Lecture 23. Grids and Clustering

So far we have been discussing solving PDEs using fixed step sizes in the discretisation process. However some problems may well be on non-uniform boundaries, such as shown in left hand plots in Figure 23.1. Defining a uniform grid in such situations clearly is not possible. Under such scenarios a coordinate mapping is usually undertaken to transform the physical domain of interest into a more tractable computational domain, on which the numerical discretisation can be performed with ease. Thus a general transformation

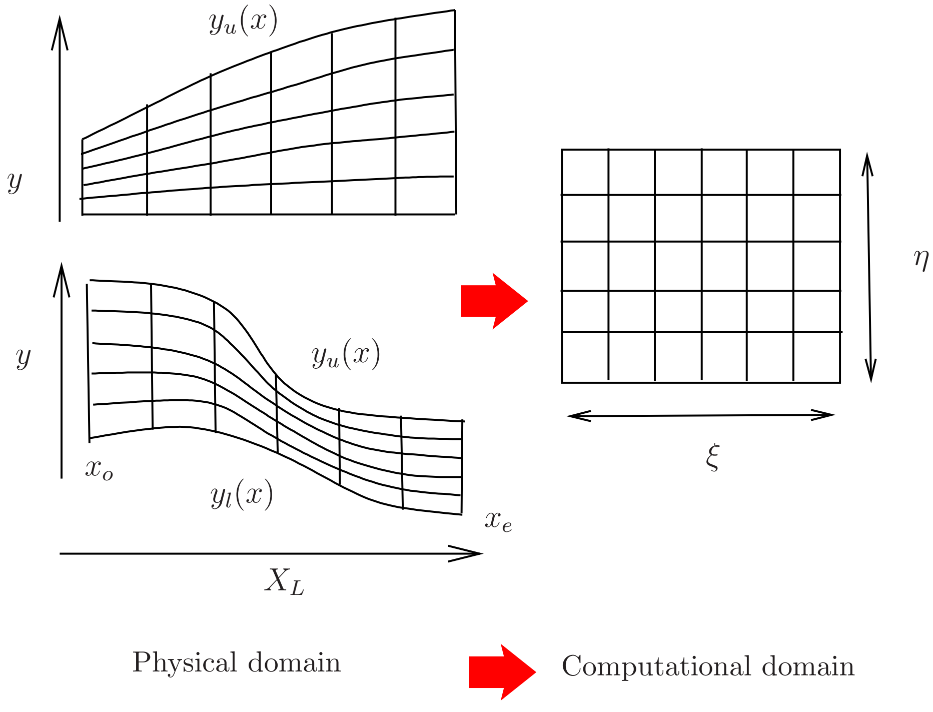

Figure 23.1: Mapping from physical plane to computational grid.

(mapping) of the physical coordinates (x, y) to computational plane (ξ, η), is one technique to ease the solution process. In Figure 23.1, one possible mapping could be

$$\xi = \frac{x - x_o}{x_e - x_o}, \qquad \eta = \frac{y - y_l(x)}{y_u(x) - y_l(x)}; \tag{23.1}$$

thus mapping $x_o \leq x \leq x_u$ to $0 \leq \xi \leq 1$, and $y_l(x) \leq y \leq y_u(x)$ to $0 \leq \eta \leq 1$. The numerical operations and solution process is thus undertaken in the (ξ, η)-coordinates, on transforming the PDEs derivatives from the (x, y)-coordinates into the (ξ, η)-coordinates. Thus first derivatives will be transformed as follows:

$$\frac{\partial}{\partial x} = \frac{\partial \xi}{\partial x}\frac{\partial}{\partial \xi} + \frac{\partial \eta}{\partial x}\frac{\partial}{\partial \eta}; \qquad \frac{\partial}{\partial y} = \frac{\partial \xi}{\partial y}\frac{\partial}{\partial \xi} + \frac{\partial \eta}{\partial y}\frac{\partial}{\partial \eta}; \tag{23.2}$$

and the second and mixed derivatives as follows:

$$\frac{\partial^2}{\partial x^2} = \left(\frac{\partial \xi}{\partial x}\frac{\partial}{\partial \xi} + \frac{\partial \eta}{\partial x}\frac{\partial}{\partial \eta}\right)\left(\frac{\partial \xi}{\partial x}\frac{\partial}{\partial \xi} + \frac{\partial \eta}{\partial x}\frac{\partial}{\partial \eta}\right), \tag{23.3}$$

$$\frac{\partial^2}{\partial y^2} = \left(\frac{\partial \xi}{\partial y}\frac{\partial}{\partial \xi} + \frac{\partial \eta}{\partial y}\frac{\partial}{\partial \eta}\right)\left(\frac{\partial \xi}{\partial y}\frac{\partial}{\partial \xi} + \frac{\partial \eta}{\partial y}\frac{\partial}{\partial \eta}\right), \tag{23.4}$$

$$\frac{\partial^2}{\partial x \partial y} = \left(\frac{\partial \xi}{\partial x}\frac{\partial}{\partial \xi} + \frac{\partial \eta}{\partial x}\frac{\partial}{\partial \eta}\right)\left(\frac{\partial \xi}{\partial y}\frac{\partial}{\partial \xi} + \frac{\partial \eta}{\partial y}\frac{\partial}{\partial \eta}\right). \tag{23.5}$$

As can be seen from Figure 23.1, in the transformed grid, one can then make use of fixed discretisation step sizes $(\Delta\xi, \Delta\eta)$ in the transformed PDEs.

Alternative techniques to work in the physical plane are available, but we do not consider these in this course – (Google, finite elements and unstructured grids). Examples of quite complex problems are shown in Figure 23.2, these as you will appreciate are quite advanced.

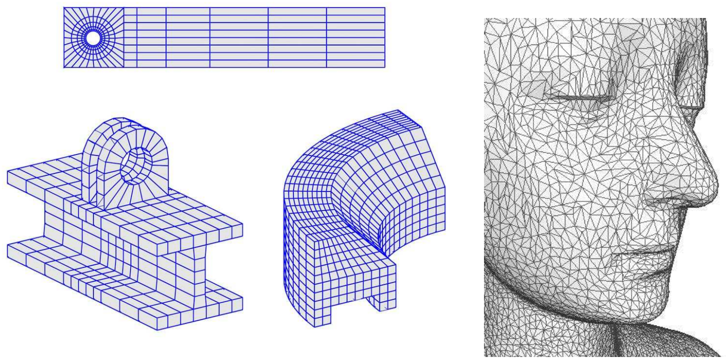

Figure 23.2: a: Finite element and unstructured grids in complex geometry (see: http://www.featool.com/tutorial/2015/08/24/creating-structured-grids-using-featool-matlab-functions. http://www.sci.utah.edu/the-institute/highlights/24-research-highlights/cibc-highlights/439-cleaver.html)

Grid Stretching

We illustrate the need for grid stretching by considering the ordinary differential equation:

$$\epsilon\frac{d^2u}{dx^2} - \frac{du}{dx} = f(x), \quad \text{satisfying} \;\; u(0) = \alpha, \;\; u(1) = \beta, \tag{23.6}$$

where ϵ is some parameter, and we set $\alpha = 1, \; \beta = 3$. Observe that as $\epsilon \to 0$, the equation reduces to $-u' = f(x)$, and thus both boundary conditions can not be satisfied – a more interesting situation develops when we consider ϵ very small but not identically equal to zero. An exact solution to this equation is given by

$$u(x) = \alpha + x + (\beta - \alpha - 1)\left(\frac{e^{x/\epsilon} - 1}{e^{1/\epsilon} - 1}\right), \tag{23.7}$$

and solutions for varying ϵ with $f(x) = -1$ are shown in Figure 23.3. Observe that as $\epsilon \to 0$, $u(x)$ at the $x = 1$ boundary develops a very localised and rapidly changing solution. This region is known as a *boundary-layer* with thickness of $O(\epsilon)$ (see LeVeque, chapter 2.17).

We next attempt a numerical solution with finite differences (second-order accurate), for the case $\epsilon = 0.01$, and use 11 equi-spaced grid points to discretise (23.6). The solution is shown in Figure 23.4 and we also compare the numerical result with the exact result (given by the blue curve). We see some significant differences between the two results,

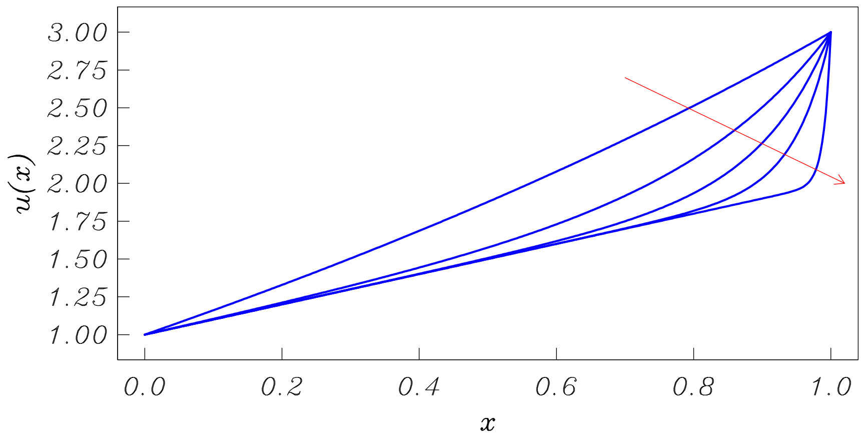

Figure 23.3: Exact solutions of (23.6), with $\epsilon = (1.0,\ 0.2,\ 0.1,\ 0.05,\ 0.01)$, arrow in direction of decreasing ϵ.

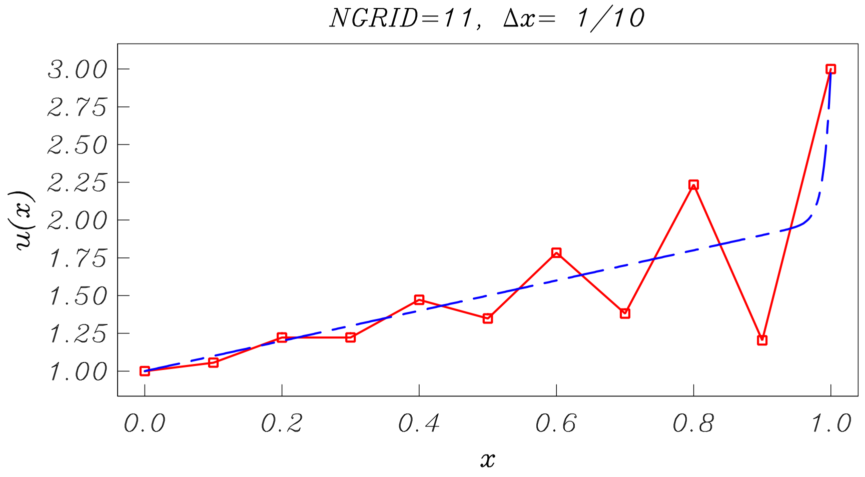

Figure 23.4: Numerical solution of (23.6), with $\epsilon = 0.01$), with $N = 11$ grid points, $\Delta x = 1/(N-1)$. Exact solution is given by the blue dashed line.

with the numerical result also displaying ”wiggles”. On using successively finer grids, the numerical solution eventually agrees with the exact result, as shown in Figure’s 23.5 & 23.6.

The above results thus indicate that to obtain a reasonably accurate result we need to have a number of points within the ”boundary-layer” region. In the final result shown in Figure 23.6, note that for a large part of the solution, i.e. $x < 0.9$ a nearly linear behaviour arises in the solution, so in this region, ideally we should require less density of grid points to resolve the solution structure there, whereas in the ”boundary-layer” region $x \sim> 0.95$ it is clear many more points are needed to resolve the solution there if we were to use a uniform grid.

In more practical larger-scale problems (such as in two- and three-dimensions) exhibiting such behaviour, it would clearly be nice if one could some how cluster, or have more points only in regions where rapid variations of the solution are known to arise – thus reducing the number of grid points overall, and thus speeding up (hopefully) the solution process. This is the idea behind ”grid-stretching” (or mapping function). We discuss this aspect in the context of a 1-dimensional problem, i.e. (23.6), but the basic ideas will be applicable to more dimensions, albeit, using slightly more complicated expressions of the so-called mapping functions. This is best illustrated through an example.

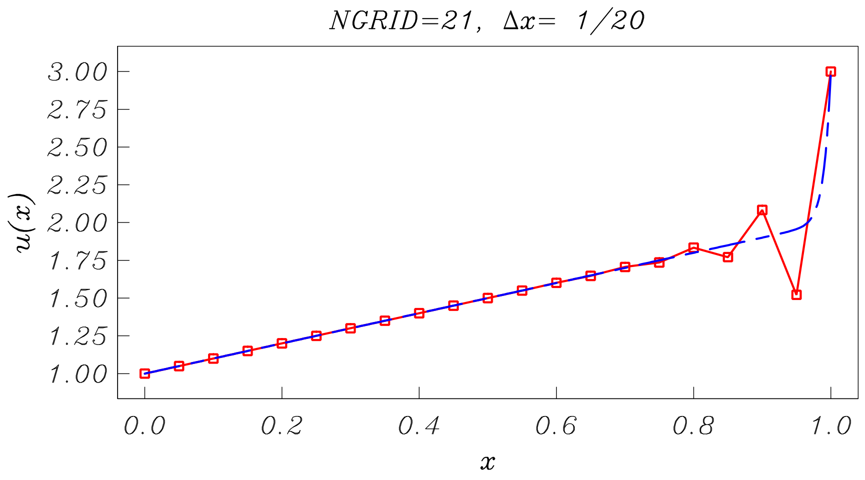

Figure 23.5: Numerical solution of (23.6), with $\epsilon = 0.01$), with $N = 21$ grid points, $\Delta x = 1/(N-1)$. Exact solution is given by the blue dashed line.

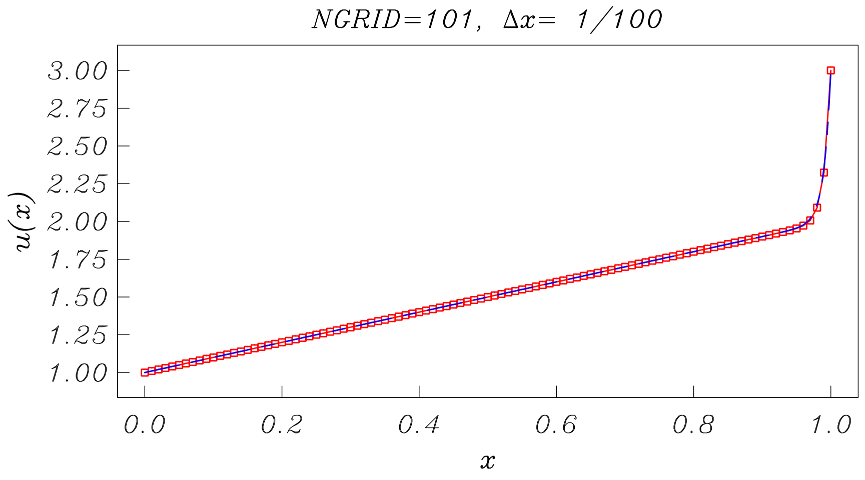

Figure 23.6: Numerical solution of (23.6), with $\epsilon = 0.01$), with $N = 101$ grid points, $\Delta x = 1/(N-1)$. Exact solution is given by the blue dashed line.

We start with a uniform grid in our "computational" Z-frame, and then use a grid mapping function to define the associated grid points in the physical x-reference frame. There are many means of defining such functions (see LeVeque), but we choose to use the following map:

$$x = c_2 + \frac{1}{c_1}\tan(\lambda\,[Z - \eta_o]), \tag{23.8}$$

where

$$\eta_o = \frac{z-1}{z+1}, \quad z = \frac{\tan^{-1}(c_1(1+c_2))}{\tan^{-1}(c_1(1-c_2))}, \quad \lambda = \frac{\tan^{-1}\left[c_1(1-c_2)\right]}{1-\eta_o}. \tag{23.9}$$

This maps $-1 \le Z \le 1$ to $0 \le x \le 1$. Some examples are shown in Figures 23.7- 23.9 – here we have applied a further linear uniform mapping whereby

$$\xi = (Z+1)/2 \tag{23.10}$$

to thus map $-1 \le Z \le 1$ to $0 \le \xi \le 1$.

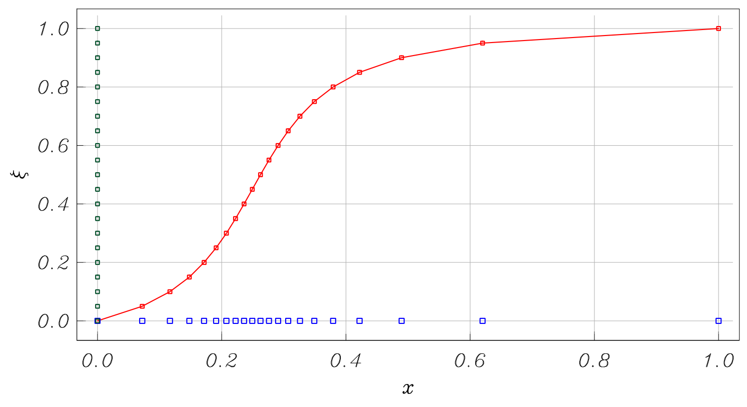

Figure 23.7: Grid mapping with $c_1 = -5$, and $c_2 = 0.25$. The Δx step variations are given by the blue symbols, while note the $\Delta\xi$ spacing is uniform and given in green symbols along the vertical ξ axis. In this mapping the points are most clustered at $x = 0.25$

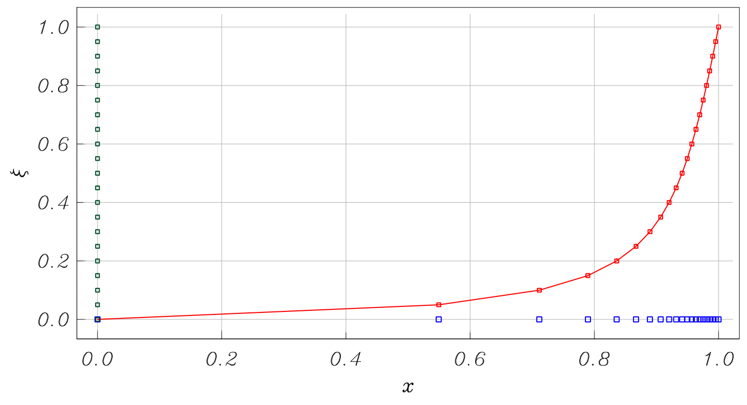

Figure 23.8: Grid mapping with $c_1 = -8$, and $c_2 = 1$. The Δx step variations are given by the blue symbols, while note the $\Delta\xi$ spacing is uniform and given in green symbols along the vertical ξ axis. In this mapping the points are most clustered at $x = 1.0$. In this plot the mapping and clustering is quite extreme.

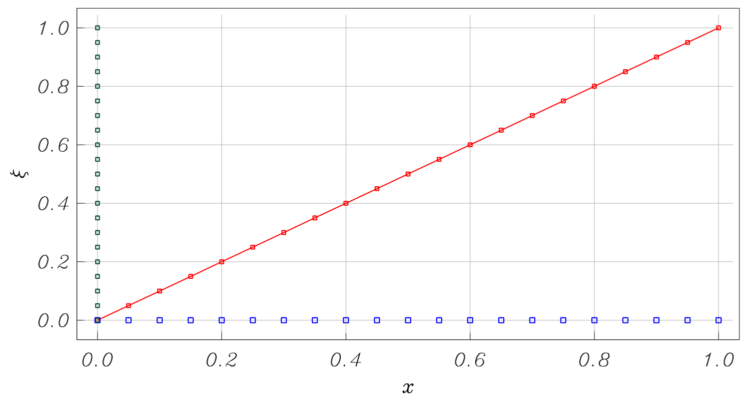

Figure 23.9: Grid mapping with $c_1 = -1 \times 10^{-22}$, and $c_2 = 0$. The Δx step variations are given by the blue symbols, while note the $\Delta\xi$ spacing is uniform and given in green symbols along the vertical ξ axis. In this mapping the points are uniformly distributed with a 1-to-1 correspondence.

Hence it therefore is easy to see, for example how such a mapping (still quite simple, and also not very flexible) could be applied to a 2D problem, as indicated in Figure 23.10.

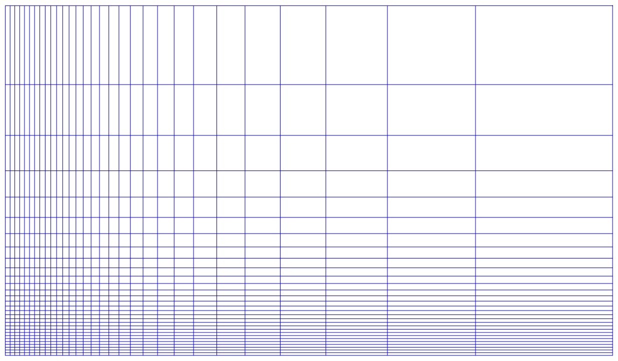

Figure 23.10: Example of 2D Grid clustering at the lower and left boundaries in the physical plane.

In all of the above, again the PDEs in the physical plane require to be transformed into the computational coordinates followed by appropriate discretisations undertaken in the computational coordinates. In this case, Expressions (23.2)–(23.5) thus require to be utilised and operated upon using expressions such as (23.8) and (23.10).

Adaptive Stencils

A somewhat different approach is taken by a variety of methods that adapt the stencil as the boundary is encountered (**see Problem sheet 2, Question 6**). One of the earliest techniques is to modify the weights of the discretisation stencil as soon as the stencil crosses the boundary.

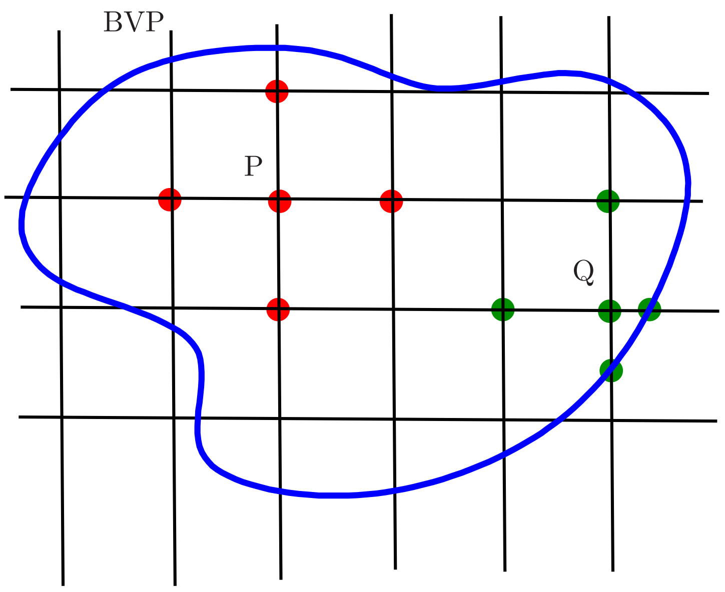

Figure 23.11: Irregular boundary without mappings.

The general technique is straightforward: assume that we wish to discretise the Poisson equation using the standard five-point stencil. For each point on the grid where all five points fall within the computational domain we get

$$\frac{1}{(\Delta x)^2}(u_{i+1,j}+u_{i-1,j})+\frac{1}{(\Delta y)^2}(u_{i,j+1}+u_{i,j-1})-(\frac{2}{(\Delta x)^2}+\frac{2}{(\Delta x)^2})u_{i,j}=f_{i,j}. \qquad (23.11)$$

This discretisation may be used, as is, if all the grid points fall within the computational domain, as the P-data point in Figure 23.11. As one or more points cross the boundary/interface, such as point Q in Figure 23.11 modifications have to made. In this case, we use the general non-equidistant grid formulae. Even though this scheme is rather straightforward to implement and thus quite popular, it has the disadvantage of disrupting the uniform stencil structure of the elliptic operator, and requires considerable care to implement, and cater for due to the many scenarios possible at the boundary.

Various other techniques, based on mappings, reflections and interpolations are common, but in all cases, both the right-hand side and the discretisation stencil have to be modified to accommodate the presence of the boundary.

Elliptic Schemes

Elliptic PDES have the property that solutions are generally very smooth. The smoothness property is an advantage, and for this reason Laplace and Poisson Equations are a very good choice. With Poisson grid generators the mapping is obtained by specifying the desired grid points (x, y) say, on the boundary of the physical domain with the interior point distribution determined through solution of the equations

$$\xi_{xx}+\xi_{yy}=\mathcal{P}(\xi,\eta), \qquad \eta_{xx}+\eta_{yy}=\mathcal{Q}(\xi,\eta), \qquad (23.12)$$

with (ξ, η) representing the coordinates of the computational grid, with $(\mathcal{P}, \mathcal{Q})$ used to control point spacings within the grid interior. The above are then transformed to computational space by changing the roles of the independent and dependent variables, such that

$$\left.\begin{aligned}\alpha x_{\xi\xi}-2\beta x_{\xi\eta}+\gamma x_{\eta\eta}&=J^2(\mathcal{P}x_\xi+\mathcal{Q}x_\eta)\\ \alpha y_{\xi\xi}-2\beta y_{\xi\eta}+\gamma y_{\eta\eta}&=J^2(\mathcal{P}y_\xi+\mathcal{Q}y_\eta)\end{aligned}\right\}, \qquad (23.13)$$

where

$$\left.\begin{aligned}\alpha&=x_\eta^2+y_\eta^2\\ \beta&=x_\xi x_\eta+y_\xi y_\eta\\ \gamma&=x_\xi^2+y_\xi^2\\ J&=x_\xi y_\eta-x_\eta y_\xi\end{aligned}\right\}, \qquad (23.14)$$

These are solved on a uniformly spaced computational grid, and thus provides the (x, y) grid points in the physical field. Usually simple Dirichlet boundary conditions suffice.The advantage is that the resulting grid is smooth and complex boundaries are easily accommodated. Prescribing or choosing $(\mathcal{P}, \mathcal{Q})$ though can prove time consuming, since grid point control within the grid interior is more difficult to control.

A typical example of what is though achievable is shown in Figure 23.13. Here with reference to Figure 23.12, a cut in the physical plane is introduced along o–a, with a requirement that the solution along o–a must be identical to the solution along b–c. Surface

boundary conditions (inviscid flow tangency condition) along a–b is thus mapped to the lower a–b boundary in the figure on the right. The flow far-field conditions are thus imposed along the upper boundary (o-g-f-e-d-c). The flow field equations (Laplace Equation) along with the equations (23.12) for the grid mappings are both solved using Successive Over Relaxation (SOR) discussed in earlier lectures.
(See Tannehill, Anderson & Pletcher, *Computational Fluid Mechanics and Heat Transfer*).

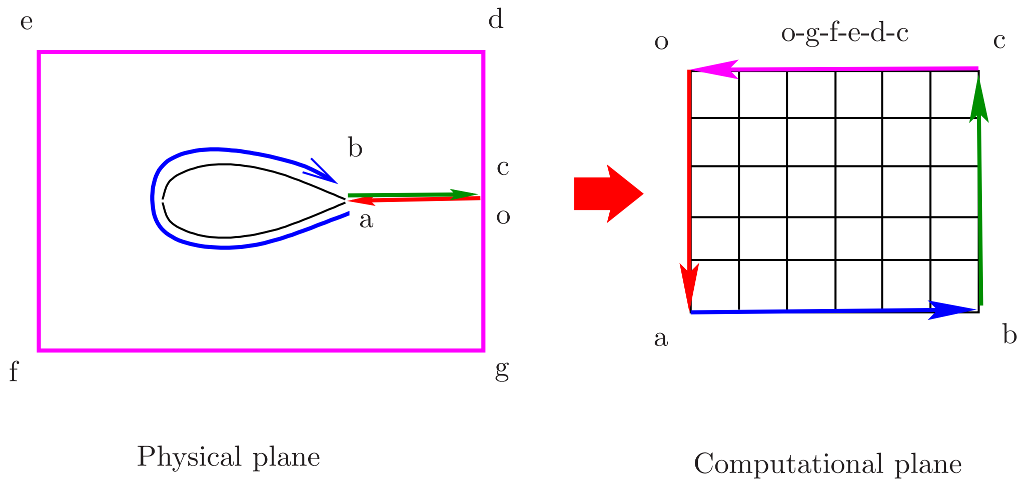

Figure 23.12: Example of complex boundary using Elliptic mapping.

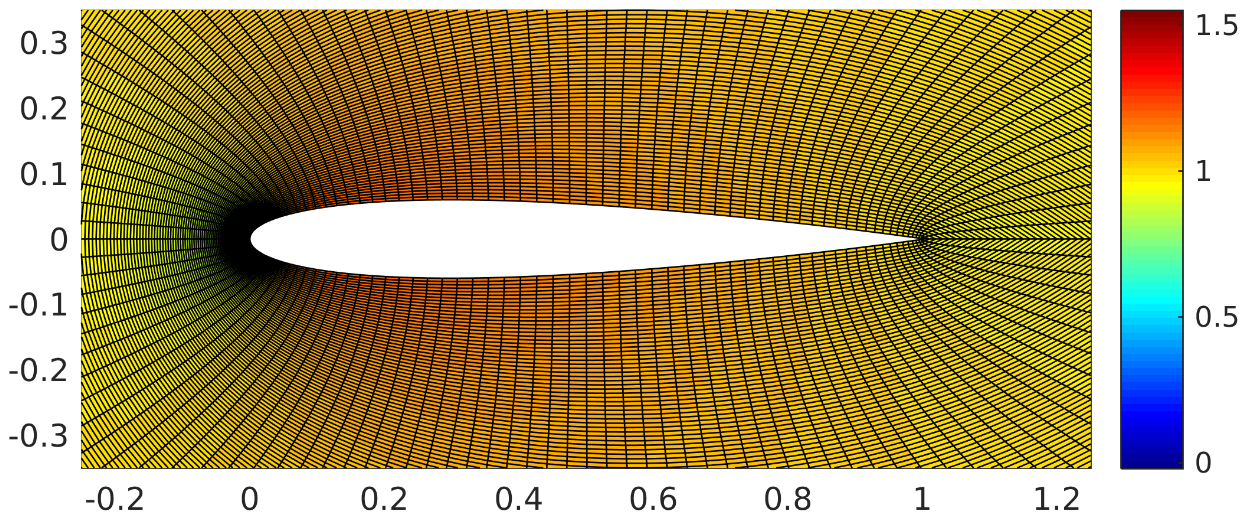

Figure 23.13: Examples of grid generated using (23.13). The strategy used is shown in Figure 23.12.