Computational Fluid Dynamics
http://www.nd.edu/~gtryggva/CFD-Course/

Elementary Grid Generation

Grétar Tryggvason
Spring 2011

Computational Fluid Dynamics
Outline

Stretched grids for rectangular geometries

Bilinear Interpolation

Elliptic grid generation

Unstructured hexahedron grids and block-structured grids

Imbedded boundaries

Adaptive Mesh Refinement

Computational Fluid Dynamics

There are two main reasons for using grids that are not rectangular with uniform grid spacing

1. Representing a domain with complex boundaries
2. Put grid points in parts of the domain where high resolution is needed

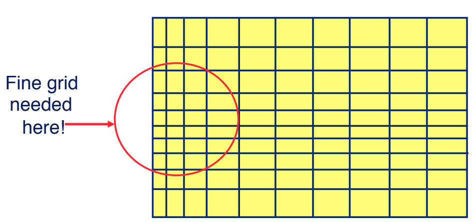
Frequently, it is necessary to deal with both issues

Computational Fluid Dynamics

Stretched Grids

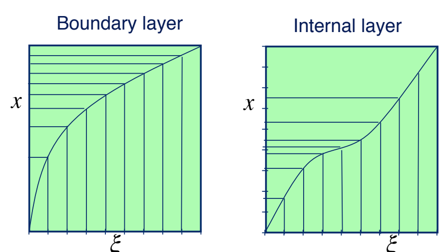
Computational Fluid Dynamics

Gridlines are straight but unevenly spaced:



Computational Fluid Dynamics

Simple stretching

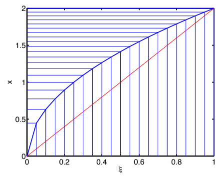


$$x = x(\xi)$$

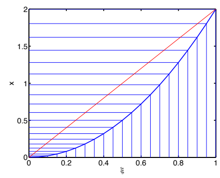
Computational Fluid Dynamics

Stretching functions-examples

"one-sided" boundary layer $x = \xi^n$



$x = \sqrt{\xi}$



$x = \xi^2$

Computational Fluid Dynamics
Stretching functions-examples

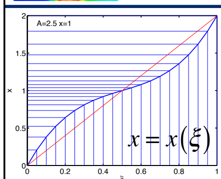
Internal layer

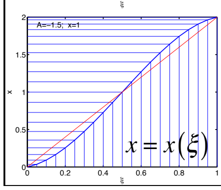
$$x = L\xi + A\underbrace{\left(x_C - L\xi\right)}_{\text{Changes signs at } x_c}\underbrace{\left(1-\xi\right)\xi}_{\text{Zero at endpoints}}$$

Line

"strength" and gives concentrates points inside domain (A>0) or at the ends (A<0)

Computational Fluid Dynamics
Stretching functions-examples



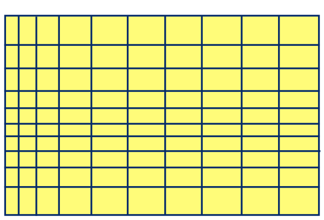



```
% A program to test 1D grid refinement.

L=2.0;xc=1.0; A=2.5
s=[0:0.05:1];n=size(s);

for i=1:n(2),
  x(i)=L*s(i)+A*(xc-L*s(i))*s(i)*(1-s(i));
End
plot(s,x,'LineWidth',2);hold on

for i=1:n(2),
  plot([s(i),s(i)],[0.0, x(i)]);
  plot([0.0,s(i)],[x(i), x(i)]);
end
xlabel('\xi','Fontsize',18)
ylabel('x','Fontsize',18)
set(gca,'Box','on');
set(gca,'Fontsize',18, 'LineWidth',2)
plot([0,1],[0,L],'r')
text(0.05,L-0.1,['A=',num2str(A),...
' x=',num2str(xc)],'Fontsize',18)

hold off; % print -depsc exampleplot
```

Computational Fluid Dynamics

The MAC method on Stretched Grids

Computational Fluid Dynamics



Computational Fluid Dynamics
Stretched grids

Special case:

$$\begin{aligned} x &= x(\xi) \\ y &= y(\eta) \end{aligned} \longrightarrow \begin{aligned} q_1 &= y_\eta^2 \\ q_2 &= 0 \\ q_3 &= x_\xi^2 \end{aligned} \qquad J = x_\xi y_\eta$$

$$u = \frac{1}{J}\left(Ux_\xi\right) = \frac{1}{x_\xi y_\eta}\left(Ux_\xi\right) = \frac{U}{y_\eta}$$

$$v = \frac{1}{J}\left(Vy_\eta\right) = \frac{1}{y_\eta x_\xi}\left(Vy_\eta\right) = \frac{V}{x_\xi}$$

$$\frac{\partial U}{\partial \xi} + \frac{\partial V}{\partial \eta} = 0 \longrightarrow y_\eta \frac{\partial u}{\partial \xi} + x_\xi \frac{\partial v}{\partial \eta} = 0 \longrightarrow \frac{1}{x_\xi}\frac{\partial u}{\partial \xi} + \frac{1}{y_\eta}\frac{\partial v}{\partial \eta} = 0$$

Computational Fluid Dynamics
Stretched grids

Approximate the conservation equation $\Delta\xi = \Delta\eta = 1$

$$y_\eta = \frac{y_{j+1} - y_j}{1} = \Delta y_{j+1/2}$$

$$x_\xi = \frac{x_{i+1} - x_i}{1} = \Delta x_{i+1/2}$$

Define:

$$\Delta y_j = \frac{1}{2}\left(\Delta y_{j+1/2} + \Delta y_{j-1/2}\right)$$

$$\Delta x_i = \frac{1}{2}\left(\Delta x_{i+1/2} + \Delta x_{i-1/2}\right)$$

$$\frac{1}{x_\xi}\frac{\partial u}{\partial \xi} + \frac{1}{y_\eta}\frac{\partial v}{\partial \eta} = 0 \longrightarrow \frac{u_{i+1/2,j} - u_{i-1/2,j}}{\Delta x_i} + \frac{v_{i,j+1/2} - v_{i,j-1/2}}{\Delta y_j} = 0$$

Computational Fluid Dynamics
Stretched grids

u-Momentum Equation

$$\frac{\partial u}{\partial t} + \frac{1}{x_\xi}\frac{\partial u^2}{\partial \xi} + \frac{1}{y_\eta}\frac{\partial vu}{\partial \eta} = -\frac{1}{x_\xi}\frac{\partial p}{\partial \xi} + \nu\left[\frac{1}{x_\xi}\frac{\partial}{\partial \xi}\left(\frac{u_\xi}{x_\xi}\right) + \frac{1}{y_\eta}\frac{\partial}{\partial \eta}\left(\frac{u_\eta}{y_\eta}\right)\right]$$

$$\frac{u^{n+1}_{i+1/2,j} - u^n_{i+1/2,j}}{\Delta t} = -\frac{(u^2)^n_{i+1,j} - (u^2)^n_{i,j}}{\Delta x_{i+1/2}} - \frac{(uv)^n_{i+1/2,j+1/2} - (uv)^n_{i+1/2,j-1/2}}{\Delta y_j}$$
$$-\frac{P_{i+1,j} - P_{i,j}}{\Delta x_{i+1/2}} + \nu\left(\frac{1}{\Delta x_{i+1/2}}\left(\frac{u^n_{i+3/2,j} - u^n_{i+1/2,j}}{\Delta x_{i+1}} - \frac{u^n_{i+1/2,j} - u^n_{i-1/2,j}}{\Delta x_i}\right)\right.$$
$$\left. + \frac{1}{\Delta y_j}\left(\frac{u^n_{i+1/2,j+1} - u^n_{i+1/2,j}}{\Delta y_{j+1/2}} - \frac{u^n_{i+1/2,j} - u^n_{i+1/2,j-1}}{\Delta y_{j-1/2}}\right)\right)$$

Computational Fluid Dynamics
Stretched grids

v-Momentum Equation

$$\frac{\partial v}{\partial t} + \frac{1}{x_\xi}\frac{\partial uv}{\partial \xi} + \frac{1}{y_\eta}\frac{\partial v^2}{\partial \eta} = -\frac{1}{y_\eta}\frac{\partial p}{\partial \eta} + \nu\left[\frac{1}{x_\xi}\frac{\partial}{\partial \xi}\left(\frac{v_\xi}{x_\xi}\right) + \frac{1}{y_\eta}\frac{\partial}{\partial \eta}\left(\frac{v_\eta}{y_\eta}\right)\right]$$

$$\frac{v^{n+1}_{i,j+1/2} - v^n_{i,j+1/2}}{\Delta t} = -\frac{(uv)^n_{i+1,j+1/2} - (uv)^n_{i,j+1/2}}{\Delta x_{i+1/2}} - \frac{(v^2)^n_{i,j+1} - (v^2)^n_{i,j}}{\Delta y_j}$$
$$-\frac{P_{i,j+1} - P_{i,j}}{\Delta y_{j+1/2}} + \nu\left(\frac{1}{\Delta x_i}\left(\frac{v^n_{i+1,j+1/2} - v^n_{i,j+1/2}}{\Delta x_{i+1/2}} - \frac{v^n_{i,j+1/2} - v^n_{i-1,j+1/2}}{\Delta x_{i-1/2}}\right)\right.$$
$$\left. + \frac{1}{\Delta y_{j+1/2}}\left(\frac{v^n_{i,j+3/2} - v^n_{i,j+1/2}}{\Delta y_{j+1}} - \frac{v^n_{i,j+1/2} - v^n_{i,j-1/2}}{\Delta y_j}\right)\right)$$

Computational Fluid Dynamics
Stretched grids

The pressure equation is:

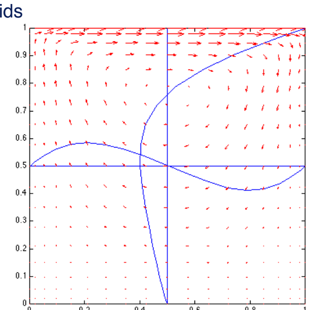
$$\frac{1}{\Delta x_i}\left(\frac{P_{i+1,j} - P_{i,j}}{\Delta x_{i+1/2}} - \frac{P_{i,j} - P_{i-1,j}}{\Delta x_{i-1/2}}\right) + \frac{1}{\Delta y_j}\left(\frac{P_{i,j+1} - P_{i,j}}{\Delta y_{j+1/2}} - \frac{P_{i,j} - P_{i,j-1}}{\Delta y_{j-1/2}}\right)$$
$$= \Delta t\left(\frac{u^*_{i+1/2,j} - u^*_{i-1/2,j}}{\Delta x_i} - \frac{v^*_{i,j+1/2} - v^*_{i,j-1/2}}{\Delta y_j}\right)$$

Which can be solved by iteration

Computational Fluid Dynamics
Stretched grids

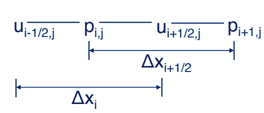
Colocated stretched grids

Re=10



Computational Fluid Dynamics
Stretched grids

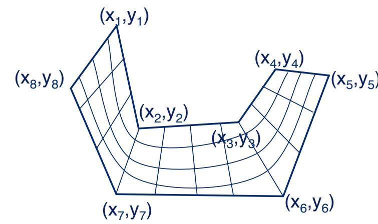
For one dimensional stretched grids, we simply replace the global Δx by the local Δx



Bilinear Interpolation

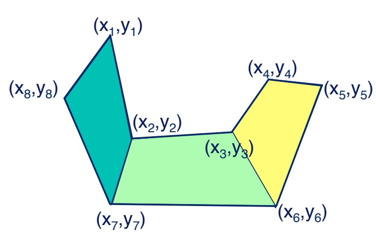
Simples grid generation is to break the domain into blocks and use bilinear interpolation within each block

As an example, we will write a simple code to grid the domain to the right



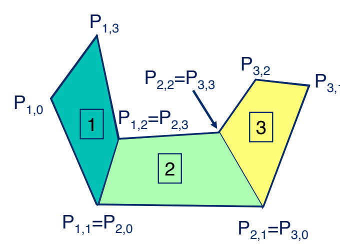
Start by breaking the domain into blocks

Then grid each block separately



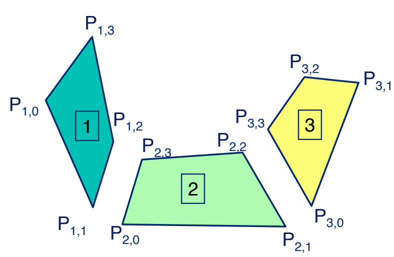
Start by breaking the domain into blocks

Then grid each block separately



Start by breaking the domain into blocks

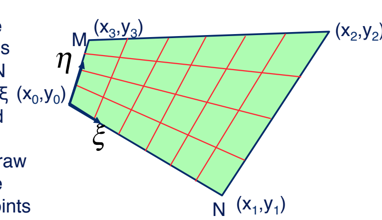
Then grid each block separately



Consider an arbitrary shaped quadrilateral block

1. Select the ξ and ξ direction.

2. Divide the opposite sides evenly with N points in the ξ direction and M in the ξ direction and draw straights line between the points on the opposite sides



Computational Fluid Dynamics
Bilinear Interpolation

Along the edge between points 0 and 1

$$x(\xi,1)=\left(\frac{N-\xi}{N-1}\right)x_0+\left(\frac{\xi-1}{N-1}\right)x_1$$

Along the edge between points 3 and 2

$$x(\xi,M)=\left(\frac{N-\xi}{N-1}\right)x_3+\left(\frac{\xi-1}{N-1}\right)x_2$$

Then interpolate again for points between the edges

$$x(\xi,\eta)=\left(\frac{M-\eta}{M-1}\right)\left(\left(\frac{N-\xi}{N-1}\right)x_0+\left(\frac{\xi-1}{N-1}\right)x_1\right)+\left(\frac{\eta-1}{M-1}\right)\left(\left(\frac{N-\xi}{N-1}\right)x_3+\left(\frac{\xi-1}{N-1}\right)x_2\right)$$

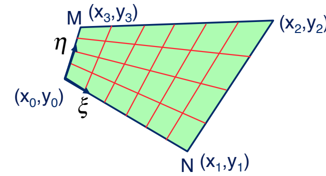
The y-coordinate is found in the same way

Computational Fluid Dynamics
Bilinear Interpolation

For a single block, we therefore have:

$$x(\xi,\eta)=\left(\frac{M-\eta}{M-1}\right)\left(\left(\frac{N-\xi}{N-1}\right)x_0+\left(\frac{\xi-1}{N-1}\right)x_1\right)+\left(\frac{\eta-1}{M-1}\right)\left(\left(\frac{N-\xi}{N-1}\right)x_3+\left(\frac{\xi-1}{N-1}\right)x_2\right)$$

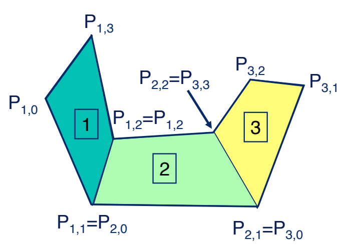
$$y(\xi,\eta)=\left(\frac{M-\eta}{M-1}\right)\left(\left(\frac{N-\xi}{N-1}\right)y_0+\left(\frac{\xi-1}{N-1}\right)y_1\right)+\left(\frac{\eta-1}{M-1}\right)\left(\left(\frac{N-\xi}{N-1}\right)y_3+\left(\frac{\xi-1}{N-1}\right)y_2\right)$$



Computational Fluid Dynamics
Bilinear Interpolation

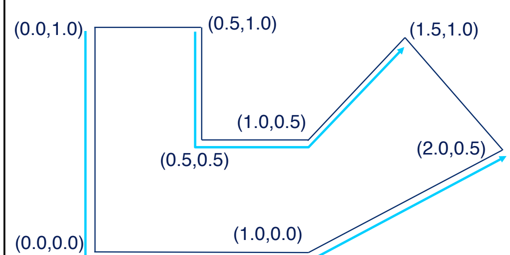
For many blocks:

1. we must interpolate for each block and
2. ensure that their boundaries are correct



Computational Fluid Dynamics
Bilinear Interpolation

Short MATLAB program to generate the grid



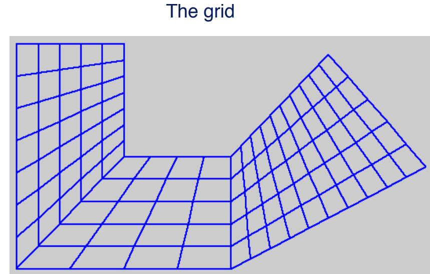

```
xp(1,1:4)=[0,0,1,2];xp(2,1:4)=[0.5,0.5,1,1.5];
yp(1,1:4)=[1,0,0,0.5];yp(2,1:4)=[1,0.5,0.5,1];
```

Computational Fluid Dynamics
Bilinear Interpolation

```
xp(1,1:4)=[0,0,1,2];xp(2,1:4)=[0.5,0.5,1,1.5];
yp(1,1:4)=[1,0,0,0.5];yp(2,1:4)=[1,0.5,0.5,1];

NumBlocks=3;
Nblock(1)=8;Nblock(2)=5;Nblock(3)=12; NTot=0;
Mblock(1)=6;Mblock(2)=6;Mblock(3)=6; MTot=6;

for l=1:NumBlocks
N=Nblock(l);M=Mblock(l);
x0=xp(1,l);x1=xp(1,l+1);x3=xp(2,l);x2=xp(2,l+1);
y0=yp(1,l);y1=yp(1,l+1);y3=yp(2,l);y2=yp(2,l+1);
Nf=2;if(l == 1),Nf=1;end
for i=Nf:N, for j=1:M
ii=i+NTot;
x(ii,j)=((M-j)/(M-1))*(((N-i)/(N-1))*x0+((i-1)/(N-1))*x1)+...
    ((j-1)/(M-1))*(((N-i)/(N-1))*x3+((i-1)/(N-1))*x2);
y(ii,j)=((M-j)/(M-1))*(((N-i)/(N-1))*y0+((i-1)/(N-1))*y1)+...
    ((j-1)/(M-1))*(((N-i)/(N-1))*y3+((i-1)/(N-1))*y2);
end, end;
NTot=NTot+Nblock(l)-1;
end

axis('equal'), hold on
for j=1:MTot;plot(x(1:NTot,j),y(1:NTot,j));end;
for i=1:NTot;plot(x(i,1:MTot),y(i,1:MTot));end;
```

Computational Fluid Dynamics
Bilinear Interpolation

The grid



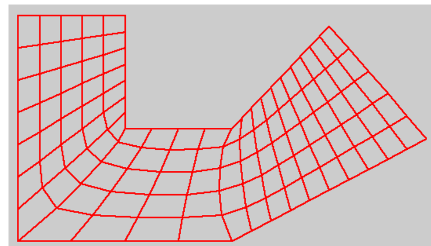
Computational Fluid Dynamics
Bilinear Interpolation

Sometimes the grid can be improved by smoothing. The simples smoothing is to replace the coordinate of each grid point by the average of the coordinates around it. This process can be repeated several times to improve the smoothness.

$$x(i,j) = 0.25*\left(x(i+1,j) + x(i-1,j) + x(i,j+1) + x(i,j-1)\right)$$
$$y(i,j) = 0.25*\left(y(i+1,j) + y(i-1,j) + y(i,j+1) + y(i,j-1)\right)$$

Computational Fluid Dynamics
Bilinear Interpolation

The grid after two smoothing iterations



Computational Fluid Dynamics
Bilinear Interpolation

Bilinear Interpolation can also be used for curved boundaries, if the points on the boundaries are given.

Higher order Interpolation functions can also be used to generate stretched grids for complex boundaries.

Computational Fluid Dynamics
Bilinear Interpolation

The grid generation results in an array of x and y coordinates for each grid point $x(\xi,\eta)$ $y(\xi,\eta)$

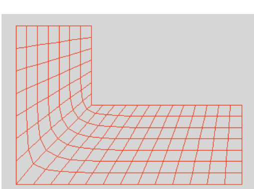
For simple problems we can include the grid generator in the fluid solver and generate the coordinates before we solve for the fluid motion

For more complex problems the grid generation step is usually separated from the flow solver and the grid points are read from a file

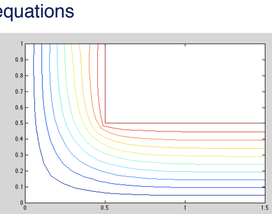
For commercial codes you can generally use many different grid generators—as long as the data format is consistent

Computational Fluid Dynamics
Bilinear Interpolation

Flow in an elbow, computed using a body fitted grid, using the streamfunction-vorticity formulation of the Navier-Stokes equations



Grid—bilinear interpolation with smoothing



Streamfunction

Computational Fluid Dynamics
Bilinear Interpolation

While bilinear interpolation is often the simplest approach for relatively simple domains, it usually requires fairly large amount of human input.

Thus, there have been major attempts to make the grid generation more automatic.

Computational Fluid Dynamics

Elliptic Grid Generation

Computational Fluid Dynamics
Elliptic Grid Generation

Elliptic Scheme

"Isotherms" of the conduction equation

$$\nabla^2 T = 0$$

has nice properties of smoothness and concentrated contour spacing where solution has a large spatial gradient.

➔ Why not use the solution to the Laplace equation as the new coordinate?

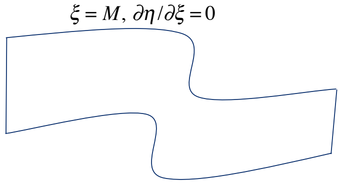
$$\nabla^2 \xi = 0, \ \nabla^2 \eta = 0$$

Computational Fluid Dynamics
Elliptic Grid Generation

Elliptic Scheme $\nabla^2 \xi = 0, \ \nabla^2 \eta = 0$

$\xi = M, \ \partial\eta/\partial\xi = 0$

$\eta = 0$, $\dfrac{\partial\xi}{\partial\eta} = 0$

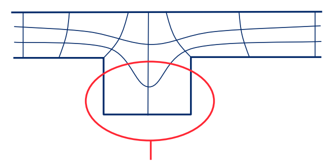


$\eta = N$, $\dfrac{\partial\xi}{\partial\eta} = 0$

$\xi = 0, \ \partial\eta/\partial\xi = 0$

Computational Fluid Dynamics
Elliptic Grid Generation

$$\nabla^2 \xi = 0, \ \nabla^2 \eta = 0$$



Few points in regions of interest

Need more control over the point location

Computational Fluid Dynamics
Elliptic Grid Generation

Further control can be applied by adding a source term:

$$\nabla^2 \xi = P(\xi,\eta), \ \nabla^2 \eta = Q(\xi,\eta)$$

Proper choice of P, Q provides the shape of the mesh.

In practice, instead of solving for ξ, η in terms of x, y, we want to solve for x, y in terms of ξ, η.

Transform the Poisson equation into (ξ, η) space.

Computational Fluid Dynamics
Elliptic Grid Generation

Adding the two equations:

$$x_\xi \nabla^2 \xi + x_\eta \nabla^2 \eta = x_\xi P(\xi,\eta) + x_\eta Q(\xi,\eta)$$
$$y_\xi \nabla^2 \xi + y_\eta \nabla^2 \eta = y_\xi P(\xi,\eta) + y_\eta Q(\xi,\eta)$$

we obtain

$$q_3 x_{\xi\xi} - 2q_2 x_{\xi\eta} + q_1 x_{\eta\eta} = -J^2 \left(P x_\xi + Q x_\eta\right)$$
$$q_3 y_{\xi\xi} - 2q_2 y_{\xi\eta} + q_1 y_{\eta\eta} = -J^2 \left(P y_\xi + Q y_\eta\right)$$

where $q_1 = x_\eta^2 + y_\eta^2$
$q_2 = x_\xi x_\eta + y_\xi y_\eta$
$q_3 = x_\xi^2 + y_\xi^2$

In discretized form (x-equation):

$$\alpha\left(x_{i+1,j}-2x_{i,j}+x_{i-1,j}\right)-0.5\beta\left(x_{i+1,j+1}-x_{i+1,j-1}-x_{i-1,j+1}+x_{i-1,j-1}\right)$$
$$\gamma\left(x_{i,j+1}-2x_{i,j}+x_{i,j-1}\right)+0.5\delta\left[P\left(x_{i+1,j}-x_{i-1,j}\right)+Q\left(x_{i,j+1}-x_{i,j-1}\right)\right]=0$$

where

$$\alpha=0.25\left[\left(x_{i,j+1}-x_{i,j-1}\right)^2+\left(y_{i,j+1}-y_{i,j-1}\right)^2\right]$$
$$\beta=0.25\left[\left(x_{i+1,j}-x_{i-1,j}\right)\left(x_{i,j+1}-x_{i,j-1}\right)+\left(y_{i+1,j}-y_{i-1,j}\right)\left(y_{i,j+1}-y_{i,j-1}\right)\right]$$
$$\gamma=0.25\left[\left(x_{i+1,j}-x_{i-1,j}\right)^2+\left(y_{i+1,j}-y_{i-1,j}\right)^2\right]$$
$$\delta=\frac{1}{16}\left[\left(x_{i+1,j}-x_{i-1,j}\right)\left(y_{i,j+1}-y_{i,j-1}\right)-\left(y_{i+1,j}-y_{i-1,j}\right)\left(x_{i,j+1}-x_{i,j-1}\right)\right]^2$$

Computational Fluid Dynamics
Elliptic Grid Generation

P,Q suggested by Thompson (1977)

$$\operatorname{sgn}(x)=\begin{cases}1 \text{ if } x>0\\ 0 \text{ if } x=0\\ -1 \text{ if } x<0\end{cases}$$

$$P(\xi,\eta)=-\sum_{l=1}^{L}a_l\operatorname{sgn}(\xi-\xi_l)\exp\left(-c_l|\xi-\xi_l|\right)$$
$$-\sum_{m=1}^{M}b_m\operatorname{sgn}(\xi-\xi_m)\exp\left(-d_m\left[(\xi-\xi_m)^2+(\eta-\eta_m)^2\right]^{1/2}\right)$$

$$Q(\xi,\eta)=-\sum_{l=1}^{L}a_l\operatorname{sgn}(\eta-\eta_l)\exp\left(-c_l|\eta-\eta_l|\right)$$
$$-\sum_{m=1}^{M}b_m\operatorname{sgn}(\eta-\eta_m)\exp\left(-d_m\left[(\xi-\xi_m)^2+(\eta-\eta_m)^2\right]^{1/2}\right)$$

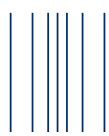
where a_l,b_m,c_l,d_m are chosen to generate appropriate grid clustering.

Computational Fluid Dynamics
Elliptic Grid Generation

Selection of P,Q

$$P(\xi,\eta)=-a_l\operatorname{sgn}(\xi-\xi_l)\exp\left(-c_l|\xi-\xi_l|\right)$$

Attracts grid lines to the line $\xi=\xi_l$.
a_l determines how strongly and
c_l determines the "reach" of the attraction



ξ_l

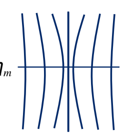
Computational Fluid Dynamics
Elliptic Grid Generation

Selection of P,Q

$$P(\xi,\eta)=-b_m\operatorname{sgn}(\xi-\xi_m)\exp\left(-d_m\left[(\xi-\xi_m)^2+(\eta-\eta_m)^2\right]^{1/2}\right)$$

Attracts grid lines to the line $\xi=\xi_m$, near the point $(\xi,\eta)=\xi_m,\eta_m$).
b_m determines how strongly and
d_m determines the "reach" of the attraction

η_m



ξ_m

Computational Fluid Dynamics
Elliptic Grid Generation

Further Recommended Reading:

Fletcher, C. A. J., *Computational Techniques for Fluid Dynamics,* V. 2, Springer-Verlag (1991)

Thompson, J. F., Warsi, Z. U. A., and Wayne Mastin, C., *Numerical Grid Generation*, North-Holland (1985).

Hoffmann, K. A., *Computational Fluid Dynamics for Engineers* (1991).