Reference Manual for Multilevel Iterative Methods

HW 1.6. Implementation of multigrid in 1D using Matlab, C, Fortran, or Python. Try to study the efficiency of your implementation.

Instruction. Consider the 1D Poisson's equation with the Dirichlet boundary conditions:

$$-u'' = f, \quad \text{in } (0,1) \quad \text{and} \quad u(0) = u(1) = 0.$$

Consider its corresponding finite difference scheme, namely

$$A\vec{u} = \vec{f}, \quad A = \frac{1}{h^2}\text{tridiag}(-1, 2, -1), \quad f_i = f(x_i),$$

where h represents the meshsize of a uniform grid.

Suppose there are a hierarchy of $L+1$ grids with mesh sizes $h_l = \frac{1}{2^{l+1}} (l = 0, 1, ..., L)$, respectively; see Figure 1.

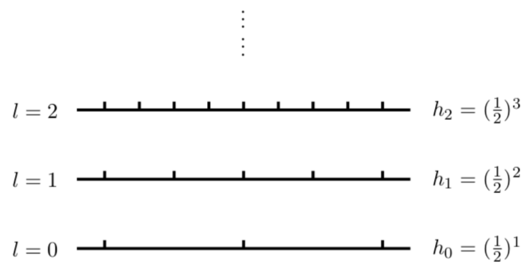

Figure 1: Hierarchical grids for 1D multigrid method.

One iteration of the multigrid algorithm is given as follows:

Algorithm 1 (One iteration of multigrid method). $\vec{u}_l = MG(l, \vec{f}_l, \vec{u}_l)$

(i) **Pre-smoothing**: $\vec{u}_l \leftarrow \vec{u}_l + \frac{1}{2}D_l^{-1}(\vec{f}_l - A_l\vec{u}_l)$

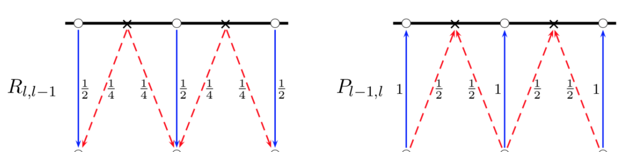

Figure 2: Restriction and prolongation between two consecutive levels.

(ii) **Restriction**: $\vec{r}_{l-1} \leftarrow \vec{R}_{l,l-1}(\vec{f}_l - A_l\vec{u}_l)$

(iii) **Coarse-grid correction**: $\vec{e}_{l-1} \leftarrow MG(l-1, \vec{r}_{l-1}, \vec{0}_{l-1})$

(iv) **Prolongation**: $\vec{u}_l \leftarrow \vec{u}_l + \vec{P}_{l-1,l}(\vec{e}_{l-1})$

(v) **Post-smoothing**: $\vec{u}_l \leftarrow \vec{u}_l + \frac{1}{2} D_l^{-1}(\vec{f}_l - A_l \vec{u}_l)$

Coarsest-level solver. In our tests, the solution at level $l = 0$ is solved directly.

Stopping criterion. $\|\vec{r}\|_2 / \|\vec{f}\|_2 \leqslant 10^{-6}$.

We now give a matrix-free implementation in Matlab of the above 1D multigrid method. Here the term matrix-free means that the matrix-vector product Au is implemented without assembling A explicitly, same applies for the restriction R and prolongation P. Note that, in order to update all components u_i in a uniform way, we first extend the vector u to $[0; u; 0]$. One can also do this differently.

Smoother on the fine space. One step of Gauss–Seidel iteration can be implemented as

```
for i=2:N+1
    u(i) = (h^2*fl(i)+u(i-1)+u(i+1))/2;
end
```

Restriction to the coarse space. Firstly, Au can be computed by the following Matlab code on all interior nodes.

```
Au(2:N) = (2*u(2:N)-u(1:N-1)-u(3:N+1))/h^2;
```

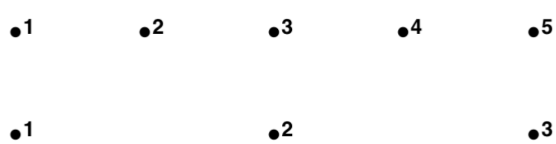

Figure 3: Indices of a coarse and fine grids.

Then forming the residual and the restriction step will be

```
    rf = zeros(N+2,1);
    i = 2:N+1; j = 2:Nc+1;
    rf(i) = fl(i)+(-2*u(i)+u(i-1)+u(i+1))/h^2;
    rc = zeros(Nc+2,1);
    rc(j) = 0.5*rf(2*j-1)+0.25*rf(2*j-2)+0.25*rf(2*j);
```

Prolongation back to the fine space. It is easy to figure out the index map from the coarse level to the fine level: $i \to 2i - 1$; see Figure 3. We use the linear interpolation to construct the prolongation. Then, the matrix-free implementation is as follows

```
    Pro_ec = zeros(N+2,1);
    Pro_ec(1:2:N+2) = ec;
    Pro_ec(2:2:N+1) = 0.5*ec(1:Nc+1)+0.5*ec(2:Nc+2);
```

where the item ec is the coarse correction computed by calling MG(l-1, rc, $\vec{0}$).

Now we perform a few simple numerical tests. Let the exact solution $u(x) = \sin(\pi x)$ and the forcing term $f(x) = \pi^2 \sin(\pi x)$. We solve the exact solution $\vec{u}$ with the Matlab direct solver for comparison. Initial $\vec{u}_0$ is taken to be $\vec{0}$. Suppose that the coarsest level is $l = 0$ and L stands for the finest level. The Gauss–Seidel smoother is used and the three smoothing steps are used. We compute the relative error $\frac{\|\vec{u}-\vec{u}_{\mathrm{MG}}\|_2}{\|\vec{u}\|_2}$ as well as $\frac{\|\vec{u}-\vec{u}_{\mathrm{GS}}\|_2}{\|\vec{u}\|_2}$ in each iteration.

For the multigrid method, the errors decrease very quickly to a small value, and we take logarithm of the relative errors and observe their change with iterations; see Figure 4 and Table 1. Moreover, the figure also indicates that the error converges uniformly with respect to the meshsize h. On the other hand, for the Gauss–Seidel method, even if the number of iteration steps reaches 1000, the error is still large; see Figure 5. Moreover, as the number of levels increases, the convergence speed deteriorates.

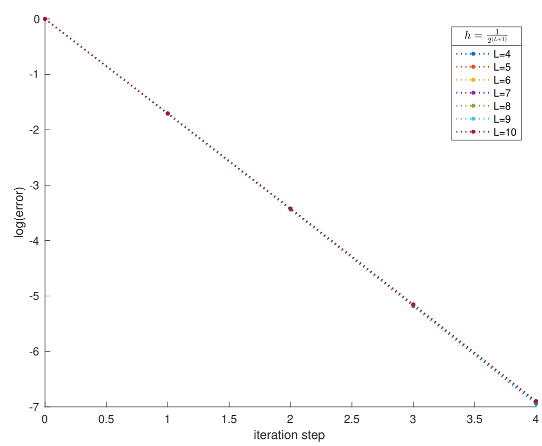

Figure 4: Error Decay $\log(\frac{\|\vec{u}-\vec{u}_{\mathrm{MG}}\|_2}{\|\vec{u}\|_2})$ for the multigrid method for different mesh sizes.

We attach the Matlab code for the main and MG functions in the following listings.

```
%% 1D multigrid method
%%%%%%%%%%%%%%%%%%%%%
%-u''=f  in (0,1);
% u(0)=u(1)=0;
% where f=pi^2*sin(x*pi);
% Suppose a hierarchy of L+1 grids with mesh sizes hl ;
% hl = (1/2)^(l+1), (l=0,1,...,L);
% N =2^(L+1)-1;
```

Table 1: Convergence behavior of 1D multigrid method.

#Levels	#DOF	#Iter	#$\frac{\|u-u_{\mathrm{MG}}\|_2}{\|f\|_2}$	Contract factor
5	31	4	1.15e-7	0.0174
6	63	4	1.24e-7	0.0179
7	127	4	1.26e-7	0.0180
8	255	4	1.26e-7	0.0180
9	511	4	1.27e-7	0.0180
10	1023	4	1.27e-7	0.0180
11	2047	4	1.27e-7	0.0180

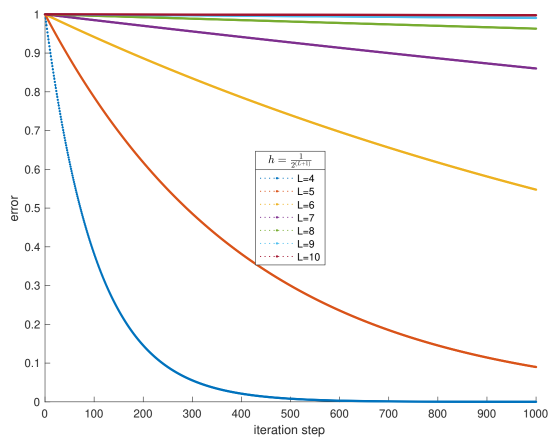

Figure 5: Error Decay $\frac{\|\vec{u}-\vec{u}_{\mathrm{GS}}\|_2}{\|\vec{u}\|_2}$ for the Gauss–Seidel method for different mesh sizes.

```
% level L:finest level , level 0 :the coarsest level;
% smoother: Guass Seidel;
% stop criterion: ||r||_2/||f||_2<=1e-6
%%%%%%%%%%%%%%%%%%%%%%
%% input
clear,clc,close all
f = @(x) pi^2*sin(x*pi);
Ls = [4 5 6 7 8 9 10];
m = size(Ls,2);
mu = 3; % number of smoothing steps
itMax = 1000;
M_Itstep = zeros(m,1);
GS_Itstep = zeros(m,1);
M_uerror = zeros(m,itMax);
GS_uerror = zeros(m,itMax);
fprintf('Convergence behavior of 1D Multigrid method\n');
```

```
for k = 1:m
    L= Ls(k);
    N =2^(L+1)-1; %DOF
    h = (1/2)^(L+1);
    x=h:h:1-h; %coordinates
    b = [0;f(x)';0]; %bi=f(xi)
    u0 = zeros(N+2,1); %initialize u0
    e = ones (N,1);
    A = spdiags ([[-e;0;0],[1;2*e;1],[0;0;-e]],[-1,0,1],N+2,N+2)/h^2;
    u_exact = A\b; %exact solution
    nb = norm(b);
    %%% Multigrid Method %%%
    n = 1;
    u_M = u0;
    r = b(2:N+1)+(-2*u_M(2:N+1)+u_M(1:N)+u_M(3:N+2))/h^2;
    Rerr = norm(r)/nb;
    nu = norm(u_exact);
    uerr = norm(u_M-u_exact)/nu;
    M_uerror(k,1) = uerr;
    while Rerr > 1e-6 && n<=itMax
        u_M= MG(L,b,u_M,mu);
        r = b(2:N+1)+(-2*u_M(2:N+1)+u_M(1:N)+u_M(3:N+2))/h^2;
        Rerr = norm(r)/nb;
        uerr = norm(u_M-u_exact)/nu;
        n = n + 1;
        M_uerror(k,n) = uerr;
    end
    M_Itstep(k) = n-1;
    %% contraction factor
    for j = 1:n-1
        Contract(k,j) = M_uerror(k,j+1)/max(M_uerror(k,j),eps);
    end
    %% output
    fprintf('Level:%5.0u, #DoF:%5.0u, Iter:%2.0u, Error(|u-uh|/|u|)=%8.2e\n',...
        L+1, N, M_Itstep(k),M_uerror(k,n))
    %%% Gauss--Seidel method %%%%
    [~,n_G,uerrG] = GS(b,u0,u_exact,itMax);
    GS_Itstep(k) = n_G;
    GS_uerror(k,1:n_G+1)=uerrG;
end
```

```
function u = MG(l,fl,u,mu)
% % %initial parameter
N = 2^(l+1)-1;
Nc = 2^(l)-1;
h = 1/(N+1);
h2 = h^2;
% directly solve on coarsest level
```

```
if l==0
    e = ones(N,1);
    % suppose FD discretization is used
    A = spdiags([[-e;0;0],[1;2*e;1],[0;0;-e]],[-1,0,1],N+2,N+2)/h2;
    u = A\fl;
else
    %%% Presmoothing
    for n = 1:mu
        %One step of Gauss-Seidel iteration
        for i = 2:N+1
            u(i) = (h2*fl(i)+u(i-1)+u(i+1))/2;
        end
    end
    %%% Restriction
    rf = zeros(N+2,1);
    i = 2:N+1;
    rf(i) = fl(i)+(-2*u(i)+u(i-1)+u(i+1))/h2;
    rc = zeros(Nc+2,1);
    j = 2:Nc+1;
    rc(j) = 0.5*rf(2*j-1)+0.25*rf(2*j-2)+0.25*rf(2*j);
    %%% Coarse grid correction
    ec = MG(l-1,rc,zeros(2^l+1,1),mu);
    %%% Prolongation
    Pro_ec = zeros(N+2,1);
    Pro_ec(1:2:N+2) = ec;
    Pro_ec(2:2:N+1) = 0.5*ec(1:Nc+1)+0.5*ec(2:Nc+2);
    u = u + Pro_ec;
    %%% Postsmoothing
    for n = 1:mu
        % One step of Gauss--Seidel iteration
        for i = 2:N+1
            u(i) = (h2*fl(i)+u(i-1)+u(i+1))/2;
        end
    end
end
end
```

HW 2.1. Show the identity (2.7), i.e., $\mathcal{B}^* = \mathcal{A}^{-1}\mathcal{B}^T\mathcal{A}$.

Proof. By the definition of the adjoint operator, we have $(\mathcal{B}^*u, v)_{\mathcal{A}} = (u, \mathcal{B}v)_{\mathcal{A}}$, for any $u, v \in \mathcal{V}$. This is equivalent to

$$(\mathcal{A}\mathcal{B}^*u, v) = (\mathcal{A}u, \mathcal{B}v) = (\mathcal{B}^T\mathcal{A}u, v),$$

which implies $\mathcal{A}\mathcal{B}^* = \mathcal{B}^T\mathcal{A}$. Hence the result. □

HW 2.5. Prove Lemma 2.3 and 2.4.

Proof of Lemma 2.3. We prove Lemma 2.3 in the following equivalent form:

a. $(r^{(i)}, p^{(i)}) = (r^{(i)}, r^{(i)}), \quad i = 0, 1, 2, ...$

b. $(r^{(k)}, p^{(i)}) = 0, \quad i = 0, 1, 2, ..., k-1$

c. $(p^{(k)}, p^{(i)})_{\mathcal{A}} = 0, \quad i = 0, 1, 2, ..., k-1$

d. $(r^{(k)}, r^{(i)}) = 0, \quad i = 0, 1, 2, ..., k-1$

In order to apply mathematical induction, we first check that, for $k = 1$,

$$\begin{aligned}
&(r^{(0)}, p^{(0)}) = (r^{(0)}, r^{(0)}), \\
&(r^{(1)}, p^{(0)}) = (r^{(1)}, r^{(0)}) = 0, \\
&(r^{(1)}, p^{(1)}) = (r^{(1)}, r^{(1)} + \beta_0 p^{(0)}) = (r^{(1)}, r^{(1)}), \\
&(p^{(1)}, p^{(0)})_{\mathcal{A}} = 0.
\end{aligned}$$

Assume that the theorem is true for any $k = j$. We now consider the case of $k = j+1$:

(1) Since $r^{(j+1)} = r^{(j)} - \alpha_j \mathcal{A} p^{(j)}$, we have $(r^{(j+1)}, p^{(i)}) = (r^{(j)}, p^{(i)}) - \alpha_j (p^{(j)}, p^{(i)})_{\mathcal{A}}$. Thus the induction assumption yields that

$$(r^{(j+1)}, p^{(i)}) = 0, \quad i = 1, 2, ..., j-1.$$

From the definition of α_j, we can obtain that

$$(r^{(j+1)}, p^{(j)}) = (r^{(j)}, p^{(j)}) - \alpha_j (p^{(j)}, p^{(j)})_{\mathcal{A}} = (r^{(j)}, p^{(j)}) - (r^{(j)}, r^{(j)}),$$

which vanishes, thanks to the induction assumption. Therefore, (b) is true for $k = j+1$.

(2) It follows from the definition of $p^{(j+1)}$ and (b) for $k = j+1$, i.e.,

$$(r^{(j+1)}, p^{(j+1)}) = (r^{(j+1)}, r^{(j+1)} + \beta_j p^{(j)}) = (r^{(j+1)}, r^{(j+1)}),$$

and, hence, (a) holds for $k = j+1$.

(3) Using the relation among $p^{(i)}$, $r^{(i)}$ and $p^{(i-1)}$ and (b) for $k = j+1$, we obtain

$$(r^{(j+1)}, r^{(i)}) = (r^{(j+1)}, p^{(i)} - \beta_{i-1} p^{(i-1)}) = 0, \quad i = 0, 1, ..., j.$$

Hence (d) holds for $k = j+1$.

(4) From $\mathcal{A} p^{(i)} = \frac{1}{\alpha_{i+1}} (r^{(i)} - r^{(i+1)})$ and (d) in the case of $k = j+1$, we obtain

$$\begin{aligned}
(p^{(j+1)}, p^{(i)})_{\mathcal{A}} &= (r^{(j+1)}, p^{(i)})_{\mathcal{A}} + \beta_j (p^{(j)}, p^{(i)})_{\mathcal{A}} \\
&= (r^{(j+1)}, p^{(i)})_{\mathcal{A}} = (r^{(j+1)}, \mathcal{A} p^{(i)}) \\
&= \frac{1}{\alpha_{i+1}} (r^{(j+1)}, r^{(i)} - r^{(i+1)}) = 0, \quad i = 1, 2, .., j-1.
\end{aligned}$$

By the definition of $p^{(j+1)}$, we have $(p^{(j+1)}, p^{(j)})_{\mathcal{A}} = 0$ as well. Hence (c) holds for $k = j + 1$. $\square$

Proof of Lemma 2.4. We will prove the following expressions:

a. $\alpha_m = \frac{(r^{(m)}, r^{(m)})}{(\mathcal{A}p^{(m)}, p^{(m)})}$,

b. $\beta_m = \frac{(r^{(m+1)}, r^{(m+1)})}{(r^{(m)}, r^{(m)})}$.

Recall that $\alpha_m = \frac{(r^{(m)}, p^{(m)})}{(\mathcal{A}p^{(m)}, p^{(m)})}$. We immediately obtain (a) for $m = 0$. From $p^{(m)} = r^{(m)} + \beta_{m-1}p^{(m-1)}$ and $(r^{(m)}, p^{(m-1)}) = 0$, it is clear that (a) holds for any m. From the definition of β_m and Lemma 2.3, we immediately obtain

$$\begin{aligned}\beta_m &= \frac{-(\mathcal{A}r^{(m+1)}, p^{(m)})}{(\mathcal{A}p^{(m)}, p^{(m)})} = \frac{-(r^{(m+1)}, \mathcal{A}p^{(m)})}{(\mathcal{A}p^{(m)}, p^{(m)})} = \frac{(r^{(m+1)}, r^{(m+1)} - r^{(m)})}{(r^{(m)} - r^{(m+1)}, p^{(m)})} \\ &= \frac{(r^{(m+1)}, r^{(m+1)})}{(r^{(m)}, p^{(m)})} = \frac{(r^{(m+1)}, r^{(m+1)})}{(r^{(m)}, r^{(m)})}.\end{aligned}$$

Hence the results. $\square$

HW 2.8. Show that (2.30) and (2.31) are equivalent to each other, i.e., if $\mathcal{A}$ and $\mathcal{B}$ are symmetric positive definite operators on a finite-dimensional space V, $\alpha > 0$ and $0 < \delta < 1$, then it is easy to verify the following two conditions are equivalent:

a. $-\alpha(\mathcal{A}u, u) \leqslant (\mathcal{A}(\mathcal{I} - \mathcal{B}\mathcal{A})u, u) \leqslant \delta(\mathcal{A}u, u)$, $\forall u \in V$,

b. $(1 + \alpha)^{-1}(\mathcal{A}u, u) \leqslant (\mathcal{B}^{-1}u, u) \leqslant (1 - \delta)^{-1}(\mathcal{A}u, u)$, $\forall u \in V$.

Proof. Rearrange the inequalities (a), we obtain $(1 - \delta)(\mathcal{A}u, u) \leqslant (\mathcal{A}\mathcal{B}\mathcal{A}u, u) \leqslant (1 + \alpha)(\mathcal{A}u, u)$. From Lemma 2.6, it is equivalent to show $(1 + \alpha)^{-1}(\mathcal{A}u, u) \leqslant (\mathcal{B}^{-1}u, u) \leqslant (1 - \delta)^{-1}(\mathcal{A}u, u)$, $\forall u \in V$, which is exactly (b). $\square$

HW 2.9. Let $\mathcal{A}$ be SPD and $\mathcal{B}$ be a symmetric iterator. If $\rho = \|\mathcal{I} - \mathcal{B}\mathcal{A}\|_{\mathcal{A}} < 1$, then $\mathcal{B}$ is SPD, and $\kappa(\mathcal{B}\mathcal{A}) \leqslant \frac{1+\rho}{1-\rho}$.

Proof. Since $\mathcal{A}$ is SPD and $\mathcal{B}^T = \mathcal{B}$, we have $(\mathcal{B}\mathcal{A})^* = \mathcal{B}\mathcal{A}$ and

$$\rho(\mathcal{I} - \mathcal{B}\mathcal{A}) = \|\mathcal{I} - \mathcal{B}\mathcal{A}\|_{\mathcal{A}} = \rho < 1.$$

It indicates that $\mathcal{B}\mathcal{A}$ is SPD w.r.t. $(\cdot, \cdot)_{\mathcal{A}}$. It is equivalent to $\mathcal{B}$ is SPD w.r.t. $(\cdot, \cdot)$. Furthermore, by definition, we have

$$((\mathcal{I} - \mathcal{B}\mathcal{A})^2 u, u)_{\mathcal{A}} \leqslant \rho^2 (u, u)_{\mathcal{A}}.$$

Expanding it, we obtain

$$((\mathcal{A} - 2\mathcal{A}\mathcal{B}\mathcal{A} + \mathcal{A}\mathcal{B}\mathcal{A}\mathcal{B}\mathcal{A})u, u) \leqslant \rho^2 (u, u)_{\mathcal{A}}.$$

Changing variable $v = \mathcal{A}^{1/2}u$, we obtain

$$\Big((\mathcal{I} - \mathcal{A}^{1/2}\mathcal{B}\mathcal{A}^{1/2})^2 v, v\Big) \leqslant \rho^2 (v, v) \quad \Longrightarrow \quad \Big|\Big((\mathcal{I} - \mathcal{A}^{1/2}\mathcal{B}\mathcal{A}^{1/2}) v, v\Big)\Big| \leqslant \rho(v, v)$$
$$\Longrightarrow \quad \Big|((\mathcal{A} - \mathcal{A}\mathcal{B}\mathcal{A})u, u)\Big| \leqslant \rho(\mathcal{A}u, u), \quad \forall u \in V.$$

From HW 2.8, we obtain

$$(1+\rho)^{-1}(\mathcal{A}u, u) \leqslant (\mathcal{B}^{-1}u, u) \leqslant (1-\rho)^{-1}(\mathcal{A}u, u), \quad \forall u \in V.$$

Hence Lemma 2.5 yields $\kappa(\mathcal{B}\mathcal{A}) \leqslant \frac{1+\rho}{1-\rho}$.

A simple proof. Since $\rho = ||\mathcal{I} - \mathcal{B}\mathcal{A}||_{\mathcal{A}} < 1$, with the assumptions, we have $\rho = \rho(\mathcal{I} - \mathcal{A}^{1/2}\mathcal{B}\mathcal{A}^{1/2}) < 1$. Hence, $\lambda(\mathcal{B}\mathcal{A}) \subseteq [1-\rho, 1+\rho]$, which yields the desired result. □

HW 3.5. Give a complete proof of Remark 3.15, i.e., let u_h and u_H be the finite element solutions on V_h and $V_H \subset V_h$, respectively, and then we have $\|u_h - u_H\|_0 \lesssim H|||u_h|||$.

Proof. From the definitions of u_h and u_H, we have

$$a[u_h - u_H, v_H] = 0, \quad \forall v_H \in V_H. \tag{1}$$

Using the Aubin-Nitshe's argument, we consider a boundary value problem

$$\begin{cases} -\Delta w = u_h - u_H & \text{in } \Omega, \\ \quad w = 0 & \text{on } \partial\Omega. \end{cases}$$

Its variational form in V and discrete approximation in V_H are given below, i.e.,

$$a[w, v] = (u_h - u_H, v), \quad \forall v \in V, \tag{2}$$
$$a[w_H, v_H] = (u_h - u_H, v_H), \quad \forall v_H \in V_H. \tag{3}$$

Assume that we have full elliptic regularity, namely, $\|w\|_2 \leqslant C\|u_h - u_H\|_0$. Submit $v = u_h - u_H$ in (2). Then we obtain

$$\|u_h - u_H\|_0^2 = a[w, u_h - u_H] = a[w - w_H, u_h - u_H] \leqslant |||w - w_H||| \, |||u_h - u_H|||$$
$$\lesssim H|w|_2 |||u_h - u_H||| \lesssim H\|u_h - u_H\|_0 |||u_h - u_H|||.$$

By eliminating $\|u_h - u_H\|_0$ on both sides of the last inequality, we obtain the following inequality

$$\|u_h - u_H\|_0 \lesssim H|||u_h - u_H||| \lesssim H|||u_h|||,$$

where we have used the inequality $|||u_h - u_H||| \lesssim |||u_h|||$ due to the fact that u_H is the $\mathcal{A}$-projection of u_h into V_H. □

HW 3.6. Write the 1D multigrid method in §1.4 as a two-grid method (Algorithm 3.2) called recursively and modify your implementation in this way.

Algorithm (One iteration of multigrid method). $\vec{u}_l = MG(l, \vec{f}_l, \vec{u}_l)$

(i) **Judge if it reaches the finest grid**: If l is equal to 1, return $\vec{u}_l = A_l^{-1}\vec{f}_l$, else, go to (ii)

(ii) **Pre-smoothing**: $\vec{u}_l \leftarrow \vec{u}_l + S(\vec{f}_l - A_l\vec{u}_l)$

(iii) **Restriction**: $\vec{r}_{l-1} \leftarrow \vec{R}_{l,l-1}(\vec{f}_l - A_l\vec{u}_l)$

(iv) **Coarse-grid correction**: $\vec{e}_{l-1} \leftarrow MG(l-1, \vec{r}_{l-1}, \vec{0}_{l-1})$

(v) **Prolongation**: $\vec{u}_l \leftarrow \vec{u}_l + \vec{P}_{l-1,l}\vec{e}_{l-1}$

(vi) **Post-smoothing**: $\vec{u}_l \leftarrow \vec{u}_l + S^T(\vec{f}_l - A_l\vec{u}_l)$, return $\vec{u}_l$

The code is easier to implement this way. The other way is to use for loops instead of recursive calls, which will usually yield better performance. □

HW 4.3. Show that the block G-S method for the expanded system is just the SSC method for the original problem.

Proof. Consider a linear equation $\mathcal{A}u = f$ and it's expanded equation $\mathbf{Au} = \mathbf{f}$. We now apply the block G-S method for the expanded system, i.e.,

$$\mathbf{u}^{\text{new}} = \mathbf{u}^{\text{old}} + (\mathbf{D} + \mathbf{L})^{-\mathbf{1}}(\mathbf{f} - \mathbf{A}\mathbf{u}^{\text{old}}).$$

We can rewrite this method as

$$(\mathbf{D} + \mathbf{L})\mathbf{u}^{\text{new}} = (\mathbf{D} + \mathbf{L})\mathbf{u}^{\text{old}} + (\mathbf{f} - \mathbf{A}\mathbf{u}^{\text{old}}).$$

Hence we have

$$\mathbf{D}\mathbf{u}^{\text{new}} = \mathbf{D}\mathbf{u}^{\text{old}} + \mathbf{f} - \mathbf{L}\mathbf{u}^{\text{new}} - (\mathbf{D} + \mathbf{U})\mathbf{u}^{\text{old}};$$

in turn, we get

$$\mathbf{u}^{\text{new}} = \mathbf{u}^{\text{old}} + \mathbf{D}^{-\mathbf{1}}\left(\mathbf{f} - \mathbf{L}\mathbf{u}^{\text{new}} - (\mathbf{D} + \mathbf{U})\mathbf{u}^{\text{old}}\right).$$

For $j = 1, ..., J$, the block G-S method can be written as

$$u_j^{\text{new}} = u_j^{\text{old}} + \mathcal{A}_j^{-1}(\mathcal{I}_j^T f - \sum_{i<j} \mathcal{I}_j^T \mathcal{A} \mathcal{I}_i u_i^{\text{new}} - \sum_{i \geqslant j} \mathcal{I}_j^T \mathcal{A} \mathcal{I}_i u_i^{\text{old}}).$$

We define iteration

$$u^{\frac{j}{J}} := \sum_{i<j} u_i^{\text{new}} + \sum_{i\geqslant j} u_i^{\text{old}} = \sum_{i<j} \mathcal{I}_i u_i^{\text{new}} + \sum_{i\geqslant j} \mathcal{I}_i u_i^{\text{old}}, \quad j = 1, ..., J.$$

By this definition, we can see that

$$u^{\frac{j+1}{J}} = u^{\frac{j}{J}} + \mathcal{I}_j u_j^{\text{new}} - \mathcal{I}_j u_j^{\text{old}} = u^{\frac{j}{J}} + \mathcal{I}_j \mathcal{A}_j^{-1} \mathcal{I}_j^T (f - \mathcal{A} u^{\frac{j}{J}}).$$

Here the term $f - \mathcal{A}u^{\frac{j}{J}}$ is sometimes called the *dynamic residual*, which is the residual at an inner iteration of the G-S method. From the above equation, we notice that the block G-S method is just the SSC method with exact subspace solvers $S_j = \mathcal{A}_j^{-1}$ for the original linear equation. □

HW 4.4. Prove Theorem 4.2, that is, if $\overline{\mathbf{S}} := \mathbf{S}^{\mathrm{T}} + \mathbf{S} - \mathbf{S}^{\mathrm{T}}\mathbf{D}\mathbf{S}$ (or $\mathbf{K} := \mathbf{S}^{-\mathrm{T}} + \mathbf{S}^{-1} - \mathbf{D}$) is SPD, then the modified block G-S method converges and

$$\|\mathbf{I} - \mathbf{B}\mathbf{A}\|_{\mathbf{A}}^2 = 1 - \frac{1}{1 + c_0}, \quad \text{with} \quad c_0 := \sup_{\|\mathbf{v}\|_{\mathbf{A}}=1} \|\mathbf{K}^{-\frac{1}{2}}(\mathbf{D} + \mathbf{U} - \mathbf{S}^{-1})\mathbf{v}\|^2.$$

Proof. Consider the expanded system $\mathbf{A}\mathbf{u} = \mathbf{f}$. The modified block G-S method is defined as $\mathbf{B} = (\mathbf{S}^{-1} + \mathbf{L})^{-1}$, where $\mathbf{S}$ is an invertible smoother. From the definition of $\mathbf{K}$, we also have $\mathbf{K} = \mathbf{B}^{-\mathrm{T}} + \mathbf{B}^{-1} - \mathbf{A}$, and the symmetrization operator is denoted as $\overline{\mathbf{B}} = \mathbf{B}^{\mathrm{T}}\mathbf{K}\mathbf{B}$.

From the definition of $\overline{\mathbf{S}}$, we have $\mathbf{K} = \mathbf{S}^{-\mathrm{T}} + \mathbf{S}^{-1} - \mathbf{D} = \mathbf{S}^{-\mathrm{T}}\overline{\mathbf{S}}\mathbf{S}^{-1}$. Therefore, if $\overline{\mathbf{S}}$ is SPD (or $\mathbf{K}$ is SPD), then we can obtain that $\overline{\mathbf{B}}$ is SPD, which implies the convergence of the modified block G-S method.

From $\mathbf{K} = \mathbf{B}^{-\mathrm{T}} + \mathbf{B}^{-1} - \mathbf{A}$, we have $\mathbf{B}^{-1} = \mathbf{K} + \mathbf{A} - \mathbf{B}^{-\mathrm{T}}$. Hence we have a representation of $\overline{\mathbf{B}}^{-1}$ by simple manipulations:

$$\overline{\mathbf{B}}^{-1} = (\mathbf{K} + \mathbf{A} - \mathbf{B}^{-\mathrm{T}})\mathbf{K}^{-1}(\mathbf{K} + \mathbf{A} - \mathbf{B}^{-1}) = \mathbf{A} + (\mathbf{A} - \mathbf{B}^{-\mathrm{T}})\mathbf{K}^{-1}(\mathbf{A} - \mathbf{B}^{-1}).$$

The last equality and $\mathbf{B} = (\mathbf{S}^{-1} + \mathbf{L})^{-1}$ yields that

$$(\overline{\mathbf{B}}^{-1}\mathbf{v}, \mathbf{v}) = (\mathbf{A}\mathbf{v}, \mathbf{v}) + \big(\mathbf{K}^{-1}(\mathbf{D} + \mathbf{U} - \mathbf{S}^{-1})\mathbf{v}, (\mathbf{D} + \mathbf{U} - \mathbf{S}^{-1})\mathbf{v}\big), \quad \forall \mathbf{v} \in \mathbf{V}.$$

The SSPD operator $\mathbf{A}$ is SPD if we restrict its domain inside range($\mathbf{A}$). Therefore, we can apply the convergence rate estimate in Theorem 2.3, i.e., $\|\mathbf{I} - \mathbf{B}\mathbf{A}\|_{\mathbf{A}}^2 = 1 - \frac{1}{c_1}$, with

$$\begin{aligned} c_1 := \sup_{\|\mathbf{v}\|_{\mathbf{A}}=1} (\overline{\mathbf{B}}^{-1}\mathbf{v}, \mathbf{v}) &= \sup_{\|\mathbf{v}\|_{\mathbf{A}}=1} \Big((\mathbf{A}\mathbf{v}, \mathbf{v}) + \Big(\mathbf{K}^{-1}(\mathbf{D} + \mathbf{U} - \mathbf{S}^{-1})\mathbf{v}, (\mathbf{D} + \mathbf{U} - \mathbf{S}^{-1})\mathbf{v}\Big)\Big) \\ &= 1 + \sup_{\|\mathbf{v}\|_{\mathbf{A}}=1} \|\mathbf{K}^{-\frac{1}{2}}(\mathbf{D} + \mathbf{U} - \mathbf{S}^{-1})\mathbf{v}\|^2. \end{aligned}$$

Hence the result. □