Prediction error interpolation in bathymetry

Stewart A. Levin & Christopher M. Castillo

ABSTRACT

We have implemented missing data interpolation for bathymetry under an ArcGIS™ Python framework. We report on both stationary and nonstationary prediction error filter applications to synthetic and field data and the method compares favorably to several widely available interpolation methods.

INTRODUCTION

Claerbout and Fomel (2014, chap. 7.6–9) show how to apply prediction error filter (PEF) theory to interpolation of missing data in seismic and topographic datasets. After optional preprocessing and regridding, a PEF is designed on the existing data and the missing data are determined by calculating what values produce the least prediction error when the PEF is applied to the whole grid.

To assist in coauthor Castillo's thesis work, we reimplemented the PEF interpolation under the ArcGIS framework using the ArcPy tool API. Subsequently, Claerbout and Wang (2018) introduced a spatially-variable version of PEF interpolation which we also implemented in ArcGIS.

TOOL DESIGN

ArcGIS Python provides has a well-defined template for building a Python toolbox that leverages an extensive library of GIS functionality (Zandbergen, 2013) interfaced to the NumPy package (SciPy.org, 2018). A toolbox contains one or more individual tools which retrieve and validate parameters, execute code, and inform the user of progress status. In addition, tools may be combined into scripts and macros.

Our PEF-Tools Arc Toolbox provides both a missing data interpolation "PEF-S" using a stationary PEF as described in Claerbout and Fomel (2014) and one, "PEF-N", using the nonstationary PEF from Claerbout and Wang (2018). These are supplemented and/or wrapped with:

Auto-PEF Automatically identifies holes in the input data, creates areas of interest (AOI) around them based on a user-specified edge width, and runs PEF-S separately for each. Useful for large datasets with many small holes.

Define AOI Creates polygons around data gaps in the input data to be used as the area of interest for PEF-S infill.

Detrend Subtracts box-car filtered data from the input to emphasize the local texture around data gaps.

Benchmark Compares available interpolation methods in ArcGIS Spatial Analyst for in input tiff image.

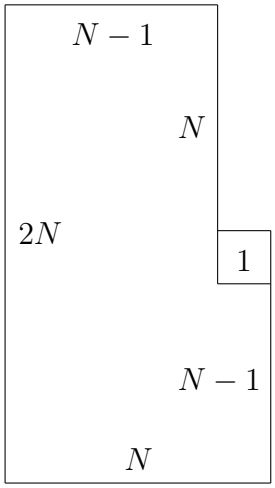
No-Edge Slope Outputs a slope map and digital elevation map (DEM) with no visible edges.

No-Edge Shade Outputs a hillshade and DEM with no visible edges.

SESRM Outputs a slope-enhanced shaded hillside and DEM with no visible edges.

IMPLEMENTATION CHOICES AND ISSUES

For coding and computational efficiency, we limited our PEFs to have the shape



where the sample count N is limited to be at most 5 (45 coefficients) and the coefficient at the prediction location is unity. N is determined by counting the number of places it can overlay known data in the current AOI and ensuring that number is at least 9 times as large as the number of coeffients. We note that the orientation of the PEF, e.g. 90° rotation or 180° reflection, factors into this calculation.[1]

Having chosen the PEF shape, the next step for both the stationary and nonstationary cases is to compute the PEF that minimized the L_2 prediction error for the

[1]Claerbout and Fomel (2014,chap. 7.10.3) suggests that orientation of the PEF should not make a difference in the interpolation output. We must note that, unlike them, we are not solving for the filter simultaneously with the missing data.

known data. This serves as the known PEF for the stationary case and as the initial PEF estimate for the nonstationary case.

PEF-S

PEF-S designs its single PEF on the live data in the AOI using the LSMR iterative solver (Fong and Saunders, 2011) in SciPy. The missing data are obtained using the SciPy *spsolve* direct sparse solver. We have found that the filter orientation can affect the result when the input is nonstationary or otherwise too complex to be fully represented with our largest choice of filter. Detrending the surrounding data has helped to minimize this effect when data gaps were relatively small.

PEF-N

For this algorithm, there are two adaptation parameters as well as 8 choices of filter orientation and the ability to make multiple passes over the grid to improve the missing data estimates. We chose to use all 8 orientations sequentially in each outer iteration. One adaptation parameter, ϵ_A, controls how quickly the PEF, A, is allowed to change from sample to sample as it shifts along the grid; the other, ϵ_Y, controls how much to update the missing data, Y, estimate with the changing PEF. Of the two, ϵ_Y proved to be the most critical. Indeed, as Fig. 1 illustrates, the interpolated values can diverge for what seems a reasonable choice of unitless ϵ_Y.

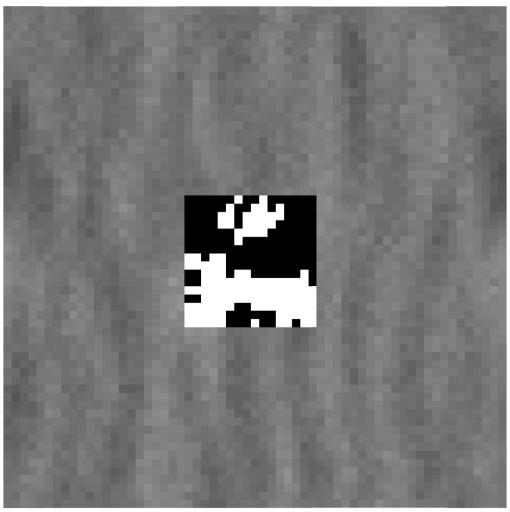


Figure 1: Infill of hole surrounded by random Gaussian numbers with $\epsilon_Y = \frac{1}{2}$ [**CR**]

After experimenting with a range of smaller values for ϵ_Y, we realized that the issue was tied to the PEF and not ϵ_Y. While a PEF reduces the norm of its output

with respect to the input on which it was designed, the fact that the PEF begins with a leading coefficient of 1 says that, as an operator, it always has a norm greater than 1 unless it is the trivial identity filter. There is no a priori reason that its action on trial values for missing data should not reflect that potential magnification. So, to account for this, we rescaled our ϵ_Y by dividing it by the norm of the initial PEF filter we designed on the live data in the AOI. This, too, managed to blow up! We finally realized that because the filter, A, was adapting as it entered the gaps, and adapting differently from iteration to iteration, we had to recalculate the filter norm at every step in order to ensure that our missing data updates were under control and stayed usefully incremental.

The situation for ϵ_A was less challenging because the guideline in Claerbout and Wang (2018,chap. 1.3) to use the reciprocal of the variance of the data, Y, to rescale it to a unitless value worked reasonably well. The value we settled on, $\epsilon_A = 0.05/\sum d_i^2$, where d_i are the samples under the current filter, was a compromise between adapting painfully slowly and allowing the last orientation in an iteration to introduce a visible directional bias in the infill.

In addition to setting ϵ_Y and ϵ_A, we also could vary the number of interpolation passes over the grid. We settled on 64 eight-way passes, admittedly overkill, but sufficient to converge the infill of all but the largest gaps.

EXAMPLES

Prediction error filters are designed to track individual sinusoids and combinations of them, limited by just the number of PEF coefficients. Figures 2 and 3 are evidence that our code is working. In both of these examples we specifically avoided cases where either the hole or the sinusoids serendipitously aligned with coordinate axes. In addition, we made sure that the criss-crossing sinusoids had distinctly different wavenumbers and slopes.

While Fourier theory says that even the most complex seafloor grid of bathymetry can be perfectly reconstructed with a finite number of sinusoids, there is no reason to expect that a global Fourier reconstruction will infill data gaps in a useful way when, as is almost always the case, the statistics and pattern of the bathymetry change locally. This is illustrated in the challenging dataset of Figure 4. Our interpolation is highly realistic, exhibiting the characteristics of the data that our eye expects to see in the hole.

Finally, Figure 5 shows a large scale application of PEF interpolation to infill typical gaps in seafloor coverage that arise as the echo sounding vessel shifts and meanders as it moves along its nominal path.

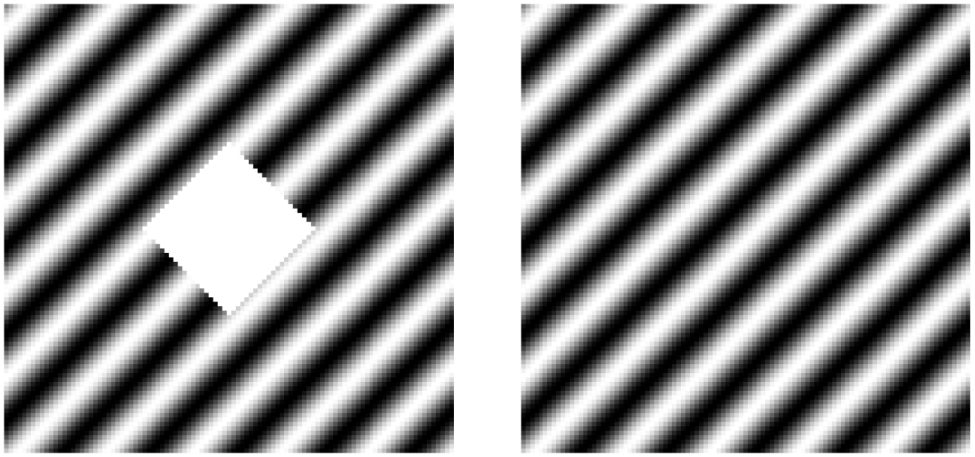


Figure 2: Test of PEF-S to ensure it reproduces input spatial sinusoid in a hole that is not aligned with x and y axes. [**CR**]

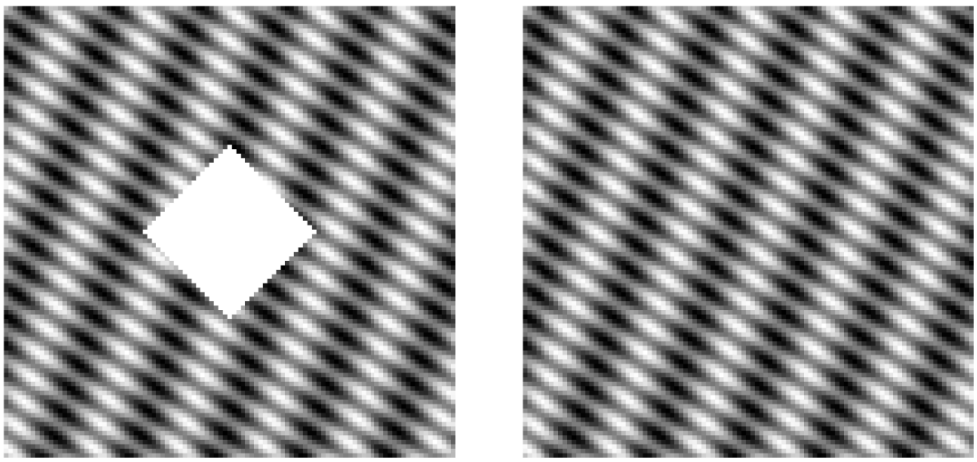


Figure 3: Test of PEF-S to ensure it reproduces two input spatial sinusoids with incommensurate orientations and wavenumbers. [**CR**]

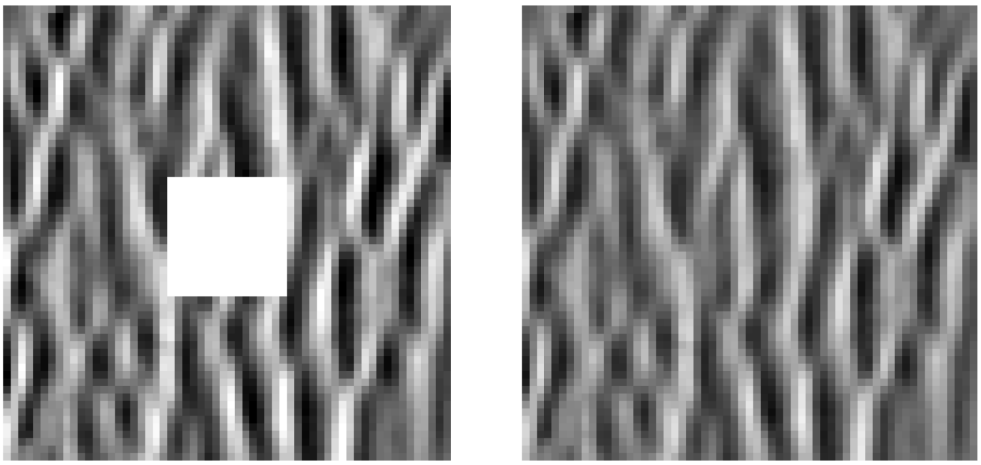


Figure 4: Portion of seafloor texture after detrending and removing a 16x16 square in the center alongside a PEF-N interpolation. [**CR**]

SUMMARY

We have implemented both stationary and nonstationary PEF missing data interpolation within the ArcGIS system. We developed reasonable choices for the adjustable parameters in the nonstationary setting using a combination of theoretical reasoning and empirical trials. While further optimization, including GPU coding, can help reduce the time it takes to iterate to an answer, we would challenge anyone to find the location of the originally missing data in our results.

REFERENCES

Claerbout, J. and K. Wang, 2018, Data Fitting with Nonstationary Statistics: Lulu Press, www.lulu.com.

Claerbout, J. F. and S. Fomel, 2014, Geophysical Image Estimation by Example: Stanford Exploration Project.

Fong, D. C.-L. and M. A. Saunders, 2011, LSMR: An iterative algorithm for sparse least-squares problems: SIAM J. Sci. Comput., **33**, 2950–2971.

SciPy.org, 2018, NumPy v1.14 Manual.

Zandbergen, P. A., 2013, Python scripting for ArcGIS: Esri Press, Redlands, California.

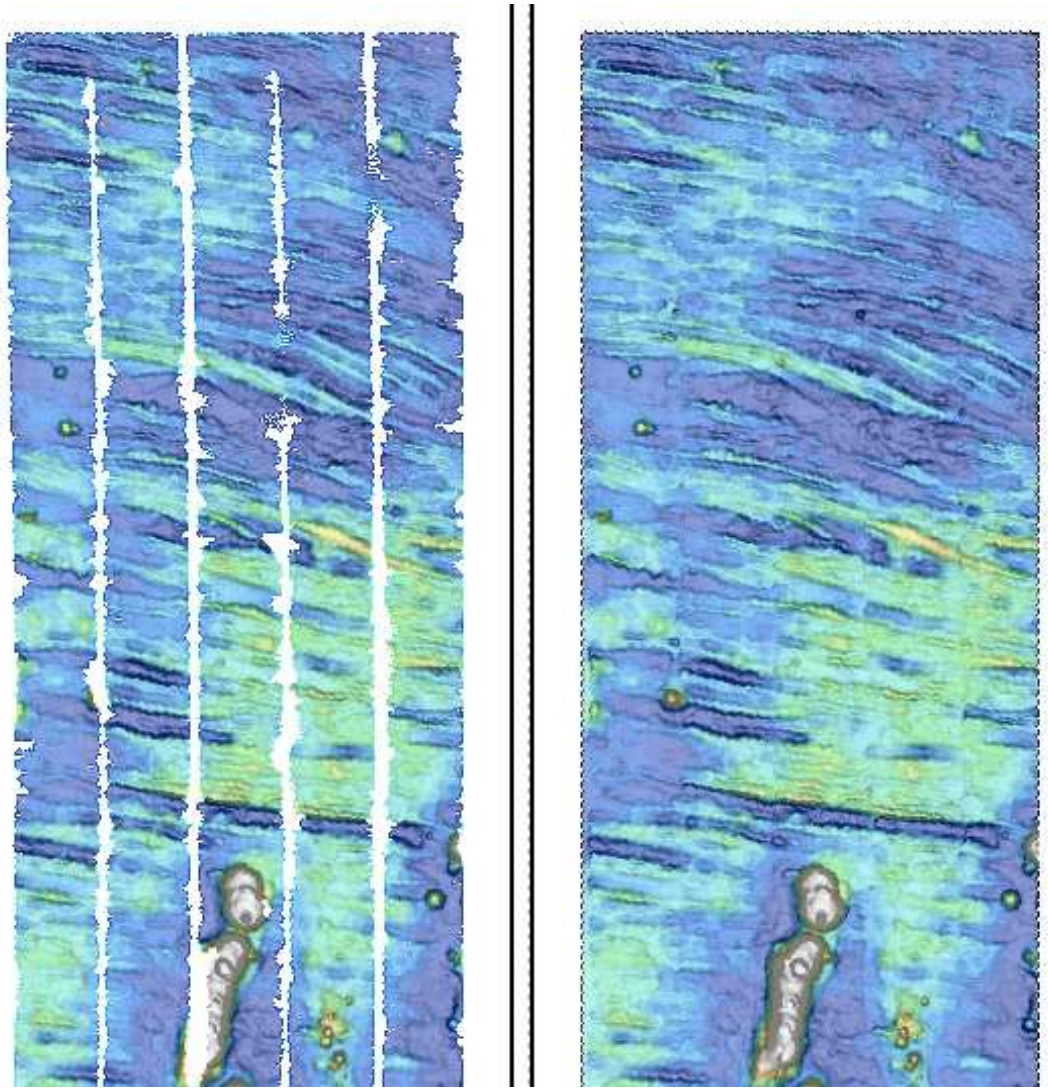


Figure 5: PEF-S bathymetry interpolation. [**CR**]