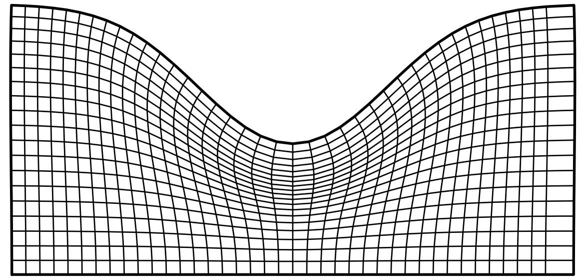
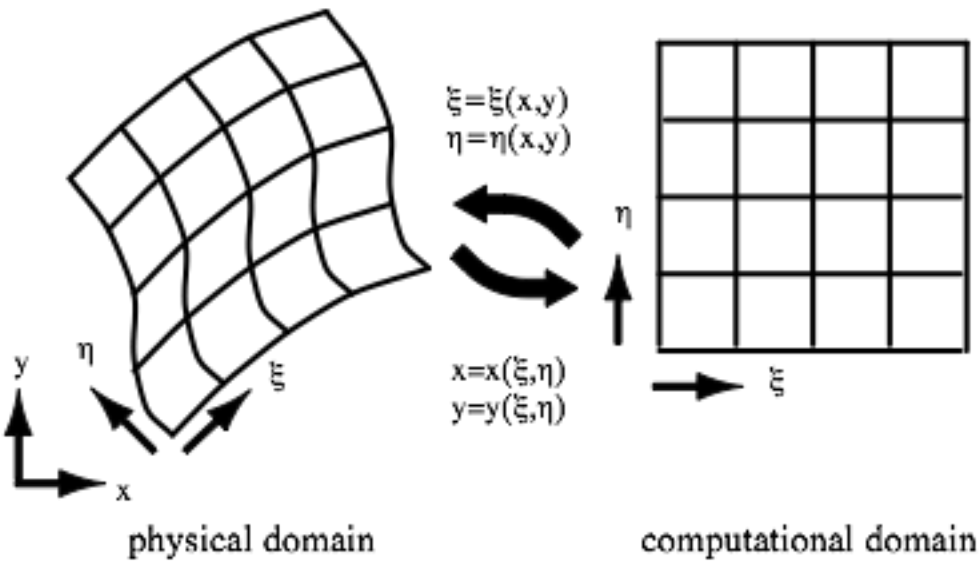
Recall 2-surface method. Grid lines are curved; can use finite diff or finite vol. Former here; will be lectures on latter, but FYI. Actually, grid is a set of points organized into cells.

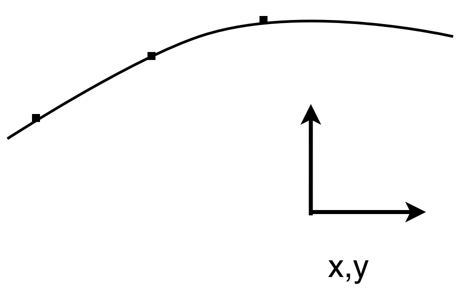


Equations on curvilinear grids: "metric" tensor

A. Computational and physical space



Grid is not in x-y direction:



Grid generation produces [x(i,j),y(i,j)]. Now use that to solve equations. Map from computational to physical. Think of i,j as a grid in ξ - η space. For example

$$\delta f/\delta\xi = (f(i+1,j)-f(i-1,j))/(\xi(i+1,j)-\xi(i-1,j)) = (f(i+1,j)-f(i-1,j))/2$$

Data values of f are stored at i,j, so they are defined in computational space. Need mapping from physical to computational; but only local (differential geometry; hence term `metric' is used).

B. But equations are in *physical* space. Use chain rule:

$\partial f/\partial x = \partial \xi/\partial x \; \partial f/\partial \xi + \partial \eta/\partial x \; \partial f/\partial \eta$

Here is the issue: we know grid in *computational* space: x(i,j) -- corresponding to $x(\xi,\eta)$.
Can compute $\partial x/\partial \xi = [x(i+1,j)-x(i-1,j)]/2$; but chain rule involves $\partial \xi/\partial x$.

Equation is in physical pace

$\partial^2\phi/\partial x^2 + \partial^2\phi/\partial y^2 = 0$

Function $\phi(i,j)$ is stored in computational space. So derivatives are in physical space, function is stored in computational space. Need differential mapping from physical to computational space. Grid x,y(i,j) maps computational to physical space.

C. Metric terms

$$\frac{\partial \phi}{\partial x} = \frac{\partial \xi}{\partial x}\frac{\partial \phi}{\partial \xi} + \frac{\partial \eta}{\partial x}\frac{\partial \phi}{\partial \eta}$$
$$\frac{\partial \phi}{\partial y} = \frac{\partial \xi}{\partial y}\frac{\partial \phi}{\partial \xi} + \frac{\partial \eta}{\partial y}\frac{\partial \phi}{\partial \eta}$$

or

$$\begin{bmatrix} \frac{\partial \phi}{\partial x} \\ \frac{\partial \phi}{\partial y} \end{bmatrix} = \underbrace{\begin{pmatrix} \frac{\partial \xi}{\partial x} & \frac{\partial \eta}{\partial x} \\ \frac{\partial \xi}{\partial y} & \frac{\partial \eta}{\partial y} \end{pmatrix}}_{Rxy} \cdot \begin{bmatrix} \frac{\partial \phi}{\partial \xi} \\ \frac{\partial \phi}{\partial \eta} \end{bmatrix}$$

$$\begin{bmatrix} \frac{\partial \phi}{\partial x} \\ \frac{\partial \phi}{\partial y} \end{bmatrix} = \boldsymbol{R_{xy}} \cdot \begin{bmatrix} \frac{\partial \phi}{\partial \xi} \\ \frac{\partial \phi}{\partial \eta} \end{bmatrix}$$

$$\begin{bmatrix} \frac{\partial \phi}{\partial \xi} \\ \frac{\partial \phi}{\partial \eta} \end{bmatrix} = \boldsymbol{R_{xy}}^{-1} \cdot \begin{bmatrix} \frac{\partial \phi}{\partial x} \\ \frac{\partial \phi}{\partial y} \end{bmatrix}$$

From chain rule

$$\begin{bmatrix} \frac{\partial \phi}{\partial \xi} \\ \frac{\partial \phi}{\partial \eta} \end{bmatrix} = \underbrace{\begin{pmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{pmatrix}}_{R_{xy}^{-1}} \cdot \begin{bmatrix} \frac{\partial \phi}{\partial x} \\ \frac{\partial \phi}{\partial y} \end{bmatrix}$$

The matrix $\mathbf{R}_{xy}^{-1}$ can be computed if the grid, x(i,j),y(i,j) is given. E.g.

$$\frac{\partial x}{\partial \xi} = \frac{x_{i+1,j} - x_{i-1,j}}{2} \qquad \frac{\partial x}{\partial \eta} = \frac{x_{i,j+1} - x_{i,j-1}}{2}$$

A common shorthand is $\mathbf{R}_{xy}^{-1} = \partial (x,y)/\partial(\xi,\eta)$.
Another view of the same: contra-variant version (note that ${}^{T}\mathbf{R}_{xy}$ is the transpose of previous)

$$\begin{bmatrix} dx \\ dy \end{bmatrix} = \underbrace{\begin{pmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial x}{\partial \eta} \\ \frac{\partial y}{\partial \xi} & \frac{\partial y}{\partial \eta} \end{pmatrix}}_{{}^{T}Rxy^{-1}} \cdot \begin{bmatrix} d\xi \\ d\eta \end{bmatrix} \qquad \begin{bmatrix} d\xi \\ d\eta \end{bmatrix} = \underbrace{\begin{pmatrix} \frac{\partial \xi}{\partial x} & \frac{\partial \xi}{\partial y} \\ \frac{\partial \eta}{\partial x} & \frac{\partial \eta}{\partial y} \end{pmatrix}}_{{}^{T}Rxy} \cdot \begin{bmatrix} dx \\ dy \end{bmatrix}$$

(Rtxy in code). We are able to evaluate ${}^{T}\mathbf{R}_{xy}^{-1}$ From inversion formula for 2-D matrix
Can be computed and stored [Rxy(*,*;2,2)]. Don't need to know ξ, η as function of x,y.

$${}^{T}\boldsymbol{R}_{xy} = \begin{pmatrix} \frac{\partial \xi}{\partial x} & \frac{\partial \xi}{\partial y} \\ \frac{\partial \eta}{\partial x} & \frac{\partial \eta}{\partial y} \end{pmatrix} = \frac{1}{\frac{\partial y}{\partial \eta}\frac{\partial x}{\partial \xi} - \frac{\partial y}{\partial \xi}\frac{\partial x}{\partial \eta}} \begin{pmatrix} \frac{\partial y}{\partial \eta} & -\frac{\partial x}{\partial \eta} \\ -\frac{\partial y}{\partial \xi} & \frac{\partial x}{\partial \xi} \end{pmatrix}$$

Pseudo-code

Rtxy(:,:,1,1) = ∂ξ/∂x ; Rtxy(:,:,1,2) = ∂ξ/∂y
Rtxy(:,:,2,1) = ∂η/∂x ; Rtxy(:,:,2,2) = ∂η/∂y

```
! i-direction differences

  DO  i=2,imax-1
    dx_i(i,:) = 0.5*( x(i+1,:) - x(i-1,:) )
    dy_i(i,:) = 0.5*( y(i+1,:) - y(i-1,:) )
  ENDDO

! one-sided difference at ends

  i = 1
    dx_i(i,:) = 0.5*( -x(i+2,:) + 4.*x(i+1,:) - 3.*x(i,:) )
    dy_i(i,:) = 0.5*( -y(i+2,:) + 4.*y(i+1,:) - 3.*y(i,:) )
  i = imax
    dx_i(i,:) = 0.5*( 3.*x(i,:) - 4.*x(i-1,:) + x(i-2,:) )
    dy_i(i,:) = 0.5*( 3.*y(i,:) - 4.*y(i-1,:) + y(i-2,:) )

! j-direction differences

  DO j=2,jmax-1
     dx_j(:,j) = 0.5*( x(:,j+1) - x(:,j-1) )
     dy_j(:,j) = 0.5*( y(:,j+1) - y(:,j-1) )
  ENDDO
  j = 1
      dx_j(:,j) = 0.5*( -x(:,j+2) + 4.*x(:,j+1) - 3.*x(:,j) )
      dy_j(:,j) = 0.5*( -y(:,j+2) + 4.*y(:,j+1) - 3.*y(:,j) )
  j = jmax
      dx_j(:,j) = 0.5*( 3.*x(:,j) - 4.*x(:,j-1) + x(:,j-2) )
      dy_j(:,j) = 0.5*( 3.*y(:,j) - 4.*y(:,j-1) + y(:,j-2) )

!  Jacobian and metrics

  DO j=1,jmax
  DO i=1,imax
    rdj           =  dx_i(i,j)*dy_j(i,j) - dx_j(i,j)*dy_i(i,j)
    rtxy(i,j,1,1) =  dy_j(i,j)/rdj
    rtxy(i,j,1,2) = -dx_j(i,j)/rdj
    rtxy(i,j,2,1) = -dy_i(i,j)/rdj
    rtxy(i,j,2,2) =  dx_i(i,j)/rdj
  ENDDO
  ENDDO
```