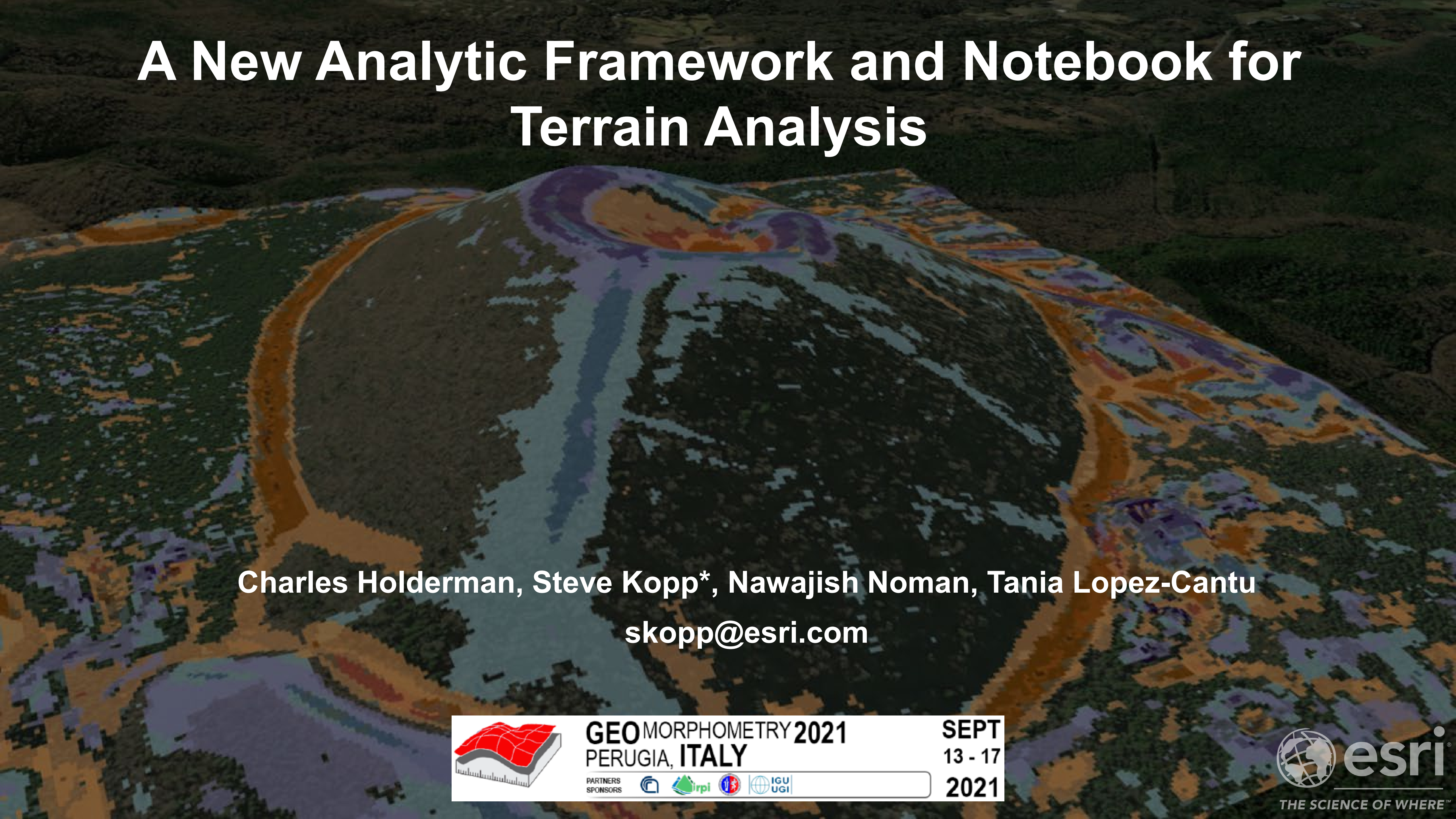
A New Analytic Framework and Notebook for Terrain Analysis



Charles Holderman, Steve Kopp*, Nawajish Noman, Tania Lopez-Cantu

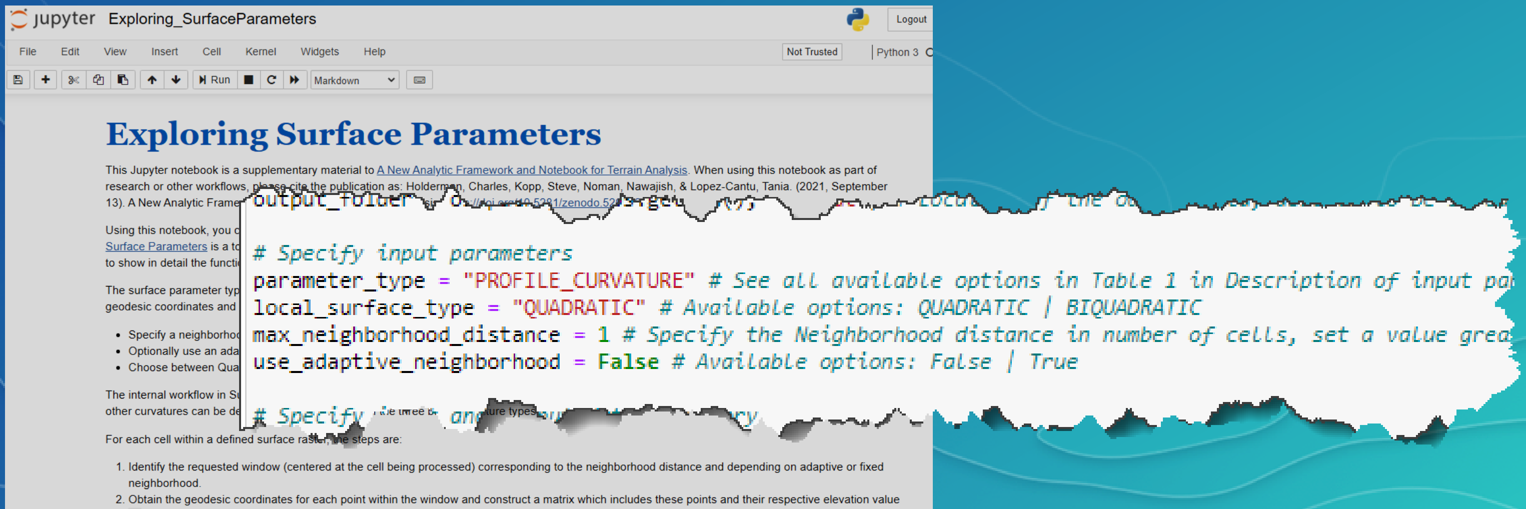
skopp@esri.com

GEOMORPHOMETRY 2021
PERUGIA, ITALY
SEPT 13 - 17 2021
PARTNERS SPONSORS irpi IGU UGI

esri
THE SCIENCE OF WHERE

A Single Tool and Companion Jupyter Notebook

- Many terrain analysis metrics in a single ArcGIS tool
  - Easy to find many terrain metric
  - Easy to re-run with similar options
  - Easy to add new terrain metrics
- Notebook to show how it works
  - Modify and experiment with our settings
  - Add your own new metrics reusing our framework


jupyter Exploring_SurfaceParameters
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Markdown

Exploring Surface Parameters

This Jupyter notebook is a supplementary material to A New Analytic Framework and Notebook for Terrain Analysis. When using this notebook as part of research or other workflows, please cite the publication as: Holderman, Charles, Kopp, Steve, Noman, Nawajish, & Lopez-Cantu, Tania. (2021, September 13). A New Analytic Frame

Using this notebook, you c
Surface Parameters is a to
to show in detail the functi

The surface parameter typ
geodesic coordinates and

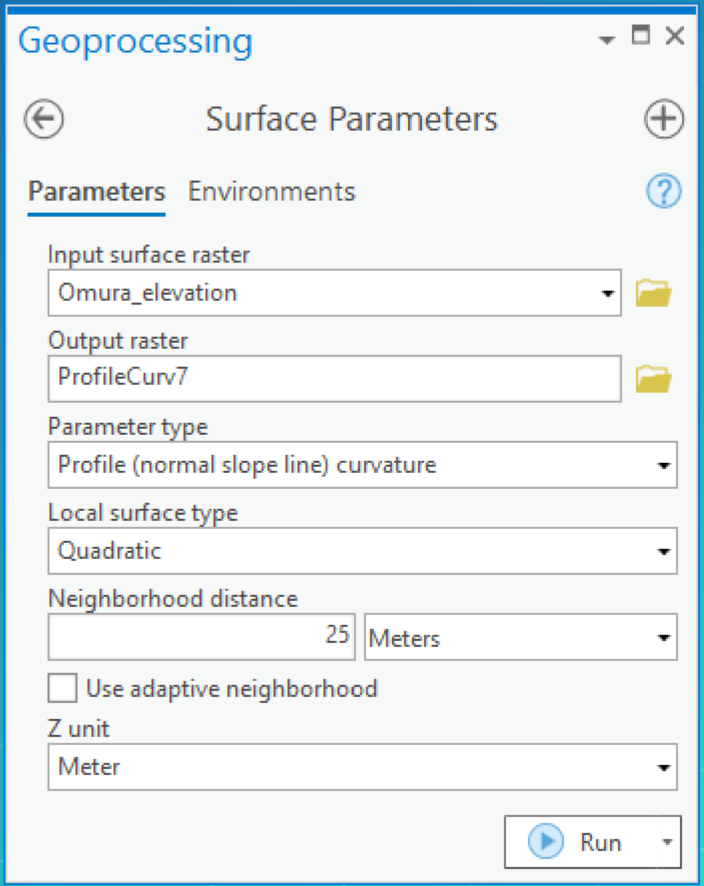
- Specify a neighborho
- Optionally use an ada
- Choose between Qua

The internal workflow in Su
other curvatures can be de

For each cell within a defined surface raster, the steps are:

1. Identify the requested window (centered at the cell being processed) corresponding to the neighborhood distance and depending on adaptive or fixed neighborhood.
2. Obtain the geodesic coordinates for each point within the window and construct a matrix which includes these points and their respective elevation value

```
# Specify input parameters
parameter_type = "PROFILE_CURVATURE" # See all available options in Table 1 in Description of input pa
local_surface_type = "QUADRATIC" # Available options: QUADRATIC | BIQUADRATIC
max_neighborhood_distance = 1 # Specify the Neighborhood distance in number of cells, set a value grea
use_adaptive_neighborhood = False # Available options: False | True
```



Input surface raster: Omura_elevation

Output raster: ProfileCurv7

Parameter type: Profile (normal slope line) curvature

Local surface type: Quadratic

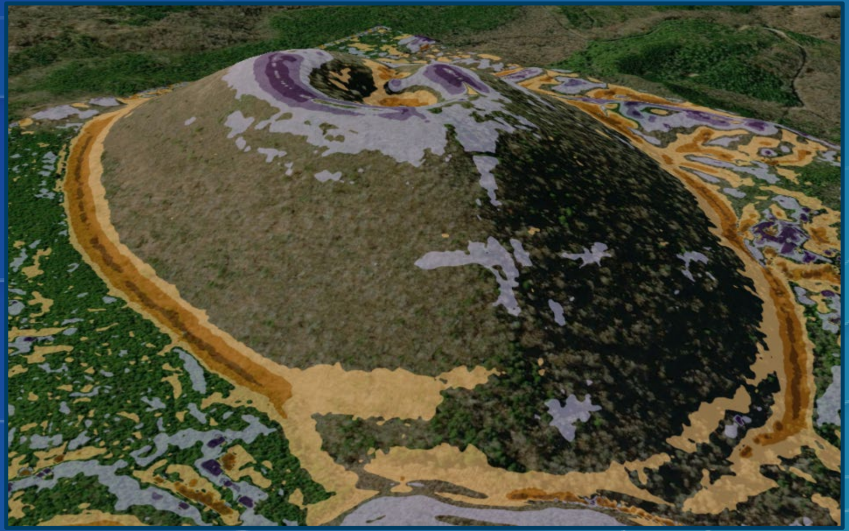
Neighborhood distance: 25 Meters

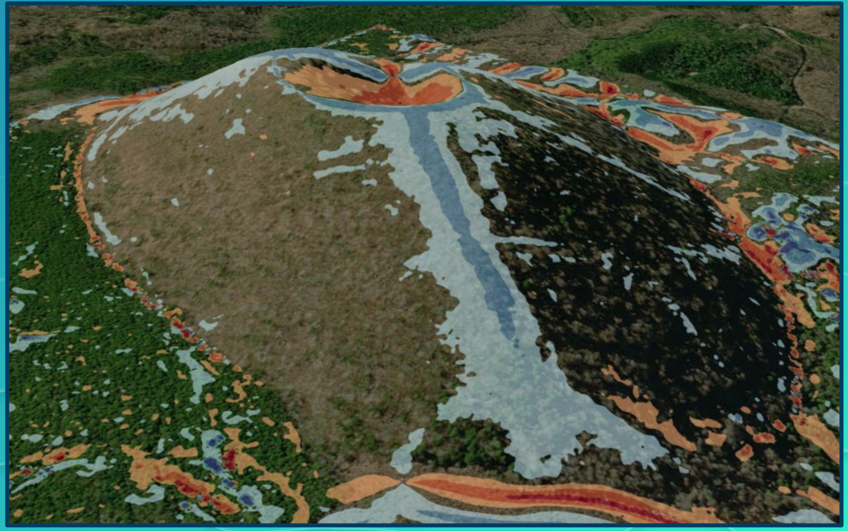
☐ Use adaptive neighborhood

Z unit: Meter

Topics Addressed

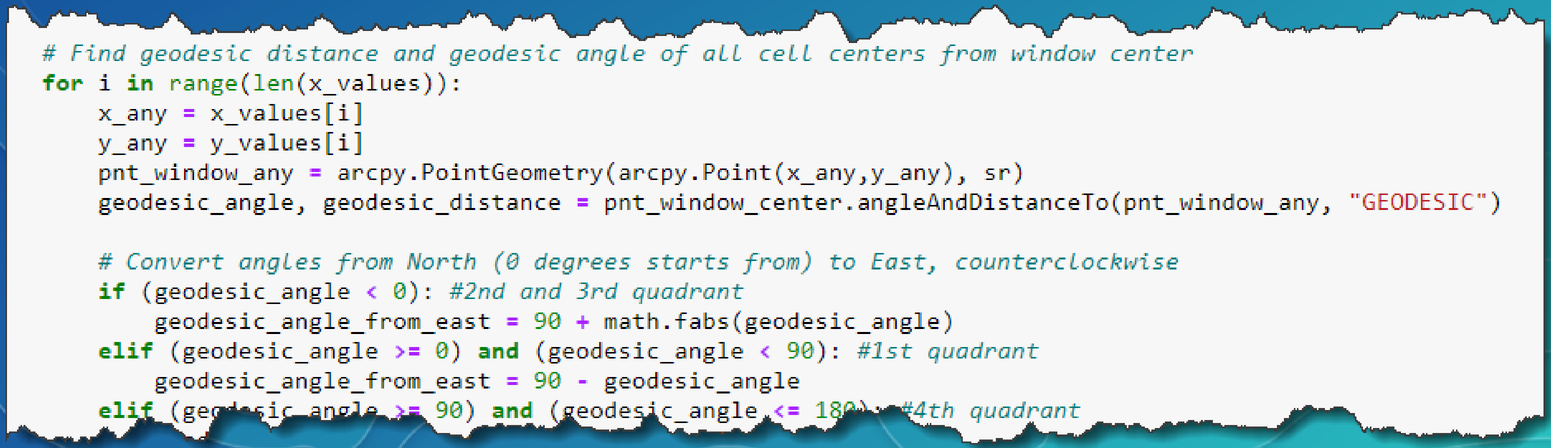
- Remove map projection distortion
- Surface fitting options to improve results from noisy high resolution DEMs
- Neighborhood window size appropriate for DEM resolution
- Support for spatial scaling
- Clarity of curvature names and formulas



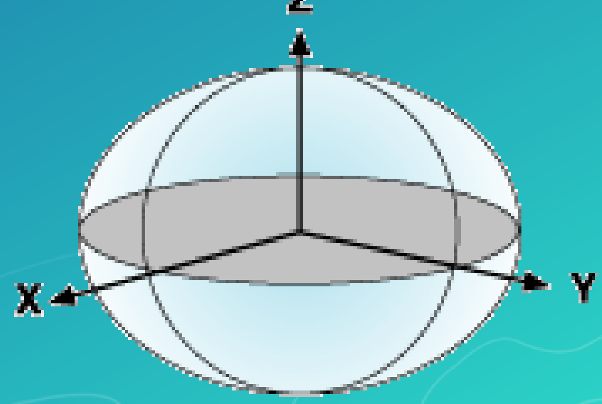


No Map Projection Distortion

- All calculations in the ArcGIS Surface Parameters tool and in the notebook are geodesic.
- Coordinate system of input data does not matter (spherical or planar).
- Eliminates map projection distortion of distances and angles.



```
# Find geodesic distance and geodesic angle of all cell centers from window center
for i in range(len(x_values)):
    x_any = x_values[i]
    y_any = y_values[i]
    pnt_window_any = arcpy.PointGeometry(arcpy.Point(x_any,y_any), sr)
    geodesic_angle, geodesic_distance = pnt_window_center.angleAndDistanceTo(pnt_window_any, "GEODESIC")

    # Convert angles from North (0 degrees starts from) to East, counterclockwise
    if (geodesic_angle < 0): #2nd and 3rd quadrant
        geodesic_angle_from_east = 90 + math.fabs(geodesic_angle)
    elif (geodesic_angle >= 0) and (geodesic_angle < 90): #1st quadrant
        geodesic_angle_from_east = 90 - geodesic_angle
    elif (geodesic_angle >= 90) and (geodesic_angle <= 180): #4th quadrant
```



Surface Fitting options

- **QUADRATIC** (default)

  aka second order polynomial

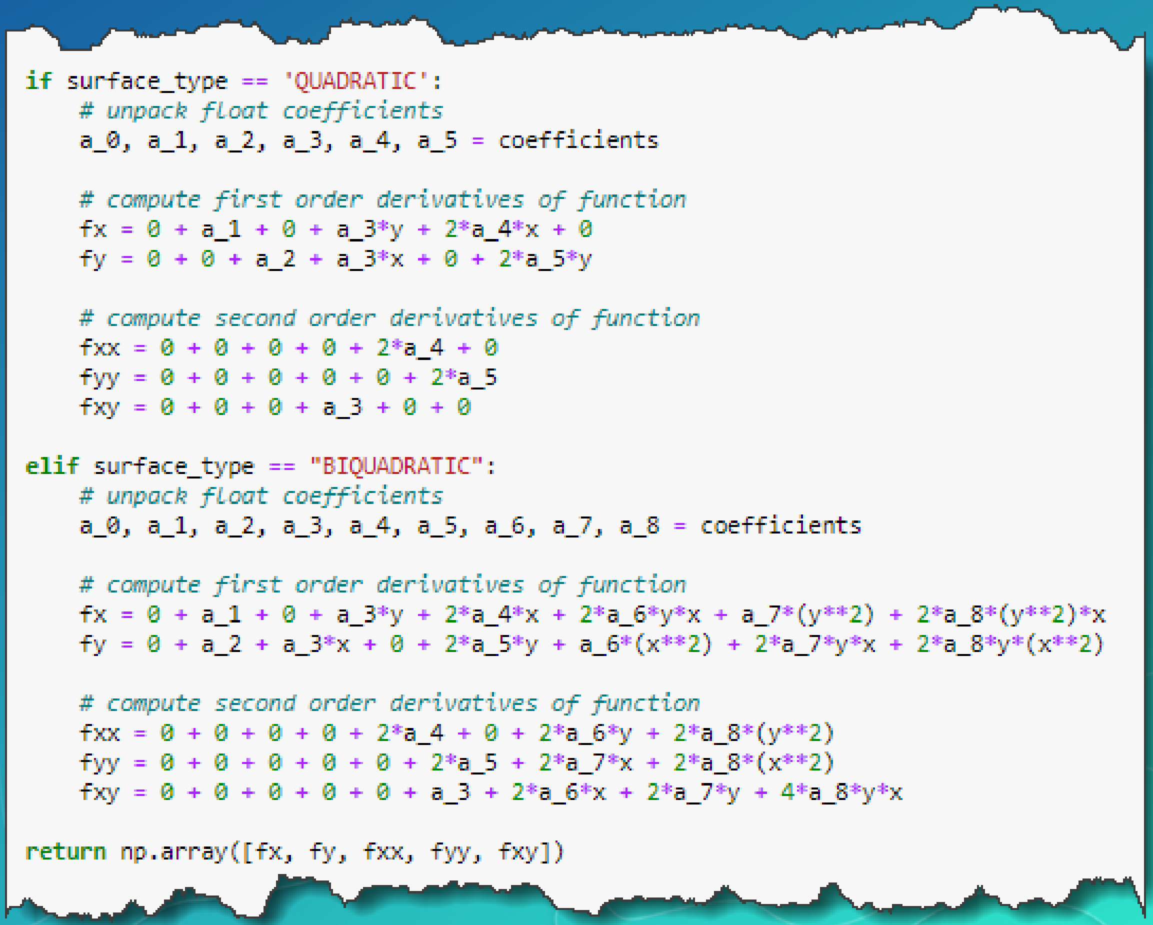
$$z = f(x, y) = a_0 + a_1 x + a_2 y + a_3 xy + a_4 x^2 + a_5 y^2$$

- **BIQUADRATIC**

  aka fourth order polynomial

  aka partial quartic

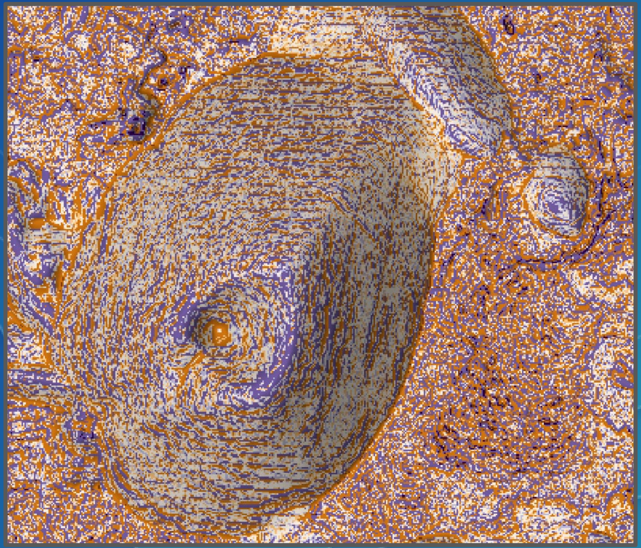
$$z = f(x, y) = a_0 + a_1 x + a_2 y + a_3 xy + a_4 x^2 + a_5 y^2 + a_6 x^2 y + a_7 y^2 + a_8 y^2 x^2$$

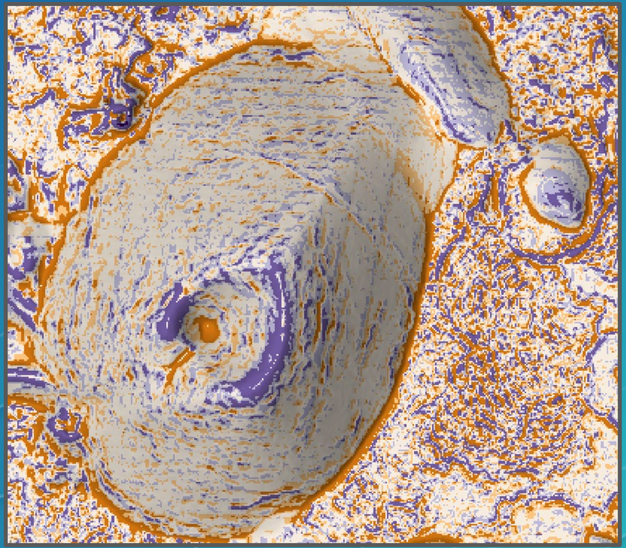


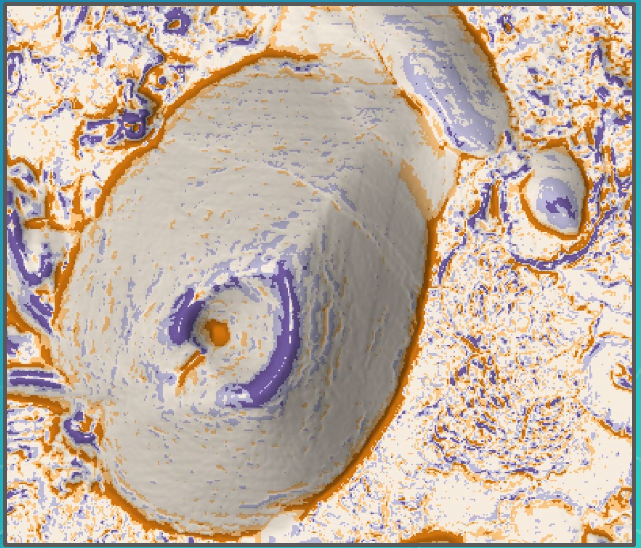
```
if surface_type == 'QUADRATIC':
    # unpack float coefficients
    a_0, a_1, a_2, a_3, a_4, a_5 = coefficients

    # compute first order derivatives of function
    fx = 0 + a_1 + 0 + a_3*y + 2*a_4*x + 0
    fy = 0 + 0 + a_2 + a_3*x + 0 + 2*a_5*y

    # compute second order derivatives of function
    fxx = 0 + 0 + 0 + 0 + 2*a_4 + 0
    fyy = 0 + 0 + 0 + 0 + 0 + 2*a_5
    fxy = 0 + 0 + 0 + a_3 + 0 + 0

elif surface_type == "BIQUADRATIC":
    # unpack float coefficients
    a_0, a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8 = coefficients

    # compute first order derivatives of function
    fx = 0 + a_1 + 0 + a_3*y + 2*a_4*x + 2*a_6*y*x + a_7*(y**2) + 2*a_8*(y**2)*x
    fy = 0 + a_2 + a_3*x + 0 + 2*a_5*y + a_6*(x**2) + 2*a_7*y*x + 2*a_8*y*(x**2)

    # compute second order derivatives of function
    fxx = 0 + 0 + 0 + 0 + 2*a_4 + 0 + 2*a_6*y + 2*a_8*(y**2)
    fyy = 0 + 0 + 0 + 0 + 0 + 2*a_5 + 2*a_7*x + 2*a_8*(x**2)
    fxy = 0 + 0 + 0 + 0 + 0 + a_3 + 2*a_6*x + 2*a_7*y + 4*a_8*y*x

return np.array([fx, fy, fxx, fyy, fxy])
```

User Specified Neighborhood Window

- Specify a window size that matches the cellsize of your elevation and the size of landscape features of interest.
- Run multiple times with different window sizes for multiscale analysis
- The neighborhood window is square, in odd intervals; 3x3, 5x5, etc.


3x3


9x9

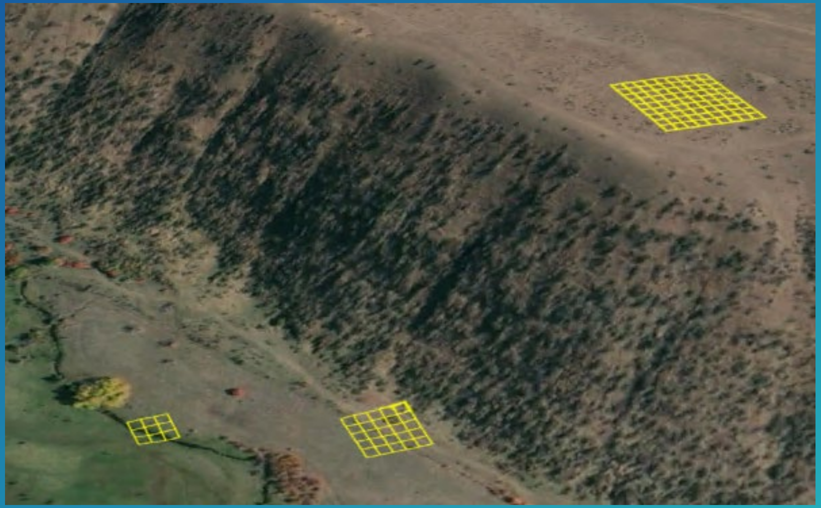

15x15

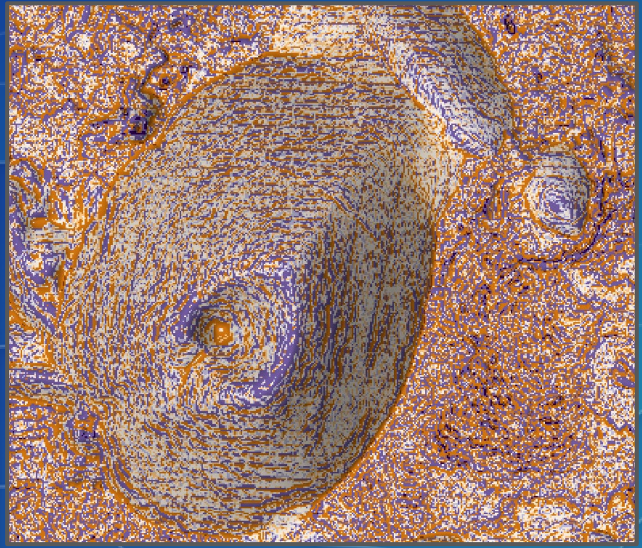
Optional Adaptive Neighborhood Window

Auto adapts from 3x3 to user specific maximum

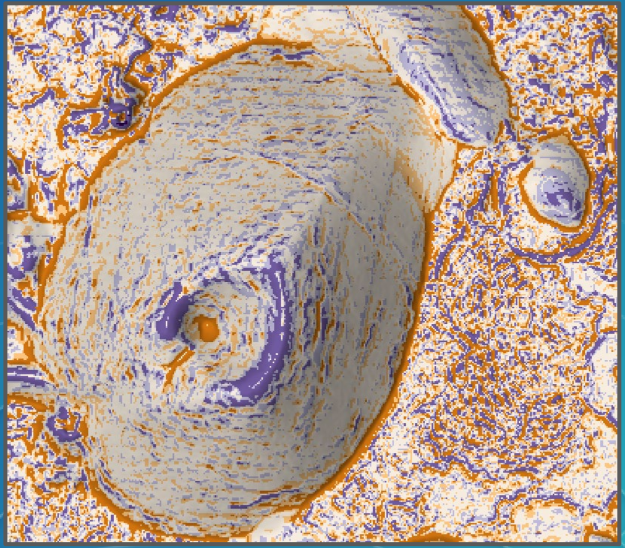
- Evaluates surface complexity for each cell at progressively smaller window sizes until a threshold is met.
- Uses deviation from mean elevation.
- Follows approach of *James et al., 2014*

$$\mathrm{DEV} = \frac{z_0 - \bar{z}}{\mathrm{SD}}$$

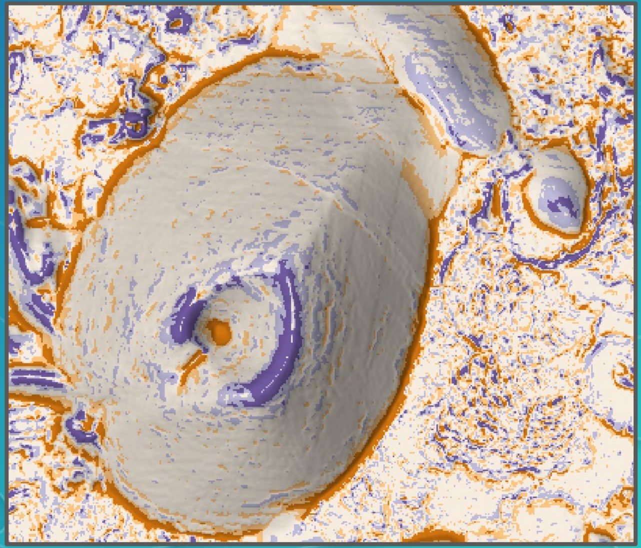




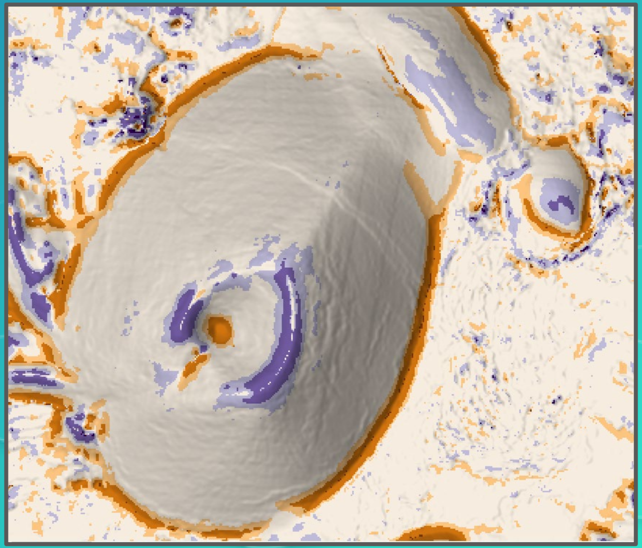
3x3



9x9



15x15



15x15 to 3x3 Adaptive

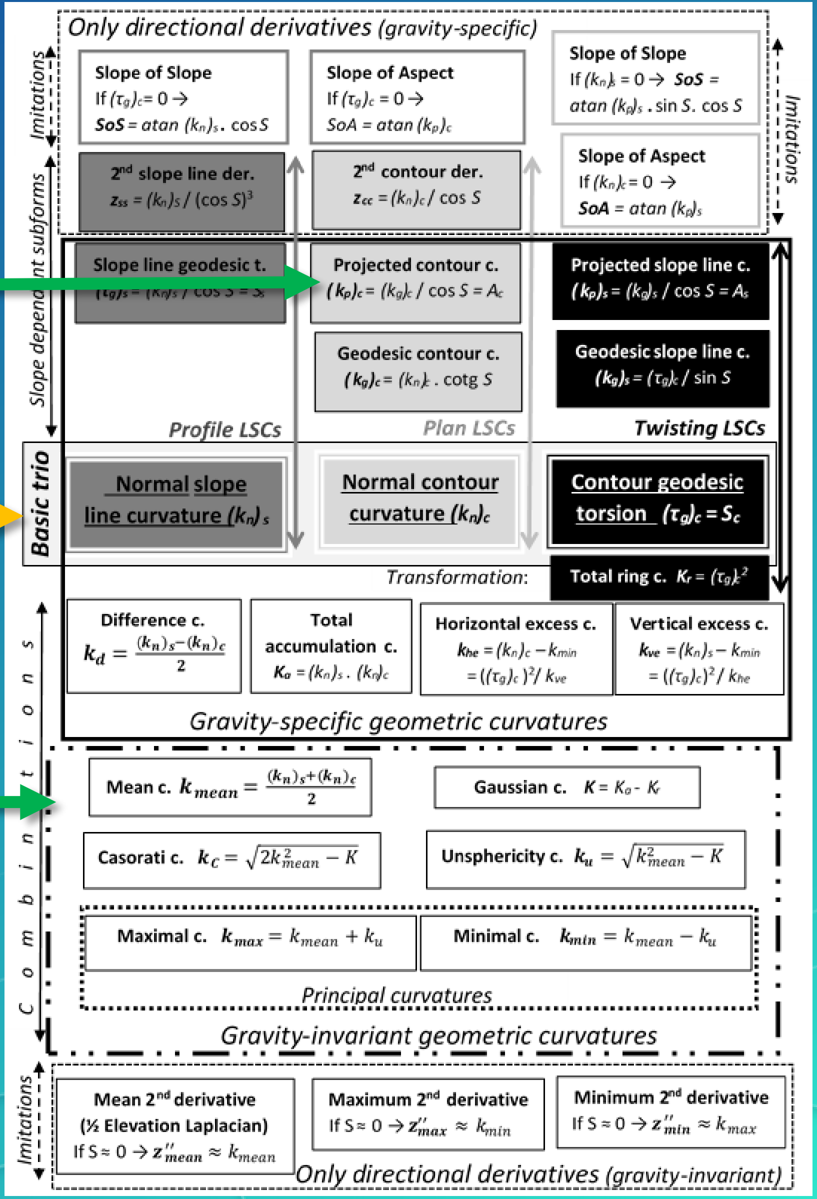
New Geometric Curvatures

- **Following naming system of:**
  Minár, Evans, and Jenčo 2020

Contour (projected contour) curvature

Profile (normal slope line) curvature
Tangential (normal contour) curvature
Contour geodesic torsion curvature

Mean curvature
Gaussian curvature
Casorati curvature



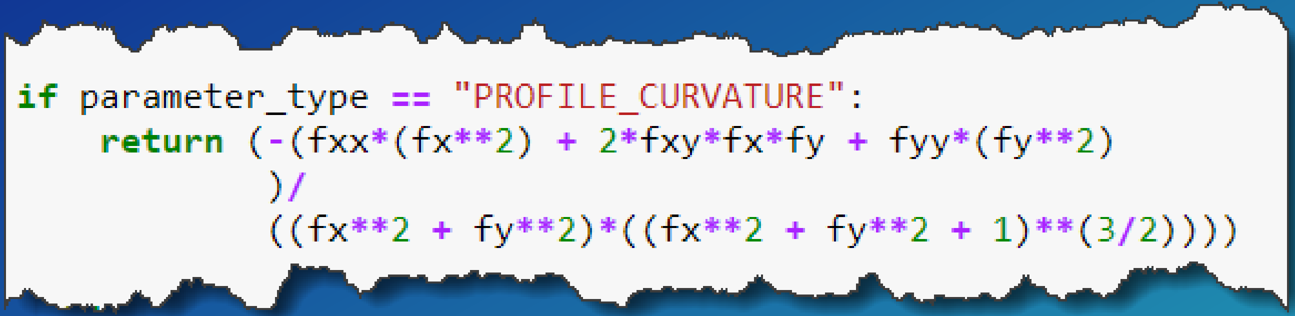
Only directional derivatives (gravity-specific)
Imitations
Slope of Slope If $(\tau_g)_c = 0 \rightarrow$ $SoS = atan\,(k_n)_s \,.\, \cos S$
Slope of Aspect If $(\tau_g)_c = 0 \rightarrow$ $SoA = atan\,(k_p)_c$
Slope of Slope If $(k_n)_s = 0 \rightarrow$ $SoS = atan\,(k_p)_s \,.\, \sin S \,.\, \cos S$
Slope of Aspect If $(k_n)_c = 0 \rightarrow$ $SoA = atan\,(k_p)_s$
2nd slope line der. $z_{ss} = (k_n)_s / (\cos S)^3$
2nd contour der. $z_{cc} = (k_n)_c / \cos S$
Slope dependent subforms
Slope line geodesic t. $(\tau_g)_s = (k_n)_s / \cos S = S_s$
Projected contour c. $(k_p)_c = (k_g)_c / \cos S = A_c$
Projected slope line c. $(k_p)_s = (k_g)_s / \cos S = A_s$
Geodesic contour c. $(k_g)_c = (k_n)_c \,.\, \text{cotg}\, S$
Geodesic slope line c. $(k_g)_s = (\tau_g)_c / \sin S$
Profile LSCs
Plan LSCs
Twisting LSCs
Basic trio
Normal slope line curvature $(k_n)_s$
Normal contour curvature $(k_n)_c$
Contour geodesic torsion $(\tau_g)_c = S_c$
Transformation:
Total ring c. $K_r = (\tau_g)_c^2$
Combinations
Difference c. $k_d = \frac{(k_n)_s - (k_n)_c}{2}$
Total accumulation c. $K_a = (k_n)_s \,.\, (k_n)_c$
Horizontal excess c. $k_{he} = (k_n)_c - k_{min} = ((\tau_g)_c)^2 / k_{ve}$
Vertical excess c. $k_{ve} = (k_n)_s - k_{min} = ((\tau_g)_c)^2 / k_{he}$
Gravity-specific geometric curvatures
Mean c. $k_{mean} = \frac{(k_n)_s + (k_n)_c}{2}$
Gaussian c. $K = K_a - K_r$
Casorati c. $k_C = \sqrt{2k_{mean}^2 - K}$
Unsphericity c. $k_u = \sqrt{k_{mean}^2 - K}$
Maximal c. $k_{max} = k_{mean} + k_u$
Minimal c. $k_{min} = k_{mean} - k_u$
Principal curvatures
Gravity-invariant geometric curvatures
Mean 2nd derivative (½ Elevation Laplacian) If $S \approx 0 \rightarrow z''_{mean} \approx k_{mean}$
Maximum 2nd derivative If $S \approx 0 \rightarrow z''_{max} \approx k_{min}$
Minimum 2nd derivative If $S \approx 0 \rightarrow z''_{min} \approx k_{max}$
Only directional derivatives (gravity-invariant)

from Minár, Evans, and Jenčo 2020

New Geometric Curvatures

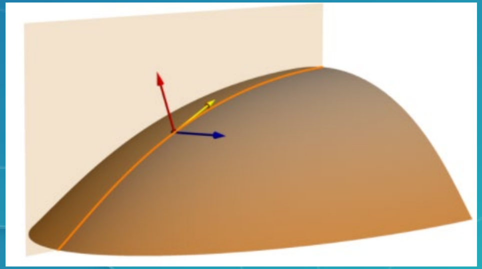
- **Notebook computes "Basic Trio" from derivatives**

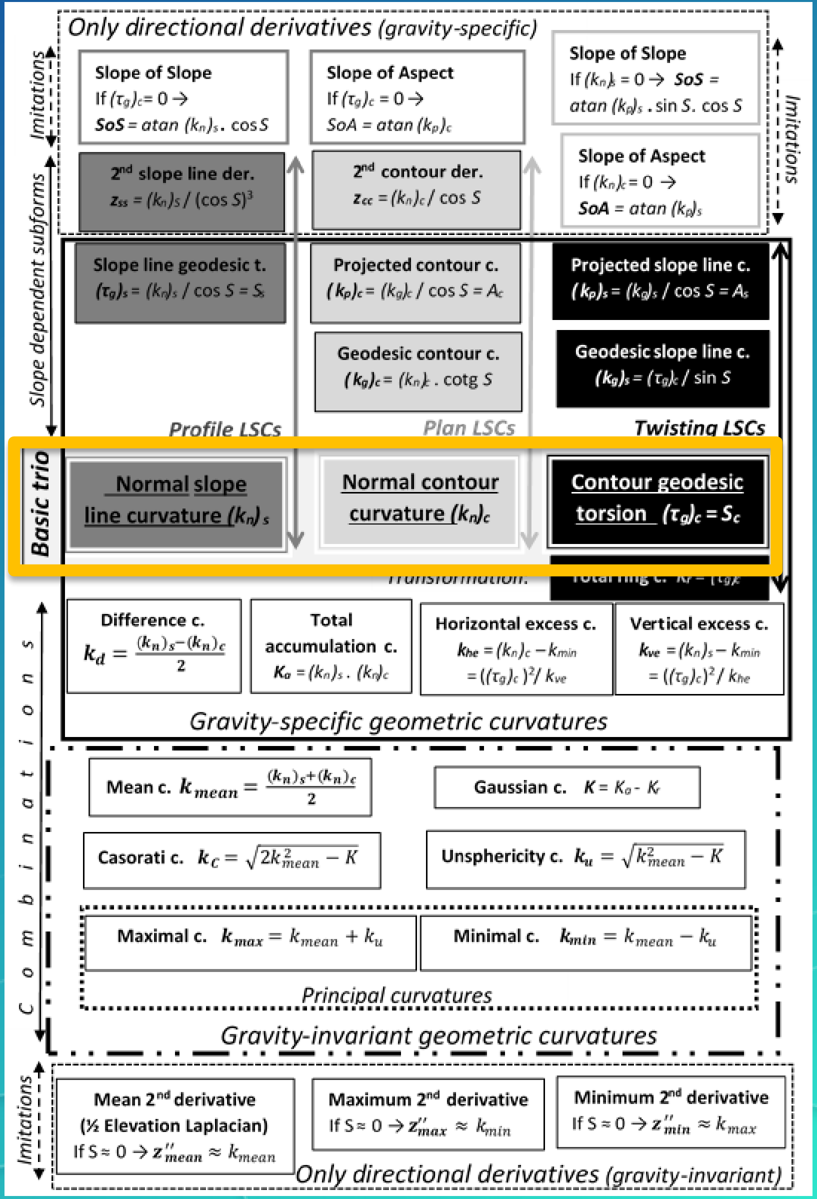
Profile (normal slope line) curvature



```
if parameter_type == "PROFILE_CURVATURE":
    return (-(fxx*(fx**2) + 2*fxy*fx*fy + fyy*(fy**2)
        )/
        ((fx**2 + fy**2)*((fx**2 + fy**2 + 1)**(3/2))))
```

$$K_P = -\frac{f_{xx}f_x^2 + 2f_{xy}f_xf_y + f_{yy}f_y^2}{(f_x^2 + f_y^2)(f_x^2 + f_y^2 + 1)^{3/2}}$$



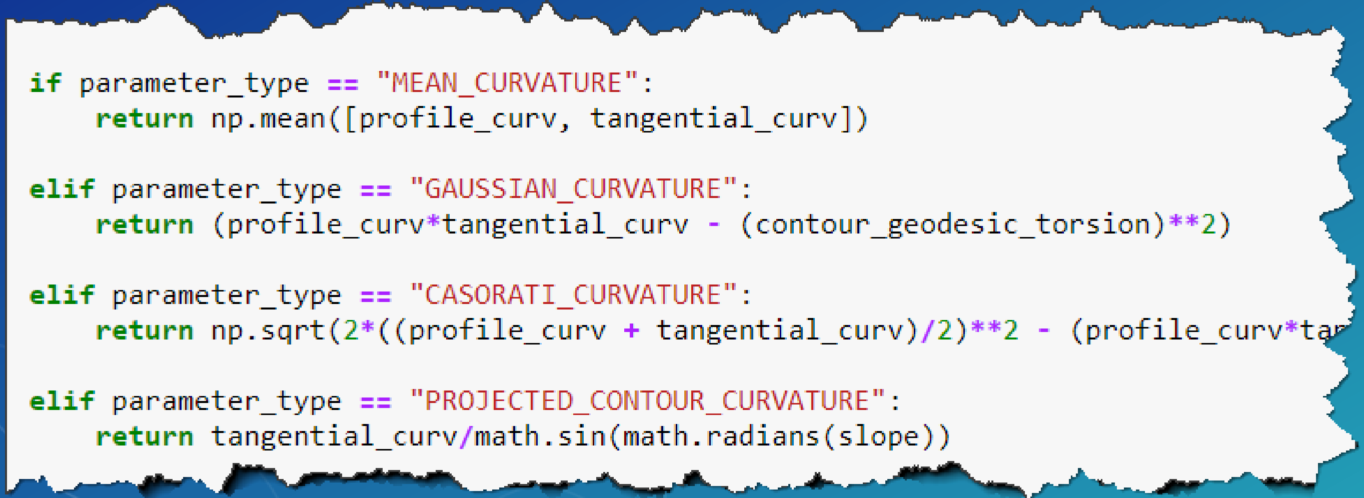


Only directional derivatives (gravity-specific)
Imitations
Slope of Slope If $(\tau_g)_c = 0 \rightarrow$ $SoS = \mathrm{atan}\,(k_n)_s \,.\, \cos S$
Slope of Aspect If $(\tau_g)_c = 0 \rightarrow$ $SoA = \mathrm{atan}\,(k_p)_c$
Slope of Slope If $(k_n)_c = 0 \rightarrow$ $SoS = \mathrm{atan}\,(k_p)_s \,.\, \sin S \,.\, \cos S$
Slope of Aspect If $(k_n)_c = 0 \rightarrow$ $SoA = \mathrm{atan}\,(k_p)_s$
Imitations
Slope dependent subforms
2nd slope line der. $z_{ss} = (k_n)_s / (\cos S)^3$
2nd contour der. $z_{cc} = (k_n)_c / \cos S$
Slope line geodesic t. $(\tau_g)_s = (k_n)_s / \cos S = S_s$
Projected contour c. $(k_p)_c = (k_g)_c / \cos S = A_c$
Projected slope line c. $(k_p)_s = (k_g)_s / \cos S = A_s$
Geodesic contour c. $(k_g)_c = (k_n)_c \,.\, \mathrm{cotg}\, S$
Geodesic slope line c. $(k_g)_s = (\tau_g)_c / \sin S$
Profile LSCs
Plan LSCs
Twisting LSCs
Basic trio
Normal slope line curvature $(k_n)_s$
Normal contour curvature $(k_n)_c$
Contour geodesic torsion $(\tau_g)_c = S_c$
Transformation:
Total ring c. $K_r = (\tau_g)_c^2$
Combinations
Difference c. $k_d = \frac{(k_n)_s - (k_n)_c}{2}$
Total accumulation c. $K_a = (k_n)_s \,.\, (k_n)_c$
Horizontal excess c. $k_{he} = (k_n)_c - k_{min} = ((\tau_g)_c)^2 / k_{ve}$
Vertical excess c. $k_{ve} = (k_n)_s - k_{min} = ((\tau_g)_c)^2 / k_{he}$
Gravity-specific geometric curvatures
Mean c. $k_{mean} = \frac{(k_n)_s + (k_n)_c}{2}$
Gaussian c. $K = K_a - K_r$
Casorati c. $k_C = \sqrt{2k_{mean}^2 - K}$
Unsphericity c. $k_u = \sqrt{k_{mean}^2 - K}$
Maximal c. $k_{max} = k_{mean} + k_u$
Minimal c. $k_{min} = k_{mean} - k_u$
Principal curvatures
Gravity-invariant geometric curvatures
Imitations
Mean 2nd derivative (½ Elevation Laplacian) If $S \approx 0 \rightarrow z''_{mean} \approx k_{mean}$
Maximum 2nd derivative If $S \approx 0 \rightarrow z''_{max} \approx k_{min}$
Minimum 2nd derivative If $S \approx 0 \rightarrow z''_{min} \approx k_{max}$
Only directional derivatives (gravity-invariant)

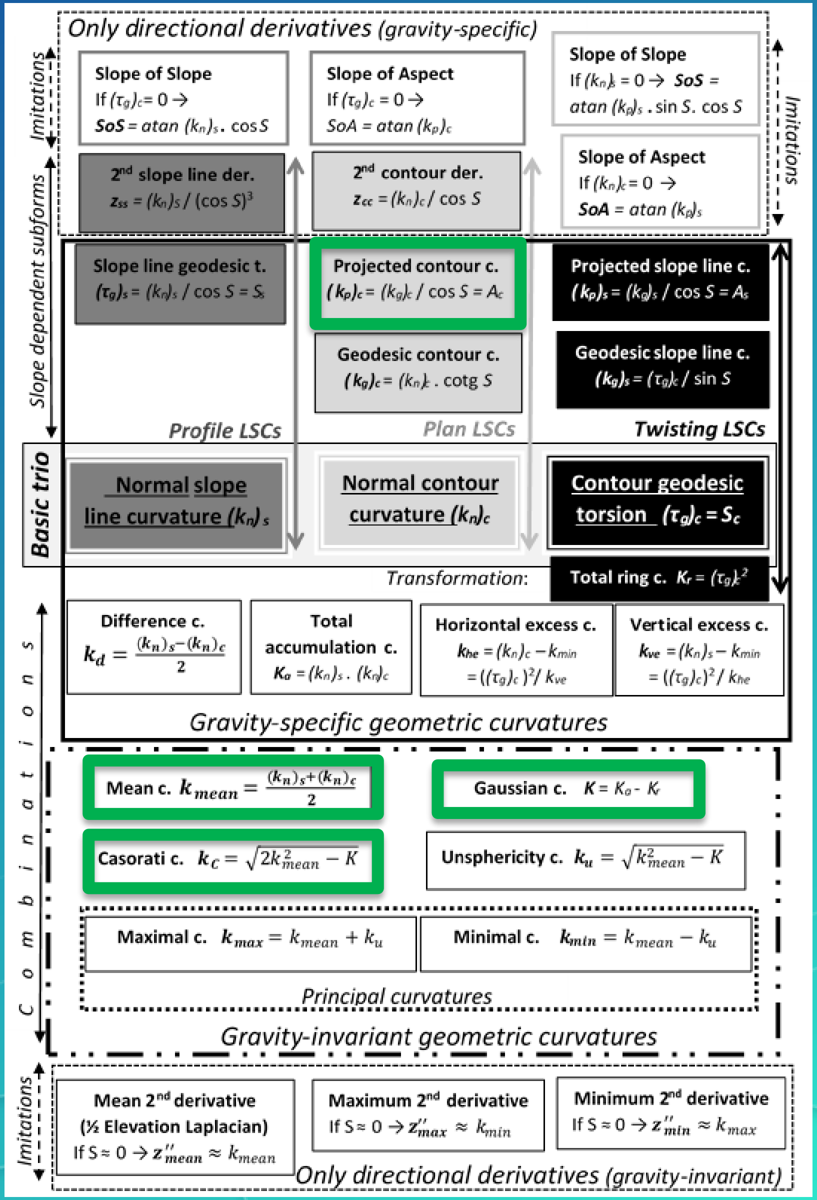
from Minár, Evans, and Jenčo 2020

New Geometric Curvatures

- Additional curvatures computed with simple math using the “Basic trio” and slope



```
if parameter_type == "MEAN_CURVATURE":
    return np.mean([profile_curv, tangential_curv])

elif parameter_type == "GAUSSIAN_CURVATURE":
    return (profile_curv*tangential_curv - (contour_geodesic_torsion)**2)

elif parameter_type == "CASORATI_CURVATURE":
    return np.sqrt(2*((profile_curv + tangential_curv)/2)**2 - (profile_curv*ta

elif parameter_type == "PROJECTED_CONTOUR_CURVATURE":
    return tangential_curv/math.sin(math.radians(slope))
```



Only directional derivatives (gravity-specific)
Imitations
Slope of Slope If $(\tau_g)_c = 0 \rightarrow$ $SoS = atan\ (k_n)_s . \cos S$
Slope of Aspect If $(\tau_g)_c = 0 \rightarrow$ $SoA = atan\ (k_p)_c$
Slope of Slope If $(k_n)_c = 0 \rightarrow$ $SoS = atan\ (k_p)_s . \sin S . \cos S$
Slope of Aspect If $(k_n)_c = 0 \rightarrow$ $SoA = atan\ (k_p)_s$
Slope dependent subforms
2nd slope line der. $z_{ss} = (k_n)_s / (\cos S)^3$
2nd contour der. $z_{cc} = (k_n)_c / \cos S$
Slope line geodesic t. $(\tau_g)_s = (k_n)_s / \cos S = S_s$
Projected contour c. $(k_p)_c = (k_g)_c / \cos S = A_c$
Projected slope line c. $(k_p)_s = (k_g)_s / \cos S = A_s$
Geodesic contour c. $(k_g)_c = (k_n)_c . \mathrm{cotg}\ S$
Geodesic slope line c. $(k_g)_s = (\tau_g)_c / \sin S$
Profile LSCs
Plan LSCs
Twisting LSCs
Basic trio
Normal slope line curvature $(k_n)_s$
Normal contour curvature $(k_n)_c$
Contour geodesic torsion $(\tau_g)_c = S_c$
Transformation:
Total ring c. $K_r = (\tau_g)_c^2$
Combinations
Difference c. $k_d = \frac{(k_n)_s - (k_n)_c}{2}$
Total accumulation c. $K_a = (k_n)_s . (k_n)_c$
Horizontal excess c. $k_{he} = (k_n)_c - k_{min} = ((\tau_g)_c)^2 / k_{ve}$
Vertical excess c. $k_{ve} = (k_n)_s - k_{min} = ((\tau_g)_c)^2 / k_{he}$
Gravity-specific geometric curvatures
Mean c. $k_{mean} = \frac{(k_n)_s + (k_n)_c}{2}$
Gaussian c. $K = K_a - K_r$
Casorati c. $k_C = \sqrt{2k_{mean}^2 - K}$
Unsphericity c. $k_u = \sqrt{k_{mean}^2 - K}$
Maximal c. $k_{max} = k_{mean} + k_u$
Minimal c. $k_{min} = k_{mean} - k_u$
Principal curvatures
Gravity-invariant geometric curvatures
Imitations
Mean 2nd derivative (½ Elevation Laplacian) If $S \approx 0 \rightarrow z''_{mean} \approx k_{mean}$
Maximum 2nd derivative If $S \approx 0 \rightarrow z''_{max} \approx k_{min}$
Minimum 2nd derivative If $S \approx 0 \rightarrow z''_{min} \approx k_{max}$
Only directional derivatives (gravity-invariant)

from Minár, Evans, and Jenčo 2020

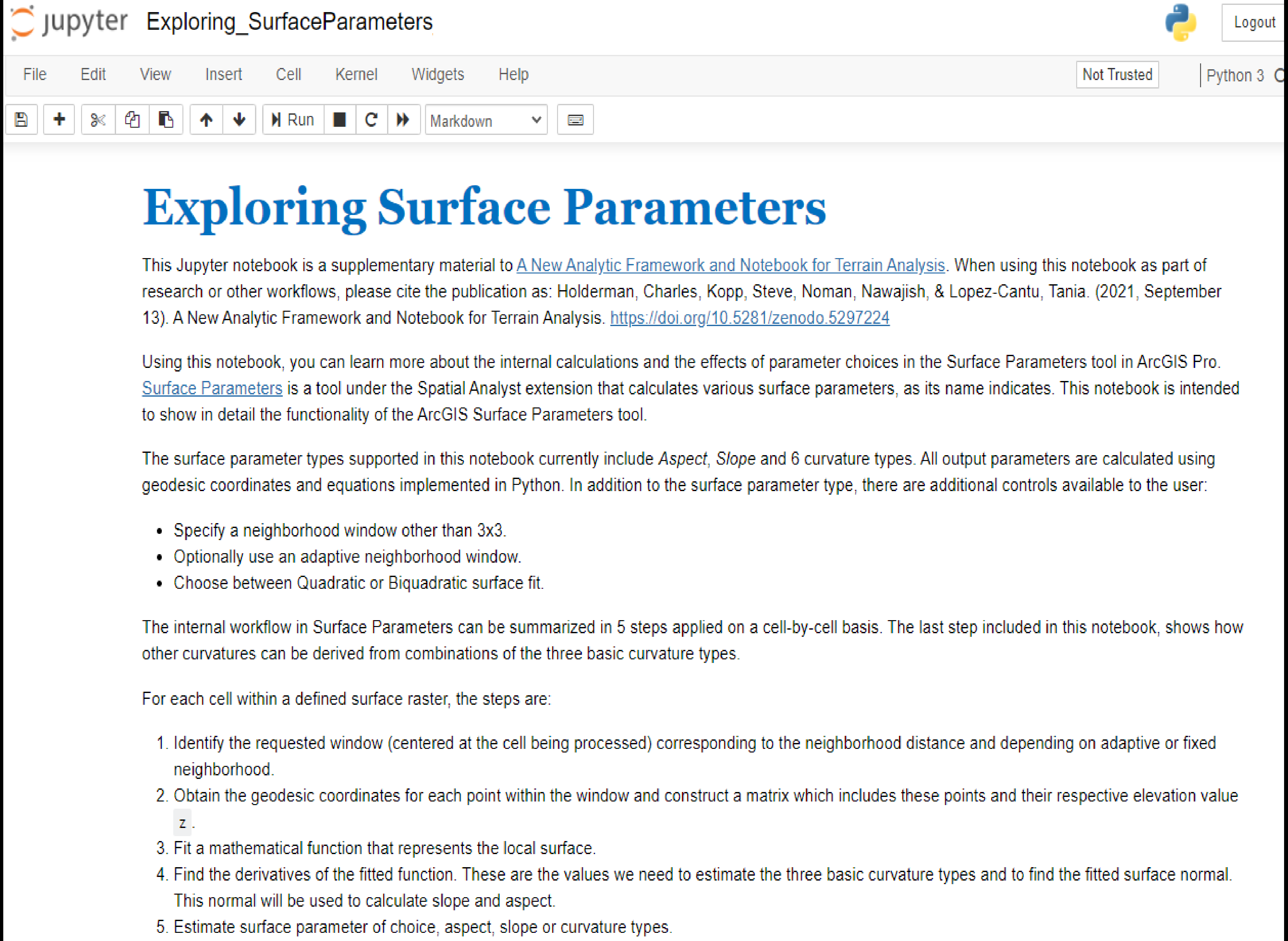
Known Differences between ArcGIS tool and Notebook results

- 10^{-7} differences in Curvature output from math libraries of C++ vs Python
- Differences in handling of missing values at edges of input data.
- Differences in geodesic calculation approach

  10^{-4} differences in Slope output

  10^{-1} differences in Aspect output

- Performance:
  - ArcGIS Surface Parameters tool is optimized C++, parallel.
  - Notebook is written to easily follow the logic and math, it is verbose and intentionally contains no optimizations or tuning tricks. Use the notebook to understand and experiment.
  - For project work we recommend using the ArcGIS tool


jupyter Exploring_SurfaceParameters
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Markdown

Exploring Surface Parameters

This Jupyter notebook is a supplementary material to A New Analytic Framework and Notebook for Terrain Analysis. When using this notebook as part of research or other workflows, please cite the publication as: Holderman, Charles, Kopp, Steve, Noman, Nawajish, & Lopez-Cantu, Tania. (2021, September 13). A New Analytic Framework and Notebook for Terrain Analysis. https://doi.org/10.5281/zenodo.5297224

Using this notebook, you can learn more about the internal calculations and the effects of parameter choices in the Surface Parameters tool in ArcGIS Pro. Surface Parameters is a tool under the Spatial Analyst extension that calculates various surface parameters, as its name indicates. This notebook is intended to show in detail the functionality of the ArcGIS Surface Parameters tool.

The surface parameter types supported in this notebook currently include *Aspect*, *Slope* and 6 curvature types. All output parameters are calculated using geodesic coordinates and equations implemented in Python. In addition to the surface parameter type, there are additional controls available to the user:

- Specify a neighborhood window other than 3x3.
- Optionally use an adaptive neighborhood window.
- Choose between Quadratic or Biquadratic surface fit.

The internal workflow in Surface Parameters can be summarized in 5 steps applied on a cell-by-cell basis. The last step included in this notebook, shows how other curvatures can be derived from combinations of the three basic curvature types.

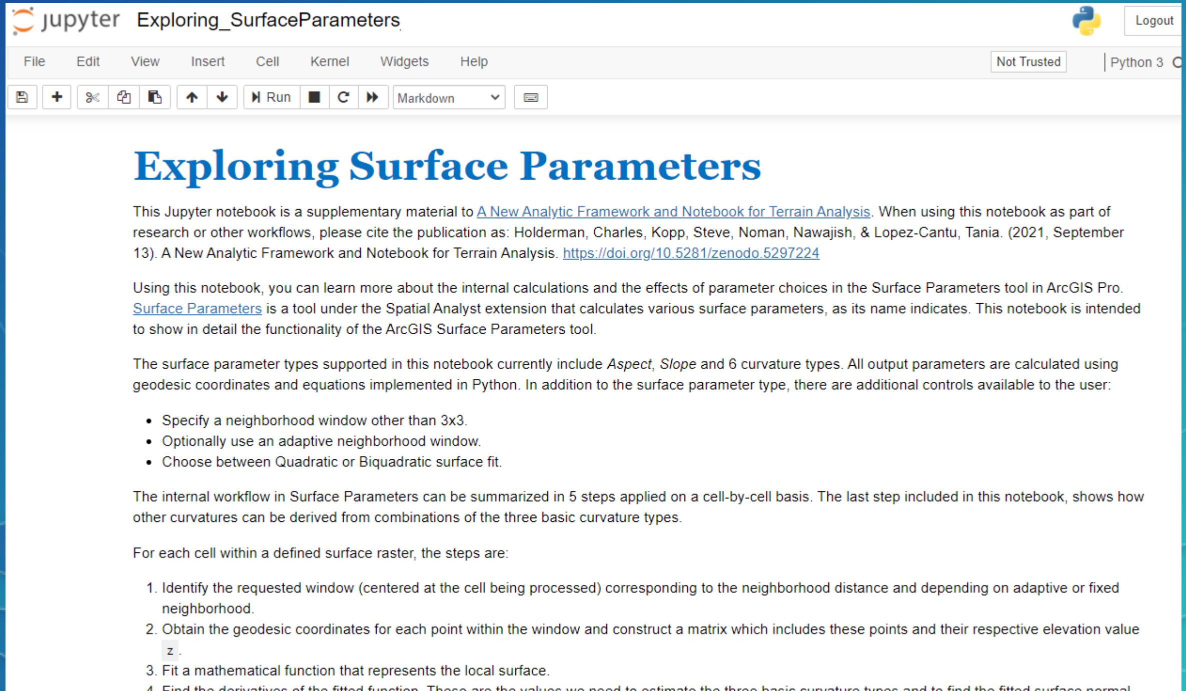
For each cell within a defined surface raster, the steps are:

1. Identify the requested window (centered at the cell being processed) corresponding to the neighborhood distance and depending on adaptive or fixed neighborhood.
2. Obtain the geodesic coordinates for each point within the window and construct a matrix which includes these points and their respective elevation value `z`.
3. Fit a mathematical function that represents the local surface.
4. Find the derivatives of the fitted function. These are the values we need to estimate the three basic curvature types and to find the fitted surface normal. This normal will be used to calculate slope and aspect.
5. Estimate surface parameter of choice, aspect, slope or curvature types.

Thank you

https://www.esriurl.com/SurfaceParameters (download)

https://www.esriurl.com/SurfaceParametersPreview (preview)


jupyter Exploring_SurfaceParameters
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Markdown

Exploring Surface Parameters

This Jupyter notebook is a supplementary material to A New Analytic Framework and Notebook for Terrain Analysis. When using this notebook as part of research or other workflows, please cite the publication as: Holderman, Charles, Kopp, Steve, Noman, Nawajish, & Lopez-Cantu, Tania. (2021, September 13). A New Analytic Framework and Notebook for Terrain Analysis. https://doi.org/10.5281/zenodo.5297224

Using this notebook, you can learn more about the internal calculations and the effects of parameter choices in the Surface Parameters tool in ArcGIS Pro. Surface Parameters is a tool under the Spatial Analyst extension that calculates various surface parameters, as its name indicates. This notebook is intended to show in detail the functionality of the ArcGIS Surface Parameters tool.

The surface parameter types supported in this notebook currently include *Aspect*, *Slope* and 6 curvature types. All output parameters are calculated using geodesic coordinates and equations implemented in Python. In addition to the surface parameter type, there are additional controls available to the user:

- Specify a neighborhood window other than 3x3.
- Optionally use an adaptive neighborhood window.
- Choose between Quadratic or Biquadratic surface fit.

The internal workflow in Surface Parameters can be summarized in 5 steps applied on a cell-by-cell basis. The last step included in this notebook, shows how other curvatures can be derived from combinations of the three basic curvature types.

For each cell within a defined surface raster, the steps are:

1. Identify the requested window (centered at the cell being processed) corresponding to the neighborhood distance and depending on adaptive or fixed neighborhood.
2. Obtain the geodesic coordinates for each point within the window and construct a matrix which includes these points and their respective elevation value `z`.
3. Fit a mathematical function that represents the local surface.
4. Find the derivatives of the fitted function. These are the values we need to estimate the three basic curvature types and to find the fitted surface normal