Folding Functions II: Methods for Mathematically Manipulating Miura-ori Models

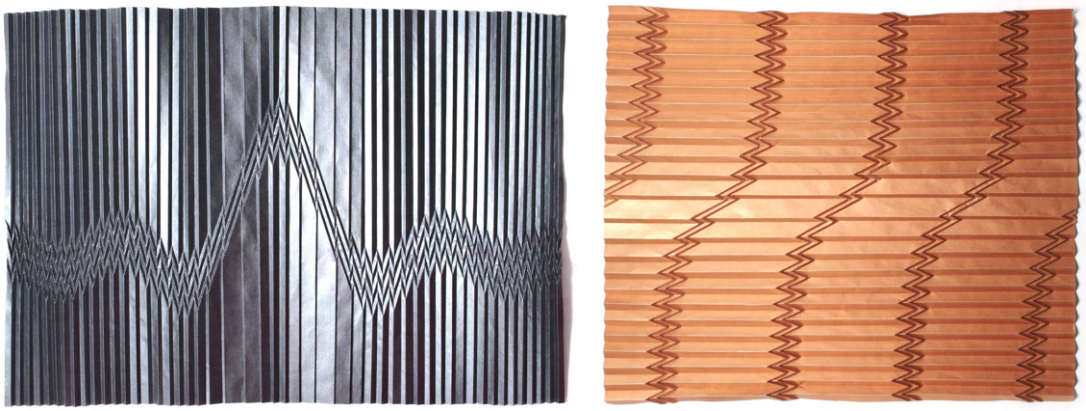
Uyen Nguyen

New York City, New York, USA; win@winwin.fashion

Abstract

This paper demonstrates a method for designing origami representations of various functions by using the Miura-ori fold and adjusting the spacing between creases. Limitations and design suggestions are discussed.

Introduction



(a) $y = \text{sinc}(x)$ **(b)** $y = \tan(x)$

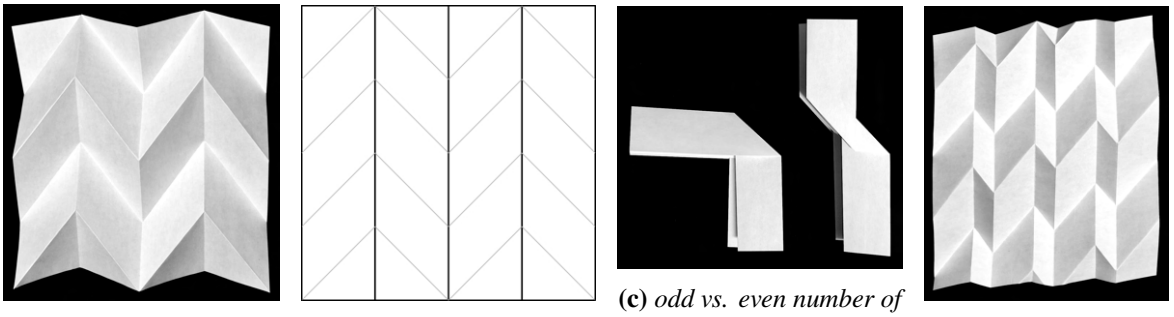
Figure 1: *Example foldings.*

One of my major areas of exploration is designing flat-foldable origami that can approximate equations. In 2019 I coauthored a paper detailing an algorithm that does just that [5]. However, that method had its limits: all lines normal to the curve had to intersect along axes of mirror symmetry. Hence, while it was possible to design a sine wave, which has an axis of reflective symmetry at all local extrema, it could not produce a foldable pattern for the sinc function, which only has the y-axis as a line of symmetry, and the remaining extrema are asymmetric. Similarly, crease patterns for certain asymptotic functions such as the tangent function weren't possible as they lack any axes of reflective symmetry. To overcome these limitations, I developed a new algorithm that doesn't have constraints for symmetry. By using the Miura-ori fold, but varying the spacing between the creases, it creates the appearance of folds that follow a given function. Figure 1 shows a folded sinc function and tangent function, respectively, using this method.

The Miura-Ori Fold

The Miura-ori fold, sometimes called chevron pleating, is a common origami corrugation. One of its earliest appearances was in a preliminary course in paper study by Josef Albers at the Bauhaus in 1927-1928 [1]. However, the design is widely attributed to Koryo Miura, a Japanese astrophysicist who developed the pattern as a means to package and deploy large membranes in space [3]. Figure 2a shows a basic Miura-ori fold. The

crease pattern (Figure 2b) consists of a series of parallel lines (in this case vertical, in black) that are traversed by a series of zigzags (in gray). Each zigzag can be seen as a diagonal coming from the edge of the paper that gets reflected each time it crosses a vertical line. The number of zigzags and their angle can be varied. I prefer an even number of zigzags to allow the bulk of the paper to rest on parallel planes (Figure 2c). Many origami designers have come up with variations to the Miura-ori fold, where the spacing between the parallel lines is modified. In particular, Ben Parker's works [6] inspired me to develop this algorithm. For example, Figure 2d shows a variation where the spacing between the vertical lines alternates small and large widths. The result makes the zigzags travel upwards as you move to the right, instead of heading horizontally across the page. Seeing this behavior led me to develop a way to determine the spacing between the vertical lines such that the zigzags would follow a curve.

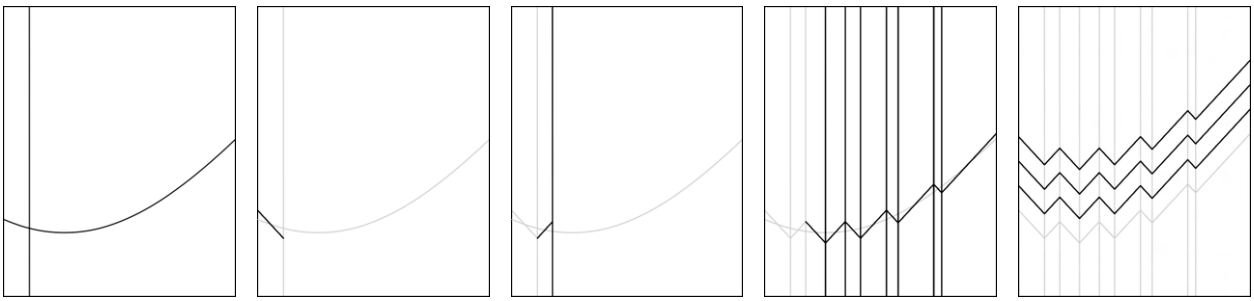


(a) *a basic Miura-ori fold* **(b)** *its crease pattern* **(c)** *odd vs. even number of zigzags* **(d)** *varied spacing*

Figure 2: *The Miura-ori fold.*

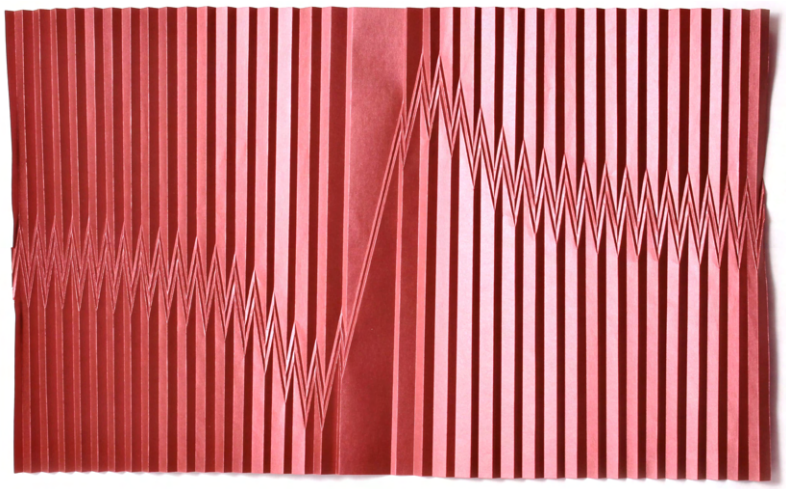
The Algorithm

To generate the crease pattern, start with the desired curve and place a vertical line an initial step size s_i from the left edge (Figure 3a). Then place the first diagonal line segment such that it intersects the function at a distance $\frac{s_i}{2}$ from the edge. One endpoint should lie along the left edge, and the other should lie along the vertical line (Figure 3b). Next, reflect this segment across the vertical line and extend or shorten its length as necessary, such that the segment's midpoint intersects the function. Place another vertical line at its endpoint (Figure 3c). Repeat step (c) until the zigzag is completed and has reached the opposite edge, at which point the original curve can be deleted (Figure 3d). Any number of zigzags can be replicated above or below the original to fill the page and achieve a desired aesthetic (Figure 3e). Figure 4 shows a variety of functions that I folded from Stardream paper (chosen for its metallic properties and variety of colors) using this method. In the supplementary materials, I provide a Python script that automates this process.

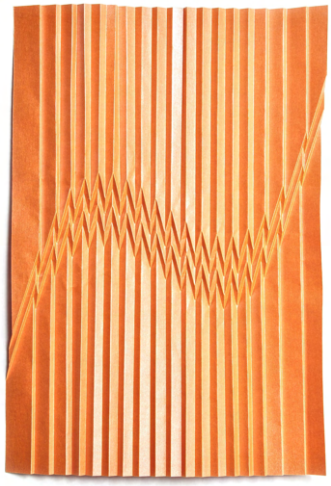


(a) **(b)** **(c)** **(d)** **(e)**

Figure 3: *Generating the crease pattern.*

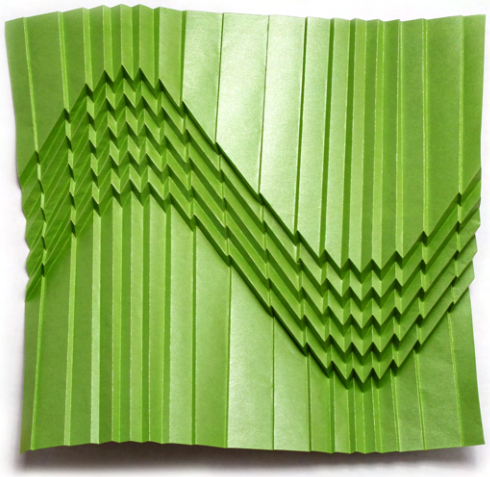


(a) $D_+(x) = e^{-x^2} \int_0^x e^{t^2}\, dt$

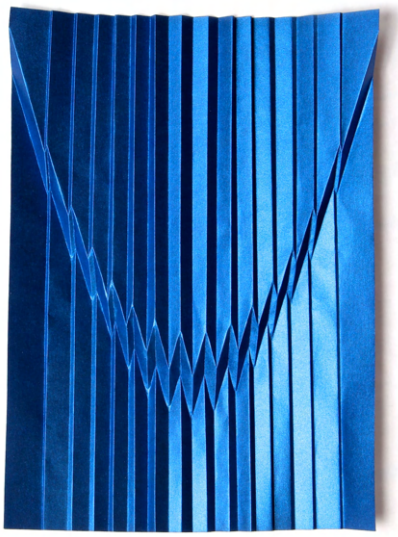


(b) $y = x^3 + 3x^2 + x + 2$

(c) $x^2 - y^2 = 1$



(d) $y = \sin(x)$



(e) $y = x^2$

(f) $y = \sec(x)$

Figure 4: *The algorithm applied to various functions.*

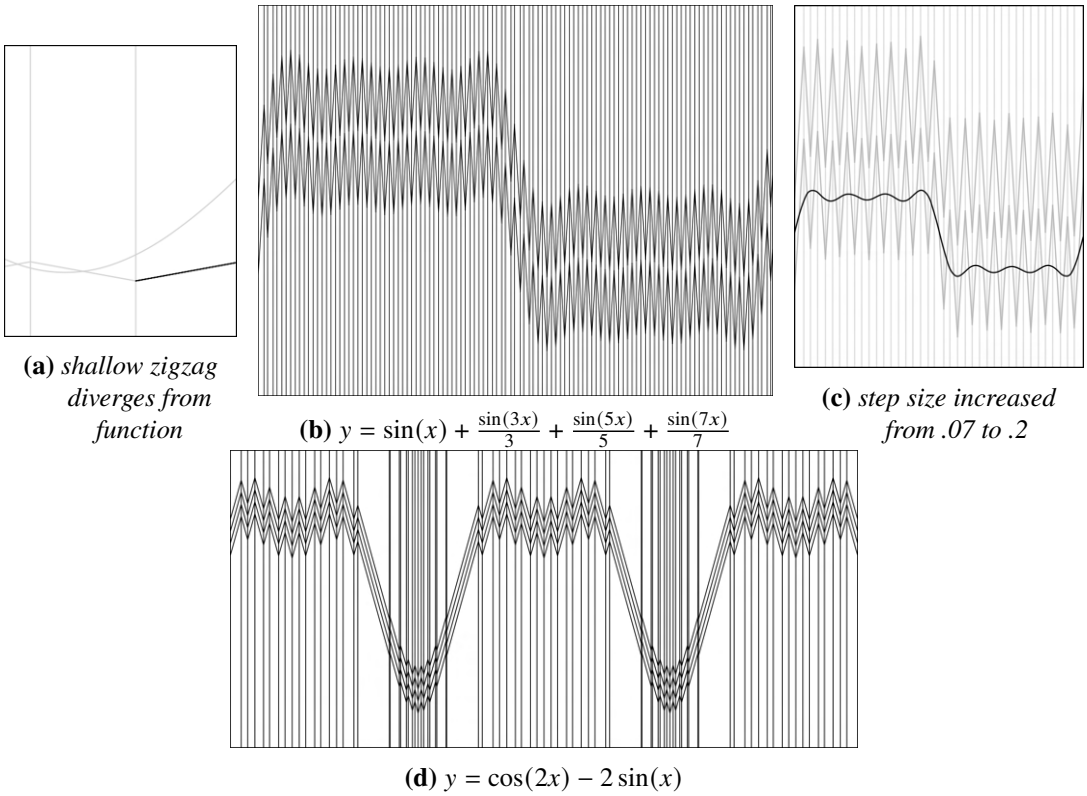
Design Limitations and Considerations

Usable Functions

This method can be implemented for a large variety of functions, such as exponentials (like the Dawson function, Figure 4a), polynomials (Figures 4b & 4e), hyperbolas (Figure 4c), and trigonometric functions and their variants (Figures 1, 4d, 4f, 5b, 5d, 8, & 9). However, there exist some functions for which this method cannot generate crease patterns. With some exceptions that I discuss later, the curve used must pass the vertical line test and cannot include regions of vertical tangency. Therefore, closed loops like circles and ellipses, as well as space-filling curves like the Peano and Hilbert curves cannot be created with this method. And while it can model discontinuous functions such as tangent (Figure 1b) and secant (Figure 4f), the discontinuities themselves must exist outside the edges of the paper. For example, the floor function would not be possible with this method, as the algorithm does not address discontinuities that are on the page. The end points of each line segment would create vertices that are not flat-foldable. As such, my Python code only handles continuous functions that pass the vertical line test.

Slope of the Zigzag

Another mathematical limitation to this algorithm is the angle of the zigzag. In order to ensure the zigzag can follow the curve, the magnitude of its slope must exceed the maximum value of the function's derivative on the interval. Figure 5a shows the algorithm applied to the same curve with the same initial step size as Figure 3. However, the slope of the initial zigzag is too shallow, so after the second reflection, it diverges from the function and can no longer follow the curve.



(a) *shallow zigzag diverges from function*

(b) $y = \sin(x) + \frac{\sin(3x)}{3} + \frac{\sin(5x)}{5} + \frac{\sin(7x)}{7}$

(c) *step size increased from .07 to .2*

(d) $y = \cos(2x) - 2\sin(x)$

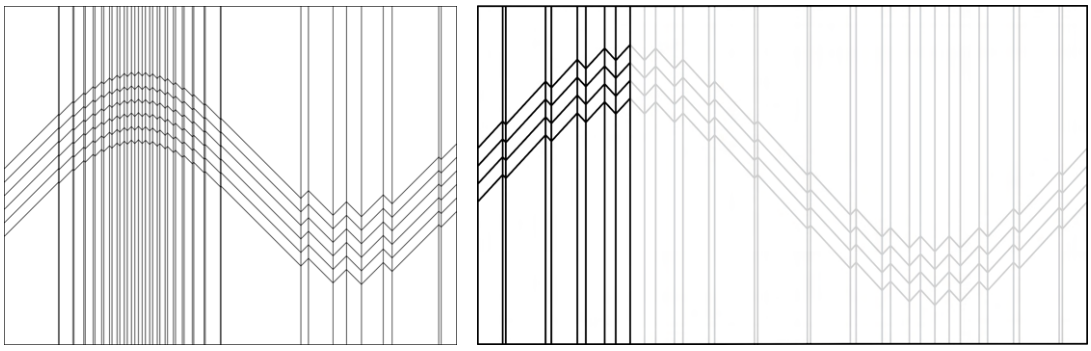
Figure 5: *Slope of the zigzag.*

Furthermore, this algorithm doesn't lend itself well to functions with steep slopes, both aesthetically and practically. Aesthetically, using a very steep zigzag makes the function being modeled appear very thick, so

it is harder to discern the original function. Practically, patterns with steep slopes are more difficult to fold because the space between folds becomes miniscule. Figure 5b shows the crease pattern for a fourth order Fourier series approximation of a square wave to illustrate these concepts. We can increase this space by using a larger step size; however, too large a step size and the pattern may not be recognizable as the input function (Figure 5c, the black line represents the original function being modeled). The smaller the step size, the more closely the crease pattern will resemble the function. Figure 5d shows a different trigonometric function that, despite having additional zigzags, is perhaps more manageable and visually clear, because its maximum slope is less than that of the square wave approximation. I've tested these crease patterns using Origami Simulator [2] but leave the actual folding as an exercise for the reader. PDFs of these patterns (as well as for every design shown in this paper) are included in the supplementary materials.

Symmetry

The algorithm is somewhat sensitive to the initial step size and zigzag slope. Slightly tweaking those inputs can vary how symmetric the algorithm's output appears. Figure 6a shows a Python output for one period (0 to 2π) of a sine wave, where the result has an imbalanced distribution of folds for the crest and the trough. To achieve a more symmetric result, the crease pattern can be constructed from 0 to $\frac{\pi}{2}$, and the remainder of the pattern can be obtained by using a combination of reflections and rotations (Fig 6b).



(a) *imbalanced sine wave crease pattern* **(b)** *a more uniform pattern*

Figure 6: *Manipulating the crease pattern to achieve a symmetric result.*

Quantity of Zigzags

There is no mathematical constraint on the quantity of or spacing between zigzags. Both of these variables can be input into the Python program. Figure 7 shows a sine wave folded with an increasing number of zigzags until it fills the entire page.

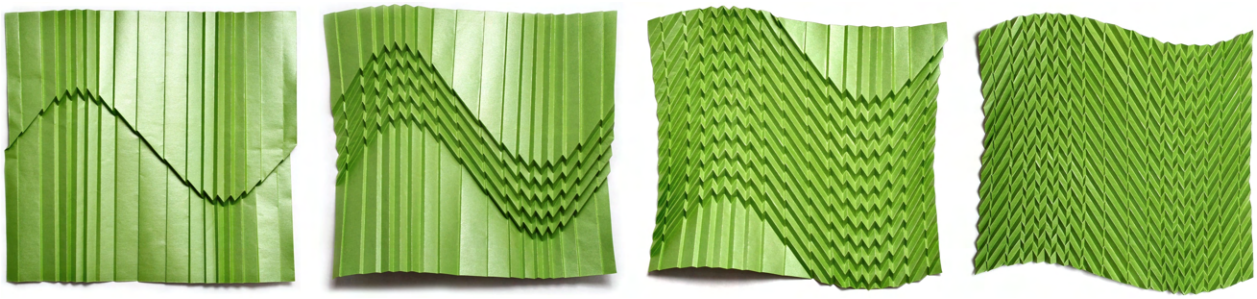
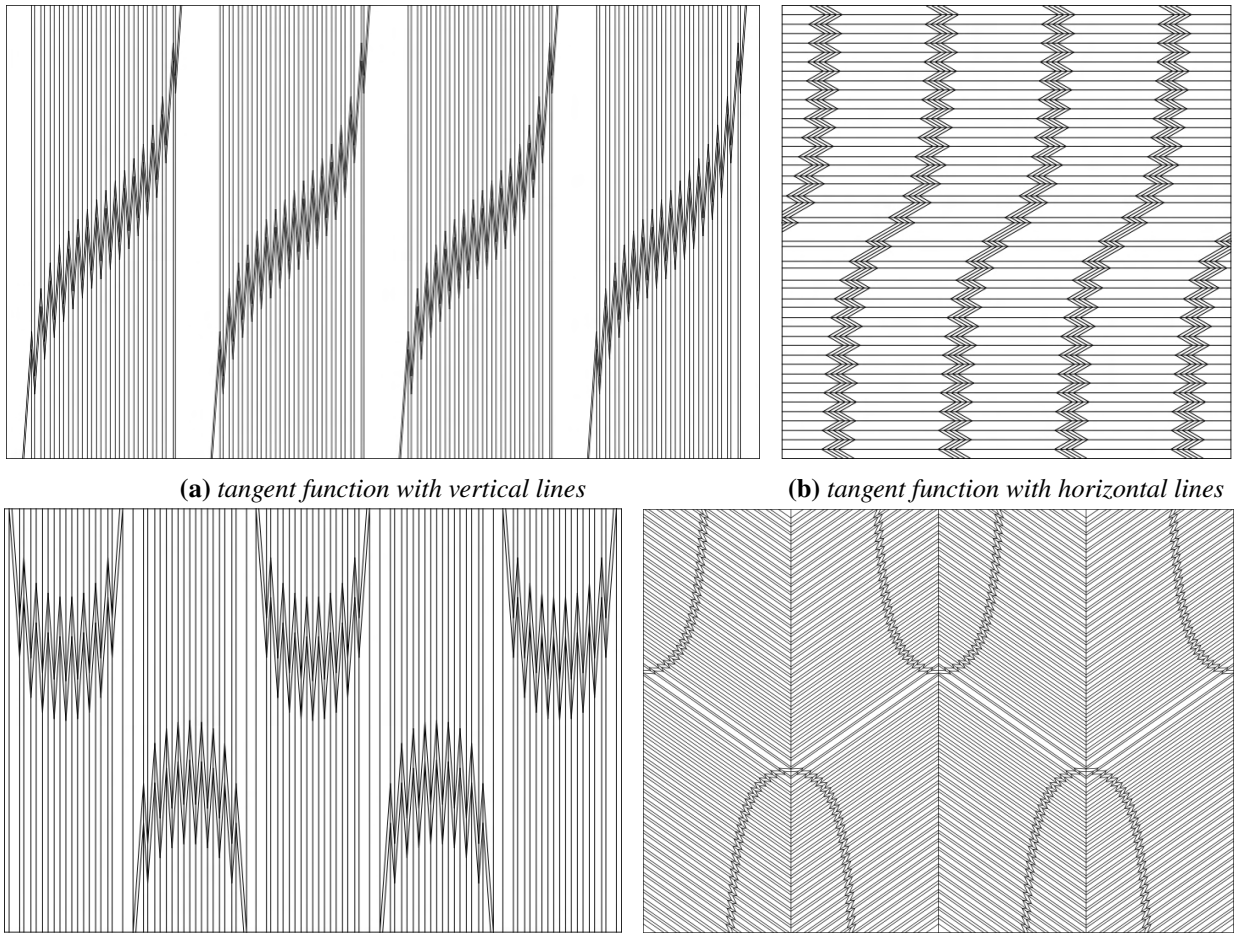


Figure 7: *Sine wave with an increasing number of zigzags.*

Rotating the axis of the function

So far the algorithm has only been discussed for zigzags traversing a set of vertical lines. There are some instances where we might want to design the function relative to a different axis. The background lines should remain parallel, but they can be rotated to a non-vertical orientation to create a more pleasing crease pattern. For instance, the tangent function has a very steep slope as it approaches the asymptotes of the function. Figure 8a shows the algorithm applied to the tangent function keeping the background lines vertical. The steepness required of the slope makes the function appear thick and blurry, and it makes the design more difficult to fold. However, creating the function against a backdrop of horizontal lines creates a more pleasing result (Figure 8b & 1b). The slope of the zigzag doesn't need to be as steep, because the derivative with respect to the rotated axis has a maximum magnitude of one, instead of approaching ∞ at the asymptotes.



(a) *tangent function with vertical lines*

(b) *tangent function with horizontal lines*

(c) *secant function with vertical lines*

(d) *secant function with rotated and mirrored lines*

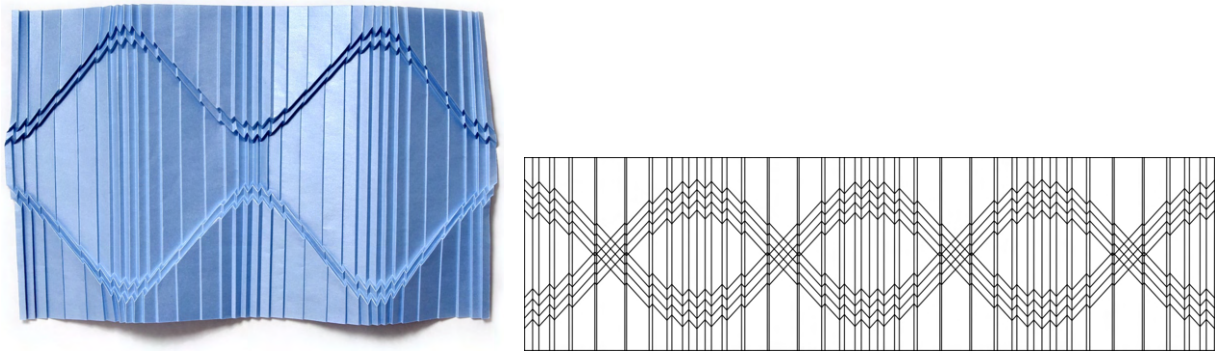
Figure 8: *Rotating the parallel lines to get a more pleasing result.*

The secant function has similar issues to the tangent function. Using a background of vertical lines yields a result with an incredibly steep slope, so the crease pattern looks messy and isn't immediately recognizable as the secant function (Fig 8c). Unlike the tangent function, switching to horizontal lines won't work, since $y = \sec(x)$ has points of horizontal tangency. Any rotation of the secant function will still result in a curve whose derivative has magnitudes that span 0 to ∞ so there is no one axis of rotation that will work on its own. However, we can exploit the symmetry of the function by using an axis that works well for the region between two adjacent lines of mirror symmetry for the secant function, and then simply reflect the crease

pattern across those lines. Figure 8d shows this result, which looks much cleaner than simply using vertical lines. The folded version of this crease pattern can be seen in Figure 4f.

Exceptions to the Vertical Line Test

In order for a function to be foldable using this method, it usually must pass a "direction" line test, where the "direction" is the orientation of the parallel background lines. If the lines are vertical, then it should pass the vertical line test. The exception is that multiple curves can be placed against the same set of parallel lines so long as their derivatives have the same magnitude at each point that a given background line intersects the curve. For example, the tangent function in Figure 1b doesn't pass the horizontal line test, but the derivative is constant along any given horizontal line. If the parallel lines were rotated to some angle that was neither vertical nor horizontal, the tangent function would fail the "direction" line test, as the points where a given line intersect the function would have derivatives of different magnitudes. The hyperbola in Figure 4c also doesn't pass the horizontal line test, but the derivative has the same magnitude as you move horizontally across the function. As a final example, the sine curve on its own passes the vertical line test, but a result where the function is mirrored across a horizontal axis can also be folded even though the combination of functions fails the vertical line test (Figure 9a).



(a) *a folded sine wave and its reflection* **(b)** *improper crease pattern of intersecting sine waves*

Figure 9: *A sine wave and its reflection on the same folding pattern.*

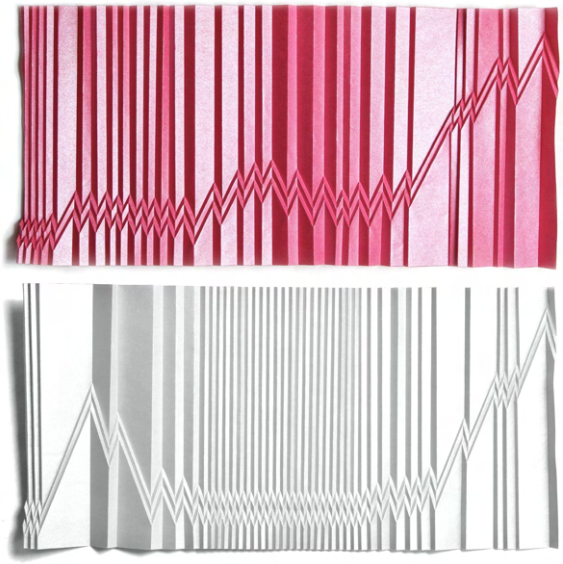
Future Work

I want to continue exploring this algorithm and pushing its limitations. One area I haven't addressed yet but would like to explore is that of intersecting curves. For instance, in Figure 9b, the two sine waves have been pushed closer together so that they intersect to resemble the outline of a standing wave. The overlapping zigzags in the crease pattern yield a result which is not neatly foldable, as something must be done to address these intersections. Another area I'd like to explore is the use of curved creases, as the vast majority of my designs involve only linear creases. I hope to eventually modify this algorithm so it can be implemented with curved folding. I would also like to see if I can find a way to work around the "direction" line test. That, combined with resolving intersections would allow me to explore functions in polar coordinates using origami.

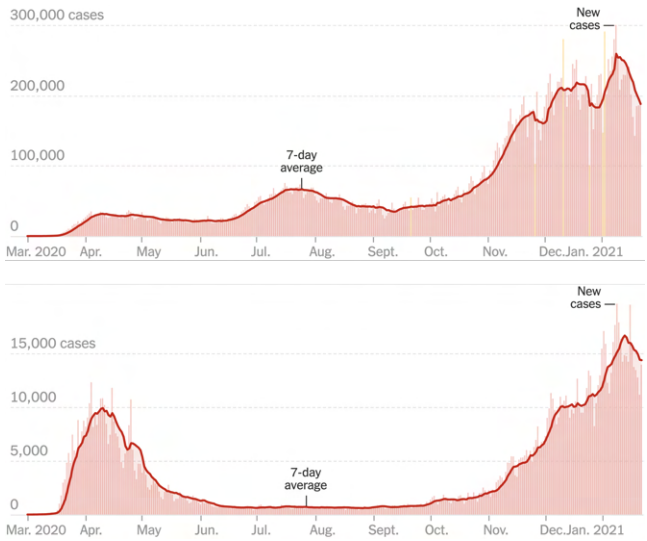
Final Remarks

I developed this algorithm while in isolation due to the pandemic. The algorithm need not be limited only to functions defined by equations. As a parting note, Figure 10a shows a pair of folded designs modeled on

the data for the number of new cases of COVID-19 in the United States (above) and New York (below) from March 1, 2020 through January 20, 2021. Figure 10b shows the charts [4] I used as my reference to create these crease patterns.



(a) *Cases of Covid in Creasing*



(b) *reference data [4]*

Figure 10: *Using the algorithm to follow data.*

Acknowledgements

I'd like to thank Marcus Michelen, Ben Fritzson, Ben Parker, and Sarah Gelsinger Brewer, for ideas, inspiration, and useful discussions. I'd also like to thank Jen Tashman, Nate Biagiotti, Leanna Pancoast, and Ninh Tran for comments on a previous draft.

References

[1] E. Demaine and M. Demaine. "History of Curved Origami Sculpture." http://erikdemaine.org/curved/history/.

[2] A. Ghassaei. "Origami Simulator." https://origamisimulator.org

[3] K. Miura. "Method of Packaging and Deployment of Large Membranes in Space." *The Institute of Space and Astronautical Science*. Report No. 618. December 1985.

[4] The New York Times. "Coronavirus in the U.S.: Latest Map and Case Count." https://www.nytimes.com/interactive/2020/us/coronavirus-us-cases.html

[5] U. Nguyen, B. Fritzson, and M. Michelen. "Folding Functions: Origami Corrugations from Equations." *Bridges Conference Proceedings*, Linz, Austria, July 16-20, 2019, pp. 43-50. http://archive.bridgesmathart.org/2019/bridges2019-43.pdf.

[6] B. Parker. "Waves." http://www.brdparker.com/waves---2018.html.