CFD for Aerospace Applications

Direct and Iterative Methods

Dr. Ziad Boutanios

Concordia University - MIAE

February 22, 2022

Recap of the previous lecture

- In the previous lecture we examined the issue of accuracy vs. stability as defined by the Godunov order barrier theorem, which states that the only one-step monotone advection scheme is the first-order upwind scheme.
- We also got an estimate of the maximum order of accuracy of a stable semi-discrete advection scheme from the Iserles barrier theorem, which implied that upwind bias could be used for higher-order stability.
- We introduced Total Variation as a measure of solution oscillation.
- We also introduced the Total Variation Diminishing condition.

- We studied the MUSCL scheme which allows designing higher-order accurate stable schemes through reconstruction of the solution fluxes at the cell faces.
- This allowed the use of upwind bias for higher-order stability.
- We studied error analysis based on the L^2 norm, which allowed an assessment of the following,
  - Initial value problem well-posedness
  - Consistency
  - Stability
  - Convergence
- We introduced the computer round-off error.
- We studied linear stability analysis of the round-off error, or the Neumann stability method, using a generic time variation of the error and an exponential time variation.

- The Neumann stability method allowed us to set stability limits on the Courant number depending on the discretized PDE.
- We introduced time stepping, where explicit time stepping uses only values from previous time levels to calculate the solution at the next time level, and implicit time stepping uses values from the next time level as well.
- Finally, we presented a physical explanation of the Courant number as the ratio between the flow velocity and the grid velocity.
- For explicit stepping stability one needs the flow velocity to be less than the grid velocity.

Introduction

- The result of PDE discretization is an algebraic system for the flow variables at the mesh points.
- Take for example the 3D compressible isothermal Navier-Stokes equations with a two-equation turbulence model, having 7 variables per computational cell.
- A typical coarse mesh of 5 million cells will result in 35 million variables and as many algebraic equations.
- Solving such an extensive system is far from trivial.

The transient case

- Consider the transient 1D linear advection equation of a vector variable $\mathbf{q}$,

$$\frac{\partial \mathbf{q}}{\partial t} + \mathbf{a}\frac{\partial \mathbf{q}}{\partial x} = 0,$$

  where $\mathbf{a}$ is the advection velocity.
- The explicit finite difference discretization of this equation will typically look like this,

$$\mathbf{q_i^{t+1}} = \mathbf{q_i^t} + \frac{\mathbf{a}\Delta t}{\Delta x}\left(\mathbf{q_i^t} - \mathbf{q_{i-1}^t}\right)$$

- The only unknown variable is the one we want to solve for at the next time level, which can be done algebraically.
- This is trivial and we can proceed one algebraic equation at a time for the whole system, whether in serial or in parallel.

- On the other hand the implicit finite difference discretization of the 1D linear advection equation will typically look like this,

$$\mathbf{q}_\mathbf{i}^{\mathbf{t+1}} = \mathbf{q}_\mathbf{i}^{\mathbf{t}} + \frac{\mathbf{a}\Delta t}{\Delta x}\left(\mathbf{q}_\mathbf{i}^{\mathbf{t+1}} - \mathbf{q}_{\mathbf{i-1}}^{\mathbf{t+1}}\right)$$

- There are other unknown variables from the future time level at the RHS, and an explicit algebraic solution is not possible.
- Instead we must assemble a matrix system of all the algebraic equations at all cells and invert it.
- This is also the case for steady-state problems which are discretized in space, resulting in a matrix of coupled algebraic equations.
- There are two main types of methods for such linear algebraic systems, the direct and the iterative methods.

The steady-state case

- Consider the steady-state 1D linear advection equation of a vector variable $\mathbf{q}$,
$$\mathbf{a}\frac{\partial \mathbf{q}}{\partial x} = b,$$
where b is a source/sink term.
- The explicit finite difference discretization of this equation will typically look like this,
$$\mathbf{q_i^n} - \mathbf{q_{i-1}^n} = \frac{\mathbf{a}\Delta x}{b}.$$
- Here all variables are at the same iteration level n and unknown, and none can be solved for in an algebraic explicit manner.
- This discretization leads to a matrix of algebraic equations for the whole system, which is equivalent to the transient implicit case and requires inverting a matrix.

- It is also possible to formulate a steady-state system using a pseudo-transient approach.
- Local time stepping is such an example where one introduces an artificial transient term in every cell.
- In this context one advances each cell in time with a different time step depending on the local Courant number.
- This is done until all spatial derivatives are equal to zero, which is when the system reaches steady-state.
- Other approaches are also possible with compact finite difference schemes with their own concepts of explicit and implicit formulations.
- Local time stepping and compact finite differencing are outside the context of this course and will not be covered.

- The takeaway from the transient and steady-state discretizations discussion is that extensive matrix systems will need to be solved.
- As it stands one can use either direct and iterative methods.
- These will be discussed in the next slides.

Direct methods

- Direct methods invert the matrix directly using a finite number of arithmetic operations and leading to the exact solution of the linear algebraic system.
- In practice, the number of arithmetic operations of a direct method can be pretty high, of order $O(N^3)$ where N is the number of unknowns.
- Therefore, direct methods are restricted to all but the smallest of problems with meshes rarely larger than one million cells.
- An exception is made for tridiagonal systems which can be inverted quite efficiently with fast direct methods like the Thomas algorithm or Gaussian elimination.
- We will not discuss direct methods further in this course.

Iterative methods

- Iterative methods are based on a succession of approximate solutions, which can in principle lead to the exact solution after an infinite number of steps.
- Since an infinite number of steps is impossible to achieve in practice we stop after a finite number of steps once the residual errors have been reduced to an acceptable level.
- This is acceptable since iterative methods perform a small number of operations on the matrix elements of the algebraic system in an iterative way.
- Iterative method research is a very active field with a very large number of methods available, different rates of convergence and levels of complexity.

- In this lecture we will discuss the basic concepts of iterative methods as they pertain to the simplest methods, based on the Jacobi and Gauss-Seidel methods.
- The critical objective of iterative methods is reaching the lowest possible rate of convergence as a function of number of unknowns, or convergence acceleration.
- We will discuss some of the current convergence acceleration methods, like overrelaxation and preconditioning.
- We will discuss method stability.
- We will also discuss the multigrid method, which is the most general method to accelerate convergence.
- Finally we will discuss linearization methods of non-linear methods, based on the Newton method.

Basic iterative methods

- We are essentially solving a linear system of algebraic equations of the following matrix form,

$$A\mathbf{q} = \mathbf{b}, \tag{1}$$

where A is an $N \times N$ matrix, $\mathbf{q}$ is the solution vector with N components and $\mathbf{b}$, is the right hand side vector with N components as well.

- Instead of inverting A to find the exact solution, which is quite costly in general, we will try to find an approximate solution $\mathbf{q}$ iteratively.
- The approximate solution applies to the equation

$$A\mathbf{q} - \mathbf{b} = \mathbf{r}, \tag{2}$$

where $\mathbf{r}$ is the residual error.

- The aim is to get the residual error close enough to 0, a level determined by the equations solved and the purpose of the simulation.
- The closeness of the residual error to 0 is usually assessed by calculating its norm, like the L^2 norm.
- We will discuss two basic iterative methods, the Jacobi method and the Gauss-Seidel method.
- At the end of the lecture we will also discuss the most general and successful iterative method to date, the multigrid method.

The Jacobi method

- The Jacobi method consists of breaking the matrix A into the sum of a lower-triangular, diagonal and upper-triangular matrix.
- This transforms equation 1 into,

$$(L + D + U)\mathbf{q} = \mathbf{b}, \tag{3}$$

  where L is the lower-triangular matrix, D is the diagonal matrix and U is the upper-triangular matrix.
- Since $\mathbf{q}$ is a solution of equation 3 we can rewrite it as,

$$L\mathbf{q} + D\mathbf{q} + U\mathbf{q} = \mathbf{b}.$$

- The above equation can be rearranged as,

$$D\mathbf{q} = \mathbf{b} - (L + U)\mathbf{q}.$$

- Since D is diagonal we can invert it quite easily to get,

$$\mathbf{q} = D^{-1}\left[\mathbf{b} - (L + U)\mathbf{q}\right]. \tag{4}$$

- The Jacobi method consists of starting with an initial guess of the solution $\mathbf{q}^n$ at the current level n, which we plug into the RHS to get an updated approximation $\mathbf{q}_{n+1}$ at the next level $n+1$ on the LHS.

$$\mathbf{q_{n+1}} = D^{-1}\left[\mathbf{b} - (L + U)\mathbf{q_n}\right]. \tag{5}$$

- In matrix element notation we can write equation 5 as,

$$\mathbf{q_i^{n+1}} = \frac{1}{a_{i,i}}\left(b_i - \sum_{i \neq j} a_{i,j}\mathbf{q_i^n}\right) \quad \text{for} \quad i = 1, \cdots, n. \tag{6}$$

- The iteration in equation 5 can be repeated until the required residual error is small enough.
- Each Jacobi iteration requires about $O(N^2)$ iterations, which is N times less than a direct method.
- Let's check the example from the book...

The Gauss-Seidel method

- The Gauss-Seidel method is quite similar to the Jacobi method.
- Starting from the split matrix equation 3,

$$(L + D + U)\mathbf{q} = \mathbf{b}. \tag{7}$$

- We rearrange to get,

$$(L + D)\mathbf{q} = \mathbf{b} - U\mathbf{q}. \tag{8}$$

- We note that the matrix $(L + D)$ and relatively easy to invert via back-substitution, albeit more costly than inverting D alone.
- Inverting $(L + D)$ we now get,

$$\mathbf{q} = (L + D)^{-1}\left[\mathbf{b} - U\mathbf{q}\right], \tag{9}$$

which is the basic Gauss-Seidel equation.

- Similar to the Jacobi iteration we write the Gauss-Seidel iteration as,

$$\mathbf{q}^{\mathbf{n+1}} = (L + D)^{-1} \left[\mathbf{b} - U\mathbf{q}_{\mathbf{n}} \right], \qquad (10)$$

- In matrix element notation we can write equation 5 as,

$$\mathbf{q}_{\mathbf{i}}^{\mathbf{n+1}} = \frac{1}{a_{i,i}} \left(b_i - \sum_{i \neq j} a_{i,j} \mathbf{q}_{\mathbf{i}}^{\mathbf{n}} - \sum_{j=1}^{i-1} a_{i,j} \mathbf{q}_{\mathbf{j}}^{\mathbf{n+1}} \right) \quad \text{for} \quad i = 1, \cdots, n. \qquad (11)$$

- Comparing the element notation equation 11 to equation 6 we can see that the Gauss-Seidel method uses the updated solution from the updated points as it moves down the solution vector **q**.

- The additional computational price paid to invert $(L + D)$ provides a much better approximation to the inverse of the original matrix A.
- This results in a significant reduction in the number of iterations to the same residual error as compared to the Jacobi method, but still proportional to $O(N^2)$.
- Let's check the example from the book...

Typical issues with iterative methods

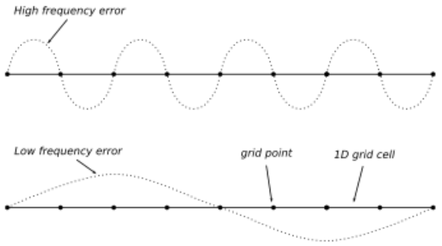

Figure 1: High and low frequency residual errors on a 1D mesh.

- The top part of figure 1 shows an example of a high-frequency error variation with its wavelength spanning the minimum possible 3 consecutive points of a 1D mesh.
- The bottom part of figure 1 shows an example of a low frequency error variation with a wavelength spanning several mesh points.

- Iterative methods are quite efficient at damping high frequency errors, and can reduce them significantly in a few iterations
- On the other hand they do a poor job at damping lower frequency errors, which can take thousands of iterations to get significant reductions.
- This poor low frequency damping leads to an asymptotically slow convergence which plagues most iterative methods.
- This drawback can be mitigated by using techniques such as Successive Over-Relaxation (SOR) or preconditioning.
- One can also resort to multiscale methods like multigrid.
- They will be discussed next.

Convergence acceleration and stability

- We will discuss two methods of convergence acceleration, over-relaxation and preconditioning, and one method of stability, under-relaxation.
- There are many others but these are quite common and arguably the most successful.

Solution relaxation

- Solution relaxation is the practice of deliberately adjusting the new solution based on a preset ratio.
- Consider the solution at the current level $\mathbf{q}^{\mathbf{n}}$ and the solution at the next level $\mathbf{q}^{\mathbf{n+1}}$ obtained from the iterative method.
- The relaxed solution is defined as,

$$\tilde{\mathbf{q}}^{\mathbf{n+1}} = \omega \mathbf{q}^{\mathbf{n+1}} + (1 - \omega)\mathbf{q}^{\mathbf{n}}, \tag{12}$$

  where ω is defined as the relaxation factor.
- Solution relaxation is depcited graphically in figure 2.
- When $\omega = 1$ we recover the original unrelaxed solution $\mathbf{q}^{\mathbf{n+1}}$.

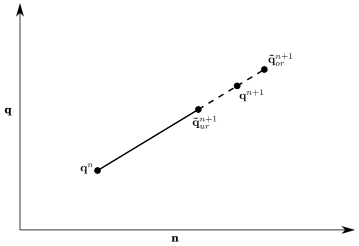
$\tilde{\mathbf{q}}_{or}^{n+1}$
$\mathbf{q}^{n+1}$
$\mathbf{q}$
$\tilde{\mathbf{q}}_{ur}^{n+1}$
$\mathbf{q}^{n}$
$\mathbf{n}$

Figure 2: Under- and over-relaxation depicted in graphical form.

- When $0 < \omega < 1$ we move the solution back to $\tilde{\mathbf{q}}_{\mathbf{ur}}^{\mathbf{n+1}}$, which is called under-relaxation and improves method stability at the expense of convergence rate.
- When $\omega > 1$ we drive the solution further to $\tilde{\mathbf{q}}_{\mathbf{or}}^{\mathbf{n+1}}$, which is called over-relaxation and improves convergence rate at the expense of stability.
- In practice ω is kept within following range $0 < \omega \leq 2$ since values exceeding 2 lead to method divergence and eventually crash.

Preconditioning

- Preconditioning is applying a transformation to a problem that makes it better suited to numerical solution.
- The expression *preconditioning* derives from the fact that the transformation effectively reduces the *condition number* of the problem.
- The condition number of a function is a measure of the change in its output for a small change/error in its input.
- A problem with a low condition number is well-conditioned, and a problem with a high condition number is ill-conditioned.
- Once preconditioned, a problem can be more easily solved with iterative methods since its rate of convergence will increase.
- In matrix form a preconditioner matrix P of a matrix A is such that $P^{-1}A$ has a smaller condition number than A.

- Take for example the system $Ax = b$. Its right-preconditioned version is the system,
$$AP^{-1}(Px) = b.$$
We set $y \equiv Px$ so we can solve
$$AP^{-1}y = b,$$
and then,
$$x = P^{-1}y.$$
- The last step is merely a matrix multiplication by an already assembled matrix P^{-1}.
- It is also quite common to refer to P^{-1} as the preconditioner instead of P.

- Examples of common preconditioners are,
  - The preconditioned conjugate gradient method (PCG)
  - The biconjugate gradient method (biCG)
  - The generalized minimal residual method (GMRES)
  - The multigrid method, which is also a full-fledged iterative method
  - ...
- We will only study the multigrid method in this course.

The multigrid method

- The multigrid method is the most efficient and general iterative method available today, and serves even as a preconditioner.
- It was originally developed for elliptic systems like the linear diffusion equation but has since been successfully applied to many other problems, including the Euler and Navier-Stokes equations.
- It reduces the error on many scales by solving on coarse and fine versions of the same problem, which effectively remedies the poor performance of iterative methods with low frequency residual errors.
- It is also equally applicable to structured and unstructured meshes.

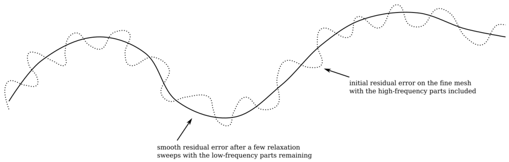
initial residual error on the fine mesh with the high-frequency parts included
smooth residual error after a few relaxation sweeps with the low-frequency parts remaining

Figure 3: Smoothing of the high-frequency residual errors into low-frequency errors.

- A typical residual error variation is shown in figure 3, where we can see both low and high frequency parts.
- With multigrid, a few iterations are first performed on the original mesh, damping out most of the high-frequency errors.
- This part is also called the relaxation sweeps, and leaves us with the lower frequency, *smooth* part of the error.
- The iterative method is acting here as an error *smoother* on the finest level of the mesh shown in figure 4 along with its coarse counterpart.

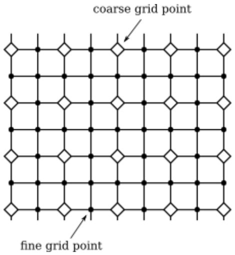

Figure 4: Original fine mesh and its coarsened counterpart.

- Then we interpolate the solution to the coarser mesh points, which is called the *restriction step*, and solve a few iterations at that level without taking the finer mesh points in consideration.
- Since the new mesh is coarser, the low frequency error that we interpolated from the finer original mesh become higher frequency errors on the coarse mesh.
- We now perform a few iterations on the coarse mesh, which dampens out the local high frequency error (previously the lower frequency error on the finer mesh).

- This coarsening cycle is repeated a few times with progressively coarser meshes to dampen all lower frequency errors.
- Assuming the mesh size of each coarse mesh is twice larger than its finer mesh, we will have half the points and the computational cost of iterating on the coarse mesh will be 4 times less.
- The more we coarsen the smaller the computational cost becomes, and the total cost of dampening all low frequency errors in all coarse meshes is actually a fraction of iterating exclusively on the finest mesh.
- At the coarsest levels we can even use a direct method which gives an exact solution to $A\mathbf{q} = b$.
- All in all the computational cost of a multigrid method is is order $O(N)$ for N mesh points and reduces the error at all frequencies.

- A typical multigrid V cycle is shown in figure 5.
- At the coarsest level we refine and interpolate the coarse solution back onto the finer mesh (the *prolongation step*) recovering the dampened low frequency errors.
- We then carry out relaxation sweeps and repeat until we reach the original finest mesh level with errors dampened at all frequencies.

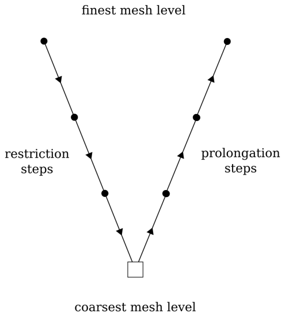

Figure 5: A multigrid V-cycle with restriction and prolongation steps.

- Another typical multigrid cycle, the W-cycle is shown in figure 6.
- Here, we insert another restriction step in the middle after the first prolongation step, and coarsen back down to finally prolong again.
- W-cycles can have several intermediate cycles and are more robust than V-cycles.
- The W-cycle in figure 6 will cost about 50% more than the V-cycle in figure 5 to compute.

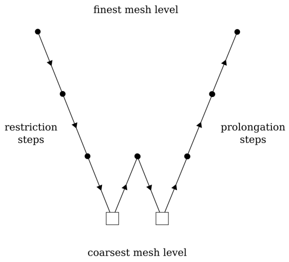

Figure 6: A multigrid W-cycle with restriction and prolongation steps.

Summary of the multigrid method

V-cycles

1 Perform relaxation sweeps at the finest mesh level.

2 Carry out a restriction step with some relaxation sweeps.

3 Repeat as many times as needed to cover the full error spectrum.

4 Perform the final relaxation sweeps at the coarsest level, optionally with a direct method.

5 Carry out a prolongation step with some relaxation sweeps.

6 Repeat as many times as needed until you reach the original mesh level.

W-cycles

6 Repeat the prolongation step a few times or carry out more restriction steps back to the coarsest level.

7 Carry out as many prolongation steps as needed to reach the original mesh level, or repeat another intermediate restriction phase.

Summary of direct and iterative methods

- Steady-state discretizations and implicit discretizations of transient problems result in large algebraic systems of equations that need to be solved in matrix form.
- Direct methods provide the exact solution to the discretized system in a number of arithmetic operations of order $O(N^3)$ and are prohibitively expensive for realistic engineering cases.
- Iterative methods iterate to a suitable approximation of the solution in a number of arithmetic operations of order $O(N^2)$.
- Iterative methods can dampen the high frequency part of the error spectrum fast, but perform poorly on the low frequency error component.

- Solution over-relaxation can help accelerate iterative method performance, with a stability penalty if the relaxation factor is close to or larger than 2.
- Unstable systems can be stabilized with under-relaxation and a relaxation factor below 1.
- Iterative method performance can also be enhanced with preconditioners, which reduces the condition number of the system matrix and makes the problem better conditioned.
- The multigrid method is a multi-scale iterative method and preconditioner.
- It works by iterating on successive coarser (restriction) and finer (prolongation) meshes, thus reducing the error at all frequencies.
- The multigrid method requires a number of arithmetic operations of order $O(N)$ and is the most general and successful iterative method known today.