USING A MONTE CARLO SIMULATION TO RESOLVE THROUGH CONFUSION THE RADIO SOURCE BACKGROUND OF WEAK RADIO SOURCES OUT TO Z = 2.9

Melanie L. Grierson

Instructor: Dr. Jim Condon

Department of Astronomy

University of Virginia

This thesis is submitted in partial completion of the requirements of the BA Astronomy Major.

May 8th, 2018

DRAFT VERSION MAY 10, 2018
Typeset using LaTeX **twocolumn** style in AASTeX61

USING A MONTE CARLO SIMULATION TO RESOLVE THROUGH CONFUSION THE RADIO SOURCE BACKGROUND OF WEAK RADIO SOURCES OUT TO Z = 2.9

M. GRIERSON[1]

[1]*Department of Astronomy, University of Virginia*

ABSTRACT

This thesis details how to create a simulated universe on a $1000'' \times 1000''$ sky with $1'' \times 1''$ pixels. The simulated sky was created to provide a realistic distribution of radio sources given a specific source count and includes instrumental observing effects associated with the Karl G. Janksy Very Large Array (VLA). 1.4 GHz counts from Condon et al. (2012), which underwent a spline interpolation, were used to create $\approx$ 37k radio sources ranging from 10^3 to 10^{-8} Jy. These realistic sources allowed for the determination of the number of galaxies per flux density bin with a spacing of 0.01 in log flux density in order to calculate the fraction of galaxies on the sky with each corresponding flux density. Each of the simulated galaxies was smeared by an $8''$ FWHM point spread function (PSF) and was assigned a random galaxy location, in which the flux densities of overlapping sources are summed. In order to more closely approximate and mimic the instrumental effects obtained on a VLA observation, both the Gaussian primary attenuation pattern with a $14'$ FWHM and a convolution of the confusion amplitude distribution with the VLA's Gaussian instrumental noise distribution are included within the simulation. This simulation program will be used to analyze a confusion-limited VLA observation intended to study the star-formation history of the universe by counting radio sources as faint as $S \sim 0.2\,\mu$Jy.

1. INTRODUCTION

1.1. *The Star Formation History of the Universe*

The star formation history of the universe (SFHU) is usually specified by the comoving star formation rate density (SFRD) $\psi(t)$ (in units $M_\odot$ yr^{-1} Mpc^{-3}) as a function of lookback time t (Feldmann et al. 2016). Madau & Dickinson (2014) used far-ultraviolet (FUV) and far-infrared (FIR) data to fit the SFRD, which Matthews (2018) plotted with a linear lookback time (in units of Gyr) axis to emphasize that $\approx 90\%$ of all stars were formed within the past 11 Gyr at redshifts $z \leq 2.9$ (Figure 1).

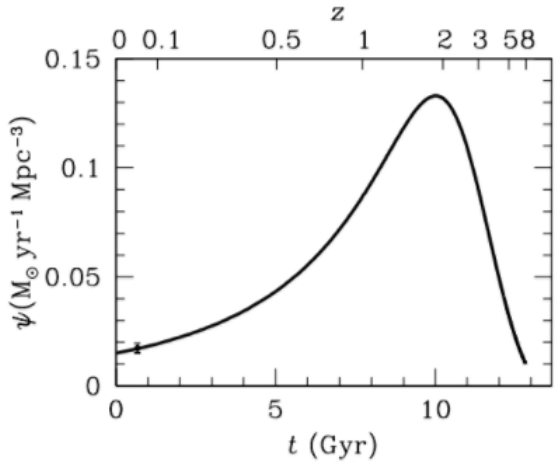

Figure 1. The Madau & Dickinson (2014) SFRD plotted by Matthews (2018) with a linear lookback time axis to show that $\approx 90\%$ of all stars have formed since $z = 2.9$.

.

Figure 1 shows that the SFRD peaks at around $z \approx 1.9$ and at a lookback time $t \approx 10$ Gyr, which defines the time called "cosmic noon". The period from $z \approx 1.5 - 3$ delineated a time in the formation of the universe when there was intense star formation. Since this peak at cosmic noon the SFRD has declined by a factor of ten. Interestingly, the decline is often connected to the impact of gaseous outflows driven by feedback from supermassive black holes (Feldmann et al. 2016). Therefore to probe the star formation history of the universe (SFHU) it's necessary to observe galaxies with redshifts around this peak.

The SFHU is fundamental for cosmology because it indirectly constrains and provides information about the gravitational collapse of dark matter and baryons which form galaxies, how the elements chemically evolved over time, the creation of interstellar dust, the formation rate of core-collapse supernovae (SNe), and the feedback between stars and supermassive black holes residing in AGNs (Feldmann et al. 2016).

Also, it is important for us to learn more about the SFHU for its own sake. Older stars within star-forming galaxies (SFGs) account for the majority of the total stellar mass and emit primarily in the near-infrared (NIR) (Jarvis et al. 2015). Current infrared and optical observations using space-based telescopes such as the *Spitzer Space Telescope* and the *Hubble Space Telescope* (HST) have gathered significant data tracing stellar emission to high redshifts.

Unfortunately, for galaxies at large redshifts (and therefore longer lookback times) it becomes increasingly difficult to trace star-formation rates and evolution due to higher levels of dust obscuration surrounding hot young blue stars. This dust obscuration of young stars causes visible and ultra-violet (UV) surveys to be error-prone and incomplete (Madau & Dickinson 2014). It has been measured from integrated optical and infrared background radiation that $\approx 50\%$ of the light from stellar activity is hidden by the surrounding dust (Buat et al. 2007).

Radio waves trace the total star-formation rate in both obscured and unobscured galaxies (Jarvis et al. 2015) because radio continuum emission is not absorbed by dust or contaminated by older stellar populations (Condon 1992). However, active galactic nuclei (AGN) within these SFGs can contaminate the radio emission and therefore steps should be taken to exclude them.

At the beginning of cosmic noon, 75% of the stars were in SFGs with radio flux densities $S > 2 \times 10^{-33}$ W m^{-2} Hz^{-1} at 1.4 GHz. Radio astronomers use a convenient non-SI unit of measurement for spectral flux density known as the Jansky (Jy) defined as 10^{-26} W m^{-2} Hz^{-1}, so 2×10^{-33} W m^{-2} Hz$^{-1} \approx 0.2$ μJy. The Jansky is typically used to describe point sources. For the remainder of this thesis flux densities will be expressed in this unit. Even the most sensitive telescopes today cannot detect individual radio galaxies with flux densities less than a few μJy. However, the populations of fainter radio sources at $z = 2.9$ can be studied statistically by the "confusion" they produce in radio sky images.

1.2. *Confusion Statistics*

Astronomers use the term "confusion" for the brightness fluctuations in a sky image produced by multiple faint sources with varying flux densities piling up in each resolution element. Historically confusion was measured from the probability distribution $P(D)$ of the deflections D on a chart-recorder plot, but today the observed "deflection" D at any point in an aperture-synthesis array image is the intensity in units of flux density per synthesized beam (the point-source response) solid angle (Condon et al. 2012). This deflection is the sum of contributions from the noise-free source confusion and the instrumental noise in the image. Therefore the observed $P(D)$ distribution is the convolution of the source confusion and noise distributions, allowing one to extract the

source confusion distribution and its width from the observed deflections and measured noise.

Condon et al. (2012) produced the 3D profile image in Figure 2 showing the 3 GHz confusion amplitude distribution observed in a single Jansky Very Large Array (VLA) pointing made with an 8″ FWHM Gaussian resolution, truncated at $D = 100\,\mu$Jy beam^{-1} to indicate the intensity scale. The VLA is an aperture-synthesis interferometer consisting of 27 25-m diameter dish antennas, and the primary beam (angular response) of each antenna at 3 GHz is a Gaussian with FWHM$\approx$ 14 arcmin. The preliminary image produced by Fourier transforming the interferometer fringes is the sky brightness attenuated by the 14 arcmin primary beam, smoothed by the point-source response of the 8 arcsec synthesized beam, and degraded by instrumental noise. The instrumental noise has a Gaussian amplitude distribution at this stage. After the preliminary image has been corrected for primary-beam attenuation, the image noise gradually grows with distance from the pointing center.

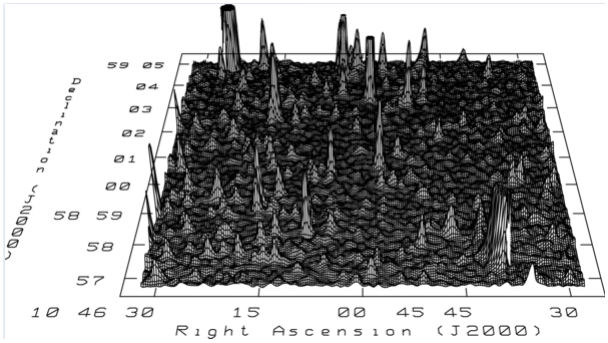

Figure 2. This profile plot shows the confusion in a 3 GHz VLA image made with an 8" FWHM Gaussian synthesized beam Condon et al. (2012). The brightest sources have been truncated at 100 μJy beam^{-1} to show the intensity scale.

Condon et al. (2012) estimated the noise in confusion-limited regions near the VLA's pointing center. They measured the noise distribution from one source-free region near the edge of their wideband image and obtained a Gaussian with rms $\sigma = 1.02\,\mu$Jy beam^{-1}. This noise distribution can be seen in Figure 3. Because Figure 3 has a logarithmic ordinate, a Gaussian appears parabolic in shape.

Condon et al. (2012) details beautifully how to resolve the radio source background through the use of confusion. It specifically details how one can extract the true source count (number of sources per steradian per unit of flux density) down to $S \sim \sigma/2 \sim 0.5\,\mu$Jy, a factor of ten below the minimum flux density $S \approx 5\sigma \sim 5\,\mu$Jy of individually detectable sources, from a confusion-limited survey. However, the analytic equations Condon (1974) used to calculate source counts from the confusion $P(D)$ distribution are exact only for power-law source counts. The actual source counts

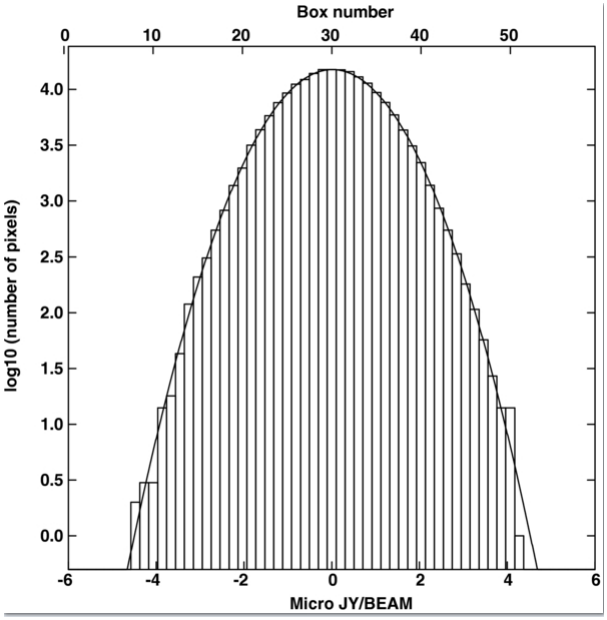

Figure 3. The Gaussian noise histogram obtained by Condon et al. (2012) has rms $\sigma = 1.02\,\mu$Jy beam^{-1} and appears parabolic on this plot because the ordinate is logarithmic.

are only approximately power law over limited flux-density ranges.

1.3. *Monte Carlo Simulation Approach*

While statistical analysis using confusion is necessary to obtain higher sensitivity, the data have only been analyzed by Condon et al. (2012) through the use of a power-law approximation. Therefore, to avoid the errors which come from using the power-law approximation, I created a Monte Carlo simulation of a patch of sky, as it would appear to the VLA. The goal of the simulation would be to:

1. Compare the observed confusion analysis of the actual source count to something other than the power-law approximation

2. Create realistic synthetic data to more accurately determine the sensitivity which can be obtained through a more sensitive future observation with the VLA C configuration.

To achieve these goals I created the Monte Carlo simulation detailed in Appendix C and explained qualitatively throughout the rest of this thesis. The simulation uses the original 1.4 GHz source count from Condon et al. (2012) by creating, first, a noiseless image of a 1000 arcsec $\times$ 1000 arcsec view of the sky and placing down at random positions the total number of galaxies. Then each galaxy flux density was smeared by a Gaussian of FWHM = 8 arcsec. Overlapping galaxies within this section of sky were summed together yielding a view of the sky from an

observer's perspective. In order to best approximate the instrumental effects one would get through observation on the VLA, the image was multiplied by the telescope's primary beam, a Gaussian attenuation pattern with FWHM = 14 arcmin. The noiseless $P(D)$ distribution was also convolved with a Gaussian noise distribution to simulate the $P(D)$ distribution that an observer would see in a VLA observation with configuration C.

This simulation was primarily developed for analysis of the potential observation proposed for Fall 2018 on the VLA, which plans to observe for 120 hr a carefully chosen sky area in order to model the SFHU since $z = 2.9$ (Matthews 2018). My aim is to not only provide realistic observation analysis with similar constraints before the supposed observation date, but also be used after the observation to test the actual source counts and constrain the star formation history of the universe back to $z \sim 2.9$.

2. MODIFYING THE 1.4 GHZ SOURCE COUNT DATA

2.1. *Using Spline Interpolation to Change Sampling Size*

The 1.4 GHz source count used by Condon et al. (2012) lists the differential source count, $n(S)$, from 10^{-8} Jy to 10^{3} Jy. For convenience, the 1.4 GHz data for this project can be seen in Appendix A. These counts were given with a spacing of 0.2 in log flux density, making the sampling size too coarse to accurately create a simulated sky. To make the sampling size of the data finer, I used a spline interpolation to expand the original data by fitting a piecewise polynomial parametric curve. By plugging in all 56 source flux densities S and counts $n(S)$ from the original 1.4 GHz data into a pre-defined function which executes a spline interpolation, I increased the sample size so that it sampled every 0.01 in log flux density instead of 0.2 in log flux density. The spline $\mathbf{S}$ maps the values on an interval $[a,b]$, which in the case of this simulation is 10^{3} to 10^{-8} Jy, to a set of real values, $\mathbb{R}$. The specific spline interpolation utilized to create the interpolated 1.4 GHz data with 0.01 intervals in log flux density is a cubic spline, which is a spline constructed of a third-order polynomial that passes through a set of control points. In the case of this specific example the control points come from the original 1.4 GHz data.

$$\mathbf{S}_j = a_j + b_j(x-x_j) + c_j(x-x_j)^2 + d_j(x-x_j)^3 \tag{1}$$

Instead of applying the spline interpolation to extend only the source flux densities, it was primarily applied to find the average number of radio sources per bin for every 0.01 in log flux density. This process was applied to both 1) a power law approximation of the source count and 2) the actual source-count data. The actual data were analyzed further and were used to create the simulated universe.

2.2. *Power Law Verification*

First the power-law count approximation was used as an initial test of the simulation. The main reason for beginning with the power law verification is that the distribution of the confusion amplitude provided by this approximation can be calculated analytically (Condon 1974). The cumulative source count $N(>S)$ is the mean number of sources per steradian stronger than S, where S is typically expressed in Jy.[1] The differential source count $n(S)$ can be defined as

$$n(S) = \left|\frac{dN}{dS}\right| = -\frac{dN}{dS} \tag{2}$$

Therefore, $N(>S)$ can be expressed

$$N(>S) = \int_S^\infty n(x)dx\ , \tag{3}$$

where $n(S)dS$ is the mean number of sources per steradian with flux densities in the infinitesimal flux density range from S to $S+dS$. The mean number of sources per steradian per flux density bin, λ, can be approximated as $\lambda \approx n(S)\Delta S$, where ΔS is a small but finite flux density range $\ll S$, with a flux-density bin spanning from $S-\Delta S/2$ to $S+\Delta S/2$. To convert λ, the mean number of sources per steradian, to being per bin I multiplied it with the solid angle Ω steradians of the sky covered. Typically for simulations of this type the range of flux densities is quite large (i.e. $S_{\max}/S_{\min} \gg 1$). Thus, it is often favorable to work with the differential source count per logarithmic flux density range $n^*(S)$, which is expressed by

$$n^*(S) = \left|\frac{dN}{d\log S}\right| = -\frac{dN}{d\log S} \tag{4}$$

In the same vein as before, the mean number of sources in the infinitesimal flux density range from S to $S+d\log S$ is $\lambda = n^*(S)d\log S$. Therefore, if $\Delta\log(S) \ll 1$, the mean number of sources per steradian in the finite narrow bins spanning the logarithmic flux density range $S-\Delta\log S/2$ to $S+\Delta\log S/2$ is $\lambda \approx n^*(S)\Delta\log S$. Multiplying by the area of the sky across Ω steradians yields the mean number of sources per bin.

Using the chain rule for derivatives the relation between $n(S)$ and $n^*(S)$ in Equations 2 and 4 is revealed as:

$$n^*(S) = -\frac{dN}{d\log S} = \frac{dN}{dS}\frac{dS}{d\ln S}\frac{d\ln S}{d\log S} = n(S)S\ln(10) \tag{5}$$

Equation 5 expresses the power law approximation for determining the differential source count per logarithmic flux density range in terms of n(S). Therefore, for example, if $n(S) = 300S^{-2}$, then Equation 5 can be re-written as:

$$n^*(S) = 300\ln(10)S^{-2} \tag{6}$$

[1] Steradian (sr) or squared radians is the SI unit of solid angle.

Equation 6 takes the functional form of a power-law source count with $n(S) = kS^{-\gamma}$, where k is a constant and γ is the differential count slope on a log-log plot. Condon et al. (2012) specifically uses the example of $\log[S^2 n(S)] \approx 300S^{-2}$ to produce some of their confusion plots.

The differential source count was calculated for each of the galaxy flux densities within the 1.4 GHz data. After calculating the differential source count for each of the flux density sources, the approximated number of galaxies per flux density bin per image area, λ, was calculated using

$$\lambda = n^*(S) \times \Delta \log S \times \left(1000'' \times \frac{\pi}{3600'' \times 180}\right)^2 \quad (7)$$

where $\Delta \log S = 0.01$, is the size of the logarithmic flux density bin. The squared factor in Equation 7 is the image solid angle Ω in sr so that λ becomes the mean number of galaxies per flux density bin per image.

The simulated image size of $1000'' \times 1000'' = 10^6$ arcseconds^2 was chosen in order to cover a 3 GHz primary beam of the VLA, which has a FWHM $\approx 14' = 840''$. This section of sky would also contain only $\approx 3.7 \times 10^4$ sources stronger than 0.01 μJy. Since our point-source response function (PSF) is a Gaussian with FWHM = $8''$, it will not be a problem that sources outside of the image, which are at least 80 arcsec outside of the primary FWHM circle, are not contributing to the inside that circle. Using a random Poisson generator provided by *numpy*, the number of sources one should find within a bin is a function of λ (from Eq. 7), the average number of galaxies per flux density bin per image. For each flux-density bin the Poisson probability that there will be n sources in a given simulation is

$$P_p(n|\lambda) = \frac{\lambda^n}{n!} e^{-\lambda} \quad (8)$$

Summing over bins should provide an estimate of the number of sources within the simulated image, which as stated earlier is $\approx 3.7 \times 10^4$. Additionally doing this verification method was important because it revealed an error in the program caused by using an integer variable where a real variable was needed. Changing from a integer to a float was necessary in order to get the Gaussian function seen in Appendix C to work.

2.3. *Actual Source Count Calculation*

The same process used for the power-law approximation was used to calculate the average number per bin for a realistic source count. Obtaining a realistic source count is best done by working with the brightness-weighted differential count $S^2 n(S)$ from the 1.4 GHz data because it is nearly constant in the flux density ranges that contribute the most to the radio sky brightness. The source counts within the 1.4 GHz data used by Condon et al. (2012) are presented with the static Euclidean normalization $S^{5/2} n(S)$. Therefore, simply we can covert $S^{5/2} n(S)$ to $S^2 n(S)$ by:

$$S^2 n(S) = \frac{S^{5/2} n(S)}{S^{1/2}} \quad (9)$$

Instead of using the Equation 5 to determine $n^*(S)$, the brightness weighted different source count from Equation 9 can be used to determine first $n(S)S$ and then $n^*(S)$ by the following relation

$$n^*(S) = n(S) S \ln(10) \quad (10)$$

Equation 7 can then be used again to solve for the average number of galaxies per bin without using a power law approximation by plugging in the resulting $n^*(S)$ from Equation 10 . The spline interpolation once again can make the sampling size finer by calculating the average number of galaxies per bin per image for every 0.01 in log flux density instead of 0.2 in log flux density. An array of the mean number of galaxies per bin from the original source data was modified with a random Poisson generator to produce the number n of galaxies in a simulation given the mean number lambda, as in Equation 8. This randomized average number of galaxies per 0.01 in log flux density was used to create the simulated sky. It should be noted that the "random" Poisson generator *can* be given a specified "seed" number to manually control whether each run of the universe simulation will yield the same average number of galaxies per flux density bin per image area, λ. By re-using the same seed, an earlier calculation can be reproduced exactly. For example, in Sect. 4.4 I cut out the faintest sources to see what effect that had on the $P(D)$ distribution, and I had to re-use the same seed in order to obtain the same strong-source contribution to the $P(D)$ distribution.

3. MONTE CARLO CONFUSION SIMULATION

There were two main steps to making the simulated sky. First, the sky was created by making a 2D array of 1 arcsecond^2 pixels on a $1000'' \times 1000''$ grid. Secondly, the sky was filled with galaxies with flux densities obtained ranging from 10^{-8} to 10^3 Jy with steps of 0.01 in log flux density (Appendix A). The number of galaxies with a certain flux density was determined from the average number of galaxies per flux density bin per image λ calculated earlier.

3.1. *2D Gaussian Equation*

To make the simulation as close to approximating the instrumental effects of the VLA as possible, a Gaussian distribution was used to smear each of the sources placed on the $1000'' \times 1000''$ sky with an $8''$ FWHM resolution. The Gaussian distribution α that was applied to each galaxy is

$$\alpha = \exp\left(-\frac{4 \ln 2\, r^2}{\theta^2}\right) \quad (11)$$

where θ = FWHM = $8''$ and r is the radial offset from the center of the galaxy and can be calculated as

$$r^2 = (x_0 - x_{gal})^2 + (y_0 - y_{gal})^2 \quad (12)$$

where x_0 and y_0 are reference points on the 2D pixel array and the central point of the galaxy is located at (x_{gal}, y_{gal}).

Therefore, the flux density for a single galaxy as a function of distance from the center of the galaxy is:

$$S(r) = S_0\alpha = S_0 \exp\left(-\frac{4\ln 2\, r^2}{\theta^2}\right) \quad (13)$$

Within the simulation the galaxy coordinates are determined first and then fed into Eq.12 to determine the radial offset. This value is then used to determine the flux density as a function of radial offset with Eq.13. The exact python function I created to place an $8''$ FWHM Gaussian at the position of each radio galaxy can be seen in Appendix C.

3.2. *Pixel Array Creation*

After creating the Gaussian function to smear each of the flux density sources, my next step was to set up the 2D pixel array. This array begins empty and then becomes filled by flux densities as a function of λ. The 2D pixel array was created by initially making a 1000 x 1000 element array filled with zeros. It was purposely made in this way so that each pixel would correspond to 1 arcsecond^2 and therefore the 1000×1000 pixel array would correspond directly with the $1000'' \times 1000''$ sky covered. Each of these pixels was given an associated x- and y- coordinate by creating two more 2D arrays for x-coordinates increasing from right to left and y-coordinates increasing from top to bottom. Starting with the empty 2D pixel array, the sky was built by first looping through the array of the randomized n number of galaxies per flux density bin. Once the simulation identified a flux density bin in which the number n of sources was not equal to 0, the program created the $8''$ FWHM Gaussian responses to the point sources, caused by the finite resolution of the VLA, for the total number of galaxies for each respective central flux density. This was done using Equation 13 which shows how the flux density decreases as a Gaussian with respect to distance from the galaxy center. Since each pixel is equal to 1 arcsecond^2 no conversion has to be made between the flux density per pixel and flux density per beam.

In order to create the sky, each of these galaxies was given a randomized x- and y- galaxy coordinate ranging from 0 to 1000 in each direction. For galaxies which overlapped, the code was made to sum the contribution of flux density per pixel. Therefore, if a galaxy with the same central flux density had the same x- and y- coordinate it would appear as if it was one Gaussian galaxy source with 2 times the flux density. For each run there are approximately 37k sources to place in a random location on the sky and then apply the Gaussian smearing α to. In order to visually see the universe, a 2D linear and log contour plot of each sky was created by using the final filled pixel array of summed Gaussian sources. An example of a linear and log 2D contour plot can be seen below in Figures 11 and 12. Larger versions of these plots can be seen in Appendix E.

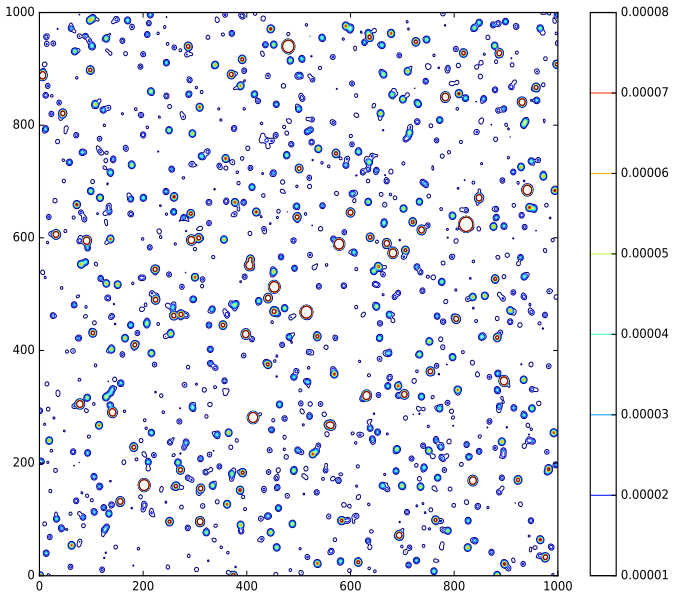

Figure 4. A 2D representation of one universe run based on the 1.4 GHz data from Condon et al. (2012) in the linear scale. Contour lines mark different levels of flux density in units of Jy.

.

Producing these 2D contour plots marks the end of creating the Monte Carlo simulation of the noiseless sky smoothed by the $8''$ FWHM PSF. In the following sections I will multiply the sky by the $14'$ FWHM primary beam attenuation pattern to simulate the still noiseless VLA image from a single-pointing observation.

4. ANALYZING SIMULATION RESULTS

One the main uses of this Monte Carlo simulation is to gain a deeper understanding of what kind of observable data can be determined from similar observations on the VLA. The results of the simulation provide realistic source counts which can be not only compared to results from the power law approximation analysis, but also with actual data from potential future observations with the VLA. This section covers various tests one can implement to fine tune the simulation and to understand what the simulation tells us.

4.1. *Probability Distributions*

One of the first tests we can do to analyze the validity of the Monte Carlo simulation is by creating probability distributions of several different runs. Because the simulation is

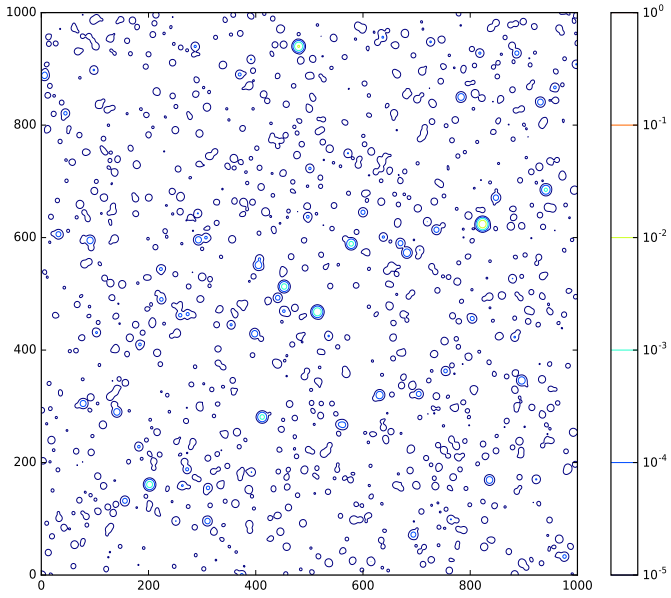

Figure 5. A 2D contour representation of the same universe from 11, but represented logarithmically.

.

reading in the same mean number of sources per bin λ and modifying it through the usage of a random Poisson function, every universe run should yield relatively similar numbers of galaxies at each flux density. Therefore, by comparing several runs, my plot in Figure 6 is showing how the slightly different numbers n end up producing slightly different $P(D)$ distributions. The method used in python to achieve the probability distributions of seven simulations is detailed in Appendix D.

The first step, was to reduce the dimensions of the 1000 $\times$ 1000 2D array. The array needs to be reduced in order to account for the simulation only being valid within the FOV of the image. Therefore, while it is equally likely for a galaxy to be located at any coordinate within the 1000 $\times$ 1000 pixel array, galaxies outside of this field which could add flux density value to the edges due to being smeared by the 8″ FWHM Gaussian PSF are not accounted for. To remove the inaccuracy at the edges I reduced the 1000 $\times$ 1000 by trimming off the radius size of the galaxies. Specifically pretending a galaxy source existed right at the edge of the field it could only affect the values within its radius. Therefore, 9 pixels from all sides of the array were trimmed off. The reduced array is 984 pixels^2, which corresponds with a 984 arcsecond^2 field of view. After correcting for the limits of the simulation the data can now be used to determine the probability distribution. This was done by reading in each flux density value and placing it within a bin of bin width $d = 0.01\,\mu$Jy. The resulting probability distributions for seven runs of the simulation can be seen in Fig.6. From the overlapping distributions it is clear that they all agree within the error of their respective noise.

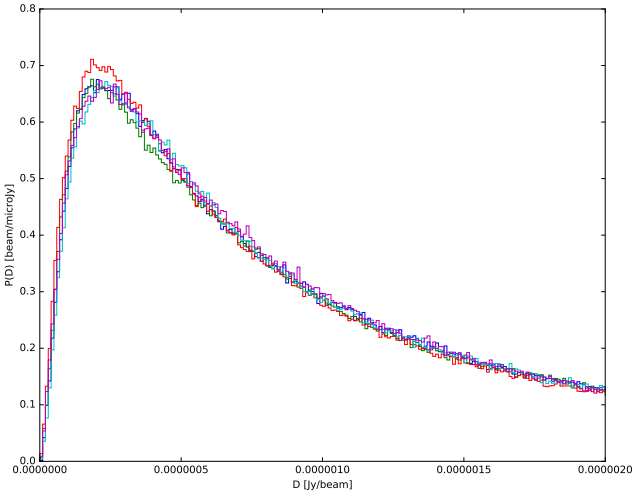

Figure 6. The $P(D)$ distributions of 7 different runs of the simulation. Note that the universe run delineated by the red line is an obvious outlier.

4.2. *Simulating VLA Instrumental Effects*

The plots seen in Figures 11 (linear contour spacaing) and 12 (logarithmic contour spacing) show simulated skies which are completely noiseless. However, observations done by the VLA introduce two instrumental effects: (1) primary beam attenuation and (2) instrumental noise.

4.2.1. *Primary Beam*

The VLA primary beam attenuation pattern at 3 GHz can be approximated as a Gaussian with FWHM = 14′. To add this effect to the contour plots seen within Figures 11 and 12 one must simply multiply each sky pixel array by a Gaussian with FWHM = 14′ and peak = 1. The Gaussian should be centered on the pixel array at pixel $x = 500$, $y = 500$. To visually see this effect the resulting linear 2D contour plot of the universe seen in Fig. 11 can be seen in Fig. 13. Notice how the galaxy sources appear to fall off in number from the center causing a circular shape of the field. For convenience a larger version of this figure can be seen in Appendix E.

4.2.2. *Noise Distribution*

The other VLA effect which must be included in order to accurately model a VLA observation is the noise which smears the observed flux density from each galaxy source. This smearing can be modeled as a convolution of the noiseless $P(D)$ distributions (seen in Fig.6) with a noise Gaussian of a specified rms value σ. It was chosen as $\sigma = 1.012\,\mu$Jy beam^{-1} to match with the rms noise measured by Condon et al. (2012) statistically and seen in Fig.3.

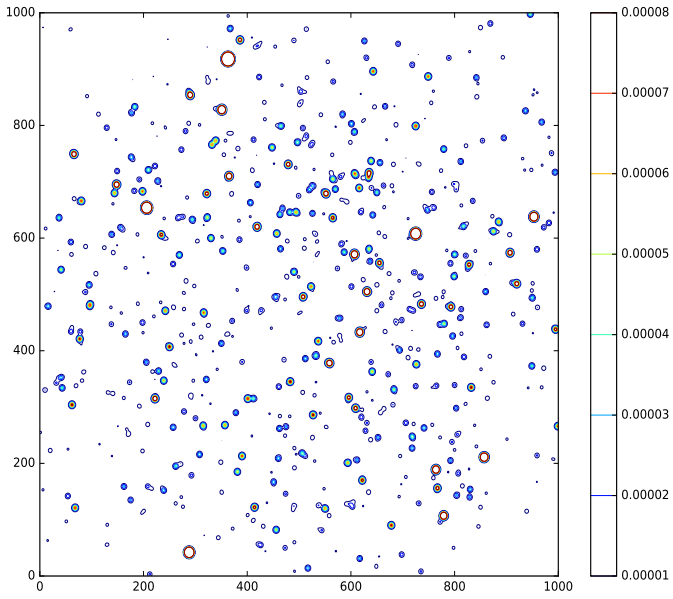

Figure 7. A 2D linear contour plot of the same universe from Figures 11 and 12 by including the effect of the primary beam attenuation on the VLA with a FWHM = 14′.

A predefined convolution function was used to read the noiseless $P(D)$ distribution and a Gaussian with $\sigma = 1.012\mu$Jy beam^{-1}. Fig.8 shows both the noiseless $P(D)$ distribution from one simulation run and the noisy $P(D)$ distribution after convolution with the instrumental noise.

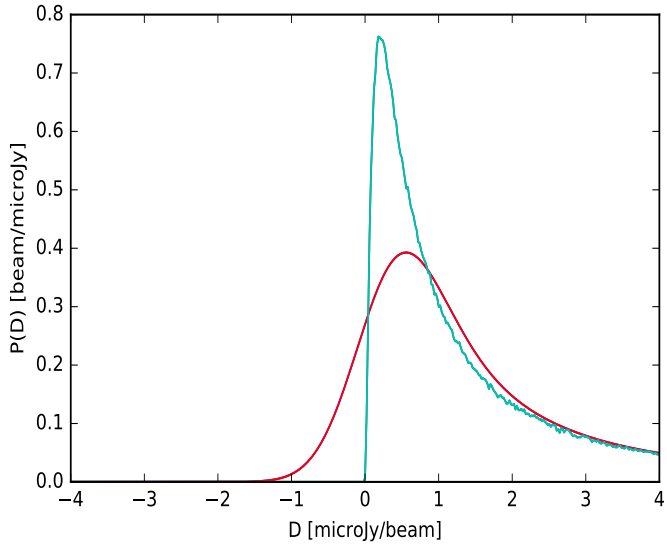

Figure 8. The noiseless probability distribution can be seen in green and the convolved distribution including instrumental noise from the VLA can be seen in red.

4.3. *Testing Optimal Sensitivity for VLA Observing*

The simulation can be used for determining the sensitivity of the VLA in its C configuration to faint sources. This can be done by analyzing to what degree one can distinguish the difference between the noisy $P(D)$ distributions when the source counts in the faintest flux-density bins are half of the original value or doubled. By manipulating the 1.4 GHz source counts at the lowest flux densities it is simple to extract the probability distributions using the same method from Sect. 4.2.2. I also re-used the Poisson generator seed to eliminate variations in the numbers and positions of the strong sources from one simulation to another.

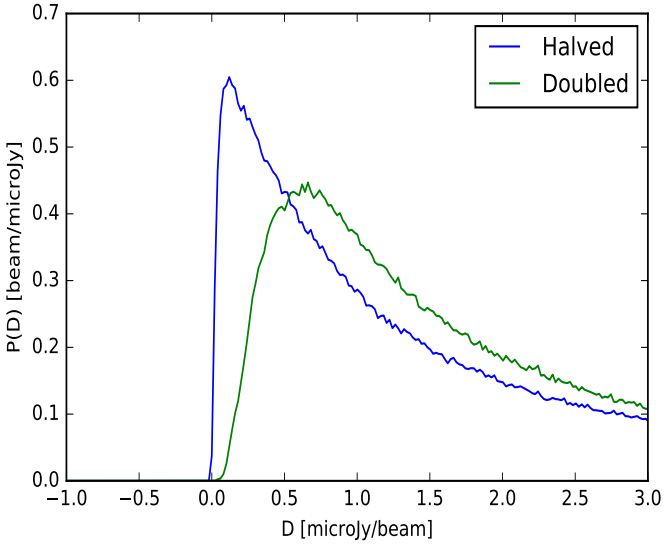

Figure 9. The probability distribution of the flux densities per beam between two different simulation runs. The cumulative source count for the smallest flux densities are either "halved" or "doubled" and are clearly labeled. Noise is included within these $P(D)$ distributions.

From Fig. 9 there is a distinct difference in the peak of the distribution with the "halved" distribution being steeper and higher at the low flux density end in comparison to the "doubled" distribution. These probability distributions show that the source counts can only be marginally constrained below S = 0.12 μJy at 3 GHz and therefore the VLA in its C configuration can reach the sensitivity of flux density $S \approx 0.2\,\mu$Jy.

4.4. *Reducing Run Time with Fewer Sources*

Unfortunately, while the simulation succeeds in producing a realistic galaxy field at a redshift of $z = 2.9$ using realistic data sources, each run of the simulation was a great computational effort and took ≈ 78 minutes to run. By having to determine about 37,000 source coordinates per image, convolve the sources with Gaussians, and add them to other overlapping galaxies, the simulation becomes bogged down in the loop which determines all of these factors.

Therefore, to decrease the computational time of the simulation I looked into decreasing the number of sources that simulation had to run, starting from the faintest flux density sources of 1×10^8 Jy. The reasoning behind this method was to determine if the analytical results changed significantly if the faintest sources were deleted. If the resulting 2D flux density pixel arrays yielded approximately the same noisy $P(D)$ distributions then I could determine whether the faintest flux density sources were necessary for the simulation to be effective. By deleting these sources from my code I significantly decreased the computational time it took to run the code because the faintest sources are the most numerous.

The process of achieving this test was to first introduce a specified "seed" number. Seed numbers are used alongside random number generators in order to provide the user control on what number will be randomly chosen. By specifying this number I was able to create the same universe for each consequent run of the simulation. A control run was done with all the source counts and seed number included. The cumulative source count $N(> S)$ was set to zero first for sources weaker than $S = 1 \times 10^{-8}$ Jy. In the next run the cumulative source count $N(> S)$ was set to zero for sources weaker than both $S = 1 \times 10^{-8}$ Jy and $S = 1.58 \times 10^{-8}$. Each subsequent run set $N(> S)$ equal to zero for sources weaker than each increasing step in flux density seen in Appendix A. This was done six times decreasing the number of counts to exclude those $< 1 \times 10^{-7}$ Jy. It was at this point where the probability distribution became noticeably skewed away from the originally unmodified run of the simulation.

What this analysis shows us, however, is that even when $N(< 3.58 \times 10^{-8}$ Jy$) = 0$ it agreed strongly with the original probability distribution curve, including all the source counts. Therefore, the faintest source counts beyond $N(< 3.58 \times 10^{-8}$ Jy) are statistically insignificant and unnecessary for this simulation to run effectively. Ridding the simulation of these faint sources significantly decreases the run time of this simulation from ≈ 78 min to ≈ 44 min.

5. CONCLUSION

In conclusion, this thesis covers the creation of Monte Carlo simulations of 1.4 GHz and 3 GHz radio sources, the implementation of using the simulation to model realistic observational data from actual source counts instead of power-law source counts, and the analytical process of introducing instrumental noise and effects from a typical VLA observation along with testing the simulation and refining it for better time efficiency. From the approximate mean number of galaxies per flux density bin λ the various probability distributions of each run were created, detailing the percentage of the flux density sources one can expect out of the total number of sources for each respective run. In order to convert the originally noiseless runs into more accurate sources observed by the VLA in configuration C, three additions were made to the code. First, the flux density point sources were each smeared by a Gaussian with FWHM = $8''$. The second modification was to include the effect of the VLA's primary beam attenuation which comes from the receivers having a greater signal in the pointing center than at the edges. This was implemented by multiplying the noiseless sky 2D pixel flux density array by a Gaussian with FWHM = $14'$. The last effect introduced to approximate the VLA was a smearing of the noiseless probability distributions due to instrumental noise of the VLA. Finally, in order to reduce run time costs it was determined that the cumulative source count $N(< 3.98 \times 10^{-8}$ Jy) from the 1.4 GHz data can be disregarded as it does not add significant accuracy to the results and is cost ineffective.

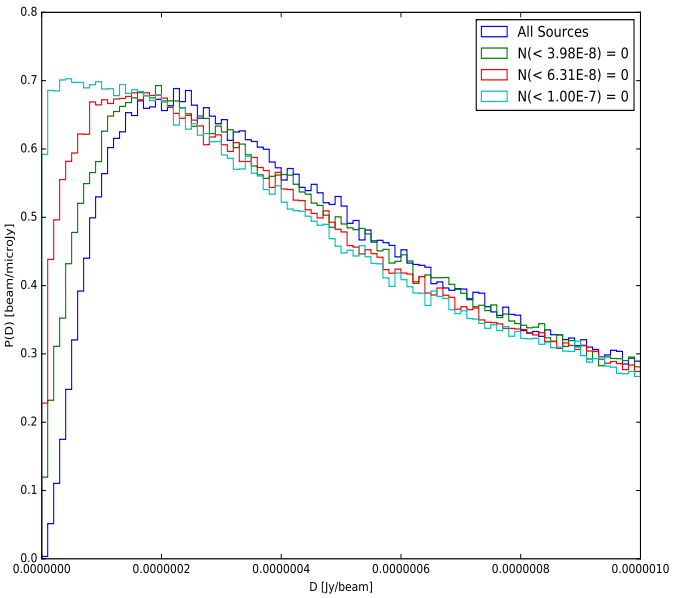

Figure 10. The probability distributions of 4 different runs of the simulation all sharing the same seed number. The range of 1.4 GHz data with cumulative source count, $N(< S) = 0$, is specified. Notice the significant deviation from the "All Sources" probability distribution and the probability distribution for "$N(< 1.00 \times 10^{-7}) = 0$".

.

6. ACKNOWLEDGEMENTS

I would like to acknowledge Dr. Jim Condon, who dedicated time in helping me complete this thesis project and who without I would not have been able to have gained the opportunity to learn more about this fascinating and meaningful topic. I would also like to acknowledge Allison Matthews and Christopher Hayes, two graduate students at the University of Virginia who helped me tremendously when it came to the specifics of implementing Python programming techniques and functions in order to get the code to work correctly and efficiently. Without them the code would have not

turned out as successful as it has. Finally, I would like to thank the Astronomy Department, for without its available resources and helpful faculty I would not have been able to complete the culmination of my undergraduate career.

REFERENCES

Buat, V., Marcillac, D., Burgarella, D., et al. 2007, A&A, 469, 19

Condon, J. J. 1974, ApJ, 188, 279

—. 1992, ARA&A, 30, 575

Condon, J. J., Cotton, W. D., Fomalont, E. B., et al. 2012, ApJ, 758, 23

Feldmann, R., Hopkins, P. F., Quataert, E., Faucher-Giguère, C.-A., & Kereš, D. 2016, MNRAS, 458, L14

Jarvis, M., Seymour, N., Afonso, J., et al. 2015, Advancing Astrophysics with the Square Kilometre Array (AASKA14), 68

Madau, P., & Dickinson, M. 2014, ARA&A, 52, 415

Matthews, A. 2018, ApJ

APPENDIX

A. 1.4 GHZ SOURCE COUNTS

Table 1. 1.4 GHz source counts retrieved from Condon et al. (2012)

S [Jy]	$S^{5/2}n(S)$ [Jy$^{3/2}$/sr]	$N(>S)$ [sr^{-1}]
1.00E+03	78.909	0
6.31E+02	80.016	0
3.98E+02	81.507	0
2.51E+02	83.611	0
1.58E+02	86.639	0
1.00E+02	91.031	0.1
6.31E+01	97.38	0.1
3.98E+01	106.46	0.3
2.51E+01	119.201	0.6
1.58E+01	136.581	1.3
1.00E+01	159.376	3
6.31E+00	187.737	6.8
3.98E+00	220.635	16
2.51E+00	255.359	37.1
1.58E+00	287.398	85.1
1.00E+00	311.08	189.9
6.31E-01	321.058	409
3.98E-01	314.167	844.5
2.51E-01	290.721	1664.2
1.58E-01	254.473	3124.6
1.00E-01	211.235	5591.3
6.31E-02	166.98	9555.4
3.98E-02	126.345	15643.8
2.51E-02	92.012	24628
1.58E-02	64.864	37435.3
1.00E-02	44.526	55173.2
6.31E-03	29.964	79192.1
3.98E-03	19.947	111243.6
2.51E-03	13.316	153884.4
1.58E-03	9.114	211500.2
1.00E-03	6.603	292842.7
6.31E-04	5.235	416983.3
3.98E-04	4.59	626102.4
2.51E-04	4.329	1009557.2
1.58E-04	4.172	1741466.6
1.00E-04	3.921	3125746
6.31E-05	3.489	5631486.5
3.98E-05	2.908	9899998
2.51E-05	2.271	16719941
1.58E-05	1.675	26985924
1.00E-05	1.176	41660408
6.31E-06	0.793	61749244
3.98E-06	0.518	88290704
2.51E-06	0.329	122353744
1.58E-06	0.205	165041696
1.00E-06	0.125	217498592
6.31E-07	0.075	280917184
3.98E-07	0.045	356548064
2.51E-07	0.026	445710112
1.58E-07	0.015	549801344
1.00E-07	0.009	670309760
6.31E-08	0.0051	808823296
3.98E-08	0.0029	967037760
2.51E-08	0.0017	1146761984
1.58E-08	0.0009	1349920640
1.00E-08	0.0005	1578557696

B. DICTIONARY OF VARIABLES

Variable	Units	Definition
α		Gaussian synthesized beam power pattern with FWHM = 8″ (Eq. 11)
λ		The average number of galaxies per flux density bin per image (Eq. 7)
FWHM		Full Width between Half Maximum points
$N(>S)$	sr^{-1}	Cumulative source count: the mean number of sources per steradian stronger than S (Eq. 3)
$n(S)$	$\mathrm{sr}^{-1}\mathrm{Jy}^{-1}$	Differential source count (Eq. 2)
$n^*(S)$		Differential source count per logarithmic flux density range (Eq. 4)
$P(D)$	beam $\mu\mathrm{Jy}^{-1}$	Confusion probability distribution of "deflections" D (μJy beam^{-1})
$P_p(n\|\lambda)$		Poisson probability that there are n sources per bin in a given simulation (Eq. 8)
r	arcsec	The radial offset from the galaxy's center (Eq. 12)
$S^2n(S)$	Jy sr^{-1}	Brightness-weighted differential count (Eq. 10)
SFHU		Star Formation History of the Universe
SFRD	$M_\odot$ yr^{-1} Mpc^3	Star formation rate density
$S(r)$	Jy	The flux density for a single galaxy as a function of radial offset r (Eq. 13)
t	Gyr	Lookback time: time elapsed between light detection on Earth and original light source emission
VLA		Very Large Array
z		Redshift

C. MONTE CARLO UNIVERSE SIMULATION

Below details specifically how I coded the Monte Carlo simulation described in the thesis. It should be noted that there are many ways to achieve the same resulting program, but the choices I made suited my coding preferences. The simulation was created using the 1.4 GHz data seen in Appendix A and the python coding language.

To start, the following python packages will be used along with associated abbreviations.

```
import matplotlib.pyplot as plt
import numpy as np
import scipy.misc
import time
from scipy import interpolate
from scipy.interpolate import interp1d
import pandas as pd
```

Briefly, the **matplotlib.pyplot** plotting package associated with Python allows the user to create various types of plots, the **numpy** package allows the user access to predefined mathematical functions and support for large, multi-dimensional arrays, the **scipy** package gives the user access to more defined scientific and technical computing abilities, the **time** package can determine the timestamp at specific parts in the code, and finally the **pandas** package allows the user predefined functions which help with reading comma separated values (.csv) files. After specifying which python libraries, it is necessary to read in the 1.4 GHz data (from Appendix A) in order to use it for calculation and creating the simulation. For this part I use the **pandas** python package to read in the different columns in "sourcedata.csv" (i.e. the 1.4 GHz data), where S is the source flux densities in Janskys, $S5\_2$ is equal to $S^{5/2}n(S)$, and N is the cumulative source count $N(>S)$. The function *np.array*() makes makes each of the three columns of values seen in Appendix A data arrays.

```
#reading in the actual source data
df = pd.read_csv('sourcedata.csv')
```

```
S = np.array(df.S) #flux densities in Jy
S5_2 = np.array(df.S5_2) #Flux Densities^(5/2)*n(S), where n(S) is the differential
    source count. Units = Jy^(3/2)/sr
N = np.array(df.N) #the number of sources per sr stronger than S
```

Now that we have all of the necessary 1.4 GHz data in arrays we can complete the first task of verifying the expected results with the power law approximation described in Section 2.2. First we find the power law approximation of the differential source count $n(S)$ by using the equation $n(S) = 300S^{-2}$ and the differential source count per logarithmic flux density range $n^*(S)$ is defined by Eq. 6. To find the average number of galaxies per flux density bin λ, Eq. 7 was used. Therefore, *powerlaw_numperbin* is an array of the mean number of galaxies at each flux density S. Since we want to create different universes for each run of the simulation with similar λ we can use the Poisson function seen in Eq. 8 to get a random sampling of λ. This is done using the predefined numpy function *np.random.poisson*(), which takes in the value $lam = \lambda$ and *size* is set to the default of "None". For convenience the total number of sources can be found for each run using *np.sum*(), which sums every value in a predefined array. The value of *axis* was set to its default of "None" at each flux density.

```
#using the power law approximation where k = 300 and alpha = 2
powerlaw_n = 300*S**(-2)
powerlaw_nstar = 300*np.log(10)*S**(-1) #the differential source count n*(S) per
    logarithmic flux density
powerlaw_numperbin = powerlaw_nstar*.01*(1000/(3600*180/np.pi))**2 #average number of
     galaxies per flux density bin
powerlaw_numsources = np.random.poisson(lam = powerlaw_numperbin, size = None) #
    number of sources within a bin
sumof_plnumsources = np.sum(powerlaw_numsources, axis = None)
```

Using the power-law approximation is important in order to verify and test my simulation of realistic sources to previous calculations using power-law sources by Condon et al. (2012). However, for a more realistic simulation of the universe at a redshift of $z = 2.9$ we must use the actual source counts to determine the differential source counts. We can do this by calculating $n(S)$ from columns 1 and 2 of Appendix A, S and $S^{5/2}n(S)$, by: $\frac{S^{5/2}n(S)}{S^{5/2}}$. Then using Eq. 10 I found the actual differential source count per logarithmic flux density range, $n^*(S)$. This array of differential source counts was used to once again calculate the average number of galaxies per flux density bin λ, with Eq.7. The predefined python functions of *np.random.poisson*() and *np.sum*() were once again used to create different universes for each run of the simulation with similar λ and sum the total number of galaxies within the universe run.

```
#using actual data to determine n(S)
actual_n = S5_2/(S**(5./2.)) #actual differential source count
actual_nstar = actual_n*S*np.log(10)
actual_numperbin = actual_nstar*.01*(1000./(3600.*180./np.pi))**2 #avg number of
    galaxies per flux density bin for actual sources
sample_poisson = np.random.poisson(lam = actual_numperbin, size = None)
meannum = np.sum(actual_numperbin, axis = None)
```

However, this is not what is used in the final simulation. The given 1.4 GHz data in Appendix A samples too coarsely the intrinsic population, therefore in order to make it finer I used the cubic spline interpolation seen in Eq. 1 in order to go from from every 0.2 in log flux density to 0.01 in log flux density. First I defined two arrays using the numpy function *np.arange*(*start*, *stop*, *step*), which generates values with the half-open interval [start, stop) with an input step size. The first array, *xold*, defines the original flux density range with a spacing of 0.2 in log flux density and the second array, *xnew*, defines the same flux density range, but with the desired spacing of 0.01 in log flux density. The predefined function *interpolate.interp*1*d*() takes in the original spacing and λ values and then executes the interpolation with the new spacing of 0.01 in log flux density. Note that the "stop" value is $-8.0+step$ because of the fact that *np.arrange*() creates values on a half-open interval. Finally, the *np.random.poisson*() function was used again to create a Poisson generated version of the final λ array determined by the actual realistic source counts from the 1.4 GHz data. The *np.sum*() function can be used to determine the final total number of galaxies being simulated. The variable *power_xnew* defines the *xnew* array converted from log-to-linear-based flux densities in units of Jy.

```
#the following uses a spline interpolation function to fit the actual_numperbin
    results to a new x-array with a spacing of .01 in log flux density
```

```
xold =  np.arange(3.,-8.2, -.2) #old array of log flux densities with spacing of .2
xnew = np.arange(3., -8.01, -.01) #new array of log flux densities with spacing of
    .01
f = interpolate.interp1d(xold, actual_numperbin)
avg_sources = f(xnew) #mean number of sources per bin
#np.random.seed(0) #by placing the seed here it fixes the number of sources per bin
    and also fixes the galaxy postions
numgalaxies = np.random.poisson(lam = avg_sources, size = None)
sumof_numgalaxies = np.sum(numgalaxies, axis = None)
power_xnew = 10**xnew
```

Next, we move onto the creation of the 2D pixel data array which will be used to create our contour plots later on, which is basically an image of the simulated universe! Using the *np.arange*() function again, I start by defining the x- and y- dimensions of the array so that it is 1000 pixels across for both. I then set up three 2D arrays with the *np.zeros*(*shape*) function, which sets the value for every index in the array as 0 across for the total shape. The arrays are:*xpix_array*, *ypix_array*, and *flux_array*. The *xpix_array* and *ypix_array* will be filled in the following steps so that they can define a central galaxy coordinate on the 1000 pixels^2 image. The *flux_array* will be filled with the flux densities of each Gaussian point source within λ, with overlapping pixel locations summing together. For-loops were used to fill the values within the *xpix_array* and *ypix_array* so that *xpix_array* increases from 0 to 1000 from left to right and *ypix_array* increases from 0 to 1000 from top to bottom.

```
xpix = np.arange(0, 1001, 1)
ypix = np.arange(0, 1001, 1)

xpix_array=np.zeros((1001, 1001))
ypix_array=np.zeros((1001, 1001))
flux_array=np.zeros((1001, 1001))

for i in range(len(xpix)):
    for j in range(len(xpix)):
        xpix_array[i,j]=xpix[j]

for i in range(len(ypix)):
    for j in range(len(ypix)):
        ypix_array[j,i]=ypix[j]
```

The majority of the computation of the simulation comes down to the next created function and for-loop. First, I defined a 2D function named *gauss2d*() to calculate the decreasing flux density with increasing radial offset r^2 from the center flux density peak, as defined in Eq. 13. This function takes in the flux density array, the x-pixel array, the y-pixel array, the central flux density in Jy, and the randomized galaxy x- and y-positions. The randomized galaxy positions are created using *np.random.uniform*(), which selects a random value from a uniform distribution between a set range. The range was chosen specifically to create an equal chance of the galaxies being placed at any position on the simulated sky. The for-loop cycles through each value within λ (*numgalaxies*) and determines a randomized galaxy x- and y-coordinate. Each source and its coordinates are fed into the *gauss2d*() function which calculates and adds the Gaussian point source to the 2D 1000 pixels^2 array. The *gauss2d*() function adds each source and eventually yields the final universe in a 2D array.

```
def gauss2d(total_flux,xpix_array,ypix_array,gal_flux,galxpos,galypos):
   FWHM = 8.
    r_2 = (xpix_array-galxpos)**2 + (ypix_array-galypos)**2
    flux_add = gal_flux*np.exp(-4.*np.log(2.)*r_2/(FWHM**2))
    flux_add[r_2>17.**2] = 0.
    total_flux+= flux_add
    return total_flux

for k in range((len(numgalaxies))):
    if numgalaxies[k] != 0:
```

```
    for m in range(numgalaxies[k]):
        galxpos = np.around(np.random.uniform(0.0, 1.00001, size = None),
            decimals = 3, out = None)*1000
        galypos = np.around(np.random.uniform(0.0, 1.00001, size = None),
            decimals = 3, out = None)*1000
        #print galxpos, galypos
        flux_array = gauss2d(flux_array, xpix_array, ypix_array, power_xnew[k],
            galxpos, galypos)
```

The last step to the simulation is to create a usable data file containing the 1000 arcsecond^2 pixel array. With the **pandas** python package we can set the 2D flux density array as a data frame and then covert it to a comma separated file by:

```
import pandas as pd
my_df = pd.DataFrame(flux_array)
my_df.to_csv('universe.csv', index=False, header=False)
```

Now that we have successfully rendered a 2D "universe" using realistic sources we can make our 2D array of flux density values into visual plots. The remainder of the code I present from here will be on how I specifically created the plots, added in the noise obtained in a typical VLA observing with configuration C, and analyzed the results.

D. ANALYSIS OF SIMULATION

Before beginning the analysis part of the simulation it is important to introduce first more python packages that were used during this section that were not part of the last section. These are:

```
import matplotlib.pyplot as plt
import matplotlib.colors as colors
from matplotlib.ticker import LogFormatter
```

Most of this appendix will detail how I created the analytical plots seen within the thesis so that one can feasibly recreate the same results on their own with similar data. It should be noted that once again the method of plotting used here is only one of many and was deemed most efficient for the results I wanted to obtain.

The last step in Appendix C was to convert the pandas data frame into a .csv file. Using the pandas function *pd.read_csv*() we can read in the 2D flux density array and then with the function *np.array*() we can covert the values within the array to ones that can be used with python functions.

```
df = pd.read_csv('universe.csv', header = None)
Z = np.array(df)
```

We can now use the created arrays of various runs of the simulation to create the 2D contour plots seen in Appendix D. The process is simply to use the *ax.contour*() function which reads three 2D arrays, the xpix_array, the ypix_array, and the flux density array we just obtained from the .csv file. I created the contour plots using both a linear and logarithmic scaling in flux density. I did this by creating an array called "levels" which specifies the steps at which the user wants the contours to be marked. Therefore to plot a linear plot I did:

```
fig = plt.figure(figsize = (10, 8))
ax = fig.add_subplot(111)
levels = np.arange(0.00001, 0.00009, 0.00001)
im = ax.contour(xpix_array, ypix_array, Z, levels, linewidths = .1)
ac = fig.colorbar(im, ax=ax)
formatter = LogFormatter(10, labelOnlyBase=False)
plt.savefig('linear.pdf', bbox_inches='tight', format = 'pdf')
```

And to create a logarithmically scaled plot I did:

```
fig = plt.figure(figsize = (10, 8))
ax = fig.add_subplot(111)
levels = [10.**(-5), 10.**(-4), 10.**(-3), 10.**(-2), 10.**(-1), 1.0 ]
```

```
im = ax.contour(xpix_array, ypix_array, Z, levels, norm = colors.LogNorm(),
    linewidths = .1)
ac = fig.colorbar(im, ax=ax)
formatter = LogFormatter(10, labelOnlyBase=False)
plt.savefig('universe_log.pdf', bbox_inches='tight', format = 'pdf')
```

For stylistic purposes I incorporated extra additions to some of the functions, but do not plan to go into detail. The python reference library defines in great detail many of the stylistic parameters I have included. The function *plt.savefig*() creates a .pdf image of your 2D contour plot, however this has not incorporated the noise and instrumental effects of VLA observations and imaging. In order to obtain a noiseless probability distribution of the each universe run we much first remove the part of the array which is not correct as described in Sect. 4.1. Then by using a function which "flattens" the 2D array into a 1D array we can bin each value into a histogram of bin width $d = 0.01\mu$Jy. It is important to apply an array of weights with the same length as the flattened array in order to contribute its relative weight towards the bin count. This ensures that the integral of the distribution remains 1. To create the weighted array I used the function *np.ones*(*shape*) which makes an array of ones for the user defined shape of a specified shape. This shape is the length of the flattened 2D contour array. Thus since there is 100 bins per μJy we can multiply the bins by 100 to convert it to per μJy. Therefore the normalized probability per bin is the number per bin divided by the length of the flattened array which took into account the error of the original 2D array's border.

```
Znew = Z[16:984,16:984]
A = Znew.flatten()
length = len(A)
weighted_value = np.ones(937024)*100./937024.
fig = plt.figure(figsize = (10, 8))
ax = fig.add_subplot(111)
ax.hist(A, bins=np.arange(0.0, 5.0*10**-6 + .01*10**-6, .01*10**-6), histtype='step',
     weights = weighted_value)
plt.xlim(0.0, 2.0*10**-6)
plt.savefig('pd-diagram.pdf', bbox_inches='tight', format = 'pdf')
```

The bins range from 0 to 5 μJy in order to see the distribution across the faintest flux density sources.

Now we can add in the instrumental effects along with the noise obtained from the VLA. The following steps detail how to include the noise from the VLA within the probability distribution. For convenience I first converted the flattened array to units of μJy to avoid having to convert within the rest of the code. Using the function *np.histogram*()[0] I took the converted array and chose the bins to range from -10 to 10 with 1000 bins total. The *np.histogram*()[0] function returns two values and since we only care about the first value I made sure that was the only value being returned by using "[0]". Next, I created a dummy array which will be necessary to perform the convolution on the probability distribution in order to incorporate noise into the noiseless distribution. Since we are doing this convolution numerically, the dummy array must be the same length as the chosen histogram. After creating the dummy array I then created a user defined function named *gauss1d*() which reads in both the dummy array and the user specified FWHM. The function returns a 1D Gaussian which can be convolved with the probability distribution.

```
choice = A
convertedhist = choice*10**6 #convert to microJy realm
hist_choice = np.histogram(convertedhist,bins= np.linspace(-10, 10, 1000), density =
    True)[0] #np.histogram returns 2 values only need first one
lenhistchoice = len(hist_choice)
dummy_array = np.linspace(-10, 10, lenhistchoice)

def gauss1d(dummy_array, FWHM):
    rms = FWHM/2.355
    return np.exp(-.5*(dummy_array/rms)**2)/(rms*np.sqrt(2.*np.pi))
```

Now that we have the dummy array and a function which can return a 1D Gaussian at a specified FWHM we can determine the Gaussian noise which we can then convolve with the chosen probability distribution using the function *np.convolve*(). Since *np.convolve*() uses discrete convolution it needs to know the step size of the dummy array. Therefore an additional factor is

multiplied with the convolution in order to obtain the correct result. Finally using simple plotting methods we can obtain the results seen in Fig. 8.

```
noise = gauss1d(dummy_array, 1.2) #FWHM from Jim's paper
convolved_choice = np.convolve(hist_choice, noise, mode = 'same')*(dummy_array[2]-
    dummy_array[1]) #np.convolve works with discrete convolution and it needs to know
    the differences in the steps
plt.plot(dummy_array, convolved_choice)
plt.plot(dummy_array, hist_choice)
plt.xlim(-4,4)
#plt.savefig('pd-convolution.pdf', bbox_inches='tight', format = 'pdf')
```

In order to implement the effect of the VLA's primary beam attenuation it requires simply for us to multiply the 2D flux density array with an array of the same shape and a 2D Gaussian with a FWHM = 14′ centered in the middle of the array. This is achieved by using the same *gauss*2*D*() function I created before, but instead of inputting random x- and y-coordinates I specified the center of the array. This resulting array can then simply be multiplied with the 2D flux density array from any simulation run to create a simulated VLA image. All other plots not described here used the same plotting methods and can be determined in a similar manner. Therefore, this concludes the implementation and plotting instructions.

E. ENLARGED 2D CONTOUR PLOTS

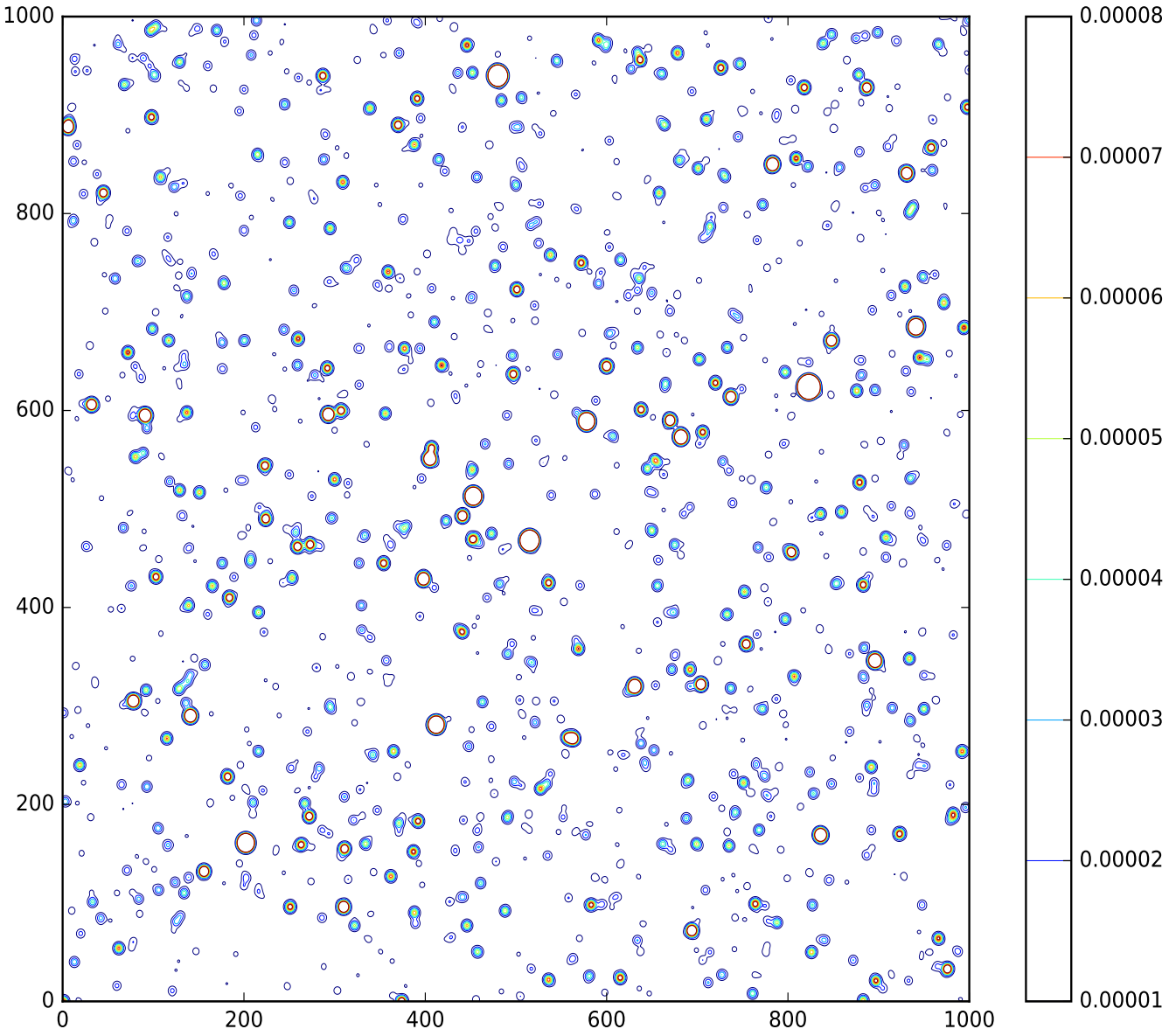

Figure 11. A 2D representation of one universe run based on the 1.4 GHz data from Condon et al. (2012) in the linear scale. Contour lines mark different levels of flux density in units of Jy.

.

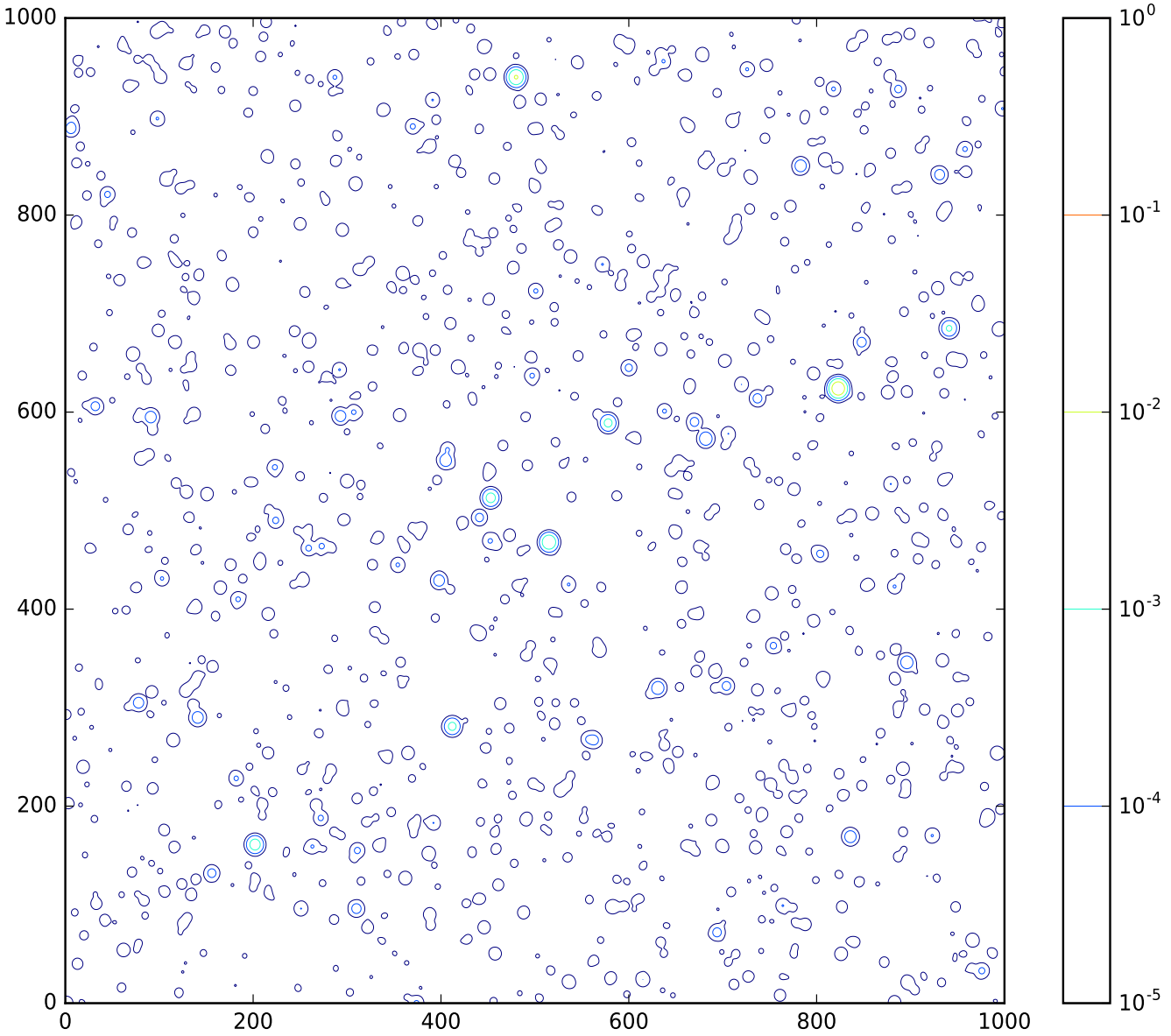

Figure 12. A 2D contour representation of the same universe from 11, but represented logarithmically.

.

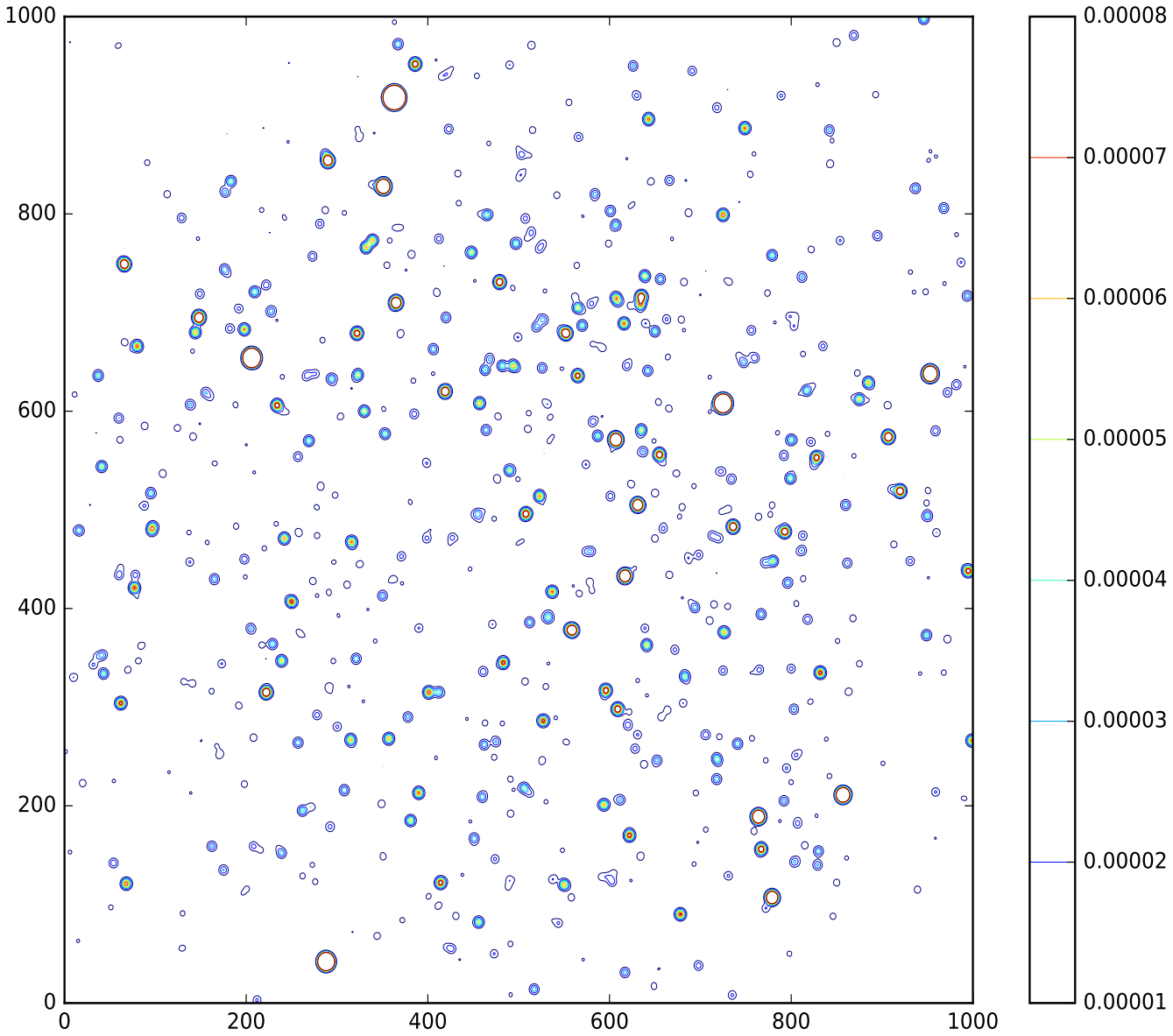

Figure 13. A 2D linear contour plot of the same universe from Figures 11 and 12 by including the effect of the primary beam on the VLA with a FWHM = $14'$.