Supplemental Material

Supplemental Material: K_{DP} Algorithm Descriptions

This supplemental material provides more detailed descriptions of the K_{DP} estimation algorithms described in the manuscript. The information provided here comes from associated publications or code documentation (if available), and by our interpretation of the source code for each algorithm. Here, Φ_{DP} represents the propagation differential phase shift, δ is the phase shift imparted on backscatter, and Ψ_{DP} is the total differential phase shift produced by the combination of Φ_{DP}, δ, and any phase shift imparted by the radar itself. K_{DP} represents the specific differential phase. None of the algorithms described below explicitly calculate δ, but if the resulting smoothed Φ_{DP} field is monotonic and appears to correctly smooth the noisy, observed Ψ_{DP} field, users can estimate δ through analyzing the difference between the processed Φ_{DP} field and the observed Ψ_{DP} field.

1 phase_proc_lp (Giangrande et al. 2013)

Here we briefly summarize the "phase_proc_lp" linear programming algorithm described fully in Giangrande et al. (2013). We relate the algorithm description to the tunable parameters available within the Py-ART phase_proc_lp function in an attempt to clarify the purpose of these various tuning parameters. In its basic form, linear programming seeks to optimize a linear function while simultaneously maintaining previously defined constraints on the linear function. In this algorithm, the authors aim to minimize the L1 norm (L1), which can be thought of as minimizing the sum of the magnitudes of vectors in space. For processing Ψ_{DP} the authors define the following minimization problem:

$$L_1 = \sum_{i=1}^{n} = |x_i - b_i| \tag{1}$$

where b represents the Ψ_{DP} data to be processed and x represents the "fitted" data. Thus, the authors attempt to minimize the difference between the processed differential phase data and the Ψ_{DP} data measured by a weather radar. By minimizing this difference, the authors determine the optimal values of x. This minimization problem can be expressed as an LP problem through the following matrix equation:

$$\mathbf{A}\mathbf{x_c} \geq \mathbf{b} \tag{2}$$

or, equivalently

$$\underbrace{\begin{bmatrix} \mathbf{I_n} & -\mathbf{I_n} \\ \mathbf{I_n} & \mathbf{I_n} \\ \mathbf{Z_{n-4,n}} & \mathbf{M_{n-4,n}} \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} \mathbf{z} \\ \mathbf{x} \end{bmatrix}}_{\mathbf{x_c}} \geq \underbrace{\begin{bmatrix} -\mathbf{b} \\ \mathbf{b} \end{bmatrix}}_{\mathbf{b}} \tag{3}$$

The first two rows of $\mathbf{A}$ serve to split up the absolute values in the minimization problem above, such that we now have two equations ($z_i \geq x_i - b_i; z_i \geq -x_i + b_i$) in matrix form. $\mathbf{I_n}$ represents an n by n identity matrix. $\mathbf{M_{n-4,n}}$(shown below) is a $n-4$ by n matrix that implements a five-point Savitzky-Golay second-order derivative filter intended to reduce instrument noise. The coefficients of the filter are: $-0.2, -0.1, 0.0, 0.1, 0.2$. The authors require that the resulting derivative of each application of the Savitzky-Golay filter be non-negative to enforce monotonicity. The matrix is $n-4$ by n because the filter is not applied to the edge of the data array.

$$\mathbf{M}_{n-4,n} = \begin{bmatrix} -0.2 & -0.1 & 0.0 & 0.1 & 0.2 & 0_6 & \dots & 0_n \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0_1 & \dots & 0_{n-5} & -0.2 & -0.1 & 0.0 & 0.1 & 0.2 \end{bmatrix} \tag{4}$$

The $\mathbf{Z_{n-4,n}}$ matrix is an $n-4$ by n matrix of zeroes. Combined, $\mathbf{Ax_c} - \mathbf{b} \geq 0$ contains all the constraints for the applicable LP problem. To solve this LP problem, the authors make use of the concept of duality (summarized in Ch. 6 of Bazaraa et al. 2010). For each primary LP problem (the primal), there exists a "dual" LP problem that is maximized as the primal is minimized. In this case:

Primal: Minimize $\mathbf{c} \cdot \mathbf{x_c}$ subject to $\mathbf{Ax_c} \geq \mathbf{b}, \mathbf{x_c} \geq 0$
Dual: Maximize $\mathbf{w} \cdot \mathbf{b}$ subject to $\mathbf{wA} \leq \mathbf{c}, \mathbf{w} \geq 0$

where $\mathbf{c}$ is a vector of length $2n$ comprising cost coefficients $\mathbf{c} = \{1_1, \dots, 1_n, 0_{n+1}, \dots, 0_{2n}\}$ that act as a dot product to sum the elements of $\mathbf{z}$. The primal and dual can be solved simultaneously, as they should approach the same solution such that $\mathbf{c} \cdot \mathbf{x_c} = \mathbf{w} \cdot \mathbf{b}$. This problem is solved via an open source package that uses the simplex method (Dantzig 1998). The specific package used to solve this problem can be defined by the user, as the algorithm provides three options.

Once the optimal values of $\mathbf{x}$ are found, a Sobel window is applied to the data to calculate the final K_{DP} profile. The length of this window can be defined by users, with the default value set to 35. This value corresponds to the number of gates used in the filter window, and therefore corresponds to the number of coefficients used.

Along with the ability to change the length of the Sobel window, the algorithm also allows the user to decide if they want to take advantage of self-consistency constraints. To implement self-consistency constraints, the primal constraint $\mathbf{b} \geq 0$ is adjusted such that $\mathbf{b}$ is now dependent on Z_H. In this case, $b_i \geq K_{\mathrm{DP}}(Z_i) = aZ^b$, or within the code the relationship is $K_{\mathrm{DP}} = (10^{0.1Z})^{coef}$. The coefficient, "coef", can be defined by the user, but the default is set to 0.914. Users can change this coefficient to change the amount of constraint

Z_H has on K_{DP}, or they can set it such that it matches theoretical expectations for the relationships between Z_H and K_{DP}. The self consistency factor can also be used to determine how much weight the relationship between Z_H and K_{DP} has in the final solution. The default value for this factor is 6×10^4. In analysis not shown, we varied the factor from 0 to 1.8×10^5, increasing by a factor of 2×10^4. Setting this value smaller than about 4×10^4 led to degraded results. Discontinuous fields can be assigned zero weight within the cost functions such that missing data does not affect the K_{DP} estimation. In this case, missing data points do not influence the Φ_{DP} estimation and no processed data is assigned to these points.

Within the algorithm, the "fzl" parameter is intended for users to define the height at which the 0°C level is located. Any Ψ_{DP} values above this level are not processed because the monotonicity constraint breaks down in the mixed phase. In addition, there exists a "high_z" threshold meant to serve as a cap on Z_H that prevents hail from contaminating the K_{DP} estimate. The authors suggest a cap between 50-53 dBZ, with the default value in the algorithm being 53 dBZ. Any value above the defined threshold is set to that threshold within the algorithm to ensure that self consistency between K_{DP} and Z_H is not contaminated by hail. Minimum thresholds also exist for the polarimetric variables ("low_z","min_phidp", "min_ncp", "min_rhv") to serve as another way to initially process the data. Any values below these minimum thresholds are set to the minimum threshold. Users may also define the system differential phase ("sys_phase") to prevent error due to system differential phase offsets. The algorithm returns both the K_{DP} estimate as well as a processed (or smoothed) Ψ_{DP} field.

2 FIR Filter (Hubbert and Bringi 1995)

The appendix in Hubbert and Bringi (1995) describes an iterative technique that uses a finite impulse response (FIR) filter to remove δ and smooth the Φ_{DP} field. Because our study assumes no δ, we simply apply the FIR filter once to our idealized tests. We describe our methodology here.

There are 21 coefficients in the Hubbert and Bringi (1995) FIR filter, meaning that 21 range gates are included within each application of the filter along a radial. The center point of the 21-point filter is the one being processed, and therefore the location in the radial at which a Φ_{DP} value is being calculated. Each of the 21 range gates included in the filter are multiplied by their respective coefficient. The 21 resulting values are then summed together and normalized by the sum of the FIR filter coefficients. Once this is complete, K_{DP} is estimated as half the slope of the processed Φ_{DP} field. The FIR coefficients are listed below in Table 1.

Table 1: FIR filter coefficients as specified in Hubbert and Bringi (1995)

FIR Filter Coefficients	
1) $1.625807356 \times 10^{-2}$	12) $6.718151185 \times 10^{-2}$
2) $2.230852545 \times 10^{-2}$	13) $6.476934523 \times 10^{-2}$
3) $2.896372364 \times 10^{-2}$	14) $6.089991897 \times 10^{-2}$
4) $3.595993808 \times 10^{-2}$	15) $5.578764970 \times 10^{-2}$
5) $4.298744446 \times 10^{-2}$	16) $4.971005447 \times 10^{-2}$
6) $4.971005447 \times 10^{-2}$	17) $4.298744446 \times 10^{-2}$
7) $5.578764970 \times 10^{-2}$	18) $3.595993808 \times 10^{-2}$
8) $6.089991897 \times 10^{-2}$	19) $2.896372364 \times 10^{-2}$
9) $6.476934523 \times 10^{-2}$	20) $2.230852545 \times 10^{-2}$
10) $6.718151185 \times 10^{-2}$	21) $1.625807356 \times 10^{-2}$
11) $6.800100000 \times 10^{-2}$	

3 kdp_schneebeli (Schneebeli et al. 2014)

Schneebeli et al. (2014) use a discrete Kalman filter (KF) to process observed Ψ_{DP} data. A discrete KF is based on the following equations:

$$\mathbf{z}(i) = \mathbf{F}_i \mathbf{s}(i) + \epsilon_{\mathbf{z}(i)} \tag{5}$$

$$\mathbf{s}(i+1) = \mathbf{T}_i \mathbf{s}(i) + \epsilon_{\mathbf{s}(i)} \tag{6}$$

In Eq. 5, the state $\mathbf{s}(i)$ is related to a set of observations $\mathbf{z}(i)$ through a linear model, called the observational model $\mathbf{F}$. In simpler terms, the state variables are converted to "observation space" through $\mathbf{F}$. $\epsilon_{\mathbf{z}(i)}$ encompasses errors due to this observation model as well as those from measurement noise. Eq. 6 represents the forward propagation of the state from location i to $i+1$ via the transition matrix, $\mathbf{T}$. $\epsilon_{\mathbf{s}(i)}$ represents the error due to uncertainty in the forward propagation that takes place through this $\mathbf{T}$ matrix. Between the propagation step established in Eq. 6, KF methods call for the calculation of an *a posteriori* state $\mathbf{s}^{(+)}$ from an *a priori* state $\mathbf{s}^{(-)}$. The *a priori* comes from Eq. 6, where $\mathbf{s}(i)$ is the *a*

posteriori state at $(i-1)$. In other words, Eq. 6 is applied as $\mathbf{s}^{(-)}(i) = \mathbf{T}_i\mathbf{s}^{(+)}(i-1) + \epsilon_{\mathbf{s}(i)}$. The *a posteriori* can then be calculated through:

$$\mathbf{s}^{(+)}(i) = \mathbf{s}^{(-)}(i) + \mathbf{K}_i[\mathbf{z}(i) - \mathbf{F}\mathbf{s}^{(-)}(i)] \tag{7}$$

$\mathbf{K}_i$ represents the Kalman gain, which is defined by the following equation:

$$\mathbf{K}_i = \mathbf{P}^{(-)}_{\mathbf{s}(i)}\mathbf{F}^{\mathrm{T}}[\mathbf{F}\mathbf{P}^{(-)}_{\mathbf{s}(i)}\mathbf{F}^{\mathrm{T}} + \mathbf{C}(\epsilon_{\mathbf{z}(i)})]^{-1} \tag{8}$$

Where $\mathbf{P}^{(-)}_{\mathbf{s}(i)}$ is the error covariance associated with the *a priori* state and $\mathbf{P}^{(+)}_{\mathbf{s}(i)}$ is associated with the *a posteriori* state. These covariance matrices are calculated as follows:

$$\mathbf{P}^{(-)}_{\mathbf{s}(i)} = \mathbf{T}\mathbf{P}^{(+)}_{\mathbf{s}_{i-1}}\mathbf{T}^{\mathrm{T}} + \mathbf{C}(\epsilon_{\mathbf{s}(i)}) \tag{9}$$

$$\mathbf{P}^{(+)}_{\mathbf{s}(i)} = (\mathbf{I} - \mathbf{K}_i\mathbf{F})\mathbf{P}^{(-)}_{\mathbf{s}i} \tag{10}$$

$\mathbf{C}(\epsilon_{\mathbf{s}(i)})$ and $\mathbf{C}(\epsilon_{\mathbf{z}(i)})$ represent the covariance matrices associated with $\epsilon_{\mathbf{s}(i)}$ and $\epsilon_{\mathbf{z}(i)}$ respectively.

When applied to polarimetric radar data, $\mathbf{z}$ includes variables measured by the radar and $\mathbf{s}$ encompasses all quantities we are attempting to retrieve. Variables with a "$\sim$" represent the variable at the next location $(i+1)$ when our current location is (i).

$$\mathbf{z}(i) = \begin{pmatrix} \Psi_{\mathrm{DP}}(i) \\ \tilde{\Psi}_{\mathrm{DP}}(i) \\ c \end{pmatrix} \tag{11}$$

$$\mathbf{s}(i) = \begin{pmatrix} K_{\mathrm{DP}}(i) \\ \delta(i) \\ \Phi_{\mathrm{DP}}(i) \\ \tilde{\Phi}_{\mathrm{DP}}(i) \end{pmatrix} \tag{12}$$

The following relationships are used to relate the values observed in $\mathbf{z}$ to those in $\mathbf{s}$.

$$\Psi_{\mathrm{DP}}(i) = \tilde{\Phi}_{\mathrm{DP}}(i) - 2\Delta r K_{\mathrm{DP}}(i) + \delta(i) \tag{13}$$

$$\tilde{\Psi}_{\mathrm{DP}}(i) = \Phi_{\mathrm{DP}}(i) + 2\Delta r K_{\mathrm{DP}}(i) + \delta(i) \tag{14}$$

$$c = \delta(i) - bK_{\mathrm{DP}}(i) \tag{15}$$

This information can be placed into matrix form through the observational model, $\mathbf{F}$:

$$\mathbf{F} = \begin{pmatrix} -2\Delta r & 1 & 0 & 1 \\ 2\Delta r & 1 & 1 & 0 \\ -b & 1 & 0 & 0 \end{pmatrix} \tag{16}$$

Equations 13 and 14 come from the relationship between Ψ_{DP} and K_{DP}. Equation 15 links K_{DP} with δ. The values of c and b are calculated through fitting linear functions to a scatter plot of K_{DP} and δ. The scatter plot is created through utilizing a large dataset of simulated

DSD fields and applying T-matrix scattering calculations (Mischenko and Travis 1998) to these DSDs. The resulting relationship changes between differing radar wavelengths. The following relationships are used within the $K_{\rm DP}$ algorithm:

$$\text{X band} : \delta^{fit} = \begin{cases} 2.37K_{\rm DP} + 0.054, K_{\rm DP} \leq 2.5^{\circ}\text{km}^{-1} \\ 0.27K_{\rm DP} + 6.16, K_{\rm DP} > 2.5^{\circ}\text{km}^{-1} \end{cases}$$

$$\text{S band} : \delta^{fit} = \begin{cases} 0.19K_{\rm DP} + 0.024, K_{\rm DP} \leq 1.1^{\circ}\text{km}^{-1} \\ 0.019K_{\rm DP} + 0.15, K_{\rm DP} > 1.1^{\circ}\text{km}^{-1} \end{cases}$$

$$\text{C band} : \delta^{fit} = \begin{cases} 0.53K_{\rm DP} + 0.036, K_{\rm DP} \leq 2.5^{\circ}\text{km}^{-1} \\ 0.15K_{\rm DP} + 1.03, K_{\rm DP} > 2.5^{\circ}\text{km}^{-1} \end{cases}$$

To propagate the system forward, the authors make a few important assumptions defined in the following equations:

$$K_{\rm DP}(i+1) = K_{\rm DP}(i) \tag{17}$$

$$\delta(i+1) = \delta(i) \tag{18}$$

$$\Phi_{\rm DP}(i+1) = \tilde{\Phi}_{\rm DP}(i) \tag{19}$$

$$\tilde{\Phi}_{\rm DP}(i+1) = \tilde{\Phi}_{\rm DP}(i) + 2\Delta r K_{\rm DP}(i) \tag{20}$$

This information can be applied in matrix form through the transition matrix, $\mathbf{T}$:

$$\mathbf{T} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 2\Delta r & 0 & 0 & 1 \end{pmatrix} \tag{21}$$

The simulated DSD dataset is also used to calculate the covariance matrices $\mathbf{C}(\epsilon_{\mathbf{s}(i)})$ and $\mathbf{C}(\epsilon_{\mathbf{z}(i)})$. Polarimetric radar T-matrix scattering calculations (described in Mishchenko and Travis 1998, not to be confused with the KF transition matrix above) provide polarimetric output for the simulated DSD fields. The forward transition matrix can be tested and its errors expressed through the following equation:

$$\epsilon_{\mathbf{s}(i)} = \mathbf{s}^{true}(i) - \mathbf{T}\mathbf{s}^{true}(i-1) \tag{22}$$

The combined DSD and T-matrix simulations can also be used to define $\epsilon_{\mathbf{z}(i)}$. In this case, observations are created through adding artificial noise to the calculated $\Psi_{\rm DP}$ profile. The error due to noise and the observation model can be calculated from:

$$\epsilon_{\mathbf{z}(i)} = \mathbf{z}^{noisy}(i) - \mathbf{F}\mathbf{s}^{true}(i) \tag{23}$$

The authors then use a large ensemble of these error vectors to create the covariance matrices $\mathbf{C}(\epsilon_{\mathbf{s}(i)})$ and $\mathbf{C}(\epsilon_{\mathbf{z}(i)})$. The Py-ART algorithm provides users the option to use their own covariance matrices through the "rcov" and "pcov" parameters.

To process Ψ_{DP} data and produce Φ_{DP} and K_{DP} for the entire radar volume, the KF is applied to each range gate along a radial, for each radial in the polarimetric radar profile. The KF is applied in the forward direction and then again in the backward direction to account for errors due to the assumption that $K_{\mathrm{DP}}(i) = K_{\mathrm{DP}}(i+1)$. This application of the KF in the forward and backward direction is applied multiple times, where the $\mathbf{C}(\epsilon_{\mathbf{s}(i)})$ is varied with a set of factors defined by $a = 10^b$ where $b = -1, -0.8, -0.6, \ldots, 0.8, 1.0$. This is done to ensure that the variability of the state from one range gate to the next is not constant throughout the KF application. The result of this methodology is an ensemble of forward and backward KF applications, defined by the scaling factor on $\mathbf{C}(\epsilon_{\mathbf{s}(i)})$.

The authors found that forward application of the KF produces better results for cases where K_{DP} is increasing and the opposite is true for cases where K_{DP} decreases. For this reason, rather than averaging the forward and backward KF results for each ensemble member, the authors set up criteria for choosing one to use. To determine if K_{DP} is increasing or decreasing, the authors compute the ensemble mean and compare the ensemble mean at location i to location $i+1$. According to Schneebeli et al. (2014), if the difference is less than or equal to $-0.1^{\circ}\mathrm{km}^{-1}$, the forward KF result is used for location i. If this difference is greater than or equal to $+0.1^{\circ}\mathrm{km}^{-1}$, then the backward KF results are used. If the difference falls between -0.1 and $+0.1^{\circ}\mathrm{km}^{-1}$, the average of the forward and backward values is used. An analysis of the Py-ART algorithm shows that these thresholds are set to -0.15 and 0.15 in the open-source software.

Finally, to calculate the final K_{DP} value from the ensemble, the average of a specific set of ensemble members is taken. The standard deviation is used to determine how many ensemble members are used in this average, where large standard deviations warrant more ensemble members. The mean is used to determine which ensemble members are included. For example, if the mean value is larger, then ensemble members with larger scaling factors are chosen because they have larger covariance matrices and allow for larger variation between range gates.

The algorithm assumes that the effects of differential phase folding, ground clutter, and low signal-to-noise ratio are already accounted for prior to its application. Regions where Ψ_{DP} is not measured (e.g., regions of no rain) or where it is discontinuous are interpolated linearly and then noise is added to the interpolated field to maintain a consistent field for the KF. The resulting K_{DP} estimates are then removed for regions where Ψ_{DP} was not measured. To account for filtering at the beginning and end of each radial, synthetic data is created to spin up the KF prior to the application of the KF to an observed gate.

4 kdp_maesaka (Maesaka et al. 2012)

This method assumes that Φ_{DP} has already been unfolded and is only applicable in pure rain. For this reason, this methodology should only be applied below the melting level. To calculate K_{DP} from Φ_{DP}, a cost function minimization is used. The cost function is bounded

by the observed $\Psi_{\rm DP}$ field. To determine the boundary values, linear regression is applied to a set number of gates at the beginning and end of the $\Psi_{\rm DP}$ radial, corresponding to the "near" and "far" boundary conditions, respectively. If the fitted line has a positive slope, the linear regression value at the beginning of the radial is assigned to the near boundary condition, and the value at the end of the radial is assigned to the far boundary condition. If the linear regression line is negative, an average of the data points is taken and used for the boundary condition.

The authors define the following cost function:

$$J = J_{\rm obs} + J'_{\rm obs} + J_{\rm lpf} \tag{24}$$

The first two terms of the right hand side of the cost function serve to fit the processed $\Phi_{\rm DP}$ profile to the observed $\Psi_{\rm DP}$ data. Therefore $J_{\rm obs}$ and $J'_{\rm obs}$ are defined as the mean square errors between $\Psi_{\rm DP}$ and the analyzed $\Phi_{\rm DP}$ field:

$$J_{\rm obs} = \frac{1}{N}\sum_{i=1}^{N}(\phi_i - \psi_i)^2 \tag{25}$$

$$J'_{\rm obs} = \frac{1}{N}\sum_{i=0}^{N-1}(\phi'_i - \psi'_i)^2 \tag{26}$$

The prime in $J'_{\rm obs}$ denotes terms that deal with the "far" boundary condition, whereas $J_{\rm obs}$ and its corresponding terms deal with the "near" boundary condition. This is better understood through the following definitions:

$$\phi_i = (\Phi_{\rm DP})_i - \Phi_{\rm near} \tag{27}$$

$$\phi'_i = \Phi_{\rm far} - (\Phi_{\rm DP})_i \tag{28}$$

$$\psi_i = \Psi_i - \delta_i - \Phi_{\rm near} \tag{29}$$

$$\psi'_i = \Phi_{\rm far} - \Psi_i + \delta_i \tag{30}$$

The kdp_maesaka algorithm does not compute δ, and thus allows users to define δ. If no δ is defined by users, then it is set to zero for all range gates. Because $K_{\rm DP}$ is the range derivative of $\Phi_{\rm DP}$, the authors further define the following relationships:

$$\phi'_N = 0 \tag{31}$$

$$\phi_0 = 0 \tag{32}$$

$$\begin{aligned}\phi'_i &= 2\sum_{j=i+1}^{N}(K_{\rm DP})_j \Delta r \\ &= \sum_{j=i+1}^{N} k_j^2, (i = 0, 1, 2, \ldots, N-1)\end{aligned} \tag{33}$$

$$\begin{aligned}\phi_i &= 2\sum_{j=0}^{i-1}(K_{\mathrm{DP}})_j \Delta r \\ &= \sum_{j=0}^{i-1} k_j^2, (i = 1, 2, 3, \ldots, N)\end{aligned} \tag{34}$$

$$k_i^2 = 2(K_{\mathrm{DP}})_i \Delta r \tag{35}$$

The third term in the cost function (J_{lpf}) functions as a low pass filter. It is defined as the mean square of the Laplacian of k:

$$J_{\mathrm{lpf}} = \frac{1}{N+1} C_{\mathrm{lpf}} \sum_{i=0}^{N} (\frac{\partial^2 k_i}{\partial r^2})^2 \tag{36}$$

The parameter C_{lpf} determines how much of an effect the low pass filter has on the final solution. When J_{lpf} is larger than $J_{\mathrm{obs}} + J'_{\mathrm{obs}}$, the result is a smoother profile. With this idea in mind, it can be understood that larger C_{lpf} values support smoother profiles. The Py-ART algorithm provides users the opportunity to change the C_{lpf} parameter values such that they control the amount of smoothing applied by the algorithm.

The overall goal of this methodology is to minimize the cost function with respect to k, where K_{DP} is calculated from the final k profile. The algorithm offers users various tunable parameters related to minimizing this cost function. First, the "method" used to solve the cost function refers to the Python package used to solve the cost function. Maesaka et al. (2012) describes the use of the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm, but the default choice within the algorithm itself does not align with that described in Maesaka et al. (2012). Instead, an algorithm available through another Python package that is faster than the BFGS algorithm is used. The "length_scale" ensures the magnitude of the C_{lpf} value aligns with the observed data. For example, when length_scale is not defined by the user, it is set to the radar range resolution. Users can define a "first_guess" for the cost function, which should be close to 0 since K_{DP} should be assumed to be 0 everywhere initially. To avoid problems with the cost function, though, it should be set to something slightly larger than 0; the default is 0.01. The "finite_order" parameter allows users to determine finite difference accuracy when the algorithm computes derivatives. The "maxiter" parameter refers to the maximum number of iterations performed when minimizing the cost function. This maximum number is only met if the cost function does not converge on a solution in sufficient time.

5 calc_kdp_bringi (Lang et al. 2007)

Available at `https://github.com/CSU-Radarmet/CSU_RadarTools`,the CSU K_{DP} algorithm offers a relatively simple solution to the differential phase processing problem. The algorithm utilizes a finite-impulse filter (FIR) like that in Hubbert and Bringi (1995) and behaves similarly, with the main differences being that the filter length is flexible and Z_H is used to assist

in the $K_{\rm DP}$ calculations (Lang et al. 2007).

The algorithm begins with an initial quality-control process in which the standard deviation of $\Phi_{\rm DP}$ is calculated along each radial. The $\Phi_{\rm DP}$ threshold ("thsd") defined in the function is compared to the calculated standard deviation. If the calculated standard deviation is larger than the threshold, those $\Phi_{\rm DP}$ values are masked out. Once the entire field is quality controlled, a FIR filter is established. The size of the filter and its corresponding coefficients are dependent on the "window" and "gs" parameters defined in the algorithm function. The "window" parameter represents the length over which the FIR filter is applied (in km) and "gs" is the gate spacing (in m). The FIR filter length is determined by dividing the window length by the gate spacing. For example, in the case where the window=3 and gs=150, the FIR filter would be of the $20^{\rm th}$ order, with 21 total coefficients applied to 21 consecutive range gates. The coefficients of the FIR filter change based on the window length/gate spacing. The user can define how many times the FIR filter is applied to the data through the "nfilter" parameter.

Once the $\Phi_{\rm DP}$ field has been processed, $K_{\rm DP}$ is calculated by computing one-half the slope of the processed $\Phi_{\rm DP}$. The window over which the slope is calculated varies based on the Z_H value at the gate of interest. If $Z_H \geq 45$ dBZ, the window over which $K_{\rm DP}$ is computed is taken as half of the FIR window size, whereas if $35 \leq Z_H < 45$ dBZ , the window size is twice the halved FIR window length. If $Z_H < 35$dBZ, the window size is 3 times the halved FIR filter length. Once $K_{\rm DP}$ is calculated at each gate, the algorithm returns the $K_{\rm DP}$ field, along with the filtered $\Phi_{\rm DP}$ field and the standard deviation of $\Phi_{\rm DP}$ calculated during the quality-control process.

6 NWS Operational Algorithm (Ryzhkov et al. 2005)

The operational algorithm used by the U.S. National Weather Service (NWS) to calculate $K_{\rm DP}$ is partially described in Ryzhkov et al. (2005), and is simple compared to many of the other algorithms described herein. First, to address the noise within the $\Phi_{\rm DP}$ data, a moving average is used to smooth the $\Phi_{\rm DP}$ field. First a 9-gate moving window is applied such that the average of the 9-gate window is applied to the center value. This is done moving from the beginning to the end of a radial, producing a smoother $\Phi_{\rm DP}$ field. This process is then repeated using a 25-gate window. $K_{\rm DP}$ is calculated for both the 9-gate and 25-gate moving average $\Phi_{\rm DP}$ fields. To calculate $K_{\rm DP}$, a least-squares fit line is found using 9 gates for the 9-gate moving average and 25 gates for the 25-gate moving-average $\Phi_{\rm DP}$ field. The slopes of the least-squares lines are halved to get a $K_{\rm DP}$ value for the center gate of the window. To calculate the final $K_{\rm DP}$ value for each gate, Z_H is consulted. Moving through each gate of the radial, if $Z_H < 40$ dBZ, the $K_{\rm DP}$ value at the same gate calculated from the 25-gate moving window is used. If $Z_H \geq 40$ dBZ, the $K_{\rm DP}$ value from the 9-gate window is used.

We are unaware of literature that discusses how the edges of the radial are treated. For

the sake of our analysis, we simply set K_{DP} to 0 for the first 4 and 12 gates of the 9-gate and 25-gate window K_{DP} calculations, respectively. We then set the last 4 and 12 gates of the 9-gate and 25-gate moving window to the last computed K_{DP} value in the radial.

7 kdp_vulpiani (Vulpiani et al. 2012)

The kdp_vulpiani algorithm found in Py-ART is based on the retrieval described in Vulpiani et al. (2012) and applied in Vulpiani et al. (2015). The algorithm follows a four step process. First, an initial guess is made through computing the finite difference across the raw, Ψ_{DP} field through the following equation:

$$K'_{\mathrm{DP}}(r_k) \approx 0.5[\Psi_{\mathrm{DP}}(r_k + \frac{L}{2}) - \Psi_{\mathrm{DP}}(r_k - \frac{L}{2})]/L \tag{37}$$

where L is the window length defined by the user ("windsize" parameter) in gates. The initial guess is then analyzed and compared to K_{DP} thresholds defined within the algorithm. The algorithm is designed to work with S-, C-, and X-band radars. The radar wavelength must be defined within the "band" parameter. The thresholds are meant to limit K_{DP} values outside the expected, physically possible, K_{DP}. The thresholds are as follows:

X band: -2 to 40°km^{-1}
C band: -2 to 20°km^{-1}
S band: -2 to 14°km^{-1}

We note that K_{DP} values larger than these are possible and have been observed in storms with large quantities of small, melting hail (Kumjian et al. 2019). If any of the K_{DP} values acquired by the first guess are smaller or equal to the minimum threshold of -2°km^{-1}, the K_{DP} value there is set to 0. If the K_{DP} value is larger than the defined maximum threshold, the K_{DP} value is also set to 0. This is done to avoid non-physical K_{DP} fluctuation owing to noise, δ, nonuniform beam filling, and residual artifacts. Vulpiani et al. (2012) also describe a method in which Ψ_{DP} values that are aliased are unfolded at this point in the algorithm, but the Py-ART implementation of the algorithm assumes that Ψ_{DP} has already been processed, and therefore does not require any sort of unfolding. We will focus on the algorithm present within Py-ART. The Py-ART algorithm adds an extra step, not described within the Vulpiani et al. (2012) paper: after initially filtering the first-guess K_{DP}, a rolling window standard deviation is calculated over the field. If the standard deviation at a specific gate is > 5°km^{-1}, then the K_{DP} value is also set to 0.

Next, Φ_{DP} is reconstructed from the processed K_{DP} field through integrating over the K_{DP} first guess as follows:

$$\hat{\Phi}_{\mathrm{DP}} = 2 \int K'_{\mathrm{DP}}(s) ds \tag{38}$$

If the user only desires one iteration of the algorithm, then a final K_{DP} estimate is taken from this reconstructed Φ_{DP} field through using finite differencing once again. Users may

define the number of iterations used within the algorithm through the "n_iter" parameter. If the number of iterations is > 1, then the reconstructed $\Phi_{\rm DP}$ field is used for the next guess of $K_{\rm DP}$ calculated using finite differencing. $\Phi_{\rm DP}$ is then reconstructed from this new $K_{\rm DP}$ guess. This is repeated for n_iter, and then a final $K_{\rm DP}$ estimate is made.

References

Bazaraa, M. S., J. J. Jarvis, and H. D. Sherali, 2010: *Linear programming and network flows*. 4th ed., John Wiley & Sons, 768 pp.

Dantzig, G. B., 1998: *Linear programming and extensions*, Vol. 48. Princeton university press, 94-119 pp.

Giangrande, S. E., R. McGraw, and L. Lei, 2013: An application of linear programming to polarimetric radar differential phase processing. *Journal of Atmospheric and Oceanic Technology*, **30 (8)**, 1716–1729, doi:10.1175/JTECH-D-12-00147.1.

Hubbert, J., and V. N. Bringi, 1995: An iterative filtering technique for the analysis of copolar differential phase and dual-frequency radar measurements. *Journal of Atmospheric and Oceanic Technology*, **12 (3)**, 643–648, doi:10.1175/1520-0426(1995)012⟨0643:AIFTFT⟩2.0.CO;2.

Kumjian, M. R., Z. J. Lebo, and A. M. Ward, 2019: Storms producing large accumulations of small hail. *Journal of Applied Meteorology and Climatology*, **58 (2)**, 341–364, doi:10.1175/JAMC-D-18-0073.1.

Lang, T. J., D. A. Ahijevych, S. W. Nesbitt, R. E. Carbone, S. A. Rutledge, and R. Cifelli, 2007: Radar-observed characteristics of precipitating systems during NAME 2004. *Journal of Climate*, **20 (9)**, 1713–1733, doi:10.1175/JCLI4082.1.

Maesaka, T., K. Iwanami, and M. Maki, 2012: Non-negative $\rm K_{DP}$ estimation by monotone increasing $\Phi_{\rm dp}$ assumption below melting layer. *Extended Abstracts, Seventh European Conf. on Radar in Meteorology and Hydrology*, Toulouse, France, ERAD, [Available online at http://www.meteo.fr/cic/meetings/2012/ERAD/extended_abs/QPE_233_ext_abs.pdf.].

Mishchenko, M. I., and L. D. Travis, 1998: Capabilities and limitations of a current fortran implementation of the t-matrix method for randomly oriented, rotationally symmetric scatterers. *Journal of Quantitative Spectroscopy and Radiative Transfer*, **60 (3)**, 309–324.

Ryzhkov, A. V., T. J. Schuur, D. W. Burgess, P. L. Heinselman, S. E. Giangrande, and D. S. Zrnić, 2005: The Joint Polarization Experiment: Polarimetric rainfall measurements and hydrometeor classification. *Bulletin of the American Meteorological Society*, **86 (6)**, 809–824, doi:10.1175/BAMS-86-6-809.

Schneebeli, M., J. Grazioli, and A. Berne, 2014: Improved estimation of the specific differential phase shift using a compilation of Kalman filter ensembles. *IEEE transactions on geoscience and remote sensing*, **52 (8)**, 5137–5149.

Vulpiani, G., L. Baldini, and N. Roberto, 2015: Characterization of mediterranean hail-bearing storms using an operational polarimetric x-band radar. *Atmospheric Measurement Techniques*, **8 (11)**, 4681–4698, doi:10.5194/amt-8-4681-2015.

Vulpiani, G., M. Montopoli, L. D. Passeri, A. G. Gioia, P. Giordano, and F. S. Marzano, 2012: On the use of dual-polarized C-band radar for operational rainfall retrieval in mountainous areas. *Journal of Applied Meteorology and Climatology*, **51 (2)**, 405–425, doi:10.1175/JAMC-D-10-05024.1.