Implementing Nilsson and Passare's Coamoeba Algorithm

Jeff Sommars
Wheaton College

Coamoeba: defined

- For any polynomial in two variables with complex coefficients, let us define the coamoeba to be:

$$(arg(x_1), arg(x_2)) | f(x_1, x_2) = 0 \text{ and } x \text{ in } (\mathbb{C}^*)^2$$

with the argument defined
as it usually is

$$arg(z) := arctan(Im(z)/Re(z))$$

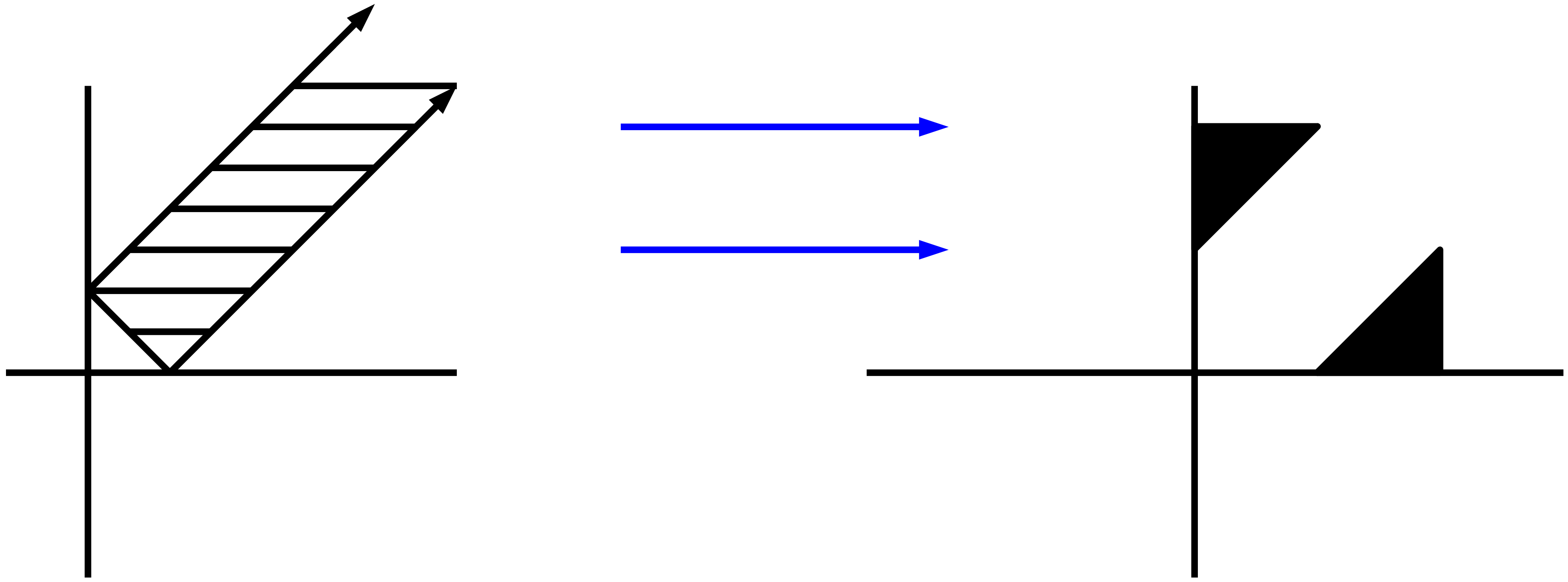
Coamoeba: extended

- Though this goes beyond the scope of my project, the true definition of a coamoeba is far more general.

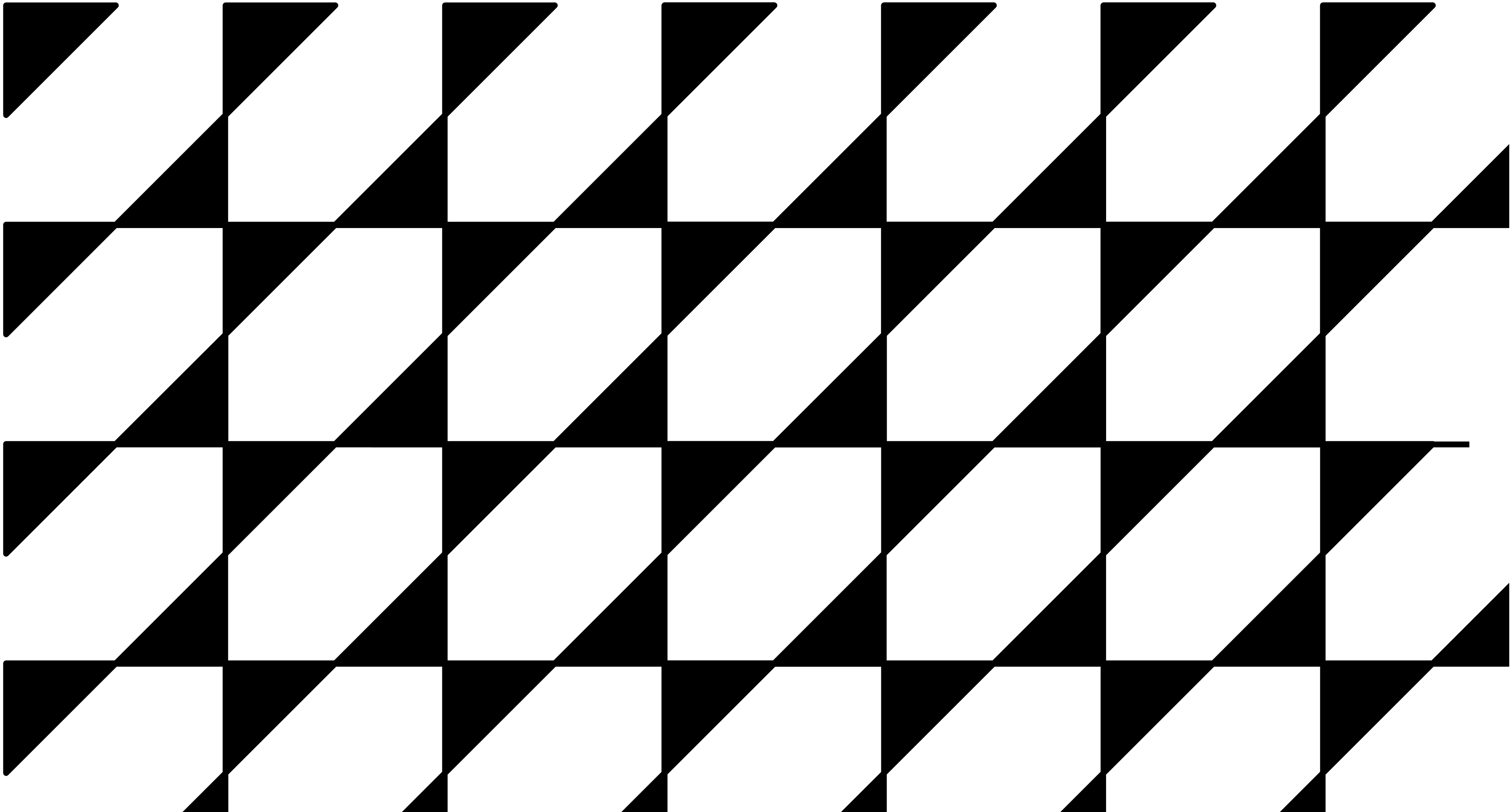
$$(arg(x_1), ..., arg(x_n)) | f(x_1 ..., x_n) = 0 \text{ and } x \text{ in } (\mathbb{C}^*)^n$$

Example

$$f(x_1, x_2) = 1 + x_1 + x_2$$



Example (continued)



A Better Approach

-Finding coamoebas for more complicated equations becomes quite tedious, and there's a high chance of error.

-Nilsson and Passare created an algorithm to draw discriminant coamoebas in the two dimensional case.

-No paper has yet been published that gives an algorithm for how to draw discriminant coamoebas in greater than two dimensions.

Importance of Coamoebas

- Mikhalkin- Correspondence Theorem
- Euler-Mellin Transform—complement component of coamoeba?
- A theorem on generic analytic curves
- Applications in physics

A matrices and B matrices

Algorithm: Step One

The Horn-Kapranov parametrization is a rational mapping given by:

$$\Psi[t_1 : t_2] = \left(\prod_{j=1} \langle b_j, t \rangle^{b_{j1}}, \prod_{j=1} \langle b_j, t \rangle^{b_{j2}} \right)$$

As you recall, our B matrix has row vectors of (-1,-1), (1,0), and (0,1).

$$x_1 = \langle(-1,-1), t\rangle^{-1} \langle(1,0), t\rangle^{1} \langle(0,1), t\rangle^{0}$$

$$x_2 = \langle(-1,-1), t\rangle^{-1} \langle(1,0), t\rangle^{0} \langle(0,1), t\rangle^{1}$$

Algorithm: Step One (continued)

The next step is obvious: simplify!

$$\Psi[1:t] = \left(-\frac{1}{1+t}, -\frac{t}{1+t}\right)$$

Now, take the limit as t approaches infinity and test whether the result is greater than or less than zero (or for this case, if it approaches zero from above or from below).

Algorithm: Step One (continued)

There are four distinct possibilities:

$$(-,-),(-,+),(-,+),(+,+)$$

Each corresponds to a different starting point:

$$(\pi,\pi),(\pi,0),(0,\pi),(0,0)$$

Algorithm: Step Two

Now we know where we need to start drawing, but what are we going to draw?

We need to reorder the rows of our B matrix so that we have the row vectors with decreasing normal slopes:

$$\beta_j = -b_{j1}/b_{j2}$$

$$\infty > \beta_1 \geq \beta_2 \geq \ldots \geq \beta_N \geq (-)\infty\,.$$

Algorithm: Step Two (continued)

For our example, our row vectors get reordered like this:

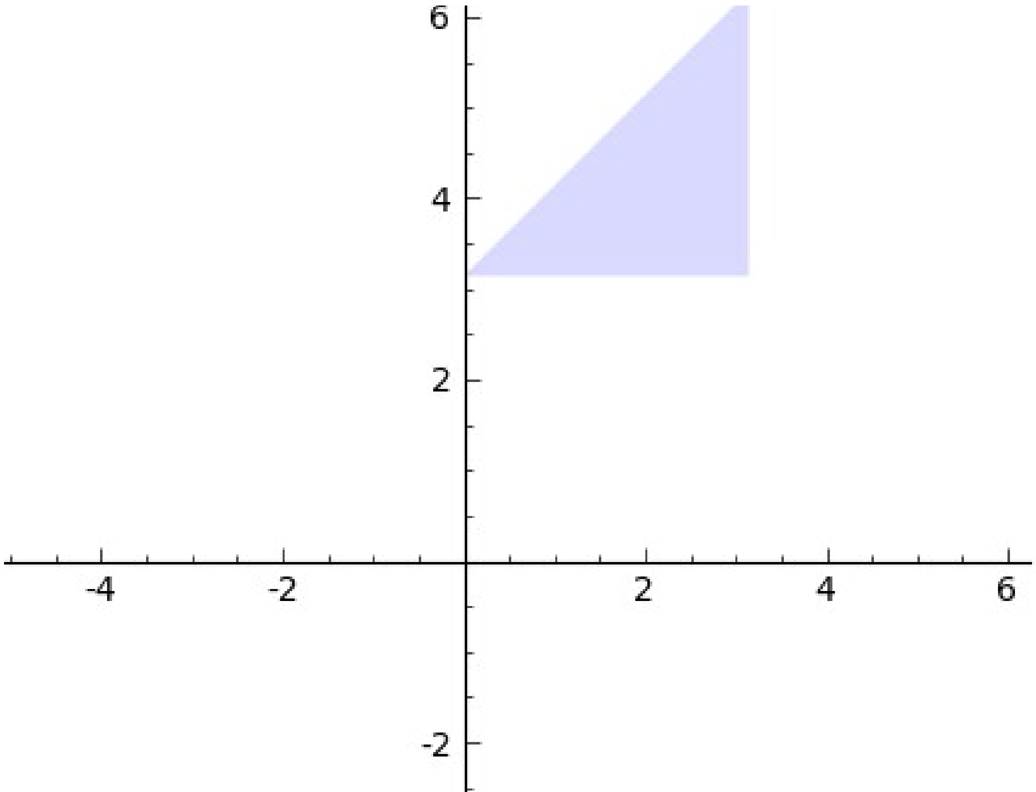
$$b_1 = (0,1),\, b_2 = (-1,-1),\, b_3 = (1,0)$$

At this point, we can start to draw the principle coamoeba.

Algorithm: Exciting Conclusion

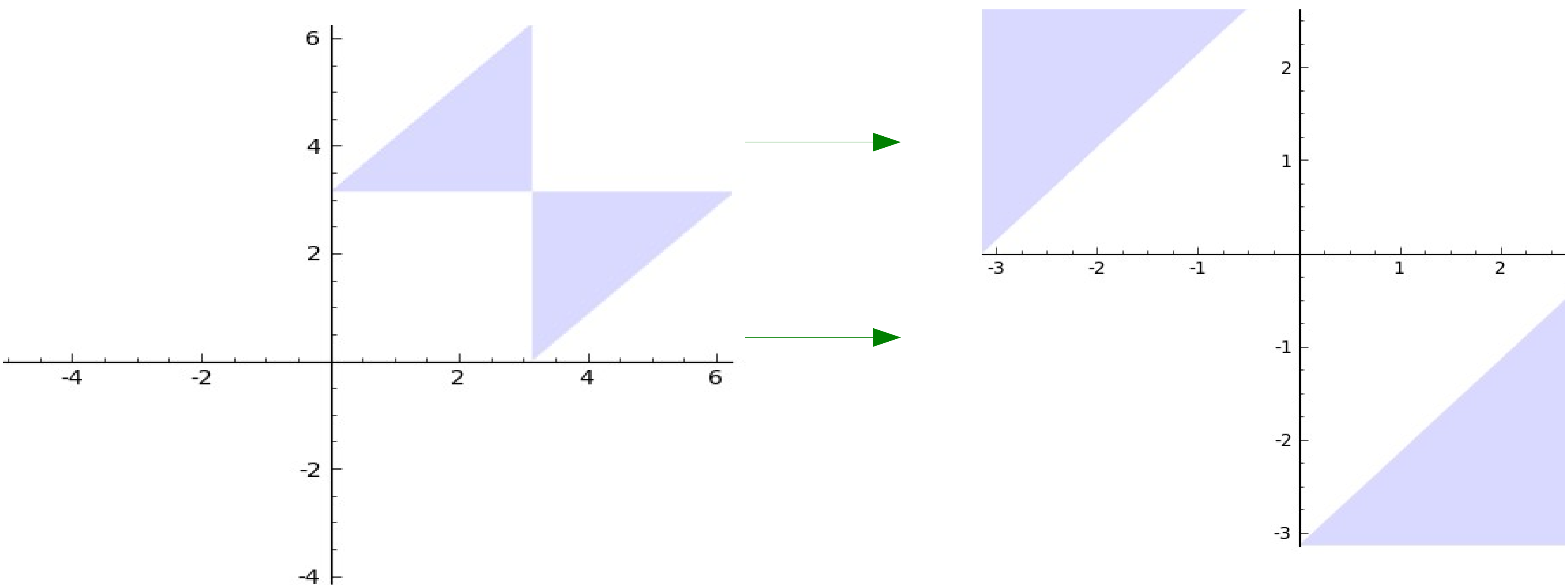
Consider the fundamental domain centered at the origin. Start at the point that you found in the Kapranov parametrization. Then start drawing the vectors in order, starting at j=1 and ending at j=N.

For our example:



Algorithm: The Conclusion (continued)

Finally, take the same starting point, and draw the vectors in the same order
but with each component
having opposite sign.



Example

$$A = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 \end{pmatrix}$$

$$D_A(a) = 27a_1^2a_4^2 + 4a_1a_3^3 + 4a_2^3a_4 - 18a_1a_2a_3a_4 - a_2^2a_3^2$$

$$D_B(x) = 27x_1^2 + 4x_1 + 4x_2^3 - 18x_1x_2 - x_2^2$$

$$B = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ -3 & -2 \\ 2 & 1 \end{pmatrix}$$

Example

$$B = \begin{pmatrix} 0 & 1 \\ -3 & -2 \\ 2 & 1 \\ 1 & 0 \end{pmatrix}$$

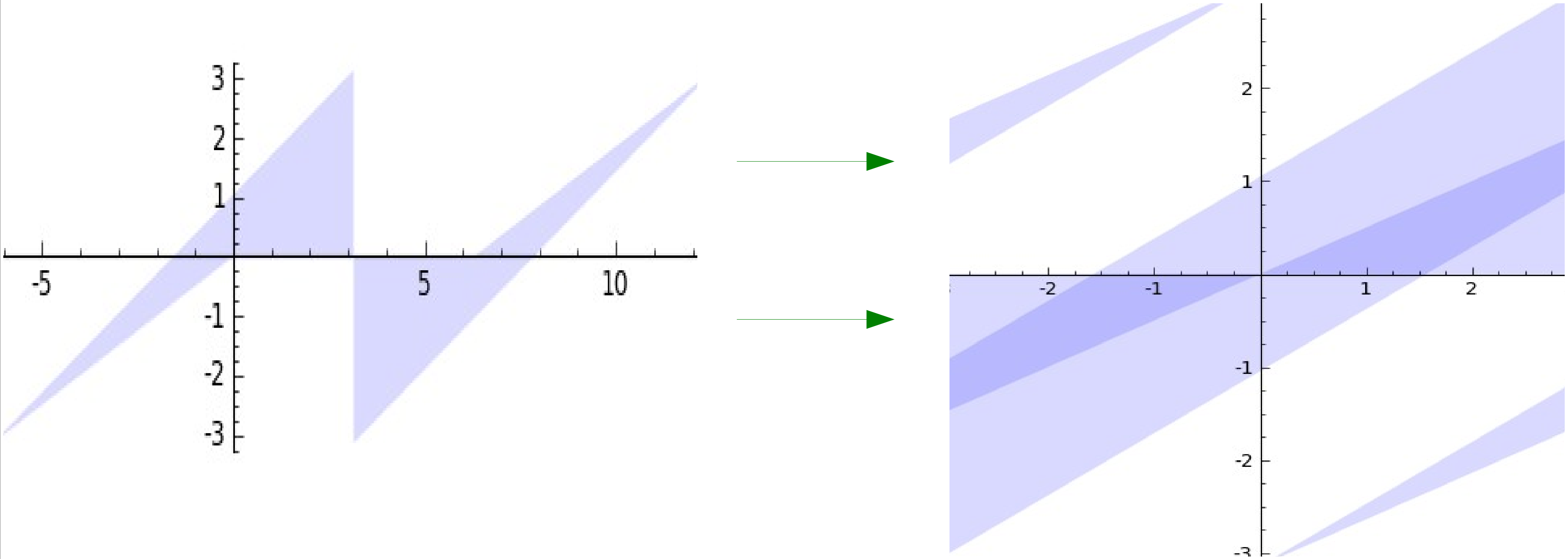
$$x_1 = -\frac{(2+t)^2}{(3+2t)^3} \quad x_2 = \frac{t(2+t)}{(3+2t)^2}$$

$$t = 2 \quad x_1 = -\frac{16}{243}, \quad x_2 = \frac{8}{49}$$

$$(\pi, 0)$$

Example

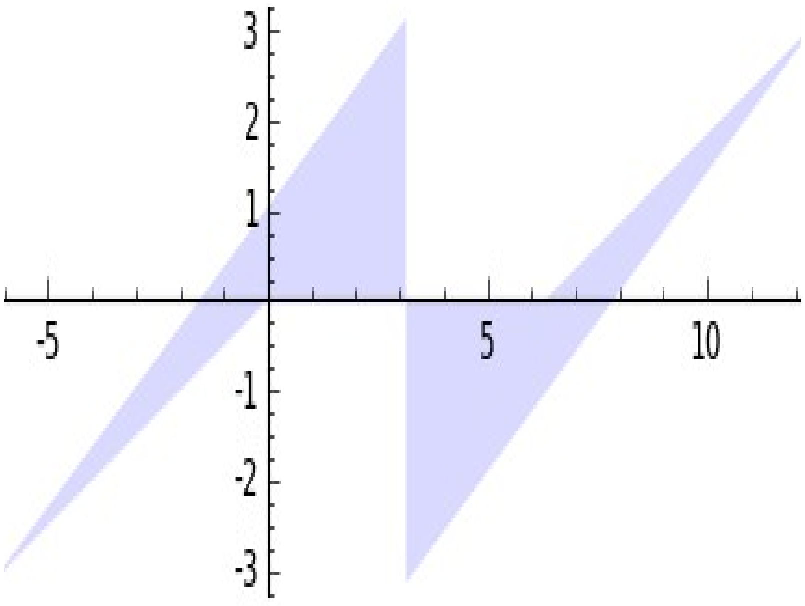
$$A = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 \end{pmatrix}$$

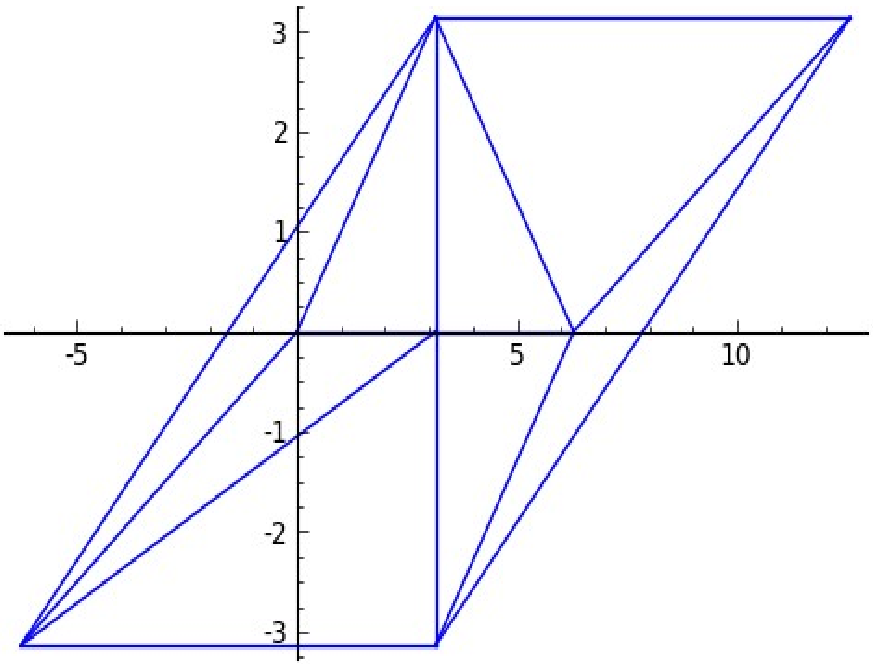


Programming Challenge

- How do you determine the multiplicities of each distinct section correctly?

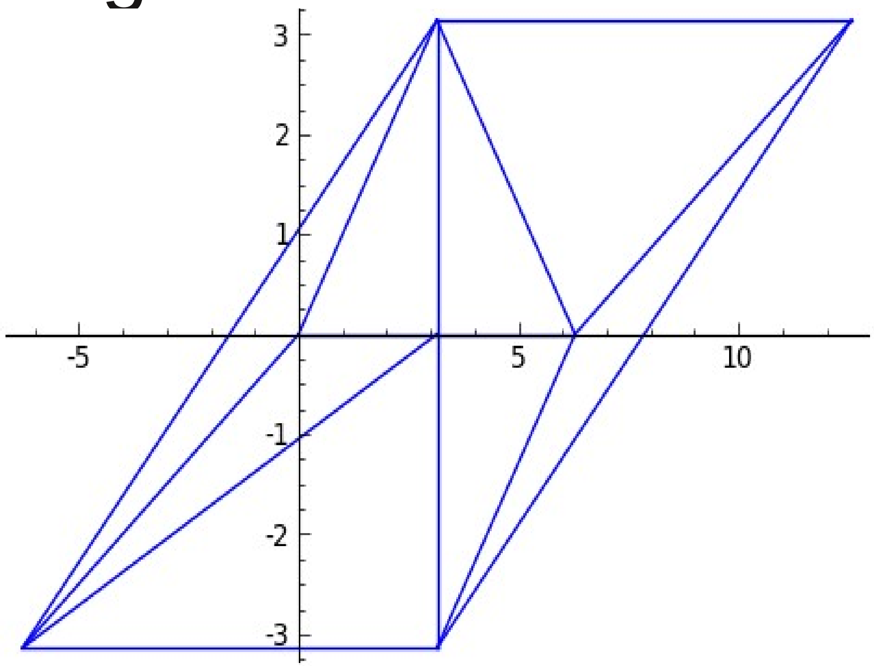
  - First, split the complex polygon into triangles.





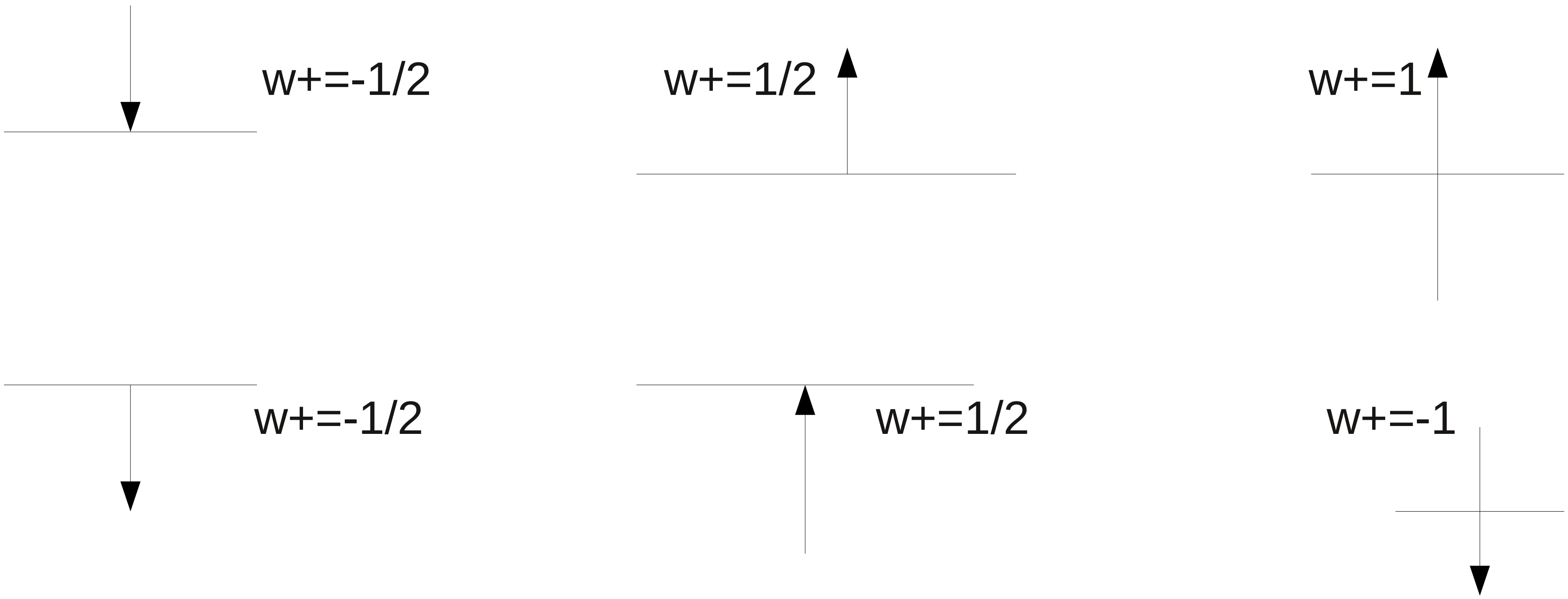
Programming (continued)

Take each triangle, test a point inside each triangle using a winding number algorithm, and then shade the entire region with a shade that corresponds to the winding number.



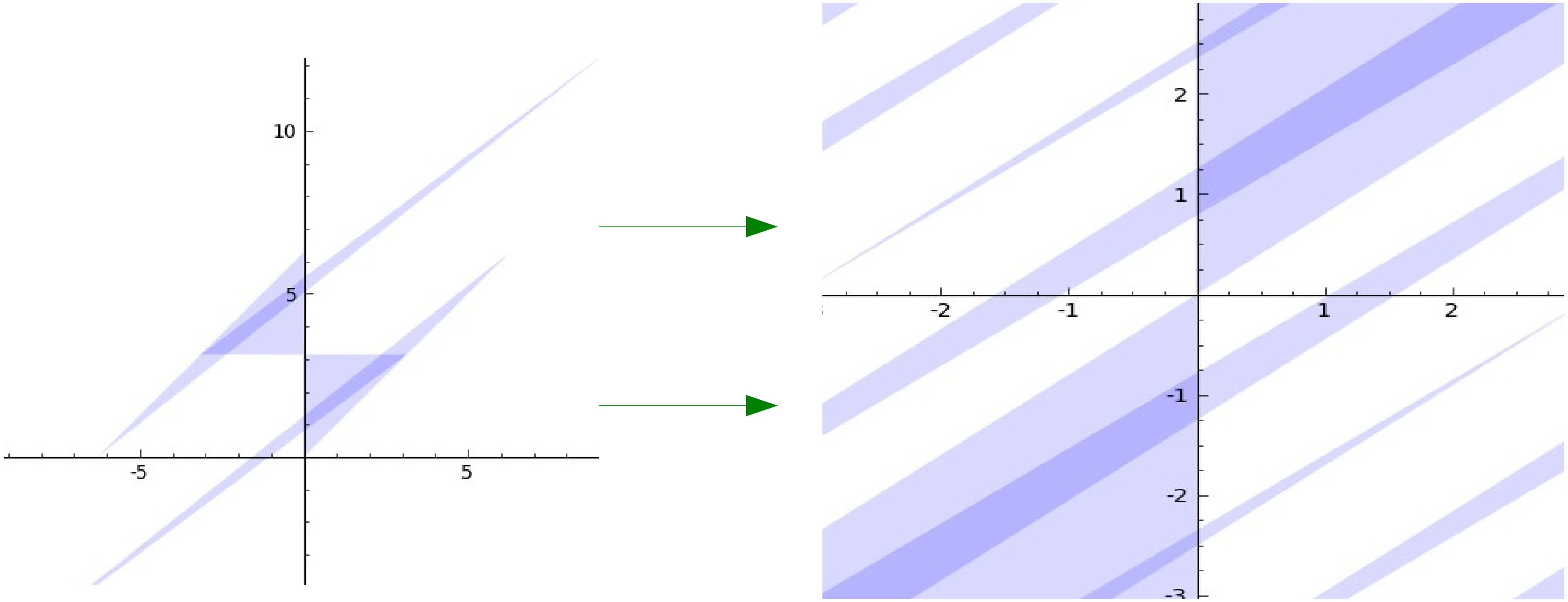
Winding Number Algorithm

- Step One: Shift the complex polygon such that the point to be tested lies at the origin.
- Step Two: Start at the first vertex, and simply connect connect the vertices



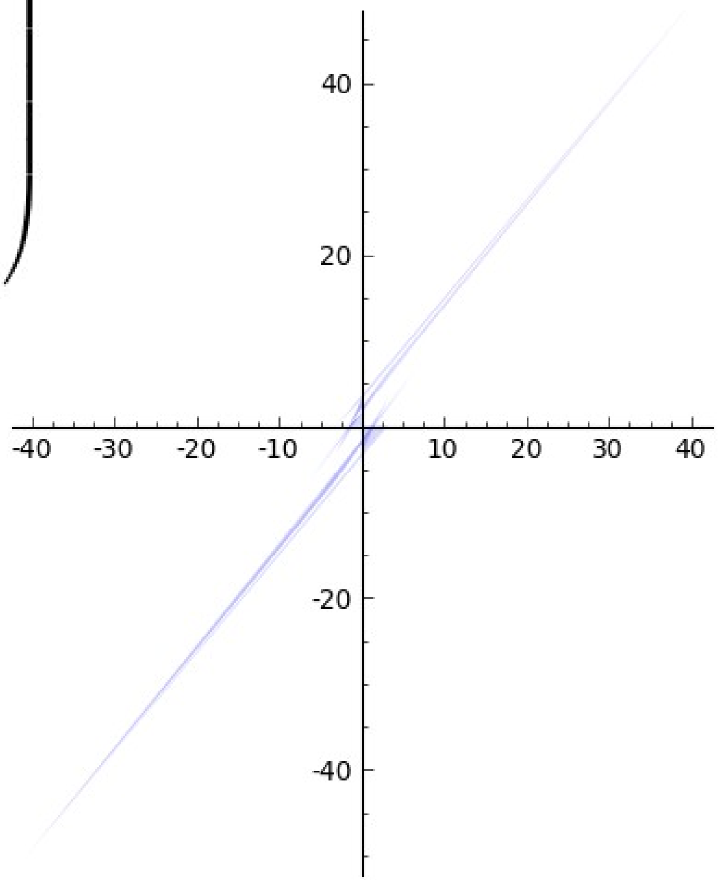
An Example Where the Winding Number is non-trivial

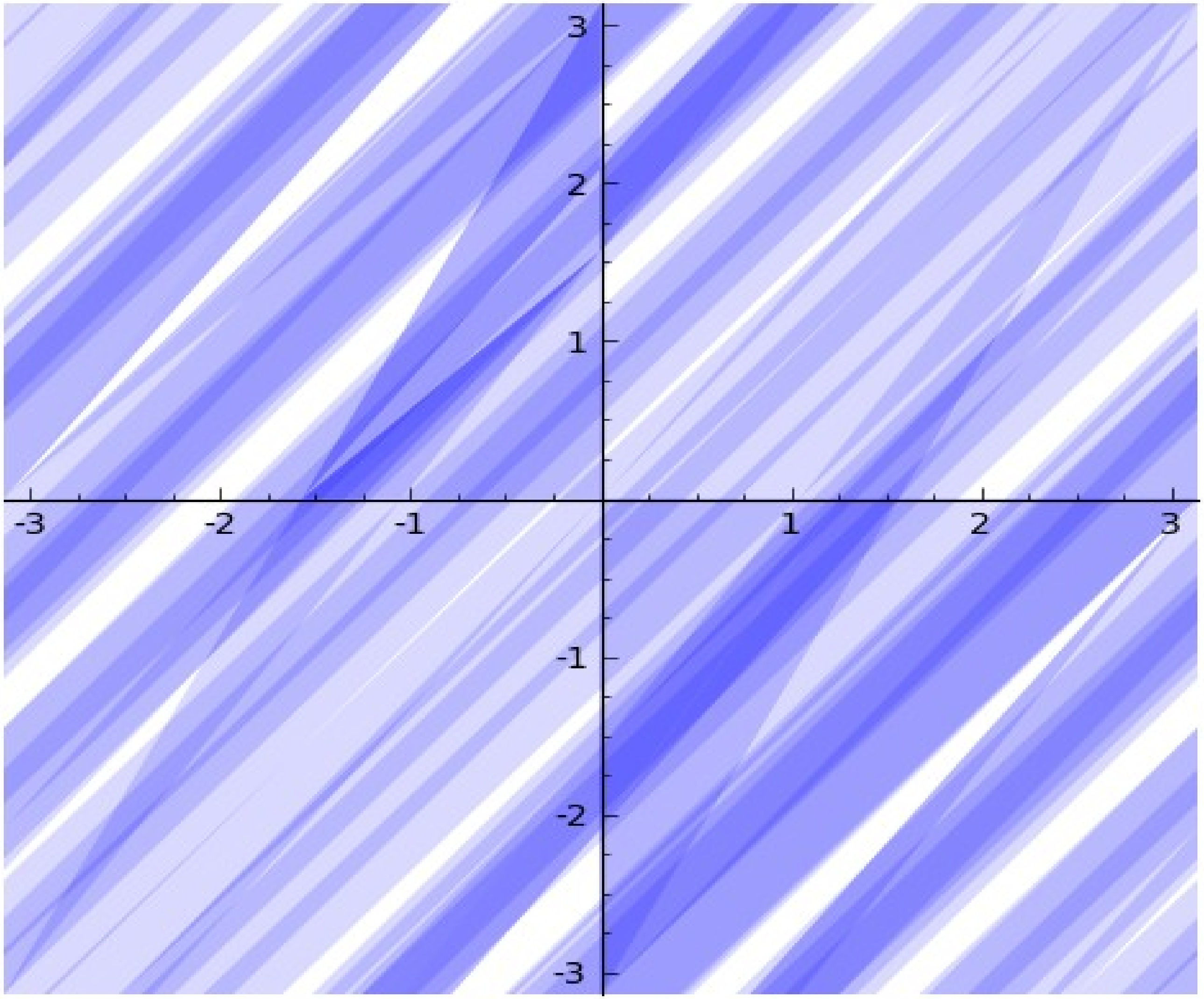
$$A = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 2 & 1 & 0 & 2 & 3 \\ 3 & 2 & 0 & 1 & 2 \end{pmatrix}$$



The Most Exciting Example

$$A = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ -2 & -3 & 0 & 1 & 0 & 0 & 0 & 0 \\ 3 & 4 & 0 & 0 & 1 & 0 & 0 & 0 \\ -4 & -5 & 0 & 0 & 0 & 1 & 0 & 0 \\ 14 & 16 & 0 & 0 & 0 & 0 & 1 & 0 \\ -11 & -13 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$





3
2
1
-3
-2
-1
1
2
3
-1
-2
-3

Much Thanks To:

- Dr. Rojas
- Dr. Nisse
- Korben Rusek
- Daniel Smith