INTRODUCTION TO MULTIGRID METHODS

LONG CHEN

We give a short introduction to multigrid methods for solving the linear algebraic equation that comes from the discretization of the Poisson equation in one dimension.

Multigrid is among the most efficient iterative solvers. It can solve a linear system of size N with optimal complexity $\mathcal{O}(N)$. To see the large saving of an $\mathcal{O}(N)$ method compared with an $\mathcal{O}(N^2)$ one, consider the following estimate. Suppose $N = 10^6$ and a standard PC can sum 10^6 numbers in one second. Then an $\mathcal{O}(N)$ algorithm will finish in a few seconds, while an $\mathcal{O}(N^2)$ algorithm will take nearly 11.6 days.

1. Frequency and Aliasing

We consider the discretization of the Poisson equation with homogeneous Dirichlet boundary conditions in one dimension:

$$-u'' = f, \quad x \in (0,1), \qquad u(0) = u(1) = 0. \tag{1}$$

For a positive integer N, choose a uniform grid, denoted by $\mathcal{T}_h$, of the interval $[0,1]$ as follows:

$$0 = x_0 < x_1 < \ldots < x_N < x_{N+1} = 1, \quad x_i = ih,\ i = 0 : N+1,$$

where $h = 1/(N+1)$ is the length of each subinterval. For finite difference methods, we consider the vector $\boldsymbol{v} = (v_1, \ldots, v_N)^{\intercal}$, where $v_i \approx v(x_i)$ for $i = 1 : N$. The boundary nodes ($i = 0, N+1$) are excluded due to the homogeneous Dirichlet boundary condition. We can connect the function values at the grid points to form a piecewise linear function $v = \sum_{i=1}^N v_i \phi_i$, where ϕ_i is the hat basis function at x_i for $i = 1 : N$.

The algebraic system for a scaled finite element or finite difference discretization is

$$\boldsymbol{A}\boldsymbol{u} = \boldsymbol{b}, \tag{2}$$

where

$$\boldsymbol{A} = \operatorname{diag}(-1, 2, -1), \qquad \boldsymbol{b} = (b_i),\ b_i = h^2 f(x_i).$$

Due to the special structure of the matrix $\boldsymbol{A}$, we can write its eigenvalues and eigenvectors explicitly. Recall that the eigenpairs of the continuous problem

$$-u'' = \lambda u \quad \text{in } (0,1), \qquad u(0) = u(1) = 0$$

are

$$\lambda_k = k^2, \qquad \xi^k(x) = \sin(k\pi x), \qquad k = 1, 2, 3, \ldots$$

At the continuous level, $\lambda_{\min} = 1$ and $\lambda_k \to \infty$ as $k \to \infty$. At the discrete level, the $N \times N$ matrix $\boldsymbol{A}$ has only N eigenvalues, and therefore eigenpairs with $k > N$ cannot be represented. After the h^2 scaling, we have $\lambda_{\min}(\boldsymbol{A}) \to 0$ and $\lambda_{\max}(\boldsymbol{A}) \le 4$.

To better describe the eigenvalues of $\boldsymbol{A}$, we introduce a uniform grid in the angle interval $(0, \pi)$:

$$0 = \theta_0 < \theta_1 < \cdots < \theta_N < \theta_{N+1} = \pi, \qquad \theta_k = k\pi h,\ k = 0 : N+1.$$

Date: Latest update on November 23, 2025.

The discrete eigenvectors are obtained by applying the nodal interpolation I_h to the continuous eigenfunctions on the grid $\{x_i\}$, and the corresponding discrete eigenvalues are functions of the angles $\{\theta_k\}$. For a continuous function u, we define $I_h u(x_i) = u(x_i)$ for $i = 0, \dots, N+1$, so $I_h u$ is the discrete function that matches u at the grid points.

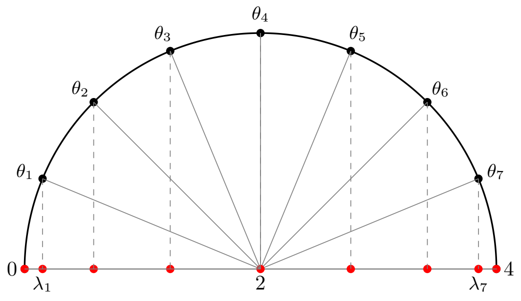


FIGURE 1. Eigenvalues of matrix $\boldsymbol{A}$ with $N = 7$.

Proposition 1.1. *If* $\boldsymbol{A} = \operatorname{diag}(b, a, b)$ *is an* $N \times N$ *tridiagonal matrix, then its eigenvalues are, for* $k = 1, \dots, N$,

$$\lambda_k = a + 2b\cos\theta_k,$$

and the corresponding eigenvectors are

$$\boldsymbol{\xi}^k = I_h \sin(k\pi x).$$

Proof. This can be verified directly by using the trigonometric identity

$$\sin((i-1)\theta) + \sin((i+1)\theta) = 2\cos(\theta)\sin(i\theta).$$

□

We use the index $i = 0 : N+1$ for the uniform discretization of $\Omega = (0,1)$ with mesh size h, so that $x_i = ih$. Similarly, we use the index $k = 0 : N+1$ for the uniform discretization of the angle interval $[0, \pi]$ with step size πh, that is, $\theta_k = k\pi h$. For each k, the eigenvector $\boldsymbol{\xi}^k$ can be viewed as a discrete function whose nodal values are given by

$$\boldsymbol{\xi}^k(x_i) = \sin(k\pi x_i) = \sin(k\pi h i) = \sin(i\theta_k), \qquad i = 0, \dots, N+1.$$

It is interesting to note that the eigenvectors are independent of a and b. In the special case $a = 2$ and $b = -1$, we obtain

$$\lambda_k(\boldsymbol{A}) = 2\big(1 - \cos\theta_k\big) = 4\sin^2\frac{\theta_k}{2}. \tag{3}$$

Notice that $\lambda_1 = \mathcal{O}(h^2)$ and $\lambda_N = \mathcal{O}(1)$ (see Fig. 1), and therefore $\kappa(\boldsymbol{A}) = \mathcal{O}(h^{-2})$, i.e., the matrix $\boldsymbol{A}$ is ill-conditioned. For the finite difference method, the corresponding matrix is $\boldsymbol{A}/h^2$, and for the finite element method it is $\boldsymbol{A}/h$. Such scalings multiply all eigenvalues by the same positive factor and thus do not change the condition number.

Exercise 1.2. Apply Proposition 1.1 to the mass matrix $M = (m_{ij})$ with $m_{ij} = \int_0^1 \phi_i \phi_j \, \mathrm{d}x$ and conclude that M is well-conditioned. □

Exercise 1.3. Determine the eigenvalues and eigenvectors for the Neumann problem $-u'' = \lambda u$ in $(0,1)$ with $u'(0) = u'(1) = 0$. □

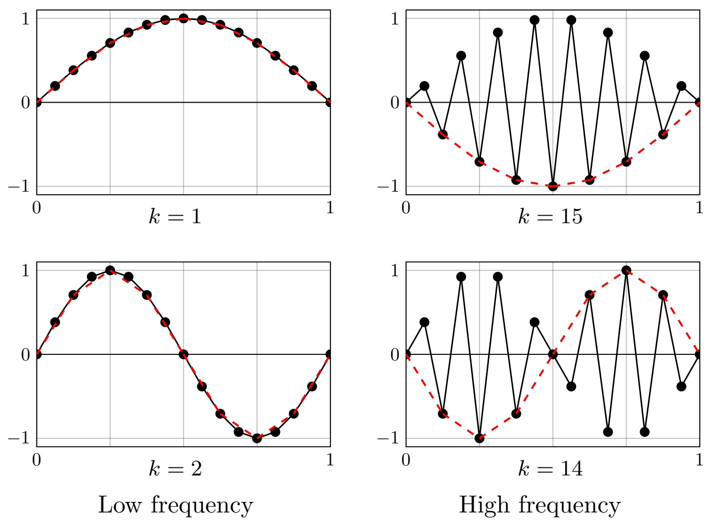


FIGURE 2. Selected eigenvectors of A for $h = 1/16$. Left: low-frequency modes that can be preserved on a coarse grid $\mathcal{T}_{2h}$. Right: high-frequency modes that will alias with low-frequency ones on the coarse grid $\mathcal{T}_{2h}$.

The integer k in the function $\sin(k\pi x)$ is called the *frequency*. For a uniform grid of $[0, 1]$ with mesh size $h = 1/(N + 1)$, the range of k is $1, \dots, N$. At the continuous level, there are infinitely many frequencies associated with all eigenfunctions, but frequencies higher than N cannot be represented on the grid $\mathcal{T}_h$. This limitation is known as *aliasing*.

To illustrate the aliasing phenomenon, consider a coarse grid with mesh size $2h$. When restricted to the coarse grid $\mathcal{T}_{2h}$, certain frequencies in the fine grid $\mathcal{T}_h$ become indistinguishable from others. This can be verified by connecting coarse-grid values in Fig. 1. Specifically, for $k = 1 : N,\ k' = N + 1 - k$, so that $\theta_k + \theta_{k'} = \pi$, then

$$I_{2h}(\boldsymbol{\xi}^k) = I_{2h}(-\boldsymbol{\xi}^{k'}).$$

Indeed, this can be verified as follows:

$$\begin{aligned} I_{2h}\sin(k\pi x) &= \big(\sin(k\pi i 2h)\big)_{i=1}^{N_c} = \big(\sin(i\, 2\theta_k)\big)_{i=1}^{N_c}, \\ I_{2h}\sin(k'\pi x) &= \big(\sin(k'\pi i 2h)\big)_{i=1}^{N_c} = \big(\sin(i\, 2\theta_{k'})\big)_{i=1}^{N_c} = -\big(\sin(i\, 2\theta_k)\big)_{i=1}^{N_c}, \end{aligned}$$

using $\theta_k + \theta_{k'} = \pi$. Note that the eigenvalues on the coarse grid still range within $(0, 4)$, but the highest resolvable frequency of the eigenvectors is restricted by the mesh size.

In general, the grid $\mathcal{T}_h$ cannot see frequencies higher than $1/h$. Therefore, the coarser grid $\mathcal{T}_{2h}$ can only resolve frequencies less than $1/(2h)$. The frequencies that can be captured by the fine grid $\mathcal{T}_h$ but not by the coarse grid $\mathcal{T}_{2h}$ lie in the range

$$\frac{1}{2}(N + 1) \le k \le N,$$

and will be referred to as the *high-frequency* components (relative to $\mathcal{T}_{2h}$). We shall show that classical iterative methods quickly smooth out the high-frequency part of the error, while the low-frequency part decays much more slowly.

2. Smoothing Property of the Richardson Iteration

As an illustrative example, we apply the simplest iterative method, the Richardson iteration, to solve (2). Recall that one iteration of the Richardson method is

$$\boldsymbol{u}_{m+1} = \boldsymbol{u}_m + \alpha(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{u}_m).$$

The corresponding error equation is

$$\boldsymbol{u} - \boldsymbol{u}_{m+1} = (I - \alpha \boldsymbol{A})(\boldsymbol{u} - \boldsymbol{u}_m),$$

and therefore,

$$\|\boldsymbol{u} - \boldsymbol{u}_{m+1}\|_{\boldsymbol{A}} \leq \rho(I - \alpha\boldsymbol{A})\, \|\boldsymbol{u} - \boldsymbol{u}_m\|_{\boldsymbol{A}}.$$

When $\alpha \in (0,\, 2/\lambda_{\max}(\boldsymbol{A}))$, the iteration is convergent, and when $\alpha = 2/(\lambda_{\min}(\boldsymbol{A}) + \lambda_{\max}(\boldsymbol{A}))$, it achieves the optimal convergence rate

$$\rho = \frac{\kappa(\boldsymbol{A}) - 1}{\kappa(\boldsymbol{A}) + 1} \leq 1 - Ch^2.$$

This means that the total error, viewed as the sum of the squared contributions from all frequency components, decays at an overall rate of only $1 - Ch^2$. Here, we use m as the iteration index, since k is reserved for the frequency index.

We now apply a refined analysis to obtain different decay rates for different frequency components. First, we change the coordinate system. The set of eigenvectors of $\boldsymbol{A}$ forms an orthogonal basis of $\mathbb{R}^N$ with respect to both the Euclidean inner product $(\cdot,\cdot)$ and the energy inner product $(\cdot,\cdot)_A$. We expand the initial error $\boldsymbol{e}_0 = \boldsymbol{u} - \boldsymbol{u}_0$ as

$$\boldsymbol{e}_0 = \sum_{k=1}^{N} c_k \boldsymbol{\xi}^k.$$

After m Richardson iterations,

$$\boldsymbol{e}_m = \boldsymbol{u} - \boldsymbol{u}_m = (I - \alpha\boldsymbol{A})^m \boldsymbol{e}_0 = \sum_{k=1}^{N} c_{m,k} \boldsymbol{\xi}^k,$$

where

$$c_{m,k} = (1 - \alpha\lambda_k)^m c_k.$$

Since the eigenvectors are A-orthogonal, we have

$$\|\boldsymbol{e}_m\|_A = \left(\sum_{k=1}^{N} c_{m,k}^2 \|\boldsymbol{\xi}^k\|_A^2\right)^{1/2}, \qquad \|\boldsymbol{e}_0\|_A = \left(\sum_{k=1}^{N} c_k^2 \|\boldsymbol{\xi}^k\|_A^2\right)^{1/2}.$$

Therefore, the coefficient of the k-th frequency component decays at the rate

$$|c_{m,k}| \leq (1 - \alpha\lambda_k)^m |c_k|.$$

We choose $\alpha = 1/4$ to simplify the rate as

$$\rho_k = |1 - \alpha\lambda_k| = \tfrac{1}{2}(1 + \cos\theta_k).$$

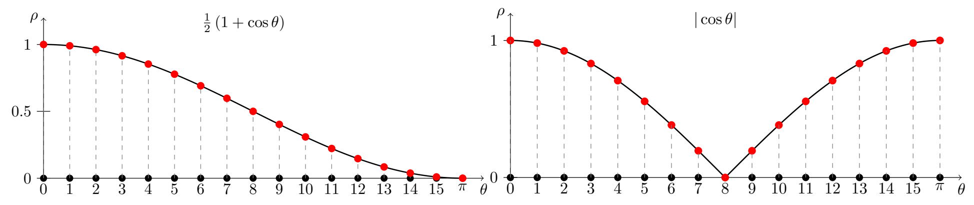
From the graph of ρ_k (see Fig. 3(a)), it is easy to see that

$$\rho_1 \eqsim 1 - Ch^2,$$

while

$$\rho_N \leq Ch^2, \qquad \rho_{(N+1)/2} = \tfrac{1}{2}.$$

This shows that *high-frequency components are damped very quickly*, a feature known as the smoothing property, whereas low-frequency components converge very slowly.



(a) Richardson iteration: $\alpha = 1/4$ (b) Richardson iteration: $\alpha = 1/2$

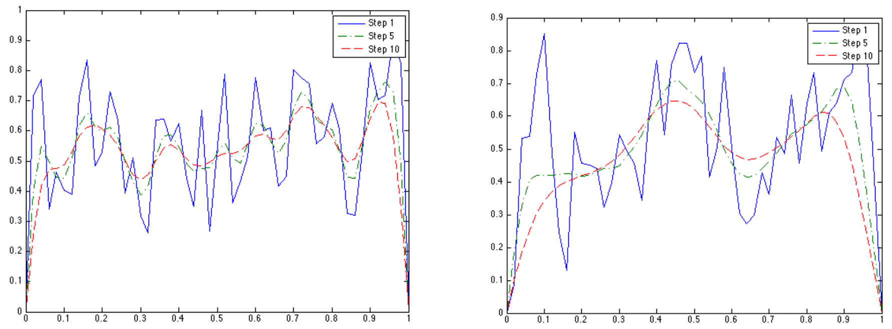
FIGURE 3. Contraction rate of the Richardson iteration for different parameters

The Gauss–Seidel method, as a more effective iterative scheme, shows the same behavior. In practice, it serves as a better smoother than the Richardson method. However, the analysis of its smoothing property is considerably more difficult.

Not all convergent iterative methods have the smoothing property. For example, the Jacobi method, which is equivalent to $\alpha = 1/2$ in the Richardson method, does not have a smoothing effect; see Fig. 4(b). For the same example, the decay rate for $\alpha = 1/2$ is

$$\rho_k = |\cos\theta_k|, \qquad \rho_1 = \rho_N \approx 1 - Ch^2.$$

As a smoother, the weighted Jacobi method αD^{-1} (with recommend weigtht $\alpha = 2/3$) is more effective and is therefore used more often in practice.



(a) Richardson iteration $\alpha = 1/4$ (b) Gauss-Seidel iteration

FIGURE 4. Smoothing effect of classic iterative methods.

3. MULTIGRID METHODS

The multigrid method is built on two basic observations:

- High–frequency errors are effectively damped by a smoother.
- Low–frequency errors can be well approximated on a coarse grid.

A key idea is that part of the low–frequency error on a fine grid becomes a high–frequency error on a coarser grid. On the coarse grid, we can again apply smoothing and scale separation. A recursive use of this process across multiple grid levels gives the classical multigrid method.

We present below the recursive subroutine of a multigrid method.

```
 1  function e = MG(r,J,mu)
 2  % r: residual;  J: level; mu: smoothing steps
 3  if J == 1  % coarsest level: exact solve
 4      e = A{J}\r;
 5      return
 6  end
 7  e = 0;
 8  % Presmoothing
 9  for i = 1:mu
10      e = e + R(r-A{J}*e);
11  end
12  % Restriction
13  rc = Res(r-A{J}*e);
14  % Coarse grid correction
15  ec = MG(rc,J-1,mu);
16  if W-cycle
17      ec = ec + MG(rc-A{J}*ec,J-1,mu);  % W-cycle
18  end
19  % Prolongation
20  e = e + Pro(ec);
21  % Postsmoothing
22  for i = 1:mu
23      e = e + R'(r-A{J}*e);
24  end
```

The operator (or subroutine) `R` used in the pre-smoothing step can be chosen as the Richardson, weighted Jacobi, or Gauss–Seidel iteration. In the post-smoothing step, we use its transpose `R'` to make the overall iteration symmetric. Two additional subroutines, `Res` and `Pro`, which connect the fine and coarse levels, will be discussed in detail in *Programming of Multigrid Methods*.

The function call `e = MG(r,*,*)` indicates that an `MG` cycle is used to solve the residual equation $Ae = r$ and serves as an iterator in the residual-correction form:

$$u_{k+1} = u_k + MG(f - Au_k).$$

Because the iteration is linear, we can also write a direct-update version of the multigrid method as `u = MG(u,f,J,mu)`; see *Programming of Multigrid Methods*.

The above `MG` scheme is the *V-cycle*. If we apply one additional coarse-grid correction using `MG` (lines 16–18), we obtain the *W-cycle*. The two typical multigrid cycles are illustrated in Figure 5.

With one more coarse-grid correction, the W-cycle becomes more robust, but the computational work increases accordingly. For a hierarchy of nested finite element spaces $\{\mathbb{V}_i\}_{i=0}^J$ with $\dim \mathbb{V}_i = N_i$ and refinement ratio $N_i = \gamma N_{i-1}$ for some $\gamma > 1$, the total work of a multigrid cycle is obtained by adding the work over all levels. A single V-cycle performs one descent and one ascent through the levels, so

$$T_V = c\sum_{i=0}^{J} N_i \leq \frac{c}{1-\gamma^{-1}} N_J,$$

based on the geometric relation $N_i = \gamma^{i-J} N_J$. Thus the V-cycle has complexity $\mathcal{O}(N_J)$ for any $\gamma > 1$.

A W-cycle recursively visits each coarse level twice, which leads to

$$T_W(i) = cN_i + 2\,T_W(i-1),$$

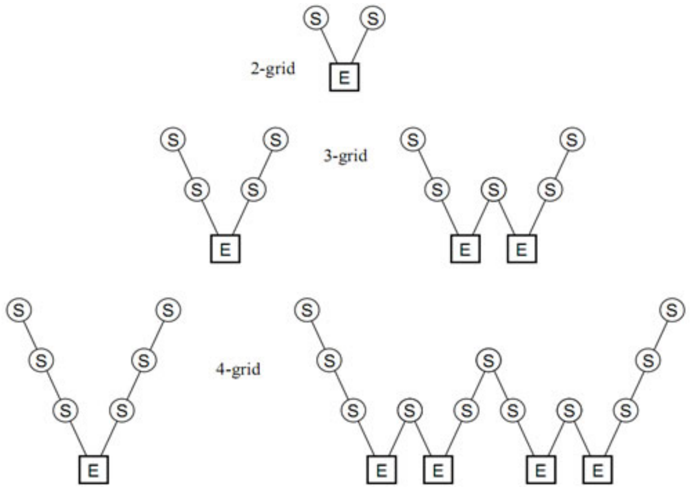


FIGURE 5. Multigrid Cycles. Left: V-cycle. Right: W-cycle

and the closed form

$$T_W = cN_J \sum_{k=0}^{J} (2/\gamma)^k.$$

If $\gamma > 2$ (for example, $\gamma \approx 2^d$ when $d \geq 2$), the geometric series stays bounded, and $T_W = \mathcal{O}(N_J)$. When $\gamma = 2$, the series grows like $J \sim \log N_J$, giving

$$T_W = \mathcal{O}(N_J \log N_J).$$

Hence, for a W-cycle, it is preferable that the coarsening ratio $\gamma = N_i/N_{i-1}$ satisfies $\gamma \geq 2$.

4. INTERPRETATION AS SUCCESSIVE SUBSPACE CORRECTION METHOD

We can interpret the multigrid method as a special case of the successive subspace correction method by choosing a multilevel decomposition of the space:

$$\mathbb{V}_1 \subset \mathbb{V}_2 \subset \cdots \subset \mathbb{V}_J = \mathbb{V},$$

together with a space decomposition

$$\mathbb{V} = \sum_{i=1}^{J} \mathbb{V}_i. \tag{4}$$

We introduce the following operators:

- $I_i : \mathbb{V}_i \hookrightarrow \mathbb{V}$, the natural inclusion;
- $Q_i : \mathbb{V} \mapsto \mathbb{V}_i$, the projection in the inner product $(\cdot,\cdot)$;
- $A_i : \mathbb{V}_i \mapsto \mathbb{V}_i$, the restriction of A to the subspace $\mathbb{V}_i \times \mathbb{V}_i$;
- $R_i : \mathbb{V}_i \mapsto \mathbb{V}_i$, an approximation of A_i^{-1}, often called a *smoother*.

By definition,

$$(Q_i^{\intercal} v_i, v) = (v_i, Q_i v) = (v_i, v) = (I_i v_i, v), \qquad \forall\, v_i \in \mathbb{V}_i,\ v \in \mathbb{V},$$

therefore $Q_i^{\intercal}$ coincides with the natural inclusion I_i, or equivalently, $I_i^{\intercal} = Q_i$. Hence, the restriction of A to subspaces can be written as

$$A_i = I_i^{\intercal} A I_i = Q_i A Q_i^{\intercal}.$$

Given the current iterate u_k, one step of the successive (multiplicative) subspace correction (SSC) method is defined as follows: for $i = J, \ldots, 1$, we approximately solve

$$A_i e_i = r_i, \qquad \text{with} \quad r_i = Q_i(f - Av^i),$$

and use the smoother $e_i = R_i r_i$.

Given the current iterate u_k, one step of the successive (multiplicative) subspace correction (SSC) method can be carried out without returning to the finest space at each subspace correction when the subspaces are nested.

Suppose the residual in $\mathbb{V}_i$ is $r_i = Q_i(r - Ae_{\text{old}})$, and the correction e_i is used to update $e_{\text{new}} = e_{\text{old}} + e_i$. We can then compute r_{i-1} recursively:

$$\begin{aligned} r_{i-1} &= Q_{i-1}(r - Ae_{\text{new}}) \\ &= Q_{i-1}Q_i(r - Ae_{\text{old}} - Ae_i) \\ &= Q_{i-1}(r_i - Q_i A Q_i^{\intercal} e_i) \\ &= Q_{i-1}(r_i - A_i e_i). \end{aligned}$$

In the second line, we used the nestedness $\mathbb{V}_{i-1} \subset \mathbb{V}_i$, which gives $Q_{i-1} = Q_{i-1}Q_i$.

The correction can also be updated recursively:

$$e_i = e_i + I_{i-1} e_{i-1}, \qquad i = 2, \ldots, J.$$

Thus only the prolongation and restriction operators between consecutive levels are required. The work on each level is $\mathcal{O}(N_i)$, and the total cost is $\mathcal{O}(N)$ since $\sum_i N_i = \mathcal{O}(N)$.

From this viewpoint, applying the SSC method to a nested space decomposition naturally produces a V-cycle multigrid method. This interpretation not only clarifies the convergence mechanism but also leads to the following non-recursive implementation of the multigrid V-cycle.

```
function e = Vcycle(r,J)
ri = cell(J,1); ei = cell(J,1);
ri{J} = r;
for i = J:-1:2
    ei{i} = R{i}*ri{i};    % pre-smoothing: one step
    ri{i-1} = Res{i-1}*(ri{i}-Ai{i}*ei{i}); % update and restrict residual
end
ei{1} = Ai{1}\ri{1}; % exact solver in the coarsest level
for i = 2:J
    ei{i} = ei{i} + Pro{i}*ei{i-1};  % prolongation and correction
    ei{i} = ei{i} + R{i}'*(ri{i}-Ai{i}*ei{i}); % post-smoothing: one step
end
e = ei{J};
```

We refer to *Subspace Correction Method and Auxiliary Space Method* for a detailed explanation and convergence analysis of subspace correction methods, and to *Programming of Multigrid Methods* for implementation details.