[AM-02-006] Grid Operations and Map Algebra

Abstract

Grid operations are manipulation and analytical computations performed on raster data. Map Algebra is a language for organizing and implementing grid operations in Geographic Information Systems (GIS) software, and is typically categorized into Local, Focal, and Zonal functions, where each function typically ingests one or more grids and outputs a new grid. The value of a specific grid cell in the output grid for Local functions is determined from the value(s) of the analogous cell position(s) in the input grid(s), for Focal functions from the grid cell values drawn from a neighborhood around the specific output grid cell, and for Zonal functions from a set of grid cells specified in a separate zone grid. Individual functions within a category vary by applying a different arithmetic, statistical, or other type of operator to the function. Map Algebra also includes Global and Block function categories. Grid operations can be categorized as data manipulation procedures or within domain-specific applications, such as terrain analysis or image processing. Grid operations are employed in a variety of GIS-based analyses, but are particularly widely used for suitability modeling and environmental analyses.

Keywords: basic analytical operations, cartographic modeling, focal function, local function, raster, zonal function

Author & citation

Mennis, J. (2022). Grid Operations and Map Algebra. The Geographic Information Science & Technology Body of Knowledge (3rd Quarter 2022 Edition). John P. Wilson (Ed.). DOI:10.22224/gistbok/2022.3.1

This Topic is also available in the following editions:

DiBiase, D., DeMers, M., Johnson, A., Kemp, K., Luck, A. T., Plewe, B., and Wentz, E. (2006). Map algebra. The Geographic Information Science & Technology Body of Knowledge. Washington, DC: Association of American Geographers.

Explanation

1. Definitions
2. Grid Operations
3. Map Algebra
4. Extensions to Map Algebra
5. Other Types of Map Algebra and Grid Operations
6. Implementation of Grid Operations and Map Algebra in GIS Software
7. Applications of Grid Operations and Map Algebra for GIS-Based Analyses

1. Definitions

Grid Operation: data manipulation and analysis procedures performed on raster data

Map Algebra: a language for organizing and implementing grid operations in GIS

Local Function: a Map Algebra function in which the value of each grid cell in the output grid is determined solely by the value(s) encoded in the analogous grid cell position in the input grid(s)

Focal Function: a Map Algebra function in which the value of each grid cell in the output grid is determined by the values of a set of grid cells within the neighborhood of the analogous grid cell position in the input grid

Focal Neighborhood: a set of neighboring grid cells used to calculate a value in a Map Algebra Focal function, typically defined as a rectangle, circle, or wedge shape surrounding or emanating from the focal cell position

Zonal Function: a Map Algebra function which summarizes the grid cells in a value grid by the zones defined in a zone grid

Operator: An arithmetic, relational, statistical, trigonometric, or Boolean/bitwise operation used in a Map Algebra function.

2. Grid Operations

Grid operations are manipulation and analytical computations performed on raster data, a data model defined by an exhaustive partition of space into a regular geometric tessellation, most commonly a tessellation of square ‘grid cells.’ The term ‘grid’ is used here to refer to a single raster layer in Geographic Information Systems (GIS) software (Pingel 2018; Williams 2019). Grid operations are similar in concept to image algebra and other image processing procedures performed on remotely sensed or digital graphic imagery, which share with raster GIS the conceptual data model of a grid, and in which grid cells are commonly referred to as ‘pixels.’ Grid operations in GIS, however, typically involve a set of operations that are particularly germane to the types of social, natural, and built environment analyses commonly carried out in a raster GIS software environment.

Map Algebra is a language for organizing and implementing GIS-oriented grid operations (Tomlin 2012) that forms the basis of raster analysis in most GIS software packages. Map Algebra can also be considered an organizing principle that is used to classify various types of grid operations in the context of a GIS software interface so that they can be accessed and implemented easily by a GIS analyst. Map Algebra is closely related to Cartographic Modeling, a framework for encoding a series of Map Algebra and other types of analytical operations in a coherent representation in order to solve a particular analytical task in GIS (Buttenfield and Charisoulis 2021).

3. Map Algebra

Map Algebra encompasses a set of specific analytical functions applied to raster data, where each function typically ingests one or more grids and outputs a new grid. Map Algebra functions are categorized into types, primarily categories of Local, Focal, and Zonal

functions. Each function category has a set of specific functions within it, where all functions within a particular category share the same conceptual basis for how the computation of the function proceeds. Specific Map Algebra functions within a category vary by applying a different operator to the function, where arithmetic (e.g. addition, multiplication, power), relational (e.g. equal to, less than, greater than), statistical (e.g. minimum, maximum, mean), trigonometric (e.g. sine, cosine, tangent), and Boolean/bitwise (e.g. AND, OR, XOR) operators are typically employed.

The initial formulation of Map Algebra by Tomlin (1990) specifies a notation and syntax to identify the input grid or grids, the output grid, and the parameter settings, where each specific function is operationalized in what is referred to as a 'statement.' Consider for example, a Local function that ingests two grids, where for the sake of this example they are referred to as INPUTGRID_A and INPUTGRID_B, and outputs the grid named OUTPUTGRID. The function can be expressed in the statement:

OUTPUTGRID = LocalFUNCTION of INPUTGRID_A and INPUTGRID_B

where 'LocalFUNCTION' indicates a general Local function that can utilize a particular statistical operator. For example, the specific Local function 'LocalSUM' would indicate that grid cell values of INPUTGRID_A and INPUTGRID_B are added to together to produce the value encoded in OUTPUTGRID, and the term 'LocalMINIMUM' would indicate the minimum value occurring in the two input grids would be encoded in the output grid. Such syntax allows for the relatively easy combination of multiple Map Algebra statements into more complex procedures, where the output of one statement can be ingested as an input into another statement. Thus, a series of Map Algebra statements can be strung together in a sequence to solve a particular GIS-based analytical task, and implemented and expressed within a Cartographic Model.

3.1 Local Functions

Local functions ingest one or more grids, and produce an output grid. Local functions compute over a single grid cell, or on a 'cell-by-cell' basis, i.e. the value of each grid cell in the output grid is determined solely by the value(s) encoded in the analogous grid cell position in the input grid(s), and not by the values in other grids cell positions. Consider for example, the LocalSUM operation shown in Figure 1, which shows an abstract example of a 5x5 cell grid with small integers simply to illustrate the principle of the Local function computation. Here, two input grids, grids A and B, are ingested into the LocalSUM function to produce the output grid. We can identify each grid cell position based on its row and column (i.e. [row, column]) position, where rows begin at the top at row 0 and columns begin at the left at column 0. The top, left (northwest) row and column position is thus [0,0], which has an input grid A value of '1'. The grids overlay each other in geographic space; thus, the grid cell position [0,0] represents the same location on the surface of the earth for each of the input grids and the output grid.

The nature of the Local function implies that the value of a grid cell in the output grid relies solely on the values for the analogous grid cell position in the input grids. For example, for the LocalSUM function shown in Figure 1, consider the calculation for the output grid cell position [0,0], shown in bold outline in each of the grids. To calculate the value for [0,0] in the output grid, the values in the analogous grid cell positions in the input grids are added together: 1 + 4 = 5. This computation is simply applied to each grid cell position in turn;

thus, for the grid cell position [0,1], the output grid cell value is calculated as 1 + 5 = 6, and for [3,4] the output grid cell value is calculated as 2 + 10 = 12.

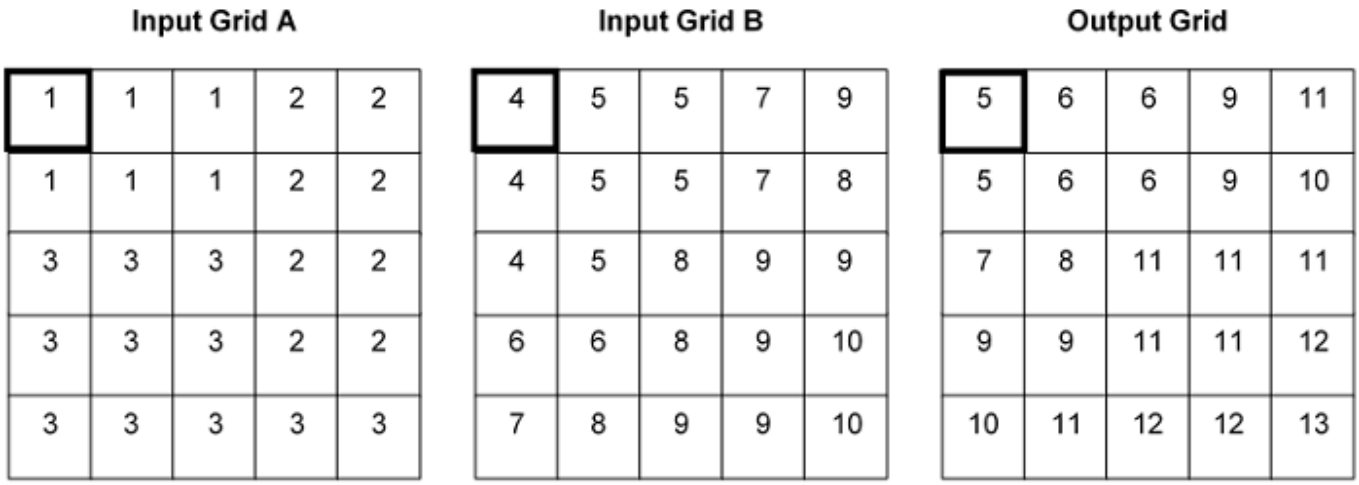


Figure 1. An example of a LocalSUM operation, where two input grids are ingested to produce an output grid. Values from analogous grid cell positions in the two input grids are added together and the resulting value encoded in the analogous grid cell position in the output grid. Source: author.

Note that the LocalSUM function applied to two input grids shown here is just one specific example of a Local function. If, instead of a LocalSUM, a LocalMINIMUM had been used, the value in the output grid for [0,0] would be '1,' since that is the minimum value occurring for that grid cell position in the input grids. Likewise, a LocalMAXIMUM function would yield a value of '5' and a LocalMEAN would yield a value of '2.5' for [0,0]. A variety of other statistical operators can be employed. It is also possible to apply a Local function to three or more input grids, or to just one input grid and a scalar value. For instance, in the latter case, if the LocaSUM show in Figure 1 ingested only input grid A and the scalar value '7,' then '7' would be added to each grid cell value in the input grid to produce the output value for each analogous grid cell position, i.e. the value in the output grid for [0,0] would be 1 + 7 = 8.

Local functions are used to facilitate raster overlay (Cai 2022), for example in employing a LocalSUM function to generate a raster representing the habitat suitability for a particular plant species based on combining standardized grids capturing elevation, precipitation, or other relevant habitat characteristics. Local functions are also used in raster selection, for example in employing a Local relational operation to generate a raster representing coastal areas vulnerable to flooding by identifying grid cells in an elevation grid that fall below a certain threshold elevation value.

3.2 Focal Functions

Focal functions typically ingest one input grid and produce an output grid. Focal functions compute over a neighborhood – the value of a grid cell position in the output grid, referred to as the focal cell, is based on the values of a set of grid cells within the neighborhood of

the focal cell. The neighborhood is typically configured as a fixed geometric shape or measurement that defines an area surrounding the focal cell. Each grid cell position takes its turn as the focal cell in the Focal function computation in an iterative procedure. The principle of Focal functions, and the definition of a neighborhood to facilitate computation, is known by a variety of names in different domains and is similar to that of the moving window, convolution, filter, or kernel operations in image processing contexts. Typically, however, in image processing, the neighborhood is defined as a square matrix of grid cells centered on the focal cell, whereas in Focal functions, while such a neighborhood configuration is certainly commonly employed, more complex neighborhood definitions are often used.

Figure 2 illustrates the computational principle of a Focal function using a simple 5x5 cell grid and a neighborhood defined as a 3x3 cell matrix centered on the focal cell. The focal cell in the output grid, grid cell position [1,1] is shown in black bold outline. The focal cell is shown similarly in the input grid, with the neighborhood shown in gray bold outline. The value of the focal cell in the output is based on the values of the nine cells in the neighborhood of the focal cell. Each grid cell takes its turn as the focal cell, where the corresponding neighborhood moves along with the focal cell, as is illustrated by the arrow, hence the term 'moving window.' Note the neighborhood for one cell position overlaps the neighborhood for adjacent cell positions. Computation is adjusted to account for cases in which the focal cell occurs at the edge of the grid, or where the neighborhood extends beyond the boundary of the grid.

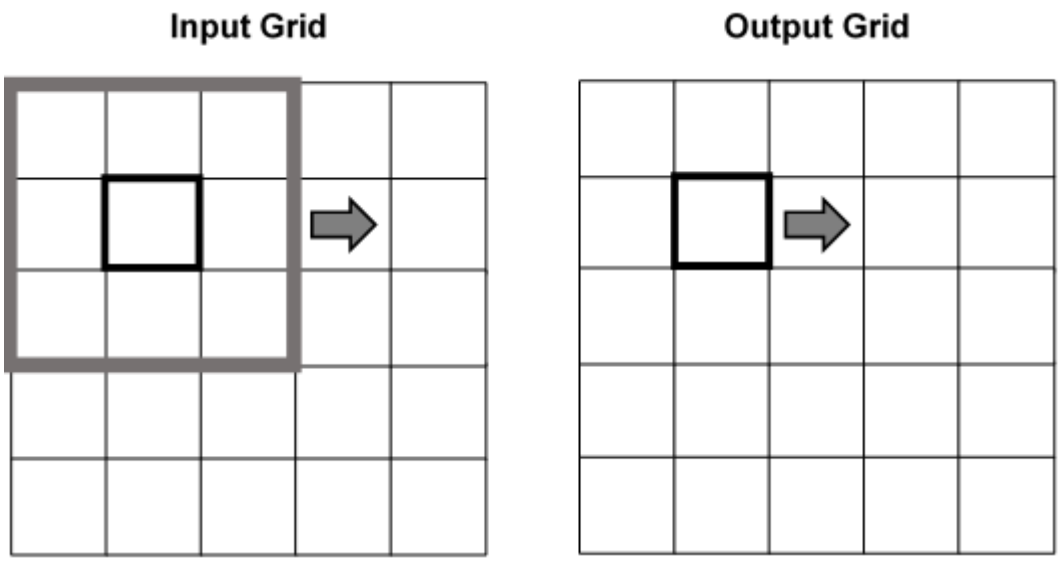


Figure 2. The principle of a Focal function. A 3x3 cell neighborhood (shown in bold gray) is placed over a focal cell (shown in bold black) in the input grid. Values within the neighborhood of the focal cell are aggregated via a statistical function and the resulting value is encoded in the analogous focal cell position in the output grid. The neighborhood is then 'moved' from cell to cell in the input grid such that each cell takes its turn iteratively as the focal cell in the function, encoding values for all the analogous cell positions in the output grid. Source: author.

Various statistical operators as described above, such as the minimum, maximum, or mean, can be applied to the Focal function calculation. Figure 3 shows an example of a

FocalMAXIMUM using a 3x3 cell neighborhood centered on the focal cell, applied to an input grid and producing an output grid. The value for each grid cell position in the output grid is calculated as the maximum value occurring in the neighborhood of the focal cell position in the input grid. For the focal cell [1,1] the following set of values in the input grid occur: [4, 5, 5, 4, 5, 5, 4, 5, 8]. The maximum value is clearly '8,' thus this value is encoded in the focal cell position in the output grid. This procedure is applied to each grid cell position in turn, with the maximum value within the neighborhood encoded in the focal cell position in the output grid. Notably, since the neighborhoods for adjacent grid cells overlap, a single grid cell value in the input can serve as the maximum value for multiple focal cells. This is the case with cells [2,3] and [2,4], both which have a value of '10' in the output grid, which is the value occurring in cell [3,4] in the input grid, which falls within the neighborhoods of both the [2,3] and [2,4] focal cell positions.

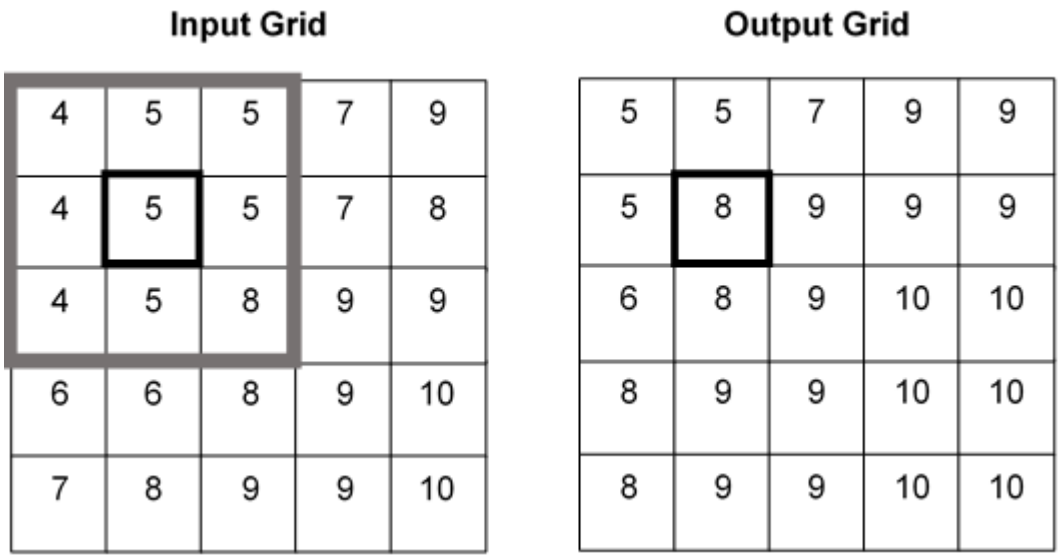


Figure 3. An example of a FocalMAXIMUM operation employing a 3x3 cell neighborhood, where an input grid is ingested to produce an output grid. The maximum (i.e. highest) value occurring among the nine cells that fall within a 3x3 cell neighborhood centered on the focal cell in the input grid is encoded in the same grid cell position in the output grid. Source: author.

Though Figure 3 shows a simple neighborhood configured as a 3x3 cell matrix centered on the focal cell, many other neighborhood configurations can be employed (Figure 4): 1) the square matrix can be enlarged to a greater number of cells, 2) different row and column cell dimensions can be specified (resulting in a rectangular neighborhood), 3) a radius can be employed to define the neighborhood, resulting in circular neighborhood shape, 4) an annulus (or doughnut shape) can be employed to proscribe a neighborhood within one distance but not within another distance, or 5) a wedge shape neighborhood can be employed by specifying a radius and a set of angular restrictions. The parameterization of the Focal function refers to the choice of neighborhood configuration chosen by the analyst. It is also possible to 'shift' the neighborhood in a particular distance and direction from the focal cell so that, say, the focal cell does not sit at the center of a square matrix neighborhood but is spatially offset. Other customized or bespoke neighborhood definitions that utilize more complex geometric shapes, or which may be based on some other

underlying raster data, such as a terrain or social data surface, are also theoretically possible, though these are not typically readily available in commonly used commercial or open source GIS software packages.

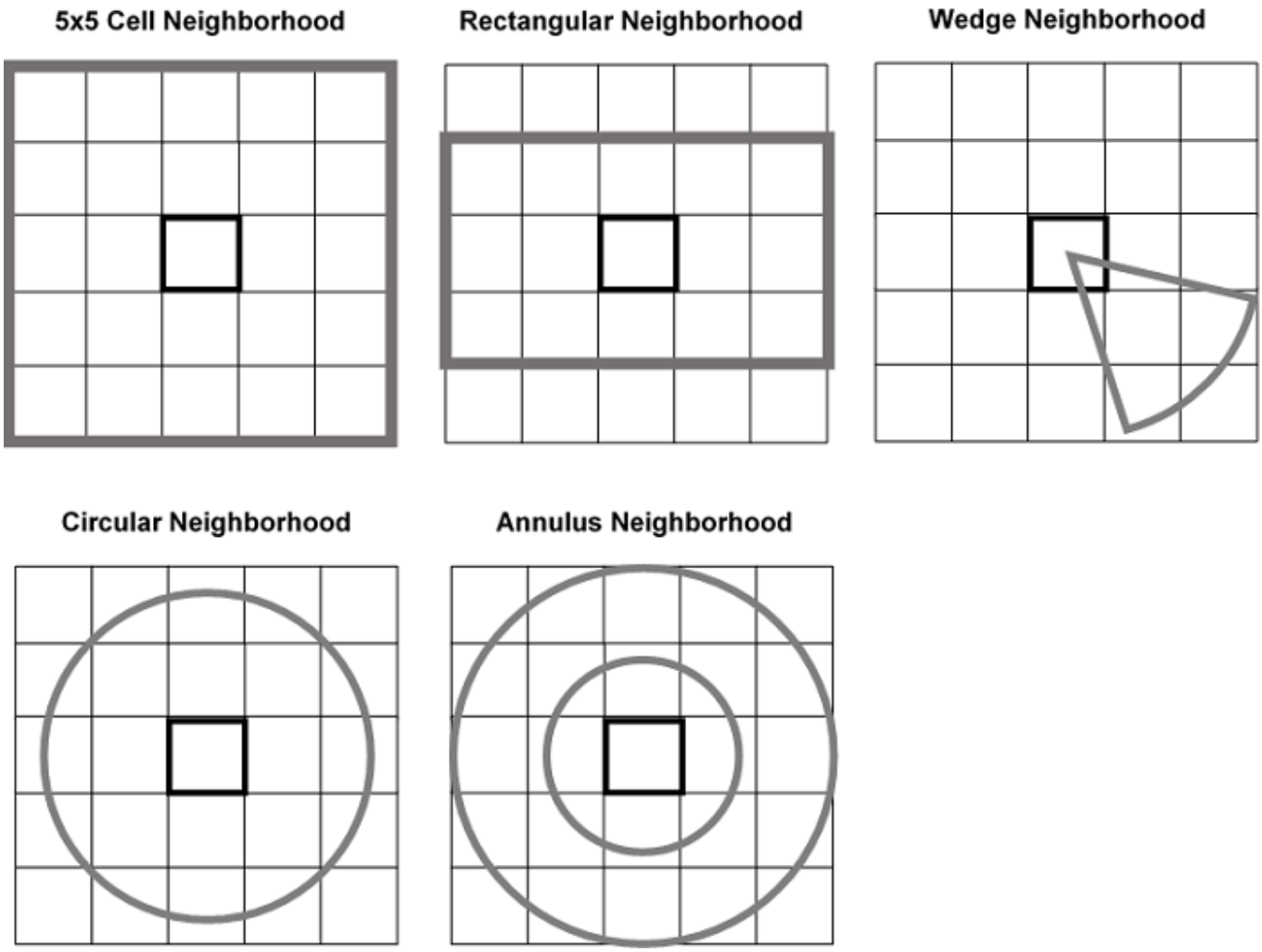


Figure 4. Various parameterizations of a Focal function neighborhood. The focal cell is shown in bold black outline and its corresponding neighborhood is shown in bold gray outline. Common neighborhood configurations include square and rectangular arrays, a wedge shape, a circular neighborhood, and an annulus (doughnut shape). Typically, grid cells that have their centroid (geometric center) within the neighborhood of the focal cell are considered within the neighborhood and used in the focal calculation. Source: author.

3.3 Zonal Functions

Zonal functions ingest two grids, in which one is specified as the ‘zone grid’ and another as the ‘value grid,’ and produce either a grid or a table as output. Zonal functions summarize the grid cells in the value grid by the zones defined in the zone grid, where the analyst specifies the summary statistical operator employed. Figure 5 provides an example of a ZonalMAJORITY function. Here, the zone grid specifies unique zones, where grid cells that share the same value are considered to belong to the same zone (whether the grid cells are adjacent or not). In the zone grid in Figure 5, there are three unique zones with values ‘1,’ ‘2,’ and ‘3.’ For each unique zone, the majority (the most commonly occurring) value

among the grid cells in the value grid that fall within the analogous cell positions for that zone is encoded in the same grid cell positions for that zone in the output grid. Thus, for zone 1, outlined in black bold, which includes the set of values [4, 5, 5, 4, 5, 5] in the value grid, each of the analogous zone 1 cell positions in the output grid will encode the value '5,' as it is the most commonly occurring value in the set.

Zone Grid

1	1	1	2	2
1	1	1	2	2
3	3	3	2	2
3	3	3	2	2
3	3	3	3	3

Value Grid

4	5	5	7	9
4	5	5	7	8
4	5	8	9	9
6	6	8	9	10
7	8	9	9	10

Output Grid

5	5	5	9	9
5	5	5	9	9
8	8	8	9	9
8	8	8	9	9
8	8	8	8	8

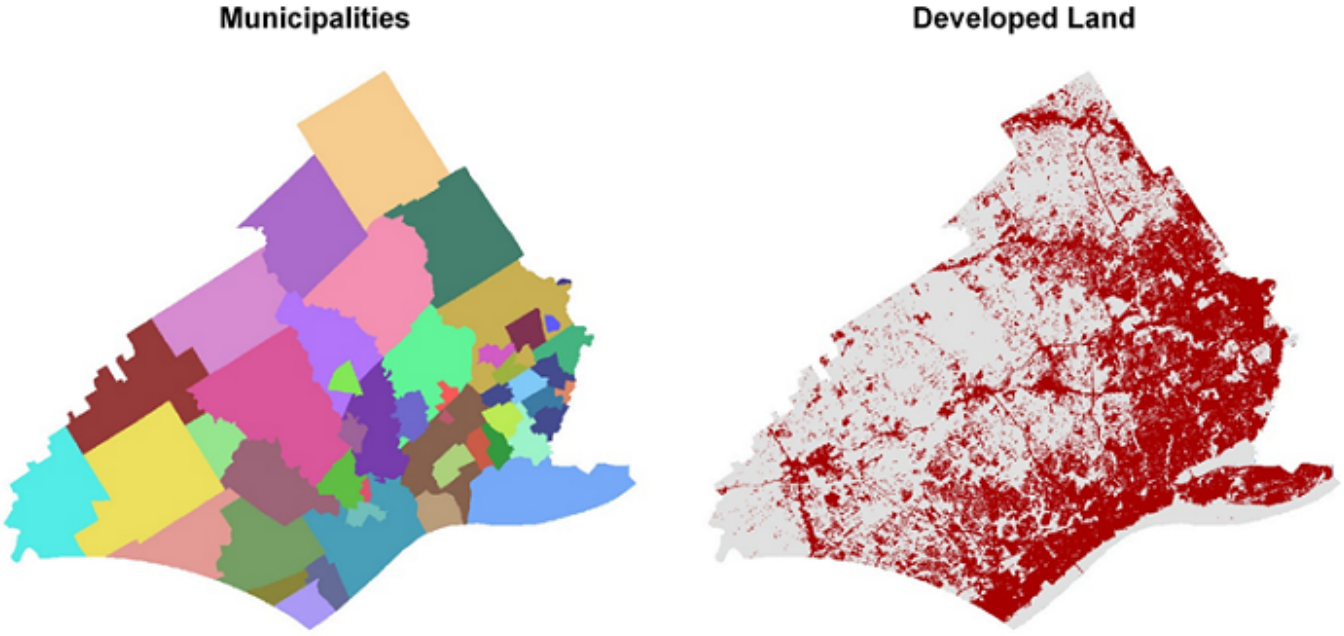
Zone	Count	Majority
1	6	5
2	8	9
3	11	8

Figure 5. An example of a ZonalMAJORITY operation, where a zone grid and a value grid are ingested to produce an output grid. For each zone specified as a unique value in the zone grid, e.g. zone '1,' outlined in black bold, the majority (most commonly occurring) value in the analogous cell positions in the value grid is encoded in the same cell positions in the output grid. The output of a Zonal function can also be expressed as a table (bottom, left), where each row in the table represents a zone, and the count of the number of grid cells in each zone is reported, along with the majority value. Source: author.

A commonly used alternative is to output a table instead of a grid, where each row in the table represents a unique zone, and the table includes a field to encode the number of grid cells in each zone (count) and the summary statistic for the statistical operator chosen, in this case, the majority. Figure 4 shows the resulting table for the example, where zone 1 has a count of six grid cells and '5' is the majority value.

An application of a Zonal function to urban land cover analysis is shown in Figure 6. A raster layer of individual municipalities is shown in Figure 6, top left, where each grid cell encodes a unique value for each municipality. A raster layer of developed (i.e. urbanized or 'built-up') land is shown in Figure 6, top right, where grid cells colored red are developed and have a value of '1' and grid cells colored gray are undeveloped and have a value of '0.' A

ZonalSUM function was used to summarize the developed land grid (value grid) by the municipalities grid (zone grid), yielding a table (shown in Figure 6, lower left) where each row in the table represents a municipality (GEOID is the unique identifier for each municipality), the COUNT field is the total number of grid cells (developed or undeveloped) in each municipality, and the SUM field is the sum of the developed land grid cell values within each municipality. Since each developed grid cell has a value of '1,' the SUM field indicates the total number of developed land grid cells within each municipality. Dividing the COUNT by the SUM would yield the percentage of developed land for each municipality. Note that only a subset of the records in the table are shown for brevity.



Developed Land by Municipality

GEOID *	COUNT	SUM
4204500676	1715	1468
4204503336	16822	8657
4204506024	15572	4719
4204509080	4913	3427
4204512442	25093	2403
4204513208	17288	12119
4204513212	4110	2574
4204513232	6373	1128
4204514264	1816	1588
4204515232	2500	1894
4204515432	745	545

Figure 6. An application of a ZonalSUM function for calculating the area of developed land by municipality in Delaware County, Pennsylvania, USA. The Municipalities grid (top left) serves as the zone grid, where each municipality has a unique grid cell value, and the Developed Land grid (top right) serves as the value grid, where developed land in red has a

value of ‘1’ and undeveloped land in gray has a value of ‘0.’ The function outputs the Developed Land by Municipality table, which reports the total number of developed land grid cells in each municipality in the SUM field. Source: author.

4. Extensions to Map Algebra

The original Map Algebra, which was developed expressly for analyses of conventional two-dimensional, raster data sets representing properties of planimetric space, has been extended by researchers in a variety of ways. It has been adapted for three-dimensional spatial data, where raster grid cells are extended to three-dimensional volumetric pixels, or ‘voxels,’ and two-dimensional and temporal raster data, where the third dimension is used to represent time instead of elevation, such that each voxel represents an area over space and a duration over time (Mennis et al 2005; Gebbert and Pebesma 2014). The same principle has been used to represent three-dimensional spatial and temporal raster data, using a four-dimensional construct (Mennis 2010). In these dimensional extensions to Map Algebra, the initial Map Algebra computations are simply extended in principle to the additional dimensions, so, for example, LocalSUM function applied to two three-dimensional input ‘grids’ would simply add the values of analogous voxel positions in the input and encode the sum in the analogous voxel position in the three-dimensional output ‘grid.’ Other extensions have focused on more sophisticated notions of regions and zones in computation (Cordeiro et al 2009), support for cellular automata modeling (Pullar 2001), vector- and network based modeling (French and Li 2010; She and Li 2016), and Map Algebra functions that employ calculus (Haklay 2004).

5. Other Types of Map Algebra and Grid Operations

Map Algebra also includes categories of Block functions and Global Functions. Block functions utilize non-overlapping neighborhoods, or blocks, on which the summary statistic is calculated and encoded for every grid cell position in the block in the output grid. Block functions can thus be conceptualized as a special case of a Zonal function, where each block serves as a separate zone. Global Functions are those that potentially utilize all grid cells in an input grid to calculate the value for a grid cell in the output grid.

There are many other types of grid operations that are not typically considered within the conventional Map Algebra categories. Such grid operations often take on the characteristics of certain Map Algebra functions, but due to their common use or domain-specific application they may carry a separate designation or presentation in the GIS software interface apart from Map Algebra. Grid operations which are commonly used for data manipulation, transformation, or preprocessing, i.e. for preparing spatial data for entry into more formal analytical sequences, including for entry into other Map Algebra functions, often fall into this category, and include:

- Masking, e.g. extracting or ‘clipping’ a grid based on the values of another grid or a geometric shape
- Reclassification, e.g. changing a set of values in a grid to another set of values based on a look-up table
- Density, e.g. calculating a grid representing the number of features per unit area

- Distance, e.g. calculating a grid representing the distance to a particular feature, or the least cost path to a particular feature
- Interpolation, e.g. calculating a grid representing a continuous a surface of values from a set of sample observations

Other types of grid operations which are not typically designated in terms of Map Algebra functions are those that are commonly presented in the context of a domain-specific application. Here, a set of specific grid operations may be grouped in a GIS software package according to the topical analytical domain in which they are often employed, such as:

- Terrain Operations, e.g. calculating a grid representing the slope or aspect, or the areas that may be a visible from a particular vantage point
- Hydrographic Operations, e.g. calculating a grid representing the flow direction or the upslope contributing area
- Image Processing Operations, e.g. smoothing or edge enhancement for interpretation of remotely sensed imagery
- Change Detection Operations (applied to remotely sensed imagery), e.g. image differencing to detect land cover change

Most of these other types of data manipulation or application domain-specific grid operations, however, adhere to the principles of one of the Map Algebra categories and can be classified as a type of Map Algebra function. For instance, the calculation of slope from a grid of elevation (a digital elevation model, or DEM) is essentially a Focal function that calculates the angle of steepest descent over the neighborhood of a focal grid cell; an image differencing operation used in remote sensing change detection is essentially a Local function where the values of one grid are arithmetically combined with (e.g. subtracted from) another grid; and a Euclidean distance operation can be considered a Global function that calculates the straight-line distance from each grid cell position to the nearest feature (Figure 7).

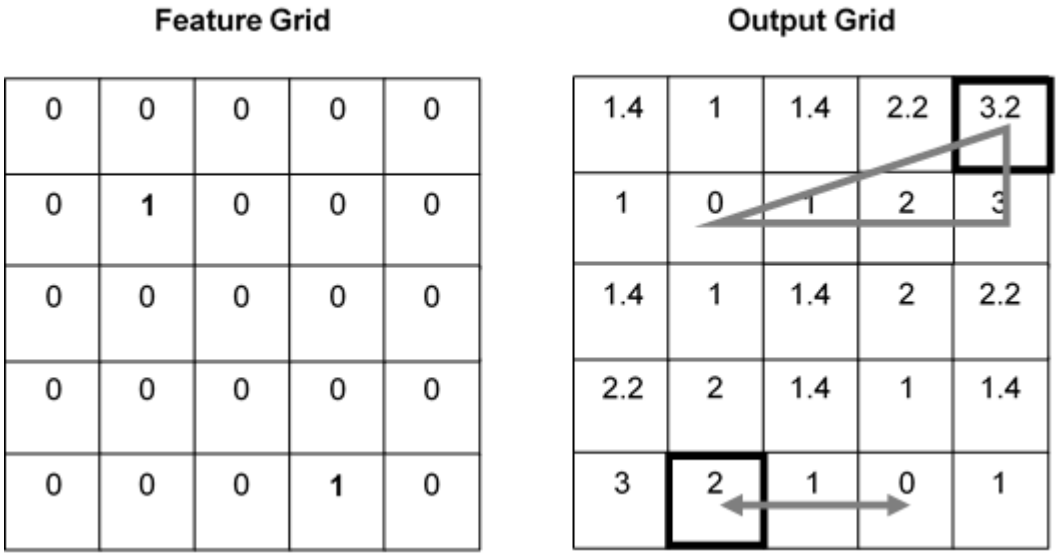


Figure 7. An illustration of the computation of a Euclidian distance grid operation, an example of a Global function, which produces a grid encoding the straight-line distance to

the nearest feature. In this example, distances in the output grid are calculated assuming a raster resolution = '1' unit, i.e. the length along the edge of a single grid cell represents 1 unit on the ground. The Feature Grid encodes the locations of the features from which distance is calculated, where '1' (in bold text) encodes the presence of the features, and '0' the absence. The output grid encodes the straight-line distance to the nearest feature for each grid cell, where measurements extend between grid cell centroids, and the cell positions where the features occur have a distance of '0.' Distance values for other grid cells in a cardinal direction from a feature grid cell can be calculated simply as the difference in the appropriate row (north-south) or column (east-west) dimension, as for the [4,1] cell position, shown in bold outline, which is a distance of 2 units directly west of the nearest feature grid cell. For other grid cells, the Euclidean distance formula can be used to calculate the distance as the length of the hypotenuse of a right triangle, as for the [0,4] cell position, also shown in bold outline, which lies 3 units east and 1 unit north of the nearest feature grid cell, resulting in a distance of 3.2 units. Source: author.

6. Implementation of Grid Operations and Map Algebra in GIS Software

Nearly all commercial, government, academic, and open source GIS software packages that support raster data handling and analysis include extensive grid operations and Map Algebra functions, including ArcGIS Pro (ESRI, Inc.), GeoMedia (Hexagon Geospatial, Inc.), IDRISI GIS (Clark Labs), GRASS GIS (GRASS Development Team), Manifold GIS (Manifold, Inc.), QGIS (QGIS Association), and MapInfo Pro, (Precisely, Inc.), among others. Certain grid operations are also available in many other software computing platforms and programming languages, such as python, R, and Matlab (Mathworks, Inc). Because of the large storage requirements of many raster data sets and the iterative nature of many raster-based algorithms, grid operations can often be computationally intensive. In response, parallel processing approaches have been developed to improve grid operation performance in some GIS software packages (Qin et al 2014; Shook 2019). Many GIS software packages also support grid operations that can ingest vector data as an input, such as the use of a vector data layer specifying zones in a Zonal function, though typically such vector data are simply converted to raster format by the algorithm prior to the application of the grid operation itself.

7. Application of Grid Operations and Map Algebra for GIS-Based Analysis

Grid operations and Map Algebra are employed in a wide variety of social and environmental GIS analyses. Multi-criteria decision analysis (MCDA) (Malczewski 1999; Rinner 2018), and particularly the application of MCDA for suitability modeling, often utilize a series of grid operations in a weighted linear overlay, where the output is typically a grid representing the suitability values for a particular land use, development, or facility location. Remotely sensed imagery has a native raster format when ingested into GIS software; thus, grid operations are widely used for manipulating and analyzing such data (Campbell & Wynne 2011). For example, grid operations are used to 1) process images to suppress or enhance variability within an image to support analysis or visualization, 2) derive environmental indices by band ratio or other types of band combination, or 3) detect and quantify land cover change using time series imagery. Additionally, data representing

environmental characteristics which are typically conceptualized and modeled as a continuous surface, such as topography, land cover, precipitation, and other environmental properties, are often derived from remotely sensed imagery or interpolated from monitoring stations or sensors, and therefore are often encoded in raster format. Grid operations are therefore widely employed in GIS-based analyses of land cover change, terrain analysis, habitat modeling, hydrologic modeling, and other analyses that make extensive use of such environmental raster data sets (Wilson and Gallant 2000).

References

Buttenfield, B., & Charisoulis, G. (2021). Cartographic Modeling. The Geographic Information Science & Technology Body of Knowledge (1st Quarter 2021 Edition), John P. Wilson (ed.).

Cai, H. (2022). Overlay. The Geographic Information Science & Technology Body of Knowledge (1st Quarter 2022 Edition), John P. Wilson (ed.).

Campbell, J. B., & Wynne, R. H. (2011). Introduction to Remote Sensing, 5th Edition. New York, NY: The Guilford Press.

Cerveira Cordeiro, J. P., Câmara, G., Moura de Freitas, U. et al. (2009). Yet Another Map Algebra. Geoinformatica 13: 183–202.

French, K., & Li, X. (2010). Feature-based cartographic modeling. International Journal of Geographical Information Science, 24, 141–164.

Gebbert, S., & Pebesma, E., 2014. A temporal GIS for field-based environmental modeling. Environmental Modelling and Software, 53: 1-12.

Haklay, M. (2004) Map Calculus in GIS: a proposal and demonstration, International Journal of Geographical Information Science, 18: 107-125.

Malczewski, J. (1999). GIS and Multicriteria Decision Analysis. John Wiley & Sons.

Mennis, J. (2010). Multidimensional Map Algebra: Design and Implementation of a Spatio-Temporal GIS Processing Language. Transactions in GIS, 14, 1–21.

Mennis, J., Viger, R., & Tomlin, C. D. (2005). Cubic Map Algebra Functions for Spatio-Temporal Analysis. Cartography and Geographic Information Science. 32(1): 17-32.

Pingel, T. (2018). The Raster Data Model. The Geographic Information Science & Technology Body of Knowledge (3rd Quarter 2018 Edition), John P. Wilson (Ed.).

Pullar, D. (2001). MapScript: A Map Algebra Programming Language Incorporating Neighborhood Analysis. GeoInformatica, 5(2), 145–163.

Qin, C.-Z., Zhan, L.-J., Zhu, A.-X., & Zhou, & C.-H. (2014). A strategy for raster-based geocomputation under different parallel computing platforms. International Journal of

Geographical Information Science, 28(11), 2127–2144.

Rinner, C. (2018). Spatial Decision Support. The Geographic Information Science & Technology Body of Knowledge (2nd Quarter 2018 Edition), John P. Wilson (Ed).

She, J. & Li, X. (2016). Map algebra based analysis for directed flow networks. Transactions in GIS, 20: 356-367.

Shook, E. (2019). GIS and Parallel Programming. The Geographic Information Science & Technology Body of Knowledge (1st Quarter 2019 Edition), John P. Wilson (Ed.).

Tomlin, C. D. (1990). Geographic Information Systems and Cartographic Modeling,1st Edition. Englewood Cliffs, New Jersey: Prentice-Hall (now Pearson Education).

Tomlin, C. D. (2013). GIS and Cartographic Modeling. Redlands, CA: Esri Press.

Williams, C. (2019). Raster Formats and Sources. The Geographic Information Science & Technology Body of Knowledge (4th Quarter 2019 Edition), John P. Wilson (Ed.).

Wilson, J. P. & Gallant, J. C., Eds. (2000). Terrain Analysis: Principles and Applications. John Wiley & Sons.