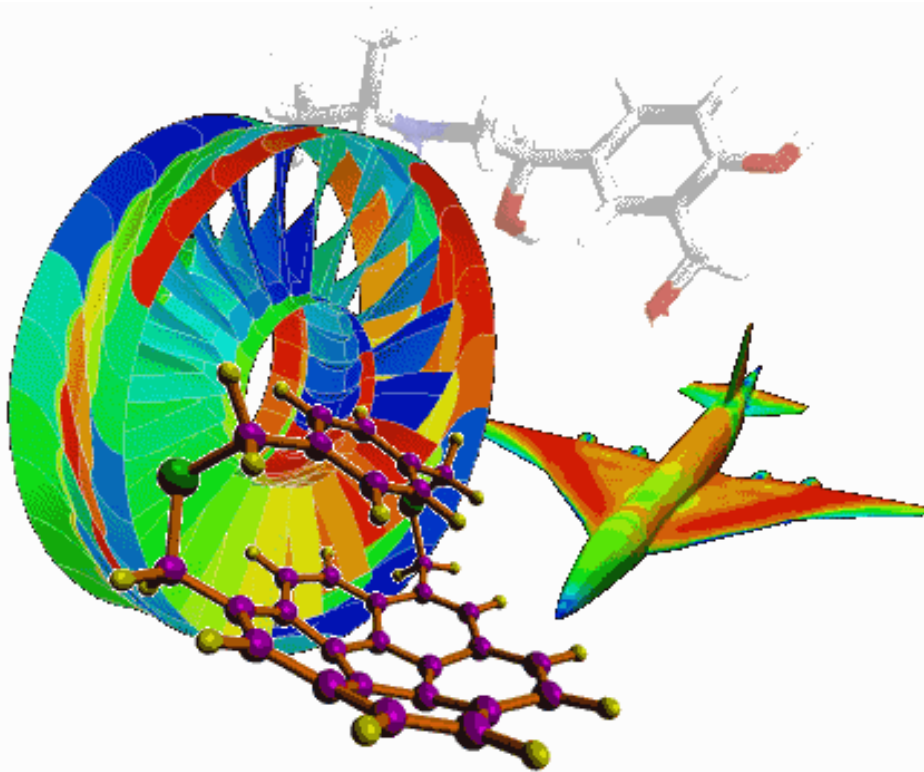


Numerical Methods for PDEs



*SSC Workgroup Meetings
Juan J. Alonso
October 8, 2001*

Overview

These notes are meant to be an overview of the various memory access patterns in typical *Partial Differential Equation* (PDE) solvers in the physical sciences. Although our experience has been mostly in fluids and structures, the comments should be applicable to other systems governed by PDEs.

The comments in these slides are based on our experience with the following types of flow solvers:

1. *Single-block* structured.
2. *Multiblock* structured.
3. *Tetrahedral* unstructured.

4. *Cartesian* unstructured.

Only a few comments about Cartesian unstructured methods will be made (for DoE Adaptive Mesh Refinement - AMR applications.)

A Bird's Eye View of a PDE

Regardless of the physical system you want to solve, a large majority of the PDEs in the physical sciences can be written in the following form

$$\frac{\partial w}{\partial t} + \mathcal{R}(w) = 0, \quad (1)$$

where w is the *vector of dependent variables* and \mathcal{R} is a typically *non-linear* vector function called the *residual*. The equation is solved in a domain of finite size subject to some boundary conditions. The domain is discretized using a *mesh* of some type and the flow variables are stored at either the nodes or the cell centers. $\mathcal{R}(w)$ results from the process of *semi-discretization* by which all of the spatial derivatives are discretized, while the time derivatives are not. All of the first, second, or higher order spatial derivatives of the dependent variables are discretized using some *stencil* and lumped into $\mathcal{R}(w)$.

How Do You Write a PDE solver? 2 Min Crash Course

Just follow the recipe:

1. Choose a suitable spatial discretization.
2. For all nodes in the mesh, calculate the spatial *residual* according to the discretization chosen.
3. Update our current guess of the solution w using either a time-stepping scheme or some sort of matrix factorization.
4. Repeat until converged.

Euler Equations in 2D

For example, the 2D compressible, Euler (inviscid) equations can be used as an approximation to the behavior of a flow. Let p , ρ , u , v , H , and E denote the pressure, density, cartesian velocity components, total enthalpy, and total energy respectively. Consider a control volume Ω with boundary $\partial\Omega$. The equations of motion of the fluid can then be written in integral form as

$$\frac{d}{dt} \iint_{\Omega} \mathbf{w} \, dx \, dy + \oint_{\partial\Omega} (\mathbf{f} \, dy - \mathbf{g} \, dx) = \mathbf{0}, \quad (2)$$

where \mathbf{w} is the vector of flow variables

$$\mathbf{w} = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho E \end{pmatrix},$$

and \mathbf{f} , \mathbf{g} are the Euler flux vectors

$$\mathbf{f} = \begin{pmatrix} \rho(u - x_t) \\ \rho u(u - x_t) + p \\ \rho v(u - x_t) \\ \rho E(u - x_t) + pu \end{pmatrix}, \quad \mathbf{g} = \begin{pmatrix} \rho(v - y_t) \\ \rho u(v - y_t) \\ \rho v(v - y_t) + p \\ \rho E(v - y_t) + pv \end{pmatrix}.$$

Also, for an ideal gas, the equation of state may be written as

$$p = (\gamma - 1) \rho \left[E - \frac{1}{2}(u^2 + v^2) \right].$$

Applying Equation 2 independently to each cell in the mesh we obtain a set of ordinary differential equations of the form

$$\frac{d}{dt}(\mathbf{w}_{ij} V_{ij}) + \mathbf{R}(\mathbf{w}_{ij}) = \mathbf{0}, \quad (3)$$

where V_{ij} is the volume of the i, j cell and the residual $\mathbf{R}(\mathbf{w}_{ij})$ is obtained by evaluating the flux integral in Equation 2.

Note that if you used the Reynolds Averaged Navier-Stokes (RANS) equations, the only thing you need to do is to add another portion of the residual $\mathbf{R}(\mathbf{w})$.

Once the residual has been calculated, it can be used to update the current guess for the solution \mathbf{w} using Equation 1. This process is repeated iteratively until convergence is reached.

Applicability

Notice that this description of semi-discrete PDEs is applicable to

1. Euler and RANS solvers.
2. LES (Large Eddy Simulation) procedures.
3. Finite Element methods.

and largely any PDE solution method that is matrix-free and that doesn't use numerical techniques such as FFTs, etc.

For purposes of SSC, the most important thing is to ask ourselves the following questions:

1. How is the memory access pattern affected by the stencil of the semi-discretization?
2. How does the choice of mesh topology affect these memory accesses?
3. Once a node in the mesh (discrete memory location) is touched, what are the operations that typically need to be performed on it?

It turns out that the stencil of the spatial discretization and the mesh choices are fundamental to understanding how memory is accessed.

Single-Block Structured Discretization

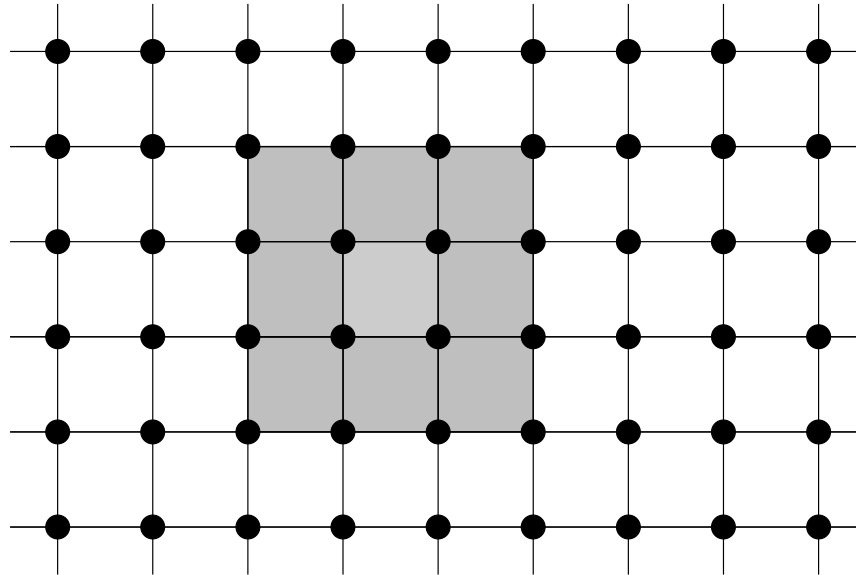


Figure 1: Structured Mesh

- Usually for domains that are not geometrically complex (although flow may be).

- Typical codes: FLO87, FLO107, DNS, development codes for testing various ideas, etc.
- Mesh ordering follows logical structure of an n-dimensional array: neighbors (important in computing the residual) can be easily accessed by unit shifts on array indices.
- Connectivity of the mesh is implicit in the data structure.
- For cache optimization purposes, varying stride in each coordinate direction is important.

Multi-Block Structured Discretization

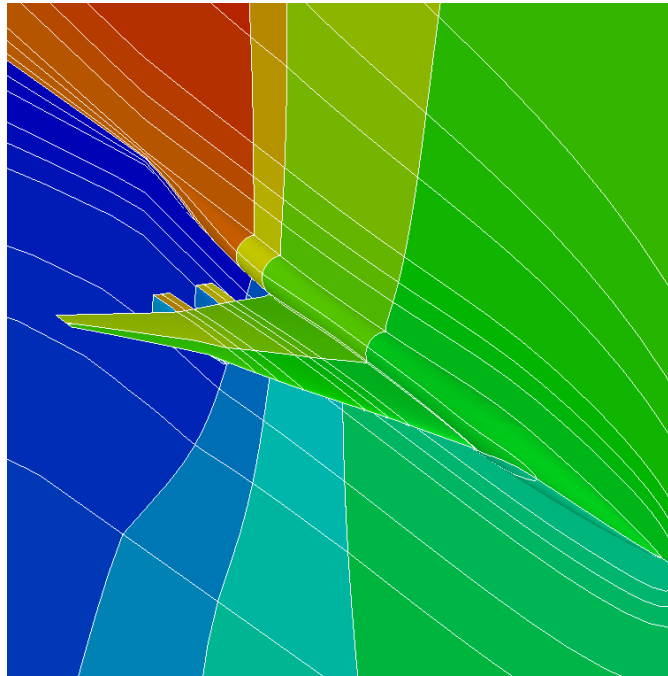


Figure 2: Multiblock Structured Mesh Around an Airplane

- Geometrically complex domains.

- Typical codes: FLO107-MB, TFLO, etc.
- Mesh ordering within each block follows logical structure of an n-dimensional array: neighbors (important in computing the residual) can be easily accessed by unit shifts on array indices.
- Connectivity of each block is implicit in the data structure.
- Inter-block connectivity must be supplied by user and defines the way in which all blocks are attached to each other. *Halo* communication typically used to isolate blocks.

Tetrahedral Unstructured Discretization

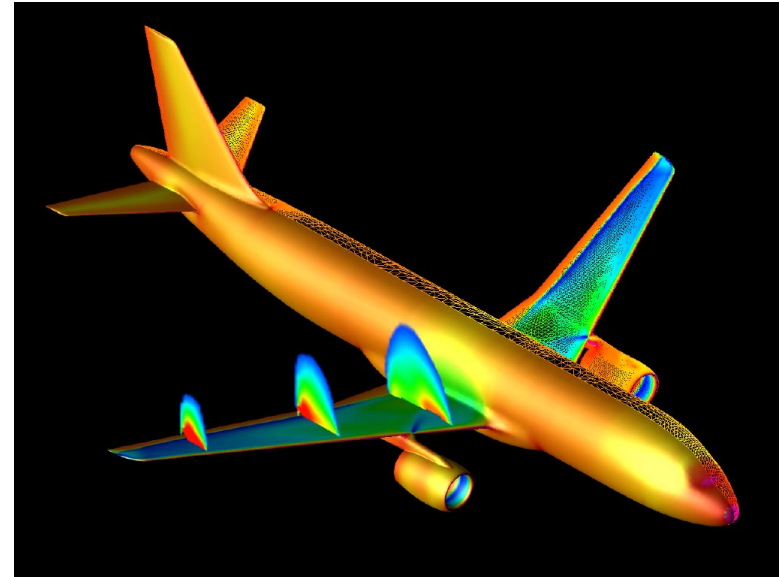
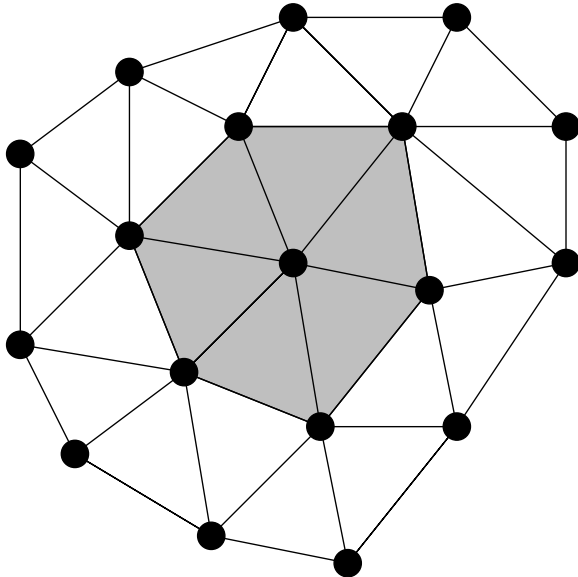


Figure 3: Unstructured Tetrahedral Mesh Around A320

- Used for domains with high geometric complexity (full aircraft, combustors, etc.)

- Typical codes: AIRPLANE, combustor LES code, etc.
- Mesh locations follow no particular order. In fact, mesh can be reordered to improve cache performance.
- Connectivity of the mesh must be supplied by user.
- Mapping of mesh to data structure fundamental to achieve high performance in a variety of computers.

AMR Cartesian Discretization

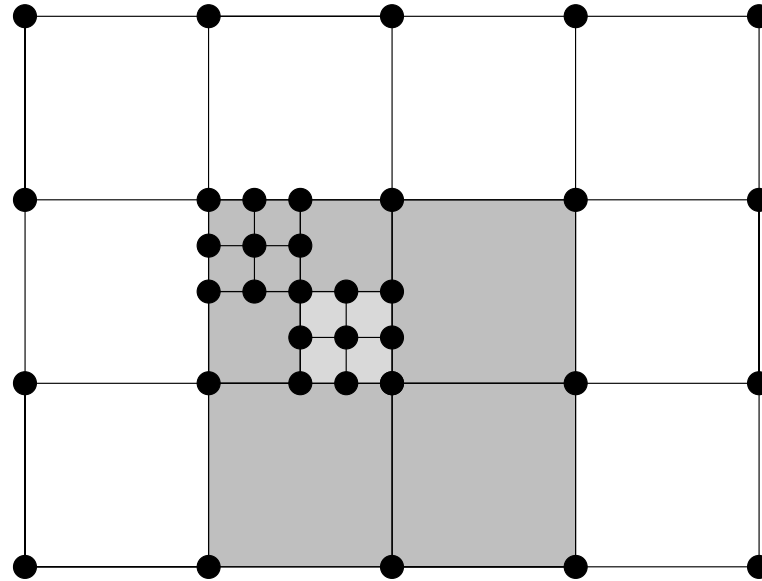


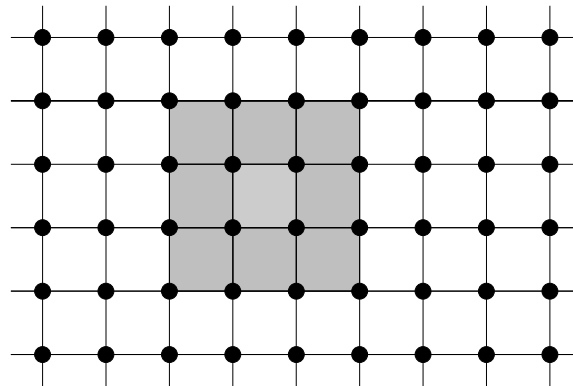
Figure 4: AMR Cartesian Mesh

- Used for domains with high geometric complexity (full aircraft, combustors, etc.) but typically only for either Euler flows or domains with complex fluid phenomena but Cartesian boundaries.

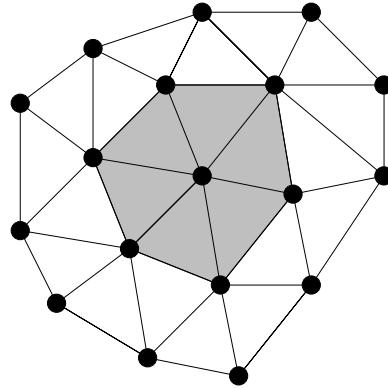
- Typical codes: DoE codes, ASCI Chicago, CART3D.
- Mesh locations are *structured* but the underlying data structure is usually of the tree type.
- Connectivity of the mesh is inherent in data structure but not *trivial*.
- Mapping of mesh to data structure fundamental to achieve high performance in a variety of computers.

Stencils in Various Types of Meshes

Single-block structured and multi-block structured codes usually have a one- or two layer stencil structure. That means that in 2D all nodes/cells that adjoin the node/cell in question are likely to belong to the stencil. Examples later on.



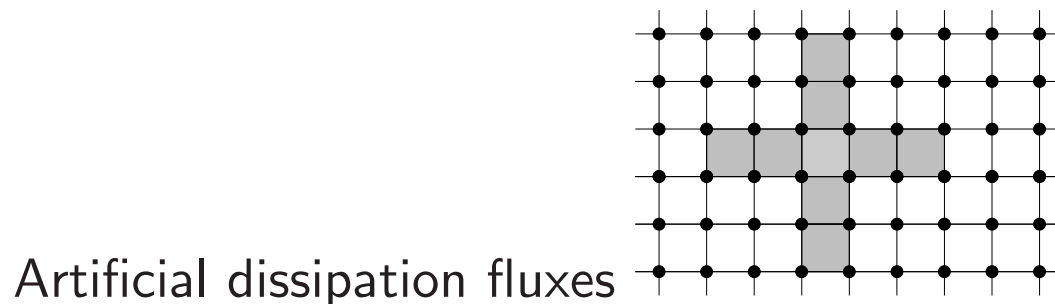
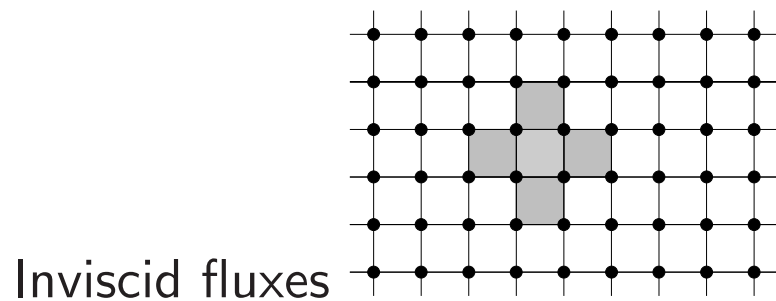
The situation in unstructured meshes is similar, although, due to the nature of the mesh, a little more random



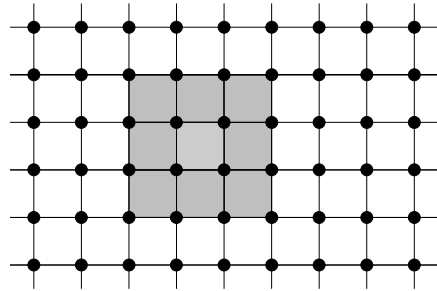
Note that in a node-based unstructured mesh solver, if a double halo is required, one may need to *reach* 2 edges over. Depending on the edge connectivity (and its incidence order) one may need a large amount of data.

Operations/Work Required

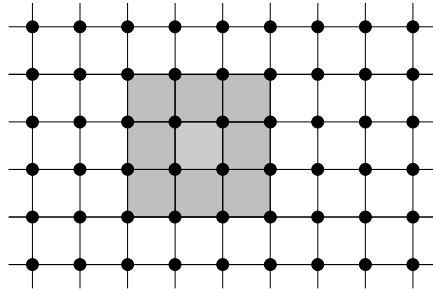
For a typical solver like TFLO, the following operations need to be performed, *every* iteration, for *every* cell in the mesh (TFLO is a cell-centered code.)



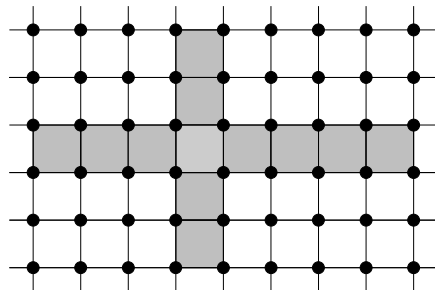
Viscous fluxes

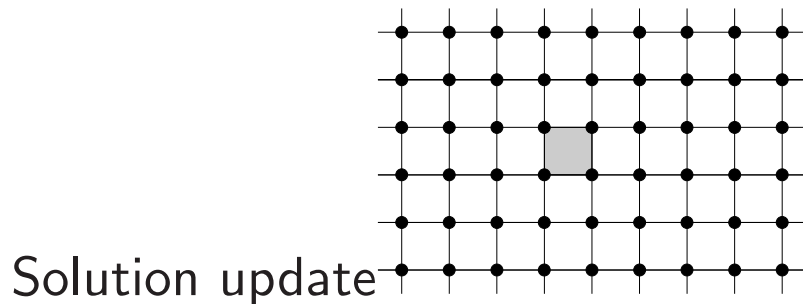


Time-step computation



Residual smoothing





Other operations need to be considered such as

1. Multigrid
2. Interpolation between meshes for multiphysics computations (DoE)
3. Matrix factorization steps (ADI and similar). Similar to Residual Smoothing.
4. Local and global preconditioning (fall in the classes described above).

5. Advanced artificial dissipation algorithms.

Code Excerpts

Subroutine EFLUX from FLO82 (2D Euler solver for airfoils)

```
C
C   FLUX IN THE J DIRECTION
C
DO 30 J=2,JL
DO 30 I=2,IL
XX      = X(I,J,1)  -X(I-1,J,1)
YX      = X(I,J,2)  -X(I-1,J,2)
PA      = P(I,J+1)  +P(I,J)
QSP     = (XX*W(I,J+1,3)  -YX*W(I,J+1,2))/W(I,J+1,1)
QSM     = (XX*W(I,J,3)    -YX*W(I,J,2))/W(I,J,1)
FS(I,J,1) = QSP*W(I,J+1,1)  +QSM*W(I,J,1)
FS(I,J,2) = QSP*W(I,J+1,2)  +QSM*W(I,J,2)  -YX*PA
FS(I,J,3) = QSP*W(I,J+1,3)  +QSM*W(I,J,3)  +XX*PA
FS(I,J,4) = QSP*(W(I,J+1,4)  +P(I,J+1))  +QSM*(W(I,J,4)  +P(I,J))
```

```
30 CONTINUE
C
C   ACCUMULATE THE FLUX IN THE J DIRECTION
C
   DO 40 N=1,4
   DO 40 J=2,JL
   DO 40 I=2,IL
   DW(I,J,N) = DW(I,J,N) +FS(I,J,N) -FS(I,J-1,N)
40 CONTINUE
```

Key Issues/Questions

In order to determine the features of a language that allows for all of these types of operations, we must be able to handle

1. All types (or a generalization) of the local stencils described earlier.
2. Ability to handle not just scalars, but also vector fields.
3. Rewrite code in such a way that all operations are performed one point at a time (outer loop on mesh nodes/cells, not operations!)
4. Must conceptualize the ability to do non-local operations (multigrid, residual smoothing, ADI).

5. Must efficiently implement shifts in all coordinate directions so that data accesses with non-unity stride are fast.
6. For unstructured/Cartesian AMR meshes, only real option is to *copy* small “chunks” of data obtained by domain decomposition (graph partitioning/graph coloring) and make sure that operations are performed efficiently within each chunk.
7. Others?