6.10. **Multigrid method in terms of finite element methods.** We consider to solve the following discrete weak formulation : Find $u_h \in V_h$ such that

$$a(u_h, v_h) = (f, v_h) \quad \forall v_h \in V_h,$$

where V_h is the space of continuous piecewise linear functions on a grid of mesh size h. We assume that the mesh was obtained from a mesh size $H = 2h$ by joining the midpoints of each triangle to form 4 sub triangles. We denote $\mathcal{T}_h$ and $\mathcal{T}_H$ by the triangulation for the space V_h and V_H, respectively. Let $V_h = \text{span}\{\phi_1, \cdots, \phi_{n_h}\}$ and $V_H = \text{span}\{\psi_1, \cdots, \psi_{n_H}\}$. We also denote $\{x_i^h\}$ and $\{x_i^H\}$ be nodes for $\mathcal{T}_h$ and $\mathcal{T}_H$, respectively. In this setting, we can say that ϕ_j is the linear continuous piecewise function defined on the mesh $\mathcal{T}_h$ such that $\phi_j(x_i^h) = \delta_{ij}$ and for any $v_h \in V_h$, we have

$$v_h = \sum_{i=1}^{n_h} v_h(x_i^h)\phi_i.$$

We shall use the notation that $v_h^j = v_h(x_j^h)$ and $\widetilde{v}_h = (v^1, \cdots, v^{n_h})^T$. The discrete weak form can then be written in terms of the following

$$A_h \widetilde{u}_h = \widetilde{f}_h, \tag{6.10}$$

where $f_h^j = (f, \phi_j)$ for $j = 1, \cdots, n_h$ and $(A_h)_{ij} = a(\phi_j, \phi_i)$. To solve the equation (6.10), we shall basically apply the iterative method given by (given $\widetilde{u}_h^k$ and $B_h \approx A_h^{-1}$),

$$\widetilde{u}_h^{k+1} = \widetilde{u}_h^k + B_h(\widetilde{f}_h - A_h\widetilde{u}_h^k).$$

It is well-known that the classical iterative methods applied on the space V_h can handle error components whose frequencies are supported in domain of size h and it is slow to handle the frequencies that are supported in domain of size $2h$ scales and larger scales. This is because the classical iterative method can be interpreted as a sequence of local solvers that can handle errors that are supported on small scale of size h. Therefore, to detect larger scale error components, we shall have to use iterative methods defined on larger scale. To achieve this goal, we consider to construct two level scheme. Given $\widetilde{u}_h^k$, the $k-$th iterate on the fine grid with the mesh size h. We perform the following:

Step 1: Presmoothing : Apply few single step iterative methods to obtain

$$\widetilde{u} = \text{smoothing}(\widetilde{f}_h, \widetilde{u}_h^k, \#\text{presmoothing})$$

Remark 6.6. *Let $\overline{u}_h = \sum_{i=1}^{n_h} \widetilde{u}^i\phi_i$. This is an approximate solution to u_h in which $h-$scale errors are already controlled. This only contains large scale error component, which can not be handled in the fine grid.*

Step 2: Coarse grid correction : Compute $w_H \in V_H$ by solving the following equation

$$a(\overline{u}_h + w_H, v_H) = (f, v_H) \quad \forall v_H \in V_H$$

and update

$$u_h^{k+1} = \overline{u}_h + w_H.$$

One can add additional step :

Step 3: Postsmoothing : Apply few single step iterative methods to obtain

$$\widetilde{u}_h^{k+1} = \text{smoothing}(\widetilde{f}_h, \widetilde{u}_h^{k+1}, \#\text{postsmoothing})$$

To write it convenient for the implementation, we shall need to define inter grid transfer operator

$$I_h^H : V_H \mapsto V_h \quad \text{and} \quad I_H^h : V_h \mapsto V_H.$$

Note that for a fixed $1 \le j \le n_H$, there exists $\mathbf{c}_j = (c_j^1, \cdots, c_j^{n_h})^T \in \mathbb{R}^{n_h}$ such that

$$\psi_j = \sum_{i=1}^{n_h} c_j^i \phi_i.$$

Example 6.6. *Examples in 1D and 2D are presented here.*

We then consider the matrix $P = [\mathbf{c}_1, \cdots, \mathbf{c}_{n_H}] \in \mathbb{R}^{n_h \times n_H}$. We call this matrix as a prolongation matrix. We can also think of $R = P^T \in \mathbb{R}^{n_h \times n_H}$, which we call a restriction matrix. Using these matries, we can define the prolongation operator as follows : for any given $w_H \in V_H$,

$$\begin{aligned} I_h^H w_H &= w_H = \sum_{j=1}^{n_H} w^j \psi_j = \sum_{j=1}^{n_H} w^j \left(\sum_{i=1}^{n_h} c_j^i \phi_i \right) \\ &= \sum_{i=1}^{n_h} \left(\sum_{j=1}^{n_H} c_j^i w^j \right) \phi_i = \sum_{i=1}^{n_h} (P\widetilde{w})^i \phi_i \in V_h, \end{aligned}$$

where $\widetilde{w} = (w^1, \cdots, w^{n_H})^T$. We shall not use the specific form of the restriction operator $I_H^h : V_h \mapsto V_H$, therefore, we skip to write it. We are in a position to write the coarse grid correction step in terms of the matrix and vector notation as follows. The coarse grid equation is equivalent to write in the following form : Find $w_H \in V_H$ such that

$$a(w_H, v_H) = (f, v_H) - a(\overline{u}_h, v_H) \quad v_H \in V_H.$$

We set $w_H = \sum_{i=1}^{J} w^j \psi_j$ and convert the aforementioned equation to find $\widetilde{w}$. To make it possible, we note that

$$a(\psi_j, \psi_i) = (P^T A_h P)_{ji},$$

where $(A_h)_{k\ell} = a(\phi_\ell, \phi_k)$. We also note that

$$(f, \psi_j) = (f, \sum_{i=1}^{n_h} c_j^i \phi_i) = \sum_{i=1}^{n_h} c_j^i (f, \phi_i) = (P^T \widetilde{f}_h)^j.$$

Lastly, to compute $a(\overline{u}_h, \psi_j)$, we note that

$$a(\phi_i, \psi_j) = a(\phi_i, \sum_{k=1}^{n_h} c_j^k \phi_k) = \sum_{k=1}^{n_h} c_j^k a(\phi_i, \phi_k) = (P^T A_h)_{ji}.$$

Therefore, $a(\overline{u}_h, \psi_j) = ((P^T A_h)\widetilde{u})^j$ for $j = 1, \cdots, n_H$, where $\widetilde{u}$ is the vector representation of $\overline{u}_h$. We then arrive at the following :

$$P^T A_h P \widetilde{w} = P^T (\widetilde{f}_h - A_h \widetilde{u}).$$

I did up to here !

In the aforementioned algorithm, In the two level scheme, we computed the exact solution of the linear system on the coarse level. Instead, we could extend this idea by incorporating additional levels. To describe the complete algorithm for several levels, we first choose a coarse triangulation $\mathcal{T}_0$ with mesh size h_0. We then subdivide each triangle into four congruent triangles by joining the midpoints of the edges and denote by $\mathcal{T}_1$ the resulting triangulation. Continue in this manner, we produce meshes $\mathcal{T}_1, \mathcal{T}_2, \cdots, \mathcal{T}_J$. With each triangulation $\mathcal{T}_k$, we associate a space of continuous, piecewise linear polynomials that we denote by V_k. The problem we want to solve is : Find $u_J \in V_J$ such that

$$a(u_J, v) = (f, v) \quad \forall v \in V_J.$$

The kth level iteration : We let $MG(k, z_0, g)$ denote the approximate solution of the equation $A_k z = g$ obtained by the kth level iteration with initial guess z_0.

For $k = 0$, $MG(0, z_0, g) = A_0^{-1} g$, i.e., the solution obtained from a direct method. For $k \geq 1$, $M(k, z_0, g)$ is obtained recursively in 3 steps. Let $m_1 > 0$ and $m_2 \geq 0$ be integers and $p = 1$ or 2.

Step 1: Presmoothing : For $1 \leq \ell \leq m_1$,

$$z_\ell = z_{\ell-1} + B_k(g - A_k z_{\ell-1})$$

Step 2: Coarse grid correction :

(1) Form residual equation : $A_k w_k = \mathrm{r}_k = g - A_k z_{m_1}$
(2) Restriction of the residual : $\mathrm{r}_{k-1} = I_{k-1}^k \mathrm{r}_k$. I_{k-1}^k is the restriction operator.
(3) Solve the coarse grid residual equation from the recursive call : with $w_0 = 0$. For $1 \leq i \leq p$,

$$w_i = MG(k-1, w_{i-1}, r_{k-1}).$$

(4) Update $z_{m_1+1} = z_{m_1} + I_{k-1}^k w_p$.

Step 3: Postsmoothing : For $1 \leq \ell \leq m_2$, with $z_0 = z_{m_1+1}$,

$$z_\ell = z_{\ell-1} + B_k(g - A_k z_{\ell-1}).$$

The advantage of applying the multigrid method to approximately solve the linear system is that the work involved is proportional to the dimension of the finite element space and it could produce a uniformly convergent iterate for fairly general problems including the elliptic second order partial differential equation. We now consider another type of multigrid methods. The Full Multigrid method

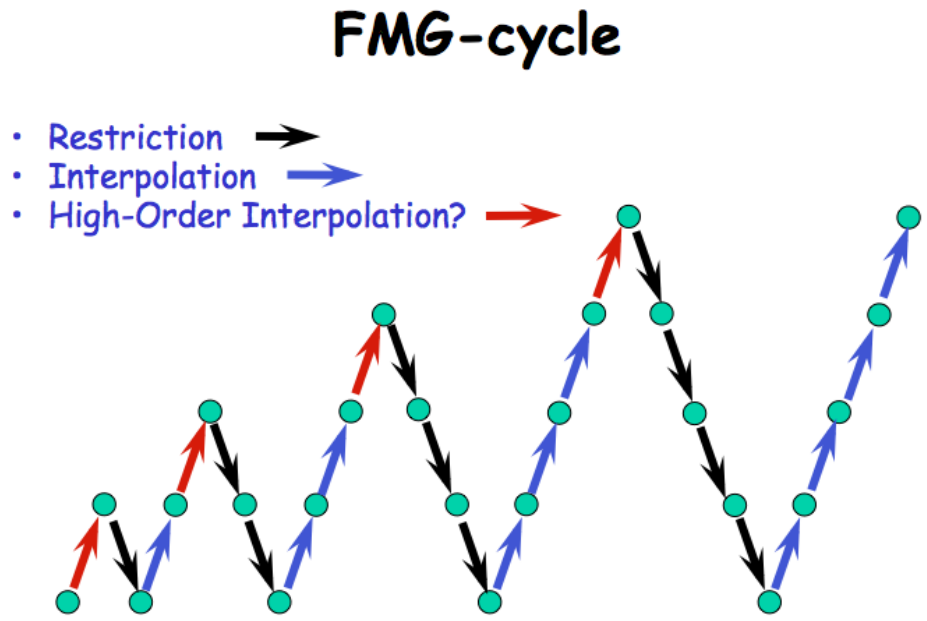

FIGURE 6.19. FMG : Full multigrid cycle

reads as follows :

For $k = 0$, let $\widehat{u}_0 = A_0^{-1} f_0$.

For $k = 1, \cdots, J$, the approximate solutions $\widehat{u}_k$ are obtained recursively by the following algorithm :

$$\begin{aligned} u_0^k &= I_{k-1}^k \widehat{u}_{k-1} \\ u_l^k &= MG(k, u_{l-1}^k, f_k) \quad 1 \leq l \leq r \\ \widehat{u}_k &= u_r^k. \end{aligned}$$

So we start at level 0 and at each finer level, we take the initial guess to be $I_{k-1}^{k}\widehat{u}_{k-1}$. For this type of algorithm, we have some nice convergence estimate, which can be referred to the Lecture note 14, by Professor Falk.