithm (see Appendix B.1).

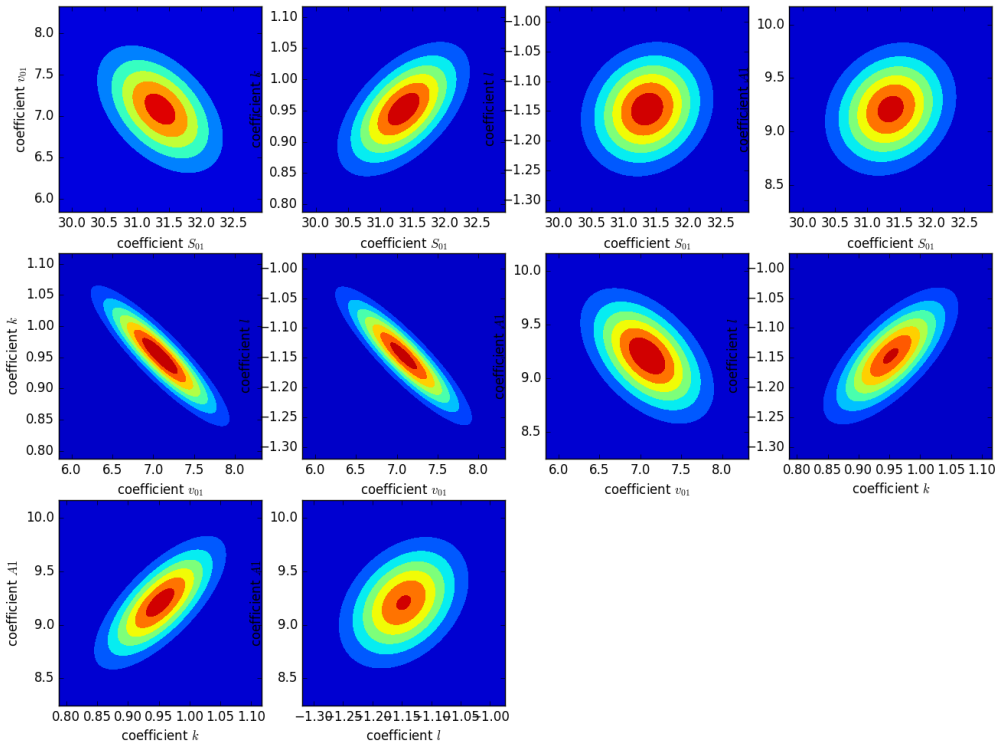


Figure 15: The contour plots of J0123+3842. These contour plots are an acceptable example for SSA GPS+power law fit contours: roughly circular shapes mean that there is only minor degeneracy, which reflects well in the uncertainties in the final fit. These contour plots were constructed from the LSq algorithm.

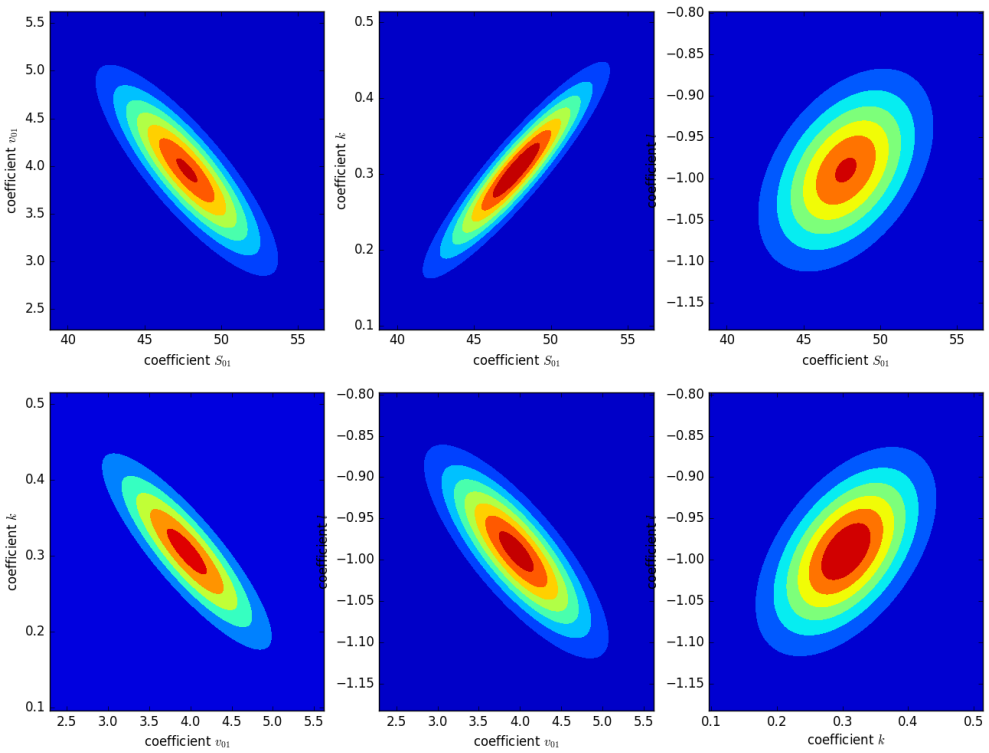


Figure 16: The contour plots of J0123+3641. Although the degeneracies are not minimal, these plots are an acceptable example for SSA GPS fits, as the magnitude of the uncertainties is small. These contour plots were constructed from the LSq algorithm.

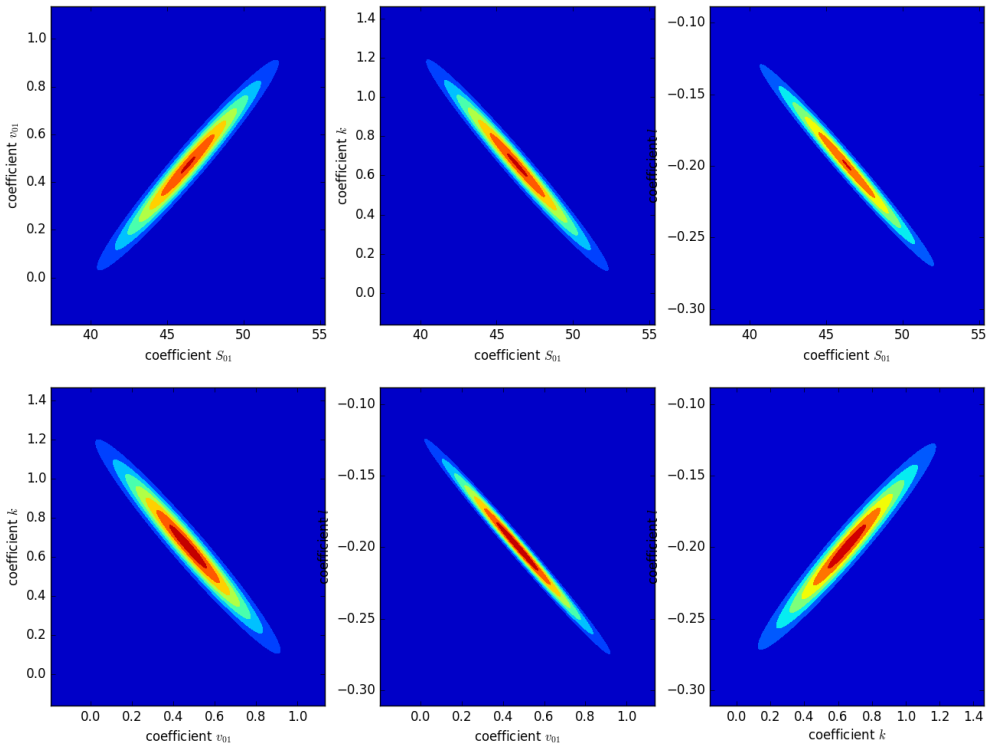


Figure 17: The contour plots of J1726+5826. As is clearly visible the degeneracies in these plots are significant, yet because of the small magnitude of the uncertainties it has little repercussions for the final fit. These contour plots were constructed from the LSq algorithm.

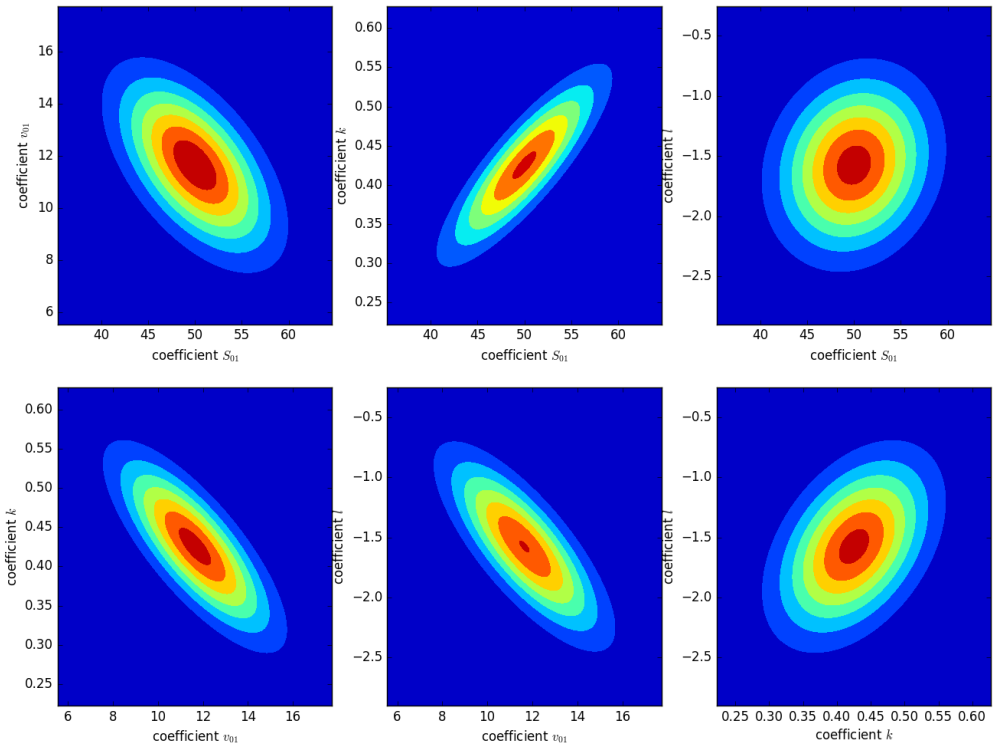


Figure 18: The contour plots of J1758+5951. While the spectral fit displays considerable uncertainties, degeneracies in these contours are smaller even than in the far more accurate cases of Figures 16 and 17, the uncertainties in the final fit are substantially larger, mainly due to sizeable magnitude of the uncertainties in the parameters themselves, leading us to conclude that degeneracies (and thus covariances) hold little to no direct relation to the quality of the fit. These contour plots were constructed from the LSq algorithm.

index	Source	type	A (mJy GHz^{-m})	m
2	J0105+3533	regular	43.4 ± 3.5	-0.5 ± 0.05
5	J0116+4420	regular	90.3 ± 2.6	-0.5 ± 0.02
6	J0119+4054	regular	75.0 ± 1.9	-0.2 ± 0.01
10	J0128+4003	aged	69.5 ± 3.2	-0.4 ± 0.03
14	J0132+4345	regular	65.5 ± 3.2	-0.3 ± 0.03
15	J0134+4018	aged	60.9 ± 0.9	-0.5 ± 0.01
16	J0135+3631	aged	52.3 ± 2.9	-0.5 ± 0.04
17	J0136+3905	regular	66.1 ± 1.9	-0.1 ± 0.02
18	J0136+3545	aged	61.2 ± 2.1	-0.6 ± 0.03
19	J0136+4401	aged	60.2 ± 3.4	-0.5 ± 0.04
21	J0151+4332	regular	36.4 ± 1.0	-0.1 ± 0.02
26	J0156+4459	regular	89.2 ± 3.5	-0.5 ± 0.03
27	J0157+4120	regular	50.8 ± 1.2	-0.4 ± 0.01
32	J1711+5803	regular	45.4 ± 4.7	-0.6 ± 0.06
33	J1715+5724	regular	56.3 ± 4.6	-0.5 ± 0.04
35	J1720+5926	regular	74.9 ± 1.3	-0.5 ± 0.01
41	J1756+5918	regular	29.0 ± 1.4	-0.1 ± 0.02
42	J1756+5806	regular	60.9 ± 0.6	-0.4 ± 0.01

Table 1: The coefficients and their uncertainties for the 18 power law fits. The first column refers to the number index of that source within the python program used. The second column gives the source designation, the third specifies the subtype of power law, the fourth gives the power law amplitude and its uncertainty, the fifth gives the power law index and its uncertainty.

index	Source	A_1 (mJy GHz^{-m})	m_1	A_2 (mJy GHz^{-m})	m_2
12	J0129+4044	79.6 ± 22.2	-0.69 ± 0.002	3.6 ± 8.3	0.61 ± 0.064
38	J1735+5650	9.8 ± 170.5	-1.25 ± 0.727	43.3 ± 118.9	-0.05 ± 0.007
23	J0153+4146	97.0 ± 432.7	-0.85 ± 0.037	1.3 ± 24.1	1.01 ± 1.379

Table 2: The coefficients and their uncertainties for the 3 power law fits. The first column refers to the number index of that source within the python program used. The second column gives the source designation, the third specifies the subtype of power law, the fourth gives the power law amplitude and its uncertainty, the fifth gives the power law index and its uncertainty. The uncertainties here are exceptionally large - fortunately, spectral data for the double power laws are irrelevant for the final results of this thesis.

index	Source	S_0 (mJy)	ν_0 (GHz)	k	l
1	J0104+3840	39.7 ± 6.4	0.53 ± 0.42	2.50 ± 4.26	-0.44 ± 0.09
3	J0106+3522	41.2 ± 1.7	1.03 ± 0.31	1.12 ± 0.32	-0.65 ± 0.07
4	J0106+4422	73.7 ± 6.4	1.25 ± 0.77	0.94 ± 0.52	-0.73 ± 0.18
9	J0123+3641	47.8 ± 3.0	3.96 ± 0.56	0.31 ± 0.07	-0.99 ± 0.06
11	J0128+4439	40.1 ± 0.5	2.41 ± 0.30	0.63 ± 0.04	-0.35 ± 0.03
24	J0154+3558	29.3 ± 5.0	5.43 ± 1.37	0.33 ± 0.12	-1.46 ± 0.33
25	J0154+4433	48.8 ± 1.6	4.96 ± 1.24	0.53 ± 0.06	-0.38 ± 0.08
29	J0159+4144	52.0 ± 7.8	2.37 ± 0.93	0.76 ± 0.23	-1.10 ± 0.33
30	J1702+5511	35.5 ± 1.1	0.56 ± 0.17	2.06 ± 1.11	-0.35 ± 0.04
31	J1702+5812	41.6 ± 0.8	2.37 ± 0.33	0.59 ± 0.09	-0.60 ± 0.04
36	J1726+5826	46.4 ± 3.0	0.47 ± 0.22	0.65 ± 0.27	-0.20 ± 0.04
37	J1728+5532	38.6 ± 3.4	14.08 ± 5.86	0.32 ± 0.09	-0.65 ± 0.23
43	J1758+5951	50.0 ± 4.9	11.65 ± 2.04	0.43 ± 0.07	-1.58 ± 0.44

Table 3: The coefficients and their uncertainties for the 13 SSA GPS fits. The first column refers to the number index of that source within the python program used. The second column gives the source designation, the third gives the scale flux and its uncertainty, the fourth gives the scale frequency and its uncertainty, the fifth gives the thick spectral index and its uncertainty, the sixth gives the thin spectral index and its uncertainty.

index	Source	ν_{max} (GHz)	S_{max} (mJy)
1	J0104+3840	0.79 ± 0.02	45.3 ± 18.6
3	J0106+3522	1.13 ± 0.02	41.3 ± 2.1
4	J0106+4422	1.20 ± 0.02	73.7 ± 6.1
9	J0123+3641	1.99 ± 0.02	55.9 ± 1.7
11	J0128+4439	2.88 ± 0.02	40.3 ± 0.4
24	J0154+3558	3.08 ± 0.02	35.9 ± 4.3
25	J0154+4433	5.01 ± 0.02	48.8 ± 1.6
29	J0159+4144	1.86 ± 0.02	54.1 ± 8.1
30	J1702+5511	0.90 ± 0.02	40.7 ± 3.6
31	J1702+5812	1.95 ± 0.02	42.0 ± 0.6
36	J1726+5826	1.04 ± 0.02	49.1 ± 0.6
37	J1728+5532	7.17 ± 0.02	42.0 ± 1.4
43	J1758+5951	7.26 ± 0.02	59.8 ± 5.2

Table 4: The peaks and their uncertainties for the SSA GPS fits. The first column refers to the number index of that source within the python program used. The second column gives the source designation, the third gives the peak frequency and its uncertainty, the fourth gives the peak flux and its uncertainty. The peak frequency was taken as the step size of the bisection algorithm and is thus constant.

index	Source	S_0 (mJy)	ν_0 (GHz)	k	l	A	m
0	J0103+4322	11.5 ± 1.8	12.75 ± 1.24	2.52 ± 0.20	-2.18 ± 0.10	44.6 ± 1.2	-0.07
7	J0123+3843	31.4 ± 0.5	7.08 ± 0.41	0.95 ± 0.06	-1.15 ± 0.06	9.2 ± 0.3	-0.46
8	J0123+3806	82.8 ± 1.1	6.09 ± 0.35	1.29 ± 0.06	-0.80 ± 0.04	21.2 ± 0.6	-0.63
13	J0131+4428	23.2 ± 1.6	8.72 ± 0.84	1.91 ± 0.35	-0.31	38.3 ± 1.0	-0.47 ± 0.01
20	J0143+3705	41.6 ± 1.7	8.39 ± 0.85	0.48 ± 0.06	-1.24 ± 0.11	26.0 ± 1.0	-0.83
22	J0151+4417	72.9 ± 1.0	6.01 ± 0.39	0.87 ± 0.06	-0.98 ± 0.04	31.5 ± 0.7	-0.64
28	J0159+4059	32.2 ± 1.4	4.52 ± 1.36	0.81 ± 0.28	-0.78 ± 0.12	20.0 ± 1.4	-0.74
34	J1720+5541	24.9 ± 1.5	4.61 ± 1.01	1.93 ± 0.39	-0.85 ± 0.17	6.3 ± 0.9	-1.41
39	J1746+5659	32.4 ± 0.2	8.80 ± 0.10	1.10 ± 0.01	-1.34 ± 0.02	7.4 ± 0.1	-0.72
40	J1747+5902	28.2 ± 3.1	13.37 ± 2.85	1.17 ± 0.02	-0.12 ± 0.01	32.1 ± 1.9	-0.33 ± 0.01

Table 5: The coefficients and their uncertainties for the 10 SSA+power law GPS fits. The first column refers to the number index of that source within the python program used. The second column gives the source designation, the third gives the scale flux and its uncertainty, the fourth gives the scale frequency and its uncertainty, the fifth gives the thick spectral index and its uncertainty, the sixth gives the thin spectral index and its uncertainty, the seventh gives the power law amplitude and its uncertainty, the eighth gives the power law spectral index and its uncertainty. As m was fixed for most of the sources, it had no uncertainty. In case 13, instead l was fixed, and in case 40 neither was.

index	Source	ν_{max} (GHz)	S_{max} (mJy)
0	J0103+4322	12.07 ± 0.01	49.2 ± 4.4
7	J0123+3843	5.80 ± 0.02	36.2 ± 0.5
8	J0123+3806	6.20 ± 0.02	89.6 ± 1.0
13	J0131+4428	11.77 ± 0.02	38.2 ± 0.9
20	J0143+3705	4.81 ± 0.02	53.7 ± 1.1
22	J0151+4417	4.73 ± 0.02	85.6 ± 0.9
28	J0159+4059	3.29 ± 0.02	40.1 ± 1.5
34	J1720+5541	5.18 ± 0.02	26.0 ± 1.4
39	J1746+5659	7.49 ± 0.02	35.0 ± 0.2
40	J1747+5902	N/A	N/A

Table 6: The peaks and their uncertainties for the SSA GPS fits. The first column refers to the number index of that source within the python program used. The second column gives the source designation, the third gives the peak frequency and its uncertainty, the fourth gives the peak flux and its uncertainty. The peak frequency was taken as the step size of the bisection algorithm, but as in the SSA+power law GPS case the bisection interval was shorter yet the number of steps constant, this varies slightly. In the case of index 40, this source behaved like an SSA+power law GPS but lacked a peak on the observed interval, hence the N/A's.

4 K-z relation

As not all GPS sources have well defined spectroscopic redshifts, another method has been devised which relates the K-band magnitude of a source to the redshift z, known as the K-z relation. This chapter discusses the K-band and subsequent redshift measurements of the sample, as well as the following calculations of luminosity distance and angular and linear size.

4.1 Introduction to the K-z relation

Light that travels over cosmic distances experiences a redshift z due to the cosmic expansion that causes the observed frequency of the emitted light to change via the relation,

$$z = \frac{\nu_{emit}}{\nu_{obs}} - 1 \approx \frac{v_{rad}}{c}. \tag{31}$$

Wherein ν_{emit} is the frequency of the emitted light, ν_{obs} the frequency of the observed light and v_{rad} the radial velocity of the source. This means that light from objects with a positive redshift and thus a positive radial velocity (i.e. away from us) is shifted towards lower frequencies.

The near-infrared K-band ($\sim 2.2\ \mu$m) magnitude of a radio galaxy is dominated by starlight from older stars and therefore associated with evolved ellipticals, although emission lines that have been redshifted into the K-band window also contribute to the band flux. The strength of these lines is intrinsically related to the underlying quasar continuum and hence the radio emission [Jarvis et al., 2001]. In our sample however, this contribution is negligible as the spectra show no lines.

Lilly and Longair found in 1984 that there was a correlation between the K-band magnitude and the redshift of radio galaxies. They found that the dispersion around this trend in the K-direction was approximately constant, Gaussian and quite tight out to redshift $z \sim 2$. The relation they found was not entirely in line with the expectations and attempts were made to explain this by assuming a changing number of red giant stars [Lilly and Longair, 1984]. Over time the dispersion along the K-band was narrowed down, first by Jarvis et al. to around 0.59, then by Cruz et al. to an improved significance of 0.593 ± 0.02 mag [Jarvis et al., 2001], [Cruz et al., 2007], the value used in this thesis as well. Willott et al. in 2003 found the optimal fit through the data to follow the logarithmic quadratic polynomial function,

$$K(z) = 17.37 + 4.53 \cdot \log_{10}(z) - 0.31 \cdot \log_{10}^2(z) \tag{32}$$

Where $K(z)$ is the magnitude in the K-band. This is the function we use in our study as well.

4.2 Sample selection

In order to obtain as complete a sample in terms of redshifts as possible, for those sources in the original sample of which a spectroscopic redshift was not known [Marlow et al., 2000], infrared measurements were taken using the United Kingdom Infrared Telescope (UKIRT) by McKean (2003). A limitation in the fact that only sources with a right ascension between 01^h and 02^h could be observed led to a final completeness of the sample (in terms of redshift by spectroscopy and K-band magnitude, which leads to K-z relation redshift) for 30/45 sources, or redshift completeness 66.6%. K-band measurements within 5 arcseconds are provided in Table 7.

4.3 Methodology

Using the *normal* function from the *numpy* module that draws random numbers from a Gaussian distribution given some mean and dispersion, five million data points were generated in $K, \log_{10}(z)$-space. This was done by first computing the mean in the K-magnitude by generating a wholly random $\log_{10}(z)$-value on the interval $-1 < \log_{10}(z) < \log_{10}(5)$ using the *numpy.random* function, plugging this into the K-z relation and using the output as the mean for the Gaussian random number generator, which then returned a K-band magnitude. This created five million data points with a random z and a Gaussian dispersion around the K-z relation of value $\sigma_K = 0.593$, as required (see Figure 19 for the K-z relation

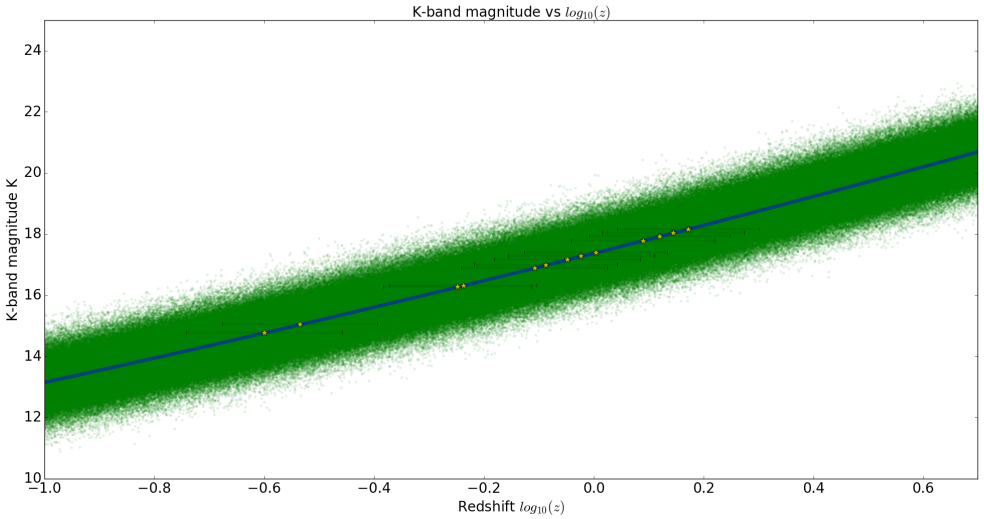


Figure 19: The K-z relation in $\log_{10}(z)$ space. Yellow stars represent the points of which K-band data is known, the blue line is the idealized K-z relation according to equation 32 and the green points are the 5 million normally distributed, simulated data points.

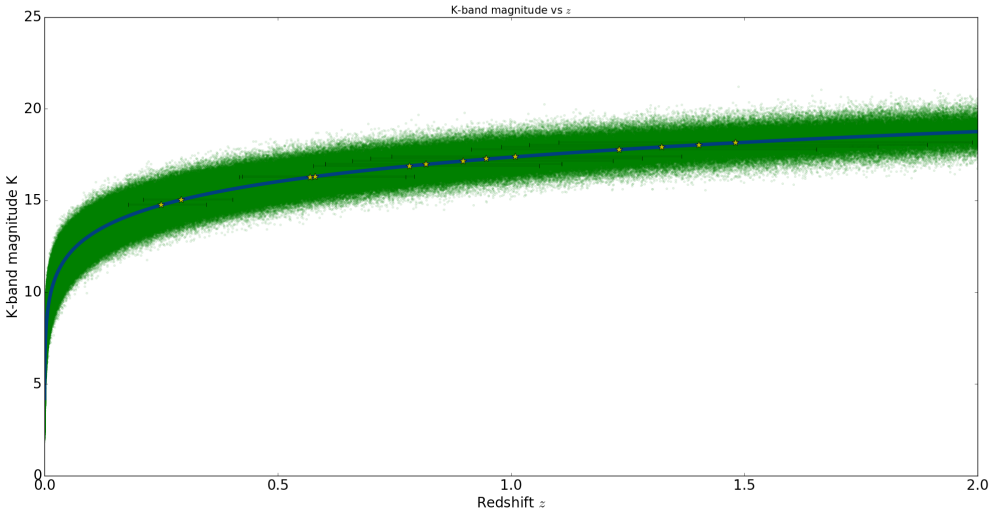


Figure 20: The K-z relation in linear z space.

in log-space and Figure 20 in linear space).

A sampler was then created which took a K-band magnitude and collected all of the random data points within a small interval around this K-magnitude (sample size ± 0.05 mag). Upon plotting the $\log_{10}(z)$-values within this interval in a histogram, this yielded a set of Gaussian distributions (see Figure 21), as one would expect from the near linear trend visible in Figure 19. When translated to linear z values this reproduced a log-normal distribution as expected (see Figure 22).

The mean (μ_y) and dispersion (σ_y) of a Gaussian distribution in $\log_{10}(x) = y$ are given by,

$$\mu_y = \frac{\sum_{i=1}^{N} y_i}{N}, \tag{33}$$

$$\sigma_y = \sqrt{\frac{\sum_{i=1}^{N} (y_i - \mu_y)^2}{N}}. \tag{34}$$

In which N is the length of the data set and y_i the i-th value in the data set. Alternatively, the corresponding statistics in the subsequent log-normal distribution in the variable x itself, namely the median and the lower and upper 68% confidence interval limits, are given by,

$$\mu_x = 10^{\mu_y}, \tag{35}$$

$$\sigma_{x,low} = 10^{\mu_y - \sigma_y} - \mu_x, \tag{36}$$

$$\sigma_{x,upp} = 10^{\mu_y + \sigma_y} - \mu_x. \tag{37}$$

Where $\sigma_{x,low}$ and $\sigma_{x,low}$ denote the upper and lower uncertainties in x. They can also be computed directly from the log-normal distribution but this is substantially more convoluted so instead the data in $K, \log_{10}(z)$-space was used to compute the above statistics in the given sequence. The results were then outputted to a table and plotted, and the resulting K-band magnitude/redshift values were overlaid onto the K-z relation plot (see Figures 19 and 20, sources are indicated by yellow stars) (see Appendix B.3).

Source	spectrum	K (mag)	$\log_{10}(z)$	redshift z
J0103+4322	GPS+PL	14.78±0.03	-0.6±0.14	$0.25^{0.10}_{0.07}$
J0106+3522	GPS	16.99±0.04	-0.087±0.13	$0.82^{0.29}_{0.21}$
J0106+4422	GPS	17.29±0.05	-0.02±0.13	$0.95^{0.34}_{0.25}$
J0116+4420	PL	16.33±0.03	-0.24±0.13	$0.58^{0.21}_{0.15}$
J0119+4054	PL	16.29±0.03	-0.25±0.14	$0.57^{0.21}_{0.15}$
J0123+3843	GPS+PL	16.99±0.05	-0.087±0.13	$0.82^{0.29}_{0.21}$
J0123+3641	GPS	18.17±0.08	0.17±0.13	$1.48^{0.51}_{0.38}$
J0129+4044	DPL	17.94±0.08	0.12±0.13	$1.32^{0.45}_{0.34}$
J0136+4401	APL	17.79±0.06	0.09±0.13	$1.23^{0.43}_{0.32}$
J0143+3705	GPS+PL	17.17±0.04	-0.048±0.13	$0.90^{0.32}_{0.23}$
J0153+4146	DPL	16.90±0.04	-0.11±0.13	$0.78^{0.28}_{0.21}$
J0154+3558	GPS	17.40±0.05	0.004±0.13	$1.01^{0.35}_{0.26}$
J0154+4433	GPS	15.05±0.03	-0.53±0.14	$0.29^{0.11}_{0.08}$
J0159+4144	GPS	18.05±0.08	0.15±0.13	$1.40^{0.48}_{0.36}$
J0123+3806	GPS+PL			1.656±0.004
J0128+4003	APL			3.525±0.003
J0128+4439	GPS			0.228±0.001
J0131+4428	GPS+PL			1.123±0.001
J0132+4345	PL			1.812±0.001
J0136+3545	APL			1.871±0.001
J0151+4332	PL			2.192±0.002
J0151+4417	GPS+PL			1.976±0.003
J0156+4459	PL			0.214±0.001
J0157+4120	PL			0.0811±0.0001
J1711+5803	PL			0.1465±0.0001
J1715+5724	PL			0.0273±0.0001
J1720+5926	PL			0.5878±0.0003
J1728+5532	GPS			1.404±0.002
J1747+5902	GPS+PL			0.981±0.001
J1756+5806	PL			0.192±0.002

Table 7: The results for the K-z relation fitting of the sources for which K-band magnitude data is known. As z is log-normally distributed, it has non-Gaussian upper and lower uncertainty boundaries. Sources in the lower section of the table have been determined spectroscopically (16 by spec., 14 by K-z) by Marlow et al. (2000). The second column indicates the fit type: GPS is a normal SSA GPS spectrum, GPS+PL is SSA GPS+power law, PL is regular power law, APL is aged power law, DPL is double power law.

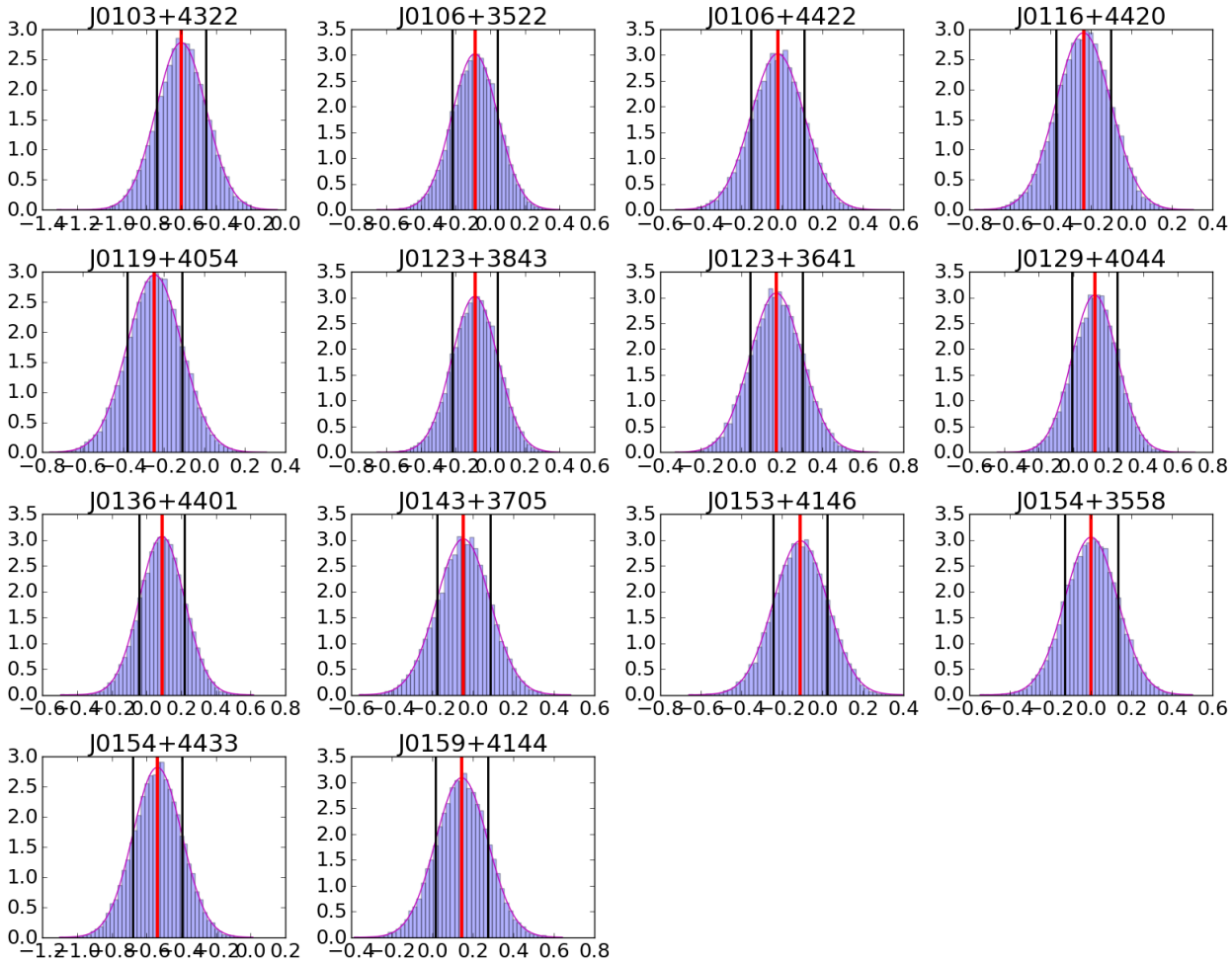


Figure 21: The log-redshift distributions around the sample K-band measurements. The red line indicates the mean, the black lines the symmetric dispersion. The red curve is the fitted continuous Gaussian distribution through the histogram data. The x-axes give the $\log_{10}(z)$-value, the y-axes the probability in arbitrary units.

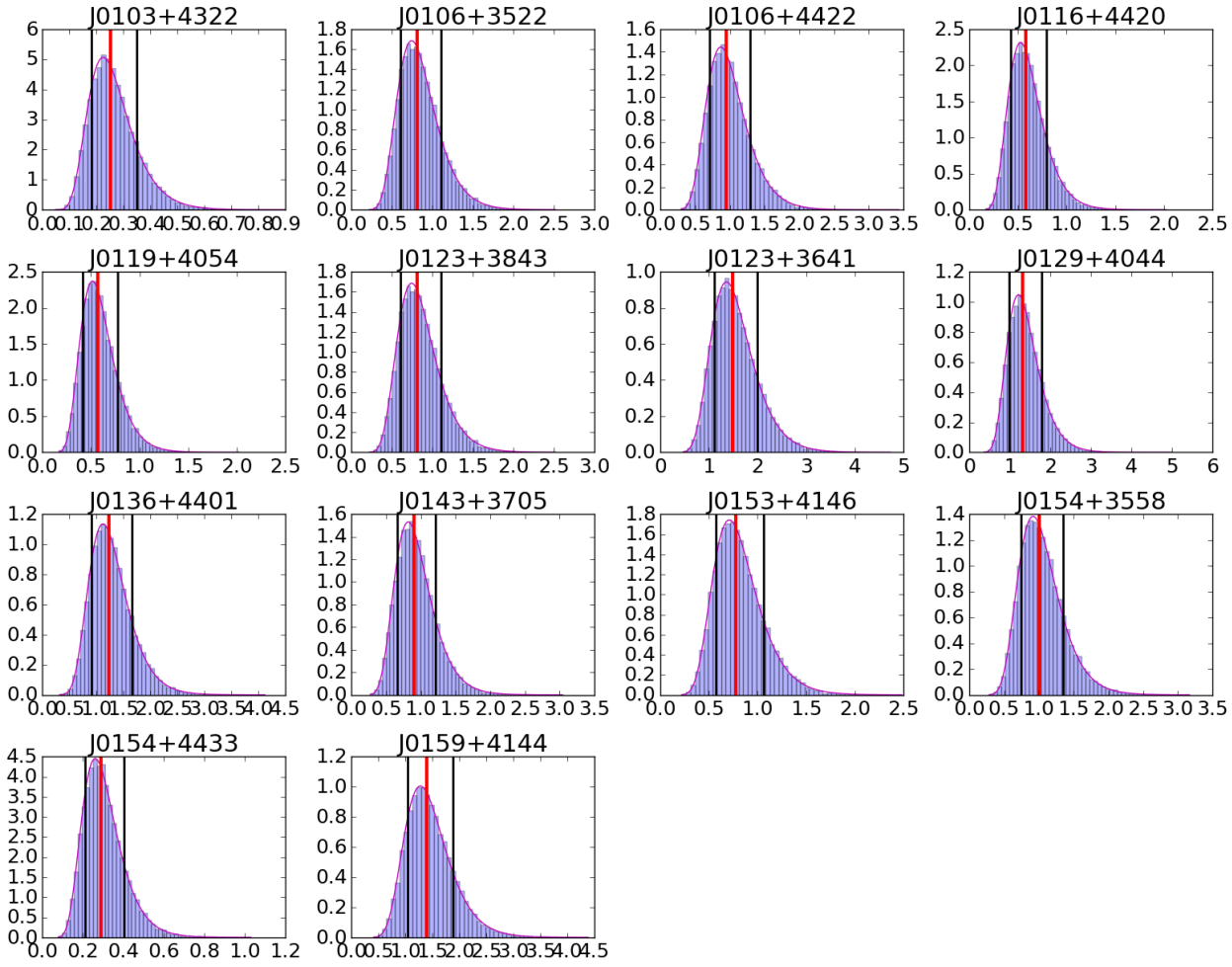
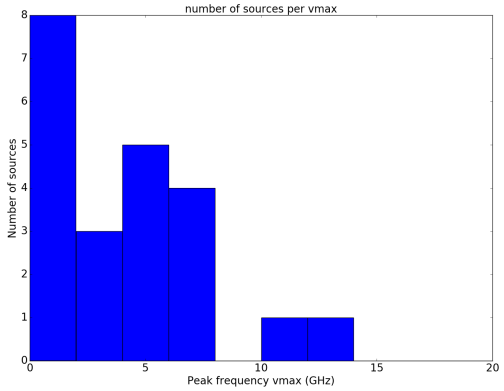
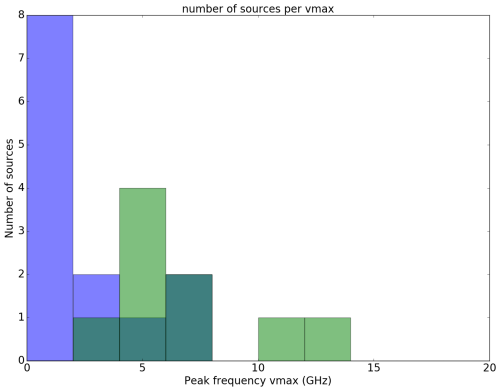


Figure 22: The redshift distributions around the sample K-band measurements. The red line indicates the median, the black lines the lower and upper boundaries of the 68% confidence interval. The red curve is the fitted continuous log-normal distribution through the histogram data. The x-axes give the z-value, the y-axes the probability in arbitrary units.

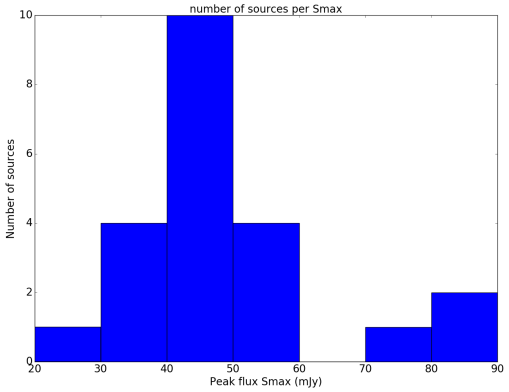


(a) SSA GPS and SSA+power law GPS populations combined.

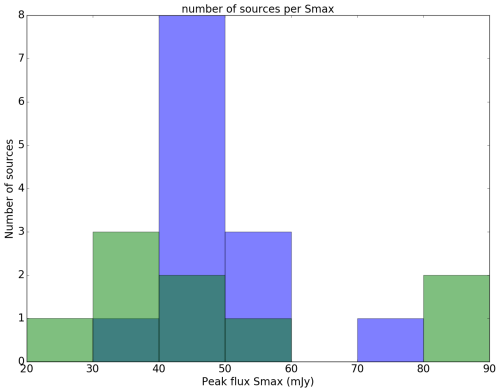


(b) Separate GPS populations (blue: SSA, green: SSA+power law).

Figure 23: Number counts for peak frequency ν_{max}. Bin width is 2 GHz.

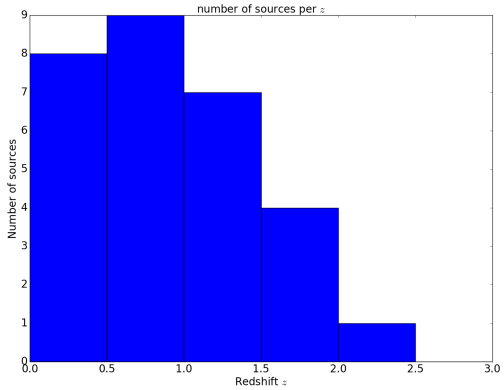


(a) SSA GPS and SSA+power law GPS populations combined.

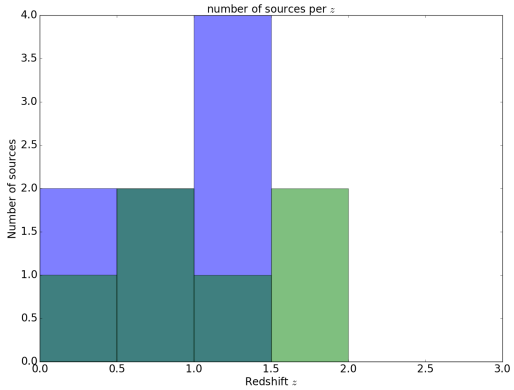


(b) Separate GPS populations (blue: SSA, green: SSA+power law).

Figure 24: Number counts for peak flux density S_{max}. Bin width is 10 mJy



(a) SSA GPS and SSA+power law GPS populations combined.



(b) Separate GPS populations (blue: SSA, green: SSA+power law).

Figure 25: Number counts for redshift z. Bin width is 0.5 redshift.

4.4 Results

The redshift results for the sample that underwent the K-z treatment are provided in the upper section of Table 7. Combined with the rest of the sample, with spectroscopic redshifts taken from [Marlow et al., 2000] and NASA Extragalactic Database (NED), we obtain the full table. These values can be translated to a histogram of number count of sources per redshift bin (Figure 25). This has also been done for peak frequency (Figure 23) and peak flux (Figure 24) of the GPS and GPS+power law sources (see Appendix B.5). While the separate populations of SSA GPS and SSA+power law GPS in the peak-related Figures add up to their combined counterparts, in Figure 25 this is not the case. This is because the combined plot also includes power law spectra whose redshift is known.

When studying the redshift number counts it becomes apparent that there is no clear sign of bimodal behaviour regarding the two separate populations. It was suggested by Dr. McKean that a bimodal trend might arise between a population at small redshift (radio galaxies) and one at large redshift (radio quasars), possibly coinciding with the separation between SSA GPS and SSA+power law GPS sources. Figure 25b however clearly shows that this is not the case, and from Figure 25a we can continue to conclude that there is no secondary peak at higher redshift.

A slightly more differentiated situation can be spotted in Figures 23 and 24. In frequency space, it can be observed that the peak frequency of pure SSA GPS sources is in general lower than those of their counterparts in the SSA+power law GPS sample, with an exceptional spike between 0 and 2 GHz. The SSA+power law GPS population shows a preference towards the multi-GHz regime, with sporadic counts exceeding 10 GHz prompting the suggestion that these would be rather classified as HFPs. In terms of peak flux, another trend becomes apparent with the SSA+power law GPS possessing an on average lower peak flux density than their pure SSA GPS counterparts.

Combining all redshifts obtained by using the K-z relation with spectroscopic redshifts from NED and Marlow et al. (2000) and averaging their squares, then taking the root of this quantity, we obtain a root mean square (RMS) redshift - the RMS is a useful statistic which gives an impression of the spread of a quantity in its space. Marlow et al. obtained an RMS of 1.27 with spread of 0.95 for a completeness level of 64% [Marlow et al., 2000] for the CLASS sample. It is interesting to note, however, that these values were seemingly obtained by setting those redshift values that Marlow et al. failed to determine spectroscopically to 0, a practice that this thesis does not follow. Hence the RMS was determined from the raw Marlow redshift data, which for the sources in Marlow's sample with known redshifts gave RMS $\langle z \rangle = 1.77$ with an RMS spread of 1.01, rather than the value mentioned in the paper's abstract. Combining the Marlow et al. CLASS sample with the sources used in this thesis, an RMS of $\langle z \rangle = 1.45$ with an RMS spread of 0.87 for a completeness level of 66% was obtained. Sources within this thesis' sample with unknown redshifts were simply omitted from this computation, rather than taken at $z = 0$. This shows that the redshifts determined by this paper using the K-z relation significantly drift the RMS of the sample towards lower redshifts, indicating that McKean's observed sources are on average closer than Marlow's CLASS sample.

5 Discussion

This chapter discusses the synthesis of the results so far obtained. Cosmological statistics of the sample will be provided by combining the results of the spectral fitting and the K-z relation, an analysis and discussion will be provided regarding these final results and suggestions will be offered for possible next steps.

5.1 Synthesis of results

Comparing the results for the spectral fitting and the determination of redshift, be it by spectroscopy or by using the K-z relation, a sub-sample was constructed or those sources with a spectral turnover. In case a redshift was not known, the mean value of the other redshifts (∼1.01) was used as an informed estimate.

Source	redshift z	S_{max} (mJy)	ν_{obs} (GHz)	θ (mas)	D (Mpc)	d (pc)	ν_e (GHz)
J0104+3840 (!)	$1.00^{+0.35}_{-0.26}$	45.3 ± 18.6	0.79 ± 0.02	$0.28^{+2.60}_{-0.28}$	6879.0	2.3	1.58
J0106+3522 (!)	$0.82^{+0.29}_{-0.21}$	41.3 ± 2.1	1.13 ± 0.02	$0.36^{+0.39}_{-0.33}$	5347.5	2.8	2.05
J0106+4422 (!)	$0.95^{+0.34}_{-0.25}$	73.7 ± 6.1	1.20 ± 0.02	$0.4^{+0.59}_{-0.40}$	6496.0	3.3	2.34
J0123+3641 (!)	$1.48^{+0.51}_{-0.38}$	55.9 ± 1.7	1.99 ± 0.02	$0.22^{+0.25}_{-0.19}$	11201.0	1.9	4.94
J0128+4439	0.228 ± 0.001	40.3 ± 0.4	2.88 ± 0.02	$2.02^{+0.02}_{-0.02}$	1174.9	7.6	3.54
J0154+3558	$1.01^{+0.35}_{-0.26}$	35.9 ± 4.3	3.08 ± 0.02	$0.3^{+0.15}_{-0.14}$	6947.4	2.5	6.19
J0154+4433	$0.29^{+0.11}_{-0.08}$	48.8 ± 1.6	5.01 ± 0.02	$1.83^{+0.03}_{-0.03}$	1553.0	8.3	6.47
J0159+4144 (!)	$1.40^{+0.48}_{-0.36}$	54.1 ± 8.1	1.86 ± 0.02	$0.23^{+0.44}_{-0.23}$	10426.4	2.0	4.45
J1702+5511 (!)	$1.00^{+0.35}_{-0.26}$	40.7 ± 3.6	0.90 ± 0.02	$0.27^{+0.69}_{-0.27}$	6879.0	2.2	1.80
J1702+5812 (!)	$1.00^{+0.35}_{-0.26}$	42.0 ± 0.6	1.95 ± 0.02	$0.30^{+0.18}_{-0.14}$	6879.0	2.5	3.89
J1726+5826 (!)	$1.00^{+0.35}_{-0.26}$	49.1 ± 0.6	1.04 ± 0.02	$0.30^{+0.48}_{-0.30}$	6879.0	2.5	2.08
J1728+5532	1.404 ± 0.002	42.0 ± 1.4	7.17 ± 0.02	$0.26^{+0.01}_{-0.01}$	10486.6	2.3	17.25
J1758+5951	$1.00^{+0.35}_{-0.26}$	59.8 ± 5.2	7.26 ± 0.02	$0.47^{+0.05}_{-0.05}$	6879.0	3.9	14.53
J0103+4322	$0.25^{+0.10}_{-0.07}$	49.2 ± 4.4	12.07 ± 0.01	$2.69^{+0.02}_{-0.02}$	1306.2	10.9	15.09
J0123+3843	$0.82^{+0.29}_{-0.21}$	36.2 ± 0.5	5.80 ± 0.02	$0.45^{+0.04}_{-0.03}$	5347.5	3.5	10.53
J0123+3806	1.656 ± 0.004	89.6 ± 1.0	6.20 ± 0.02	$0.3^{+0.01}_{-0.01}$	12866.8	2.6	16.46
J0131+4428	1.123 ± 0.001	38.2 ± 0.9	11.77 ± 0.02	$0.36^{+0.01}_{-0.01}$	7945.6	3.1	24.98
J0143+3705	$0.90^{+0.32}_{-0.24}$	53.7 ± 1.1	4.81 ± 0.02	$0.47^{+0.06}_{-0.05}$	5995.4	3.8	9.11
J0151+4417	1.976 ± 0.003	85.6 ± 0.9	4.73 ± 0.02	$0.22^{+0.01}_{-0.01}$	15999.0	1.9	14.07
J0159+4059	$1.00^{+0.35}_{-0.26}$	40.1 ± 1.5	3.29 ± 0.02	$0.33^{+0.09}_{-0.07}$	6879.0	2.7	6.59
J1720+5541	$1.00^{+0.35}_{-0.26}$	26.0 ± 1.4	5.18 ± 0.02	$0.29^{+0.03}_{-0.02}$	6879.0	2.4	10.37
J1746+5659	$1.00^{+0.35}_{-0.26}$	35.0 ± 0.2	7.49 ± 0.02	$0.36^{+0.01}_{-0.01}$	6879.0	3.0	14.98

Table 8: Cosmological statistics for those 22 sources in the sample with peaks. All objects with redshift $1.00^{+0.35}_{-0.26}$ had their redshifts estimated by taking the approximate average of all the known redshifts in the sample, namely (very close to) unity. All symbols take their usual meanings, while θ denotes the angular extension of the source (angular diameter in the sky), D the luminosity distance to the source, d the corresponding linear extension and ν_e the emitted frequency of the spectral peak. Above the horizontal line are SSA GPS; below are SSA GPS+power law spectra. Those SSA GPS sources marked with (!) deviate from the expected trend in Figure 26.

5.1.1 Angular size

Given the peak flux, the redshift, the angular size θ and the magnetic field strength B, the peak frequency of a source can be computed via,

$$\nu_{max} = f(k) \cdot B^{1/5} S_{max}^{2/5} \theta^{-4/5} (1+z)^{1/5}. \tag{38}$$

[Kellermann and Pauliny-Toth, 1981] This can be rewritten for the angular size as,

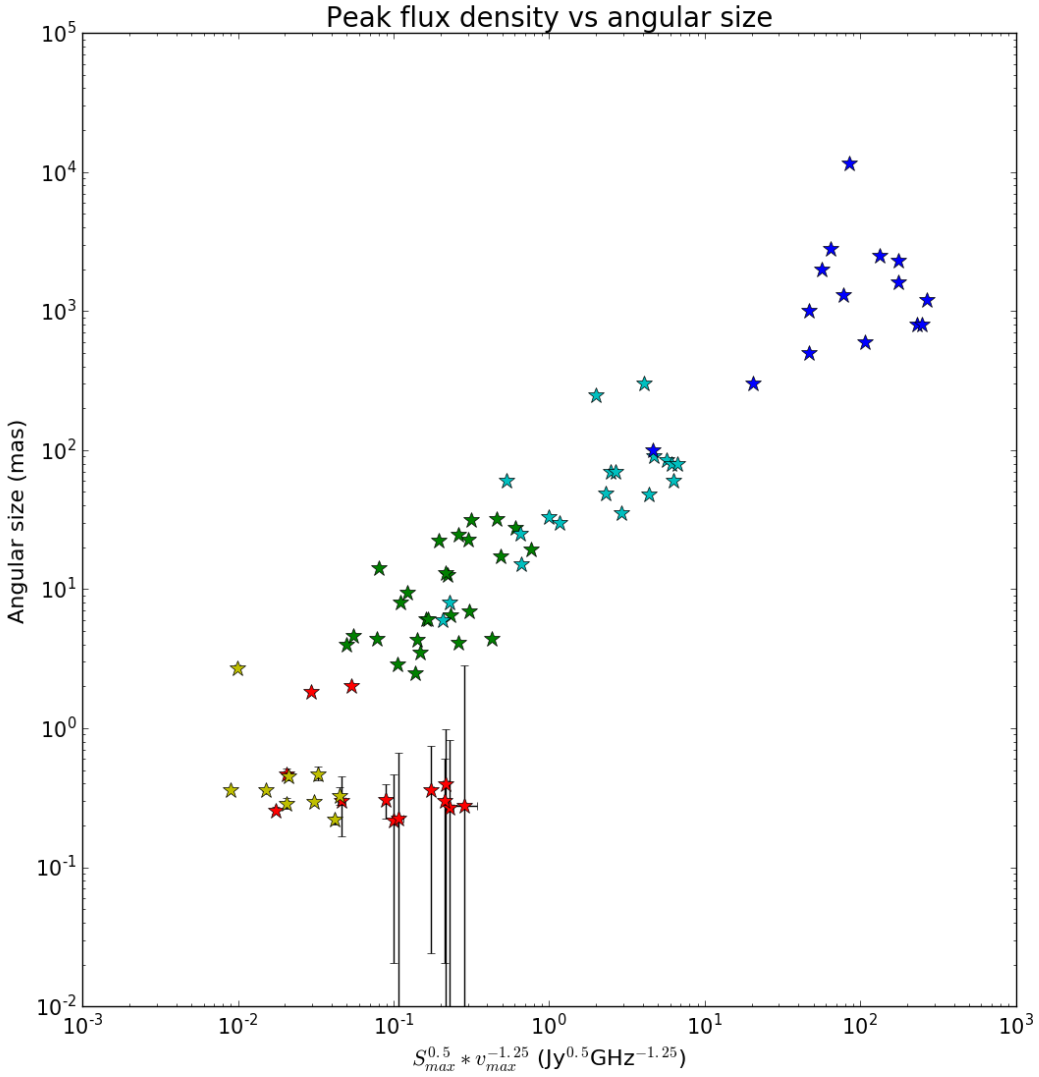


Figure 26: Angular size of source versus $S^{0.5}_{max}\nu^{-1.25}_{max}$. This plot is made within the observers frame. Cyan, blue and green stars are taken from [Snellen et al., 2000a], and designate Snellen's CSS, bright GPS and faint GPS samples respectively. The red stars represent the sources of McKean's very faint SSA GPS sample, while the yellow stars stand for the SSA GPS+power law sources found by this thesis.

$$\theta = f(k)^{5/4} \cdot \frac{B^{1/4} S^{1/2}_{max} (1+z)^{1/4}}{\nu^{5/4}_{max}}. \tag{39}$$

Wherein ν_{max} is the GPS peak frequency, $f(k)$ a function that depends weakly on the thick spectral index, but which is assumed as approximately 8 (the value it takes for $k \sim 2$), B the magnetic field strength, S_{max} the GPS peak flux density, θ the angular size and z the redshift. Furthermore, B is taken as 50 μG, based on Mingaliev et al. (2013) who used 100 μG for their sample of CRSs: however, as our sample contains fainter sources than those in Mingaliev's sample and thus sources with less synchrotron emission which indicates the presence of a weaker magnetic field, we take a smaller magnetic field strength - 50 μG was chosen in discussion with Dr. McKean as a sensible value. This led to the computation

S_{med} (mJy)	S_{bin} (mJy)	n	$\frac{dN}{dS} S_{med}^{5/2}$ ($\mathrm{Sr}^{-1}\mathrm{Jy}^{3/2}$)	$\frac{dN}{dS}$ ($\mathrm{Sr}^{-1}\mathrm{Jy}^{-1}$)
30	10.0	5	0.83±0.37	$(16.74 \pm 7.49) \cdot 10^{-5}$
60	20.0	15	7.00±1.81	$(25.11 \pm 6.48) \cdot 10^{-5}$
60	20.0	7 (8)	3.27±1.24	$(11.72 \pm 4.43) \cdot 10^{-5}$
120	40.0	2	2.64±1.87	$(1.67 \pm 1.18) \cdot 10^{-5}$

Table 9: The values of the bins for McKean's sample. S_{med} is the average bin flux density, S_{bin} the bin's radius, n the plain number of sources per bin (in brackets on the third row is provided the number of sources which are offset from the trend in Figure 26, which are all in the 60 mJy bin), $\frac{dN}{dS}$ the number of sources per sky area per bin and $\frac{dN}{dS} S_{med}^{5/2}$ that number density corrected by the average bin flux density. Higher bins did not contain any of the McKean sources and were thus omitted in this table. The second row is duplicated, once taking into account all sources, and secondly without those sources offset from the trend in Figure 26

S_{med} (mJy)	S_{bin} (mJy)	n	$\frac{dN}{dS} S_{med}^{5/2}$ ($\mathrm{Sr}^{-1}\mathrm{Jy}^{3/2}$)	$\frac{dN}{dS}$ ($\mathrm{Sr}^{-1}\mathrm{Jy}^{-1}$)
30	10.0	3	0.08±0.05	$(1.62 \pm 0.93) \cdot 10^{-5}$
60	20.0	7	0.53±0.20	$(1.89 \pm 0.71) \cdot 10^{-5}$
120	40.0	8	1.70±0.60	$(1.08 \pm 0.38) \cdot 10^{-5}$
240	80.0	12	7.23±2.08	$(8.09 \pm 2.34) \cdot 10^{-6}$
480	160.0	1	1.70±1.70	$(3.37 \pm 3.37) \cdot 10^{-7}$

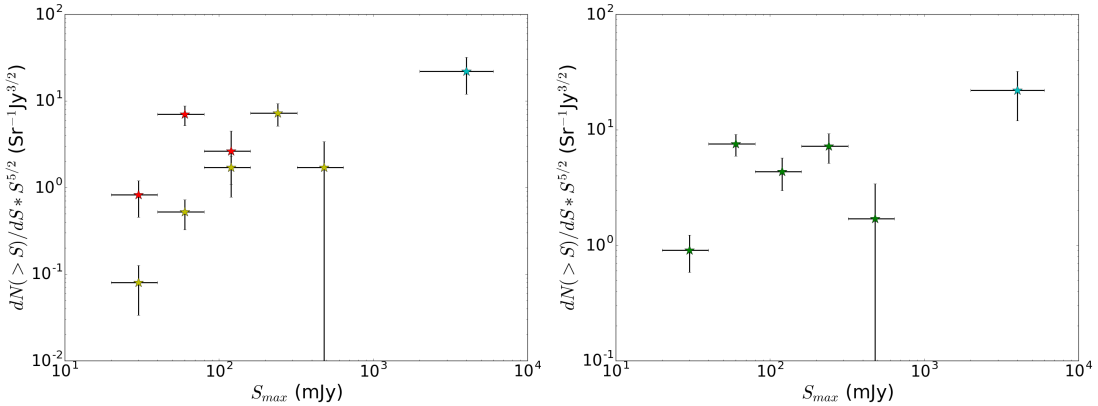
Table 10: The values of the bins for Snellen's sample. The columns represent the same quantities as in Table 9.

of angular size for our sub-sample, which turned out to be in the milli-arcseconds regime as one would expect for GPS sources. Further statistics (luminosity distance D, linear extension d and peak frequency corrected for redshift ν_{emit}) were computed using respectively the source code of a web-based Cosmology Calculator [Wright, 2006] and equation 31, which were all combined in Table 8. As was done in Snellen et al. (2000b), angular size was then plotted against a peak convolution given by $S_{max}^{0.5} \nu_{max}^{-1.25}$, results of which are shown in Figure 26 (see Appendix B.6).

The majority of the McKean sources line up well with Snellen's various samples, although a subset of the sources analysed in this thesis are offset from the trend. These coincidentally also possess large uncertainty bars in their angular size, while their uncertainty bars in the x-direction do not allow them to fall within the regime of the other, better fitting sources. There is a strong correspondence between offset sources and the spectral type: there are no SSA GPS+power law sources in the offset subset. There also seems to be a correspondence between the large uncertainties and poor spectral fits, although this is not exclusively the case. This can either indicate that indeed in the fitting procedure some parameter or function was poorly defined, or that these sources represent a new population of GPS that are not well understood. The 8 sources which both possess large uncertainties and are offset from the trend are J0104+3840, J0106+3522, J0106+4422, J0123+3641, J0159+4144, J1702+5511, J1702+5812 (although this source is a limit case, having uncertainties that are small compared to the other 7) and J1726+5826. These are exclusively SSA GPS sources with peak frequencies around 1 GHz, while the SSA GPS+power law sources line up consistently better with the trend provided by Snellen's sources. Further inspection shows that those SSA GPS sources that do line up with the trend all have peak frequencies in excess of ∼2 GHz. This leads us to conclude that the correspondence between peak frequency and angular size might be different than originally suspected in the low flux-density and small angular size regime. Another option is that the sources with low-frequency peaks suffer major variability: as the high- and low-frequency observations were not carried out simultaneously, it is well possible that if they were repeated in one sitting, the spectra would look markedly different, possibly shifting the peak to higher regimes.

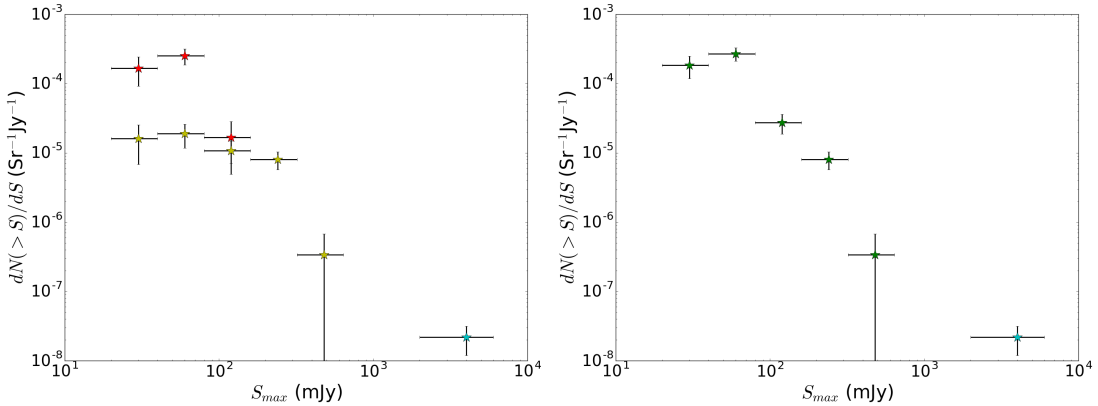
S_{med} (mJy)	S_{bin} (mJy)	n	$\frac{dN}{dS} S_{med}^{5/2}$ ($\mathrm{Sr}^{-1}\mathrm{Jy}^{3/2}$)	$\frac{dN}{dS}$ ($\mathrm{Sr}^{-1}\mathrm{Jy}^{-1}$)
30	10.0	8	0.91±0.32	$(18.36 \pm 6.49) \cdot 10^{-5}$
60	20.0	22	7.53±1.61	$(27.00 \pm 5.76) \cdot 10^{-5}$
120	40.0	10	4.34±1.37	$(2.75 \pm 0.87) \cdot 10^{-5}$
240	80.0	12	7.22±2.08	$(8.09 \pm 2.33) \cdot 10^{-6}$
480	160.0	1	1.70±1.70	$(3.37 \pm 3.37) \cdot 10^{-7}$

Table 11: The values of the bins for the combined sample. The columns represent the same quantities as in Table 9.



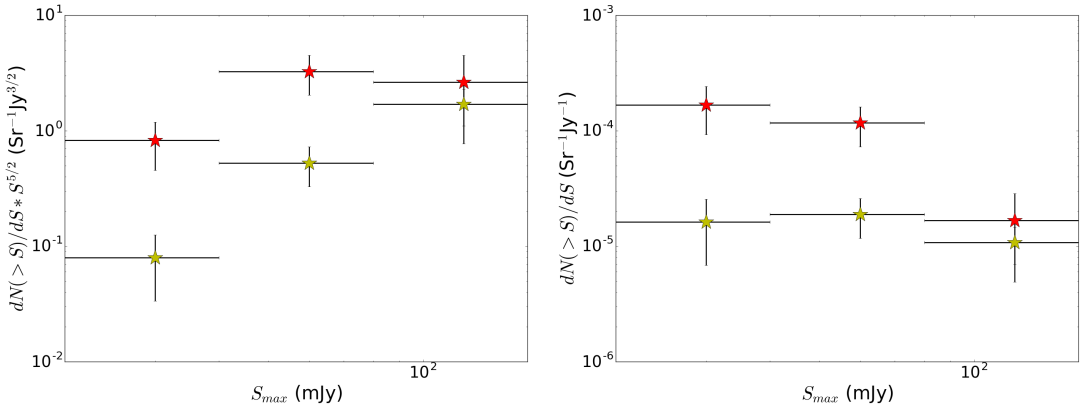
(a) Separated samples. (b) Combining both samples.

Figure 27: Differential number counts/peak convolution versus the peak flux. Red stars are the McKean sources, yellow stars are Snellen's, the cyan star was derived by Snellen from [de Vries et al., 1997]. The green stars represent the combined sample (bin width indicated by horizontal bar through stars).



(a) Separated samples. (b) Combining both samples.

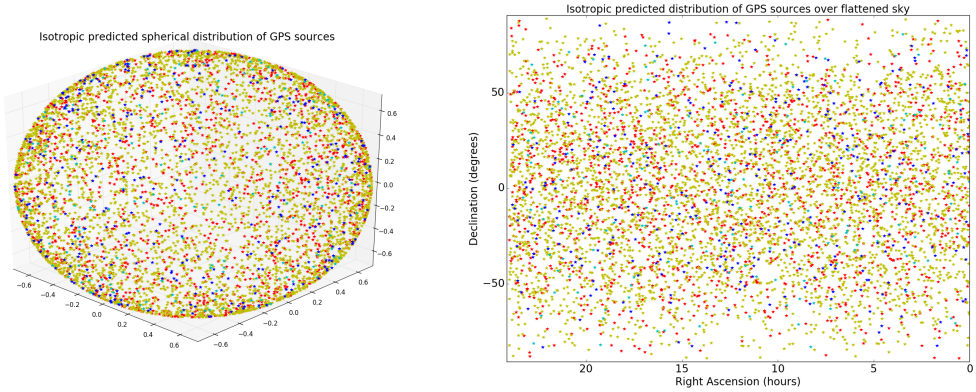
Figure 28: Plain differential number counts versus the peak flux. Red stars are the McKean sources, yellow stars are Snellen's, the cyan star was derived by Snellen from [de Vries et al., 1997]. The green stars represent the combined sample (bin width indicated by horizontal bar through stars).



(a) Discrepancy in number count/peak convolution versus peak flux.

(b) Discrepancy plain differential number count versus peak flux.

Figure 29: Closeup of the discrepancy between the McKean and Snellen samples, after removing those sources offset from the trend in Figure 26. Red stars are the McKean sources, yellow stars are Snellen's (bin width indicated by horizontal bar through stars).



(a) Populations projected onto an orb.

(b) Populations projected onto a plain.

Figure 30: Predicted population densities of GPS. The red population is 20-40 mJy, yellow 40-60 mJy, cyan 60-80 mJy and blue >80 mJy. Note that the locations are completely arbitrary, and these plots serve only to give an impression of the observed density of these sources.

5.1.2 Number density per peak flux density

Snellen, in his 1998 paper, creates a plot of differential number count of GPS sources, i.e. the number of sources within some bin of peak flux density multiplied by the average flux of the bin to the power 5/2 [Snellen et al., 1998b]. Using the data acquired by this thesis, an expanded version of this plot was created. Firstly, to obtain compatible density values, sky areas were computed for both the Snellen et al. (1998) and the McKean sample, using equation 24. This yielded a sky area for Snellen of 0.293 steradians, while the McKean field extended 0.047 steradians. Snellen's sources were subjected to the same criteria used to select the McKean sample (see chapter 3, section 1), namely a spectral index of no steeper than -0.5 between 1.4 and 4.85 GHz and a flux density of no smaller than 16.7 mJy at 8.46 GHz. These sources were then combined with McKean's and divided into in bins (which were maintained of constant size in $\log_{10}$-space) dependent on peak flux density to create Figures 27 and 28 (see Appendix B.7). The bin values, both in terms of number of sources, and the plotted values before and after multiplication by $S_{max}^{5/2}$, are presented in Tables 9, 10 and 11.

These Figures show a clear deviation from the trend expected from the Snellen and De Vries data alone. Even when one does not correct the Snellen data via the selection criteria set by McKean, resulting in more sources in the Snellen bins, his data still does not connect directly to the McKean sample and a discrepancy between Snellen's lowest and McKean's highest entry remains. This contradicts expectations, as what one would intuitively expect to see is the populations to i) link up directly to make a continuous curve and ii) the population at low flux density to shrink. This could hint at an unexplored, large population of low-flux density GPS, but it can also be due to, as previously mentioned, sources which are highly variable and which, due to non-simultaneous measurements of their high- and low-frequency regimes, show falsely peaked spectra and as such should therefore not even be included in these number density graphics. Further investigation (Figure 29) however shows that even after the removal of those possibly non-peaked variable sources masquerading as GPS, there is a considerable discrepancy of almost an order of magnitude at most between the McKean and Snellen samples, necessitating another explanation for the remaining difference.

One can use the number density algorithm developed for these plots to make a prediction of the amount of sources within some flux bin in the night sky. This is displayed in Figure 30. The positions of these sources are arbitrary, but the density impression that it gives is found by McKean's data, further giving us a predicted amount of sources satisfying McKean's criteria of approximately ∼13000. The 40-60 mJy population clearly dominates (see Appendix B.4).

5.2 Conclusions regarding the evolution of radio sources

Combining all results so far obtained, the following can be stated about this sample of very faint GPS sources: that they are small, both in angular (∼1 mas) and linear (several pc) terms, and that their emitted peaks are generally within the multi-GHz regime. We can also state that SSA GPS+power law sources in general have higher peak frequencies, which is sensible taking into account that the power law dominates the lower frequency regions of the spectrum, but there does not seem to be either a linear extension nor a redshift division between the two populations. In addition, we can conclude that during their evolution from very faint GPS (McKean's sample) to bright CSS (Snellen's sample), the relation between angular extension and $S_{max}^{0.5} \cdot \nu_{max}^{-1.25}$ remains approximately constant (see Figure 39), although an interesting subsection of pure SSA GPS sources seem offset from this trend. This implies that pure GPS sources that have just been energized increase in angular size quite explosively in the first stage of their evolution, by approximately an order of magnitude, before suffering changes in their peak flux density or peak frequency and joining the trend set out by Snellen's sample. It is also possible that during this early phase, $S_{max}^{0.5}$ and $\nu_{max}^{1.25}$ change inversely proportionally, i.e. the flux goes up as the frequency, as predicted by CRS evolution models, goes down, leading to an approximately constant quotient over this period. This is somewhat unexpected as SSA spectra ordinarily see both their flux *and* their frequency go down with time. However, the mechanism behind this change, if at all physical, is unknown at the time. It is also possible that the relation between angular size and peak flux and frequency in fact changes as the source ages, or that those offset sources are in fact non-peaked but highly variable power law sources masquerading as GPS. However, analysis done by making number counts per flux bin shows that even after accounting for this possible issue, there is still a large discrepancy between McKean's and Snellen's samples, raising questions concerning whether there might be a larger than expected population of low

flux density GPS in the night sky.

This roughly constant behaviour raises suspicions that these sources are indeed of the same nature, and that, as first hypothesised by O'Dea, Baum and Stanghellini, GPS do in fact over time evolve into CSS, although further research is required before a definitive conclusion can be reached, particularly into the transition class between CSS and FRI/II radio galaxies. Despite that, the findings so far presented provide a tantalizing first glimpse into the very early evolution of cosmic radio sources. In addition, the discovery of a dense population of low flux-density sources presents an unexpected upturn in the number count in that regime and will need to be investigated thoroughly in future research, as these objects may hold the clue to what happens in the earliest days of the life of an AGN. That there are so much more of these faint objects than brighter ones might imply that not all GPS sources go on to become bright radio galaxies, instead fading far earlier.

5.3 Next steps

Research is never complete, and this open scientific frontier forms no exception. Advances in this field can be enabled by several actions. Above all else, observations of both the very faint GPS - possibly even the HFP - and the CSS-to-FRI/II transition populations would greatly enhance our understanding of the evolution of cosmic radio sources. The actual triggering of CRSs still remains largely an enigma, as does the physical transition stage from CSS to a fully-fledged radio galaxy. Observations of these populations would enable us to study these critical evolutionary stages in unprecedented detail, and although the McKean sample forms a revolutionary, pioneering first step into the very faint CRS regime, it is hoped that it is not by far the last.

Additional improvements to our conclusions could be made with additional, more detailed observations of sources within the Snellen flux density observation window ($\sim$40 to $\sim$900 mJy). His observations stem from 1998, and although they are highly important and his paper ([Snellen et al., 1998b]) has been more than critical to the completion of this project, until a thorough re-examination of the moderate GPS population is performed, the option cannot be ruled out that the reason behind the discrepancy in Figures 27 and 28 is simply due to an incompleteness in the higher flux density samples.

The quality of the spectral fits could also be greatly improved by additional measurements in the observed band (0.325 - 22.46 GHz), especially on the low end of this range. The majority of fitting issues stemmed from the fact that either k (in SSA GPS sources) or m (in SSA GPS+power law sources) was poorly constrained, often by just two parameters. The upgraded VLA should be able to perform these observations, which would undoubtedly monumentally increase the quality of the fits, constraining the parameters more strongly than the six data points given ever could. It would also improve our fits if the measurements at 325 MHz and 1.4 GHz were redone simultaneously with the VLA measurements, as the current observations at the low and high ends of the spectrum lie years apart, something which presents an issue given the variable nature of some GPS sources. In the end, most uncertainties in the final results can be traced back to uncertainties within the spectral fits, so if they were to be more properly constrained and informed, so would all following physical results. Finally, spectroscopic analysis of those sources in the sample that lacked any redshift would greatly improve the completeness level of the sample, and would shed more light on the nature and shape of the redshift number counts, proving definitively the existence or non-existence of a bimodal behaviour in terms of redshift between the SSA GPS and SSA GPS+power law populations. Essentially, more data would be greatly appreciated.

5.4 Summary and concluding remarks

In short: a sample of 45 compact radio sources have been fitted with SSA spectra and/or power laws. The ratio between peaked and non-peaked spectra proved approximately 1:1, with most peaks in the multi-GHz regime. Redshifts of a subsample were determined using the K-z relation, while others were found to have spectroscopic redshifts, resulting in a completeness level of 66.6%. This data was combined to obtain the following conclusions:

1. there is no sign of bimodal behaviour in terms of redshift number counts between SSA GPS and SSA GPS+power law spectrum sources;

2. there are weak signs of bimodal behaviour in terms of peak flux and peak frequency number counts between SSA GPS and SSA GPS+power law spectrum sources;

3. the angular sizes of sources within the sample were of the scale $\sim$0.5 mas;

4. the linear sizes of sources within the sample were of the scale $\sim$3 pc;

5. the relation between angular size and $S_{max}^{0.5} \cdot \nu_{max}^{1.25}$ remains linear at low flux densities for SSA GPS+power law sources and falls off sharply at low flux densities for SSA GPS sources with respect to sources at larger flux densities, as obtained by Snellen, for which the origin or mechanism is unknown;

6. there is an unexpectedly dense population of low flux density sources present, compared to higher flux density, which may indicate that not all GPS sources become CSS or FRI/II radio galaxies.

In the end, suggestions are given on how to improve future research. These include measurements at both the extreme high and low frequency ends of the presently known CRS population, a redo of the sources within Snellen's observed flux density window, and additional observations at different frequencies of the sources in this sample to improve spectral fits.

Appendices

A GPS sample observations

These tables have been taken from [McKean, Browne & Jackson, in prep.]. They list the observations of the GPS sample used in this thesis. Note that in the second table, the source J0112+3759 is missing.

Table A1. The sample positional information with 1.4, 4.85 and 8.46 GHz radio flux-densities measured from NVSS, GB6 and CLASS, respectively.

Source Name	Right Ascension (° ′ ″)	Declination (° ′ ″)	$S_{0.325}$ (mJy)	$S_{1.4}$ (mJy)	$S_{4.85}$ (mJy)	$S_{8.46}$ (mJy)	$\alpha_{1.4}^{4.85}$
J0103+4322	01 03 28.8025	+43 22 59.526	45.0 ± 3.7	44.6 ± 1.6	41.0 ± 5.0	37.0 ± 1.9	−0.07 ± 0.10
J0104+3840	01 04 22.2100	+38 40 35.400	< 18.0	40.5 ± 2.7	27.0 ± 4.0		−0.33 ± 0.13
J0105+3533	01 05 57.2500	+35 33 29.400	66.0 ± 3.4	43.0 ± 1.3	28.0 ± 4.0		−0.35 ± 0.12
J0106+3522	01 06 14.4844	+35 22 26.786	< 18.0	40.5 ± 1.3	29.0 ± 5.0	18.5 ± 0.9	−0.27 ± 0.14
J0106+4422	01 06 21.3985	+44 22 27.377	33.0 ± 4.1	72.8 ± 2.2	47.0 ± 5.0	32.5 ± 1.6	−0.35 ± 0.09
J0112+3759	01 12 00.2598	+37 59 31.559	143.0 ± 3.7	61.9 ± 1.9	37.0 ± 5.0	21.8 ± 1.1	−0.41 ± 0.11
J0116+4420	01 16 56.8405	+44 20 59.871	169.0 ± 4.2	69.6 ± 2.5	41.0 ± 5.0	28.8 ± 1.5	−0.43 ± 0.10
J0119+4054	01 19 47.9105	+40 54 18.586	86.0 ± 4.0	75.2 ± 2.8	45.0 ± 5.0	48.4 ± 2.4	−0.41 ± 0.09
J0123+3843	01 23 23.8308	+38 43 57.900	< 18.0	18.5 ± 0.7	27.0 ± 4.0	36.6 ± 1.8	+0.30 ± 0.12
J0123+3806	01 23 28.8725	+38 06 36.758	46.0 ± 3.8	36.8 ± 1.5	40.0 ± 5.0	63.4 ± 3.2	+0.07 ± 0.11
J0123+3641	01 23 58.5880	+36 41 45.336	35.0 ± 3.6	54.0 ± 2.0	31.0 ± 4.0	28.9 ± 1.5	−0.45 ± 0.11
J0128+4003	01 28 13.6382	+40 03 29.866	116.0 ± 3.8	56.3 ± 1.7	47.0 ± 5.0	28.9 ± 1.5	−0.15 ± 0.09
J0128+4439	01 28 41.3396	+44 39 17.982	< 18.0	36.9 ± 1.2	35.0 ± 5.0	60.8 ± 3.0	−0.04 ± 0.12
J0129+4044	01 29 59.3005	+40 44 02.981	175.0 ± 3.6	67.9 ± 2.1	38.0 ± 5.0	24.8 ± 1.3	−0.47 ± 0.11
J0131+4428	01 31 03.3632	+44 28 01.883	65.0 ± 3.6	33.8 ± 1.4	27.0 ± 4.0	24.4 ± 1.5	−0.18 ± 0.12
J0132+4345	01 32 06.0642	+43 45 34.428	98.0 ± 3.7	55.0 ± 1.7	41.0 ± 5.0	30.1 ± 1.5	−0.24 ± 0.10
J0134+4018	01 34 40.8057	+40 18 18.619	104.0 ± 3.6	50.7 ± 1.6	29.0 ± 4.0	19.6 ± 1.0	−0.45 ± 0.11
J0135+3631	01 35 25.2841	+36 31 37.572	82.0 ± 4.1	48.9 ± 1.5	27.0 ± 4.0	16.9 ± 0.9	−0.48 ± 0.12
J0136+3905	01 36 32.5946	+39 05 59.198	84.0 ± 4.0	60.6 ± 1.9	49.0 ± 5.0	42.4 ± 2.1	−0.17 ± 0.09
J0136+3545	01 36 43.6844	+35 45 31.174	115.0 ± 4.0	53.7 ± 1.7	33.0 ± 4.0	22.4 ± 1.1	−0.39 ± 0.10
J0136+4401	01 36 47.3138	+44 01 10.216	97.0 ± 3.9	56.2 ± 1.7	36.0 ± 5.0	21.7 ± 1.1	−0.36 ± 0.11
J0143+3705	01 43 02.5385	+37 05 16.090	80.0 ± 3.7	47.5 ± 1.5	46.0 ± 5.0	40.5 ± 2.0	−0.03 ± 0.09
J0151+4332	01 51 18.3781	+43 32 00.547	42.0 ± 4.1	35.0 ± 1.1	34.0 ± 5.0	26.4 ± 1.3	−0.02 ± 0.12
J0151+4417	01 51 20.8793	+44 17 35.896	74.0 ± 4.3	57.9 ± 2.2	44.0 ± 5.0	62.7 ± 3.1	−0.22 ± 0.10
J0153+4146	01 53 42.2369	+41 46 45.167	250.0 ± 3.7	88.6 ± 3.2	49.0 ± 5.0	33.3 ± 1.7	−0.48 ± 0.09
J0154+3558	01 54 45.4614	+35 58 04.630	< 18.0	30.0 ± 1.3	34.0 ± 5.0	20.9 ± 1.1	+0.10 ± 0.12
J0154+4433	01 54 54.4682	+44 33 37.969	< 18.0	37.7 ± 1.2	40.0 ± 5.0	38.9 ± 2.0	+0.05 ± 0.10
J0156+4459	01 56 28.5219	+44 59 56.484	159.0 ± 4.3	83.5 ± 3.0	46.0 ± 5.0	28.6 ± 1.4	−0.48 ± 0.09
J0157+4120	01 57 05.0059	+41 20 30.634	72.0 ± 3.7	46.5 ± 1.4	30.0 ± 4.0	23.2 ± 1.2	−0.35 ± 0.11
J0159+4059	01 59 14.4871	+40 59 40.403	52.0 ± 3.9	35.2 ± 1.1	33.0 ± 5.0	27.0 ± 1.4	−0.05 ± 0.12
J0159+4144	01 59 49.3346	+41 44 31.946	< 18.0	51.6 ± 1.9	33.0 ± 5.0	21.9 ± 1.1	−0.36 ± 0.13
J1702+5511	17 02 34.5570	+55 11 12.432	< 18.0	38.5 ± 1.2	34.0 ± 4.0	29.5 ± 1.5	−0.10 ± 0.10
J1702+5812	17 02 41.3754	+58 13 10.082	20.0 ± 4.2	40.8 ± 1.3	38.0 ± 5.0	27.7 ± 1.4	−0.06 ± 0.11
J1711+5803	17 11 38.1200	+58 03 28.200	80.0 ± 3.7	44.1 ± 1.8	25.0 ± 4.0		−0.46 ± 0.13
J1715+5724	17 15 22.9752	+57 24 40.312	81.0 ± 3.8	57.6 ± 1.8	35.0 ± 4.0	26.0 ± 1.3	−0.40 ± 0.10
J1720+5541	17 20 09.7600	+55 41 27.300	31.0 ± 3.5	7.9 ± 0.5	27.0 ± 5.0		+0.99 ± 0.16
J1720+5926	17 21 00.6468	+59 26 49.442	137.0 ± 3.7	65.0 ± 2.4	41.0 ± 5.0	22.9 ± 1.2	−0.37 ± 0.10
J1726+5826	17 26 35.1256	+58 26 47.698	43.0 ± 3.8	48.7 ± 1.5	31.0 ± 4.0	38.4 ± 1.9	−0.36 ± 0.11
J1728+5532	17 28 11.6405	+55 32 30.471	21.0 ± 3.6	28.9 ± 1.0	42.0 ± 5.0	33.3 ± 1.7	+0.30 ± 0.10
J1735+5650	17 35 13.7695	+56 50 21.803	86.0 ± 4.1	49.2 ± 2.1	31.0 ± 4.0	33.2 ± 1.7	−0.37 ± 0.11
J1746+5659	17 46 59.5600	+56 59 11.000	< 18.0	12.6 ± 0.6	27.0 ± 4.0		+0.61 ± 0.13
J1747+5902	17 47 33.9322	+59 02 47.967	47.0 ± 4.3	31.9 ± 1.4	25.0 ± 4.0	21.7 ± 1.1	−0.20 ± 0.13
J1756+5918	17 56 11.8346	+59 18 56.824	26.0 ± 4.0	30.1 ± 1.3	25.0 ± 4.0	19.9 ± 1.0	−0.15 ± 0.13
J1756+5806	17 56 29.1449	+58 06 58.236	100.0 ± 3.8	51.6 ± 2.0	38.0 ± 5.0	24.3 ± 1.2	−0.25 ± 0.11
J1758+5951	17 58 12.1100	+59 51 56.100	< 18.0	31.4 ± 1.0	26.0 ± 4.0		−0.15 ± 0.13

Table A2. The flux densities and spectral indices from the VLA observations at 4.86, 8.46, 14.94 and 22.46 GHz. The 325 MHz and 1.4 GHz flux densities are taken from WENSS and NVSS, respectively. The letters after the source name correspond to the date of the VLA observations, where *a* is 2000 June 25, *b* is 2001 September 14 and *c* is 2001 December 31.

Source Name	$S_{4.86}$ (mJy)	$S_{8.46}$ (mJy)	$S_{14.94}$ (mJy)	$S_{22.46}$ (mJy)	$\alpha_{0.325}^{1.4}$	$\alpha_{1.4}^{4.86}$	$\alpha_{4.86}^{8.46}$	$\alpha_{8.46}^{14.94}$	$\alpha_{14.94}^{22.46}$
J0103+4322^b	41.3 ± 2.1	45.1 ± 2.3	48.1 ± 2.6	39.7 ± 3.2	-0.01 ± 0.07	-0.06 ± 0.06	$+0.16 \pm 0.13$	$+0.11 \pm 0.13$	-0.47 ± 0.24
J0104+3840^b	23.7 ± 1.2	17.8 ± 0.9	14.5 ± 1.1	15.4 ± 2.4	$> +0.56$	-0.43 ± 0.06	-0.52 ± 0.14	-0.36 ± 0.16	$+0.15 \pm 0.42$
J0105+3533^a	19.0 ± 1.0	12.8 ± 0.7	9.3 ± 0.7	8.5 ± 0.7	-0.29 ± 0.04	-0.66 ± 0.06	-0.71 ± 0.14	-0.56 ± 0.16	-0.22 ± 0.27
J0106+3522^a	22.7 ± 1.2	16.5 ± 0.9	12.3 ± 0.8	8.2 ± 0.7	$> +0.56$	-0.47 ± 0.06	-0.58 ± 0.14	-0.52 ± 0.14	-0.99 ± 0.25
J0106+4422^b	43.4 ± 2.2	26.9 ± 1.4	18.0 ± 1.2	19.1 ± 2.2	$+0.54 \pm 0.09$	-0.42 ± 0.06	-0.86 ± 0.13	-0.71 ± 0.15	$+0.15 \pm 0.32$
J0116+4420^b	38.4 ± 1.9	32.2 ± 1.6	21.6 ± 1.4	15.6 ± 2.0	-0.61 ± 0.04	-0.48 ± 0.06	-0.32 ± 0.13	-0.70 ± 0.14	-0.80 ± 0.35
J0119+4054^b	56.6 ± 2.9	54.4 ± 2.7	47.6 ± 2.5	43.8 ± 2.8	-0.09 ± 0.05	-0.23 ± 0.06	-0.07 ± 0.13	-0.23 ± 0.13	-0.20 ± 0.20
J0123+3843^a	35.2 ± 1.8	33.0 ± 1.7	21.4 ± 1.2	15.0 ± 0.9	$> +0.02$	$+0.52 \pm 0.06$	-0.12 ± 0.13	-0.76 ± 0.13	-0.87 ± 0.20
J0123+3806^b	86.1 ± 4.3	84.2 ± 4.2	64.0 ± 3.3	47.5 ± 2.9	-0.15 ± 0.07	$+0.68 \pm 0.06$	-0.04 ± 0.13	-0.48 ± 0.13	-0.73 ± 0.19
J0123+3641^a	42.4 ± 2.2	29.7 ± 1.5	19.6 ± 1.1	12.3 ± 0.8	$+0.30 \pm 0.08$	-0.19 ± 0.06	-0.64 ± 0.13	-0.73 ± 0.13	-1.14 ± 0.21
J0128+4003^b	40.0 ± 2.0	31.5 ± 1.6	20.5 ± 1.3	< 3.0	-0.49 ± 0.04	-0.27 ± 0.06	-0.43 ± 0.13	-0.76 ± 0.14	< -4.70
J0128+4439^b	38.7 ± 2.0	35.2 ± 1.8	31.0 ± 1.8	26.6 ± 2.4	$> +0.49$	$+0.04 \pm 0.06$	-0.17 ± 0.13	-0.22 ± 0.13	-0.37 ± 0.26
J0129+4044^b	35.8 ± 1.8	30.5 ± 1.6	32.9 ± 1.9	31.3 ± 2.6	-0.65 ± 0.04	-0.51 ± 0.06	-0.29 ± 0.13	$+0.12 \pm 0.13$	-0.12 ± 0.25
J0131+4428^b	30.0 ± 1.5	36.5 ± 1.9	38.4 ± 2.2	33.9 ± 2.7	-0.45 ± 0.05	-0.10 ± 0.06	$+0.35 \pm 0.14$	$+0.09 \pm 0.13$	-0.31 ± 0.24
J0132+4345^b	43.7 ± 2.2	37.9 ± 1.9	35.3 ± 2.0	33.1 ± 2.8	-0.40 ± 0.04	-0.18 ± 0.06	-0.26 ± 0.13	-0.13 ± 0.13	-0.16 ± 0.25
J0134+4018^b	30.5 ± 1.6	22.8 ± 1.2	18.0 ± 0.9	< 3.0	-0.49 ± 0.04	-0.41 ± 0.06	-0.52 ± 0.14	-0.42 ± 0.13	< -4.39
J0135+3631^b	23.3 ± 1.2	17.2 ± 0.9	13.7 ± 0.7	< 3.0	-0.35 ± 0.05	-0.60 ± 0.06	-0.55 ± 0.14	-0.40 ± 0.13	< -3.73
J0136+3905^a	52.0 ± 2.6	48.3 ± 2.4	46.8 ± 2.4	44.5 ± 2.3	-0.22 ± 0.05	-0.12 ± 0.06	-0.13 ± 0.13	-0.06 ± 0.13	-0.12 ± 0.18
J0136+3545^b	22.5 ± 1.2	16.8 ± 0.9	13.9 ± 1.4	< 3.0	-0.52 ± 0.04	-0.70 ± 0.06	-0.53 ± 0.14	-0.33 ± 0.20	< -3.76
J0136+4401^b	29.3 ± 1.5	18.7 ± 1.0	14.7 ± 1.2	< 3.0	-0.37 ± 0.04	-0.52 ± 0.06	-0.81 ± 0.14	-0.42 ± 0.17	< -3.90
J0143+3705^b	53.6 ± 2.7	46.2 ± 2.3	29.2 ± 1.9	20.5 ± 3.0	-0.36 ± 0.05	$+0.10 \pm 0.06$	-0.27 ± 0.13	-0.81 ± 0.14	-0.87 ± 0.39
J0151+4332^b	33.9 ± 1.7	34.4 ± 1.8	31.7 ± 1.8	28.5 ± 2.4	-0.12 ± 0.08	-0.03 ± 0.06	$+0.03 \pm 0.14$	-0.14 ± 0.13	-0.26 ± 0.25
J0151+4417^b	85.4 ± 4.3	72.3 ± 3.6	48.3 ± 2.6	34.9 ± 2.9	-0.17 ± 0.05	$+0.31 \pm 0.06$	-0.30 ± 0.13	-0.71 ± 0.13	-0.80 ± 0.24
J0153+4146^a	25.6 ± 1.3	32.5 ± 1.7	35.5 ± 1.9	34.8 ± 1.9	-0.71 ± 0.04	-1.00 ± 0.06	$+0.43 \pm 0.14$	$+0.16 \pm 0.13$	-0.05 ± 0.18
J0154+3558^b	30.3 ± 1.6	19.8 ± 1.0	12.8 ± 1.3	< 3.0	$> +0.35$	$+0.01 \pm 0.06$	-0.77 ± 0.14	-0.77 ± 0.20	< -3.56
J0154+4433^a	48.6 ± 2.4	46.2 ± 2.3	44.9 ± 2.3	37.4 ± 2.0	$> +0.51$	$+0.20 \pm 0.06$	-0.09 ± 0.13	-0.05 ± 0.13	-0.45 ± 0.18
J0156+4459^b	40.5 ± 2.1	27.2 ± 1.4	17.5 ± 1.2	16.3 ± 2.0	-0.44 ± 0.04	-0.58 ± 0.06	-0.72 ± 0.14	-0.78 ± 0.15	-0.17 ± 0.35
J0157+4120^a	29.8 ± 1.5	22.8 ± 1.1	19.3 ± 1.1	15.8 ± 1.0	-0.30 ± 0.05	-0.36 ± 0.06	-0.48 ± 0.14	-0.29 ± 0.13	-0.49 ± 0.21
J0159+4059^b	38.2 ± 1.9	30.1 ± 1.5	21.8 ± 1.4	15.8 ± 1.7	-0.27 ± 0.05	$+0.07 \pm 0.06$	-0.43 ± 0.14	-0.57 ± 0.14	-0.79 ± 0.30
J0159+4144^b	30.1 ± 1.5	23.0 ± 1.2	12.6 ± 1.2	< 3.0	$> +0.72$	-0.43 ± 0.06	-0.49 ± 0.14	-1.06 ± 0.18	< -3.52
J1702+5511^c	25.3 ± 1.4	22.5 ± 1.1	17.6 ± 1.0	15.1 ± 0.8	$> +0.52$	-0.34 ± 0.06	-0.21 ± 0.14	-0.43 ± 0.13	-0.38 ± 0.19
J1702+5812^c	34.2 ± 1.7	28.0 ± 1.4	20.8 ± 1.1	16.2 ± 0.9	$+0.49 \pm 0.15$	-0.14 ± 0.06	-0.36 ± 0.13	-0.52 ± 0.13	-0.61 ± 0.19
J1711+5803^a	21.2 ± 1.1	12.4 ± 0.7	6.4 ± 0.6	< 3.0	-0.41 ± 0.05	-0.59 ± 0.06	-0.97 ± 0.15	-1.16 ± 0.20	< -1.86
J1715+5724^c	28.6 ± 1.5	20.4 ± 1.0	14.6 ± 0.8	11.7 ± 0.7	-0.23 ± 0.05	-0.56 ± 0.06	-0.61 ± 0.13	-0.59 ± 0.13	-0.54 ± 0.20
J1720+5541^a	25.8 ± 1.3	22.4 ± 1.2	13.6 ± 0.9	10.8 ± 0.8	-0.94 ± 0.08	$+0.95 \pm 0.06$	-0.25 ± 0.14	-0.88 ± 0.15	-0.57 ± 0.25
J1720+5926^c	32.7 ± 1.7	21.8 ± 1.1	17.0 ± 1.0	13.6 ± 0.8	-0.51 ± 0.04	-0.55 ± 0.06	-0.73 ± 0.13	-0.44 ± 0.13	-0.55 ± 0.19
J1726+5826^a	43.6 ± 2.2	38.9 ± 2.0	35.9 ± 1.9	33.4 ± 1.8	$+0.09 \pm 0.07$	-0.09 ± 0.06	-0.21 ± 0.13	-0.14 ± 0.13	-0.18 ± 0.18
J1728+5532^c	42.7 ± 2.2	40.7 ± 2.0	38.1 ± 2.0	33.5 ± 1.7	$+0.22 \pm 0.12$	$+0.31 \pm 0.06$	-0.09 ± 0.13	-0.12 ± 0.13	-0.32 ± 0.18
J1735+5650^c	39.6 ± 2.0	40.3 ± 2.0	39.4 ± 2.0	35.6 ± 1.8	-0.38 ± 0.05	-0.17 ± 0.06	$+0.03 \pm 0.13$	-0.04 ± 0.13	-0.25 ± 0.18
J1746+5659^c	28.6 ± 1.5	34.5 ± 1.7	23.0 ± 1.2	14.7 ± 0.8	> -0.24	$+0.66 \pm 0.06$	$+0.34 \pm 0.13$	-0.71 ± 0.13	-1.10 ± 0.19
J1747+5902^a	33.3 ± 1.7	36.3 ± 1.8	45.2 ± 2.3	43.2 ± 2.2	-0.27 ± 0.07	$+0.03 \pm 0.06$	$+0.16 \pm 0.14$	$+0.39 \pm 0.13$	-0.11 ± 0.18
J1756+5918^c	25.3 ± 1.3	23.7 ± 1.2	24.2 ± 1.3	24.0 ± 1.2	$+0.10 \pm 0.11$	-0.14 ± 0.06	-0.12 ± 0.14	$+0.04 \pm 0.13$	-0.02 ± 0.18
J1756+5806^c	31.3 ± 1.6	24.4 ± 1.2	18.3 ± 1.0	15.7 ± 0.9	-0.45 ± 0.04	-0.40 ± 0.06	-0.45 ± 0.13	-0.51 ± 0.13	-0.38 ± 0.19
J1758+5951^a	53.5 ± 2.7	65.8 ± 3.3	36.3 ± 1.9	26.5 ± 1.5	$> +0.38$	$+0.43 \pm 0.06$	$+0.37 \pm 0.13$	-1.05 ± 0.13	-0.77 ± 0.19

B Source code

Relevant source code is listed here.

B.1 Least Squares fitting algorithm

The following script is used to fit all spectral types using an LSq algorithm.

Scripts/Thesis_Sv–omnifit–LSq_v2–1.py

```
#!/usr/bin/env python
from __future__ import division
import numpy as np
import scipy as sp
import scipy.interpolate as spi
import scipy.optimize as spo
import scipy.stats as sps
from matplotlib.pyplot import figure, show, rc, rcParams
###THIS SCRIPT CAN RUN ALL THREE FITS, BUT UNCERTAINTIES HAVE BEEN REPRESSED BY SOMEWHAT ARGUABLE MEANS. ONLY THE SYNCH PLUS POWER LAW SPECTRUM'S UNCERTAINTIES STAND IN MY PATH TOWARDS POWER, INFINITE POWER!
###This script utilizes a least squares algorithm to obtain the best fit.

#####Episode I: Reading Data (or: The Phantom Data).
    ##########################################################################
print(">>> Episode I: The Phantom Data")

###Reading relevant data from (mono-axial) datafile. If uncertainties were unknown, they have been set to 1.0 to prevent code collapse due to divide by zero errors.
fulllist = np.genfromtxt("Janskylist.dat",dtype=None,usecols=[0],comments="#")
errlist = np.genfromtxt("Janskylist.dat",dtype=None,usecols=[1],comments="#")

###Splitting the superarray into useful, seperate arrays with relevant data.
names,S003,S014,S048,S084,S149,S224 = np.split(fulllist,7)
junkerr,errS003,errS014,errS048,errS084,errS149,errS224 = np.split(errlist,7)
names = names.astype(str).flatten()
S003 = np.asarray(S003.astype(float).flatten())
S014 = np.asarray(S014.astype(float).flatten())
S048 = np.asarray(S048.astype(float).flatten())
S084 = np.asarray(S084.astype(float).flatten())
S149 = np.asarray(S149.astype(float).flatten())
S224 = np.asarray(S224.astype(float).flatten())

###Printing all data in a clear-cut table.
print("\033[1m","index * name * 0.325 * 1.4 * 4.85 * 8.46 * 14.94 * 22.46","\033[0m","<-- Frequency bands")
for i in range(len(names)):
    print(i,":","\033[1m",names[i],"\033[0m","*",S003[i],"*",S014[i],"*",S048[i],"*",S084[i],"*",S149[i],"*",S224[i])
n = int(input(" Pick GPS source: "))

###Creating the to be used data arrays.
v = [0.325,1.4,4.85,8.46,14.94,22.46]
S = [S003[n],S014[n],S048[n],S084[n],S149[n],S224[n]]
Serr = [errS003[n],errS014[n],errS048[n],errS084[n],errS149[n],errS224[n]]

print("Source designation:",names[n])
print("Frequency bands:",np.asarray(v))
print("Flux densities:",np.asarray(S))
print("Flux uncertainties:",np.asarray(Serr))

###Plotting the recorded data points for visualisation purposes.
fig = figure(figsize=(10,10))
frame = fig.add_subplot(1,1,1,xscale="log",yscale="log")
frame.errorbar(v,S,yerr=Serr,ecolor="k",marker="*",mfc="y",ms=10,ls="",label="Data from {}".format(names[n]))
frame.set_xlabel('Frequency v (GHz)')
frame.set_ylabel('Flux density S (mJy)')
frame.legend()
```

```
53 show()
54
55 ###While-loop to allow user-friendly cancelling of code.
56 while True:
57     print("\033[1m","Do you want to continue?","\033[0m")
58     answer = input("  :")
59     if answer.lower().startswith("y"):
60         print("Roger")
61         break
62     elif answer.lower().startswith("n"):
63         print("OKDOEI")
64         exit()
65
66 ###While-loop to allow user-friendly usage of power law or not, with q the model index,
       ndim the amount of free parameters, and os the offset within the naming list for
       easier printing of results. For q = 3, the spectrum shows signs of ageing - a final
       point that is offset from the power law-like behaviour of its predecessors. Fitting
       an ageing component is extremely difficult, so we instead ignore the final point and
        plot a power law through the others as spectral ageing is not of interest to our
       research.
67 while True:
68     print("\033[1m","What kind of fit would you like to use?","\033[0m")
69     print("0 = power law, 1 = synchrotron spectrum, 2 = synchrotron + power law, 3 =
       ageing power law spectrum, 4 = synchrotron + underdefined power law (i.e. only
       through first two points), 5 = double power law.")
70     answer = input("  :")
71     if answer.lower().startswith("0"):
72         q = 0 ; ndim = 2 ; os=4
73         print("Roger: power law online")
74         break
75     elif answer.lower().startswith("1"):
76         q = 1 ; ndim = 4 ; os=0
77         print("Roger: synch fit online")
78         break
79     elif answer.lower().startswith("2"):
80         q = 2 ; ndim = 6 ; os=0
81         print("Roger: synch fit and power law online")
82         break
83     elif answer.lower().startswith("3"):
84         q = 3 ; ndim = 2 ; os=4
85         print("Roger: ageing fit online. Final datapoint will be ignored")
86         break
87     elif answer.lower().startswith("4"):
88         q = 4 ; ndim = 5 ; os=0
89         print("Roger: synch fit and undef power law online. A fudgy powerlaw will be
       drawn.")
90         break
91     elif answer.lower().startswith("5"):
92         q = 5 ; ndim = 4 ; os=4
93         print("Roger: double power law online.")
94         break
95 modelname = ["power law","synchrotron","synchrotron/power law","ageing spectrum","
       synchrotron/undef power law","double power law"]
96 print(" Number of free parameters:",ndim)
97
98
99 #####Episode II: Creating fitting functions and derivatives (or: Attack of the Functions
       ). ##########################
100 print(">>> Episode II: Attack of the Functions")
101
102 ###Defining the fits.
103 def Sfitpl(v,A,m):
104     f1 = A*v**m                                    ###First power law
105     return f1
106
107 def Sfitplpl(v,A1,m1,A2,m2):
108     f1 = A1*v**m1                                  ###First power law
109     f2 = A2*v**m2                                  ###First power law
110     return f1+f2
111
```

```
112 def Sfitffa(v,A,m,t):
113     f1 = A*v**m                                  ###First power law
114     f2 = np.e**(-t*v**-2.1)                      ###Exponential fall-off
115     return f1*f2
116
117 def Sfitsyn(v,S01,v01,k,l):
118     f1 = S01*(v/v01)**k                          ###First power law
119     f2 = (1-np.e**(-(v/v01)**(l-k)))/(1-np.e**(-1))   ###Second power law (exponential
        fall-off)
120     return f1*f2
121
122 def Sfitplsyn(v,S01,v01,k,l,A,m):
123     f1 = S01*(v/v01)**k                          ###First power law
124     f2 = (1-np.e**(-(v/v01)**(l-k)))/(1-np.e**(-1))   ###Second power law (exponential
        fall-off)
125     f3 = A*v**m                                  ###Third power law (only when two
        turnovers are present) Wherein A=S02/(v02^m)
126     return f1*f2 + f3
127
128 def derSfitsyn(v,S01,v01,k,l):                   ###Derivative of synchrotron fit
129     E = np.e**(-(v/v01)**(l-k))
130     C = S01/(v01*(1-1/np.e))
131     f1 = C*k*((v/v01)**(k-1))*(1-E)
132     f2 = C*(l-k)*((v/v01)**(l-1))*E
133     return f1 + f2
134
135 def derSfitplsyn(v,S01,v01,k,l,A,m):             ###Derivative of power law/
        synchrotron combined fit
136     E = np.e**(-(v/v01)**(l-k))
137     C = S01/(v01*(1-1/np.e))
138     f1 = C*k*((v/v01)**(k-1))*(1-E)
139     f2 = C*(l-k)*((v/v01)**(l-1))*E
140     f3 = A*m*v**(m-1)
141     return f1 + f2 + f3
142
143 ###Array spanning the fitting space.
144 vfit = np.linspace(0.3,23,1000)
145
146 ###List that names the coefficients for the synchfit.
147 coeflist = ["S_{01}","v_{01}","k","l","A1","m1","A2","m2"]
148 ###With S_{01} the scale flux of the synchrotron model, v_{01} the scale frequency of
        the synchrotron model, k the optically thick spectral index (before the peak), l the
         optically thin spectral index (after the peak), A = S_{02}/(v_{02}^m) the fudged
        constant for the power law, combining the power law scale flux and scale frequency (
        S_{02} and v_{02}, respectively) with the power law spectral index k=l --> m (
        otherwise known as Jesper's Fudge Factor), and m the power law spectral index for a
        spectrum wherein k=l.
149
150
151 #####Episode III: Computing uncertainties in fits. (Revenge of the Uncertainties).
        ###################################
152 print(">>> Episode III: Revenge of the Uncertainties")
153
154 def errSfitpl(v,A,m,cov):                    ###Finding uncertainties function in power law
        .
155     dsdA = v**m
156     dsdm = A*(v**m)*np.log(v)
157     dsdx = np.asarray([dsdA,dsdm])
158     Svar = 0
159     for i in range(ndim):
160         for j in range(ndim):
161             Svar = Svar + dsdx[i]*dsdx[j]*cov[i,j]
162     return np.sqrt(Svar)
163
164 def errSfitplpl(v,A1,m1,A2,m2,cov):              ###Finding uncertainties function in
        power law.
165     dsdA1 = v**m1
166     dsdm1 = A1*(v**m1)*np.log(v)
167     dsdA2 = v**m2
168     dsdm2 = A2*(v**m2)*np.log(v)
```

```
    dsdx = np.asarray([dsdA1,dsdm1,dsdA2,dsdm2])
    Svar = 0
    for i in range(ndim):
        for j in range(ndim):
            Svar = Svar + dsdx[i]*dsdx[j]*cov[i,j]
    return np.sqrt(Svar)

def errSfitffa(v,A,m,t,cov):              ###Finding uncertainties function in power law.
    dsdA = (v**m)*np.e**(-t*v**-2.1)
    dsdm = A*(v**m)*np.log(v)*np.e**(-t*v**-2.1)
    dsdt = -A*(v**(m-2.1))*np.e**(-t*v**-2.1)
    dsdx = np.asarray([dsdA,dsdm,dsdt])
    Svar = 0
    for i in range(ndim):
        for j in range(ndim):
            Svar = Svar + dsdx[i]*dsdx[j]*cov[i,j]
    return np.sqrt(Svar)

def errSfitsyn(v,S01,v01,k,l,cov):        ###Finding uncertainties function in synch fit.
    C = S01/(np.e-1)
    C2 = S01/(1-1/np.e)
    E = np.e**-((v/v01)**(l-k))
    E2 = np.e**(1-((v/v01)**(l-k)))
    dsdS01 = Sfitsyn(v,S01,v01,k,l)/S01
    dsdv01 = (-((C2*(1-E)))*k*v*((v/v01)**(k-1)))/(v01**2) - (C2*(l-k)*v*E*((v/v01)**(l-1)))/(v01**2)
    dsdk = C2*(1-E)*((v/v01)**k)*np.log(v/v01) - C2*((v/v01)**(l))*np.log(v/v01)*E
    dsdl = C*((v/v01)**l)*np.log(v/v01)*E2
    dsdx = np.asarray([dsdS01,dsdv01,dsdk,dsdl])
    Svar = 0
    for i in range(ndim):
        for j in range(ndim):
            Svar = Svar + dsdx[i]*dsdx[j]*cov[i,j]
    return np.sqrt(Svar)

def errSfitplsyn(v,S01,v01,k,l,A,m,cov):  ###Finding uncertainties function in synch fit plus power law.
    C = S01/(np.e-1)
    C2 = S01/(1-1/np.e)
    E = np.e**-((v/v01)**(l-k))
    E2 = np.e**(1-((v/v01)**(l-k)))
    dsdS01 = Sfitsyn(v,S01,v01,k,l)/S01
    dsdv01 = (-((C2*(1-E)))*k*v*((v/v01)**(k-1)))/(v01**2) - (C2*(l-k)*v*E*((v/v01)**(l-1)))/(v01**2)
    dsdk = C2*(1-E)*((v/v01)**k)*np.log(v/v01) - C2*((v/v01)**(l))*np.log(v/v01)*E
    dsdl = C*((v/v01)**l)*np.log(v/v01)*E2
    dsdA = v**m
    dsdm = A*(v**m)*np.log(v)
    dsdx = np.asarray([dsdS01,dsdv01,dsdk,dsdl,dsdA,dsdm])
    Svar = 0
    for i in range(ndim):
        for j in range(ndim):
            Svar = Svar + dsdx[i]*dsdx[j]*cov[i,j]
    return np.sqrt(Svar)


#####Episode IV: Least squares fitting the data (or: A New Fit).
    ######################################################
print(">>> Episode IV: A New Fit")

###Creating some new, useful arrays and indices.
S2 = [S[1],S[2],S[3],S[4],S[5]]
Serr2 = [Serr[1],Serr[2],Serr[3],Serr[4],Serr[5]]
v2 = [v[1],v[2],v[3],v[4],v[5]]
S3 = [S[0],S[1],S[2],S[3],S[4]]
Serr3 = [Serr[0],Serr[1],Serr[2],Serr[3],Serr[4]]
v3 = [v[0],v[1],v[2],v[3],v[4]]
Smi = S.index(max(S))
```

```
Smi2 = S2.index(max(S2))
Sl = [S[0],S[1],S[2]]
vl = [v[0],v[1],v[2]]
Smil = Sl.index(max(Sl))
Sh = [S[3],S[4],S[5]]
vh = [v[3],v[4],v[5]]
Smih = Sh.index(max(Sh))

###Monstrous, hulking, hideous if-loop that creates the fit by first estimating and then least squaring parameters.
if q == 0:
    S0guess = max(S)
    v0guess = v[Smi]
    mguess = (np.log10(S[5])-np.log10(S[0]))/(np.log10(v[5])-np.log10(v[0]))
    Aguess = S0guess/(v0guess**mguess)
    estimates = [Aguess,mguess]
    p, pcov = spo.curve_fit(Sfitpl,v,S,[Aguess,mguess],sigma=Serr,bounds=([0,-10],[500,10]),**{'max_nfev':1000000})
    fitfun = Sfitpl(vfit,p[0],p[1])
    fitmax = fitfun + errSfitpl(vfit,p[0],p[1],pcov)
    fitmin = fitfun - errSfitpl(vfit,p[0],p[1],pcov)
elif q == 1:
    S0guess = max(S)
    v0guess = v[Smi]
    while True:
        print("\033[1m","Do you want to constrain k narrowly to 2.3-2.5?","\033[0m")
        answer = input("  :")
        if answer.lower().startswith("y"):
            klow = 2.3
            kguess = 2.4
            print("Roger: k constrained to 2.3-2.5")
            break
        if answer.lower().startswith("n"):
            klow = 0.0
            if S[1] > S[0]:
                kguess = (np.log10(S[1])-np.log10(S[0]))/(np.log10(v[1])-np.log10(v[0]))
            else:
                if S[2] > S[0]:
                    kguess = (np.log10(S[2])-np.log10(S[0]))/(np.log10(v[2])-np.log10(v[0]))
                else:
                    kguess = 2.0
            print("Roger: k not narrowly constrained.")
            break
    if S[5] < S[4]:
        lguess = (np.log10(S[5])-np.log10(S[4]))/(np.log10(v[5])-np.log10(v[4]))
    else:
        if S[5] < S[3]:
            lguess = (np.log10(S[5])-np.log10(S[3]))/(np.log10(v[5])-np.log10(v[3]))
        else:
            lguess = -1.0
    estimates = [S0guess,v0guess,kguess,lguess]
    p, pcov = spo.curve_fit(Sfitsyn,v,S,[S0guess,v0guess,kguess,lguess],sigma=Serr,bounds=([0,0.3,klow,-5.0],[500,23,2.5,0]),**{'max_nfev':1000000})
    fitfun = Sfitsyn(vfit,p[0],p[1],p[2],p[3])
    fitmax = fitfun + errSfitsyn(vfit,p[0],p[1],p[2],p[3],pcov)
    fitmin = fitfun - errSfitsyn(vfit,p[0],p[1],p[2],p[3],pcov)
elif q == 2:
    S0guess = max(S2)
    v0guess = v2[Smi2]
    kguess = (np.log10(S2[Smi2])-np.log10(S2[Smi2-1]))/(np.log10(v2[Smi2])-np.log10(v2[Smi2-1]))
    if S[5] < S[4]:
        lguess = (np.log10(S[5])-np.log10(S[4]))/(np.log10(v[5])-np.log10(v[4]))
    else:
        if S[5] < S[3]:
            lguess = (np.log10(S[5])-np.log10(S[3]))/(np.log10(v[5])-np.log10(v[3]))
        else:
            lguess = -1.0
    Aguess = max(S)
```

```
    if S[1] < S[0]:
        mguess = (np.log10(S[1])-np.log10(S[0]))/(np.log10(v[1])-np.log10(v[0]))
    else:
        if S[2] < S[0]:
            mguess = (np.log10(S[2])-np.log10(S[0]))/(np.log10(v[2])-np.log10(v[0]))
        else:
            mguess = -1.0
    estimates = [S0guess,v0guess,kguess,lguess,Aguess,mguess]
    p, pcov = spo.curve_fit(Sfitplsyn,v,S,[S0guess,v0guess,kguess,lguess,Aguess,mguess],
    sigma=Serr,bounds=([0,0.3,0,-5.0,0,-10],[500,23,2.5,0,500,10]),**{'max_nfev'
    :1000000})
    fitfun = Sfitplsyn(vfit,p[0],p[1],p[2],p[3],p[4],p[5])
    fitmax = fitfun + errSfitplsyn(vfit,p[0],p[1],p[2],p[3],p[4],p[5],pcov)
    fitmin = fitfun - errSfitplsyn(vfit,p[0],p[1],p[2],p[3],p[4],p[5],pcov)
elif q == 3:
    S0guess = max(S)
    v0guess = v[Smi]
    mguess = (np.log10(S[4])-np.log10(S[0]))/(np.log10(v[4])-np.log10(v[0]))
    Aguess = S0guess/(v0guess**mguess)
    estimates = [Aguess,mguess]
    p, pcov = spo.curve_fit(Sfitpl,v3,S3,[Aguess,mguess],sigma=Serr3,bounds
    =([0,-10],[500,0]),**{'max_nfev':1000000})
    fitfun = Sfitpl(vfit,p[0],p[1])
    fitmax = fitfun + errSfitpl(vfit,p[0],p[1],pcov)
    fitmin = fitfun - errSfitpl(vfit,p[0],p[1],pcov)
elif q == 4:
    S0guess = max(S2)
    v0guess = v[Smi2]
    if S2[1] > S2[0]:
        kguess = (np.log10(S2[1])-np.log10(S2[0]))/(np.log10(v2[1])-np.log10(v2[0]))
    else:
        if S2[2] > S2[0]:
            kguess = (np.log10(S2[2])-np.log10(S2[0]))/(np.log10(v2[2])-np.log10(v2[0]))
        else:
            kguess = 2.0
    if S2[4] < S2[3]:
        lguess = (np.log10(S2[4])-np.log10(S2[3]))/(np.log10(v2[4])-np.log10(v2[3]))
    else:
        if S2[4] < S2[2]:
            lguess = (np.log10(S2[4])-np.log10(S2[2]))/(np.log10(v2[4])-np.log10(v2[2]))
        else:
            lguess = -1.0
    M = (np.log10(0.5*S[1])-np.log10(S[0]))/(np.log10(v[1])-np.log10(v[0]))
    Aguess = S[0]/(v[0]**M)
    estimates = [S0guess,v0guess,kguess,lguess,Aguess,M]
    def Sfitplsyn2(v,S01,v01,k,l,A):
        f1 = S01*(v/v01)**k                                  ###First power law
        f2 = (1-np.e**(-(v/v01)**(l-k)))/(1-np.e**(-1))          ###Second power law (
    exponential fall-off)
        f3 = A*v**M                                          ###Third power law (only when two
    turnovers are present) Wherein A=S02/(v02^m)
        return f1*f2 + f3
    p, pcov = spo.curve_fit(Sfitplsyn2,v,S,[S0guess,v0guess,kguess,lguess,Aguess],sigma=
    Serr,bounds=([0,0.3,0,-5.0,0],[500,50,2.5,0,500]),**{'max_nfev':1000000})
    fitfun = Sfitplsyn(vfit,p[0],p[1],p[2],p[3],p[4],M)
    fitmax = fitfun + errSfitplsyn(vfit,p[0],p[1],p[2],p[3],p[4],M,pcov)
    fitmin = fitfun - errSfitplsyn(vfit,p[0],p[1],p[2],p[3],p[4],M,pcov)
if q == 5:
    S01guess = max(Sl)
    v01guess = vl[Smil]
    S02guess = max(Sh)
    v02guess = vh[Smih]
    m1guess = (np.log10(0.5*S[2])-np.log10(S[0]))/(np.log10(v[1])-np.log10(v[0]))
    A1guess = S01guess/(v01guess**m1guess)
    m2guess = (np.log10(S[5])-np.log10(0.5*S[3]))/(np.log10(v[5])-np.log10(v[3]))
    A2guess = S02guess/(v02guess**m2guess)
    estimates = [A1guess,m1guess,A2guess,m2guess]
    p, pcov = spo.curve_fit(Sfitplpl,v,S,[A1guess,m1guess,A2guess,m2guess],sigma=Serr,
    bounds=([0,-10,0,-10],[500,10,500,10]),**{'max_nfev':1000000})
    fitfun = Sfitplpl(vfit,p[0],p[1],p[2],p[3])
```

```
    fitmax = fitfun + errSfitplpl(vfit,p[0],p[1],p[2],p[3],pcov)
    fitmin = fitfun - errSfitplpl(vfit,p[0],p[1],p[2],p[3],pcov)


###Printing loop outputs, i.e. values of free parameters by least squares fitting.
print(" Parameter estimates are:",estimates)
print("\033[1m"+"    >>> Using scipy.optimize.curve_fit on {} model:".format(modelname[q
    ]),"\033[0m")
for i in range(ndim):
    print(" Coefficient",coeflist[i+os],"equals",p[i])
    print("  Uncertainty of coefficient",coeflist[i+os],"equals",np.sqrt(pcov[i,i]))
if q == 4:
    print(" Coefficient m (fixed) equals",M)


#####Episode V: Finding the maximum of the fit (or: The Maximum Strikes Back).
    ########################################
print(">>> Episode V: The Maximum Strikes Back")

###If loop that bisects the function if it has one or more peaks.
if q == 0 or q == 3 or q == 5:
    vmax = 0
    Smax = 0
elif q == 1 or q == 2 or q == 4:
    if q == 2 or q == 4:
        if S[2] > S[1]:
            inc = (v[2]+v[1])/2
        else:
            if S[3] > S[2]:
                inc = (v[3]+v[2])/2
            else:
                inc = v[3]
        def bisector(v):                        ###Function of one variable used by the
    bisection algorithm.
            return derSfitplsyn(v,p[0],p[1],p[2],p[3],p[4],M)
    elif q == 1:
        inc = v[0]
        def bisector(v):                        ###Function of one variable used by the
    bisection algorithm.
            return derSfitsyn(v,p[0],p[1],p[2],p[3])
    vfitbis = np.linspace(inc,23,1000)
    for i in range(len(vfitbis)-1):
        if bisector(vfitbis[i+1]) < 0 and bisector(vfitbis[i]) > 0:
            ma = vfitbis[i+1]
            mi = vfitbis[i]
    vmaxerr = (23-inc)/1000   #Stepsize of fitting space.
    vmax = spo.bisect(bisector,mi,ma,xtol=vmaxerr)
    if q == 1:
        Smax = Sfitsyn(vmax,p[0],p[1],p[2],p[3])
        Smaxerr = errSfitsyn(vmax,p[0],p[1],p[2],p[3],pcov)
    elif q == 2:
        Smax = Sfitplsyn(vmax,p[0],p[1],p[2],p[3],p[4],p[5])
        Smaxerr = errSfitplsyn(vmax,p[0],p[1],p[2],p[3],p[4],p[5],pcov)
    elif q == 4:
        Smax = Sfitplsyn(vmax,p[0],p[1],p[2],p[3],p[4],M)
        Smaxerr = errSfitplsyn(vmax,p[0],p[1],p[2],p[3],p[4],M,pcov)

###Printing peak coordinates.
print("\033[1m","    >>> Using scipy.optimize.bisect on {} model gives peak coordinates:
    ".format(modelname[q]),"\033[0m")
if q == 0 or q == 3 or q ==5:
    print(" A peak could not be determined, for a power law model is without one.")
else:
    print(" Fit frequency of peak is:",vmax,"GHz")
    print("  Uncertainty of frequency peak is:",vmaxerr,"GHz")
    print(" Fit flux density of peak is:",Smax,"mJy")
    print("  Uncertainty of flux peak is:",Smaxerr,"mJy")


###Episode VI: Creating pdf's from coeff's found in Ep.IV (or: Return of the Pdf).
```

```
    ###################################
print(">>> Episode VI: Return of the Pdf")

def mpdf(m1,m2,s11,s12,s21,s22,x): ###Multivariate pdf maker function.
    y = sps.multivariate_normal([m1,m2],[[s11,s12],[s21,s22]])
    return y.pdf(x)

def mpdfax(m1,m2,s11,s22): ###Multivariate pdf grid maker function.
    up1 = (m1+3*np.sqrt(s11)) ; up2 = (m2+3*np.sqrt(s22))
    low1 = (m1-3*np.sqrt(s11)) ; low2 = (m2-3*np.sqrt(s22))
    x1,x2 = np.mgrid[low1:up1:(up1-low1)/1000, low2:up2:(up2-low2)/1000]
    pos2 = np.empty(x1.shape + (2,))
    pos2[:, :, 0] = x1 ; pos2[:, :, 1] = x2
    return x1,x2,pos2


#####Episode VII: Plotting all results (or: The Pyplot Awakens).
    ###################################################
print(">>> Episode VII: The Pyplot Awakens")

###Plotting data, least squares fits and peaks (Episodes I, IV & V).
fig1 = figure(figsize=(40/2.54, 40/2.54))
frame1 = fig1.add_subplot(1,1,1,xscale="log",yscale="log")
frame1.set_title("{}".format(names[n]),fontsize=30)
frame1.set_xlim([0.25,30])
frame1.errorbar(v,S,yerr=Serr,ecolor="k",marker="*",mfc="y",ms=10,ls="",label="Recorded
    data")
if q == 0 or q == 3 or q == 5:
    print("Power law has no peak: no peak has been plotted.")
else:
    frame1.errorbar(vmax,Smax,xerr=vmaxerr,yerr=Smaxerr, ecolor="k",marker="o",mfc="b",
    ms=15,ls="",alpha=0.5,label="Peak")
frame1.plot(vfit,fitfun,"b",label="Fitted function (least squares)")
if q == 2:
    frame1.plot(vfit,Sfitsyn(vfit,p[0],p[1],p[2],p[3]),"r",label="Fitted function, power
     lawless")
frame1.fill_between(vfit,fitmax,fitmin,facecolor="blue",alpha=0.15)
frame1.set_xlabel('Frequency v (GHz)',fontsize=25)
frame1.set_ylabel('Flux density S (mJy)',fontsize=25)
frame1.tick_params(labelsize=25)

###Plotting 2d multivariate pdf results (Episode VI).
fig2 = figure(figsize=(40/2.54, 40/2.54))
if q == 0:
    a1,a2,pospos = mpdfax(p[0],p[1],pcov[0,0],pcov[1,1])
    frame2 = fig2.add_subplot(1,1,1)
    frame2.contourf(a1,a2,mpdf(p[0],p[1],pcov[0,0],pcov[0,1],pcov[1,0],pcov[1,1],pospos)
    )
    frame2.set_xlabel('coefficient ${}$'.format(coeflist[0+os]))
    frame2.set_ylabel('coefficient ${}$'.format(coeflist[1+os]))
elif q == 1 or q == 5:
    k = 0
    for i in range(ndim):
        for j in range(ndim):
            if j > i:
                a1,a2,pospos = mpdfax(p[i],p[j],pcov[i,i],pcov[j,j])
                frame2 = fig2.add_subplot(2,3,k+1)
                k = k + 1
                frame2.contourf(a1,a2,mpdf(p[i],p[j],pcov[i,i],pcov[i,j],pcov[j,i],pcov[
    j,j],pospos))
                frame2.set_xlabel('coefficient ${}$'.format(coeflist[i+os]))
                frame2.set_ylabel('coefficient ${}$'.format(coeflist[j+os]))
elif q == 2:
    print("Multivariate pdf's could not be plotted because of the limitations of
    detestable frequentist statistics.")
elif q == 3:
    a1,a2,pospos = mpdfax(p[0],p[1],pcov[0,0],pcov[1,1])
    frame2 = fig2.add_subplot(1,1,1)
    frame2.contourf(a1,a2,mpdf(p[0],p[1],pcov[0,0],pcov[0,1],pcov[1,0],pcov[1,1],pospos)
    )
```

```
487         frame2.set_xlabel('coefficient ${}$'.format(coeflist[0+os]))
488         frame2.set_ylabel('coefficient ${}$'.format(coeflist[1+os]))
489     elif q == 4:
490         k = 0
491         for i in range(ndim):
492             for j in range(ndim):
493                 if j > i:
494                     a1,a2,pospos = mpdfax(p[i],p[j],pcov[i,i],pcov[j,j])
495                     frame2 = fig2.add_subplot(3,4,k+1)
496                     k = k + 1
497                     frame2.contourf(a1,a2,mpdf(p[i],p[j],pcov[i,i],pcov[i,j],pcov[j,i],pcov[
        j,j],pospos))
498                     frame2.set_xlabel('coefficient ${}$'.format(coeflist[i+os]))
499                     frame2.set_ylabel('coefficient ${}$'.format(coeflist[j+os]))
500
501 show()
502
503 while True:
504     print("\033[1m","Do you wish to save the data and figures?","\033[0m")
505     answer = input("  :")
506     if answer.lower().startswith("y"):
507         print("Roger")
508         break
509     elif answer.lower().startswith("n"):
510         print("OKDOEI")
511         exit()
512
513 ###Saving figures to folder.
514 fig1.savefig("/Users/users/tjoa/Downloads/Thesis/Pics/Part1/Synchfits/Synfit{0}_{1}.png"
        .format(n,q))
515 if q == 2:
516     print("No multivariate pdf's were made and thus none were saved to directory.")
517 else:
518     fig2.savefig("/Users/users/tjoa/Downloads/Thesis/Pics/Part1/Contourplots/Conplt{0}_
        {1}.png".format(n,q))
519
520 ###Writing values to file.
521 if q == 0:
522     with open("ThesisTable_powerlaw0.dat", "a") as myfile:
523         myfile.write("{0} {1} {2} {3} {4} {5} {6}".format(n,names[n],p[0],pcov[0,0],p
        [1],pcov[1,1],"regpls"))
524         myfile.write("\n")
525 elif q == 1:
526     with open("ThesisTable_synchrotron0.dat", "a") as myfile:
527         myfile.write("{0} {1} {2} {3} {4} {5} {6} {7} {8} {9} {10} {11} {12} {13}".
        format(n,names[n],p[0],pcov[0,0],p[1],pcov[1,1],p[2],pcov[2,2],p[3],pcov[3,3],vmax,
        vmaxerr,Smax,Smaxerr,))
528         myfile.write("\n")
529 elif q == 2:
530     with open("ThesisTable_synch-powlaw0.dat", "a") as myfile:
531         myfile.write("{0} {1} {2} {3} {4} {5} {6} {7} {8} {9} {10} {11} {12} {13} {14}
        {15} {16} {17} {18}".format(n,names[n],p[0],pcov[0,0],p[1],pcov[1,1],p[2],pcov[2,2],
        p[3],pcov[3,3],p[4],pcov[4,4],p[5],pcov[5,5],vmax,vmaxerr,Smax,Smaxerr,"MCMC"))
532         myfile.write("\n")
533 elif q == 3:
534     with open("ThesisTable_powerlaw0.dat", "a") as myfile:
535         myfile.write("{0} {1} {2} {3} {4} {5} {6}".format(n,names[n],p[0],pcov[0,0],p
        [1],pcov[1,1],"agepls"))
536         myfile.write("\n")
537 elif q == 4:
538     with open("ThesisTable_synch-powlaw0.dat", "a") as myfile:
539         myfile.write("{0} {1} {2} {3} {4} {5} {6} {7} {8} {9} {10} {11} {12} {13} {14}
        {15} {16} {17}".format(n,names[n],p[0],pcov[0,0],p[1],pcov[1,1],p[2],pcov[2,2],p[3],
        pcov[3,3],p[4],pcov[4,4],M,0,vmax,vmaxerr,Smax,Smaxerr))
540         myfile.write("\n")
541 elif q == 5:
542     with open("ThesisTable_double-powlaw0.dat", "a") as myfile:
543         myfile.write("{0} {1} {2} {3} {4} {5} {6} {7} {8} {9}".format(n,names[n],p[0],
        pcov[0,0],p[1],pcov[1,1],p[2],pcov[2,2],p[3],pcov[3,3]))
544         myfile.write("\n")
```

B.2 Markov-chain Monte Carlo fitting algorithm

The following script is used to fit SSA GPS+power law spectra with fixed index m using an MCMC algorithm.

Scripts/Thesis_Sv–MCMCfit_synpl14–3.py

```
#!/usr/bin/env python
from __future__ import division
import numpy as np
import scipy.optimize as spo
import emcee as mc
import corner as co
from matplotlib.pyplot import figure, show
###THIS SCRIPT CAN RUN MCMC ON SYNCHROTRON PLUS POWER LAW SPECTRA ONLY, FOR FIXED m.

#####Episode I: Reading Data (or: The Phantom Data).
    ######################################################################
print(">>> Episode I: The Phantom Data")

###Reading relevant data from (mono-axial) datafile. If uncertainties were unknown, they
      have been set to 1.0 to prevent code collapse due to divide by zero errors.
fulllist = np.genfromtxt("Janskylist.dat",dtype=None,usecols=[0],comments="#")
errlist = np.genfromtxt("Janskylist.dat",dtype=None,usecols=[1],comments="#")

###Splitting the superarray into useful, seperate arrays with relevant data.
names,S003,S014,S048,S084,S149,S224 = np.split(fulllist,7)
junkerr,errS003,errS014,errS048,errS084,errS149,errS224 = np.split(errlist,7)
names = names.astype(str).flatten()
S003 = np.asarray(S003.astype(float).flatten())
S014 = np.asarray(S014.astype(float).flatten())
S048 = np.asarray(S048.astype(float).flatten())
S084 = np.asarray(S084.astype(float).flatten())
S149 = np.asarray(S149.astype(float).flatten())
S224 = np.asarray(S224.astype(float).flatten())

###Printing all data in a clear-cut table.
print("\033[1m","index * name * 0.325 * 1.4 * 4.85 * 8.46 * 14.94 * 22.46","\033[0m","
    <-- Frequency bands")
for i in range(len(names)):
    print(i,":","\033[1m",names[i],"\033[0m","*",S003[i],"*",S014[i],"*",S048[i],"*",
    S084[i],"*",S149[i],"*",S224[i])
n = int(input(" Pick GPS source: "))

###Creating the to be used data arrays.
v000 = [0.325,1.4,4.85,8.46,14.94,22.46]
S000 = [S003[n],S014[n],S048[n],S084[n],S149[n],S224[n]]
Serr000 = [errS003[n],errS014[n],errS048[n],errS084[n],errS149[n],errS224[n]]

v = np.asarray(v000)
S = np.asarray(S000)
Serr = np.asarray(Serr000)

print("Source designation:",names[n])
print("Frequency bands:",v)
print("Flux densities:",S)
print("Flux uncertainties:",Serr)

###Plotting the recorded data points for visualisation purposes.
fig = figure(figsize=(10,10))
frame = fig.add_subplot(1,1,1,xscale="log",yscale="log")
frame.errorbar(v,S,yerr=Serr,ecolor="k",marker="*",mfc="y",ms=10,ls="",label="Data from
    {}".format(names[n]))
frame.set_xlabel('Frequency v (GHz)')
frame.set_ylabel('Flux density S (mJy)')
frame.legend()
show()

###While-loop to allow user-friendly cancelling of code.
while True:
```

```
    print("\033[1m","Do you want to continue?","\033[0m")
    answer = input("  :")
    if answer.lower().startswith("y"):
        print("Roger: synch fit and power law online")
        break
    elif answer.lower().startswith("n"):
        print("OKDOEI")
        exit()

###Setting model type to synchrotron plus power law.
q = 2 ; ndim = 5 ; os=0
modelname = ["power law","synchrotron","synchrotron/power law"]
print(" Number of free parameters:",ndim)
M = (np.log10(0.5*S[1])-np.log10(S[0]))/(np.log10(v[1])-np.log10(v[0]))


#####Episode II: Creating probability functions (or: Attack of the Functions).
    ######################################
print(">>> Episode II: Attack of the Functions")

def Sfitsyn(v,S01,v01,k,l):
    f1 = S01*(v/v01)**k                          ###First power law
    f2 = (1-np.e**(-(v/v01)**(l-k)))/(1-np.e**(-1))   ###Second power law (exponential fall off)
    return f1*f2

def Sfitplsyn(v,S01,v01,k,l,A):
    f1 = S01*(v/v01)**k                          ###First power law
    f2 = (1-np.e**(-(v/v01)**(l-k)))/(1-np.e**(-1))   ###Second power law (exponential fall off)
    f3 = A*(v)**M                          ###Third power law (only when two turnovers are present)
    return f1*f2 + f3

def errSfitplsyn(v,S01,v01,k,l,A,cov):     ###Finding uncertainties function in synch fit plus power law.
    C = S01/(np.e-1)
    C2 = S01/(1-1/np.e)
    E = np.e**-((v/v01)**(l-k))
    E2 = np.e**(1-((v/v01)**(l-k)))
    dsdS01 = Sfitsyn(v,S01,v01,k,l)/S01
    dsdv01 = (-((C2*(1-E)))*k*v*((v/v01)**(k-1)))/(v01**2) - (C2*(l-k)*v*E*((v/v01)**(l-1)))/(v01**2)
    dsdk = C2*(1-E)*((v/v01)**k)*np.log(v/v01) - C2*((v/v01)**(l))*np.log(v/v01)*E
    dsdl = C*((v/v01)**l)*np.log(v/v01)*E2
    dsdA = v**M
    dsdm = A*(v**M)*np.log(v)
    dsdx = np.asarray([dsdS01,dsdv01,dsdk,dsdl,dsdA,dsdm])
    Svar = 0
    for i in range(ndim):
        for j in range(ndim):
            Svar = Svar + dsdx[i]*dsdx[j]*cov[i,j]
    return np.sqrt(Svar)

def derSfitplsyn(v,S01,v01,k,l,A):              ###Derivative of power law/synchrotron combined fit
    E = np.e**(-(v/v01)**(l-k))
    C = S01/(v01*(1-1/np.e))
    f1 = C*k*((v/v01)**(k-1))*(1-E)
    f2 = C*(l-k)*((v/v01)**(l-1))*E
    f3 = A*M*v**(M-1)
    return f1 + f2 + f3

###Array spanning the fitting space.
vfit = np.linspace(0.3,23,1000)

###List that names the coefficients for the synchfit.
coeflist = ["S_{01}","v_{01}","k","l","A","m"]


```

```
#####Episode III: Computing estimates for free parameters. (or: Revenge of the Estimates
    ). ##########################
print(">>> Episode III: Revenge of the Estimates")

###Creating some new, useful arrays and indices.
S2 = [S[1],S[2],S[3],S[4],S[5]]
v2 = [v[1],v[2],v[3],v[4],v[5]]
Smi = S000.index(max(S000))
Smi2 = S2.index(max(S2))

###Finding estimates for free parameters.
S0guess = S[4]
v0guess = v[4]
if S2[1] > S2[0]:
    kguess = (np.log10(S2[1])-np.log10(0.5*S2[0]))/(np.log10(v2[1])-np.log10(v2[0]))
else:
    if S2[2] > S2[1]:
        kguess = (np.log10(S2[2])-np.log10(0.5*S2[1]))/(np.log10(v2[2])-np.log10(v2[1]))
    else:
        kguess = 2.0
if S2[4] < S2[3]:
    lguess = (np.log10(S2[4])-np.log10(S2[3]))/(np.log10(v2[4])-np.log10(v2[3]))
else:
    if S2[4] < S2[2]:
        lguess = (np.log10(S2[4])-np.log10(S2[2]))/(np.log10(v2[4])-np.log10(v2[2]))
    else:
        lguess = -1.0
Aguess = S[0]/(v[0]**M)

estimates = [S0guess,v0guess,kguess,lguess,Aguess,M]
print("Parameter estimates:")
print(np.asarray(estimates))

pg, pgcov = spo.curve_fit(Sfitplsyn,v,S,[S0guess,v0guess,kguess,lguess,Aguess],sigma=
    Serr,bounds=([0,0.3,0,-2.5,0],[500,23,2.5,0,100]),**{'max_nfev':1000000})
for i in range(ndim-1):
    print(pg[i],pgcov[i,i])
print(M)

Sstd = 2.0 ; Slow = 0.0 ; Supp = 500.0                    ###Constraints on S0
vstd = 1.0 ; vlow = v[1] ; vupp = 23.0                    ###Constraints on v0
kstd = 0.5 ; klow = 0 ; kupp = 2.5                    ###Constraints on k
lstd = 0.5 ; llow = -2.5 ; lupp = 0.0                    ###Constraints on l
Astd = 2.0 ; Alow = 0.0 ; Aupp = 100.0                    ###Constraints on A

def normdist(x,mean,std):                    ###Normal (Gaussian) distribution
    expterm = ((x-mean)**2)/(2*std**2)
    f = (1/(std*np.sqrt(2*np.pi)))*np.e**-expterm
    return f

def lnprior(theta):                    ###ln of prior
    S01,v01,k,l,A = theta
    if Slow < S01 < Supp and vlow < v01 < vupp and klow < k < kupp and llow < l < lupp
    and Alow < A < Aupp:
        return np.log(normdist(k,pg[2],kstd)*normdist(l,pg[3],lstd))
    return -np.inf

def lnlike(theta,v,S,Serr):                    ###ln of likelihood
    S01,v01,k,l,A = theta
    f1 = S01*(v/v01)**k                    ###First power law
    f2 = (1-np.e**(-(v/v01)**(l-k)))/(1-np.e**(-1))    ###Second power law (exponential
    fall off)
    f3 = A*(v)**M                    ###Third power law (only when two turnovers are
    present)
    model = f1*f2+f3
    inv_sigma2 = 1.0/(Serr**2)
    return -0.5*(np.sum((S-model)**2*inv_sigma2 + np.log(2*np.pi/inv_sigma2)))

def lnprob(theta,v,S,Serr):                    ###ln of posterior
    lp = lnprior(theta)
```

```
187     if lp == -np.inf:
188         return -np.inf
189     return lp + lnlike(theta,v,S,Serr)
190
191
192 #####Episode IV: Performing MCMC. (or: A New Method).
        ##################################################################
193 print(">>> Episode IV: A New Method")
194
195 ###Setting dimensions, walkers and mainframe.
196 nwalkers = 300
197 p0 = [pg + 1e-2*np.random.randn(ndim) for i in range(nwalkers)]
198 sampler = mc.EnsembleSampler(nwalkers, ndim, lnprob, args=(v,S,Serr))
199
200 ###Burn-in & reset.
201 print(" Performing burn-in sequence...")
202 pos, prob, state = sampler.run_mcmc(p0, 200)
203 sampler.reset()
204
205 #If lnprob0 and newlnprob BOTH return -inf, then lnpdiff is nan. If lnprob0 ALONE
        returns -inf, lnpdiff is inf. If newlnprob ALONE returns -inf, lnpdiff is -inf.
206
207 ###Running real MCMC!
208 print(" Initiating main MCMC algorithm...")
209 nsteps = 1000
210 sampler.run_mcmc(pos, nsteps, rstate0=state)
211
212 ###Generating the cornerplots
213 print(" Generating cornerplots...")
214 burnin = 50
215 samples = sampler.chain[:, burnin:, :].reshape((-1, ndim))
216
217
218 #####Episode V: Finding parameter value & covariance matrix (or: The Parameters Strike
        Back). ######################
219 print(">>> Episode V: The Parameters Strike Back")
220
221 resolution = 1000
222
223 def MCs(a,i):                          ###Sampler function.
224     return a.flatchain[:,i]
225
226 def histo(x):                          ###Histogram maker.
227     hist = np.histogram(x,resolution)
228     counts = hist[0] ; values = hist[1]
229     values = np.delete(values,resolution)
230     a = np.asarray([])
231     for i in range(resolution):
232         appa = np.full(counts[i],values[i])
233         a = np.append(a,appa)
234     return a,counts,values
235
236 ###Creating total probability array from six dimensions, extracting maximum probability
        index.
237 lnprobarr = sampler.flatlnprobability
238 mindex = np.argmax(lnprobarr)
239
240 ###Filling the parameter array and the covariance matrix.
241 phist = []
242 p = np.zeros(ndim)
243 print("\033[1m"+"    >>> Using MCMC on {} model:".format(modelname[q]),"\033[0m")
244 for i in range(ndim):
245     hishis = histo(MCs(sampler,i))
246     phist.append(hishis[0])
247     p[i] = MCs(sampler,i)[mindex]
248
249 phist = np.asarray(phist)
250 pcov = np.cov(phist)
251
252 for i in range(ndim):
```

```
253     print(" Coefficient",coeflist[i+os],"equals",p[i])
254     print("   Uncertainty of coefficient",coeflist[i+os],"equals",np.sqrt(pcov[i,i]))
255 print(" Coefficient m equals:",M)
256
257
258 #####Episode VI: Finding the maximum of the fit (or: Return of the Maxima).
    ##########################################
259 print(">>> Episode VI: Return of the Maxima")
260
261 ###Array spanning the bisection space, hence excluding the minimum to get the turnover
    as return from scipy.bisect.
262 if S[2] > S[1]:
263     inc = (v[2]+v[1])/2
264 else:
265     if S[3] > S[2]:
266         inc = (v[3]+v[2])/2
267     else:
268         inc = v[3]
269
270 vfitbis = np.linspace(inc,23,1000)
271
272 def bisector(v):                    ###Function of one variable used by the bisection
    algorithm.
273     return derSfitplsyn(v,*p)
274
275 ###If loop that bisects the function if it has one or more peaks.
276 for i in range(len(vfitbis)-1):
277     if bisector(vfitbis[i+1]) < 0 and bisector(vfitbis[i]) > 0:
278         ma = vfitbis[i+1]
279         mi = vfitbis[i]
280 vmax = spo.bisect(bisector,mi,ma)
281 vmaxerr = (23-inc)/1000 #Range and stepsize of fitting space.
282 Smax = Sfitplsyn(vmax,*p)
283 Smaxerr = errSfitplsyn(vmax,*p,pcov)
284
285 ###Printing peak coordinates.
286 print("\033[1m","    >>> Using scipy.optimize.bisect on {} model gives peak coordinates:
    ".format(modelname[q]),"\033[0m")
287 print(" Fit frequency of peak is:",vmax,"GHz")
288 print("  Uncertainty of frequency peak is:",vmaxerr,"GHz")
289 print(" Fit flux density of peak is:",Smax,"mJy")
290 print("  Uncertainty of flux peak is:",Smaxerr,"mJy")
291
292
293 #####Episode VII: Plotting all results (or: The Pyplot Awakens).
    ##########################################################
294 print(">>> Episode VII: The Pyplot Awakens")
295
296 fitfun = Sfitplsyn(vfit,*p)
297 fitfun2 = Sfitplsyn(vfit,*pg)
298 fitmax = fitfun + errSfitplsyn(vfit,*p,pcov)
299 fitmin = fitfun - errSfitplsyn(vfit,*p,pcov)
300
301 ###Plotting data, least squares fits and peaks (Episodes I, IV & V).
302 fig1 = figure(figsize=(40/2.54, 40/2.54))
303 frame1 = fig1.add_subplot(1,1,1,xscale="log",yscale="log")
304 frame1.set_title("{}".format(names[n]),fontsize=30)
305 frame1.set_xlim([0.25,30])
306 frame1.errorbar(v,S,yerr=Serr,ecolor="k",marker="*",mfc="y",ms=10,ls="",label="Recorded
    data")
307 frame1.errorbar(vmax,Smax,xerr=vmaxerr,yerr=Smaxerr, ecolor="k",marker="o",mfc="b",ms
    =15,ls="",alpha=0.5,label="Peak")
308 frame1.plot(vfit,fitfun,"b",label="Fitted function (MCMC-max)")
309 frame1.plot(vfit,fitfun2,"r",label="Fitted function (least squares)")
310 frame1.fill_between(vfit,fitmax,fitmin,facecolor="blue",alpha=0.15)
311 frame1.set_xlabel('Frequency v (GHz)',fontsize=25)
312 frame1.set_ylabel('Flux density S (mJy)',fontsize=25)
313 frame1.tick_params(labelsize=25)
314
315 ###Plotting cornerplots
```

```
fig2 = co.corner(samples, labels=["$S_{01}$","$v_{01}$","$k$","$l$","$A$","$m$"])

###Plotting MCMC results (histograms for each coefficient) (Episode V).
fig3 = figure(figsize=(40/2.54, 40/2.54))
for i in range(ndim):
    frame3 = fig3.add_subplot(3,2,i+1)
    frame3.hist(MCs(sampler,i),normed=True,bins=200,alpha=0.8)
    frame3.axvline(x=p[i],linewidth=2,color='r')
    frame3.axvline(x=pg[i],linewidth=1,color='g')
    frame3.set_xlabel("coefficient {}".format(coeflist[i+os]))
    frame3.set_ylabel("walker counts")

show()

while True:
    print("\033[1m","Do you wish to save the data and figures?","\033[0m")
    answer = input("  :")
    if answer.lower().startswith("y"):
        print("Roger")
        break
    elif answer.lower().startswith("n"):
        print("OKDOEI")
        exit()

###Saving figures to folder.
fig1.savefig("/Users/users/tjoa/Downloads/Thesis/Pics/Part1/Synchfits/Synfit{0}-{1}_{2}.
    png".format(names[n],n,q))
fig2.savefig("/Users/users/tjoa/Downloads/Thesis/Pics/Part1/Contourplots/Conplt{0}-{1}_
    {2}.png".format(names[n],n,q))
fig3.savefig("/Users/users/tjoa/Downloads/Thesis/Pics/Part1/MCMCHistograms/Histogram
    {0}-{1}_{2}.png".format(names[n],n,q))


###Writing values to file.
with open("ThesisTable_synch-powlaw4.dat", "a") as myfile:
    myfile.write("{0} {1} {2} {3} {4} {5} {6} {7} {8} {9} {10} {11} {12} {13} {14} {15}
    {16} {17}".format(n,names[n],p[0],pcov[0,0],p[1],pcov[1,1],p[2],pcov[2,2],p[3],pcov
    [3,3],p[4],pcov[4,4],M,"N/A",vmax,vmaxerr,Smax,Smaxerr))
    myfile.write("\n")
```

B.3 K-z relation fitting algorithm

The following script is used to fit the K-z relation and obtain redshifts from our sample.

Scripts/Thesis_Kz–relation_v13–1.py

```
#!/usr/bin/env python
from __future__ import division
import numpy as np
import scipy.stats as sps
import numpy.random as npr
from matplotlib.pyplot import figure, show
###THIS SCRIPT COMPUTES STD IN LOG10(z) AND z FROM K-BAND MAGNITUDES.


#####Episode I: Reading Data (or: The Phantom Data).
    ##################################################################
print(">>> Episode I: The Phantom Data")

###Reading relevant data from datafile.
datanames = np.genfromtxt("fulltable.dat",dtype=str,usecols=[0],comments="#")
zlist = np.genfromtxt("fulltable.dat",dtype=float,usecols=[1],comments="#")
zstdlowlist = np.genfromtxt("fulltable.dat",dtype=float,usecols=[2],comments="#")
zstdupplist = np.genfromtxt("fulltable.dat",dtype=float,usecols=[3],comments="#")
Klist = np.genfromtxt("fulltable.dat",dtype=float,usecols=[4],comments="#")
Kstdlist = np.genfromtxt("fulltable.dat",dtype=float,usecols=[5],comments="#")

###Printing all data in a clear-cut table.
```

```
print("\033[1m","index  *   source    *   z   *zstdlow *zstdup * K-mag * K-std""\033[0m")
for i in range(len(datanames)):
    print(i,"     :","\033[1m",datanames[i],"\033[0m","{0}  {1}   {2}   {3}   {4}".format(zlist[i],zstdlowlist[i],zstdupplist[i],Klist[i],Kstdlist[i]))


#####Episode II: Defining the functions and general constants (or: Attack of the Functions). ########################
print(">>> Episode II: Attack of the Functions")

###Function block.
def Klogzrel(logz):                         ###K,log10(z)-relation function
    return 17.37 + 4.53*logz + 0.31*logz**2

def Kzrel(z):                               ###K,z-relation function
    return 17.37 + 4.53*np.log10(z) + 0.31*(np.log10(z))**2

###Specifying general constants and limits of the fit.
stdK1 = 0.593
logzmin = -4 ; logzmax = np.log10(10)
ranlen = 5000000


#####Episode III: Generating fit line and random data (or: Revenge of the Randomness). ##############################
print(">>> Episode III: Revenge of the Randomness")

###Generating plain, idealized K-log10(z) relation.
logzarr = np.linspace(logzmin,logzmax,ranlen)
Karr = Klogzrel(logzarr)

###Generating the random points around idealized K-log10(z) relation.
logzran = (logzmax-logzmin)*npr.random(ranlen)+logzmin
Kran = np.zeros(ranlen)
for i in range(ranlen):
    Kran[i] = npr.normal(Klogzrel(logzran[i]),stdK1)


#####Episode IV: Finding redshift uncertainties in data (or: A New Data). ###########################################
print(">>> Episode IV: A New Data")

def uberfunctionarr(DATA,DATAerr):  ###Function that computes distribution around an array of Ksamples
    ###Specifying arrays and values to be used in log10(z)/z uncertainty computation loop as empty zero arrays.
    Ksample = DATA                          ###Data sample of K points
    Kdataerr = DATAerr                      ###K data uncertainties
    dic = {}                                ###Empty dictionary for log10(z) values, to be filled in loop
    dic2 = {}                               ###Empty dictionary for z values, to be filled in loop
    logzsample = np.zeros(len(Ksample))         ###Sample of log10(z) points
    zsample = np.zeros(len(Ksample))            ###Corresponding sample of z points
    stdK = stdK1*np.ones(len(Ksample))          ###Array of std_K values of sample length
    stdlogz = np.zeros(len(Ksample))            ###Empty std_log10(z) array
    stdzlow = np.zeros(len(Ksample))            ###Empty std_z (lower bound) array
    stdzupp = np.zeros(len(Ksample))            ###Empty std_z (upper bound) array
    step = 0.05                             ###Step size for the loop

    ###Loop that computes the uncertainty in log10(z) and z at some z using an algorithm that takes samples from a small interval, then creates the corresponding distribution and returns the std in log10(z) and z.
    for i in range(len(Ksample)):
        Kupp = Ksample[i]+step ; Klow = Ksample[i]-step
                ###Upper and lower interval bounds
        dic["logzdist_{}".format(i)] = np.asarray([])
          ###Creating new log10(z) dictionary entry
```

```
78          dic2["zdist_{}".format(i)] = np.asarray([])
              ###Creating new z dictionary entry
79          for j in range(ranlen):
80              if Kran[j] > Klow and Kran[j] < Kupp:
              ###Acceptance condition
81                  dic["logzdist_{}".format(i)] = np.append(dic["logzdist_{}".format(i)],
    logzran[j])     ###Appending accepted values to corresponding log10(z) dictionary
    entry
82                  dic2["zdist_{}".format(i)] = np.append(dic2["zdist_{}".format(i)],10**
    logzran[j])     ###Appending accepted values to corresponding z dictionary entry
83          data = dic["logzdist_{}".format(i)]
              ###Renaming dictionary entry for easy handling
84          data2 = dic2["zdist_{}".format(i)]
              ###Renaming dictionary entry for easy handling
85
86          ###Fitting a normal distribution to dictionary data.
87          x = np.linspace(data.min(), data.max(), 100)              ###Creating regular
    possible value space
88          param = sps.norm.fit(data)                              ###Fitting normal parameters
     to the data with the mean at the corresponding log10(z) value.
89          pdf_fitted = sps.norm.pdf(x, *param)                    ###Fitting a normal
    distribution with said parameters to the interval
90          prob = pdf_fitted/pdf_fitted.sum()                      ###Normalizing pdf
91
92          ###Fitting a log-normal distribution to dictionary 2 data.
93          x2 = np.linspace(data2.min(), data2.max(), 100)                ###Creating
    regular possible value space
94          param2 = sps.lognorm.fit(data2)                         ###Fitting log-normal
    parameters to the data with the mean at the corresponding z value.
95          pdf_fitted2 = sps.lognorm.pdf(x2, *param2)              ###Fitting a log-
    normal distribution with said parameters to the interval
96          prob2 = pdf_fitted2/pdf_fitted2.sum()                   ###Normalizing pdf
97
98          ###Computing required statistics from dictionary data.
99          logzmu = x.dot(prob)                                    ###Computing mean in log10(z
    )
100         logzsample[i] = logzmu                                  ###Entering into ex-
    loop logz array
101         zmu = 10**logzmu                                        ###Computing mean in z
102         zsample[i] = zmu                                        ###Entering into ex-loop z
    array
103         stdlogz1 = np.sqrt(np.power(x,2).dot(prob) - logzmu**2)         ###Computing std
     in log10(z) (sqrt(data_logz^2 - mu_logz^2))
104         stdlogz[i] = stdlogz1                                   ###Entering into ex-loop
    stdlogz array
105         stdzlow1 = 10**(logzmu-stdlogz1)-zmu                    ###Computing std lower
     bound in z 10^(mu_logz-std_logz)-mu_z
106         stdzlow[i] = -1*stdzlow1                                ###Entering into ex-loop
    stdzlow array
107         stdzupp1 = 10**(logzmu+stdlogz1)-zmu                    ###Computing std upper
     bound in z 10^(mu_logz+std_logz)-mu_z
108         stdzupp[i] = stdzupp1                                   ###Entering into ex-loop
    stdzipp array
109     return(Ksample,Kdataerr,logzsample,stdlogz,zsample,stdzlow,stdzupp,dic,dic2)
110
111 def uberfunctionflt(DATA,DATAerr):  ###Function that computes distribution around a
    single Ksample
112     ###Specifying arrays and values to be used in log10(z)/z uncertainty computation
    loop as empty zero arrays.
113     Ksample = DATA              ###Data sample of K points
114     Kdataerr = DATAerr          ###K data uncertainties
115     stdK = stdK1          ###Array of std_K values of sample length
116     logzdist = np.asarray([]) ###Set of logz-values
117     zdist = np.asarray([])    ###Set of z-values
118     step = 0.05               ###Step size
119
120     ###Computing the uncertainty in log10(z) and z at some z using an algorithm that
    takes samples from a small interval, then creating the corresponding distribution
    and returing the std in log10(z) and z.
121     Kupp = Ksample+step ; Klow = Ksample-step                   ###Upper and lower
```

```
    interval bounds
    for j in range(ranlen):
        if Kran[j] > Klow and Kran[j] < Kupp:                    ###Acceptance
    condition
            logzdist = np.append(logzdist,logzran[j])            ###Appending accepted
    values to corresponding log10(z) dictionary entry
            zdist = np.append(zdist,10**logzran[j])              ###Appending accepted
    values to corresponding z dictionary entry
    data = logzdist                                          ###Renaming dictionary entry
     for easy handling
    data2 = zdist                                        ###Renaming dictionary entry for
    easy handling

    ###Fitting a normal distribution to data.
    x = np.linspace(data.min(), data.max(), 100)                 ###Creating regular
    possible value space
    param = sps.norm.fit(data)                                   ###Fitting normal
    parameters to the data with the mean at the corresponding log10(z) value.
    pdf_fitted = sps.norm.pdf(x, *param)                     ###Fitting a normal
    distribution with said parameters to the interval
    prob = pdf_fitted/pdf_fitted.sum()                           ###Normalizing pdf

    ###Fitting a log-normal distribution to data2.
    x2 = np.linspace(data2.min(), data2.max(), 100)              ###Creating regular
    possible value space
    param2 = sps.lognorm.fit(data2)                          ###Fitting log-normal
    parameters to the data with the mean at the corresponding z value.
    pdf_fitted2 = sps.lognorm.pdf(x2, *param2)                   ###Fitting a log-
    normal distribution with said parameters to the interval
    prob2 = pdf_fitted2/pdf_fitted2.sum()                    ###Normalizing pdf

    ###Computing required statistics from data.
    logzmu = x.dot(prob)                                     ###Computing mean in log10(z
    )
    zmu = 10**logzmu                                         ###Computing mean in z
    stdlogz1 = np.sqrt(np.power(x,2).dot(prob) - logzmu**2)      ###Computing std in
    log10(z) (sqrt(data_logz^2 - mu_logz^2))
    stdzlow1 = 10**(logzmu-stdlogz1)-zmu                     ###Computing std lower bound
     in z 10^(mu_logz-std_logz)-mu_z
    stdzupp1 = 10**(logzmu+stdlogz1)-zmu                     ###Computing std upper bound
     in z 10^(mu_logz+std_logz)-mu_z
    return(Ksample,Kdataerr,logzmu,stdlogz1,zmu,stdzlow1,stdzupp1)

newKlist = []
newKerrlist = []
newdatanames = []
for i in range(len(Klist)):
    if Klist[i] == 0.0:
        print("Entry",i,"has no known K-band magnitude.")
    else:
        UBER = uberfunctionflt(Klist[i],Kstdlist[i])
        zlist[i] = UBER[4]
        zstdlowlist[i] = np.abs(UBER[5])
        zstdupplist[i] = UBER[6]
        newKlist.append(Klist[i])
        newKerrlist.append(Kstdlist[i])
        newdatanames.append(datanames[i])
newKarr = np.asarray(newKlist)
newKerrarr = np.asarray(newKerrlist)
fdata5 = uberfunctionarr(newKarr,newKerrarr)

###Printing all data in a clear-cut table.
print("\033[1m","index  *   source    *   z   *zstdlow *zstdup * K-mag * K-std""\033[0m"
    )
for i in range(len(datanames)):
    print(i,"     :","\033[1m",datanames[i],"\033[0m","{0}   {1}   {2}   {3}   {4}".
    format(np.round(zlist[i],4),np.round(zstdlowlist[i],4),np.round(zstdupplist[i],4),
    Klist[i],Kstdlist[i]))


```

```
#####Episode V: Plotting results (or: The Pyplot Strikes Back).
    ######################################################
print(">>> Episode V: The Pyplot Strikes Back")

###Plot no.1: K-band magnitude versus log10(z) via idealized K,log10(z)-relation and
    simulated data around it.
fig1 = figure(figsize=(40/2.54, 40/2.54))
frame1 = fig1.add_subplot(1,1,1)
frame1.set_title("K-band magnitude vs log10(z)",fontsize=15)
frame1.plot(logzran,Kran,'g.',alpha=0.1,label="Simulated random distibution")
frame1.plot(logzarr,Karr,linewidth=5,alpha=0.5,label="Idealized K,log10(z)-relation")
frame1.errorbar(fdata5[2],fdata5[0],xerr=fdata5[3],yerr=fdata5[1],ecolor="k",elinewidth
    =3,marker="*",mfc="y",ms=10,ls="",label="Sample K,z")
frame1.set_xlabel('Redshift log10(z)')
frame1.set_ylabel('K-band magnitude K')
frame1.legend(loc=0)

###Plot no.2: K-band magnitude versus z via idealized K,z-relation and simulated data
    around it.
fig2 = figure(figsize=(40/2.54, 40/2.54))
frame2 = fig2.add_subplot(1,1,1)
frame2.set_title("K-band magnitude vs z",fontsize=15)
frame2.plot(10**logzran,Kran,'g.',alpha=0.1,label="Simulated random distibution")
frame2.plot(10**logzarr,Karr,linewidth=5,alpha=0.5,label="Idealized K,z-relation")
assymerr3 = [fdata5[5],fdata5[6]]
frame2.errorbar(fdata5[4],fdata5[0],yerr=fdata5[1],xerr=assymerr3,ecolor="k",elinewidth
    =3,marker="*",mfc="y",ms=10,ls="",label="Sample K,z")
frame2.set_xlabel('Redshift z')
frame2.set_ylabel('K-band magnitude K')
frame2.legend(loc=0)

###Plot no.7: (data, 5 arcsec) log10(z) samples at some value of K and log10(z), showing
     normal distributions.
fig7 = figure(figsize=(40/2.54, 40/2.54))
for i in range(len(fdata5[2])):
    frame7 = fig7.add_subplot(4,4,i+1)
    data = fdata5[7]["logzdist_{}".format(i)]
    x = np.linspace(data.min(), data.max(), 100)
    param = sps.norm.fit(data, floc=fdata5[2][i])
    pdf_fitted = sps.norm.pdf(x, *param)
    frame7.set_title("{}".format(newdatanames[i]),fontsize=22)
    frame7.plot(x, pdf_fitted, color='m')
    frame7.axvline(x=fdata5[2][i],linewidth=3,color='r')          ###Mean
    frame7.axvline(x=fdata5[2][i]-fdata5[3][i],linewidth=2,color='k')   ###Std_logz
    lower
    frame7.axvline(x=fdata5[2][i]+fdata5[3][i],linewidth=2,color='k')   ###Std_logz
    upper
    frame7.hist(data,normed=True,bins=50,alpha=0.3)
    frame7.tick_params(labelsize=18)
fig7.tight_layout()

###Plot no.8: (data, 5 arcsec) z samples at some value of K and z, showing lognormal
    distributions.
fig8 = figure(figsize=(40/2.54, 40/2.54))
for i in range(len(fdata5[4])):
    frame8 = fig8.add_subplot(4,4,i+1)
    data = fdata5[8]["zdist_{}".format(i)]
    x = np.linspace(data.min(), data.max(), 100)
    param = sps.lognorm.fit(data, floc=0)
    pdf_fitted = sps.lognorm.pdf(x, *param)
    frame8.set_title("{}".format(newdatanames[i]),fontsize=22)
    frame8.plot(x, pdf_fitted, color='m')
    frame8.axvline(x=fdata5[4][i],linewidth=3,color='r')          ###Median
    frame8.axvline(x=fdata5[4][i]-fdata5[5][i],linewidth=2,color='k')   ###Std_z lower
    frame8.axvline(x=fdata5[4][i]+fdata5[6][i],linewidth=2,color='k')   ###Std_z upper
    frame8.hist(data,normed=True,bins=50,alpha=0.3)
    frame8.tick_params(labelsize=18)
fig8.tight_layout()

show()
```

B.4 Skymap creator

The following script is used to obtain skymaps of the source positions and predicted densities of GPS.

Scripts/Thesis_synthesis3_v2–4.py

```
#!/usr/bin/env python
from __future__ import division
import numpy as np
import numpy.random as npr
import scipy.optimize as spo
import fnmatch as fm
from matplotlib.pyplot import figure, show
from mpl_toolkits.mplot3d import Axes3D
###THIS SCRIPT COMPUTES fun.


#####Synthesis I: Reading Data.
    ##############################################################################################
print(">>> Synthesis I")

###Reading Tjoa&McKean's name and data (because it's largely monocolumnar, this takes
    some time and messy coding).
dat0TJOA = np.genfromtxt("poslist-McKean.dat",dtype=None,usecols=[0],comments="#")
dat1TJOA = np.genfromtxt("poslist-McKean.dat",dtype=None,usecols=[1],comments="#")
dat2TJOA = np.genfromtxt("poslist-McKean.dat",dtype=None,usecols=[2],comments="#")

namesTJOA,RAhrsTJOA,decdegTJOA = np.split(dat0TJOA,3)
namesTJOA = namesTJOA.astype(str).flatten()
RAhrsTJOA = np.asarray(RAhrsTJOA.astype(float).flatten())
decdegTJOA = np.asarray(decdegTJOA.astype(float).flatten())

poep1TJOA,RAminTJOA,decminTJOA = np.split(dat1TJOA,3)
RAminTJOA = np.asarray(RAminTJOA.astype(float).flatten())
decminTJOA = np.asarray(decminTJOA.astype(float).flatten())

poep2TJOA,RAsecTJOA,decsecTJOA = np.split(dat2TJOA,3)
RAsecTJOA = np.asarray(RAsecTJOA.astype(float).flatten())
decsecTJOA = np.asarray(decsecTJOA.astype(float).flatten())

###Reading Tjoa&McKean's synch and synch+powlaw data for spectral peaks.
purenamesTJOA = np.genfromtxt("ThesisTable_synchrotronNEW.dat",dtype=str,usecols=[1],
    comments="#")
combinamesTJOA = np.genfromtxt("ThesisTable_synch-powlawNEW.dat",dtype=str,usecols=[1],
    comments="#")
purevmaxTJOA = np.genfromtxt("ThesisTable_synchrotronNEW.dat",dtype=float,usecols=[10],
    comments="#")
combivmaxTJOA = np.genfromtxt("ThesisTable_synch-powlawNEW.dat",dtype=float,usecols
    =[14],comments="#")
vmaxTJOA = np.concatenate((purevmaxTJOA,combivmaxTJOA))
pureSmaxTJOA = np.genfromtxt("ThesisTable_synchrotronNEW.dat",dtype=float,usecols=[12],
    comments="#")
combiSmaxTJOA = np.genfromtxt("ThesisTable_synch-powlawNEW.dat",dtype=float,usecols
    =[16],comments="#")
SmaxTJOA = np.concatenate((pureSmaxTJOA,combiSmaxTJOA))


#####Synthesis II: Reloaded. Computation of results.
    ##########################################################
print(">>> Synthesis II: Reloaded")

def posmaker(RAhrs,RAmin,RAsec,decdeg,decmin,decsec): ###Function that computes the RA
    and dec positions in fractional hours and degrees respectively for plotting purposes
    RApos = RAhrs + RAmin/60 + (RAsec/60)/60
    decpos = decdeg + decmin/60 + (decsec/60)/60
    return RApos,decpos

def hidimpos(RA,dec):                                   ###Function that projects RA and dec
    data onto a sphere
    x = np.cos(dec*np.pi/180)*np.sin(15*RA*np.pi/180)
    y = np.cos(dec*np.pi/180)*np.cos(15*RA*np.pi/180)
```

```
    z = np.sin(dec*np.pi/180)
    return x,y,z

def areacalc(RAlow,RAupp,declow,decupp):
    declow = declow*np.pi/180 ; decupp = decupp*np.pi/180
    RAlow = 15*RAlow*np.pi/180 ; RAupp = 15*RAupp*np.pi/180
    return (RAupp - RAlow)*(np.sin(decupp) - np.sin(declow))

###Computing the positions of Tjoa&McKean's, Marlow's and Willott's sources.
posTJOA = posmaker(RAhrsTJOA,RAminTJOA,RAsecTJOA,decdegTJOA,decminTJOA,decsecTJOA)

###Computing positions of pure and combined samples
pureRATJOA = []
puredecTJOA = []
for i in range(len(purenamesTJOA)):
    for j in range(len(namesTJOA)):
        if fm.fnmatch(purenamesTJOA[i],"{}".format(namesTJOA[j])) is True:
            pureRATJOA.append(posTJOA[0][j])
            puredecTJOA.append(posTJOA[1][j])
pureRATJOA = np.asarray(pureRATJOA)
puredecTJOA = np.asarray(puredecTJOA)

combiRATJOA = []
combidecTJOA = []
for i in range(len(combinamesTJOA)):
    for j in range(len(namesTJOA)):
        if fm.fnmatch(combinamesTJOA[i],"{}".format(namesTJOA[j])) is True:
            combiRATJOA.append(posTJOA[0][j])
            combidecTJOA.append(posTJOA[1][j])
combiRATJOA = np.asarray(combiRATJOA)
combidecTJOA = np.asarray(combidecTJOA)

peaknamesTJOA = list(purenamesTJOA)
peaknamesTJOA.extend(combinamesTJOA)
peakSmaxTJOA = np.concatenate((pureSmaxTJOA,combiSmaxTJOA))
peakRATJOA = np.concatenate((pureRATJOA,combiRATJOA))
peakdecTJOA = np.concatenate((puredecTJOA,combidecTJOA))

###Computing Mirach and Grumium's positions as reference material for skymaps.
Mirachpos = posmaker(1.0,9.0,43.92388,35.0,37.0,14.0075)
Grumiumpos = posmaker(17.0,53.0,31.72962,56.0,52.0,21.5143)

###Loop that splits Tjoa&McKean's data (all of it) into its two component fields of view
    (1<RA<2, 35<dec<45 low and 17<RA<18, 55<dec<60 high).
RAhigh = [] ; dechigh = []
RAlow = [] ; declow = []
for i in range(len(posTJOA[0])):
    if posTJOA[0][i] > 2.0:
        RAhigh.append(posTJOA[0][i])
        dechigh.append(posTJOA[1][i])
    elif posTJOA[0][i] < 2.0:
        RAlow.append(posTJOA[0][i])
        declow.append(posTJOA[1][i])

###Loop that splits Tjoa&McKean's data (only peaked) into its two component fields of
    view (1<RA<2, 35<dec<45 low and 17<RA<18, 55<dec<60 high).
RAhighp = [] ; dechighp = [] ; Smaxhigh = []
RAlowp = [] ; declowp = [] ; Smaxlow = []
for i in range(len(peakRATJOA)):
    if peakRATJOA[i] > 2.0:
        RAhighp.append(peakRATJOA[i])
        dechighp.append(peakdecTJOA[i])
        Smaxhigh.append(peakSmaxTJOA[i])
    elif peakRATJOA[i] < 2.0:
        RAlowp.append(peakRATJOA[i])
        declowp.append(peakdecTJOA[i])
        Smaxlow.append(peakSmaxTJOA[i])

def intnumdensflux(Slow,Sp):
    k = 0
```

```
    for i in range(len(Sp)):
        if Sp[i] >= Slow:
            k = k + 1
        else:
            print("denied")
    return k

def difnumdensflux(Slow,Supp,Sp):
    k = 0
    for i in range(len(Sp)):
        if Sp[i] >= Slow and Sp[i] <= Supp:
            k = k + 1
        else:
            print("denied")
    return k

###Computing densities for both fields and subsequently computing the total estimated
    amount of sources of this calibre.
RAlowhigh = 17 ; RAupphigh = 18 ; declowhigh = 55 ; decupphigh = 60
RAlowlow = 1 ; RAupplow = 2 ; declowlow = 35 ; decupplow = 45

areahigh = areacalc(RAlowhigh,RAupphigh,declowhigh,decupphigh)
arealow = areacalc(RAlowlow,RAupplow,declowlow,decupplow)
denshigh = len(RAhigh)/areahigh
denslow = len(RAlow)/arealow
densav = (denshigh+denslow)/2
totalsources = 4*np.pi*densav

print("Area of high RA field is:",areahigh,"steradians")
print("Area of low RA field is:",arealow,"steradians")
print("Source density in high RA field is:",denshigh,"per steradian")
print("Source density in low RA field is:",denslow,"per steradian")
print("Source density average is:",densav,"per steradian")
print("So total amount of observable sources in the sky with flux at 5 GHz of ~50 mJy
    should be:",totalsources)

###Computing number density per steradian as a function of peak flux (integrated method)
    .
Slow1 = np.asarray([20.0,40.0,60.0,80.0])
nsrinthigh = []
nsrintlow = []
for i in range(len(Slow1)):
    x1 = intnumdensflux(Slow1[i],Smaxhigh)
    x2 = intnumdensflux(Slow1[i],Smaxlow)
    nsrinthigh.append(x1)
    nsrintlow.append(x2)
nsrinthigh = np.asarray(nsrinthigh)
nsrintlow = np.asarray(nsrintlow)
densSinthigh = nsrinthigh/areahigh
densSintlow = nsrintlow/arealow
densSintav = (densSinthigh+densSintlow)/2
totalSintsources = 4*np.pi*densSintav
projSintsources = [totalSintsources[0]-totalSintsources[1],totalSintsources[1]-
    totalSintsources[2],totalSintsources[2]-totalSintsources[3],totalSintsources[3]]

###Computing number density-peak flux convolution per steradian as a function of peak
    flux (differential method).
Smed = np.asarray([30.0,60.0])
Sbinsize = np.asarray([10.0,20.0])
Slow = Smed-Sbinsize
Supp = Smed+Sbinsize
nsrdifhigh = []
nsrdiflow = []
for i in range(len(Smed)):
    x1 = difnumdensflux(Slow[i],Supp[i],Smaxhigh)
    x2 = difnumdensflux(Slow[i],Supp[i],Smaxlow)
    nsrdifhigh.append(x1)
    nsrdiflow.append(x2)
nsrdifhigh = np.asarray(nsrdifhigh)
nsrdiflow = np.asarray(nsrdiflow)
```

```
nsrdifav = nsrdifhigh+nsrdiflow
densSdifhigh = nsrdifhigh/areahigh
densSdiflow = nsrdiflow/arealow
densSdifav = nsrdifav/(areahigh+arealow)
projSdifsources = 4*np.pi*densSdifav

print(nsrdiflow)
print(nsrdifhigh)

###Snellen data.
SmedSNEL = np.asarray([75,150,300])
plotSpeakSNEL = np.asarray([1.05089459,2.37790302,5.88500972])
SbinsizeSNEL = np.asarray([25.0,50.0,100.0])
plotSerrSNEL = np.asarray([0.33232205,0.84071567,2.2243246])

###WENSS data.
SmedVRIES = np.asarray([4000])
plotSpeakVRIES = np.asarray([22.0])
SbinsizeVRIES = np.asarray([2000.0])
plotSerrVRIES = np.asarray([10.0])

###Computing 3d positions on sky for star orb.
p3dTJOA = hidimpos(-posTJOA[0],posTJOA[1])

###Computing 3d positions of predicted sources (locations are not actually physical but
    the density is).
Phi = 2*np.pi*npr.random(size=totalsources)/15*180/np.pi
Theta = np.arccos(2*npr.random(size=totalsources)-1)*180/np.pi-90
p3dPRED = hidimpos(-Phi,Theta)

Phi0 = 2*np.pi*npr.random(size=totalsources-totalSintsources[0])/15*180/np.pi
Theta0 = np.arccos(2*npr.random(size=totalsources-totalSintsources[0])-1)*180/np.pi-90
p3dPRED0 = hidimpos(-Phi0,Theta0)

Phi2 = 2*np.pi*npr.random(size=projSintsources[0])/15*180/np.pi
Theta2 = np.arccos(2*npr.random(size=projSintsources[0])-1)*180/np.pi-90
p3dPRED2 = hidimpos(-Phi2,Theta2)

Phi4 = 2*np.pi*npr.random(size=projSintsources[1])/15*180/np.pi
Theta4 = np.arccos(2*npr.random(size=projSintsources[1])-1)*180/np.pi-90
p3dPRED4 = hidimpos(-Phi4,Theta4)

Phi6 = 2*np.pi*npr.random(size=projSintsources[2])/15*180/np.pi
Theta6 = np.arccos(2*npr.random(size=projSintsources[2])-1)*180/np.pi-90
p3dPRED6 = hidimpos(-Phi6,Theta6)

Phi8 = 2*np.pi*npr.random(size=projSintsources[3])/15*180/np.pi
Theta8 = np.arccos(2*npr.random(size=projSintsources[3])-1)*180/np.pi-90
p3dPRED8 = hidimpos(-Phi8,Theta8)

dndS = densSdifav/(Supp/1000-Slow/1000)
plotSpeak = dndS*(Smed/1000)**(5/2)
plotSerr = (np.sqrt(nsrdifav)/nsrdifav)*plotSpeak
print(plotSpeak)
print(plotSerr)

def fitfun(S,a,b,c):
    x = np.log10(S)
    return 10**(a + b*x + c*x**2)

Sarr = np.arange(10,10000,0.1)
Slst = np.concatenate((Smed,SmedSNEL,SmedVRIES))
peaklst = np.concatenate((plotSpeak,plotSpeakSNEL,plotSpeakVRIES))
peakerr = np.concatenate((plotSerr,plotSerrSNEL,plotSerrVRIES))
p,pcov = spo.curve_fit(fitfun,Slst,peaklst,sigma=peakerr,**{'maxfev':1000000})
print(p)
print(pcov)
function = fitfun(Sarr,*p)

#####Synthesis III: Revolutions. Plotting the results.
```

```
    ##################################################################
print(">>> Synthesis III: Revolutions")

###Plot no.1: number of sources per area.
fig1 = figure(figsize=(40/2.54, 40/2.54))
frame1 = fig1.add_subplot(1,1,1)
frame1.set_title("Source position in sky",fontsize=15)
frame1.set_xlim([RAlowlow,RAupphigh])
frame1.set_ylim([declowlow,decupphigh])
frame1.axvline(x=RAlowlow,linewidth=1,color='b') ; frame1.axvline(x=RAupplow,linewidth
    =1,color='b')
frame1.axhline(y=declowlow,linewidth=1,color='b') ; frame1.axhline(y=decupplow,linewidth
    =1,color='b')
frame1.axvline(x=RAlowhigh,linewidth=1,color='r') ; frame1.axvline(x=RAupphigh,linewidth
    =1,color='r')
frame1.axhline(y=declowhigh,linewidth=1,color='r') ; frame1.axhline(y=decupphigh,
    linewidth=1,color='r')
frame1.plot(posTJOA[0],posTJOA[1],'y*',ms=11)
frame1.set_xlabel('Right Ascension (hours) ')
frame1.set_ylabel('Declination (deg)')
frame1.invert_xaxis()

###Plot no.2: number of sources per area, closeup low RA.
fig2 = figure(figsize=(40/2.54, 40/2.54))
frame2 = fig2.add_subplot(1,1,1)
frame2.set_title("Source position in sky",fontsize=15)
frame2.set_xlim([RAlowlow,RAupplow])
frame2.set_ylim([declowlow,decupplow])
frame2.plot(posTJOA[0],posTJOA[1],'y*',ms=15)
frame2.plot(peakRATJOA,peakdecTJOA,'c*',ms=15)
frame2.plot(Mirachpos[0],Mirachpos[1],'r*',ms=20)
frame2.annotate("Mirach (Beta Andromedae)",xy=(Mirachpos[0]+0.05,Mirachpos[1]+0.1))
frame2.set_xlabel('Right Ascension (hours) ')
frame2.set_ylabel('Declination (deg)')
frame2.invert_xaxis()

###Plot no.3: number of sources per area, closeup high RA.
fig3 = figure(figsize=(40/2.54, 40/2.54))
frame3 = fig3.add_subplot(1,1,1)
frame3.set_title("Source position in sky",fontsize=15)
frame3.set_xlim([RAlowhigh,RAupphigh])
frame3.set_ylim([declowhigh,decupphigh])
frame3.plot(posTJOA[0],posTJOA[1],'y*',ms=15)
frame3.plot(peakRATJOA,peakdecTJOA,'c*',ms=15)
frame3.plot(Grumiumpos[0],Grumiumpos[1],'b*',ms=20)
frame3.annotate("Grumium (Xi Draconis)",xy=(Grumiumpos[0]+0.05,Grumiumpos[1]-0.2))
frame3.set_xlabel('Right Ascension (hours) ')
frame3.set_ylabel('Declination (deg)')
frame3.invert_xaxis()

###Plot no.4: sources both real and predicted projected on sphere, assuming density
    found from McKean data.
fig4 = figure(figsize=(40/2.54, 40/2.54))
frame4 = fig4.add_subplot(111, projection='3d')
frame4.plot(p3dPRED[0],p3dPRED[1],p3dPRED[2],'y*',markersize=8)
frame4.plot(p3dTJOA[0],p3dTJOA[1],p3dTJOA[2],'r*',markersize=12)
frame4.set_title('Isotropic predicted spherical distribution of GPS sources')
frame4.set_xlabel("x")
frame4.set_ylabel("y")
frame4.set_zlabel("z")

###Plot no.5: sources both real and predicted projected on plane.
fig5 = figure(figsize=(40/2.54, 40/2.54))
frame5 = fig5.add_subplot(1, 1, 1)
frame5.plot(Phi,Theta,'y*',markersize=8,label="Predicted source density")
frame5.plot(posTJOA[0],posTJOA[1],'r*',markersize=12,label="Our data")
frame5.set_title('Isotropic predicted distribution of GPS sources over flattened sky')
frame5.set_xlabel("Right Ascension (hours)")
frame5.set_ylabel("Declination (degrees)")
frame5.invert_xaxis()
```

```
321
322 ###Plot no.6: number density as a function of peak flux.
323 fig6 = figure(figsize=(40/2.54, 40/2.54))
324 frame6 = fig6.add_subplot(1, 1, 1)
325 frame6.set_xlim([0,100])
326 frame6.plot(Slow1,densSintav,'r*',markersize=15,label="averaged sample")
327 frame6.set_title('cumulative number density as a function of peak flux')
328 frame6.set_xlabel("Smax (mJy)")
329 frame6.set_ylabel("number density with Smax in excess of S-value")
330 frame6.legend()
331
332 ###Plot no.7: number as a function of peak flux.
333 fig7 = figure(figsize=(40/2.54, 40/2.54))
334 frame7 = fig7.add_subplot(1, 1, 1)
335 frame7.set_xlim([0,100])
336 frame7.plot(Slow1,totalSintsources,'r*',markersize=15,label="predicted sources")
337 frame7.set_title('predicted cumulative number as a function of peak flux')
338 frame7.set_xlabel("Smax (mJy)")
339 frame7.set_ylabel("number with Smax in excess of S-value")
340 frame7.legend()
341
342 ###Plot no.8: predicted sources separated by peak flux population projected on sphere,
      assuming density found from McKean data.
343 fig8 = figure(figsize=(40/2.54, 40/2.54))
344 frame8 = fig8.add_subplot(111, projection='3d')
345 #frame8.plot(p3dPRED0[0],p3dPRED0[1],p3dPRED0[2],'m*',markeredgewidth=0,markersize=5,
      label='No peak',alpha=0.5)
346 frame8.plot(p3dPRED2[0],p3dPRED2[1],p3dPRED2[2],'r*',markeredgewidth=0,markersize=7,
      label='20-40 mJy')
347 frame8.plot(p3dPRED4[0],p3dPRED4[1],p3dPRED4[2],'y*',markeredgewidth=0,markersize=7,
      label='40-60 mJy')
348 frame8.plot(p3dPRED6[0],p3dPRED6[1],p3dPRED6[2],'c*',markeredgewidth=0,markersize=7,
      label='60-80 mJy')
349 frame8.plot(p3dPRED8[0],p3dPRED8[1],p3dPRED8[2],'b*',markeredgewidth=0,markersize=7,
      label='>80 mJy')
350 frame8.set_title('Isotropic predicted spherical distribution of GPS sources')
351 frame8.set_xlabel("x")
352 frame8.set_ylabel("y")
353 frame8.set_zlabel("z")
354 frame8.legend()
355
356 ###Plot no.9: predicted sources projected on plane.
357 fig9 = figure(figsize=(40/2.54, 40/2.54))
358 frame9 = fig9.add_subplot(1, 1, 1)
359 #frame9.plot(Phi0,Theta0,'m*',markeredgewidth=0,markersize=5,label='No peak',alpha=0.5)
360 frame9.plot(Phi2,Theta2,'r*',markeredgewidth=0,markersize=7,label='20-40 mJy')
361 frame9.plot(Phi4,Theta4,'y*',markeredgewidth=0,markersize=7,label='40-60 mJy')
362 frame9.plot(Phi6,Theta6,'c*',markeredgewidth=0,markersize=7,label='60-80 mJy')
363 frame9.plot(Phi8,Theta8,'b*',markeredgewidth=0,markersize=7,label='>80 mJy')
364 frame9.set_title('Isotropic predicted distribution of GPS sources over flattened sky')
365 frame9.set_xlabel("Right Ascension (hours)")
366 frame9.set_ylabel("Declination (degrees)")
367 frame9.legend()
368 frame9.invert_xaxis()
369
370
371 show()
```

B.5 Number count calculator

The following script is used to obtain number counts.

Scripts/Thesis_synthesis4_v1-2.py

```
1 #!/usr/bin/env python
2 from __future__ import division
3 import numpy as np
4 import scipy as sp
5 import scipy.stats as sps
```

```
import fnmatch as fm
import numpy.random as npr
from matplotlib.pyplot import figure, show
###THIS SCRIPT COMPUTES STD IN LOG10(z) AND z.


#####Synthesis I: Reading Data.
    ################################################################################
print(">>> Synthesis I")

###McKean's & Tjoa's data (K,z)
znamesTJOA = np.genfromtxt("K,z-data-fulltable.dat",dtype=str,usecols=[0],comments="#")
zdatTJOA = np.genfromtxt("K,z-data-fulltable.dat",usecols=[1],comments="#")
zstdlowTJOA = np.genfromtxt("K,z-data-fulltable.dat",usecols=[2],comments="#")
zstduppTJOA = np.genfromtxt("K,z-data-fulltable.dat",usecols=[3],comments="#")
KdatTJOA = np.genfromtxt("K,z-data-fulltable.dat",usecols=[4],comments="#")
KstdTJOA = np.genfromtxt("K,z-data-fulltable.dat",usecols=[5],comments="#")

purenamesTJOA = np.genfromtxt("ThesisTable_synchrotronNEW.dat",dtype=str,usecols=[1],
    comments="#")
combinamesTJOA = np.genfromtxt("ThesisTable_synch-powlawNEW.dat",dtype=str,usecols=[1],
    comments="#")
purevmaxTJOA = np.genfromtxt("ThesisTable_synchrotronNEW.dat",dtype=float,usecols=[10],
    comments="#")
combivmaxTJOA = np.genfromtxt("ThesisTable_synch-powlawNEW.dat",dtype=float,usecols
    =[14],comments="#")
vmaxTJOA = np.concatenate((purevmaxTJOA,combivmaxTJOA))
pureSmaxTJOA = np.genfromtxt("ThesisTable_synchrotronNEW.dat",dtype=float,usecols=[12],
    comments="#")
combiSmaxTJOA = np.genfromtxt("ThesisTable_synch-powlawNEW.dat",dtype=float,usecols
    =[16],comments="#")
SmaxTJOA = np.concatenate((pureSmaxTJOA,combiSmaxTJOA))


#####Synthesis II: Reloaded. Computation of results.
    ###################################################################
print(">>> Synthesis II: Reloaded")

def Kzrel(z):                                   ###K,z-relation function
    return 17.37 + 4.53*np.log10(z) + 0.31*(np.log10(z))**2

###Generating plain, idealized K-log10(z) relation.
zarr = np.linspace(0.05,5,10000)
Karr = Kzrel(zarr)

znamesTJOA2 = []
zdatTJOA2 = []
zstdlowTJOA2 = []
zstduppTJOA2 = []
KdatTJOA2 = []
KstdTJOA2 = []
for i in range(len(zdatTJOA)):
    if zdatTJOA[i] == 0.0:
        print("This point has a nonexistent redshift.")
    else:
        znamesTJOA2.append(znamesTJOA[i])
        zdatTJOA2.append(zdatTJOA[i])
        zstdlowTJOA2.append(zstdlowTJOA[i])
        zstduppTJOA2.append(zstduppTJOA[i])
        KdatTJOA2.append(KdatTJOA[i])
        KstdTJOA2.append(KstdTJOA[i])
znamesTJOA2 = np.asarray(znamesTJOA2)
zdatTJOA2 = np.asarray(zdatTJOA2)
zstdlowTJOA2 = np.asarray(zstdlowTJOA2)
zstduppTJOA2 = np.asarray(zstduppTJOA2)
KdatTJOA2 = np.asarray(KdatTJOA2)
KstdTJOA2 = np.asarray(KstdTJOA2)

znamesTJOApure = []
zdatTJOApure = []
```

```
zstdlowTJOApure = []
zstduppTJOApure = []
KdatTJOApure = []
KstdTJOApure = []
for i in range(len(purenamesTJOA)):
    for j in range(len(znamesTJOA2)):
        if fm.fnmatch(purenamesTJOA[i],"{}".format(znamesTJOA2[j])) is True:  ###Check whether the chosen name in z-data matches any known pure spectral peak name
            znamesTJOApure.append(znamesTJOA2[j])                              ###If so, append indices to specnames- and Knames lists.
            zdatTJOApure.append(zdatTJOA2[j])
            zstdlowTJOApure.append(zstdlowTJOA2[j])
            zstduppTJOApure.append(zstduppTJOA2[j])
            KdatTJOApure.append(KdatTJOA2[j])
            KstdTJOApure.append(KstdTJOA2[j])
znamesTJOApure = np.asarray(znamesTJOApure)
zdatTJOApure = np.asarray(zdatTJOApure)
zstdlowTJOApure = np.asarray(zstdlowTJOApure)
zstduppTJOApure = np.asarray(zstduppTJOApure)
KdatTJOApure = np.asarray(KdatTJOApure)
KstdTJOApure = np.asarray(KstdTJOApure)

znamesTJOAcombi = []
zdatTJOAcombi = []
zstdlowTJOAcombi = []
zstduppTJOAcombi = []
KdatTJOAcombi = []
KstdTJOAcombi = []
for i in range(len(combinamesTJOA)):
    for j in range(len(znamesTJOA2)):
        if fm.fnmatch(combinamesTJOA[i],"{}".format(znamesTJOA2[j])) is True: ###Check whether the chosen name in z-data matches any known pure spectral peak name
            znamesTJOAcombi.append(znamesTJOA2[j])                             ###If so, append indices to specnames- and Knames lists.
            zdatTJOAcombi.append(zdatTJOA2[j])
            zstdlowTJOAcombi.append(zstdlowTJOA2[j])
            zstduppTJOAcombi.append(zstduppTJOA2[j])
            KdatTJOAcombi.append(KdatTJOA2[j])
            KstdTJOAcombi.append(KstdTJOA2[j])
znamesTJOAcombi = np.asarray(znamesTJOAcombi)
zdatTJOAcombi = np.asarray(zdatTJOAcombi)
zstdlowTJOAcombi = np.asarray(zstdlowTJOAcombi)
zstduppTJOAcombi = np.asarray(zstduppTJOAcombi)
KdatTJOAcombi = np.asarray(KdatTJOAcombi)
KstdTJOAcombi = np.asarray(KstdTJOAcombi)


#####Synthesis III: Revolutions. Plotting the results.
    ################################################################
print(">>> Synthesis III: Revolutions")

###Plot no.1: K-band magnitude versus z.
fig1 = figure(figsize=(40/2.54, 40/2.54))
frame1 = fig1.add_subplot(1,1,1)
frame1.set_title("K-band magnitude vs z",fontsize=20)
frame1.plot(zarr,Karr,"k",label="Idealized K,z-relation")
assymerrzTJOA2 = [zstdlowTJOA2,zstduppTJOA2]
frame1.errorbar(zdatTJOA2,KdatTJOA2,xerr=assymerrzTJOA2,yerr=KstdTJOA2,ecolor="k",
    elinewidth=1,marker="*",mfc="y",ms=12,ls="",label="Our data points (K,z)")
frame1.set_xlabel('Redshift z',fontsize=20)
frame1.set_ylabel('K-band magnitude',fontsize=20)
frame1.tick_params(labelsize=20)

###Plot no.2: number of sources per redshift bin (pure and combined separately).
fig2 = figure(figsize=(40/2.54, 40/2.54))
frame2 = fig2.add_subplot(1,1,1)
frame2.set_title("number of sources per $z$",fontsize=20)
frame2.hist(zdatTJOApure,bins=[0.0,0.5,1.0,1.5,2.0,2.5,3.0],label='GPS',alpha=0.5)
frame2.hist(zdatTJOAcombi,bins=[0.0,0.5,1.0,1.5,2.0,2.5,3.0],label='GPS + power law',
    alpha=0.5)
```

```
#for i in zdatTJOApure:
#    frame2.axvline(x=i,linewidth=2,color='y')
#for i in zdatTJOAcombi:
#    frame2.axvline(x=i,linewidth=2,color='g')
frame2.set_xlabel('Redshift $z$',fontsize=20)
frame2.set_ylabel('Number of sources',fontsize=20)
frame2.tick_params(labelsize=20)

###Plot no.3: number of sources per redshift bin (all of them).
fig3 = figure(figsize=(40/2.54, 40/2.54))
frame3 = fig3.add_subplot(1,1,1)
frame3.set_title("number of sources per $z$",fontsize=20)
frame3.hist(zdatTJOA2,bins=[0.0,0.5,1.0,1.5,2.0,2.5,3.0])
#for i in zdatTJOA2:
#    frame3.axvline(x=i,linewidth=1,color='r')
frame3.set_xlabel('Redshift $z$',fontsize=20)
frame3.set_ylabel('Number of sources',fontsize=20)
frame3.tick_params(labelsize=20)

###Plot no.4: number of sources per redshift bin (pure and combined separately).
fig4 = figure(figsize=(40/2.54, 40/2.54))
frame4 = fig4.add_subplot(1,1,1)
frame4.set_title("number of sources per z",fontsize=20)
frame4.hist(np.log10(zdatTJOApure),bins=[-2.0,-1.5,-1.0,-0.5,0.0,0.5,1.0,1.5,2.0],label=
    'GPS',alpha=0.5)
frame4.hist(np.log10(zdatTJOAcombi),bins=[-2.0,-1.5,-1.0,-0.5,0.0,0.5,1.0,1.5,2.0],label
    ='GPS + power law',alpha=0.5)
#for i in zdatTJOApure:
#    frame4.axvline(x=np.log10(i),linewidth=1,color='y')
#for i in zdatTJOAcombi:
#    frame4.axvline(x=np.log10(i),linewidth=1,color='g')
frame4.set_xlabel('Redshift log10z',fontsize=20)
frame4.set_ylabel('Number of sources',fontsize=20)
frame4.tick_params(labelsize=20)

###Plot no.5: number of sources per redshift bin (all of them).
fig5 = figure(figsize=(40/2.54, 40/2.54))
frame5 = fig5.add_subplot(1,1,1)
frame5.set_title("number of sources per z",fontsize=20)
frame5.hist(np.log10(zdatTJOA2),bins=[-2.0,-1.5,-1.0,-0.5,0.0,0.5,1.0,1.5,2.0])
#for i in zdatTJOA2:
#    frame5.axvline(x=np.log10(i),linewidth=1,color='r')
frame5.set_xlabel('Redshift log10z',fontsize=20)
frame5.set_ylabel('Number of sources',fontsize=20)
frame5.tick_params(labelsize=20)

###Plot no.6: number of sources per vmax bin (pure and combined separately).
fig6 = figure(figsize=(40/2.54, 40/2.54))
frame6 = fig6.add_subplot(1,1,1)
frame6.set_title("number of sources per vmax",fontsize=20)
frame6.hist(purevmaxTJOA,bins=[0.0,2.0,4.0,6.0,8.0,10.0,12.0,14.0,16.0,18.0,20.0],label=
    'GPS',alpha=0.5)
frame6.hist(combivmaxTJOA,bins=[0.0,2.0,4.0,6.0,8.0,10.0,12.0,14.0,16.0,18.0,20.0],label
    ='GPS + power law',alpha=0.5)
#for i in purevmaxTJOA:
#    frame6.axvline(x=i,linewidth=2,color='y')
#for i in combivmaxTJOA:
#    frame6.axvline(x=i,linewidth=2,color='g')
frame6.set_xlabel('Peak frequency vmax (GHz)',fontsize=20)
frame6.set_ylabel('Number of sources',fontsize=20)
frame6.tick_params(labelsize=20)

###Plot no.7: number of sources per vmax bin (all of them).
fig7 = figure(figsize=(40/2.54, 40/2.54))
frame7 = fig7.add_subplot(1,1,1)
frame7.set_title("number of sources per vmax",fontsize=20)
frame7.hist(vmaxTJOA,bins=[0.0,2.0,4.0,6.0,8.0,10.0,12.0,14.0,16.0,18.0,20.0])
#for i in vmaxTJOA:
#    frame7.axvline(x=i,linewidth=2,color='r')
frame7.set_xlabel('Peak frequency vmax (GHz)',fontsize=20)
```

```
frame7.set_ylabel('Number of sources',fontsize=20)
frame7.tick_params(labelsize=20)

###Plot no.8: number of sources per Smax bin (pure and combined separately).
fig8 = figure(figsize=(40/2.54, 40/2.54))
frame8 = fig8.add_subplot(1,1,1)
frame8.set_title("number of sources per Smax",fontsize=20)
frame8.hist(pureSmaxTJOA,bins=[20.0,30.0,40.0,50.0,60.0,70.0,80.0,90.0],label='GPS',
    alpha=0.5)
frame8.hist(combiSmaxTJOA,bins=[20.0,30.0,40.0,50.0,60.0,70.0,80.0,90.0],label='GPS +
    power law',alpha=0.5)
#for i in pureSmaxTJOA:
#    frame8.axvline(x=i,linewidth=2,color='y')
#for i in combiSmaxTJOA:
#    frame8.axvline(x=i,linewidth=2,color='g')
frame8.set_xlabel('Peak flux Smax (mJy)',fontsize=20)
frame8.set_ylabel('Number of sources',fontsize=20)
frame8.tick_params(labelsize=20)

###Plot no.9: number of sources per Smax bin (all of them).
fig9 = figure(figsize=(40/2.54, 40/2.54))
frame9 = fig9.add_subplot(1,1,1)
frame9.set_title("number of sources per Smax",fontsize=20)
frame9.hist(SmaxTJOA,bins=[20.0,30.0,40.0,50.0,60.0,70.0,80.0,90.0])
#for i in SmaxTJOA:
#    frame9.axvline(x=i,linewidth=2,color='r')
frame9.set_xlabel('Peak flux Smax (mJy)',fontsize=20)
frame9.set_ylabel('Number of sources',fontsize=20)
frame9.tick_params(labelsize=20)

show()
```

B.6 Size & distance cosmological calculator

The following script is used to synthesize results to obtain luminosity distances and angular & linear sizes.

Scripts/Thesis_synthesis6_v2–3.py

```
#!/usr/bin/env python
from __future__ import division
import numpy as np
import scipy.optimize as spo
import fnmatch as fm
from cosmodistfun import cosmocalc
from matplotlib.pyplot import figure, show
###THIS SCRIPT COMPUTES STD IN LOG10(z) AND z.


#####Synthesis I: Reading Data.
    ################################################################################
print(">>> Synthesis I")

###Reading relevant data from datafiles: McKean&Tjoa data.
znames = np.genfromtxt("K,z-data-ztable.dat",dtype=str,usecols=[0],comments="#")
zdat,zstdlow,zstdupp,Kdat,Kstd = np.genfromtxt("K,z-data-ztable.dat",usecols
    =[1,2,3,4,5],unpack=True,comments="#")

spec1names = np.genfromtxt("ThesisTable_synchrotronNEW.dat",dtype=str,usecols=[1],
    comments="#")
vmax1dat,vmax1std,Smax1dat,Smax1std = np.genfromtxt("ThesisTable_synchrotronNEW.dat",
    dtype=float,usecols=[10,11,12,13],unpack=True,comments="#")
spec2names = np.genfromtxt("ThesisTable_synch-powlawNEW.dat",dtype=str,usecols=[1],
    comments="#")
vmax2dat,vmax2std,Smax2dat,Smax2std = np.genfromtxt("ThesisTable_synch-powlawNEW.dat",
    dtype=float,usecols=[14,15,16,17],unpack=True,comments="#")

###Reading relevant data from datafiles: Snellen faint dataset.
```

```
specnamesSNELf = np.genfromtxt("Snellensample-faint.dat",dtype=str,usecols=[0],comments="#")
zdatSNELf,vmaxdatSNELf,SmaxdatSNELf,angmarSNELf = np.genfromtxt("Snellensample-faint.dat",dtype=float,usecols=[1,2,3,5],unpack=True,comments="#")

###Reading relevant data from datafiles: Snellen bright dataset.
specnamesSNELb = np.genfromtxt("Snellensample-bright.dat",dtype=str,usecols=[0],comments="#")
zdatSNELb,vmaxdatSNELb,SmaxdatSNELb,angmarSNELb = np.genfromtxt("Snellensample-bright.dat",dtype=float,usecols=[1,2,3,5],unpack=True,comments="#")
SmaxdatSNELb = SmaxdatSNELb*1000

###Reading relevant data from datafiles: Snellen CSS dataset.
specnamesSNELc = np.genfromtxt("Snellensample-CSS.dat",dtype=str,usecols=[0],comments="#")
zdatSNELc,vmaxdatSNELc,SmaxdatSNELc,angmarSNELc = np.genfromtxt("Snellensample-CSS.dat",dtype=float,usecols=[1,2,3,5],unpack=True,comments="#")
SmaxdatSNELc = SmaxdatSNELc*1000
angmarSNELc = angmarSNELc*1000

###Concatenating arrays.
typelst = []
for i in range(len(vmax1dat)):
    typelst.append("GPS")
for i in range(len(vmax2dat)):
    typelst.append("GPPL")

vmaxdat = np.concatenate((vmax1dat,vmax2dat)) ; vmaxstd = np.concatenate((vmax1std,vmax2std))                                    ###Concatenating vmax data and std's
Smaxdat = np.concatenate((Smax1dat,Smax2dat)) ; Smaxstd = np.concatenate((Smax1std,Smax2std))                                    ###Concatenating Smax data and std's

specnames = []                  ###Creating spectral names array
specnames.extend(spec1names)    ###Appending pure synch names
specnames.extend(spec2names)    ###Appending synch+powlaw names

###Loop to create an array with the peak data for those sources whose redshift is known: McKean&Tjoa data.
names = []                      ###Empty names list
Smaxlst = []                          ###Empty S-peak list
vmaxlst = []                          ###Empty v-peak list
Smaxstdlst = []                       ###Empty S-peak std list
vmaxstdlst = []                       ###Empty v-peak std list
zlst = []                       ###Empty z list
zstdlowlst = [] ; zstdupplst = []   ###Empty z lower- and upper boundaries lists
for i in range(len(specnames)):
    k = 0
    names.append(specnames[i])                          ###Appending name to name list
    Smaxlst.append(Smaxdat[i])                          ###Appending relevant entries from Smax-data
    vmaxlst.append(vmaxdat[i])                          ###Appending relevant entries from vmax-data
    Smaxstdlst.append(Smaxstd[i])               ###Appending relevant entries from Smax-std-data
    vmaxstdlst.append(vmaxstd[i])               ###Appending relevant entries from vmax-std-data
    for j in range(len(znames)):
        if fm.fnmatch(specnames[i],"{}".format(znames[j])) is True:          ###Check whether the chosen name in z-data matches any known spectral peak name
            zlst.append(zdat[j])                                             ###Appending relevant entries from z-data
            zstdlowlst.append(zstdlow[j]) ; zstdupplst.append(zstdupp[j])    ###Appending relevant entries from z lower- and upper boundaries-data
            k = k + 1
    print(specnames[i],k)
    if k == 0:
        zlst.append(1.0)
                    ###Appending relevant entries from z-data
```

```
75          zstdlowlst.append(0.26) ; zstdupplst.append(0.36)          ###Appending relevant
        entries from z lower- and upper boundaries-data
76      else:
77          print("no randomness for you!")
78
79  ###Converting lists to arrays: McKean&Tjoa data.
80  Smaxarr = np.asarray(Smaxlst)
81  vmaxarr = np.asarray(vmaxlst)
82  Smaxerr = np.asarray(Smaxstdlst)
83  vmaxerr = np.asarray(vmaxstdlst)
84  zarr = np.asarray(zlst)
85  zerrlow = np.asarray(zstdlowlst) ; zerrupp = np.asarray(zstdupplst)
86
87
88  #####Synthesis II: Reloaded. Computation of results.
        ##################################################################
89  print(">>> Synthesis II: Reloaded")
90
91  B = 50e-6 #Gauss  ###Estimate for magnetic field strength of source
92
93  def unredshiftinator(z,vmaxobs):    ###Function to unshift the redshifted peak
        frequencies
94      return (1+z)*vmaxobs
95
96  def angsize(S,z,v):                 ###Function to compute angular size in mas
97      S = S/1000
98      return ((8**(5/4))*(B**(1/4))*(S**(1/2))*((1+z)**(1/4)))/(v**(5/4))
99
100 def distance(z):                ###Function to compute distance in Mpc
101     return np.log(1+z)*299792.458/67.80
102
103 def linsize(theta,D):               ###Function to compute linear size in pc
104     theta = (((theta/1000)/60)/60)*np.pi/180
105     return 1000000*np.tan(theta)*D
106
107 def angerr(S,z,vobs,Serr,zerr,vobserr):   ###Uncertainty function for angular size.
108     C = (8**(5/4))*(B**(1/4))
109     dadS = C*((1+z)**(1/4))/(2*(vobs**(5/4))*(S**(1/2)))
110     dadz = C*(S**(1/2))/(4*(vobs**(5/4))*((1+z)**(3/4)))
111     dadvobs = -C*(5*(S**(1/2))*((1+z)**(1/4)))/(4*(vobs**(9/4)))
112     return np.sqrt((dadS*Serr)**2 + (dadz*zerr)**2 + (dadvobs*vobserr)**2)
113
114 def plotx(S,v):
115     return ((S/1000)**0.5)*(v**-1.25)
116
117 def plotxerr(S,v,Serr,verr):
118     dxdS = (0.5*(1000*S)**-0.5)*(v**-1.25)
119     dxdv = -1.25*((S/1000)**0.5)*(v**-2.25)
120     return np.sqrt((dxdS*Serr)**2 + (dxdv*verr)**2)
121
122 ###Computing relevant data arrays using previously defined functions: McKean&Tjoa data.
123 vmaxemit = unredshiftinator(zarr,vmaxarr)
124 angmar = angsize(Smaxarr,vmaxarr,zarr)
125 angerrlow = angerr(Smaxarr,zarr,vmaxarr,Smaxerr,zerrlow,vmaxerr)
126 for i in range(len(angerrlow)):
127     if angerrlow[i] > angmar[i]:
128         angerrlow[i] = angmar[i]
129 angerrupp = angerr(Smaxarr,zarr,vmaxarr,Smaxerr,zerrupp,vmaxerr)
130 lindist = np.zeros(len(angmar))
131 extdist = np.zeros(len(angmar))
132 for i in range(len(zarr)):
133     entry = cosmocalc(zarr[i])
134     lindist[i] = entry[1]
135     extdist[i] = entry[0]
136 linext = linsize(angmar,extdist)
137
138 ###Printing table of relevant statistics: McKean&Tjoa data.
139 print("\033[1m","Name      *    z    *  Lum. Distance  * Smax  * vmax  *vmax emit*ang.
        size*lin.ext. ","\033[0m")
140 print("\033[1m","          *         *       (Mpc)      * (mJy) * (GHz) *  (GHz)  * (mas)
```

```
      * (pc)","\033[0m")
for i in range(len(angmar)):
    print("{0}    {2}    {3}  {4}   {5}    {6}   {7}    {8}".format(names[i]," N/A ",np
    .round(zarr[i],3),np.round(lindist[i],1),np.round(Smaxarr[i],1),np.round(vmaxarr[i
    ],2),np.round(vmaxemit[i],2),np.round(angmar[i],2),np.round(angerrlow[i],2),np.round
    (angerrupp[i],2),np.round(linext[i],1)))

angmarGPS = []
angerrlowGPS = [] ; angerruppGPS = []
angmarGPPL = []
angerrlowGPPL = [] ; angerruppGPPL = []
for i in range(len(typelst)):
    if fm.fnmatch(typelst[i],"GPS") is True:
        angmarGPS.append(angmar[i])
        angerrlowGPS.append(angerrlow[i]) ; angerruppGPS.append(angerrupp[i])
    elif fm.fnmatch(typelst[i],"GPPL") is True:
        angmarGPPL.append(angmar[i])
        angerrlowGPPL.append(angerrlow[i]) ; angerruppGPPL.append(angerrupp[i])
angmarGPS = np.asarray(angmarGPS)
angerrlowGPS = np.asarray(angerrlowGPS) ; angerruppGPS = np.asarray(angerruppGPS)
angmarGPPL = np.asarray(angmarGPPL)
angerrlowGPPL = np.asarray(angerrlowGPPL) ; angerruppGPPL = np.asarray(angerruppGPPL)

###Creating the data to be plotted on the x-axis.
plotdatGPS = plotx(Smax1dat,vmax1dat)
plotdatGPPL = plotx(Smax2dat,vmax2dat)
plotdatSNELf = plotx(SmaxdatSNELf,vmaxdatSNELf)
plotdatSNELb = plotx(SmaxdatSNELb,vmaxdatSNELb)
plotdatSNELc = plotx(SmaxdatSNELc,vmaxdatSNELc)

###Uncertainties in our data.
plotstdGPS = plotxerr(Smax1dat,vmax1dat,Smax1std,vmax1std)
plotstdGPPL = plotxerr(Smax2dat,vmax2dat,Smax2std,vmax2std)


#####Synthesis III: Revolutions. Plotting the results.
    ##################################################################
print(">>> Synthesis III: Revolutions")

###Plot no.2: Peak flux versus angular size.
fig2 = figure(figsize=(40/2.54, 40/2.54))
frame2 = fig2.add_subplot(1,1,1,xscale='log',yscale='log')
frame2.set_title("Peak flux density vs angular size",fontsize=20)
assymerryGPS = [angerrlowGPS,angerruppGPS]
assymerryGPPL = [angerrlowGPPL,angerruppGPPL]
frame2.errorbar(plotdatGPS,angmarGPS,xerr=plotstdGPS,yerr=assymerryGPS,ecolor="k",
    elinewidth=1,marker="*",mfc="r",ms=12,ls="",label="McKean very faint GPS sample")
frame2.errorbar(plotdatGPPL,angmarGPPL,xerr=plotstdGPPL,yerr=assymerryGPPL,ecolor="k",
    elinewidth=1,marker="*",mfc="y",ms=12,ls="",label="McKean very faint GPS+PL sample")
frame2.errorbar(plotdatSNELf,angmarSNELf,ecolor="k",elinewidth=1,marker="*",mfc="g",ms
    =12,ls="",label="Snellen's faint GPS sample")
frame2.errorbar(plotdatSNELb,angmarSNELb,ecolor="k",elinewidth=1,marker="*",mfc="c",ms
    =12,ls="",label="Snellen's bright GPS sample")
frame2.errorbar(plotdatSNELc,angmarSNELc,ecolor="k",elinewidth=1,marker="*",mfc="b",ms
    =12,ls="",label="Snellen's CSS sample")
frame2.set_xlabel('$S_{max}^{0.5}*v_{max}^{-1.25}$ (Jy$^{0.5}$GHz$^{-1.25}$)',fontsize
    =16)
frame2.set_ylabel('Angular size (mas)',fontsize=16)
frame2.tick_params(labelsize=15)
#frame2.legend(loc=0)

show()
```

B.7 Differential number density calculator

The following script is used to synthesize results to obtain differential number densities.

Scripts/Thesis_synthesis8_v6–3.py

```
#!/usr/bin/env python
from __future__ import division
import numpy as np
import numpy.random as npr
import scipy.optimize as spo
import fnmatch as fm
from matplotlib.pyplot import figure, show
###THIS SCRIPT COMPUTES DIFFERENTIAL NUMBER COUNTS. SOMETHING IS AMISS!


#####Synthesis I: Reading Data.
    ##############################################################################################
print(">>> Synthesis I")

###Reading McKean&Tjoa's name and data (because it's largely monocolumnar, this takes
    some time and messy coding).
dat0TJOA = np.genfromtxt("poslist-McKean.dat",dtype=None,usecols=[0],comments="#")
dat1TJOA = np.genfromtxt("poslist-McKean.dat",dtype=None,usecols=[1],comments="#")
dat2TJOA = np.genfromtxt("poslist-McKean.dat",dtype=None,usecols=[2],comments="#")

namesTJOA,RAhrsTJOA,decdegTJOA = np.split(dat0TJOA,3)
namesTJOA = namesTJOA.astype(str).flatten()
RAhrsTJOA = np.asarray(RAhrsTJOA.astype(float).flatten())
decdegTJOA = np.asarray(decdegTJOA.astype(float).flatten())

poep1TJOA,RAminTJOA,decminTJOA = np.split(dat1TJOA,3)
RAminTJOA = np.asarray(RAminTJOA.astype(float).flatten())
decminTJOA = np.asarray(decminTJOA.astype(float).flatten())

poep2TJOA,RAsecTJOA,decsecTJOA = np.split(dat2TJOA,3)
RAsecTJOA = np.asarray(RAsecTJOA.astype(float).flatten())
decsecTJOA = np.asarray(decsecTJOA.astype(float).flatten())

###Reading McKean&Tjoa's synch and synch+powlaw data for spectral peaks.
purenamesTJOA = np.genfromtxt("ThesisTable_synchrotronNEW.dat",dtype=str,usecols=[1],
    comments="#")
combinamesTJOA = np.genfromtxt("ThesisTable_synch-powlawNEW.dat",dtype=str,usecols=[1],
    comments="#")
purevmaxTJOA = np.genfromtxt("ThesisTable_synchrotronNEW.dat",dtype=float,usecols=[10],
    comments="#")
combivmaxTJOA = np.genfromtxt("ThesisTable_synch-powlawNEW.dat",dtype=float,usecols
    =[14],comments="#")
vmaxTJOA = np.concatenate((purevmaxTJOA,combivmaxTJOA))
pureSmaxTJOA = np.genfromtxt("ThesisTable_synchrotronNEW.dat",dtype=float,usecols=[12],
    comments="#")
combiSmaxTJOA = np.genfromtxt("ThesisTable_synch-powlawNEW.dat",dtype=float,usecols
    =[16],comments="#")
SmaxTJOA = np.concatenate((pureSmaxTJOA,combiSmaxTJOA))

###Reading Snellen's name and data.
rawnamesSNEL = np.genfromtxt("Snellenfulltab.dat",dtype=str,usecols=[0],comments="#")
rawRAhrsSNEL,rawRAminSNEL,rawRAsecSNEL,rawdecdegSNEL,rawdecminSNEL,rawdecsecSNEL = np.
    genfromtxt("Snellenfulltab.dat",dtype=float,usecols=[1,2,3,4,5,6],unpack=True,
    comments="#")
Sgb5,Svla8,Snvss14 = np.genfromtxt("Snellenfulltab.dat",dtype=float,usecols=[9,13,16],
    unpack=True,comments="#")

###Selecting only Snellen data which satisfy McKean's criteria.
namesSNEL = []
RAhrsSNEL = [] ; RAminSNEL = [] ; RAsecSNEL = []
decdegSNEL = [] ; decminSNEL = [] ; decsecSNEL = []
k = len(rawnamesSNEL)
for i in range(len(Svla8)):
    print("8.4 GHz flux is:",Svla8[i])
    if Svla8[i] > 16.5:
        alpha = (np.log10(Sgb5[i])-np.log10(Snvss14[i]))/(np.log10(5.0)-np.log10(1.4))
        print(" alpha is:",alpha)
        if 10.0 > alpha > -0.55:
            k = k - 1
```

```
59                  namesSNEL.append(rawnamesSNEL[i])
60                  RAhrsSNEL.append(rawRAhrsSNEL[i])
61                  RAminSNEL.append(rawRAminSNEL[i])
62                  RAsecSNEL.append(rawRAsecSNEL[i])
63                  decdegSNEL.append(rawdecdegSNEL[i])
64                  decminSNEL.append(rawdecminSNEL[i])
65                  decsecSNEL.append(rawdecsecSNEL[i])
66              elif alpha == np.inf:
67                  print(rawnamesSNEL[i],"missed either its 5 or 1.4 GHz entry.")
68              else:
69                  print(rawnamesSNEL[i],"had a spectral index between 5 and 1.4 GHz steeper
        than -0.5.")
70          else:
71              print(rawnamesSNEL[i],"had a 8.6 GHz flux which was too low to register.")
72  print("Number of rejected entries:",k)
73  RAhrsSNEL = np.asarray(RAhrsSNEL) ; RAminSNEL = np.asarray(RAminSNEL) ; RAsecSNEL = np.
        asarray(RAsecSNEL)
74  decdegSNEL = np.asarray(decdegSNEL) ; decminSNEL = np.asarray(decminSNEL) ; decsecSNEL =
         np.asarray(decsecSNEL)
75
76  ###Reading Snellen's synch data for spectral peaks.
77  peaknamesSNEL = np.genfromtxt("Snellenpeaktab.dat",dtype=str,usecols=[0],comments="#")
78  vmaxSNEL = np.genfromtxt("Snellenpeaktab.dat",dtype=float,usecols=[1],comments="#")
79  SmaxSNELraw = np.genfromtxt("Snellenpeaktab.dat",dtype=float,usecols=[2],comments="#")
80
81
82  #####Synthesis II: Reloaded. Computation of results.
        ##################################################################
83  print(">>> Synthesis II: Reloaded")
84
85  def posmaker(RAhrs,RAmin,RAsec,decdeg,decmin,decsec): ###Function that computes the RA
        and dec positions in fractional hours and degrees respectively for plotting purposes
86      RApos = RAhrs + RAmin/60 + (RAsec/60)/60
87      decpos = decdeg + decmin/60 + (decsec/60)/60
88      return RApos,decpos
89
90  def areacalc(RAlow,RAupp,declow,decupp):
91      declow = declow*np.pi/180 ; decupp = decupp*np.pi/180
92      RAlow = 15*RAlow*np.pi/180 ; RAupp = 15*RAupp*np.pi/180
93      return (RAupp - RAlow)*(np.sin(decupp) - np.sin(declow))
94
95  def difnumdensflux(Slow,Supp,Sp):
96      k = 0
97      reject = 0
98      for i in range(len(Sp)):
99          if Sp[i] >= Slow and Sp[i] <= Supp:
100             k = k + 1
101     return k
102
103 ###Computing the positions of Tjoa&McKean's sources.
104 posTJOA = posmaker(RAhrsTJOA,RAminTJOA,RAsecTJOA,decdegTJOA,decminTJOA,decsecTJOA)
105
106 ###Computing the positions of Snellen's sources.
107 posSNEL = posmaker(RAhrsSNEL,RAminSNEL,RAsecSNEL,decdegSNEL,decminSNEL,decsecSNEL)
108
109 ###Computing positions of GPS samples, McKean&Tjoa.
110 pureRATJOA = []
111 puredecTJOA = []
112 for i in range(len(purenamesTJOA)):
113     for j in range(len(namesTJOA)):
114         if fm.fnmatch(purenamesTJOA[i],"{}".format(namesTJOA[j])) is True:
115             pureRATJOA.append(posTJOA[0][j])
116             puredecTJOA.append(posTJOA[1][j])
117 pureRATJOA = np.asarray(pureRATJOA)
118 puredecTJOA = np.asarray(puredecTJOA)
119 peaknamesTJOA = list(purenamesTJOA)
120
121 combiRATJOA = []
122 combidecTJOA = []
123 for i in range(len(combinamesTJOA)):
```

```
    for j in range(len(namesTJOA)):
        if fm.fnmatch(combinamesTJOA[i],"{}".format(namesTJOA[j])) is True:
            combiRATJOA.append(posTJOA[0][j])
            combidecTJOA.append(posTJOA[1][j])
combiRATJOA = np.asarray(combiRATJOA)
combidecTJOA = np.asarray(combidecTJOA)
peaknamesTJOA.extend(combinamesTJOA)

SmaxTJOA = np.concatenate((pureSmaxTJOA,combiSmaxTJOA))
peakRATJOA = np.concatenate((pureRATJOA,combiRATJOA))
peakdecTJOA = np.concatenate((puredecTJOA,combidecTJOA))

###Computing positions of GPS samples, Snellen.
SmaxSNEL = []
peakRASNEL = []
peakdecSNEL = []
for i in range(len(peaknamesSNEL)):
    for j in range(len(namesSNEL)):
        if fm.fnmatch(peaknamesSNEL[i],"{}".format(namesSNEL[j])) is True:
            SmaxSNEL.append(SmaxSNELraw[i])
            peakRASNEL.append(posSNEL[0][j])
            peakdecSNEL.append(posSNEL[1][j])
SmaxSNEL = np.asarray(SmaxSNEL)
peakRASNEL = np.asarray(peakRASNEL)
peakdecSNEL = np.asarray(peakdecSNEL)

###Loop that splits Tjoa&McKean's data (only peaked) into ists two component fields of
    view and computing areas.
RAlowhighTJOA = 17 ; RAupphighTJOA = 18 ; declowhighTJOA = 55 ; decupphighTJOA = 60
RAlowlowTJOA = 1 ; RAupplowTJOA = 2 ; declowlowTJOA = 35 ; decupplowTJOA = 45

RAhighpTJOA = [] ; dechighpTJOA = [] ; SmaxhighTJOA = []
RAlowpTJOA = [] ; declowpTJOA = [] ; SmaxlowTJOA = []
for i in range(len(peakRATJOA)):
    if peakRATJOA[i] > 2.0:
        RAhighpTJOA.append(peakRATJOA[i])
        dechighpTJOA.append(peakdecTJOA[i])
        SmaxhighTJOA.append(SmaxTJOA[i])
    elif peakRATJOA[i] < 2.0:
        RAlowpTJOA.append(peakRATJOA[i])
        declowpTJOA.append(peakdecTJOA[i])
        SmaxlowTJOA.append(SmaxTJOA[i])

areahighTJOA = areacalc(RAlowhighTJOA,RAupphighTJOA,declowhighTJOA,decupphighTJOA)
arealowTJOA = areacalc(RAlowlowTJOA,RAupplowTJOA,declowlowTJOA,decupplowTJOA)
print("Our high field:",areahighTJOA,"sr")
print("Our low field:",arealowTJOA,"sr")
print("Our full field:",areahighTJOA+arealowTJOA,"sr")

###Loop that splits Snellen's data (only peaked) into its two component fields of view
    and computing areas.
RAlowhighSNEL = 15 ; RAupphighSNEL = 20 ; declowhighSNEL = 58 ; decupphighSNEL = 75
RAlowlowSNEL = 4 ; RAupplowSNEL = 8.5 ; declowlowSNEL = 58 ; decupplowSNEL = 75

RAhighpSNEL = [] ; dechighpSNEL = [] ; SmaxhighSNEL = []
RAlowpSNEL = [] ; declowpSNEL = [] ; SmaxlowSNEL = []
for i in range(len(peakRASNEL)):
    if peakRASNEL[i] > 9.0:
        RAhighpSNEL.append(peakRASNEL[i])
        dechighpSNEL.append(peakdecSNEL[i])
        SmaxhighSNEL.append(SmaxSNEL[i])
    elif peakRASNEL[i] < 9.0:
        RAlowpSNEL.append(peakRASNEL[i])
        declowpSNEL.append(peakdecSNEL[i])
        SmaxlowSNEL.append(SmaxSNEL[i])

areahighSNEL = areacalc(RAlowhighSNEL,RAupphighSNEL,declowhighSNEL,decupphighSNEL)
arealowSNEL = areacalc(RAlowlowSNEL,RAupplowSNEL,declowlowSNEL,decupplowSNEL)
print("Snellen's high field:",areahighSNEL,"sr")
print("Snellen's low field:",arealowSNEL,"sr")
```

```
print("Snellen's full field:",areahighSNEL+arealowSNEL,"sr")

Smed = np.asarray([30,60,120,240,480])
###Computing number density-peak flux convolution per steradian as a function of peak flux (differential method).
def peakconvo(Smaxhigh,Smaxlow,areatot):
    Sbinsize = Smed/3          ###Binsize for number counts
    Slow = (2/3)*Smed          ###Lower bound of bin
    Supp = (4/3)*Smed          ###Upper bound of bin

    nsrdifhigh = []
    nsrdiflow = []
    for i in range(len(Smed)):      ###Loop that uses function to differentially count number of sources per bin
        nsrdifhigh.append(difnumdensflux(Slow[i],Supp[i],Smaxhigh))
        nsrdiflow.append(difnumdensflux(Slow[i],Supp[i],Smaxlow))
    nsrdifhigh = np.asarray(nsrdifhigh)   ###Converting counts in high field to array
    nsrdiflow = np.asarray(nsrdiflow)     ###Converting counts in low field to array

    nsrdiftot = nsrdifhigh+nsrdiflow      ###Calculating total amount of sources per bin

    densdiftot = nsrdiftot/areatot        ###Computing source density per bin in total

    dndS = nsrdiftot/((Supp-Slow)/1000)                ###dn/dS -> density over binwidth
    plotypeak = (dndS/areatot)*((Smed/1000)**(5/2))    ###Plotted values on y-axis
    plotyerr = (np.sqrt(nsrdiftot)/nsrdiftot)*plotypeak     ###Poissonian uncertainties in y-axis values
    ###    0        1         2          3         4
    return Sbinsize,nsrdiftot,densdiftot,plotypeak,plotyerr

###McKean&Tjoa data.
TJOA = peakconvo(SmaxhighTJOA,SmaxlowTJOA,areahighTJOA+arealowTJOA)
SbinsizeTJOA = TJOA[0]
plotypeakTJOA = TJOA[3]
plotyerrTJOA = TJOA[4]
plotypeak2TJOA = plotypeakTJOA/(Smed**(5/2))
plotyerr2TJOA = plotyerrTJOA/(Smed**(5/2))
print(TJOA[1])

###Snellen data.
SNEL = peakconvo(SmaxhighSNEL,SmaxlowSNEL,areahighSNEL+arealowSNEL)
SbinsizeSNEL = SNEL[0]
plotypeakSNEL = SNEL[3]
plotyerrSNEL = SNEL[4]
plotypeak2SNEL = plotypeakSNEL/(Smed**(5/2))
plotyerr2SNEL = plotyerrSNEL/(Smed**(5/2))
print(SNEL[1])

###Combined data
SbinsizeCOMB = SbinsizeTJOA
nsrCOMB = TJOA[1]+SNEL[1]
plotypeakCOMB = plotypeakSNEL + plotypeakTJOA
plotyerrCOMB = (np.sqrt(nsrCOMB)/nsrCOMB)*plotypeakCOMB
plotypeak2COMB = plotypeak2SNEL + plotypeak2TJOA
plotyerr2COMB = (np.sqrt(nsrCOMB)/nsrCOMB)*plotypeak2COMB
print(nsrCOMB)

###De Vries data.
SmedVRIES = 4000
SbinsizeVRIES = 2000.0
plotypeakVRIES = 22.0
plotyerrVRIES = 10.0
plotypeak2VRIES = plotypeakVRIES/(SmedVRIES**(5/2))
plotyerr2VRIES = plotyerrVRIES/(SmedVRIES**(5/2))

with open("ThesisTable_numdifdensTJOA2.dat", "a") as myfile:
    for i in range(len(nsrCOMB)):
        myfile.write("{0} & {1} & {2} & {5}$\pm${6} & {11}$\pm${12} \\\\".format(Smed[i],Smed[i]/3,TJOA[1][i],SNEL[1][i],nsrCOMB[i],np.round(plotypeakTJOA[i],3),np.round(plotyerrTJOA[i],3),np.round(plotypeakSNEL[i],3),np.round(plotyerrSNEL[i],3),np.round
```

```
    (plotypeakCOMB[i],3),np.round(plotyerrCOMB[i],3),np.round(plotypeak2TJOA[i],10),np.
    round(plotyerr2TJOA[i],10),np.round(plotypeak2SNEL[i],10),np.round(plotyerr2SNEL[i
    ],10),np.round(plotypeak2COMB[i],10),np.round(plotyerr2COMB[i],10)))
257         myfile.write("\n")
258
259
260 #####Synthesis III: Revolutions. Plotting the results.
    ##############################################################
261 print(">>> Synthesis III: Revolutions")
262
263 ###Plot no.0: number density/peak convolution as a function of peak flux.
264 fig0 = figure(figsize=(40/2.54, 40/2.54))
265 frame0 = fig0.add_subplot(1,1,1,xscale='log',yscale='log')
266 frame0.errorbar(Smed,plotypeakTJOA,xerr=SbinsizeTJOA,yerr=plotyerrTJOA,ecolor="k",
    elinewidth=2,marker="*",mfc="r",ms=30,ls="",label="McKean's very faint GPS sample")
267 frame0.errorbar(Smed,plotypeakSNEL,xerr=SbinsizeSNEL,yerr=plotyerrSNEL,ecolor="k",
    elinewidth=2,marker="*",mfc="y",ms=30,ls="",label="Snellen's GPS sample")
268 #frame0.errorbar(Smed,plotypeakCOMB,xerr=SbinsizeCOMB,yerr=plotyerrCOMB,ecolor="k",
    elinewidth=2,marker="*",mfc="g",ms=20,ls="",label="Total GPS sample")
269 frame0.errorbar(SmedVRIES,plotypeakVRIES,xerr=SbinsizeVRIES,yerr=plotyerrVRIES,ecolor="k
    ",elinewidth=2,marker="*",mfc="c",ms=20,ls="",label="De Vries' sample")
270 frame0.set_xlabel("$S_{max}$ (mJy)",fontsize=35)
271 frame0.set_ylabel("$dN(>S)/dS*S^{5/2}$ (Sr$^{-1}$Jy$^{3/2}$)",fontsize=35)
272 frame0.tick_params(labelsize=30)
273
274 ###Plot no.1: number density as a function of peak flux.
275 fig1 = figure(figsize=(40/2.54, 40/2.54))
276 frame1 = fig1.add_subplot(1,1,1,xscale='log',yscale='log')
277 frame1.errorbar(Smed,plotypeak2TJOA,xerr=SbinsizeTJOA,yerr=plotyerr2TJOA,ecolor="k",
    elinewidth=2,marker="*",mfc="r",ms=30,ls="",label="McKean's very faint GPS sample")
278 frame1.errorbar(Smed,plotypeak2SNEL,xerr=SbinsizeSNEL,yerr=plotyerr2SNEL,ecolor="k",
    elinewidth=2,marker="*",mfc="y",ms=30,ls="",label="Snellen's GPS sample")
279 #frame1.errorbar(Smed,plotypeak2COMB,xerr=SbinsizeCOMB,yerr=plotyerr2COMB,ecolor="k",
    elinewidth=2,marker="*",mfc="g",ms=20,ls="",label="Total GPS sample")
280 frame1.errorbar(SmedVRIES,plotypeak2VRIES,xerr=SbinsizeVRIES,yerr=plotyerr2VRIES,ecolor=
    "k",elinewidth=2,marker="*",mfc="c",ms=20,ls="",label="De Vries' sample")
281 frame1.set_xlabel("$S_{max}$ (mJy)",fontsize=35)
282 frame1.set_ylabel("$dN(>S)/dS$ (Sr$^{-1}$Jy$^{-1}$)",fontsize=35)
283 frame1.tick_params(labelsize=30)
284
285 show()
```

References

[Allen et al., 1962] Allen, L. R., Brown, R. H., and Palmer, H. P. (1962). An analysis of the angular sizes of radio sources. *Monthly Notices of the Royal Astronomical Society*, 125:57.

[Blake, 1970] Blake, G. M. (1970). Observations of Extragalactic Radio Sources Having Unusual Spectra. *Astrophysical Letters*, 6:201.

[Brooks et al., 2011] Brooks, S., Gelman, A., Jones, G., and Xiao-Li, M. (2011). *Handbook of Markov Chain Monte Carlo*. Chapman and Hall/CRC.

[Condon et al., 1998] Condon, J. J., Cotton, W. D., Greisen, E. W., Yin, Q. F., Perley, R. A., Taylor, G. B., and Broderick, J. J. (1998). The NRAO VLA Sky Survey. *The Astronomical Journal*, 115:1693–1716.

[Conway et al., 1963] Conway, R. G., Kellermann, K. I., and Long, R. J. (1963). The radio frequency spectra of discrete radio sources. *Monthly Notices of the Royal Astronomical Society*, 125:261.

[Cruz et al., 2007] Cruz, M. J., Jarvis, M. J., Rawlings, S., and Blundell, K. M. (2007). The 6C** sample of steep-spectrum radio sources - II. Redshift distribution and the space density of high-redshift radio galaxies. *Monthly Notices of the Royal Astronomical Society*, 375:1349–1363.

[de Vries et al., 1997] de Vries, W. H., Barthel, P. D., and O'Dea, C. P. (1997). Radio spectra of Gigahertz Peaked Spectrum radio sources. *Astronomy and Astrophysics*, 321:105–110.

[Gopal-Krishna et al., 1983] Gopal-Krishna, Patnaik, A. R., and Steppe, H. (1983). A sample of 25 extragalactic radio sources having a spectrum peaked around 1 GHz. *Astronomy and Astrophysics*, 123:107–110.

[Gregory et al., 1996] Gregory, P. C., Scott, W. K., Douglas, K., and Condon, J. J. (1996). The GB6 Catalog of Radio Sources. *Astrophysical Journal Supplement*, 103:427.

[Harwood et al., 2016] Harwood, J. J., Croston, J. H., Intema, H. T., Stewart, A. J., Ineson, J., Hardcastle, M. J., Godfrey, L., Best, P., Brienza, M., Heesen, V., Mahony, E. K., Morganti, R., Murgia, M., Orrú, E., Röttgering, H., Shulevski, A., and Wise, M. W. (2016). FR II radio galaxies at low frequencies - I. Morphology, magnetic field strength and energetics. *Monthly Notices of the Royal Astronomical Society*, 458:4443–4455.

[Hogg et al., 2010] Hogg, D. W., Bovy, J., and Lang, D. (2010). Data analysis recipes: Fitting a model to data. *ArXiv e-prints*.

[Jansky, 1933] Jansky, K. G. (1933). Electrical phenomena that apparently are of interstellar origin. *Popular Astronomy*, 41:548.

[Jarvis et al., 2001] Jarvis, M. J., Rawlings, S., Eales, S., Blundell, K. M., Bunker, A. J., Croft, S., McLure, R. J., and Willott, C. J. (2001). A sample of 6C radio sources designed to find objects at redshift z¿4 - III. Imaging and the radio galaxy K-z relation. *Monthly Notices of the Royal Astronomical Society*, 326:1585–1600.

[Kameno et al., 2003] Kameno, S., Inoue, M., Wajima, K., Sawada-Satoh, S., and Shen, Z.-Q. (2003). Free-Free Absorption and the Unified Scheme. *Publications of the Astronomical Society of Australia*, 20:213–221.

[Kellermann, 1972] Kellermann, K. I. (1972). Radio Emission from Compact Objects (survey Lecture). In Evans, D. S., Wills, D., and Wills, B. J., editors, *External Galaxies and Quasi-Stellar Objects*, volume 44 of *IAU Symposium*, page 190.

[Kellermann and Pauliny-Toth, 1981] Kellermann, K. I. and Pauliny-Toth, I. I. K. (1981). Compact radio sources. *Annual review of astronomy and astrophysics*, 19:373–410.

[Kharb, 2004] Kharb, P. (2004). *A Pc-scale study of radio-loud AGN: The Fanaroff-Riley Divide and Unification*. PhD thesis, Indian Institute of Astrophysics.

[Lilly and Longair, 1984] Lilly, S. J. and Longair, M. S. (1984). Stellar populations in distant radio galaxies. *Monthly Notices of the Royal Astronomical Society*, 211:833–855.

[Marlow et al., 2000] Marlow, D. R., Rusin, D., Jackson, N., Wilkinson, P. N., Browne, I. W. A., and Koopmans, L. (2000). Redshifts of CLASS Radio Sources. *The Astronomical Journal*, 119:2629–2633.

[Mingaliev et al., 2013] Mingaliev, M. G., Sotnikova, Y. V., Mufakharov, T. V., Erkenov, A. K., and Udovitskiy, R. Y. (2013). Gigahertz-peaked spectrum (GPS) galaxies and quasars. *Astrophysical Bulletin*, 68:262–272.

[Myers et al., 2003] Myers, S. T., Jackson, N. J., Browne, I. W. A., de Bruyn, A. G., Pearson, T. J., Readhead, A. C. S., Wilkinson, P. N., Biggs, A. D., Blandford, R. D., Fassnacht, C. D., Koopmans, L. V. E., Marlow, D. R., McKean, J. P., Norbury, M. A., Phillips, P. M., Rusin, D., Shepherd, M. C., and Sykes, C. M. (2003). VizieR Online Data Catalog: CLASS survey of radio sources (Myers+, 2003). *VizieR Online Data Catalog*, 8072.

[O'Dea, 1998] O'Dea, C. P. (1998). The Compact Steep-Spectrum and Gigahertz Peaked-Spectrum Radio Sources. *The Publications of the Astronomical Society of the Pacific*, 110:493–532.

[O'Dea et al., 1991] O'Dea, C. P., Baum, S. A., and Stanghellini, C. (1991). What are the gigahertz peaked-spectrum radio sources? *Astrophysical Journal*, 380:66–77.

[Orienti, 2016] Orienti, M. (2016). Radio properties of Compact Steep Spectrum and GHz-Peaked Spectrum radio sources. *Astronomische Nachrichten*, 337:9.

[Peacock and Wall, 1981] Peacock, J. A. and Wall, J. V. (1981). Bright extragalactic radio sources at 2.7 GHz. I - The Northern Hemisphere catalogue. *Monthly Notices of the Royal Astronomical Society*, 194:331–349.

[Peacock and Wall, 1982] Peacock, J. A. and Wall, J. V. (1982). Bright extragalactic radio sources at 2.7 GHz. II - Observations with the Cambridge 5-km telescope. *Monthly Notices of the Royal Astronomical Society*, 198:843–860.

[Rengelink et al., 1997] Rengelink, R. B., Tang, Y., de Bruyn, A. G., Miley, G. K., Bremer, M. N., Roettgering, H. J. A., and Bremer, M. A. R. (1997). The Westerbork Northern Sky Survey (WENSS), I. A 570 square degree Mini-Survey around the North Ecliptic Pole. *Astronomy and Astrophysics Supplement series*, 124.

[Rudnick and Jones, 1982] Rudnick, L. and Jones, T. W. (1982). Compact radio sources - The dependence of variability and polarization on spectral shape. *Astrophysical Journal*, 255:39–47.

[Rybicki and Lightman, 1986] Rybicki, G. B. and Lightman, A. P. (1986). *Radiative Processes in Astrophysics*.

[Sadler, 2016] Sadler, E. M. (2016). GPS/CSS radio sources and their relation to other AGN. *Astronomische Nachrichten*, 337:105.

[Snellen et al., 1998a] Snellen, I. A. G., Schilizzi, R. T., Bremer, M. N., de Bruyn, A. G., Miley, G. K., Rottgering, H. J. A., McMahon, R. G., and Perez Fournon, I. (1998a). Optical and near-infrared imaging of faint Gigahertz Peaked Spectrum sources. *Monthly Notices of the Royal Astronomical Society*, 301:985–1000.

[Snellen et al., 1998b] Snellen, I. A. G., Schilizzi, R. T., de Bruyn, A. G., Miley, G. K., Rengelink, R. B., Roettgering, H. J., and Bremer, M. N. (1998b). A new sample of faint Gigahertz Peaked Spectrum radio sources. *Astronomy and Astrophysics Supplement*, 131:435–449.

[Snellen et al., 2000a] Snellen, I. A. G., Schilizzi, R. T., Miley, G. K., de Bruyn, A. G., Bremer, M. N., and Röttgering, H. J. A. (2000a). On the evolution of young radio-loud AGN. *Monthly Notices of the Royal Astronomical Society*, 319:445–456.

[Snellen et al., 2000b] Snellen, I. A. G., Schilizzi, R. T., and van Langevelde, H. J. (2000b). Multifrequency VLBI observations of faint gigahertz peaked spectrum sources. *Monthly Notices of the Royal Astronomical Society*, 319:429–444.

[Spoelstra et al., 1985] Spoelstra, T. A. T., Patnaik, A. R., and Gopal-Krishna (1985). A sample of 25 extragalactic radio sources having a spectrum peaked around 1 GHz (List 2). *Astronomy and Astrophysics*, 152:38–41.

[Torniainen, 2008] Torniainen, I. (2008). *Multifrequency studies of gigahertz-peaked spectrum sources and candidates*. PhD thesis, Helsinki University of Technology.

[Wilson et al., 2012] Wilson, T. L., Rohlfs, K., and Huttemeister, S. (2012). *Tools of Radio Astronomy, 5th edition*.

[Wright, 2006] Wright, E. L. (2006). A Cosmology Calculator for the World Wide Web. *The Publications of the Astronomical Society of the Pacific*, 118:1711–1715.