14 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 Fig. 14.1. 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 (mapping) of the physical coordinates (x, y) to computational plane (ξ, η), is one technique to ease the solution process. In Fig. 14.1, one possible mapping could be

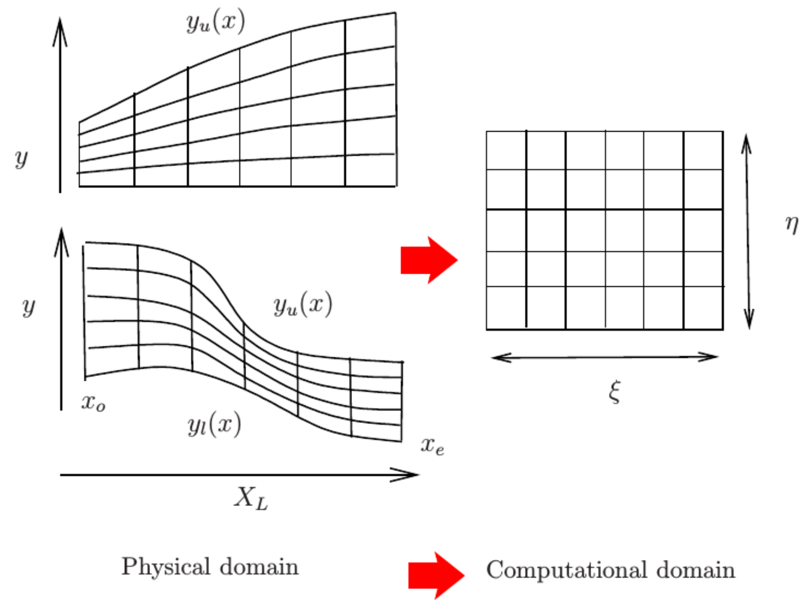

Figure 14.1: Mapping from physical plane to computational grid.

$$\xi = \frac{x - x_o}{x_e - x_o}, \qquad \eta = \frac{y - y_l(x)}{y_u(x) - y_l(x)} \tag{14.1}$$

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:

$$\frac{\partial}{\partial x} = \frac{\partial \xi}{\partial x}\frac{\partial}{\partial \xi} + \frac{\partial \eta}{\partial x}\frac{\partial}{\partial \eta}; \qquad \frac{\partial}{\partial y} = \frac{\partial \xi}{\partial y}\frac{\partial}{\partial \xi} + \frac{\partial \eta}{\partial y}\frac{\partial}{\partial \eta} \tag{14.2}$$

and the second and mixed derivatives as follows:

$$\frac{\partial^2}{\partial x^2} = \left(\frac{\partial \xi}{\partial x}\frac{\partial}{\partial \xi} + \frac{\partial \eta}{\partial x}\frac{\partial}{\partial \eta}\right)\left(\frac{\partial \xi}{\partial x}\frac{\partial}{\partial \xi} + \frac{\partial \eta}{\partial x}\frac{\partial}{\partial \eta}\right) \tag{14.3}$$

$$\frac{\partial^2}{\partial y^2} = \left(\frac{\partial \xi}{\partial y}\frac{\partial}{\partial \xi} + \frac{\partial \eta}{\partial y}\frac{\partial}{\partial \eta}\right)\left(\frac{\partial \xi}{\partial y}\frac{\partial}{\partial \xi} + \frac{\partial \eta}{\partial y}\frac{\partial}{\partial \eta}\right), \tag{14.4}$$

$$\frac{\partial^2}{\partial x \partial y} = \left(\frac{\partial \xi}{\partial x}\frac{\partial}{\partial \xi} + \frac{\partial \eta}{\partial x}\frac{\partial}{\partial \eta}\right)\left(\frac{\partial \xi}{\partial y}\frac{\partial}{\partial \xi} + \frac{\partial \eta}{\partial y}\frac{\partial}{\partial \eta}\right) \tag{14.5}$$

As can be seen from Fig. 14.1, in the transformed grid, one can then make use of fixed discretisation step sizes $(\Delta\xi, \Delta\eta)$ in the transformed PDEs.

A difficulty above, in the more general case where $\xi = \xi(x, y)$ together with $\eta = \eta(x, y)$, is the evaluation of the $\xi_x,\ \eta_y$ etc. metric terms due to the **now variable** $(x,\ y)$ grid spacings. This is circumvented by observing, that if we take partial derivatives of $\xi(x, y)$ with respect to ξ and η we get

$$\begin{aligned} \xi_x\, x_\xi + \xi_y\, y_\xi &= 1 \\ \xi_x\, x_\eta + \xi_y\, y_\eta &= 0. \end{aligned} \tag{14.6}$$

If we then do the same with $\eta(x, y)$ with respect to ξ and η we get

$$\begin{aligned} \eta_x\, x_\xi + \eta_y\, y_\xi &= 0 \\ \eta_x\, x_\eta + \eta_y\, y_\eta &= 1. \end{aligned} \tag{14.7}$$

Hence the following results arise

$$\begin{aligned} \xi_x &= y_\eta / J \\ \xi_y &= -x_\eta / J \\ \eta_x &= -y_\xi / J \\ \eta_y &= x_\xi / J. \end{aligned} \tag{14.8}$$

Here J is the Jacobian given by

$$J = x_\xi y_\eta - x_\eta y_\xi. \tag{14.9}$$

The higher derivative terms such as $\xi_{xx},\ \xi_{xy},\ \eta_{yy}$ etc. follow by further differentiation and manipulation of expressions (14.6)–(14.9).

14.1 Grid Stretching, an ODE example

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

$$\epsilon \frac{d^2u}{dx^2} - \frac{du}{dx} = f(x), \text{ satisfying } u(0) = \alpha, \quad u(1) = \beta \tag{14.10}$$

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

$$u(x) = \alpha + x + (\beta - \alpha - 1)\left(\frac{e^{x/\epsilon} - 1}{e^{1/\epsilon} - 1}\right) \tag{14.11}$$

and solutions for varying ϵ with $f(x) = -1$ are shown in Fig. 14.2. 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, §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 Eqn. 14.10. The solution is shown in Fig. 14.3 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, with the numerical result also displaying "wiggles". On using successively finer grids, the numerical solution eventually agrees with the exact result, as shown in Figs. 14.4 & 14.5.

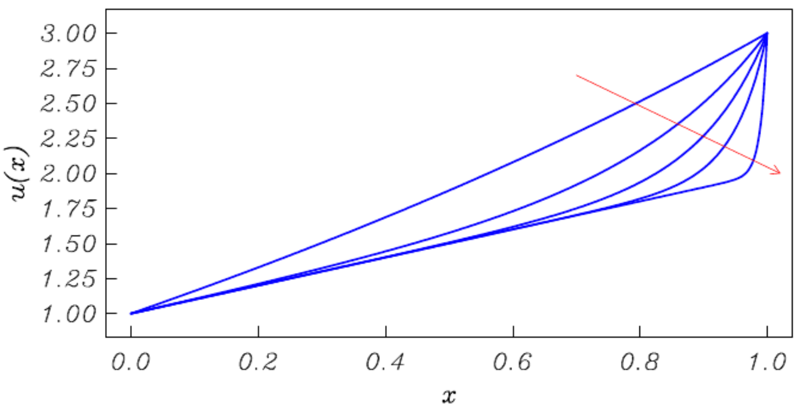

Figure 14.2: Exact solutions of Eqn. 14.10, with $\epsilon = (1.0, 0.2, 0.1, 0.05, 0.01)$, arrow in direction of decreasing ϵ

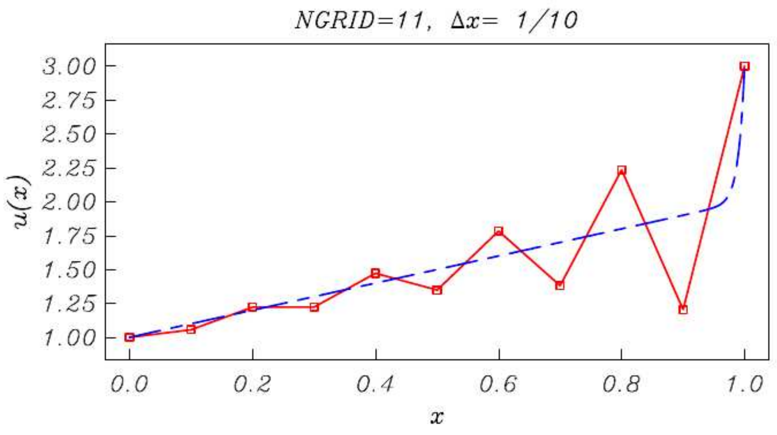

Figure 14.3: Numerical solution of Eqn. 14.10, with $\epsilon = 0.01$, with $N = 11$ grid points, $\Delta x = 1/(N-1)$. Exact solution is given by the blue dashed line.

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 Fig. 14.3, 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.* Eqn. 14.10, 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.

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

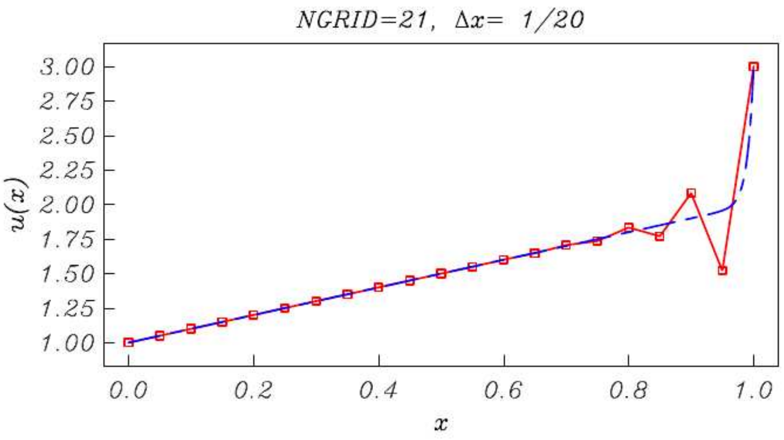

Figure 14.4: Numerical solution of Eqn. 14.10, with $\epsilon = 0.01$, with $N = 21$ grid points, $\Delta x = 1/(N-1)$. Exact solution is given by the blue dashed line.

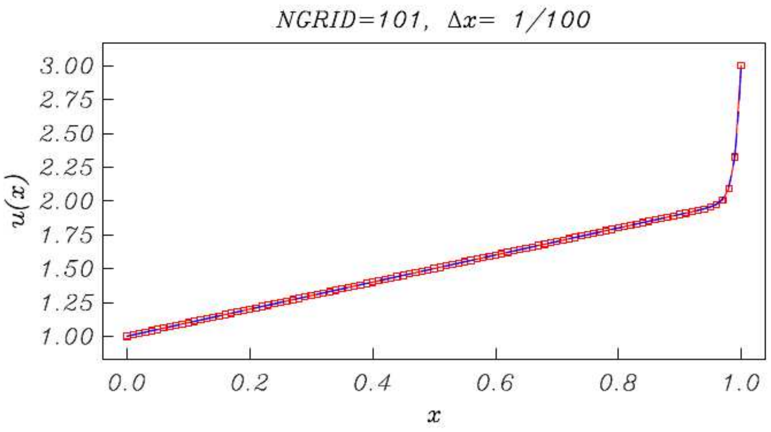

Figure 14.5: Numerical solution of Eqn. 14.10, with $\epsilon = 0.01$, with $N = 101$ grid points, $\Delta x = 1/(N-1)$. Exact solution is given by the blue dashed line.

map:

$$x = c_2 + \frac{1}{c_1}\tan\left(\lambda\left[Z - \eta_o\right]\right) \tag{14.12}$$

where

$$\eta_o = \frac{z-1}{z+1}, \quad z = \frac{\tan^{-1}\left(c_1\left(1+c_2\right)\right)}{\tan^{-1}\left(c_1\left(1-c_2\right)\right)}, \quad \lambda = \frac{\tan^{-1}\left[c_1\left(1-c_2\right)\right]}{1-\eta_o}. \tag{14.13}$$

This maps $-1 \leq Z \leq 1$ to $0 \leq x \leq 1$. Some examples are shown in Figs. 14.6–14.8; here we have applied a further linear uniform mapping whereby

$$\xi = (Z+1)/2 \tag{14.14}$$

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

Hence it 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 Fig. 14.9.

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, Eqns. 14.2-14.5 thus require to be utilised and operated upon using expressions such as (14.12) and (14.14).

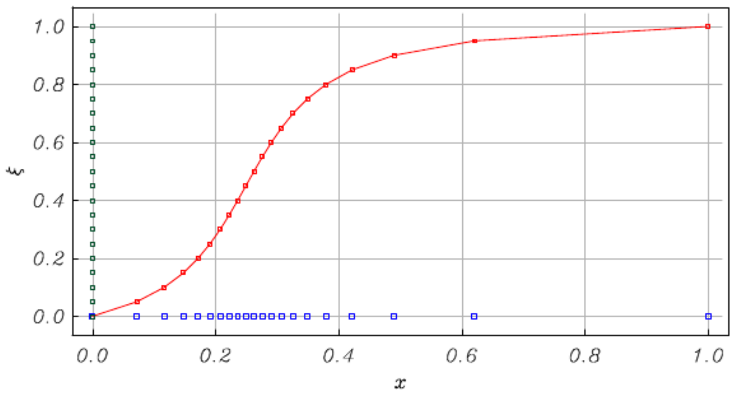

Figure 14.6: Grid mapping with $c_1 = -5$, and $c_2 = 0.25$. The Δx step variations are given by the blue symbols, while note the $\Delta\xi$ spacing is uniform and given in green symbols along the vertical ξ axis. In this mapping the points are most clustered at $x = 0.25$

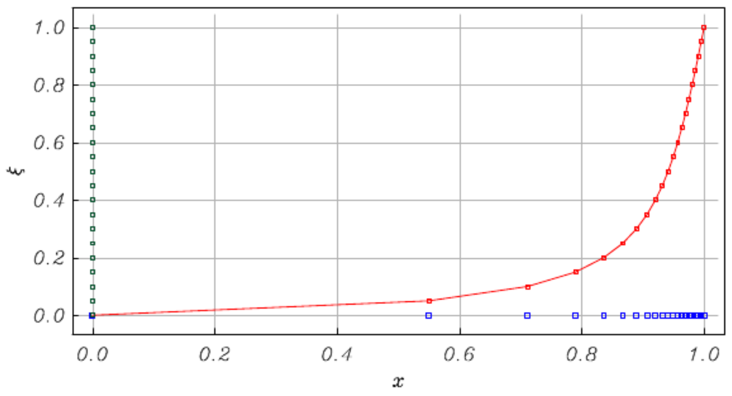

Figure 14.7: Grid mapping with $c_1 = -8$, and $c_2 = 1$. The Δx step variations are given by the blue symbols, while note the $\Delta\xi$ spacing is uniform and given in green symbols along the vertical ξ axis. In this mapping the points are most clustered at $x = 1.0$. In this plot the mapping and clustering is quite extreme.

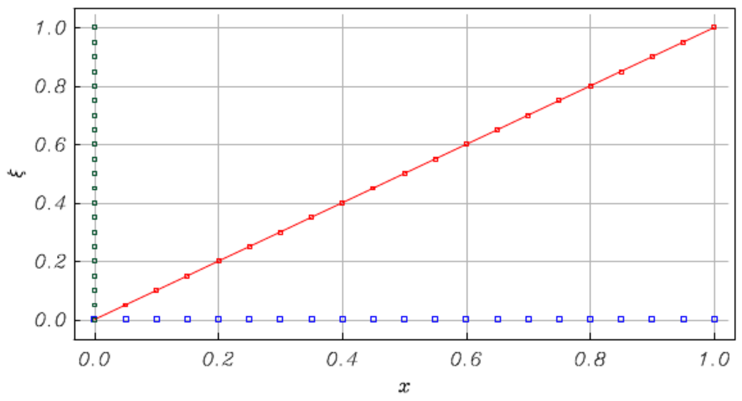

Figure 14.8: Grid mapping with $c_1 = -1 \times 10^{-22}$, and $c_2 = 0$. The Δx step variations are given by the blue symbols, while note the $\Delta\xi$ spacing is uniform and given in green symbols along the vertical ξ axis. In this mapping the points are uniformly distributed with a 1-to-1 correspondence.

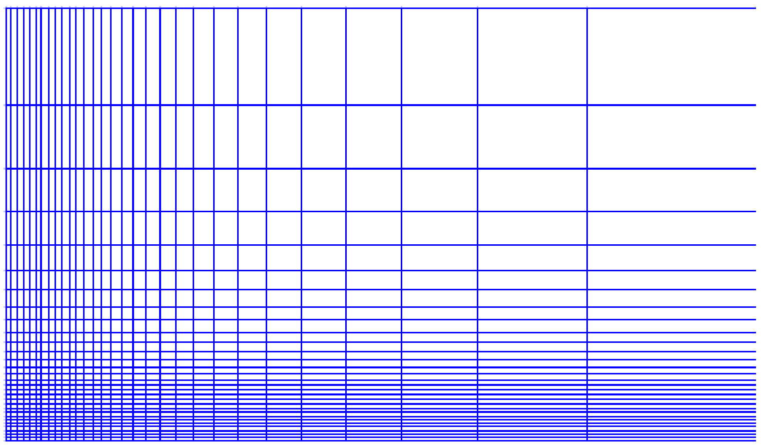

Figure 14.9: Example of 2D Grid clustering at the lower and left boundaries in the physical plane.

14.2 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.

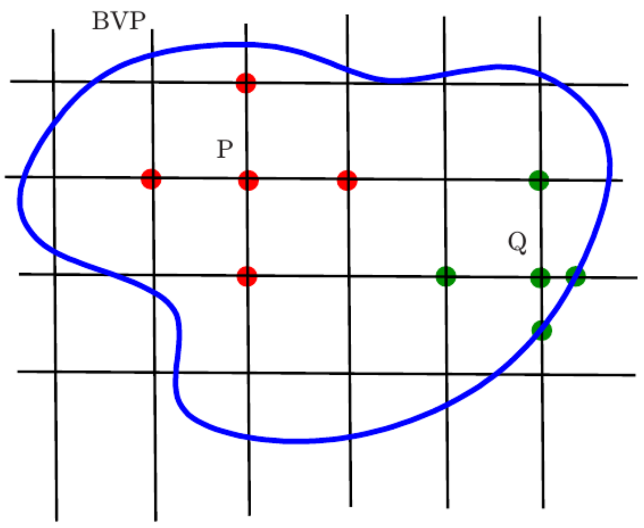

Figure 14.10: Irregular boundary without mappings

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

$$\frac{1}{(\Delta x)^2}\left(u_{i+1,j}+u_{i-1,j}\right)+\frac{1}{(\Delta y)^2}\left(u_{i,j+1}+u_{i,j-1}\right)-\left(\frac{2}{(\Delta x)^2}+\frac{2}{(\Delta x)^2}\right)u_{i,j}=f_{i,j}. \tag{14.15}$$

This discretisation may be used, as is, if all the grid points fall within the computational domain, as the P-data point in Fig. 14.10. As one or more points cross the boundary/interface, such as point Q in Fig. 14.10 (also see Fig. 14.11) 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.

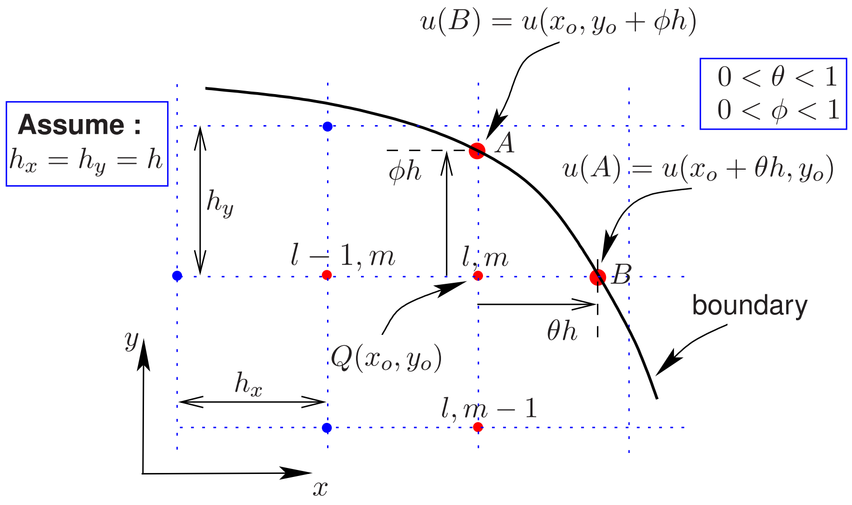

Figure 14.11: Non-uniform stencil at boundary treatment.

The nonuniform grid formulae are derived by usage of Taylor series expansions. With reference to Fig. 14.11 where for ease of discussion we assume uniform grid-spacings $h_x = h_y = h$, suppose we required to discretise Laplace's equation, namely the u_{xx}, u_{yy} terms at the point Q. Taylor expansions thus give

$$u(x_o + \theta h, y_o) = u(x_o, y_o) + \theta h \frac{\partial u}{\partial x}|_{l,m} + \frac{(\theta h)^2}{2}\frac{\partial^2 u}{\partial x^2}|_{l,m} + \frac{(\theta h)^3}{3!}\frac{\partial^3 u}{\partial x^3}|_{l,m} + \dots . \tag{14.16}$$

Similarly

$$u(x_o - h, y_o) = u(x_o, y_o) - h\frac{\partial u}{\partial x}|_{l,m} + \frac{h^2}{2}\frac{\partial^2 u}{\partial x^2}|_{l,m} - \frac{h^3}{3!}\frac{\partial^3 u}{\partial x^3}|_{l,m} + \dots . \tag{14.17}$$

Elimination of the $(\partial u/\partial x)|_{l,m}$ term gives

$$\theta u(x_o - h, y_o) + u(x_o + \theta h, y_o) = (1+\theta)u(x_o, y_o) + \frac{\theta h^2}{2}(1+\theta)\frac{\partial^2 u}{\partial x^2}|_{l,m} + O(h^3), \tag{14.18}$$

or (on truncation of the $O(h^3)$ term),

$$\theta U_{l-1,m} + U(A) = (1+\theta)U_{l,m} + \frac{\theta h^2}{2}(1+\theta)\frac{\partial^2 u}{\partial x^2}|_{l,m}. \tag{14.19}$$

Hence,

$$\frac{\partial^2 u}{\partial x^2}|_{l,m} = \frac{2\theta U_{l-1,m} + 2U(A) - 2(1+\theta)U_{l,m}}{\theta h^2(1+\theta)}. \tag{14.20}$$

Similar treatment in the y-direction results in

$$\frac{\partial^2 u}{\partial y^2}|_{l,m} = \frac{2\phi U_{l,m-1} + 2U(B) - 2(1+\phi)U_{l,m}}{\phi h^2(1+\phi)}. \tag{14.21}$$

Thus if one was discretising Poisson's equation $\nabla^2 u = f(x,y)$ (say) it follows, the discretised form at the boundary would be

$$\frac{2\theta U_{l-1,m} + 2U(A)}{\theta(1+\theta)} + \frac{2\phi U_{l,m-1} + 2U(B)}{\phi(1+\phi)} - 2U_{l,m}\left(\frac{1}{\theta} + \frac{1}{\phi}\right) = h^2 f_{l,m}. \tag{14.22}$$

Observe that the truncation error term indicates that this would though only be $O(h)$ accurate; $O(h^2)$ accuracy is recovered on setting both $\phi = \theta = 1$ in the grid interior!

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.

14.3 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

$$\xi_{xx} + \xi_{yy} = \mathcal{P}(\xi,\eta), \quad \eta_{xx} + \eta_{yy} = \mathcal{Q}(\xi,\eta) \tag{14.23}$$

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

$$\left.\begin{array}{ll} \alpha x_{\xi\xi} - 2\beta x_{\xi\eta} + \gamma x_{\eta\eta} & = J^2\left(\mathcal{P}x_\xi + \mathcal{Q}x_\eta\right) \\ \alpha y_{\xi\xi} - 2\beta y_{\xi\eta} + \gamma y_{\eta\eta} & = J^2\left(\mathcal{P}y_\xi + \mathcal{Q}y_\eta\right) \end{array}\right\}, \tag{14.24}$$

where

$$\left.\begin{array}{ll} \alpha & = x_\eta^2 + y_\eta^2 \\ \beta & = x_\xi x_\eta + y_\xi y_\eta \\ \gamma & = x_\xi^2 + y_\xi^2 \\ J & = x_\xi y_\eta - x_\eta y_\xi \end{array}\right\}. \tag{14.25}$$

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 achieve.

A typical example of what is though achievable is shown in Fig. 14.13. Here with reference to Fig. 14.12, 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 $\mathrm{a-b}$ is thus mapped to the lower $\mathrm{a-b}$ 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 (transformed Laplace Equation, say as an example) along with Eqns. (14.24) for the grid mappings are both solved using Successive Over Relaxation (SOR) discussed in earlier lectures.

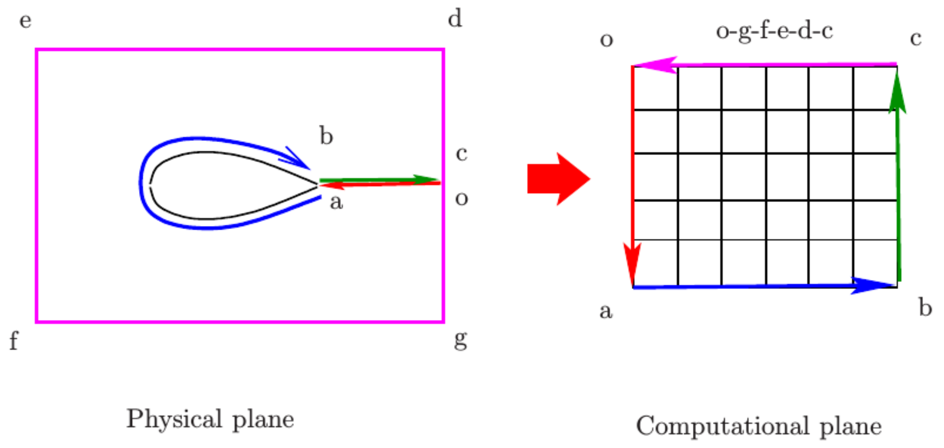

Figure 14.12: Example of complex boundary using Elliptic mapping.

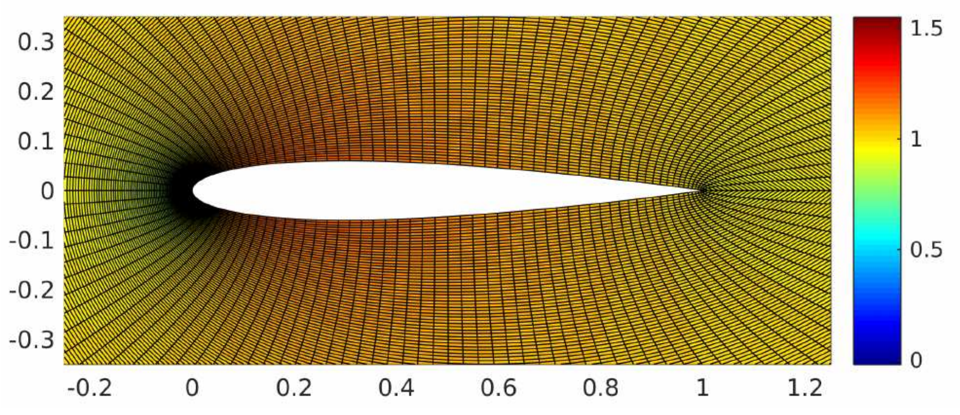

Figure 14.13: Examples of grid generated using (14.24). The strategy used is shown in Figure 14.12

Aside: Derivation of Eqn. (14.24)

This is arrived at by use of (14.8) and (14.9), noting that

$$\xi_{xx} = \frac{\partial}{\partial x}\xi_x = \left(\xi_x \frac{\partial}{\partial \xi} + \eta_x \frac{\partial}{\partial \eta}\right)\xi_x = \left(\frac{y_\eta}{J}\frac{\partial}{\partial \xi} - \frac{y_\xi}{J}\frac{\partial}{\partial \eta}\right)(y_\eta J^{-1}) \tag{14.26}$$

$$\xi_{yy} = \frac{\partial}{\partial y}\xi_y = \left(\xi_y \frac{\partial}{\partial \xi} + \eta_y \frac{\partial}{\partial \eta}\right)\xi_y = \left(\frac{x_\eta}{J}\frac{\partial}{\partial \xi} - \frac{x_\xi}{J}\frac{\partial}{\partial \eta}\right)(x_\eta J^{-1}). \tag{14.27}$$

Similarly

$$\eta_{xx} = \frac{\partial}{\partial x}\eta_x = \left(\xi_x \frac{\partial}{\partial \xi} + \eta_x \frac{\partial}{\partial \eta}\right)\eta_x = \left(\frac{y_\eta}{J}\frac{\partial}{\partial \xi} - \frac{y_\xi}{J}\frac{\partial}{\partial \eta}\right)(-y_\xi J^{-1}) \tag{14.28}$$

$$\eta_{yy} = \frac{\partial}{\partial y}\eta_y = \left(\xi_y \frac{\partial}{\partial \xi} + \eta_y \frac{\partial}{\partial \eta}\right)\eta_y = \left(-\frac{x_\eta}{J}\frac{\partial}{\partial \xi} + \frac{x_\xi}{J}\frac{\partial}{\partial \eta}\right)(x_\xi J^{-1}). \tag{14.29}$$

After some considerable manipulation of the above expressions (task made easier by use of a symbolic manipulator, eg. *Mathematica* or *Maple* software), Eqns. (14.23) can be shown to reduce to (14.24).

14.4 Complex 3D meshing

Generating meshes is a vast field, we have only discussed a few of the most basic approaches – think about meshing a three-dimensional object (or Domain) and what that would entail.

Alternative techniques to work in the physical plane are available, but we do not consider them in this course - (Google, finite elements and unstructured grids). Examples of quite complex problems are shown in Fig. 14.14, these as you will appreciate are quite advanced[*].

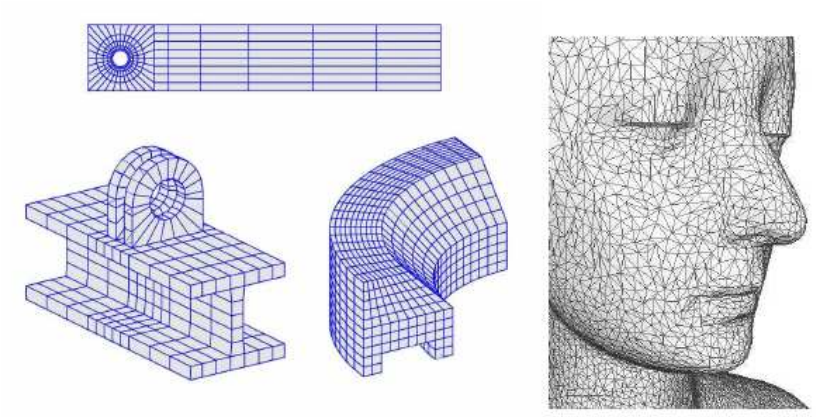

Figure 14.14: Finite element and unstructured grids in complex geometry (references below in footnote).

For further details, see Tannehill, Anderson & Pletcher, Computational Fluid Mechanics and Heat Transfer).

[*]see: http://www.featool.com/tutorial/2015/08/24/creatingstructured-grids-using-featool-matlab-functions. http://www.sci.utah.edu/the-institute/highlights/24research-highlights/cibc-highlights/439-cleaver.html