ChatGPT

Rainfall Surface Interpolation and Basin Averaging Methods

Overview and Goal

You can improve your model by generating a continuous rainfall surface (a spatial map of 10-minute rainfall) from point gauge data and then computing each basin's average rainfall per time step. Instead of using just the 3 nearest gauges (which can introduce high variability due to differing distances/directions per basin), this approach uses **spatial interpolation** across all 77 rain gauges to estimate rainfall everywhere in the region. From this interpolated rain field, you then calculate the mean rainfall over each of the 35 basin polygons (using their shapefiles) for each 10-minute interval. This yields one representative rainfall value per basin per timestep. The key tasks are: (1) choosing a suitable interpolation method, (2) applying it in Python, and (3) ensuring robust results even if some gauges are missing or faulty.

Common Spatial Interpolation Methods

Several well-established methods can create a rain surface from point measurements. Each has pros and cons in terms of accuracy and robustness. Common techniques include the following [1] [2]:

- **Thiessen Polygons (Voronoi method):** This classical approach divides the region into polygons around each gauge such that every point in a polygon is closest to that gauge. The rainfall is assumed uniform within each polygon, equal to the gauge value. It's simple and has historically been used to compute Mean Areal Precipitation by assigning each gauge a weight proportional to its polygon area. However, it produces a *piecewise constant* rainfall field with abrupt boundaries – spatial gradients or storm patterns within a polygon are ignored. Thiessen interpolation can misrepresent localized storms, especially in **small basins** or sparse networks. Studies show that for smaller catchments (e.g. <500 km^2), Thiessen estimates can diverge significantly from more continuous methods [3]. In sparse gauge networks, Thiessen tends to **overestimate** peak area-average rainfall (because a high reading at one gauge is applied to that gauge's entire polygon) [4]. It is also very sensitive to gauge issues: if a gauge malfunctions or reads an extreme value, that entire polygon's rain estimate is affected. Adding or removing a gauge requires redrawing polygons, which can change basin averages discontinuously. In short, Thiessen is easy to implement but may oversimplify spatial variability and is not robust to gauge errors.

- **Inverse Distance Weighting (IDW):** IDW is a deterministic interpolation that estimates rainfall at an unknown location as a weighted average of nearby gauges, with weights decreasing with distance (often as $1/distance^2$ by default) [5] [6]. Essentially, closer gauges influence the estimate more, reflecting the assumption that rain at nearby points is more similar (an expression of spatial autocorrelation). IDW produces a smooth, continuous rain surface. It's straightforward to implement (e.g. using Python's GIS libraries or SciPy), and you can customize the power parameter or search radius (e.g. include all 77 gauges or only the closest N gauges). Compared to Thiessen, IDW yields gradations of rainfall between stations instead of sharp boundaries. This usually gives more realistic

spatial patterns and reduces reliance on any single gauge. **Advantages:** simplicity, no need for complex modeling of spatial statistics, and it naturally handles missing gauges by using whatever data is available. **Limitations:** IDW performance is best when gauges are **densely and evenly spaced** [7]. If the network is irregular or sparse in some areas, IDW can introduce biases – for example, it may **underestimate** true peak rainfall in sparse networks because it smooths extreme values with surrounding lower values [4]. The choice of power exponent affects this: a higher power (e.g. 2 or 3) makes the interpolation mimic nearest-gauge influence (less smoothing), while a lower power smooths more. IDW does not inherently account for physical factors like terrain, and it can't estimate beyond the range of observed values (the interpolated surface stays within the min–max of gauge values) [8]. In practice, IDW is a flexible baseline; however, researchers often find that more advanced techniques (like kriging) can provide a **more robust model** for rainfall interpolation [6] when data are variable or unevenly spaced.

- **Spline or Radial Basis Function Interpolation:** These methods (e.g. thin-plate splines, multiquadric RBF) fit a smooth surface through the data points. They are less commonly used than IDW or kriging for rainfall, but they can produce realistic continuous surfaces. For example, **Hardy's multiquadric interpolation** (a type of RBF) was one method evaluated alongside IDW and kriging [1]. Splines tend to minimize curvature, producing a gently varying rain field that passes through all gauge values. The advantage is a smooth interpolation that can extrapolate trends beyond the convex hull of points. However, like IDW, spline methods are deterministic and don't provide an explicit error estimate. They can also overshoot (predict values outside the data range) and may require parameter tuning (e.g. tension or smoothing factor). In terms of robustness, a spline will incorporate all gauges, so no single station wholly controls a region (unlike Thiessen). Still, a grossly bad gauge reading will force the surface to bend through that value, potentially causing ripples or bullseye artifacts around that gauge. Spline methods usually don't inherently consider terrain or other variables unless used in a modified form.

- **Ordinary Kriging (Geostatistical Interpolation):** Kriging is a statistically-based interpolation that models the spatial correlation structure of rainfall. It treats rainfall as a random field and uses a semivariogram to weight gauge contributions based on distance (and optionally direction). Ordinary kriging provides the **Best Linear Unbiased Estimator** under certain assumptions, and it can also give an estimation error/uncertainty for each location. The benefit of kriging is that it can capture spatial patterns more optimally – for example, if you determine that gauges within 10 km have high correlation but beyond 30 km have little influence, kriging will weight accordingly rather than using an arbitrary distance exponent. Kriging often produces a smoother result than IDW for the same data, due to its tendency to **smooth out extremes** in the absence of nearby supporting data [9]. This can be a limitation: **univariate kriging tends to smooth the rainfall field**, which might underestimate peak intensities of very localized storms (similar to IDW's smoothing, unless a gauge is exactly at the storm center) [9]. It requires more effort: you need to fit a variogram model to your gauge data (which can be tricky for highly transient 10-min rainfall patterns), and kriging is computationally heavier (though with 77 gauges it's still feasible in Python, e.g. using libraries like `pykrige` or `skgstat`). On the positive side, kriging can naturally handle missing data by using the covariance structure – if a gauge is missing, it simply isn't included in the system of equations for that time step, and other gauges fill in the estimate. **Extensions:** You mentioned having a DEM (terrain elevation) but being unsure about using it. Kriging can incorporate elevation as a secondary variable through techniques like *Kriging with External Drift (KED)* or co-kriging. This is useful if rainfall has a known spatial trend (e.g. increasing with altitude). In mountainous orographic climates,

including elevation can improve estimates for monthly or annual precipitation [10]. However, for short-duration rainfall (10-minute storms), the correlation with elevation is often weak [10] – in other words, storms at 10-minute scale are more influenced by local convective cells or frontal systems than by gradual elevation trends. Studies have found that adding elevation helped interpolation accuracy for long-term accumulations, but gave little improvement for daily or hourly rainfall in some regions [10]. Given Israel's Mediterranean climate and the short time step, you might not gain much from using DEM in the interpolation unless you know of a strong elevation effect on intensity. Overall, ordinary kriging (or its variants) is powerful but requires statistical expertise and careful quality control. If set up properly, it can yield a robust interpolation and even quantify the uncertainty in each basin's rainfall estimate.

Summary of Method Trade-offs: Simpler methods (Thiessen, IDW) are easier to implement and require few assumptions, but they can either oversimplify or overly smooth the spatial variability. Thiessen gives each gauge a strict zone of influence (good for quick basin averages but *very* sensitive to that gauge's correctness), while IDW and splines involve multiple gauges per location, producing smoother transitions. Advanced geostatistical methods like kriging consider spatial correlation explicitly and often outperform simpler methods in accuracy tests, especially as network or event complexity increases [6]. Notably, as your gauge network density increases, the differences between interpolation methods diminish [11] – with 77 gauges over your region, even a basic method might perform reasonably, but a more refined method can still handle edge cases better. Many hydrologists use Thiessen or IDW for simplicity, but turn to kriging when they need maximum reliability or have irregular gauge spacing [6]. In fact, one study found that in sparse gauge networks, Thiessen tended to overestimate and IDW to underestimate area-average rainfall, while kriging (and similar methods) provided a balanced estimate [4]. It's wise to test a couple of methods on your data (e.g. compare basin averages from Thiessen vs IDW) to see how much difference it makes given your network and climate.

Computing Basin Average Rainfall

Once you have an interpolated rain field (a continuous surface), the basin average at each timestep can be obtained by **zonal averaging** – essentially integrating the rainfall over the basin area and dividing by the area. There are a few ways to do this:

- **Raster Approach:** Create a raster grid covering your region, with a reasonable resolution (finer than the smallest basin's scale). Use the chosen interpolation method to compute the rainfall value at each grid cell for a given time step. In Python, you could use libraries like **SciPy** (`scipy.interpolate.griddata` for IDW-like methods or linear interpolation), **PyKrige** (for kriging), or GIS libraries (e.g. `GDAL` or `Rasterio`) to produce this gridded rainfall. Once you have a raster of rainfall, you can use a library such as **rasterstats** (`zonal_stats`) or **geopandas/shapely** to calculate the average value of all grid cells that fall inside each basin polygon. Essentially, you overlay the basin shapefile on the rainfall raster and compute mean values per polygon. This process is repeated for each timestep (you can loop over time or use vectorized operations if data fits in memory). The result will be a time series of average rainfall for each of the 35 basins. This raster method is straightforward and takes advantage of existing GIS tools. The resolution should be chosen fine enough that basin boundaries are well-resolved (to avoid big partial-cell errors), but note that very high resolution will increase computation time for each timestep. Since you have 10-minute data (potentially many timesteps), you might opt for a moderate resolution and test the sensitivity. The raster approach is essentially a numerical integration of the rain surface over each basin.

- **Analytical/Vector Approach:** You can avoid generating a full raster each time by leveraging the interpolation function directly. For example, with **Thiessen polygons**, you don't need a raster at all: you can pre-compute the intersection of each basin with each gauge's Thiessen polygon to get the fraction of basin area controlled by each gauge. The basin average at time *t* is then simply a weighted sum of the gauges' rainfall at time *t*, with weights equal to those area fractions (which sum to 1 for each basin). This is how Thiessen weighted averaging is traditionally done in hydrology. For IDW or other distance-based methods, an analytical integration is more complicated, but you can still do a vectorized calculation: sample the interpolation at a set of points within the basin (or use integration if a simple form). However, it may be easier to sample many points or use a grid as an approximation. Another approach is to compute **barycentric weights** for each basin relative to the gauges. For instance, if you use a **triangulated irregular network (TIN)** interpolation (linear interpolation on a Delaunay triangulation of gauge points, which is similar to a planar spline), you could intersect the TIN with the basin polygon to integrate exactly. These approaches can be complex to implement from scratch. In practice, many people stick to a grid and zonal stats for ease.

- **Hydrological software/QGIS:** Since you mentioned QGIS, note that GIS software can perform these tasks as well (e.g. QGIS or ArcGIS can do IDW interpolation and zonal statistics). But since you prefer Python, the above programmatic approaches apply. QGIS could be used in a preliminary step (to create Thiessen polygons or verify interpolation results visually), but it's not strictly necessary.

Regardless of the approach, the end product is that for each basin and each timestep, you compute an area-weighted mean of rainfall values. Make sure the units make sense (if gauges are in mm per 10 min, the average will also be mm per 10 min for the basin). Also, consider the **projection and coordinate system** when doing distance-based interpolation or area weighting – working in a projected coordinate system (like UTM or Israeli Grid) is important for distance calculations and area computations (to avoid distortion from degrees). This ensures that, for example, IDW uses true distances in kilometers and basin areas are correctly measured.

Limitations and Considerations

When creating rain surfaces from gauges, be mindful of the method limitations and the nature of your data:

- **Spatial Coverage and Storm Scale:** With 77 gauges over your region, the interpolation can capture large-scale rainfall patterns, but small-scale convective cells might still be missed. Any purely gauge-based surface will not "see" rainfall where there are no gauges. For example, if a heavy downpour happens between stations and doesn't hit a gauge, all interpolation methods will underestimate that area's rainfall (essentially showing something closer to the surrounding lighter rains). This is a fundamental limitation without radar data. The **spatial and temporal resolution** (10-minute) is high, meaning storms can be very localized. In a Mediterranean climate, convective storms or localized intense bursts are possible – these will challenge the interpolation. Methods that spread influence (IDW, kriging) will smooth out such peaks unless gauges are very dense. Thiessen won't smooth (it will give full value to a polygon), but it assumes the storm covers the whole polygon, which can **overestimate area** if the storm cell was actually smaller. Be aware that for **very large basins**, even a perfect interpolation might not reflect variations across the basin if the storm only affects one part. In such cases, the basin average from gauges is an approximation of true distributed rainfall.

- **Gauge Density and Basin Size:** The reliability of basin average estimates improves with gauge density. If some basins have multiple gauges inside or very nearby, their average will be more accurate; basins far from any gauge rely on extrapolation. Research has found that as the number of gauges in a catchment increases, differences between interpolation methods shrink and the mean areal precipitation estimates improve [11]. Conversely, in basins with few or no gauges, methods can disagree more and error is higher [3]. Small basins (with area much smaller than the typical gauge spacing) are particularly prone to estimation error – one gauge might represent the whole basin's rainfall, so if the storm missed that gauge, the basin's estimated rainfall could be very wrong. In the cited study, Thiessen performed notably worse for basins <500 km^2 compared to smoother interpolations [3]. In your case, keep an eye on basins that are under-gauged. You might consider attaching an uncertainty or confidence to each basin's rainfall (though that's advanced, kriging could provide a variance map).

- **Method-Specific Issues:** Each interpolation has quirks. Thiessen polygons don't adapt over time (the polygons are fixed based on gauge layout), so they ignore any dynamic patterns. IDW's results depend on the chosen power and radius; too high power makes it similar to Thiessen (very localized influence), too low power overly smooths. Kriging requires that you have a reasonable variogram for 10-min rainfall – you might need to compute experimental variograms from your data (possibly pooling many time steps or looking at a particular event) to fit a model. The variogram might differ by event type (convective vs widespread rain). If variogram fitting is difficult, you could use a default like an exponential model with range roughly the average spacing of gauges that you consider correlated. Note that **ordinary kriging assumes the mean is constant across the field** (within each time slice) – if there's a broad gradient (say more rain in north than south due to a storm front at that moment), you might violate that assumption. In such cases, universal kriging or adding a trend could be considered, but that's an advanced topic. For most practical purposes, ordinary kriging with a sensible variogram should work. Also, **check for negative kriging weights** if some gauges are very close together – sometimes kriging can assign slight negative weights to some data points to satisfy unbiasedness, which is normal but underscores the need for a good variogram.

- **Temporal Consistency:** Since you'll produce a separate interpolation each 10-minute step, there is a chance of unrealistic temporal fluctuations in the surface if the method isn't constrained in time. For example, if at 10:00 a certain area is estimated dry and at 10:10 a gauge 20 km away measures a spike, the interpolation might suddenly splash rainfall into that area. Real rainfall fields move continuously. Pure spatial interpolation on each time step won't inherently ensure that the rain maps vary smoothly in time. This might or might not be a concern for your modeling (if you are feeding these basin averages into a hydrological model, some noise might be acceptable). If it is a concern, one approach is to apply a slight temporal smoothing or to incorporate motion extrapolation (which is complex without radar). In most cases, using the gauge data as-is at each timestep is fine, just be aware of possible temporal artifacts if a gauge's influence jumps around.

- **Edge Effects:** Basins at the edge of your gauge network (especially beyond the convex hull of gauge locations) will have more uncertainty. Interpolation beyond the convex hull is extrapolation – e.g., a basin north of the northernmost gauge will just get the nearest gauge's value (Thiessen) or an extrapolated trend (IDW/Kriging) which can be unreliable. If any of your 35 basins lie outside or at the fringe of the gauge coverage, treat those averages with caution. You might even exclude far-out basins or note that their rainfall is less certain.

- **Computational cost:** Interpolating 77 points over a region for every 10-minute interval ~could be computationally heavy if you have a long period of record. However, 77 points is not huge; methods like IDW or even kriging (with 77 points) can run fairly quickly in Python, especially if optimized. You might use vectorized numpy operations for IDW. If performance is an issue, consider interpolating only within each basin's bounding box or using the Thiessen weight approach (which is instant per time step once weights are precomputed). But given modern computing, you can likely interpolate the whole region at a reasonable resolution and do zonal stats in a loop without trouble.

Robustness to Missing or Faulty Gauges

It's crucial to handle situations where one or more gauges fail to report or give erroneous readings at certain timesteps. **Robustness** means the basin averages shouldn't be thrown off wildly by a single bad gauge. Here are strategies and considerations:

- **Handling Missing Data:** If a gauge has no data for a timestep (e.g. communication dropout or maintenance), the interpolation methods discussed naturally accommodate this by simply using the remaining gauges. For Thiessen, you'd effectively omit that gauge – one way is to recompute the Thiessen polygons without it (the neighboring polygons would expand to cover its area). This can be done on the fly, but it's easier if missing data is infrequent. IDW and kriging seamlessly handle missing gauges by just not including them in the weighting. As long as other stations are in the vicinity, the gap will be filled by their influence. The impact is that areas near the missing gauge will take on the values of the next nearest data. If a large portion of the region has no active gauges (e.g. a widespread outage), then of course the surface becomes unreliable there. In summary, *missing gauge readings will typically just reduce the local information*, but your process should still produce an estimate rather than failing entirely. You should ensure your code checks for NaNs or missing values and skips those points when interpolating, so you don't propagate invalid data.

- **Detecting Outlier Readings:** A malfunctioning gauge might still report data, but it could be spurious (e.g. extremely high or stuck at a constant value). These errors can severely skew the interpolation if not caught. For example, if one gauge erroneously logs 200 mm in 10 minutes (far beyond anything plausible for your climate) due to a sensor fault, an interpolation would create a fictitious downpour around that station. To guard against this, implement basic **quality control (QC)** on the gauge data before or during interpolation. Common QC steps include:

- **Range checks:** Define physically reasonable limits (based on climate records) for 10-min rainfall. Anything above, say, 50 mm in 10 min (depending on what's realistic in Israel) might be flagged for verification.
- **Spatial consistency (buddy check):** Compare each gauge's reading to neighboring gauges' readings. If one station reports a high value while all surrounding stations (within a certain radius) report near-zero, the station is suspect. Meteorologically, isolated heavy convective bursts can occur, but if a gauge is a huge outlier by several orders of magnitude, it's likely an error. Automated QC algorithms often use a **neighbor median** or mean to identify outliers – e.g., flag a gauge if its value deviates greatly from the median of its five closest neighbors [12]. In practice, one might remove a gauge's data at a timestep if it differs from neighbors by more than some factor (taking into account that rain fields can be patchy – you don't want to remove a real extreme event, but truly faulty readings usually stand out).

- **Consistency over time:** If a gauge shows a sudden unrealistic jump or is stuck at a value (e.g. repeating the same value each interval, or zero when neighbors show rain), those are red flags. For instance, a **stuck gauge** that keeps reporting zero while others measure rainfall can be detected by checking if it's surrounded by non-zero reports – one method flags a gauge as "stuck" if rainfall is reported in all four quadrants around it but the gauge itself stays zero [13]. In such cases, you might treat the gauge as missing until it recovers.

Implementing these checks in Python can be done with pandas or numpy operations over the gauge dataframe. For example, you could at each timestep compute z-scores or ratio to neighbor average for all gauges and mask out those beyond a threshold. You can also maintain a list of known bad sensors and exclude them entirely if needed.

- **Robust Interpolation Choices:** Using interpolation that involves multiple gauges for each estimate inherently provides some robustness. **Why?** Because the influence is shared, an error at one gauge is diluted by others. For example, with IDW or kriging, if one gauge is reading abnormally, it will still affect the surface (especially right near that gauge), but other gauges' data will counterbalance it as distance grows. In contrast, the Thiessen method would assign that bad value to the entire polygon area with full weight, which could severely mis-estimate basin rainfall. So, moving away from a single-neighbor method (your old approach of 3 nearest gauges or pure Thiessen) toward a blended approach is already a big step in robustness. Kriging in particular, if you include a nugget or moderate smoothing, won't create a huge spike unless surrounding data support it. If a gauge is clearly an outlier, the kriging system (depending on variogram) might down-weight it as not correlating well with neighbors. Still, kriging will **honor the data point exactly** at its location if no measurement error is assumed, so you might consider **Kriging with a nugget** (which is like allowing some measurement error) – effectively this can prevent a single gauge from overly influencing the estimates at distance.

- **Filling or Imputation:** In cases where a gauge has a temporary gap in data, you might choose to fill that gap before interpolation to avoid losing coverage. One simple approach is to interpolate that gauge's value from neighbors *in time* or *space*. However, directly inferring a missing gauge value from others at the same timestep is essentially what spatial interpolation does anyway. So an explicit imputation may not be needed if your interpolation method doesn't require a complete data matrix. If you do want to fill missing gauge data (for record-keeping), you could use techniques like nearest-neighbor in space, or a regression from nearby gauges (e.g. if some gauges are highly correlated, use one to predict another). But this can get complicated and might not add much value beyond just excluding the missing gauge at that time.

- **Validation:** To ensure your approach is robust, you can perform some tests. For example, try removing one gauge's data and see how much the basin averages change (sensitivity analysis). Or if you suspect a gauge error on a known date, compare the interpolated basin rain with and without that gauge. If results are dramatically different, you know that gauge has a big influence – then double-check if that influence is deserved (was there truly a storm only that gauge caught?) or if it's likely spurious. In model training, having a smoother, less noisy input (basin average rain) can help learning, but you also don't want to smooth out real extremes too much. It's a balance.

- **Fail-safe and Monitoring:** For an operational system, you might implement a fail-safe: if a gauge is detected as malfunctioning, automatically exclude it from the interpolation until it is fixed, and

maybe notify that the network has a gap. Because you have 77 gauges, losing one or two for a short period is usually not catastrophic – the others can cover, with some increase in interpolation uncertainty in that locale. The robust design is to never rely on a single gauge's reading for critical decisions if you can help it; always corroborate with neighbors. By producing a spatial surface, you inherently do that.

In summary, to achieve robust basin averages: **choose an interpolation method that uses multiple gauges per basin (not one gauge per area)**, apply quality control to your gauge inputs (filter out obvious errors), and have a strategy for missing data (usually, just use the remaining gauges and perhaps interpolate over the gap). This will ensure that no single bad sensor reading will overly bias a basin's rainfall. The result will be a set of smoothed, area-averaged rainfall time series for your 35 basins that should serve as more stable input features for your model, compared to the raw nearest-gauge values. By accounting for spatial distribution and being resilient to point failures, you'll likely improve the model's performance and reliability.

References and Further Reading

- Spatial interpolation methods for rainfall are discussed in many hydrology/GIS resources. For example, common techniques like Thiessen polygons, IDW, splines, and kriging are overviewed in research on rainfall mapping [1] [2]. IDW assumes nearer gauges have more similar rainfall, which works best with dense, even station spacing [7]. It's noted that more advanced methods (e.g. kriging) can produce a more robust rainfall estimate in complex situations [6].
- A 2020 study compared Thiessen, IDW, multiquadric, and kriging for short-duration rains and found Thiessen can diverge for small catchments and sparse networks, where it often overestimates area rainfall, while IDW tends to underestimate peaks in those cases [3] [4]. As gauge density increases, however, all methods converge toward similar results [11].
- On using elevation: research indicates that incorporating DEM data via co-kriging or KED helps mainly for long-term accumulations (monthly/yearly rainfall), whereas at sub-hourly scales, the benefit is limited since rainfall variability is less tied to topography [10].
- For data quality, meteorological agencies employ neighbor-based QC checks. For instance, an **outlier check** might compare each gauge's reading to the median of neighboring stations to detect implausibly high values [12]. Likewise, a gauge reporting zero while all surrounding gauges see rain is flagged as a likely malfunction [13]. These principles can be adapted in your code to ensure faulty data doesn't skew your interpolation.

By implementing an interpolation-based rain surface with these considerations, you will obtain robust basin-average rainfall estimates ready for use in your hydrological model or machine learning pipeline. Good luck with your next stage of modeling, and enjoy the more stable and spatially representative rainfall inputs! [6] [3] [12]

---

[1] [3] [4] [11] Comparison of methods to estimate areal means of short duration rainfalls in small catchments, using rain gauge and radar data-Bohrium
https://www.bohrium.com/paper-details/comparison-of-methods-to-estimate-areal-means-of-short-duration-rainfalls-in-small-catchments-using-rain-gauge-and-radar-data/812638427211104257-26

2 9 10 Spatial interpolation of precipitation from multiple rain gauge networks and weather radar data for operational applications in Alpine catchments
https://crealp.ch/wp-content/uploads/2021/09/2018_Foehn_et_al_Spatial.pdf

5 6 7 8 Inverse Distance Weighting (IDW) Interpolation - GIS Geography
https://gisgeography.com/inverse-distance-weighting-idw-interpolation/

12 HESS - Technical note: A guide to using three open-source quality control algorithms for rainfall data from personal weather stations
https://hess.copernicus.org/articles/28/4715/2024/

13 Microsoft Word - AMS_06_Rev.doc
https://www.weather.gov/media/owp/oh/hrl/docs/AMS_06_Rev.pdf