Numerical Simulation Methods
Part 2 - Applied Concepts

Lecture 13: Solution storage approaches

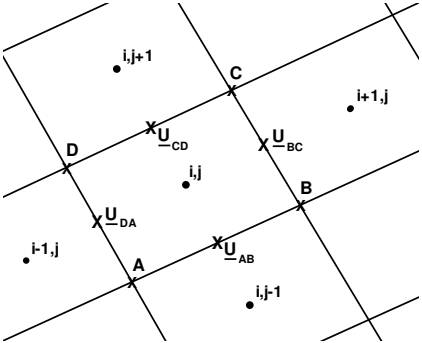
Roadmap

Applied Concepts

- Introduction to meshing
- The finite volume method
- Jameson's scheme
- **Today: Solution storage approaches and their implications**
  - **Challenges with the cell-centred approach**
  - **Cell-vertex schemes**
  - **Vertex-centred schemes**
  - **Mesh and solution storage**
- Next Lecture: Advanced implicit methods
- Introduction to computer hardware and high performance computing
- Parallel decomposition and efficiency

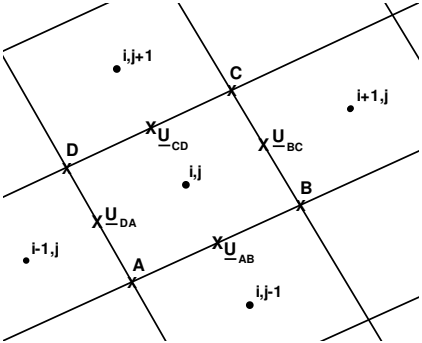
Cell-Centred Finite Volume

So far, in formulating the finite-volume method, we have assumed that the solution values are stored at the cell centres. This is a natural choice since the finite volume method produces a residual update based on the cell boundary integral.



Cell-Centred Finite Volume

So far, in formulating the finite-volume method, we have assumed that the solution values are stored at the cell centres. This is a natural choice since the finite volume method produces a residual update based on the cell boundary integral.

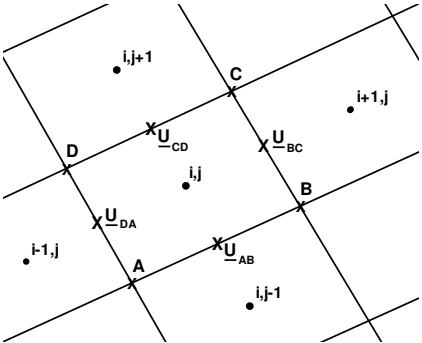


However, there are some challenges when using the cell-centred approach:

- **Boundary conditions:** how do we apply our stencil at the boundaries where one or more stencil points will be missing?

Cell-Centred Finite Volume

So far, in formulating the finite-volume method, we have assumed that the solution values are stored at the cell centres. This is a natural choice since the finite volume method produces a residual update based on the cell boundary integral.



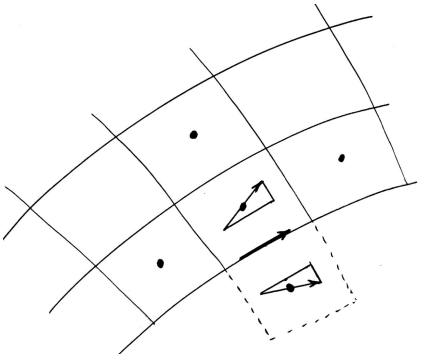
However, there are some challenges when using the cell-centred approach:

- **Boundary conditions:** how do we apply our stencil at the boundaries where one or more stencil points will be missing?
- **Obtaining surface values:** surface values are important for determining body forces, however the cell-centred approach does not store values on the surface but half a cell away from the surface.

Cell-Centred Boundary Conditions (1)

Halo cells

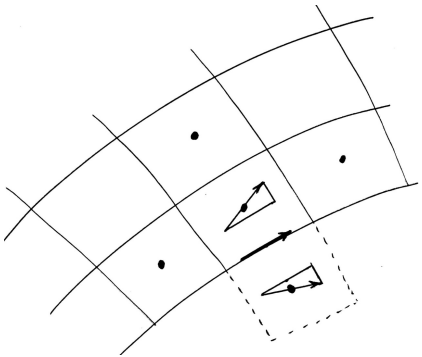
Extra fictitious cells added outside the mesh boundary to complete the stencil. The solution is not updated here but explicitly specified in order to enforce boundary conditions.



Cell-Centred Boundary Conditions (1)

Halo cells

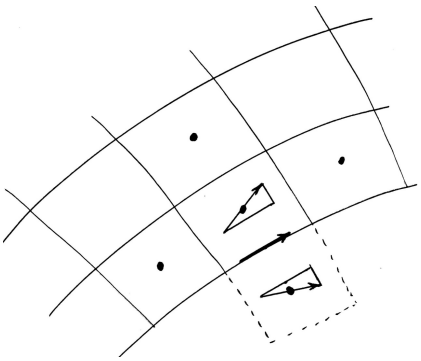
Extra fictitious cells added outside the mesh boundary to complete the stencil. The solution is not updated here but explicitly specified in order to enforce boundary conditions.



Consider an inviscid boundary where the flow must be tangential.

- Since the solution at each cell face is an average of that either side, we reflect the normal component of the velocity vector, and keep the same tangential part, and so half of the values either side gives us tangential flow since the normal components cancel

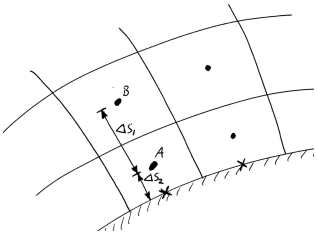
Cell-Centred Boundary Conditions (2)



- For scalar components (*e.g.* density) we simply use the same value as the first inner cell
- We could use the local gradient to get a more accurate value, but then we need to do something more clever with the velocity, since changing the pressure and density without changing the velocity vector changes the total pressure, and this is very dangerous!

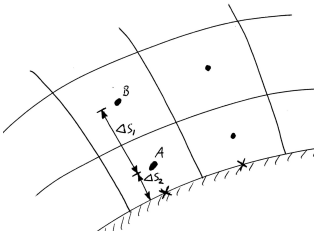
Cell-Centred Surface Values

Surface pressures in order to integrate forces for lift and drag etc.



Cell-Centred Surface Values

Surface pressures in order to integrate forces for lift and drag etc.

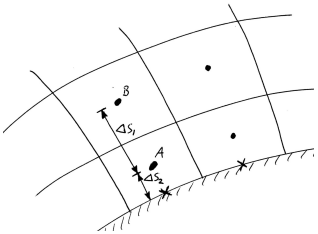


- **First-order:** assume value at the surface equal to the first cell:

$$\underline{\mathbf{U}}_{surface} = \underline{\mathbf{U}}_A$$

Cell-Centred Surface Values

Surface pressures in order to integrate forces for lift and drag etc.



- **First-order:** assume value at the surface equal to the first cell:

$$\underline{\mathbf{U}}_{surface} = \underline{\mathbf{U}}_A$$

- **Second-order:** assume linear variation and extrapolate value down to the surface:

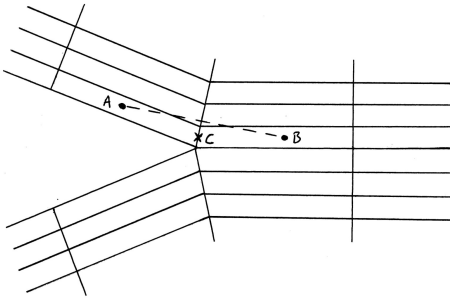
$$\underline{\mathbf{U}}_{surface} = \underline{\mathbf{U}}_A + \frac{\underline{\mathbf{U}}_A - \underline{\mathbf{U}}_B}{\Delta s_1} \Delta s_2$$

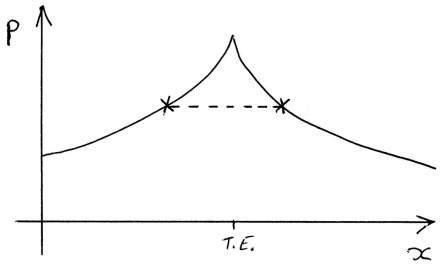
$$\underline{\mathbf{U}}_{surface} = \underline{\mathbf{U}}_A + \nabla \mathbf{U} \cdot \mathbf{\Delta s}$$

$$\int \nabla \phi dV = \int \phi \mathbf{n} dS$$

Cell-Centred - Mesh Quality

We encounter a further problem with the cell-centred approach being affected by mesh quality. Consider highly skewed cells:

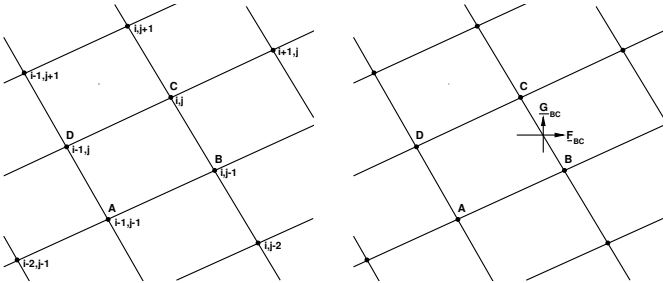




The values at 'C' are unlikely to be accurately given by $\frac{1}{2}(A + B)$.

Cell-Vertex - Introduction

The problem encountered with the cell-centred approach can be avoided by storing the solution at the cell nodes / mesh vertices. This is the **cell-vertex** approach.



The flux across each face is simply obtained from an average of the fluxes at the vertices.

$$\underline{\mathbf{F}}_{BC} = \frac{1}{2}(\underline{\mathbf{F}}_B + \underline{\mathbf{F}}_C) \qquad \underline{\mathbf{G}}_{BC} = \frac{1}{2}(\underline{\mathbf{G}}_B + \underline{\mathbf{G}}_C) \quad \text{etc}$$

Cell-Vertex - Update

Note that the updating scheme for each cell is the same regardless of the storage approach used:

$$\Delta \underline{\mathbf{U}}_{i,j} = -\frac{\Delta t}{A} \sum_{k=1}^{4} \underline{\mathbf{F}}_k \Delta y_k - \underline{\mathbf{G}}_k \Delta x_k \tag{1}$$

The flux integral gives the change in the solution for the volume surrounded by the vertices, i.e. it is not associated with a specific solution point.
The difference between the cell-centred and cell-vertex schemes is in the evaluation of the face fluxes $\underline{\mathbf{F}}_k$ and $\underline{\mathbf{G}}_k$.

Cell-Vertex - Update

Note that the updating scheme for each cell is the same regardless of the storage approach used:

$$\Delta \underline{\mathbf{U}}_{i,j} = -\frac{\Delta t}{A} \sum_{k=1}^{4} \underline{\mathbf{F}}_k \Delta y_k - \underline{\mathbf{G}}_k \Delta x_k \tag{1}$$

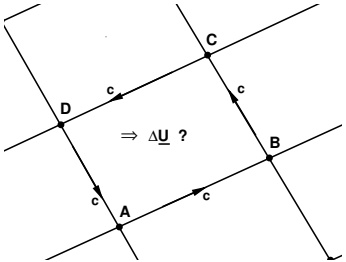
The flux integral gives the change in the solution for the volume surrounded by the vertices, i.e. it is not associated with a specific solution point.
The difference between the cell-centred and cell-vertex schemes is in the evaluation of the face fluxes $\underline{\mathbf{F}}_k$ and $\underline{\mathbf{G}}_k$.

So, if the solution is stored at the mesh points;

- What does this mean for the stencil ?
- How much information is required outside the stencil ?
- What do we need to do at boundaries ?
- How would we apply an upwind scheme ?

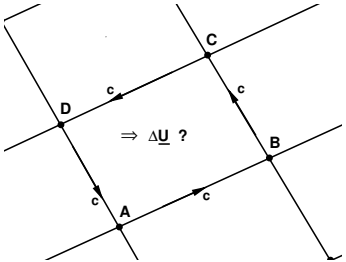
Cell-Vertex - Update, redistribution

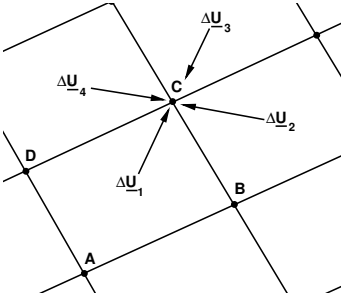
The flux integral gives the change in the solution for the volume surrounded by the vertices, i.e. it is not associated with a specific solution point.



Cell-Vertex - Update, redistribution

The flux integral gives the change in the solution for the volume surrounded by the vertices, i.e. it is not associated with a specific solution point.





Hence, this change has to be redistributed to the cell vertices by a simple averaging process

$$\Delta \underline{\mathbf{U}}_C = \frac{1}{4}(\Delta \underline{\mathbf{U}}_1 + \Delta \underline{\mathbf{U}}_2 + \Delta \underline{\mathbf{U}}_3 + \Delta \underline{\mathbf{U}}_4) \quad (2)$$

But **this is diffusive**

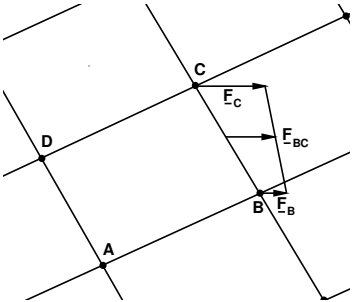
The redistribution procedure needs modifying at boundaries and upwind implementation is difficult!

Cell-Vertex - Stability (1)

Similar to the cell-centred central discretisation, the cell-vertex scheme is fundamentally and unstable and requires artificial dissipation to stabilise.

However, there is an additional cause of instability for cell-vertex schemes. Consider the evaluation of the flux at a face:

$$\underline{\mathbf{F}}_{BC} = \frac{1}{2}(\underline{\mathbf{F}}_B + \underline{\mathbf{F}}_C) \qquad (3)$$

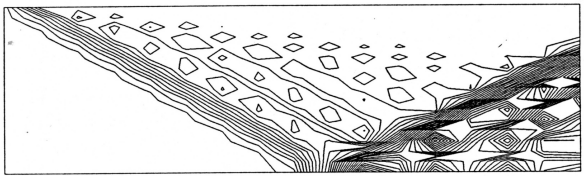


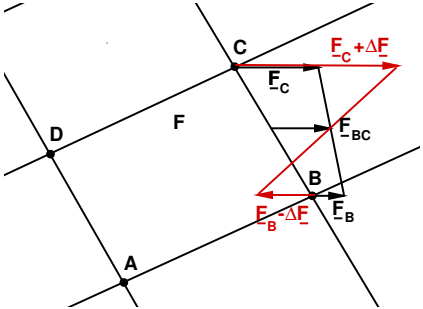
Cell-Vertex - Stability (2)

The fluxes computed at each face may satisfy the equations, but there is an unbounded error, $\Delta\underline{\mathbf{F}}$, which can cause an odd-even instability, called the CHEQUERBOARD instability

$$\underline{\mathbf{F}}_{BC} = \frac{1}{2}\left((\underline{\mathbf{F}}_B + \Delta\underline{\mathbf{F}}) + (\underline{\mathbf{F}}_C - \Delta\underline{\mathbf{F}})\right)$$

The values of cell face fluxes are unaffected by this error, so the scheme cannot correct it.





Lecture 13: Solution storage approaches

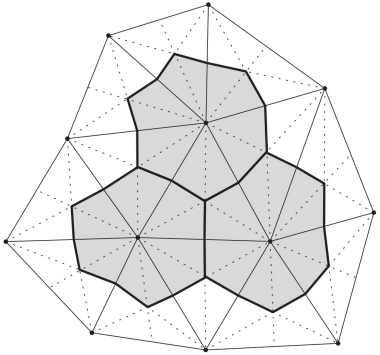
Part 2

Vertex-Centred Scheme

A third approach is a combination of cell-centred and cell-vertex; solution data is stored at cell vertices, but the mesh cells are not used as the control volumes.

Vertex-centred scheme

Control volumes are created around each mesh vertex (usually called nodes) by joining each mesh cell centre with each mesh face centre; this is called a **dual mesh**, and is normally constructed during a preprocessing stage, prior to running a simulation.



Vertex-Centred Scheme - comparison

Advantages:

- Have points on boundary: no need for special boundary treatment;
- For an unstructured mesh, each control volume has more edges, so more mesh points influence the solution; much more accurate as covers spatial variation better.
- For both unstructured and structured meshes allows higher order flux integration. One value of flux is assumed across each control volume face; more faces equals more values of flux, or higher spatial accuracy.

Vertex-Centred Scheme - comparison

Advantages:

- Have points on boundary: no need for special boundary treatment;
- For an unstructured mesh, each control volume has more edges, so more mesh points influence the solution; much more accurate as covers spatial variation better.
- For both unstructured and structured meshes allows higher order flux integration. One value of flux is assumed across each control volume face; more faces equals more values of flux, or higher spatial accuracy.

Disadvantage: This approach means the control volumes are expensive to store and the flux integral expensive to compute, as the control volume is constructed from many smaller edges (2D) or faces (3D).

Structured solver data

Consider the algorithm to solve/update the solution at every storage point (cell centre or cell vertex) every time-step.

For a structured mesh the code would loop over each storage point/cell, and for each cell compute the residual by summing over each edge/face - the number of edges/faces and connectivity is known. 2D example:

```
DO J=1,JMAX
  DO I=1,IMAX
    Rright=0.5*(F(I+1,J)+F(I,J))*dyright-0.5*(G(I+1,J)+G(I,J))*dxright
    Rleft=0.5*(F(I,J)+F(I-1,J))*dyleft-0.5*(G(I,J)+G(I-1,J))*dxleft
    Rupper=0.5*(F(I,J+1)+F(I,J))*dyupper-0.5*(G(I,J+1)+G(I,J))*dxupper
    Rlower=0.5*(F(I,J)+F(I,J-1))*dylower-0.5*(G(I,J)+G(I,J-1))*dxlower
    RES(I,J)=(Rright-Rleft+Rupper-Rlower)/AREA(I,J)
  ENDDO
ENDDO
```

Unstructured solver data (cell-iteration)

For an unstructured mesh, can loop over cells, and again loop over the number of edges/faces (variable).

Data stored as 1D arrays. Every cell needs to have a number of edges/faces stored, and for each edge/face the neighbour cell needs to be stored. Geometric data also needs to be stored as linked to cells and cell face numbers.

```
DO II=1,NCELLS
  NE=NEDGES(II)
  RES(II)=0.0
  DO I=1,NE
    NEXT=NEIGHBOUR(II,I)
    RES(II)=RES(II)+0.5*(F(II)+F(NEXT))*dy(II,I)-0.5*(G(II)+G(NEXT))*dx(II,I)
  ENDDO
  RES(II)=RES(II)/AREA(II)
ENDDO
```

Unstructured solver data (edge/face-iteration)

Alternatively, unstructured storage can be edge-based (or face-based in 3D), where all mesh edge/face data is stored, with each edge/face associated to the two cells either side of it.

```
DO II=1,NCELLS
  RES(II)=0.0
ENDDO
DO I=1,NEDGES
  IL=LEFT(I)
  IR=RIGHT(I)
  Redge=0.5*(F(IL)+F(IR))*dy(I)-
          0.5*(G(IL)+G(IR))*dx(I)
  RES(IL)=RES(IL)+Redge
  RES(IR)=RES(IR)-Redge
ENDDO
DO II=1,NCELLS
  RES(II)=RES(II)/AREA(II)
ENDDO
```

This means that numbers of edges/faces making each control volume are not required, and all data is only associated with edges/faces, for example dx() and dy() ('LEFT', 'RIGHT' defined using normal vector)

Summary

- Cell-centred approach
  - Intuitive; used by 90% of finite-volume codes
  - Has challenges at boundaries and on skewed meshes
- Cell-vertex approach
  - Solution stored at vertices; overcomes challenges with cell-centred
  - Cell updates need to be redistributed back to vertices; still need special treatment at boundaries
  - Needs extra dissipation to avoid odd-even decoupling and chequerboard instability
- Vertex-centred approach
  - Construct a dual mesh of control volumes around each vertex in the original mesh
  - Use dual mesh for finite-volume integration
  - No need for special boundary treatment since we have the solution on the boundary
  - Allows higher spatial accuracy
  - More expensive to store and access control volumes

Next Lecture: Advanced implicit schemes: how can we apply implicit in 3D?