Synthetic Polarimetry and Velocity Moments: Athena Simulations

Patrick King

October 7, 2016

1 Overview

- The pipeline produces column integrations of the Stokes parameters Q and U; the inclination-corrected column density N_c (and M_0); and the density-weighted velocity moments M_1 and M_2. A simple density cutoff (5000 cm^{-3}) is used, so any simulation cells with a density below the threshold do not contribute to the calculation of these quantities.
- The line-derived quantities (the moments M_0, M_1, and M_2) are computed according to the NRAO CASA definition, under the assumption that the (line) intensity is related linearly to the gas density ($I \propto \rho$); in other words, the lines are assumed optically thin and in LTE. In particular, the M_2 is not simply the LOS standard deviation of the velocity, since the standard deviation is weighted linearly, not quadratically, by the intensity.
- These observables are then smoothed to the desired resolution, for a specified width gaussian beam given in parsecs. Both the continuum-derived quantities (Q, U, and N_c) and the line-derived quantities (M_0, M_1, and M_2) are smoothed to the same resolution.
- The polarimetric quantities p (the polarization fraction) and χ (the plane-of-sky inferred magnetic field angle) are computed, as well as the plane-of-sky gradients in the velocity moments and the polarization angle dispersion function S. These gradients are computed at the appropriate length scale (the beamwidth), corresponding to a gaussian derivative.
- The pipeline then produces maps of these quantities. The beam is annotated in the lower left corner when applicable (that is, not for the simulation resolution.)
- Next the pipeline samples the maps (Nyquist sampling) such that each beam, roughly, has two points in each dimension per beam, or 4 points total. This is done adaptively with an arbitrary beam specified by the user. Then these sampled points are plotted in the 2d-histogram scatter plots, and a simple linear regression (in log-log space) is plotted with the inferred power-law index and the r^2 coefficient of determination for the fit annotated. Nyquist sampling is not conducted at simulation resolution.
- Finally, the data - Q, U, N_c, M_1, M_2, p, S, and the POS gradients of M_0, M_1, M_2, p, Q, and U - are optionally output in a FITS file as Image HDUs.
- An optional *emmisivity* field has been implemented that modifies the polarization efficiency as a function of (gas) density. The emmisivity is set to 1 (no dependence) for the plots in this document, but a power-law index can now be simply implemented.

2 1 pc Box

2.1 Maps

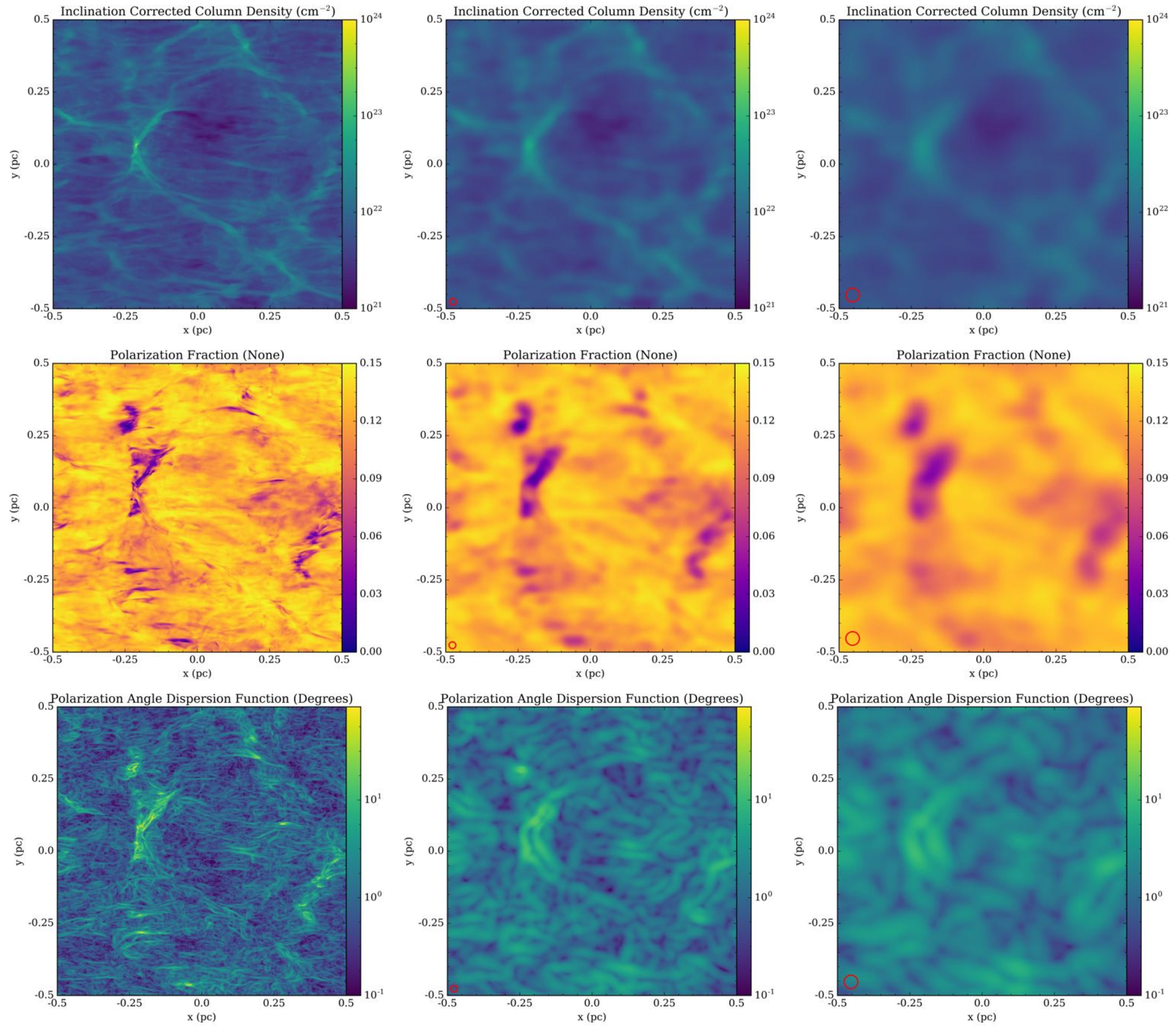

Figure 1: Polarimetric maps for the 1 pc box simulation. Columns: 1 is simulation resolution (0.002 pc); 2 is 0.0125 pc resolution; and 3 is 0.025 pc resolution. Rows: 1 is N_c; 2, p; 3 S.

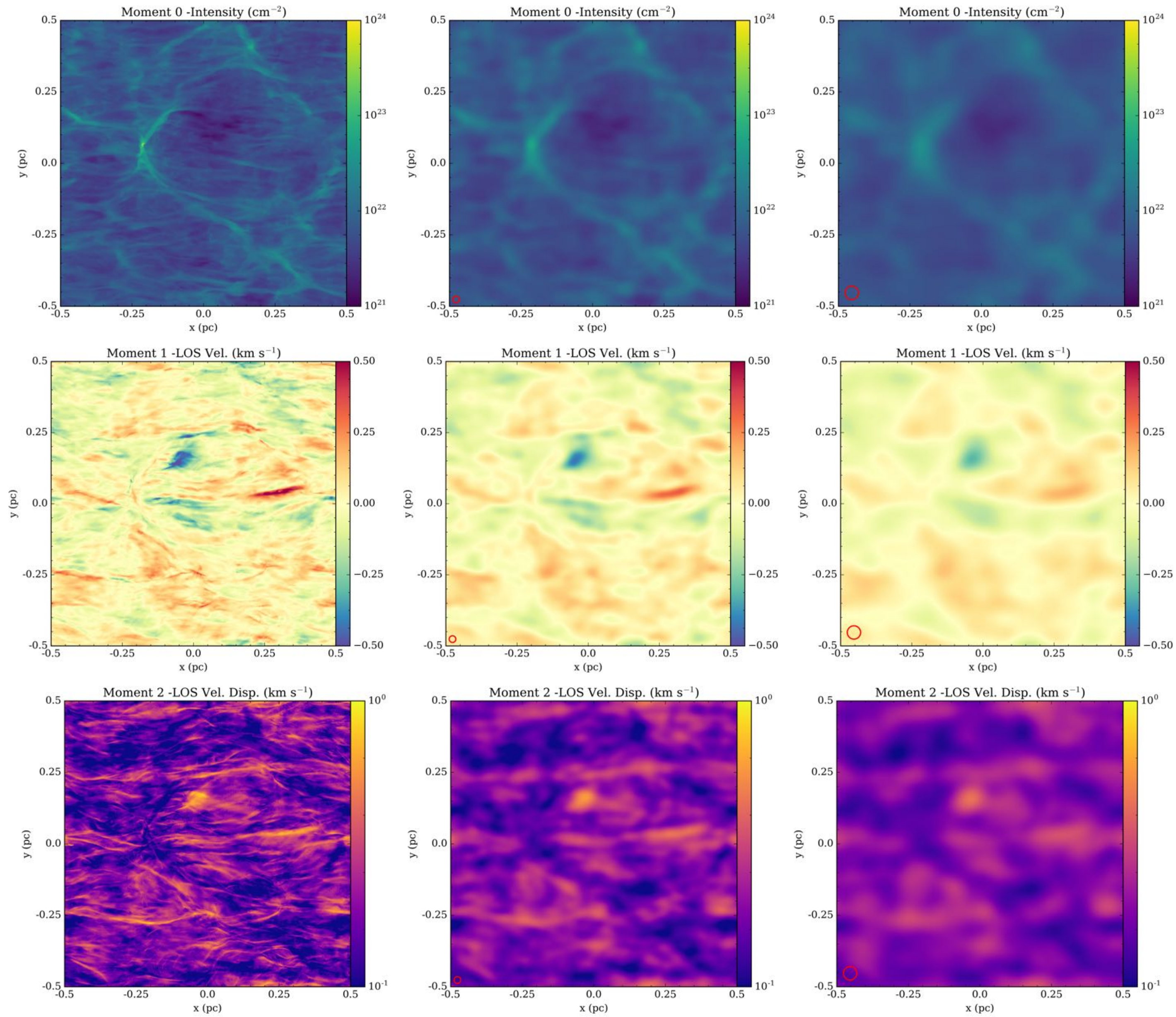

Figure 2: Moment maps for the 1 pc box simulation. Columns: 1 is simulation resolution (0.002 pc); 2 is 0.0125 pc resolution; and 3 is 0.025 pc resolution. Rows: 1 is M_0; 2, M_1; 3 M_2.

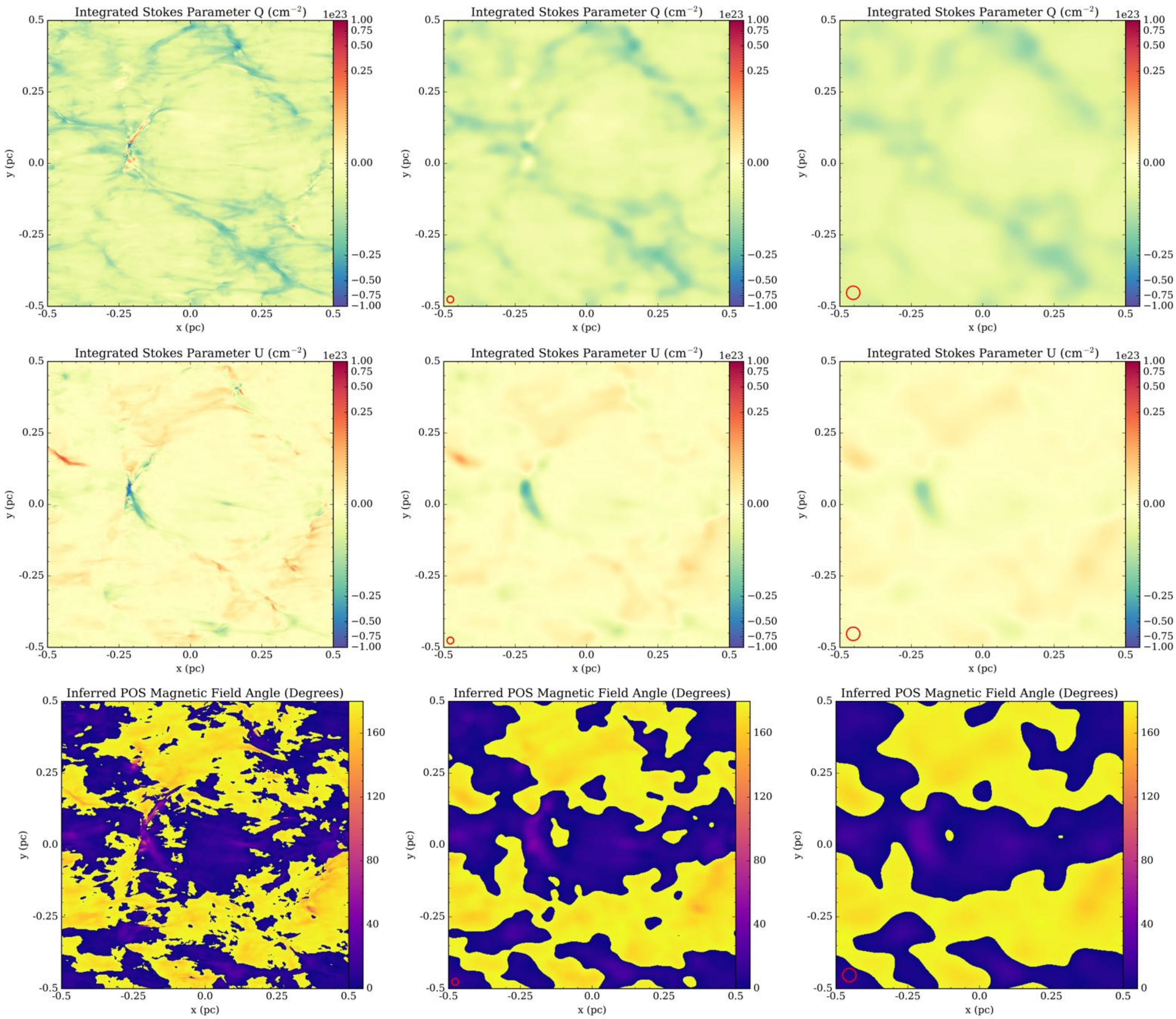

Figure 3: Further polarimetric maps for the 1 pc box simulation. Columns: 1 is simulation resolution (0.002 pc); 2 is 0.0125 pc resolution; and 3 is 0.025 pc resolution. Rows: 1 is Q; 2, U; 3 ψ.

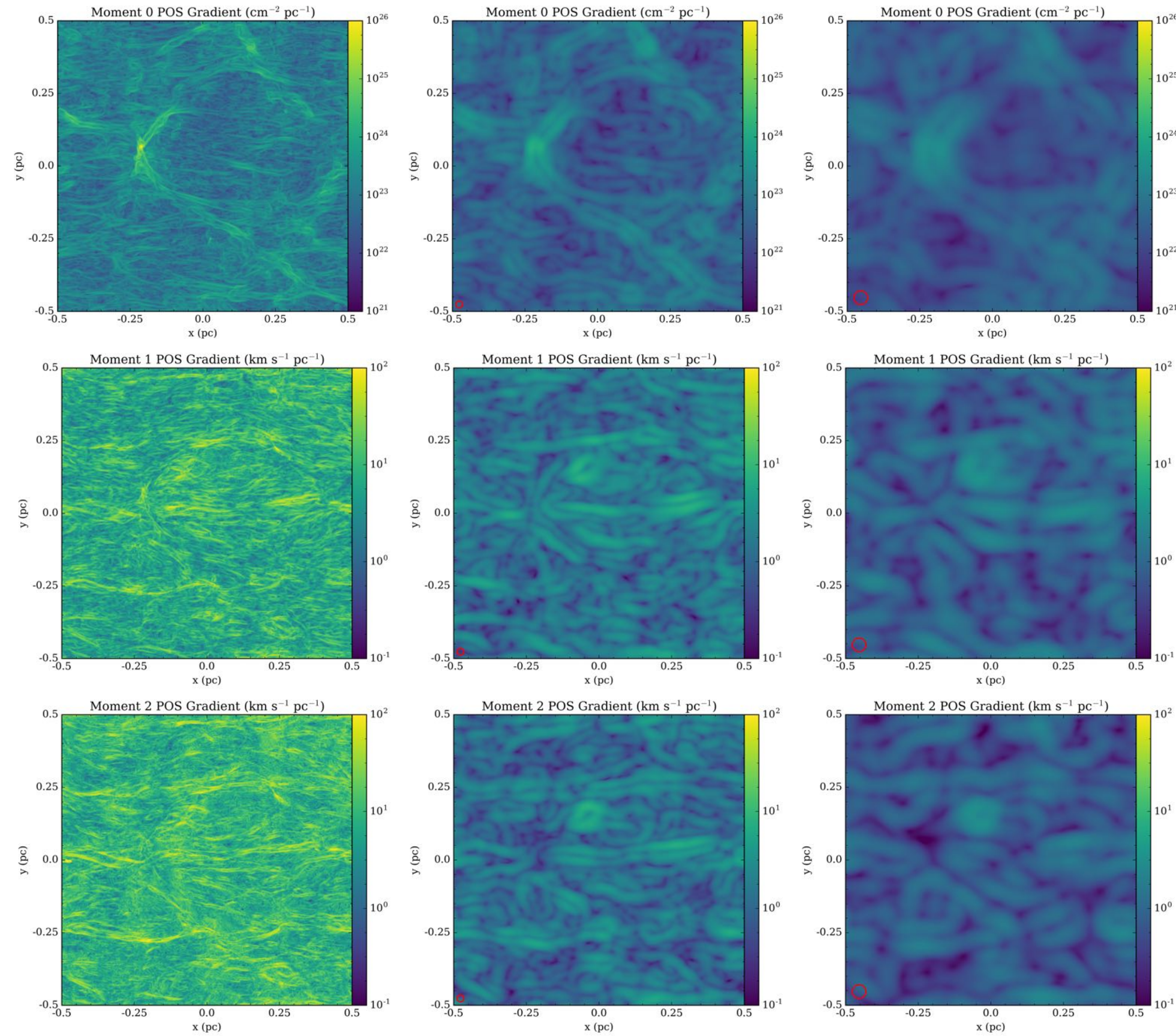

Figure 4: POS gradient moment maps for the 1 pc box simulation. Columns: 1 is simulation resolution (0.002 pc); 2 is 0.0125 pc resolution; and 3 is 0.025 pc resolution. Rows: 1 is ∇M_0; 2, ∇M_1; 3 ∇M_2.

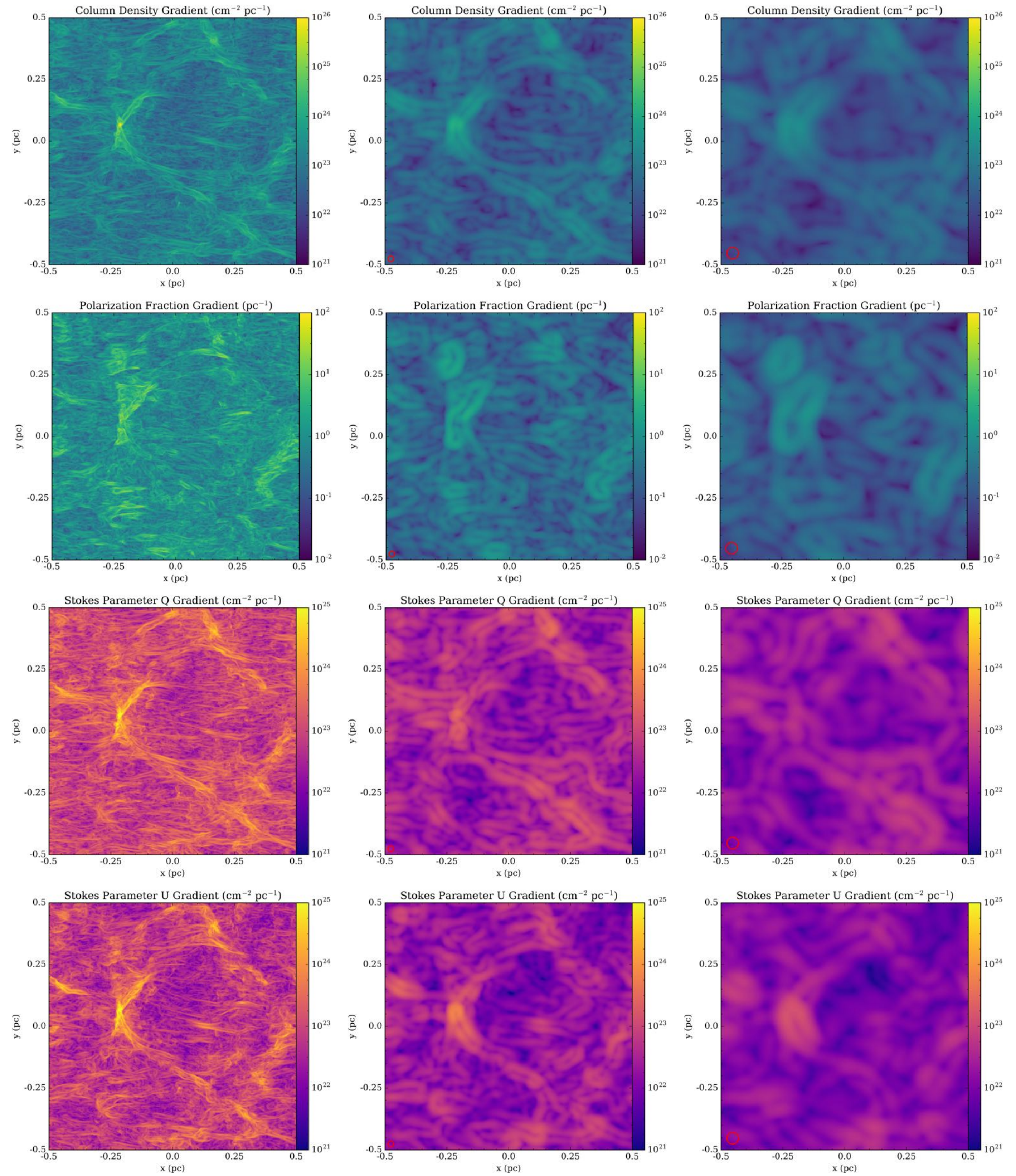

Figure 5: POS gradient polarimetric maps for the 1 pc box simulation. Columns: 1 is simulation resolution (0.002 pc); 2 is 0.0125 pc resolution; and 3 is 0.025 pc resolution. Rows: 1 is ∇N_c; 2, ∇p; 3, ∇Q; 4, ∇U.

2.2 Scatter Plots

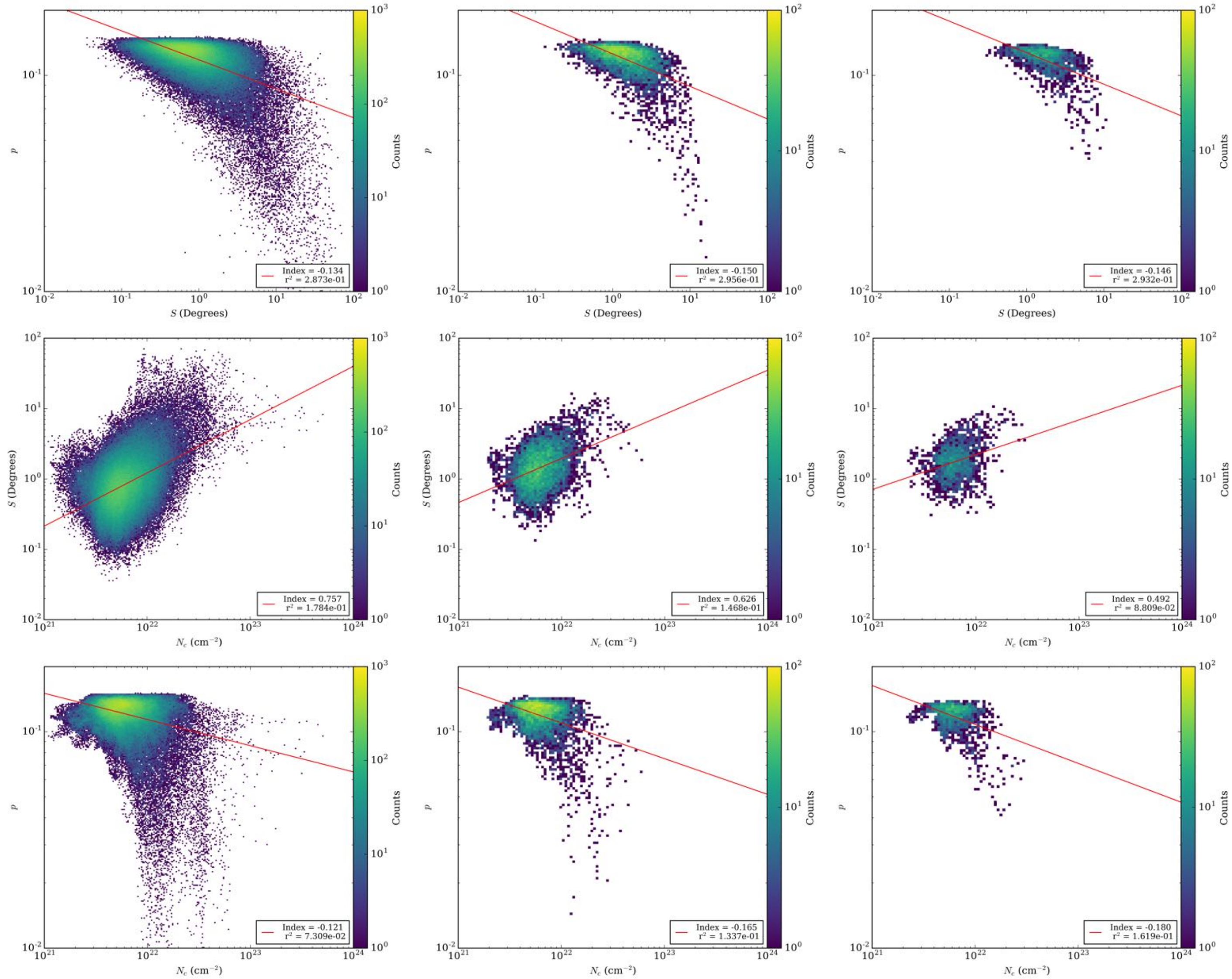

Figure 6: Polarimetric scatter plots for the 1 pc box simulation. Columns: 1 is simulation resolution (0.002 pc); 2 is 0.0125 pc resolution; and 3 is 0.025 pc resolution. Rows: 1 is p vs. S; 2, N_c vs. S; 3, p vs. N_c.

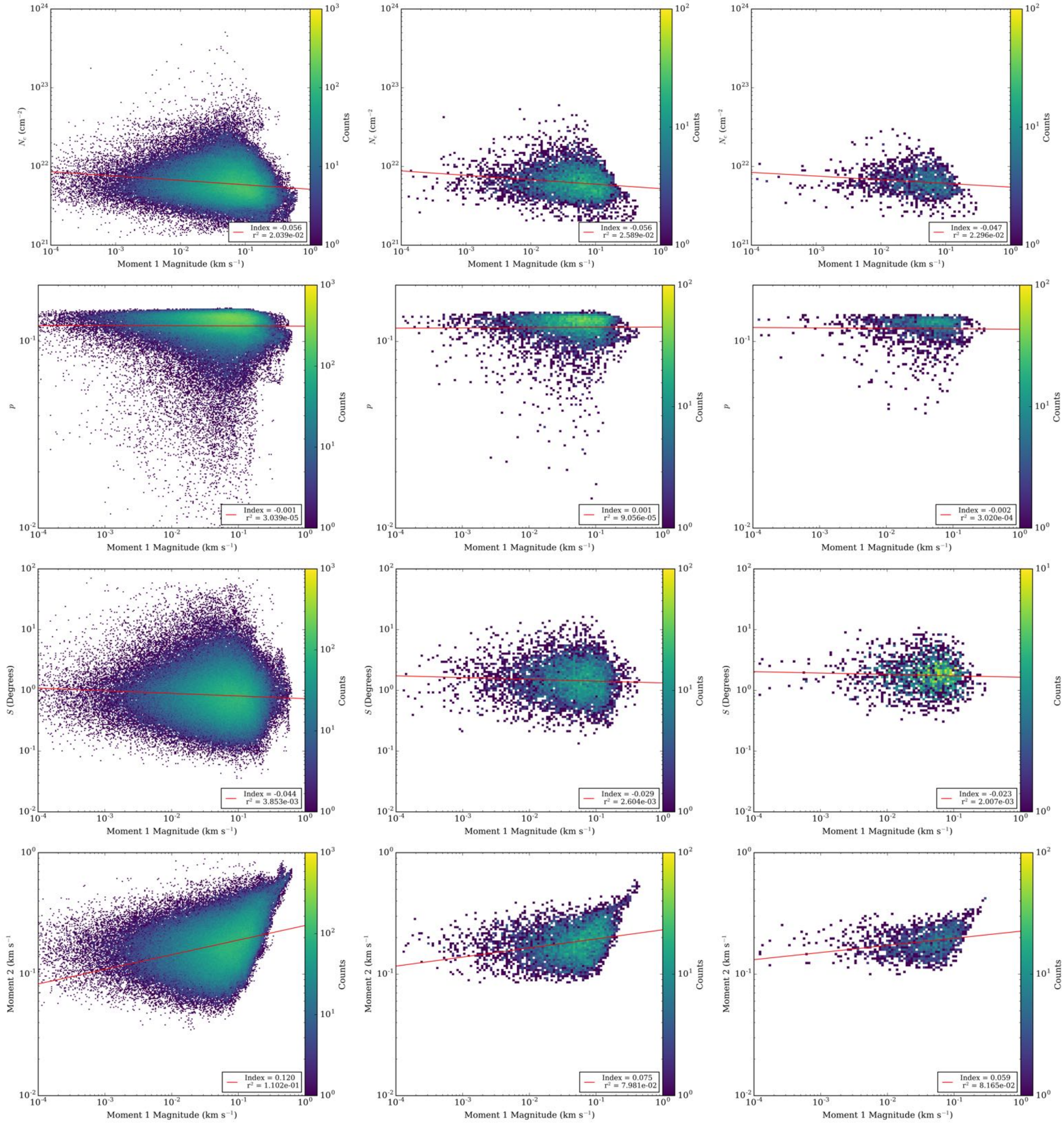

Figure 7: Magnitude of moment 1 scatter plots for the 1 pc box simulation. Columns: 1 is simulation resolution (0.002 pc); 2 is 0.0125 pc resolution; and 3 is 0.025 pc resolution. Rows: 1 is $|M_1|$ vs. N_c; 2, $|M_1|$ vs. p; 3, $|M_1|$ vs. S; 4, $|M_1|$ vs M_2.

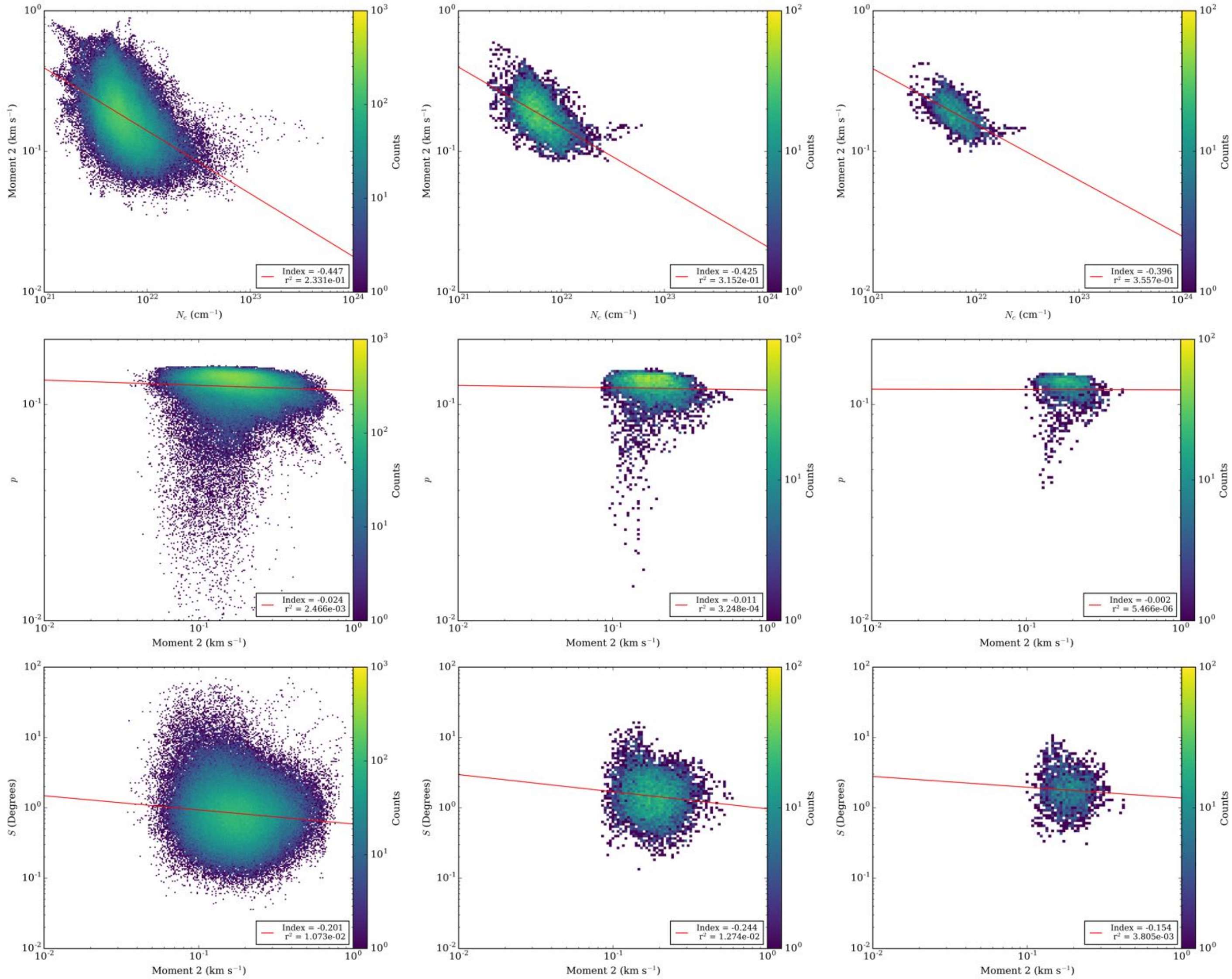

Figure 8: Moment 2 scatter plots for the 1 pc box simulation. Columns: 1 is simulation resolution (0.002 pc); 2 is 0.0125 pc resolution; and 3 is 0.025 pc resolution. Rows: 1 is N_c vs. M_2; 2, M_2 vs. p; 3, M_2 vs. S.

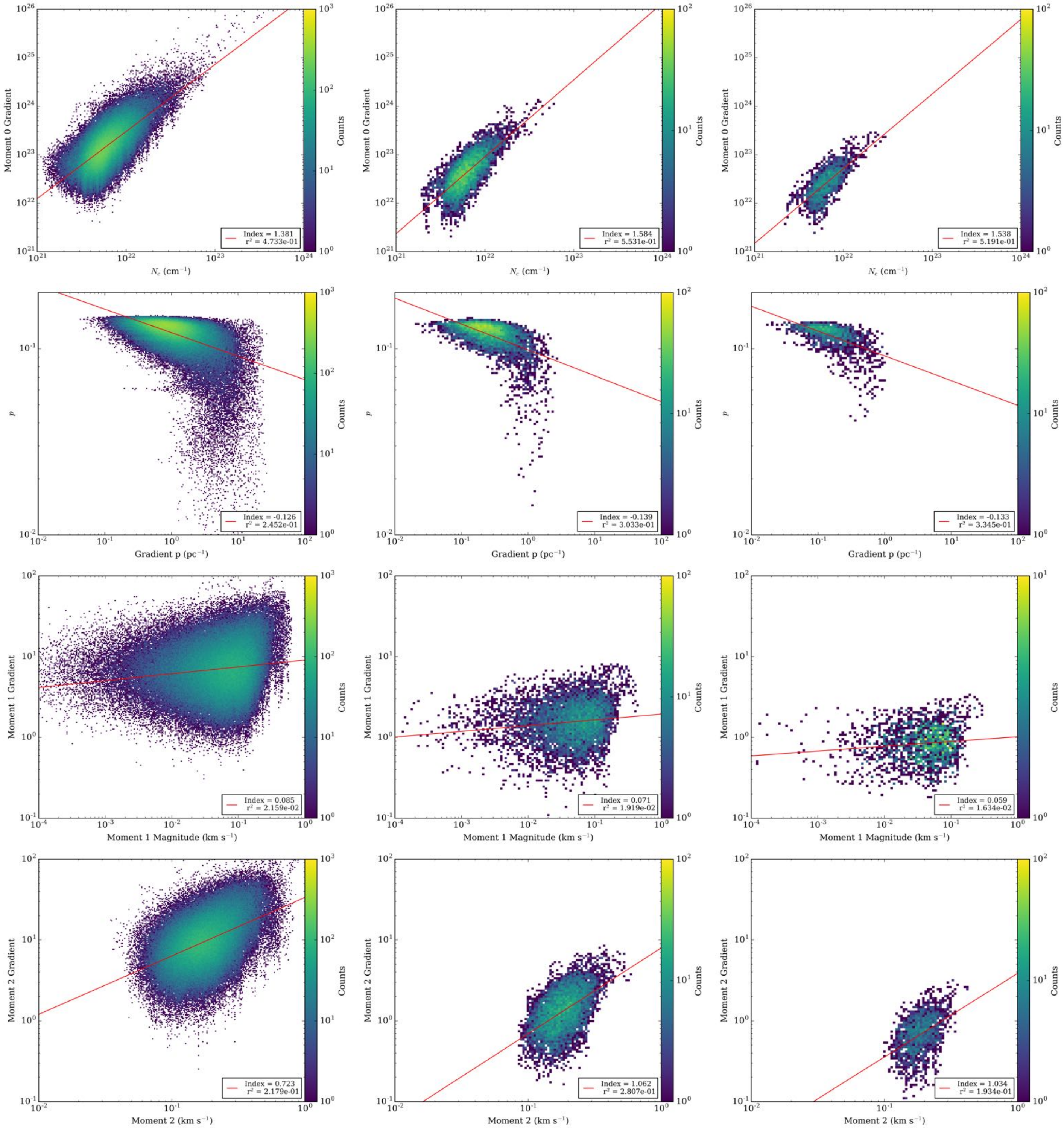

Figure 9: Quantity vs. Gradient in Quantity scatter plots for the 1 pc box simulation. Columns: 1 is simulation resolution (0.002 pc); 2 is 0.0125 pc resolution; and 3 is 0.025 pc resolution. Rows: 1 is N_c vs. ∇M_0; 2, p vs. ∇p; 3, $|M_1|$ vs. ∇M_1; 4, M_2 vs ∇M_2.

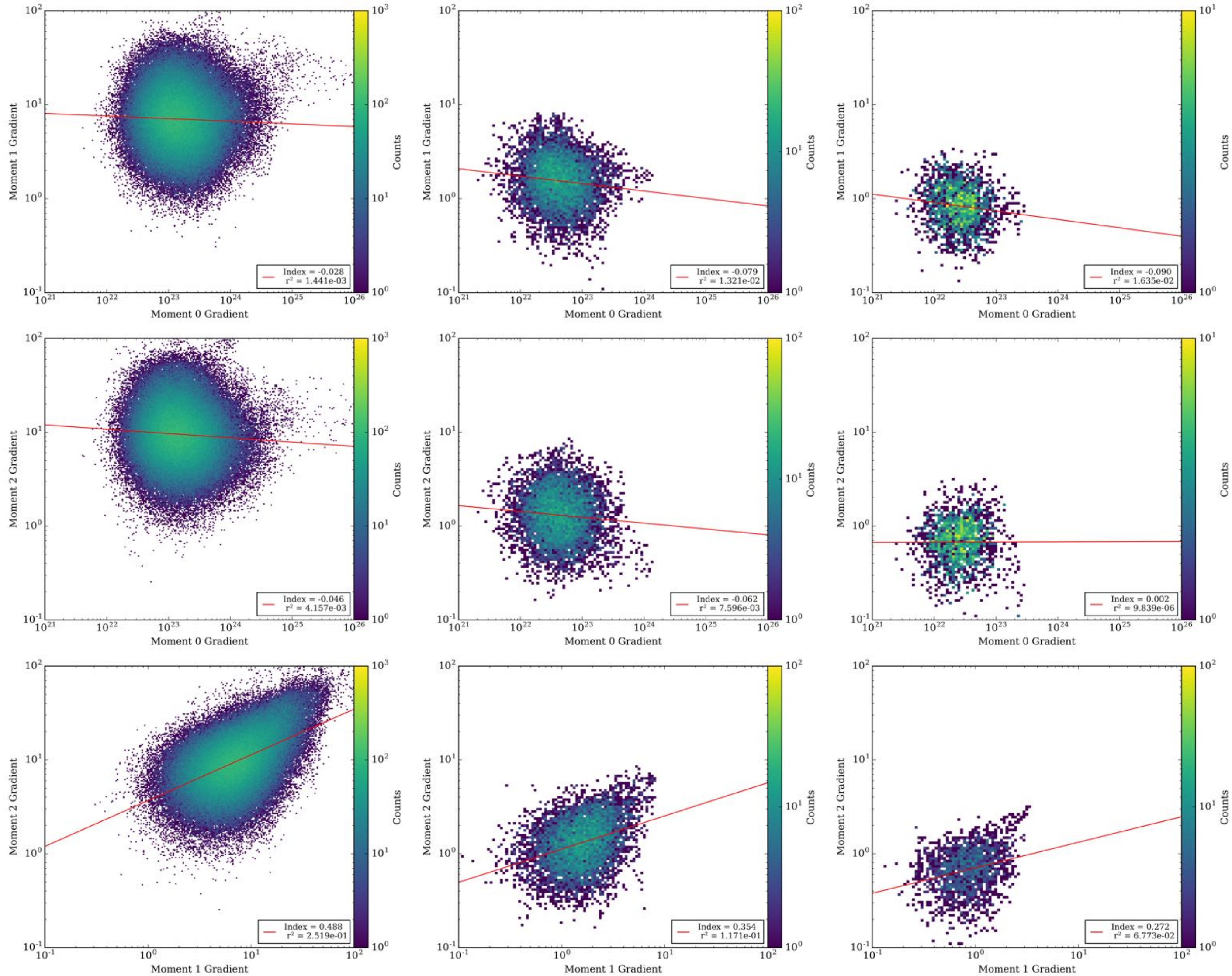

Figure 10: Moment Gradient scatter plots for the 1 pc box simulation. Columns: 1 is simulation resolution (0.002 pc); 2 is 0.0125 pc resolution; and 3 is 0.025 pc resolution. Rows: 1 is ∇M_0 vs. ∇M_1; 2, ∇M_0 vs. ∇M_2; 3, ∇M_1 vs. ∇M_2.

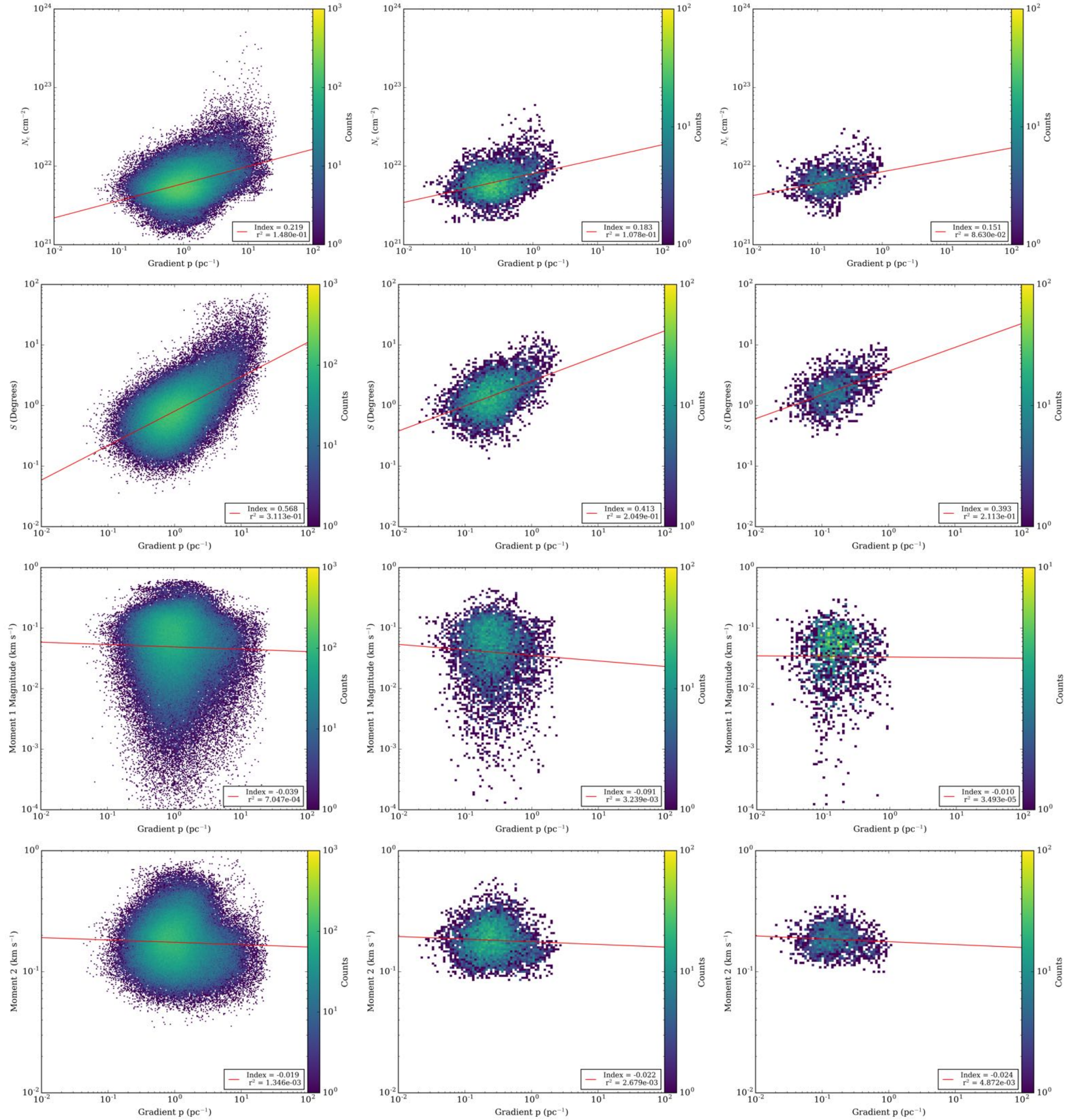

Figure 11: Polarization fraction gradient scatter plots for the 1 pc box simulation. Columns: 1 is simulation resolution (0.002 pc); 2 is 0.0125 pc resolution; and 3 is 0.025 pc resolution. Rows: 1 is N_c vs. ∇p; 2, S vs. ∇p; 3, $|M_1|$ vs. ∇p; 4, M_2 vs ∇p.

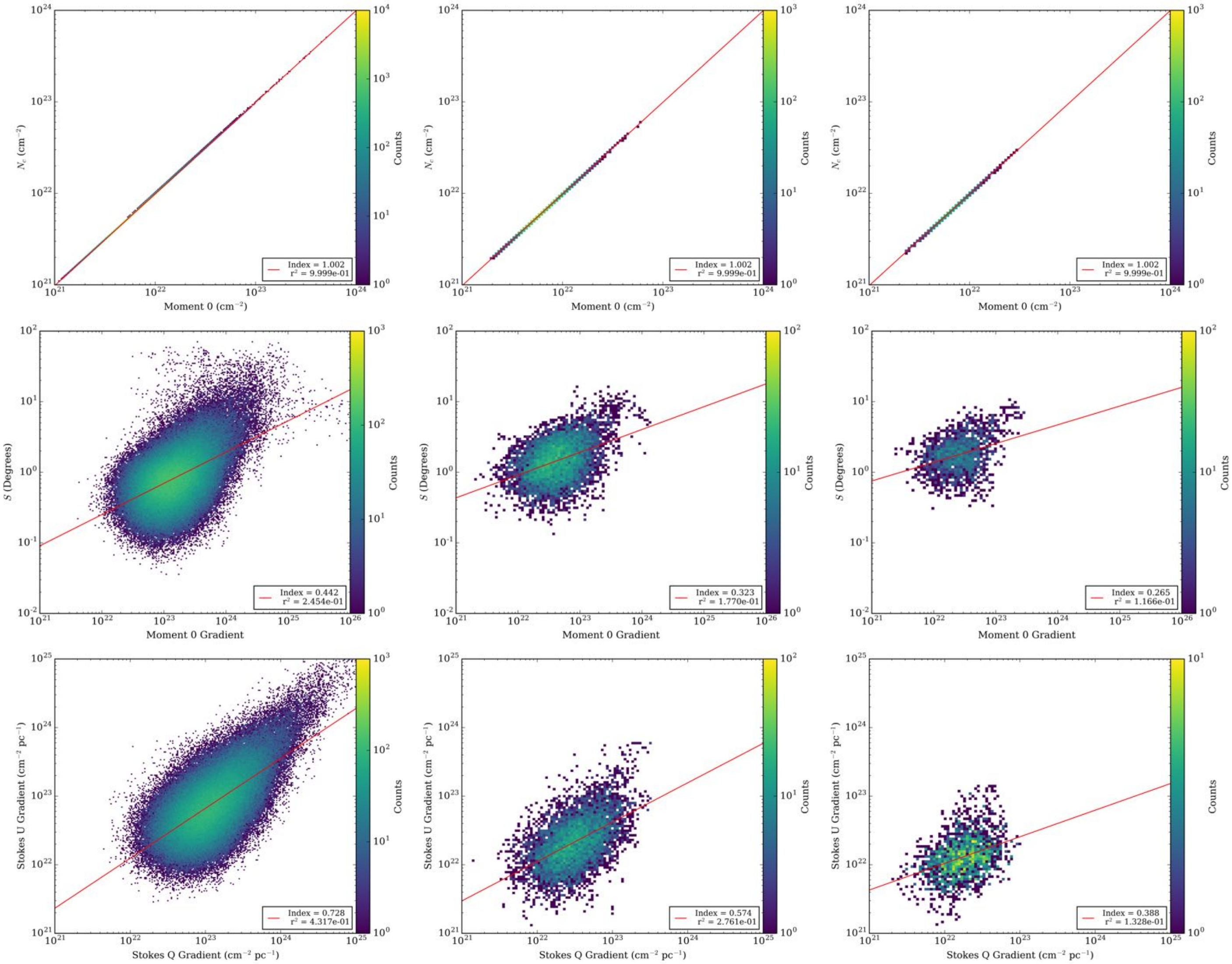

Figure 12: Miscellaneous scatter plots for the 1 pc box simulation. Columns: 1 is simulation resolution (0.002 pc); 2 is 0.0125 pc resolution; and 3 is 0.025 pc resolution. Rows: 1 is N_c vs. M_0; 2, ∇M_0 vs. S; 3, ∇Q vs. ∇U.

3 10 pc Box

3.1 Maps

3.2 Scatter Plots

4 Notes on Pipeline

4.1 Details on Observables

Previously my polarimetry pipeline relied on using the visualization package VisIt (through a python scripting interface) to first produce column integrations and then use a second, independent script to conduct analysis and plotting. I've overhauled this and completely replaced the first script with a much more streamlined interace using yt, which allows the two parts of the pipeline to be combined. This has greatly increased the performance of the pipeline, which can now be run from a single command line execution. (For boring technical reasons, the VisIt-based pipeline has a hard time interpreting the Athena .vtk files, and is resistant to a parallel execution due to some bugs in part of the VisIt source. On the other hand, some analysis, such as computing line-of-sight dispersions, are much faster in VisIt.)

Given the large number of plots that are needed, I added some abstraction layers in the form of callable plotting functions, which (in the future) could form the basis for plotting class objects in a more comprehensive code release. I've built in a lot of flexibility in these functions that should make it easier for it to adapt to different simulations. Any plot format - contour maps, vector annotation, etc. - can be now implemented like this. There is still a lot of abstraction and case-handling to be done but the pipeline might be something we could (eventually) release for use by the community.

Features I want to/ could add, but will absolutely wait until the current paper is finished:

- **Parallelization**: the script is fast - a complete pipeline for a single simulation timeslice takes about 10 minutes on my (admittedly very fast) machine. It could be improved by addressing some of the bottlenecks in calculation that are parallelizable. Additionally, larger beams take longer to run, which might be solved by improving some of the algorithms.

- **Support for Other Codes:** this may be very easy. It would be good to have simple compatibility with a variety of simulation codes out there. I know that yt is very good in this regard, and since yt is the only frontend, I think this could be done trivially, in most cases.

- **Radiative Transfer**: Adding a radiative transfer module would be a nice feature, and I can think of a couple ways to do it. However, it is total overkill for the purposes of our work. It would also be grossly expensive in terms of computation time. Since our simulations are isothermal and optically thin to most tracers, and the sources we are comparing to nearly so, it is unlikely that adding RT would give us anything more than we already have.

- **Asymmetric Beam**: This feature is actually relatively straightforward to implement. Unclear whether it is useful.

- **Rotation of Viewing Angle**: This is a feature which I haven't gotten around to implementing. I'd like to do it in an abstract way (implement a Rotator function or something) but I haven't gotten around to it.