Convergence acceleration techniques

Praveen. C
praveen@math.tifrbng.res.in

Tata Institute of Fundamental Research
Center for Applicable Mathematics
Bangalore 560065
http://math.tifrbng.res.in/~praveen

April 8, 2013

Time step restrictions

Time step of explicit methods restricted by stability condition, CFL condition.

Convection problems

$$\Delta t \leq \frac{\Delta x}{\lambda}, \qquad \lambda = \text{largest wave speed} = |u| + a$$

Diffusion problems

$$\Delta t \leq \frac{\Delta x^2}{2\nu}, \qquad \nu = \text{kinematic viscosity}$$

We solve unsteady Euler/NS equations even when we are interested in the steady state solution. We march the solution forward in time until steady solution is obtained. This is because unsteady Euler equations are hyperbolic at all Mach numbers and we can construct good schemes for hyperbolic equations.

Time step restrictions

Steady Euler equations are not of fixed type. In subsonic regions they are of elliptic type and in supersonic regions they are of hyperbolic type. Constructing schemes for this case is complicated.

Semi-discrete finite volume scheme

$$|C_j| \frac{\mathrm{d}U_j}{\mathrm{d}t} + R_j(U) = 0$$

We want the steady state solution U^∞ such that

$$R_j(U^\infty) = 0, \qquad j = 1, 2, \ldots, N_c$$

Forward Euler explicit scheme

$$|C_j| \frac{U_j^{n+1} - U_j^n}{\Delta t} + R_j(U^n) = 0$$

Time step restrictions

Update equation

$$U_j^{n+1} = U_j^n - \frac{\Delta t}{|C_j|} R_j(U^n)$$

Set initial condition U^0. For aerodynamic problems, we use free-stream conditions as initial conditions throughout the domain. Then apply update scheme repeatedly until convergence is achieved, e.g., the residual becomes small

$$\|R^n\| \leq TOL$$

where

$$\|R^n\| = \left(\sum_j [R_j(U^n)]^2 \right)^{\frac{1}{2}}$$

The convergence is usually measured with respect to the initial residual

$$\frac{\|R^n\|}{\|R^0\|} \leq TOL$$

Time step restrictions

For stability the time step should satisfy (assuming convection problem)

$$\Delta t \leq \frac{h_j}{\lambda_j}, \qquad \text{for } j\text{'th cell}$$

If we are doing a time accurate simulation, then we have to choose the smallest time step from all the cells

$$\Delta t \leq \min_j \frac{h_j}{\lambda_j}$$

which gives us the **global time step**

$$\Delta t = \mathrm{CFL} \min_j \frac{h_j}{\lambda_j}$$

The smallest cell determines the global time step.

Time step restrictions

If we want only steady state solution, we do not care for time accuracy. Then in each cell we can use the local allowed time step

$$\Delta t_j = \mathrm{CFL}\frac{h_j}{\lambda_j}, \qquad U_j^{n+1} = U_j^n - \frac{\Delta t_j}{|C_j|} R_j(U^n)$$

This is known as **local time stepping** and leads to faster convergence to steady state than global time stepping.

Time step restrictions

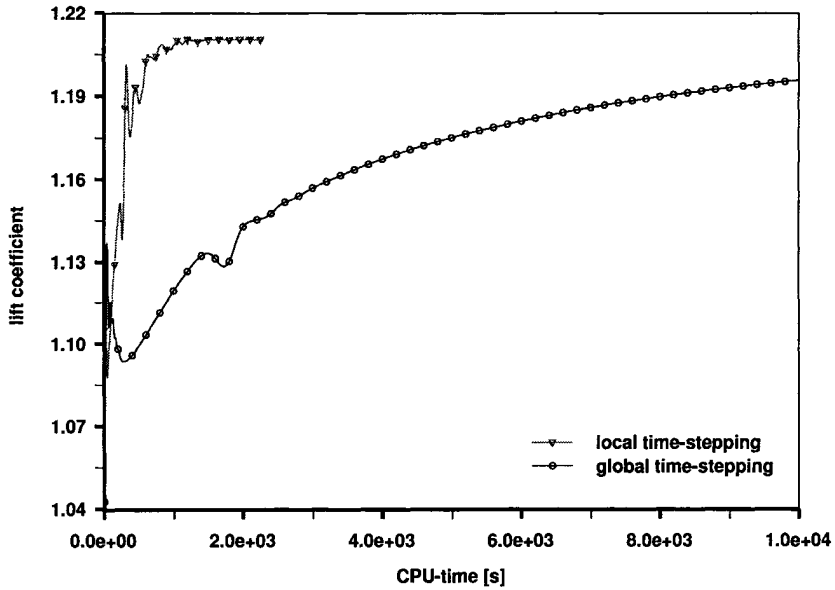

Figure 9.1: Comparison of convergence histories of the lift coefficient with and without local time-stepping for an inviscid subsonic flow on an unstructured grid.

Multi-stage RK scheme

$$
\begin{aligned}
\vec{W}_I^{(0)} &= \vec{W}_I^n \\
\vec{W}_I^{(1)} &= \vec{W}_I^{(0)} - \alpha_1 \frac{\Delta t_I}{\Omega_I} \vec{R}_I^{(0)} \\
\vec{W}_I^{(2)} &= \vec{W}_I^{(0)} - \alpha_2 \frac{\Delta t_I}{\Omega_I} \vec{R}_I^{(1)} \\
&\vdots \\
\vec{W}_I^{n+1} = \vec{W}_I^{(m)} &= \vec{W}_I^{(0)} - \alpha_m \frac{\Delta t_I}{\Omega_I} \vec{R}_I^{(m-1)}
\end{aligned}
$$

	first-order scheme			second-order scheme		
stages	3	4	5	3	4	5
σ	1.5	2.0	2.5	0.69	0.92	1.15
α_1	0.1481	0.0833	0.0533	0.1918	0.1084	0.0695
α_2	0.4000	0.2069	0.1263	0.4929	0.2602	0.1602
α_3	1.0000	0.4265	0.2375	1.0000	0.5052	0.2898
α_4		1.0000	0.4414		1.0000	0.5060
α_5			1.0000			1.0000

Table 6.1: Multistage scheme: optimised stage coefficients (α) and CFL numbers (σ) for first- and second-order upwind spatial discretisations.

Hybrid Multi-stage RK scheme

Residual is made of convective and dissipation flux (includes artificial dissipation)

$$\vec{R}_I = (\vec{R}_c)_I - (\vec{R}_d)_I$$

$$(\vec{R}_c)_I = \sum_{k=1}^{N_F} \left[\vec{F}_c(\vec{W}_{av})\Delta S\right]_k - (\vec{Q}\Omega)_I$$

$$(\vec{R}_d)_I = \sum_{k=1}^{N_F} \left[\vec{F}_v\,\Delta S + \vec{D}\right]_k ,$$

Diffusive flux is not computed in every stage. Example: (5,3)-scheme

Hybrid Multi-stage RK scheme

$$
\begin{aligned}
\vec{W}_I^{(0)} &= \vec{W}_I^n \\
\vec{W}_I^{(1)} &= \vec{W}_I^{(0)} - \alpha_1 \frac{\Delta t_I}{\Omega_I} \left[\vec{R}_c^{(0)} - \vec{R}_d^{(0)} \right]_I \\
\vec{W}_I^{(2)} &= \vec{W}_I^{(0)} - \alpha_2 \frac{\Delta t_I}{\Omega_I} \left[\vec{R}_c^{(1)} - \vec{R}_d^{(0)} \right]_I \\
\vec{W}_I^{(3)} &= \vec{W}_I^{(0)} - \alpha_3 \frac{\Delta t_I}{\Omega_I} \left[\vec{R}_c^{(2)} - \vec{R}_d^{(2,0)} \right]_I \\
\vec{W}_I^{(4)} &= \vec{W}_I^{(0)} - \alpha_4 \frac{\Delta t_I}{\Omega_I} \left[\vec{R}_c^{(3)} - \vec{R}_d^{(2,0)} \right]_I \qquad \vec{R}_d^{(2,0)} = \beta_3 \vec{R}_d^{(2)} + (1-\beta_3)\vec{R}_d^{(0)} \\
\vec{W}_I^{n+1} &= \vec{W}_I^{(0)} - \alpha_5 \frac{\Delta t_I}{\Omega_I} \left[\vec{R}_c^{(4)} - \vec{R}_d^{(4,2)} \right]_I , \qquad \vec{R}_d^{(4,2)} = \beta_5 \vec{R}_d^{(4)} + (1-\beta_5)\vec{R}_d^{(2,0)}
\end{aligned}
$$

Hybrid Multi-stage RK scheme

	central scheme $\sigma = 3.6$		1st-order upwind $\sigma = 2.0$		2nd-order upwind $\sigma = 1.0$	
stage	α	β	α	β	α	β
1	0.2500	1.00	0.2742	1.00	0.2742	1.00
2	0.1667	0.00	0.2067	0.00	0.2067	0.00
3	0.3750	0.56	0.5020	0.56	0.5020	0.56
4	0.5000	0.00	0.5142	0.00	0.5142	0.00
5	1.0000	0.44	1.0000	0.44	1.0000	0.44

Table 6.2: Hybrid multistage scheme: optimised stage (α) and blending (β) coefficients, as well as CFL numbers (σ) for central and upwind spatial discretisations. Note the identical coefficients for the 1st- and the 2nd-order upwind scheme but the different CFL numbers.

Implicit scheme

Backward Euler scheme

$$U_j^{n+1} = U_j^n - \frac{\Delta t}{|C_j|} R_j(U^{n+1})$$

Crank-Nicholson scheme

$$U_j^{n+1} = U_j^n - \frac{\Delta t}{|C_j|} \left[\frac{R_j(U^n) + R_j(U^{n+1})}{2} \right]$$

The unknown U^{n+1} is inside the non-linear term R. We have to solve this iteratively using some Newton-type method. For steady state problems, we need not solve for U^{n+1} exactly which allows us to make approximations in the Newton method.

Remark: One can try to solve $R(U) = 0$ directly using Newton method. But this problem is usually ill-conditioned and would require a good preconditioner.

Backward Euler scheme

Let us look at the backward Euler scheme. Write

$$U^{n+1} = U^n + \delta U^n, \qquad \delta U^n = U^{n+1} - U^n$$

Then we linearize the residual term

$$R_j(U^{n+1}) = R_j(U^n) + \frac{\partial R_j}{\partial U_j}(U^n)\delta U_j^n + \sum_k \frac{\partial R_j}{\partial U_k}(U^n)\delta U_k^n + \mathcal{O}\left(\delta U\right)^2$$

The summation on the right is over all cells in the stencil of j'th cell. For a first order scheme, the stencil involves only the neighbouring cells (maybe more for NS) while for second order scheme, the stencil contains neighbours of neighbours due to the reconstruction process.

For steady state problems, we can use the first order scheme to compute the Jacobian terms. So we linearise as

$$R_j(U^{n+1}) = R_j(U^n) + \frac{\partial R_j^{(1)}}{\partial U_j}(U^n)\delta U_j^n + \sum_{k\in N_j} \frac{\partial R_j^{(1)}}{\partial U_k}(U^n)\delta U_k^n$$

Backward Euler scheme

$$\left[\frac{|C_j|}{\Delta t} I + \frac{\partial R_j^{(1)}}{\partial U_j}\right] \delta U_j^n + \sum_{k \in N_j} \frac{\partial R_j^{(1)}}{\partial U_k} \delta U_k^n = -R_j(U^n)$$

We have one such equation from each cell and they form a coupled matrix equation.

$$A(U^n)\delta U^n = -R(U^n)$$

The most obvious way to solve them is using a Gauss-Jacobi, Gauss-Seidel method or Symmetric Gauss-Seidel method.

The matrix A is sparse since it mostly has zeroes. We can store only the non-zero blocks in the matrix in sparse format.

Remark: In the case of Navier-Stokes equations, the jacobians must include the viscous fluxes also.

Simplifications

The first order residual

$$R_j^{(1)} = \sum_{k \in N_j} H(U_j, U_k, n_{jk}) |S_{jk}|$$

Use Steger-Warming flux

$$H(U_j, U_k, n_{jk}) = A^+(U_j, n_{jk}) U_j + A^-(U_k, n_{jk}) U_k$$

Then Jacobians are approximated as

$$\frac{\partial R_j^{(1)}}{\partial U_j} = \sum_{k \in N_j} A^+(U_j, n_{jk}) |S_{jk}|, \qquad \frac{\partial R_j^{(1)}}{\partial U_k} = A^-(U_k, n_{jk}) |S_{jk}|$$

Lax-Friedrich's type flux

$$H(U_j, U_k, n_{jk}) = \frac{f_j + f_k}{2} \cdot n_{jk} - \frac{1}{2} \lambda_{jk} (U_k - U_j)$$

Simplifications

where λ_{jk} is maximum wave speed along n_{jk}, e.g.,

$$\lambda_{jk} = |u_{jk} \cdot n_{jk}| + a_{jk}$$

Then the split Jacobians are approximated as

$$A^+(U_j, n_{jk}) = \frac{1}{2}[A(U_j, n_{jk}) + \lambda_{jk} I]$$

$$A^-(U_k, n_{jk}) = \frac{1}{2}[A(U_k, n_{jk}) - \lambda_{jk} I]$$

This simplifies the Jacobian terms, e.g.,

$$\frac{\partial R_j^{(1)}}{\partial U_j} = \frac{1}{2} \sum_{k \in N_j} [A(U_j, n_{jk}) + \lambda_{jk} I] = \frac{1}{2} \left(\sum_{k \in N_j} \lambda_{jk} |S_{jk}| \right) I$$

Simplifications

The implicit scheme becomes

$$\left[\frac{|C_j|}{\Delta t} + \sum_{k \in N_j} \lambda_{jk} |S_{jk}|\right] \delta U_j^n + \sum_{k \in N_j} \frac{1}{2} [A(U_k, n_{jk}) - \lambda_{jk} I] \delta U_k^n = -R_j(U^n)$$

The coefficient of δU_j is a scalar which makes it easy to implement Jacobi/Seidel methods.

Structure of the implicit matrix

$$|C_j|\frac{U_j^{n+1}-U_j^n}{\Delta t}+R(U_{j-1}^{n+1},U_j^{n+1},U_{j+1}^{n+1})=0$$

$$\frac{\partial R}{\partial U_{j-1}}\delta U_{j-1}^n+\left[\frac{|C_j|}{\Delta t}I+\frac{\partial R}{\partial U_j}\right]\delta U_j^n+\frac{\partial R}{\partial U_{j+1}}\delta U_{j+1}^n=-R(U_{j-1}^n,U_j^n,U_{j+1}^n)$$

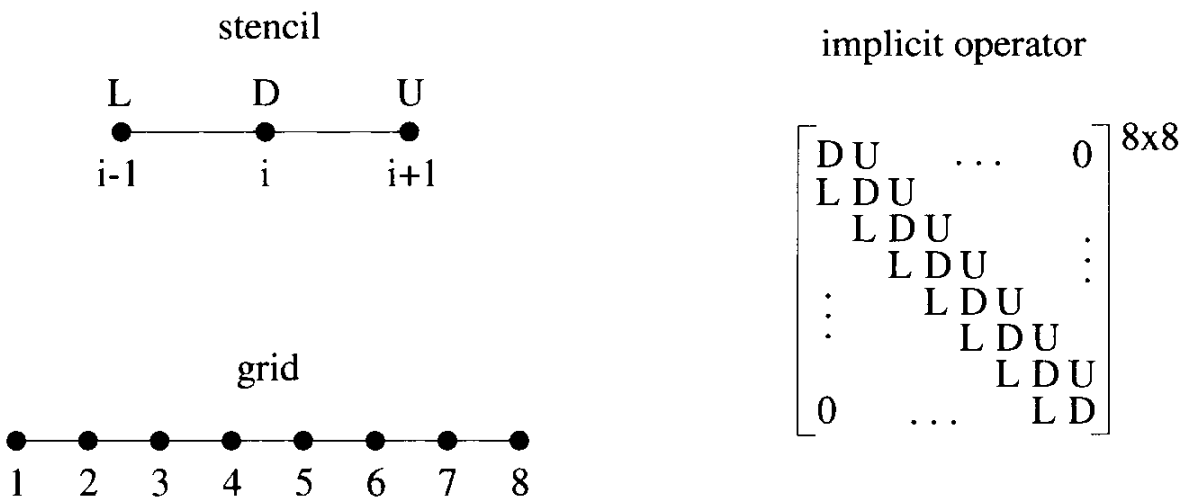

Figure 6.1: 1-D structured grid and the associated implicit operator matrix for a 3-point stencil.

Structure of the implicit matrix

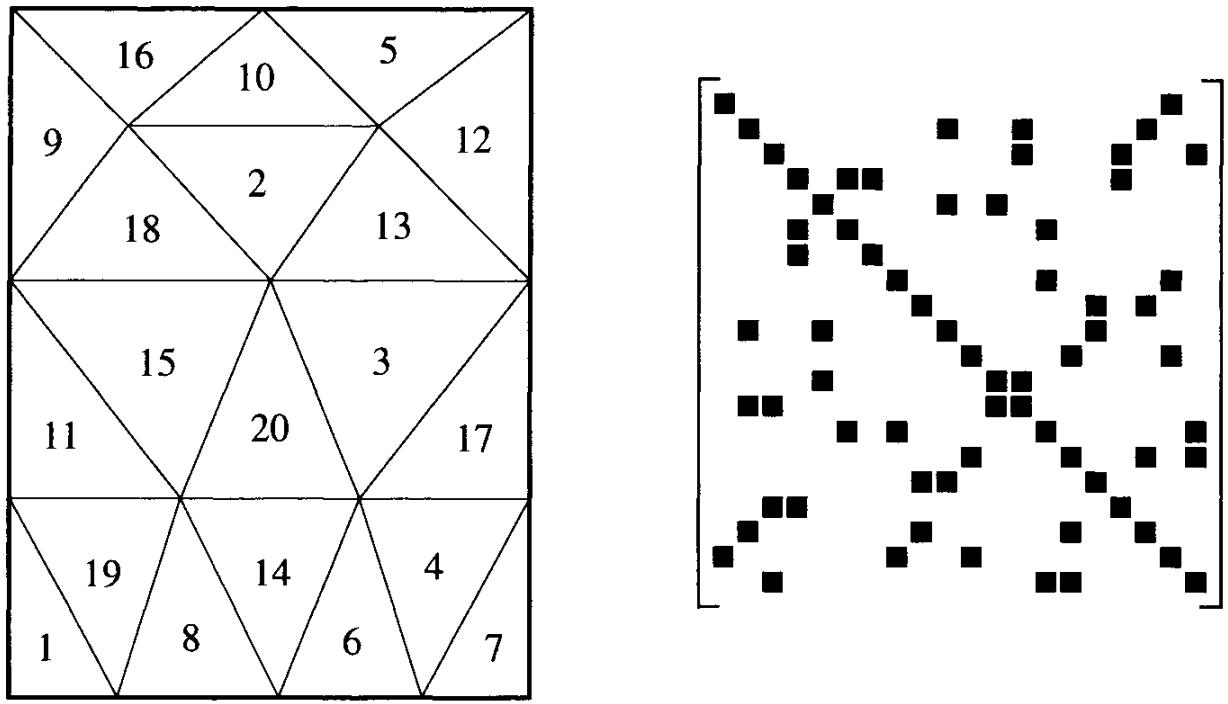

Figure 6.3: 2-D unstructured grid (left) and the associated implicit operator matrix for a nearest neighbour stencil (right). Nonzero block matrices displayed as filled rectangles.

Structure of the implicit matrix

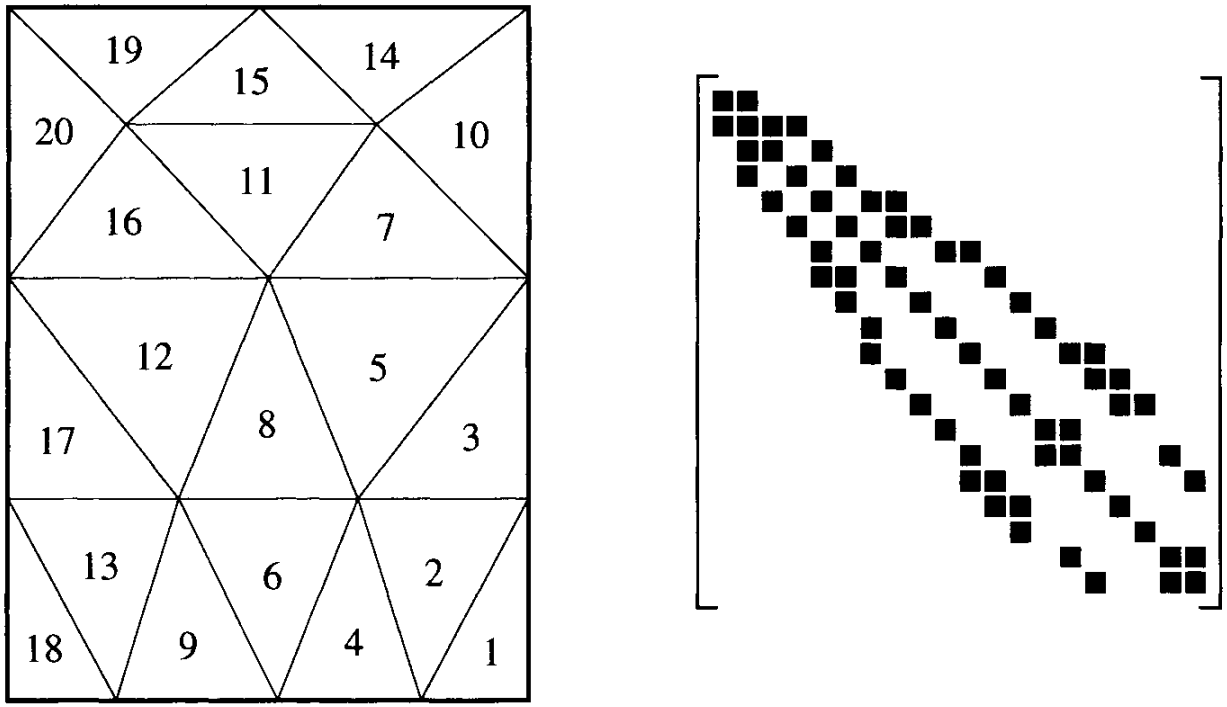

Figure 6.4: Reduced bandwidth (from 18 to 5) of the implicit operator from Fig. 6.3 with reverse-Cuthill-McKee ordering. Nonzero block matrices displayed as filled rectangles.

LU-SGS method

Consider the implicit matrix equation

$$A\delta U^n = -R(U^n)$$

where A might involve the Jacobian simplifications we have discussed.

$$A = L + D + U$$

We can approximately factorize matrix A

$$(L + D)D^{-1}(U + D)\delta U^n = -R(U^n)$$

Note that there is some error in the factorization

$$(L + D)D^{-1}(U + D) = A + LD^{-1}U$$

LU-SGS method

The factorised system can be solved in two steps

$$\begin{aligned}(L+D)\delta\bar{U} &= -R(U^n)\\ (U+D)\delta U^n &= D\delta\bar{U}\end{aligned}$$

First step $(L+D)\delta\bar{U} = -R(U^n)$. For j'th cell

$$A_{jj}\delta\bar{U}_j + \sum_{k\in L(j)} A_{jk}\delta\bar{U}_k = -R_j(U^n)$$

Forward loop: For $j = 1, 2, \ldots, N_c$

$$A_{jj}\delta\bar{U}_j = -R_j(U^n) - \sum_{k\in L(j)} A_{jk}\delta\bar{U}_k$$

Second step $(U+D)\delta U^n = D\delta\bar{U}$ which for j'th cell is

$$A_{jj}\delta U^n_j + \sum_{k\in U(j)} A_{jk}\delta U^n_k = A_{jj}\delta\bar{U}_j$$

LU-SGS method

Backward loop: For $j = N_c, N_c - 1, \ldots, 2, 1$

$$A_{jj}\delta U_j^n = A_{jj}\delta \bar{U}_j - \sum_{k \in U(j)} A_{jk}\delta U_k^n$$

Matrix-free LU-SGS method

Let

$$L(j) = \{C_k\text{: neighbour of } C_j \text{ such that } k < j\}$$

$$U(j) = \{C_k\text{: neighbour of } C_j \text{ such that } k > j\}$$

First step $(L + D)\delta\bar{U} = -R(U^n)$. For j'th cell

$$D_j\delta\bar{U}_j + \sum_{k\in L(j)} \frac{1}{2}[A(U_k, n_{jk}) - \lambda_{jk}I]\delta\bar{U}_k = -R_j(U^n)$$

$$A(U_k, n_{jk})\delta\bar{U}_k \approx F(U_k + \delta\bar{U}_k, n_{jk}) - F(U_k, n_{jk}) = \delta\bar{F}_k$$

$$D_j\delta\bar{U}_j + \sum_{k\in L(j)} \frac{1}{2}[\delta\bar{F}_k - \lambda_{jk}\delta\bar{U}_k] = -R_j(U^n)$$

The first step involves forward loop: $j = 1, 2, \ldots, N$

$$D_j\delta\bar{U}_j = -R_j(U^n) - \sum_{k\in L(j)} \frac{1}{2}[\delta\bar{F}_k - \lambda_{jk}\delta\bar{U}_k]$$

Matrix-free LU-SGS method

Second step $(U + D)\delta U^n = D\delta\bar{U}$ which for j'th cell is

$$D_j \delta U_j^n + \sum_{k \in U(j)} \frac{1}{2}[A(U_k, n_{jk}) - \lambda_{jk} I]\delta U_k^n = D_j \delta \bar{U}_j$$

$$A(U_k, n_{jk})\delta U_k^n \approx F(U_k + \delta U_k^n, n_{jk}) - F(U_k, n_{jk}) = \delta F_k$$

The second step involves: $j = N, N-1, \ldots, 2, 1$

$$D_j \delta U_j^n = D_j \delta \bar{U}_j - \sum_{k \in U(j)} \frac{1}{2}[\delta F_k - \lambda_{jk} \delta U_k^n]$$

Remark: For details about including viscous fluxes in the implicit scheme and further refinements, see Blazek, section 6.2.4 and the quoted references.

Unsteady simulations: dual time stepping

Crank-Nicholson, second order accurate

$$M\frac{U^{n+1}-U^n}{\Delta t}+\frac{1}{2}[R(U^n)+R(U^{n+1})]=0$$

Second order BDF time integration

$$M\frac{\frac{3}{2}U^{n+1}-2U^n+\frac{1}{2}U^{n-1}}{\Delta t}+R(U^{n+1})=0$$

Now we have to compute U^{n+1} exactly since want time accurate solution. We cannot perform approximate linearizations on this problem. But exact linearization will lead to a bigger matrix problem with more fillin. Define

$$R^*(U)=M\frac{\frac{3}{2}U-2U^n+\frac{1}{2}U^{n-1}}{\Delta t}+R(U)$$

Use pseudo-time integration: Solve for steady state solution of

$$M\frac{\partial U}{\partial \tau}+R^*(U)=0$$

Multigrid method

(For good introduction see: Briggs, Emden, McCormick, "Multigrid Tutorial", SIAM)

Classical iterative methods like Jacobi, LU-SGS, GMRES are effective in removing high frequency errors.

Basic idea of multi-grid: Construct a sequence of meshes

- Apply a smoother on the finest mesh
- Transfer solution (correction) to coarser mesh
- On the coarse mesh, the low frequency errors appear as high frequency
- Apply a smoother on the coarse mesh
- Go down to still coarser mesh
- ...
- On coarsest mesh, solve matrix problem exactly
- Transfer solution (correction) to finer mesh
- Do some post-smoothing
- ...

Multigrid method

Structured grids: Straightforward to generate coarser meshes.

Unstructured grids: Not easy to generate coarser meshes unless the fine mesh was itself obtained by successive refinement. In realistic CFD situations we usually only have the fine mesh. Then we can generate coarser meshes by **agglomerating** the cells to form bigger cells. This is the approach used in SU2.

Non-linear multigrid method

Backward Euler

$$M\frac{U^{n+1}-U^n}{\Delta t}+R(U^{n+1})=0$$

$$\frac{1}{\Delta t}MU^{n+1}+R(U^{n+1})=\frac{1}{\Delta t}MU^n$$

Denote the unknown solution $u_h = U^{n+1}$ on the finest mesh Ω_h

$$L_h(u_h) = f_h \qquad \text{in} \quad \Omega_h$$

Apply a few iterations of smoothing (usually 1 to 3) to obtain new approximation $\bar{u}_h$. But $L_h(\bar{u}_h) \neq f_h$. Define residual

$$L_h(\bar{u}_h) - f_h = r_h$$

r_h has smooth part of error. Improved solution

$$u_h = \bar{u}_h + c_h, \qquad L_h(u_h) = L_h(\bar{u}_h + c_h) = f_h$$

Non-linear multigrid method

$$L_h(\bar{u}_h + c_h) - L_h(\bar{u}_h) = f_h - L_h(\bar{u}_h) = -r_h$$

We will solve for the correction c_h on a coarser mesh Ω_H. We need to transfer the above problem onto Ω_H. This is achieved using a **restriction operator**.

Conservative restriction operator: used for residual

$$I_h^H : \Omega_h \to \Omega_H, \qquad r_H = I_h^H r_h$$

Interpolatory restriction operator: used for solution

$$\bar{I}_h^H : \Omega_h \to \Omega_H, \qquad u_H = \bar{I}_h^H \bar{u}_h + c_H$$

Coarse grid problem

$$L_H(\bar{I}_h^H \bar{u}_h + c_H) - L_H(\bar{I}_h^H \bar{u}_h) = -I_h^H r_h$$

$$L_H(u_H) = L_H(\bar{I}_h^H \bar{u}_h) - I_h^H r_h$$

Non-linear multigrid method

We apply a few iterations of the smoother to compute an approximation to u_H. This gives us the correction

$$c_H = u_H - \bar{I}_h^H \bar{u}_h$$

This correction term is also smooth since the high frequency errors have been eliminated. We now transfer the correction c_H to the finer mesh Ω_h and update the solution on fine mesh

$$\bar{u}_h^{new} = \bar{u}_h + I_H^h c_H$$

In practice we may have several levels of coarse meshes but the above procedure can be still applied. The levels can be arranged in a **V**-cycle or a **W**-cycle.

Remark: For the agglomeration procedure, see the SU2 AIAA paper and the quoted references.