Math 511 Final Project: Custom Grid Generation

Shane McQuarrie

Introduction

Most techniques for computing numerical solutions to PDEs assume a nice domain, such as the unit square or circle, for the physical domain of the problem. In real-world applications, however, the domain is rarely so regular. The goal of this project is to generate smooth, computationally friendly grids for various domains. We will use three different techniques, each of which uses successive over-relaxation (SOR) to iteratively compute a smooth grid.

Let D be a physical domain over which we want to generate a smooth grid, and D' be a rectangular computational domain. Denote the points of D as (x, y) and the points of D' as (ξ, η). For each method, we must define a transformation $T : D' \to D$ or $S : D \to D'$ to relate (x, y) and (ξ, η). For each domain, we must also specify the boundary transformation $\partial T : \partial D' \to \partial D$ or $\partial S : \partial D \to \partial D'$.

Next, denote $(X_{i,j}, Y_{i,j})$ as the discretization of the physical grid D and $(\xi_{i,j}, \eta_{i,j})$ as the discretization of the computational grid D'. If there are N_ξ points in the ξ-direction (i) and N_η points in the η direction (j), then assuming a uniform spacing in D' of $\Delta\xi = \Delta\eta = 1$, we have $D' = [1, N_\xi] \times [1, N_\eta]$, $\xi_{i,j} = i$, and $\eta_{i,j} = j$. There will also be N_ξ points in the ξ-direction and N_η points in the η-direction of D, but since our desired grids will not follow a regular Cartesian or polar structure, it doesn't make sense to talk about Δx or Δy in the usual sense.

The Amsden-Hirt Algorithm

The Amsden-Hirt procedure claims that grid points $(x(\xi, \eta), y(\xi, \eta))$ satisfying the following Laplace equations will be smooth.

$$\Delta_{\xi,\eta} x = x_{\xi\xi} + x_{\eta\eta} = 0 \tag{1}$$

$$\Delta_{\xi,\eta} y = y_{\xi\xi} + y_{\eta\eta} = 0. \tag{2}$$

We approximate these equations with second-order centered difference quotients. In this case, the x- and y-coordinates are totally decoupled, which simplifies the approximation.

First, (1) becomes

$$0 = \frac{X_{i-1,j} - 2X_{i,j} + X_{i+1,j}}{(\Delta\xi)^2} + \frac{X_{i,j-1} - 2X_{i,j} + X_{i,j+1}}{(\Delta\eta)^2}$$

$$0 = (\Delta\eta)^2 (X_{i-1,j} - 2X_{i,j} + X_{i+1,j}) + (\Delta\xi)^2 (X_{i,j-1} - 2X_{i,j} + X_{i,j+1})$$

$$2((\Delta\xi)^2 + (\Delta\eta)^2)X_{i,j} = (\Delta\eta)^2 (X_{i-1,j} + X_{i+1,j}) + (\Delta\xi)^2 (X_{i,j-1} + X_{i,j+1})$$

$$X_{i,j} = \frac{(\Delta\eta)^2}{2((\Delta\xi)^2 + (\Delta\eta)^2)} (X_{i-1,j} + X_{i+1,j}) + \frac{(\Delta\xi)^2}{2((\Delta\xi)^2 + (\Delta\eta)^2)} (X_{i,j-1} + X_{i,j+1})$$

$$X_{i,j} = \theta_\eta (X_{i-1,j} + X_{i+1,j}) + \theta_\xi (X_{i,j-1} + X_{i,j+1}).$$

This general formula reduces nicely under the assumption that $\Delta\xi = \Delta\eta = 1$, since in that case $\theta_\xi = \theta_\eta = 1/4$.

$$X_{i,j} = \frac{1}{4} (X_{i-1,j} + X_{i+1,j} + X_{i,j-1} + X_{i,j+1}) \tag{3}$$

For this algorithm, the equation for Y is identical.

$$Y_{i,j} = \frac{1}{4} (Y_{i-1,j} + Y_{i+1,j} + Y_{i,j-1} + Y_{i,j+1}) \tag{4}$$

The Iterative Algorithm

Instead of explicitly formulating and solving a finite difference system, we apply SOR to iteratively converge to a solution. We assign the kth iteration of $X_{i,j}$, denoted $X_{i,j}^{(k)}$, based on the points in (3) that have already been calculated for the kth iteration, those that were calculated in the $(k-1)$th iteration, and the previous $X_{i,j}$, weighting each piece based on a *relaxation factor* $\omega \in (0, 2)$. We do the same for Y, arriving at the SOR equations for this algorithm.

$$X_{i,j}^{(k)} = \frac{\omega}{4} \left(X_{i-1,j}^{(k)} + X_{i+1,j}^{(k-1)} + X_{i,j-1}^{(k)} + X_{i,j+1}^{(k-1)} \right) + (1-\omega) X_{i,j}^{(k-1)} \tag{5}$$

$$Y_{i,j}^{(k)} = \frac{\omega}{4} \left(Y_{i-1,j}^{(k)} + Y_{i+1,j}^{(k-1)} + Y_{i,j-1}^{(k)} + Y_{i,j+1}^{(k-1)} \right) + (1-\omega) Y_{i,j}^{(k-1)} \tag{6}$$

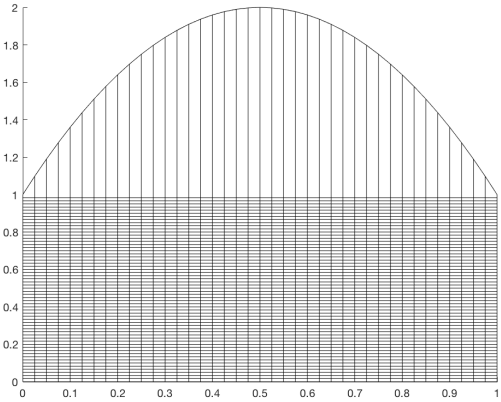
To preserve the grid boundaries, we force our initial grid to coincide with ∂D and only update the interior grid points. For stopping criterion, we compute at most N iterations or stop when $\|X^{(k)} - X^{(k-1)}\|$ and $\|Y^{(k)} - Y^{(k-1)}\|$ are sufficiently small. The complete algorithm is given below, followed by an implementation for MATLAB.

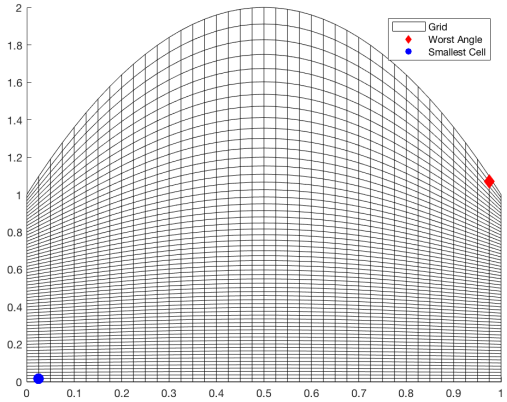
procedure AMSDEN-HIRT-SOR(X, Y, ω, ϵ, N)
  $N_x, N_y \leftarrow \text{size}(X)$
  for $k = 1, \ldots, N$ **do**
    $X^{(k-1)} \leftarrow X$
    $Y^{(k-1)} \leftarrow Y$ ▷ Store the last iteration.
    for $j = 2, \ldots, N_y - 1$ **do**
      for $i = 2, \ldots, N_x - 1$ **do**
        $X_{i,j} \leftarrow \frac{\omega}{4}\left(X_{i-1,j} + X^{(k-1)}_{i+1,j} + X_{i,j-1} + X^{(k-1)}_{i,j+1}\right) + (1-\omega)X^{(k-1)}_{i,j}$
        $Y_{i,j} \leftarrow \frac{\omega}{4}\left(Y_{i-1,j} + Y^{(k-1)}_{i+1,j} + Y_{i,j-1} + Y^{(k-1)}_{i,j+1}\right) + (1-\omega)Y^{(k-1)}_{i,j}$
    if $\|X - X^{(k-1)}\| < \epsilon$ and $\|Y - Y^{(k-1)}\| < \epsilon$ **then**
      break ▷ Stopping criterion.
  return X, Y

```
function gridgenAH(X, Y, w, tol, maxiters)

[Nx,Ny] = size(X);
for k=1:maxiters
    X0 = X;
    Y0 = Y;
    for j=2:Ny-1
        for i=2:Nx-1
            X(i,j) = w*(X(i-1,j)+X0(i+1,j)+X(i,j-1)+X0(i,j+1))/4 + (1-w)*X(i,j);
            Y(i,j) = w*(Y(i-1,j)+Y0(i+1,j)+Y(i,j-1)+Y0(i,j+1))/4 + (1-w)*Y(i,j);
        end
    end
    if norm(X-X0, Inf) < tol && norm(Y-Y0, Inf) < tol
        break                    % Check for convergence.
    end
end

end
```

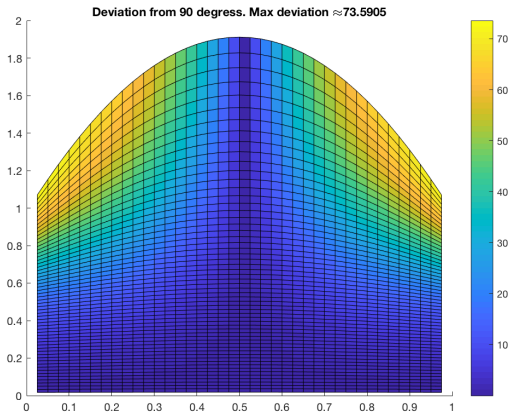
Results

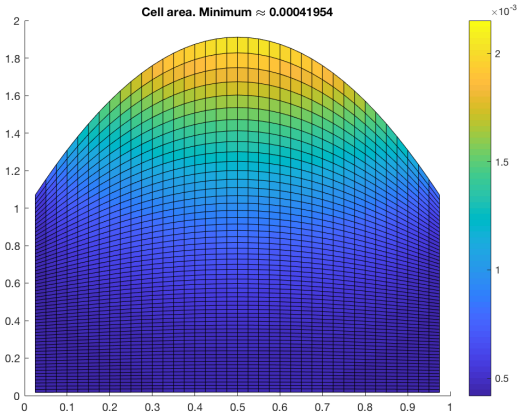
For this and the next algorithm, we examine two domains: one that is convex, and one that is non-convex. The *dome domain* is the convex region enclosed by the lines $x = 0$, $y = 0$, $x = 1$, and the curve $y = 2 - 4(x - \frac{1}{2})^2$, and the *swan domain* is the non-convex region enclosed by the curves $x = 0$, $y = 0$, $x = 1 + 2y + 2y^2$, and $y = 1 - 2x + 2x^2$. We begin with the initial and final grids for the dome domain.



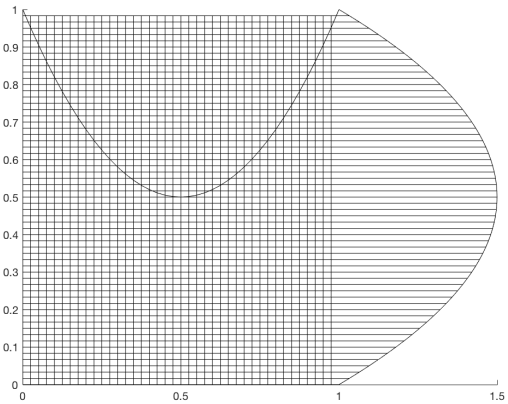


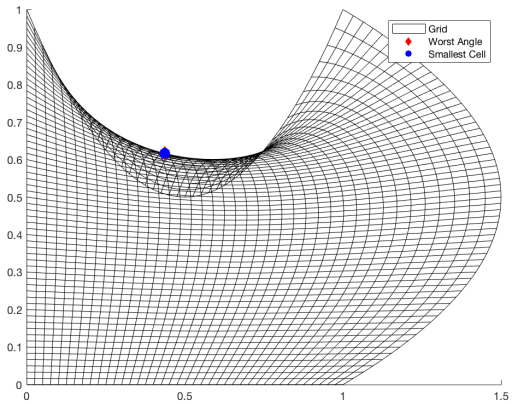
To measure the quality of the final grid, we calculate the angles at each interior grid point and the area of each cell. The results are best visualized as surface plots.

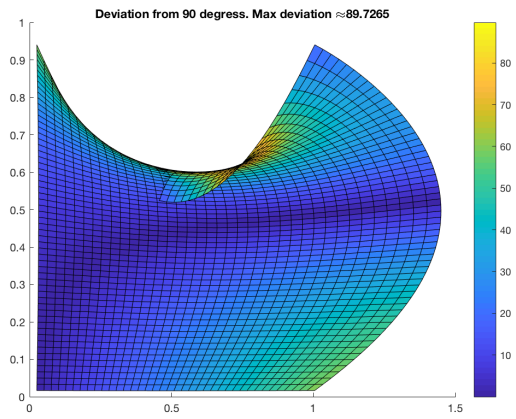


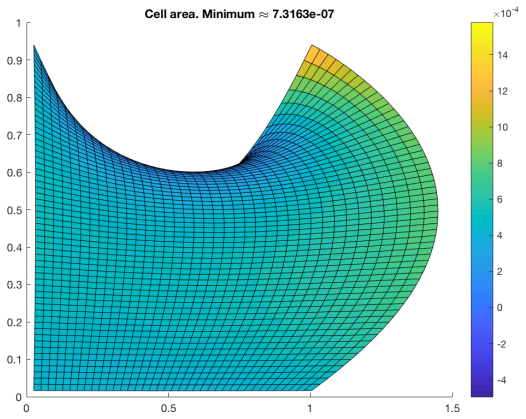


The swan domain gives different results because of its lack of convexity.









The following table summarizes the numerical results of the previous figures. MDO stands for *maximum deviation from orthogonality*, and ADO stands for *average deviation from orthogonality*. Both are a measure of the orthogonality of the grid, and are reported in degrees (not radians). The jacobian J measures the cell area, and $|J|_{min}$ is thus the area of the smallest cell in the grid, located at the physical point (x_{min}, y_{min}). Finally, k is the number of iterations required for convergence.

Domain	Grid Size	MDO	ADO	$\|J\|_{min}$	(x_{min}, y_{min})	k
Dome	(41x61)	73.5905	17.3856	0.0004	(0.02,0.02)	120
	(61x61)	74.3185	24.1013	0.0003	(0.02,0.02)	154
Swan	(41x61)	89.7265	26.2538	0.0000	(0.44,0.62)	119
	(61x61)	89.9416	25.0593	0.0000	(0.45,0.54)	145

Table 1

For the dome domain, the final grid is very nicely spaced. The cell size varies the most in the dome region, which is to be expected since the dome is the irregular part of the domain. This suggests that the dome region may be the least-accurate region for a computation, but it will certainly be much better than the initial grid. The deviation from orthogonality is fairly low, with the most problematic areas at the corners of the dome. Interestingly, adding more points in the xi-direction (more initial gridlines parallel to the x-axis) causes the curvature increase and the raises the ADO significantly. Other than that, this domain behaves very will with respect to this particular algorithm.

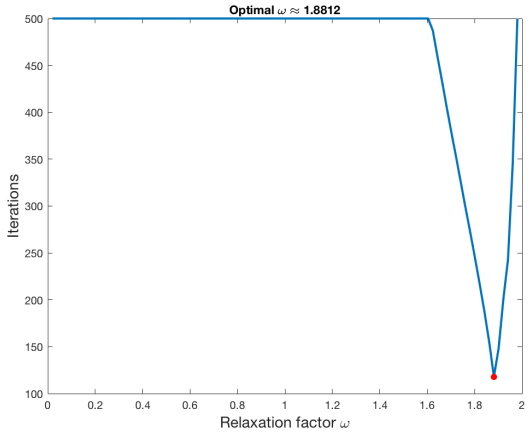
The swan domain, on the other hand, exhibits some serious problems. In the non-convex region (the top), the final grid folds over on itself and loses injectivity. Within the folding region, the cells nearly flatten out at some points, resulting in an MDO that is near 90 degrees and cells that have neglibigle area. In fact, the Jacobian matrix changes sign at the folding, indicating that nonconvexity leads to a noninvertible Jacobian, violating the convergence hypotheses of this algorithm. The resulting grid would be awful for computation! However, away from the non-convex boundary region, the grid is pretty good: the cell size is quite

uniform, and the deviation from orthogonality seems to be comparable to that of the dome domain. We may conclude that the Amsden-Hirt algorithm is not suitable for nonconvex domains.

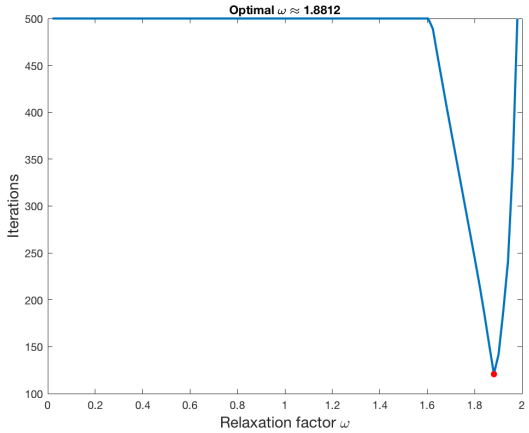
Before moving on, we examine the effect of the relaxation factor. It can be shown that the optimal ω for this algorithm for this problem is given by

$$\omega_{opt} = \frac{4}{2 + \sqrt{4 - \left(\cos(\pi/N_\xi) + \cos(\pi/N_\eta)\right)^2}} \quad (\approx 1.8775 \text{ for } N_\xi = 41,\ N_\eta = 61).$$

To confirm this claim, we run the algorithm for various values of ω and count the iterations until convergence. The results are summarized in the following plots, with iterations capped at 500. Note that the optimal ω is the same for either domain.



(a) Dome domain convergence.



(b) Swan domain convergence.

The Winslow Algorithm

The Winslow procedure claims that grid points $(\xi(x,y), \eta(x,y))$ satisfying the following Laplace equations will be smooth.

$$\Delta_{x,y}\xi = \xi_{xx} + \xi_{yy} = 0 \tag{7}$$

$$\Delta_{x,y}\eta = \eta_{xx} + \eta_{yy} = 0 \tag{8}$$

This is, in a way, the opposite of the claim made by the Amsden-Hirt procedure. Unfortunately, these equations are useless for computations because $X_{i,j}$ and $Y_{i,j}$ are the unknowns, not $\xi_{i,j}$ and $\eta_{i,j}$. We must therefore invert (7) and (8) to be in terms of x and y before we can write a useful discretization. The following theorems establish this inversion.

Theorem 1. *(Rado) Let $S : D \to D'$ be the transformation from the interior of a physical domain D to the interior of a computational domain D', given by $(x,y) \mapsto (\xi(x,y), \eta(x,y))$ and satisfying (7) and (8), with accompanying boundary transformation $\partial S : \partial D \to \partial D'$. If*

D and D' are simply connected and bounded (so ∂D and $\partial D'$ are simple closed curves), ∂S is a homeomorphism, and D' is a convex domain, then the Jacobian $J(x,y)$ of the transformation S is nonzero for all $(x,y) \in D$.

Theorem 2. *Under the assumptions of Rado's Theorem, S has an inverse operator T satisfying Winslow's quasi-linear elliptic system:*

$$\alpha x_{\xi\xi} - 2\beta x_{\xi\eta} + \gamma x_{\eta\eta} = 0 \tag{9}$$
$$\alpha y_{\xi\xi} - 2\beta y_{\xi\eta} + \gamma y_{\eta\eta} = 0, \tag{10}$$

where

$$\begin{aligned}\alpha &= x_\eta^2 + y_\eta^2 \\ \beta &= x_\xi x_\eta + y_\xi y_\eta \\ \gamma &= x_\xi^2 + y_\xi^2.\end{aligned}$$

Proof. Due to the inverse function theorem, x and y can be expressed in terms of ξ and η.

$$x = x(\xi, \eta) \tag{11}$$
$$y = y(\xi, \eta) \tag{12}$$

Differentiating (11) and (12) with respect to x, we have

$$\begin{aligned}1 &= x_\xi \xi_x + x_\eta \eta_x \\ 0 &= y_\xi \xi_x + y_\eta \eta_x,\end{aligned}$$

which can be written in matrix form as

$$\begin{bmatrix} x_\xi & x_\eta \\ y_\xi & y_\eta \end{bmatrix} \begin{bmatrix} \xi_x \\ \eta_x \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}.$$

Similarly, differentiating (11) and (12) with respect to y, we arrive at the linear system

$$\begin{bmatrix} x_\xi & x_\eta \\ y_\xi & y_\eta \end{bmatrix} \begin{bmatrix} \xi_y \\ \eta_y \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

Combining these two system, we have

$$\begin{bmatrix} x_\xi & x_\eta \\ y_\xi & y_\eta \end{bmatrix} \begin{bmatrix} \xi_x & \xi_y \\ \eta_x & \eta_y \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad \Longrightarrow \quad \begin{bmatrix} x_\xi & x_\eta \\ y_\xi & y_\eta \end{bmatrix}^{-1} = \begin{bmatrix} \xi_x & \xi_y \\ \eta_x & \eta_y \end{bmatrix}.$$

Defining the Jacobian to be $J(x,y) = x_\xi y_\eta - x_\eta y_\xi$, we have

$$\frac{1}{J(x,y)} \begin{bmatrix} y_\eta & -x_\eta \\ -y_\xi & x_\xi \end{bmatrix} = \begin{bmatrix} \xi_x & \xi_y \\ \eta_x & \eta_y \end{bmatrix}.$$

Note that $J(x,y) \neq 0$ by assumption, so that the inverse is defined. Then we have the following expressions for the first-order derivatives of ξ and η.

$$\xi_x = \frac{y_\eta}{J(x,y)} \qquad \xi_y = -\frac{x_\eta}{J(x,y)} \tag{13}$$

$$\eta_x = -\frac{y_\xi}{J(x,y)} \qquad \eta_y = \frac{x_\xi}{J(x,y)} \tag{14}$$

To get the second-order derivatives of ξ and η, we differentiate (11) and (12) twice with respect to x, then substitute for the first-order terms using (13) and (14).

$$\begin{aligned}
0 &= \left(x_{\xi\xi}\xi_x^2 + 2x_{\xi\eta}\xi_x\eta_x + x_{\eta\eta}\eta_x^2\right) + x_\xi\xi_{xx} + x_\eta\eta_{xx} \\
&= \frac{1}{J^2}\left(x_{\xi\xi}y_\eta^2 - 2x_{\xi\eta}y_\eta y_\xi + x_{\eta\eta}y_\xi^2\right) + x_\xi\xi_{xx} + x_\eta\eta_{xx} = -\frac{A}{J^2} + x_\xi\xi_{xx} + x_\eta\eta_{xx} \\
0 &= y_{\xi\xi}\xi_x^2 + 2y_{\xi\eta}\xi_x\eta_x + y_{\eta\eta}\eta_x^2 + y_\xi\xi_{xx} + y_\eta\eta_{xx} \\
&= \frac{1}{J^2}\left(y_{\xi\xi}y_\eta^2 - 2y_{\xi\eta}y_\eta y_\xi + y_{\eta\eta}y_\xi^2\right) + y_\xi\xi_{xx} + y_\eta\eta_{xx} = -\frac{B}{J^2} + y_\xi\xi_{xx} + y_\eta\eta_{xx}
\end{aligned}$$

In matrix form, this is the linear system

$$\begin{bmatrix} x_\xi & x_\eta \\ y_\xi & y_\eta \end{bmatrix} \begin{bmatrix} \xi_{xx} \\ \eta_{xx} \end{bmatrix} = \frac{1}{J^2}\begin{bmatrix} A \\ B \end{bmatrix}. \tag{15}$$

Repeating the process but differentiating with respect to y yields the linear system

$$\begin{bmatrix} x_\xi & x_\eta \\ y_\xi & y_\eta \end{bmatrix} \begin{bmatrix} \xi_{yy} \\ \eta_{yy} \end{bmatrix} = \frac{1}{J^2}\begin{bmatrix} C \\ D \end{bmatrix}, \tag{16}$$

where

$$\begin{aligned}
A &= -(x_{\xi\xi}y_\eta^2 - 2x_{\xi\eta}y_\eta y_\xi + x_{\eta\eta}y_\xi^2), \\
B &= -(y_{\xi\xi}y_\eta^2 - 2y_{\xi\eta}y_\eta y_\xi + y_{\eta\eta}y_\xi^2), \\
C &= -(x_{\xi\xi}x_\eta^2 - 2x_{\xi\eta}x_\eta x_\xi + x_{\eta\eta}x_\xi^2), \text{ and} \\
D &= -(y_{\xi\xi}x_\eta^2 - 2y_{\xi\eta}x_\eta x_\xi + y_{\eta\eta}x_\xi^2).
\end{aligned}$$

Now combining (15) and (16), we have

$$\begin{bmatrix} x_\xi & x_\eta \\ y_\xi & y_\eta \end{bmatrix} \begin{bmatrix} \xi_{xx} & \xi_{yy} \\ \eta_{xx} & \eta_{yy} \end{bmatrix} = \frac{1}{J^2}\begin{bmatrix} A & C \\ B & D \end{bmatrix} \quad \Longrightarrow \quad \begin{bmatrix} \xi_{xx} & \xi_{yy} \\ \eta_{xx} & \eta_{yy} \end{bmatrix} = \frac{1}{J^3}\begin{bmatrix} y_\eta & -x_\eta \\ -y_\xi & x_\xi \end{bmatrix} \begin{bmatrix} A & C \\ B & D \end{bmatrix}.$$

Therefore, we have the following second-order derivatives for ξ and η.

$$\begin{aligned}
\xi_{xx} &= \frac{1}{J^3}(Ay_\eta - Bx_\eta) \\
\xi_{yy} &= \frac{1}{J^3}(Cy_\eta - Dx_\eta) \\
\eta_{xx} &= \frac{1}{J^3}(Bx_\xi - Ay_\xi) \\
\eta_{yy} &= \frac{1}{J^3}(Dx_\xi - Cy_\xi).
\end{aligned}$$

Finally, we return to the original equations (7) and (8) and substitute using the previous equations.

$$0 = \xi_{xx} + \xi_{yy} = \frac{1}{J^3}((A+C)y_\eta - (B+D)x_\eta) \quad \Longrightarrow \quad 0 = (A+C)y_\eta - (B+D)x_\eta \tag{17}$$

$$0 = \eta_{xx} + \eta_{yy} = \frac{1}{J^3}((B+D)x_\xi - (A+C)y_\xi) \quad \Longrightarrow \quad 0 = (B+D)x_\xi - (A+C)y_\xi \tag{18}$$

Multiplying (17) by x_ξ and (18) by x_η, then adding the result, gives the following.

$$\begin{aligned} 0 &= (A+C)(x_\xi y_\eta - x_\eta y_\xi) + (B+D)(x_\xi x_\eta - x_\eta x_\xi) \\ &= (A+C)J(x,y) \quad \Longrightarrow \quad -(A+C) = 0 \end{aligned} \tag{19}$$

Similarly, multiplying (17) by y_ξ and (18) by y_η, then adding the result, we have the following.

$$\begin{aligned} 0 &= (A+C)(y_\eta y_\xi - y_\xi y_\eta) + (B+D)(x_\xi y_\eta - y_\xi x_\eta) \\ &= (B+D)J(x,y) \quad \Longrightarrow \quad -(B+D) = 0 \end{aligned} \tag{20}$$

Finally, we expand out (19) and (20).

$$\begin{aligned} 0 = -(A+C) &= x_{\xi\xi}y_\eta^2 - 2x_{\xi\eta}y_\eta y_\xi + x_{\eta\eta}y_\xi^2 + x_{\xi\xi}x_\eta^2 - 2x_{\xi\eta}x_\eta x_\xi + x_{\eta\eta}x_\xi^2 \\ &= (x_\eta^2 + y_\eta^2)x_{\xi\xi} - 2(x_\xi x_\eta + y_\xi y_\eta)x_{\xi\eta} + (x_\xi^2 + y_\xi^2)x_{\eta\eta} \\ &= \alpha x_{\xi\xi} - 2\beta x_{\xi\eta} + \gamma x_{\eta\eta} \\ 0 = -(B+D) &= y_{\xi\xi}y_\eta^2 - 2y_{\xi\eta}y_\eta y_\xi + y_{\eta\eta}y_\xi^2 + y_{\xi\xi}x_\eta^2 - 2y_{\xi\eta}x_\eta x_\xi + y_{\eta\eta}x_\xi^2 \\ &= (x_\eta^2 + y_\eta^2)y_{\xi\xi} - 2(x_\xi x_\eta + y_\xi y_\eta)y_{\xi\eta} + (x_\xi^2 + y_\xi^2)y_{\eta\eta} \\ &= \alpha y_{\xi\xi} - 2\beta y_{\xi\eta} + \gamma y_{\eta\eta} \end{aligned}$$

Thus

$$\begin{aligned} 0 &= \alpha x_{\xi\xi} - 2\beta x_{\xi\eta} + \gamma x_{\eta\eta} \\ 0 &= \alpha y_{\xi\xi} - 2\beta y_{\xi\eta} + \gamma y_{\eta\eta}, \end{aligned}$$

where

$$\begin{aligned} \alpha &= x_\eta^2 + y_\eta^2 \\ \beta &= x_\xi x_\eta + y_\xi y_\eta \\ \gamma &= x_\xi^2 + y_\xi^2, \end{aligned}$$

as desired. □

Discretization of Winslow's Equations

To carry out Winslow's algorithm, we discretize Winslow's equations with second-order centered difference quotients. To begin, we examine (9). Then $x_{\xi\xi}$ and $x_{\eta\eta}$ terms can be

discretized in the same way that they were for the Amsden-Hirt algorithm, but the cross term $x_{\xi\eta}$ must be computed a little differently.

$$\begin{aligned}(x_{\xi\eta})_{i,j} &= \frac{1}{2\Delta\eta}((x_\xi)_{i,j+1} - (x_\xi)_{i,j-1}) \\ &= \frac{1}{2\Delta\eta}\left(\frac{1}{2\Delta\xi}(X_{i+1,j+1} - X_{i-1,j+1}) - \frac{1}{2\Delta\xi}(X_{i+1,j-1} - X_{i-1,j-1})\right) \\ &= \frac{1}{4}(X_{i+1,j+1} - X_{i-1,j+1} - X_{i+1,j-1} - X_{i-1,j-1})\end{aligned}$$

The full discretized equation is thus

$$\begin{aligned}0 &= \alpha(X_{i-1,j} - 2X_{i,j} + X_{i+1,j}) \\ &\quad - \frac{\beta}{2}(X_{i+1,j+1} - X_{i-1,j+1} - X_{i+1,j-1} - X_{i-1,j-1}) \\ &\quad + \gamma(X_{i,j-1} - 2X_{i,j} + X_{i,j+1}) \\ 2(\alpha+\gamma)X_{i,j} &= \alpha(X_{i-1,j} + X_{i+1,j}) \\ &\quad - \frac{\beta}{2}(X_{i+1,j+1} - X_{i-1,j+1} - X_{i+1,j-1} - X_{i-1,j-1}) \\ &\quad + \gamma(X_{i,j-1} + X_{i,j+1})\end{aligned}$$

To use SOR, we again use a relaxation factor ω and use any already computed values when looping over j and i.

$$X_{i,j}^{(k)} = \omega\big[\frac{1}{2(\alpha_{i,j}+\gamma_{i,j})}\alpha_{i,j}(X_{i-1,j}^{(k)} + X_{i+1,j}^{(k-1)}) - 2\beta_{i,j}(X_{i+1,j+1}^{(k-1)} - X_{i-1,j+1}^{(k-1)} - X_{i+1,j-1}^{(k)} - X_{i-1,j-1}^{(k)}) + \gamma_{i,j}(X_{i,j-1}^{(k)} + X_{i,j+1}^{(k-1)})\big] + (1-\omega)X_{i,j}^{(k-1)}$$

$$\alpha_{i,j} = (\frac{X_\eta}{2})^2 + (\frac{Y_\eta}{2})^2 = \frac{1}{4}\left((X_{i,j+1}^{(k-1)} - X_{i,j-1}^{(k)})^2 + (Y_{i,j+1}^{(k-1)} - Y_{i,j-1}^{(k)})^2\right)$$

$$\beta_{i,j} = (\frac{X_\xi}{2})(\frac{X_\eta}{2}) + (\frac{Y_\xi}{2})(\frac{Y_\eta}{2}) = \frac{1}{4}\big((X_{i+1,j}^{(k-1)} - X_{i-1,j}^{(k)})(X_{i,j+1}^{(k-1)} - X_{i,j-1}^{(k)}) + (Y_{i,j+1}^{(k-1)} - Y_{i,j-1}^{(k)})(Y_{i+1,j}^{(k-1)} - Y_{i-1,j}^{(k)})\big)$$

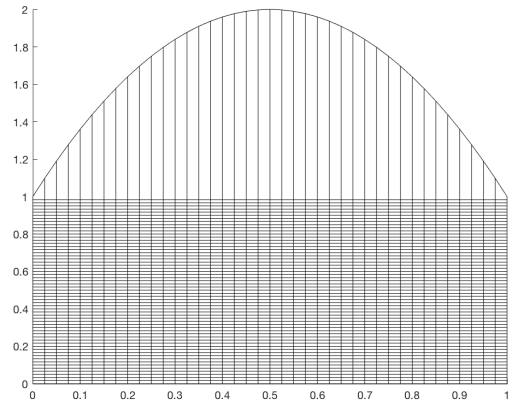
$$\gamma_{i,j} = (\frac{X_\xi}{2})^2 + (\frac{Y_\xi}{2})^2 = \frac{1}{4}\left((X_{i+1,j}^{(k-1)} - X_{i-1,j}^{(k)})^2 + (Y_{i+1,j}^{(k-1)} - Y_{i-1,j}^{(k)})^2\right)$$

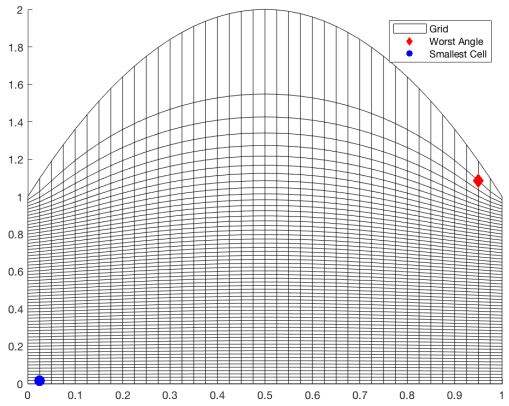
The equations are similar for y. Below, we include an implementation of the algorithm for MATLAB.

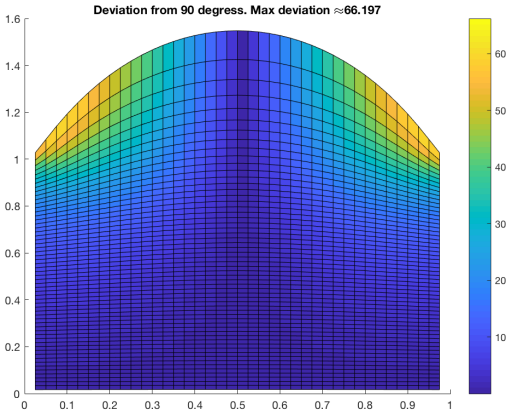
```
function gridgenWinslowSC(X, Y, w, tol, maxiters)

[Nx,Ny] = size(X);
for k=1:maxiters                % SOR ITERATION
    X0 = X;
    Y0 = Y;
    for j=2:Ny-1
        for i=2:Nx-1
            % First order centered differentials.
            X_xi  = X0(i+1,j) - X(i-1,j);
            X_eta = X0(i,j+1) - X(i,j-1);
            Y_xi  = Y0(i+1,j) - Y(i-1,j);
            Y_eta = Y0(i,j+1) - Y(i,j-1);

            % Nonlinear coefficients for quasilinear system.
            a = (X_eta^2 + Y_eta^2) / 4;
            b = (X_xi*X_eta + Y_xi*Y_eta) / 4;
            g = (X_xi^2 + Y_xi^2) / 4;

            % Main computation.
            X(i,j) = w*(a*(X0(i+1,j)+X(i-1,j)) + g*(X0(i,j+1)+X(i,j-1)) ...
                    -.5*b*(X0(i+1,j+1)-X(i+1,j-1)-X0(i-1,j+1)+X(i-1,j-1)) ...
                    )/(2*(a+g)) + (1-w)*X0(i,j);
            Y(i,j) = w*(a*(Y0(i+1,j)+Y(i-1,j)) + g*(Y0(i,j+1)+Y(i,j-1)) ...
                    - .5*b*(Y0(i+1,j+1)-Y(i+1,j-1)-Y0(i-1,j+1)+Y(i-1,j-1)) ...
                    )/(2*(a+g)) + (1-w)*Y0(i,j);
        end
    end
    if norm(X-X0, Inf) < tol && norm(Y-Y0, Inf) < tol
        break               % Check for convergence.
    end
end
end
```

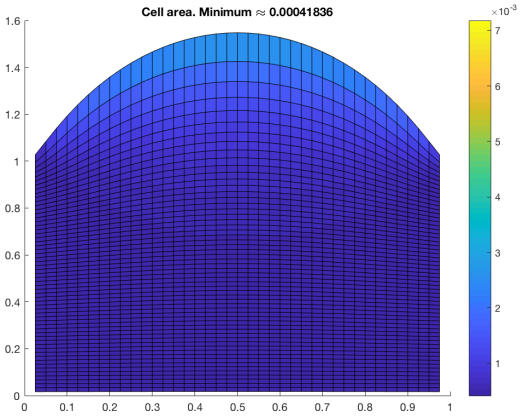
Results

We begin with the initial and final grids for the dome domain, and display the same grid quality metrics. The final grid is slightly different that that of the Amsden-Hirt algorithm.

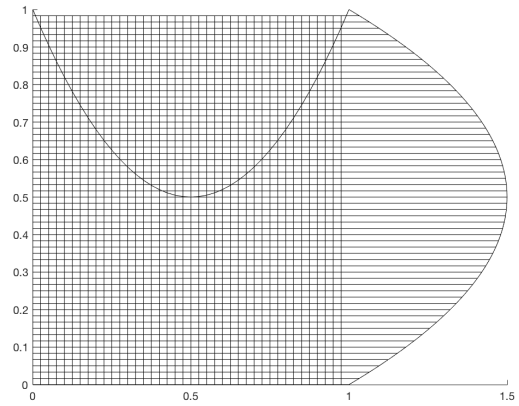


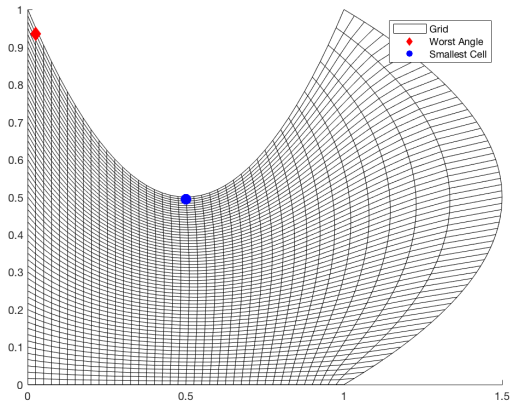


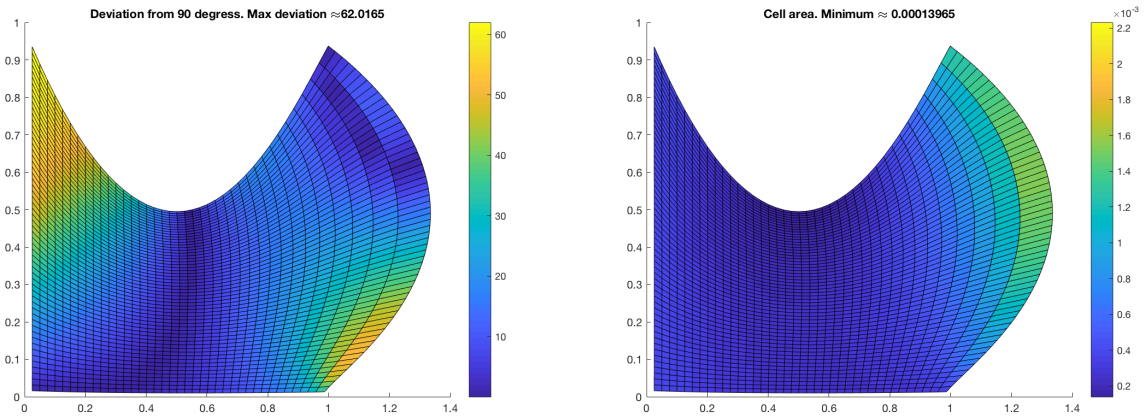




The swan domain has less problems this time around.





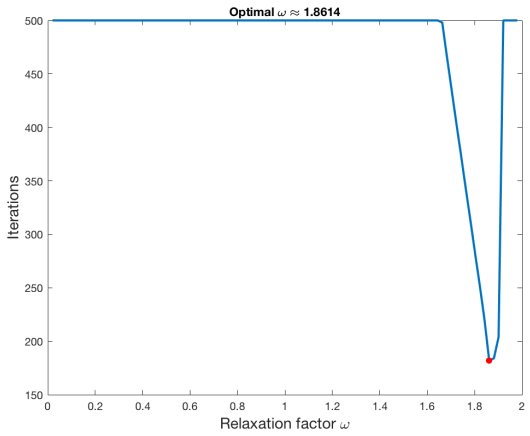


Domain	Grid Size	MDO	ADO	$\|J\|_{min}$	(x_{min}, y_{min})	k
Dome	(41x61)	66.1970	9.2467	0.0004	(0.02,0.02)	170
	(61x61)	66.4995	9.2666	0.0003	(0.98,0.02)	289
Swan	(41x61)	62.0165	17.6436	0.0001	(0.50,0.49)	154
	(61x61)	62.4366	17.8045	0.0001	(0.48,0.50)	230

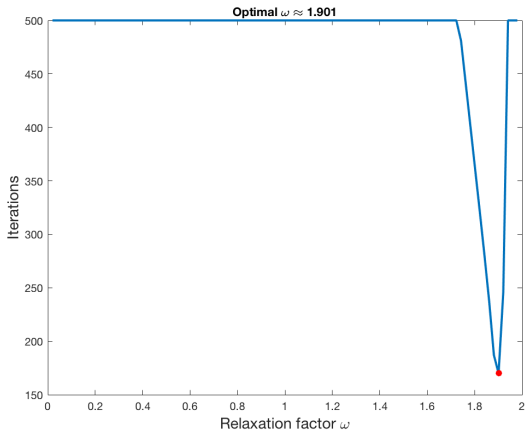
Table 2

The results for this algorithm differ from our previous results in a few important ways. The Amsden-Hirt algorithm seemed to be good at generating grids with fairly uniform cell sizes (except in non-convex regions), but wasn't as good at preserving orthogonality. The Winslow algorithm seems to do the opposite: the resulting grids have very low deviations from orthogonality, but the cell sizes are much less uniform (especially for the top section of the dome domain). However, the Winslow algorithm succeeds in generating a one-to-one grid over the swan domain, which is a significant improvement over Amsden-Hirt. The Jacobian of the final grid never changes sign, which confirms the absence of folding. In addition, Winslow is slightly more expensive than Amsden-Hirt, both in number of iterations needed for convergence and in FLOPS per loop. We conclude that, in general, it is better to use the Winslow algorithm for nonconvex domains (to avoid folding), but Amsden-Hirt is probably a better choice of algorithm for convex domains.

Another problem with the Winslow algorithm is that we cannot establish an optimal choice for ω analytically. However, we can still do empirical tests to make an informed choice. In the following figures, note that the optimal ω is **not** the same for both domains.



(a) Dome domain convergence.



(b) Swan domain convergence.

Multiply-connected Domains

As we have seen, the Winslow algorithm is a good choice for generating grids over non-convex domains. We conclude the project by applying the Winslow algorithm to *multiply-connected domains*, domains that are no longer simply connected because of a hole. In this case we generate the grid using polar coordinates, constructing the grids so that the ξ direction corresponds to the angle and the η direction corresponds to the radius.

The only difference in the algorithm is that continuity must be enforced at $\theta = 0$ and $\theta = 2\pi$, which means we must handle the special case of $i = 1$ a little differently.

```
function gridgenWinslowMC(X, Y, w, tol, maxiters)

[Nx,Ny] = size(X);
for k=1:maxiters                % SOR ITERATION
    X0 = X;
    Y0 = Y;
    for j=2:Ny-1                % Iterate radius
        % Branch cut case: i==1
        I = Nx-1;
        X_xi  = X0(2,j) - X(I,j);
        X_eta = X0(1,j+1) - X(1,j-1);
        Y_xi  = Y0(2,j) - Y(I,j);
        Y_eta = Y0(1,j+1) - Y(1,j-1);

        a = (X_eta^2 + Y_eta^2) / 4;
        b = (X_xi*X_eta + Y_xi*Y_eta) / 4;
        g = (X_xi^2 + Y_xi^2) / 4;

        X(1,j) = w*(a*(X0(2,j)+X(I,j)) + g*(X0(1,j+1)+X(1,j-1)) ...
                - .5*b*(X0(2,j+1)-X(2,j-1)-X0(I,j+1)+X(I,j-1)) ...
                )/(2*(a+g)) + (1-w)*X0(1,j);
        Y(1,j) = w*(a*(Y0(2,j)+Y(I,j)) + g*(Y0(1,j+1)+Y(1,j-1)) ...
```

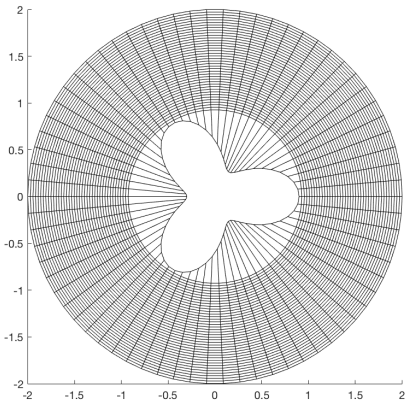
```
                - .5*b*(Y0(2,j+1)-Y(2,j-1)-Y0(I,j+1)+Y(I,j-1)) ...
                )/(2*(a+g)) + (1-w)*Y0(1,j);

        % Non-branch cut cases.
        for i=2:Nx-1            % Iterate angle
            % First order centered differentials.
            X_xi  = X0(i+1,j) - X(i-1,j);
            X_eta = X0(i,j+1) - X(i,j-1);
            Y_xi  = Y0(i+1,j) - Y(i-1,j);
            Y_eta = Y0(i,j+1) - Y(i,j-1);

            % Nonlinear coefficients for quasilinear system.
            a = (X_eta^2 + Y_eta^2) / 4;
            b = (X_xi*X_eta + Y_xi*Y_eta) / 4;
            g = (X_xi^2 + Y_xi^2) / 4;

            % Main computation.
            X(i,j) = w*(a*(X0(i+1,j)+X(i-1,j)) + g*(X0(i,j+1)+X(i,j-1)) ...
                    -.5*b*(X0(i+1,j+1)-X(i+1,j-1)-X0(i-1,j+1)+X(i-1,j-1)) ...
                    )/(2*(a+g)) + (1-w)*X0(i,j);
            Y(i,j) = w*(a*(Y0(i+1,j)+Y(i-1,j)) + g*(Y0(i,j+1)+Y(i,j-1)) ...
                    - .5*b*(Y0(i+1,j+1)-Y(i+1,j-1)-Y0(i-1,j+1)+Y(i-1,j-1)) ...
                    )/(2*(a+g)) + (1-w)*Y0(i,j);
        end
    end
    % Enforce continuity
    X(end,:) = X(1,:);
    Y(end,:) = Y(1,:);
end
end
```

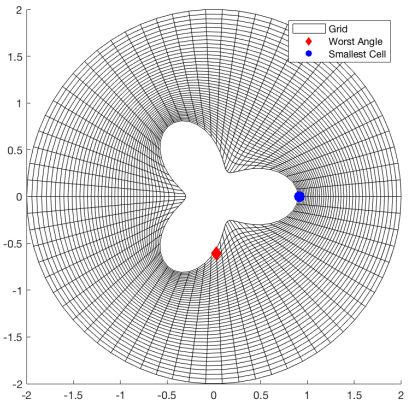
Results

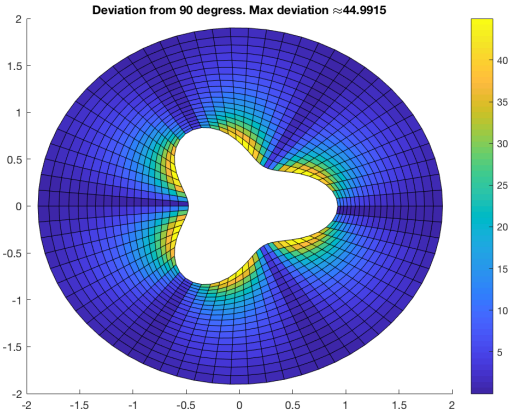
The *three-leafed rose domain* is the region bounded between the parametric curve

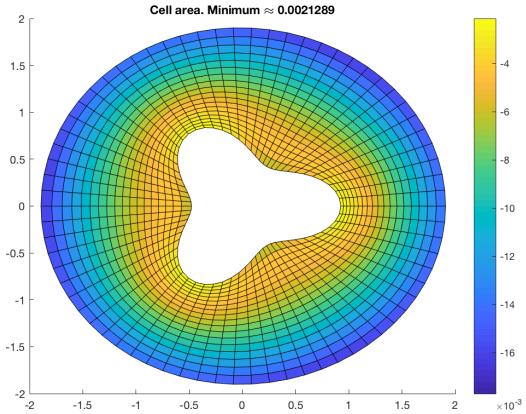
$$x(\theta) = 0.3(2 + \cos(3\theta))\cos(\theta), \qquad y(\theta) = 0.3(2 + \cos(3\theta))\sin(\theta), \qquad \theta \in [0, 2\pi]$$

and the outer circle $x^2 + y^2 = 2^2$. As before, we plot the initial and final grids, together with colorplots for grid orthogonality and cell size. The following plots use 71 divisions in the θ direction and 41 divisions in the radial direction.





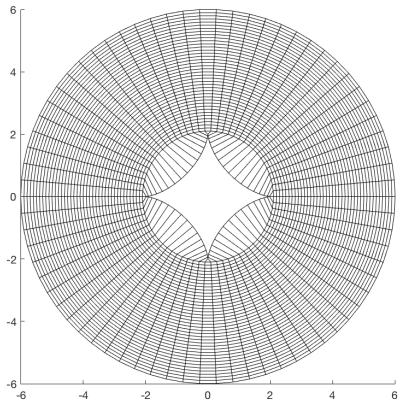


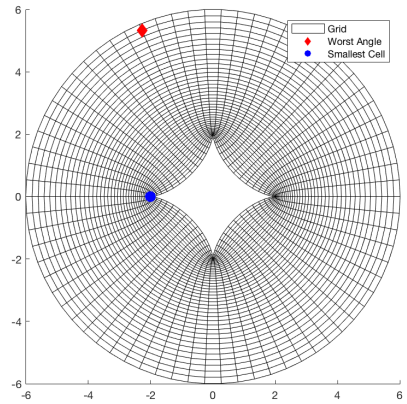


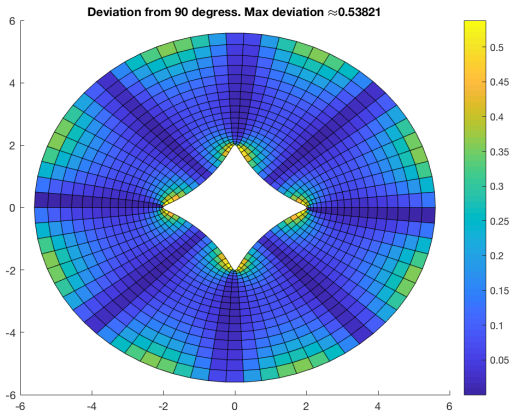
The *astroid domain* is the region bounded by the parametric curve

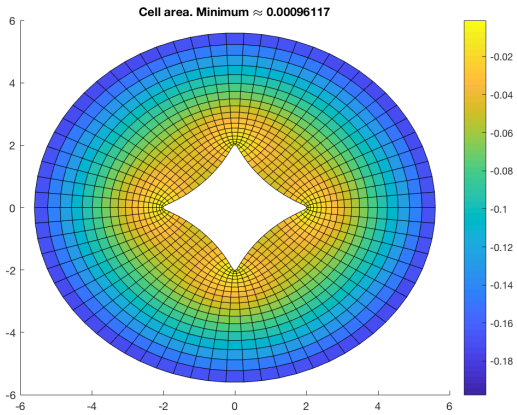
$$x(\theta) = \frac{1}{2}(3\cos(\theta) + \cos(3\theta)), \qquad y(\theta) = \frac{1}{2}(3\sin(\theta) - \sin(3\theta)), \qquad \theta \in [0, 2\pi]$$

and the outer circle $x^2 + y^2 = 6^2$. The results for this region are also quite good.







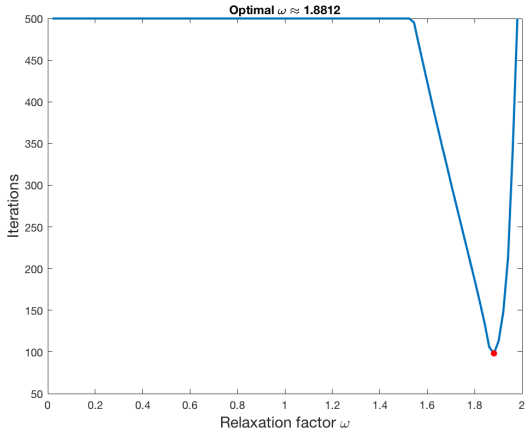


Domain	Grid Size	MDO	ADO	$\|J\|_{min}$	(x_{min}, y_{min})	k
Rose	(71x21)	44.9915	9.2716	0.0021	(0.92,0.00)	58
	(71x41)	50.2706	9.4681	0.0010	(0.91,0.00)	90
Astroid	(71x21)	0.5382	0.1314	0.0010	(-2.01,0.00)	153
	(71x41)	0.5287	0.1173	0.0001	(-2.00,0.00)	172

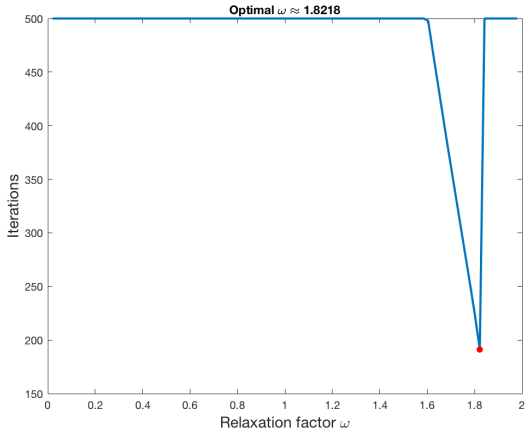
Table 3

As we saw from the simply-connected domains, the main strength of the Winslow algorithm over the Amsden-Hirt approach is smaller deviation from orthogonality and no folding outside of a non-convex domain. The results here are very similar: the deviation from orthogonality is low—**extremely** low for the astroid region—even at the inner boundaries. On the other hand, the cell sizes shrink significantly at the inner boundaries, but the grids still maintain their injectivity and don't fold over the boundary. This is confirmed by the fact that the Jacobain never switches sign.

As before, we don't have an analytic method for choosing ω, but the following tests locate good choices for each domain.



(a) Dome domain convergence.



(b) Swan domain convergence.

Amsden-Hirt for Multiply-connected Domains

Finally, it is worth noting that the Amdsen-Hirt algorithm can be modified to work for polar-based domains like the rose and astroid domains.

```
function gridgenAH(X, Y, w, tol, maxiters)

[Nx,Ny] = size(X);
for k=1:maxiters                % SOR ITERATION
    X0 = X;
    Y0 = Y;
    for j=2:Ny-1
        I = Nx-1;               % Branch cut
        X(1,j) = w*(X(I,j) + X0(2,j) + X(1,j-1) + X0(1,j+1))/4 + (1-w)*X0(1,j);
        Y(1,j) = w*(Y(I,j) + Y0(2,j) + Y(1,j-1) + Y0(1,j+1))/4 + (1-w)*Y0(1,j);
        for i=2:Nx-1            % Other terms
            X(i,j) = w*(X(i-1,j) + X0(i+1,j) + X(i,j-1) + X0(i,j+1))/4 + (1-w)*X0(i,j↩
                );
            Y(i,j) = w*(Y(i-1,j) + Y0(i+1,j) + Y(i,j-1) + Y0(i,j+1))/4 + (1-w)*Y0(i,j↩
                );
        end
    end
    % Enforce continuity
    X(end,:) = X(1,:);
    Y(end,:) = Y(1,:);
    if norm(X-X0, Inf) < tol && norm(Y-Y0, Inf) < tol
        break               % Convergence criterion
    end
end
end
```

However, as we might expect, the final grid completely loses injectivity for domains that are not simply connected, as shown in the following figures.