Lab 1 实验报告

PB20000296 郑滕飞

1、功能实现

实验要求：实现模型问题三种迭代方法求解，并对比不同初始估计的影响。
代码结构：
main1.m 主程序脚本-复现书上配图[调用 iteration]
main2.m 主程序脚本-初始估计研究[调用 iteration]
iteration.m 迭代方法实现

2、算法简述

对模型问题

$$-v'' + \sigma v = f$$
$$v(0) = v(1) = 0$$

离散后，假设步长 h，格点处 v 与 f 为v_i, f_i，且$v_0 = v_n = 0$。

权重ω的 Jacobi 迭代算法为（v 上标 k 表示第 k 次迭代后）

$$v_i^{j+1} = (1-\omega)v_i^j + \omega\frac{v_{i-1}^j + v_{i+1}^j - h^2 f_i}{2+\sigma h^2}$$

取定权重为 2/3。

Gauss-Seidel 迭代为

$$v_i^{j+1} = \frac{v_{i-1}^{j+1} + v_{i+1}^j - h^2 f_i}{2+\sigma h^2}$$

实际实现通过按下标从小到大进行更新即可。

红黑 Gauss-Seidel 迭代分为

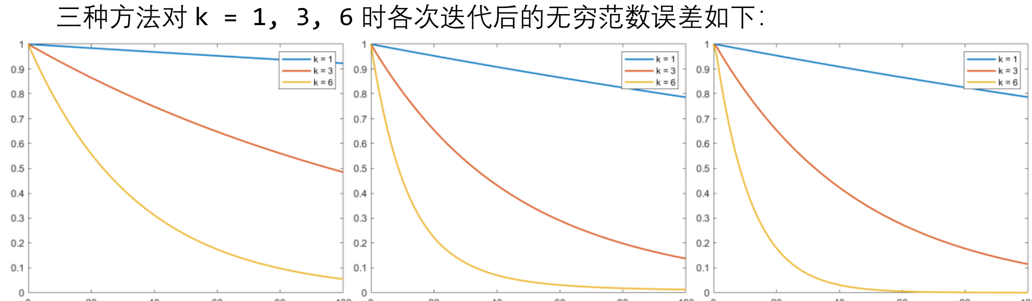
$$v_{2i}^{j+1} = \frac{v_{2i-1}^j + v_{2i+1}^j - h^2 f_{2i}}{2+\sigma h^2}$$
$$v_{2i+1}^{j+1} = \frac{v_{2i}^{j+1} + v_{2i+2}^{j+1} - h^2 f_{2i+1}}{2+\sigma h^2}$$

两步。

考虑$f = \sigma = 0$的问题，精确解为$v = 0$，波数为 k 的初始估计是指$v_0(x) = \sin k\pi x$。取网格步长 h = 0.015625。

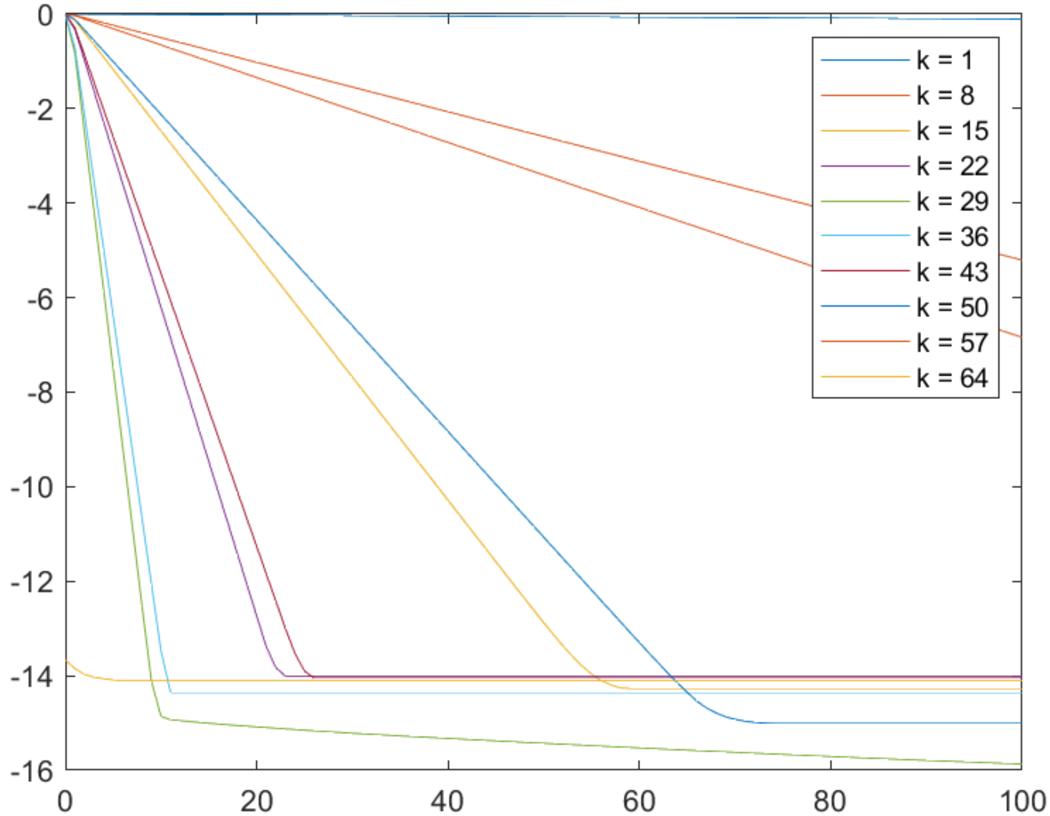
3、结果展示

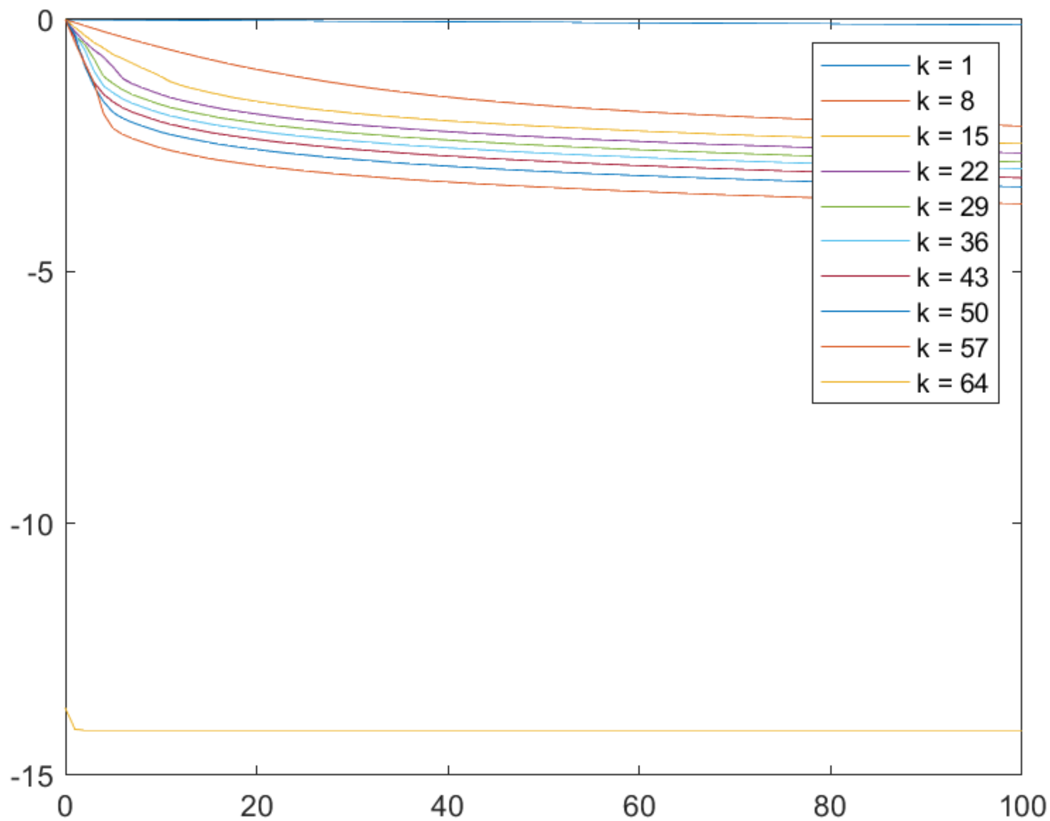
三种方法对 k = 1, 3, 6 时各次迭代后的无穷范数误差如下：

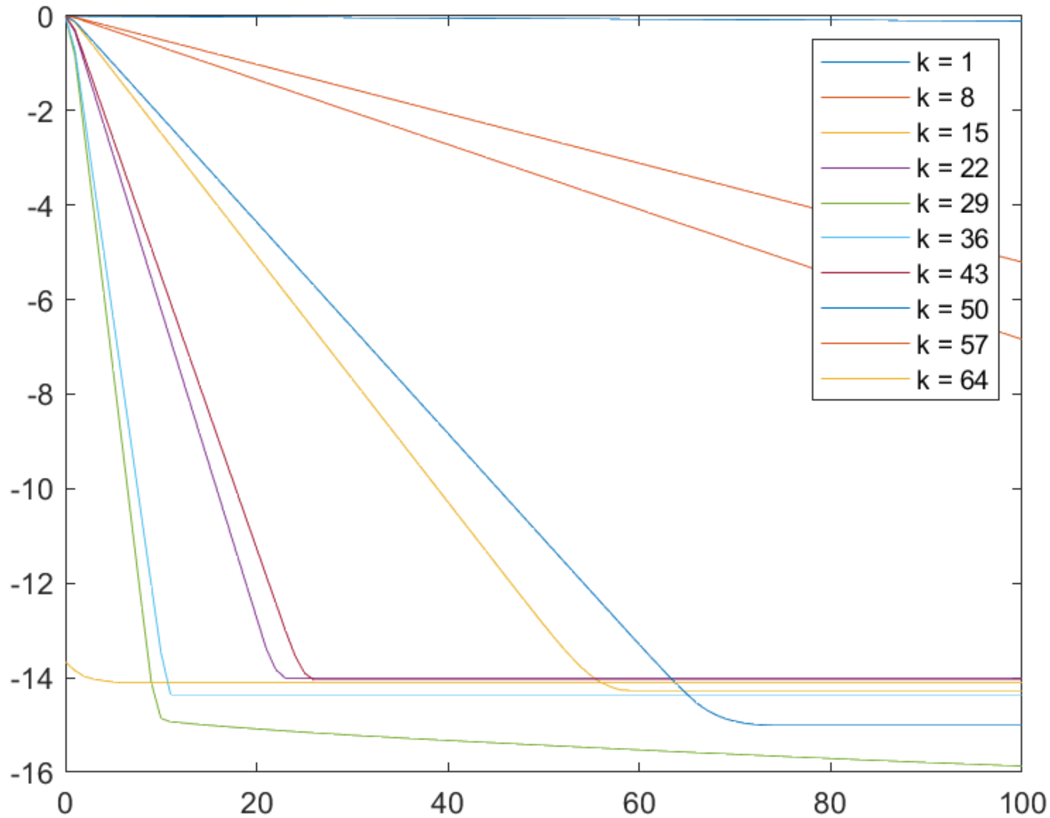


此结果与书中图 2.3 一致。

令 k 在 1 到 64 中等间隔变化，对无穷范数误差以 10 为底对数作图：







4、讨论

从复现图 2.3 的结果可以直观看出，三种方法的收敛速率逐渐加快。但在 k 增大时可以看出，GS 迭代由于每次更新都会累计 N 次误差，很快就达到了精度上限 1e-4，而加权 Jacobi 迭代和红黑 GS 迭代则能接近机器精度。由于红黑 GS 迭代收敛更快，且原地迭代不需要额外的空间，一般来说是更好的选择。

附录-代码

```
x = 0:1/64:1;
f = 0 * x;
res = 0 * x;
sigma = 0;
steps = 100;

method = "weighted-Jacobi";
omega = 2/3;
k = 1;
v0 = sin(k * x * pi);
loss1 = iteration(v0, f, sigma, steps, method, res, omega);
k = 3;
v0 = sin(k * x * pi);
```

```
loss2 = iteration(v0, f, sigma, steps, method, res, omega);
k = 6;
v0 = sin(k * x * pi);
loss3 = iteration(v0, f, sigma, steps, method, res, omega);
figure;
plot(0:steps, loss1, 0:steps, loss2, 0:steps, loss3, "linewidth", 1.5);
legend("k = 1", "k = 3", "k = 6");

method = "Gauss-Seidel";
omega = 2/3;
k = 1;
v0 = sin(k * x * pi);
loss1 = iteration(v0, f, sigma, steps, method, res, omega);
k = 3;
v0 = sin(k * x * pi);
loss2 = iteration(v0, f, sigma, steps, method, res, omega);
k = 6;
v0 = sin(k * x * pi);
loss3 = iteration(v0, f, sigma, steps, method, res, omega);
figure;
plot(0:steps, loss1, 0:steps, loss2, 0:steps, loss3, "linewidth", 1.5);
legend("k = 1", "k = 3", "k = 6");

method = "red-black";
omega = 2/3;
k = 1;
v0 = sin(k * x * pi);
loss1 = iteration(v0, f, sigma, steps, method, res, omega);
k = 3;
v0 = sin(k * x * pi);
loss2 = iteration(v0, f, sigma, steps, method, res, omega);
k = 6;
v0 = sin(k * x * pi);
loss3 = iteration(v0, f, sigma, steps, method, res, omega);
figure;
plot(0:steps, loss1, 0:steps, loss2, 0:steps, loss3, "linewidth", 1.5);
legend("k = 1", "k = 3", "k = 6");
```

```
x = 0:1/64:1;
f = 0 * x;
res = 0 * x;
sigma = 0;
steps = 100;

method = "weighted-Jacobi";
```

```
omega = 2/3;
figure;
for k = 1:7:64
    v0 = sin(k * x * pi);
    loss = iteration(v0, f, sigma, steps, method, res, omega);
    plot(0:steps, log(loss) / log(10), 'DisplayName', "k = " + num2str(k));
    hold on;
end
legend;
hold off;

method = "Gauss-Seidel";
figure;
for k = 1:7:64
    v0 = sin(k * x * pi);
    loss = iteration(v0, f, sigma, steps, method, res, omega);
    plot(0:steps, log(loss) / log(10), 'DisplayName', "k = " + num2str(k));
    hold on;
end
legend;
hold off;

method = "red-black";
figure;
for k = 1:7:64
    v0 = sin(k * x * pi);
    loss = iteration(v0, f, sigma, steps, method, res, omega);
    plot(0:steps, log(loss) / log(10), 'DisplayName', "k = " + num2str(k));
    hold on;
end
legend;
hold off;
```

```
function loss = iteration(v0, f, sigma, steps, method, res, omega)
    loss = zeros(1, steps + 1);
    v = v0;
    loss(1) = max(abs(v - res));
    for i = 1:steps
        if method == "weighted-Jacobi"
           v = wJacobiIter(v, f, sigma, omega);
        end
        if method == "Gauss-Seidel"
            v = GaussSeidel(v, f, sigma);
        end
        if method == "red-black"
```

```
            v = RBGaussSeidel(v, f, sigma);
        end
        loss(i + 1) = max(abs(v - res));
    end
end

function v = wJacobiIter(v, f, sigma, omega)
    h = 1 / (length(v) - 1);
    v(2:end-1) = (1 - omega) * v(2:end-1) + omega * ...
        (v(1:end-2) + v(3:end) + h^2 * f(2:end-1)) / (2 + sigma * h^2);
end

function v = GaussSeidel(v, f, sigma)
    h = 1 / (length(v) - 1);
    for i = 2:length(v)-1
        v(i) = (v(i-1) + v(i+1) + h^2 * f(i)) / (2 + sigma * h^2);
    end
end

function v = RBGaussSeidel(v, f, sigma)
    h = 1 / (length(v) - 1);
    v(3:2:end-1) = (v(2:2:end-2) + v(4:2:end) + h^2 * f(3:2:end-1)) / ...
        (2 + sigma * h^2);
    v(2:2:end-1) = (v(1:2:end-2) + v(3:2:end) + h^2 * f(2:2:end-1)) / ...
        (2 + sigma * h^2);
end
```

Lab 2 实验报告

PB20000296 郑滕飞

1、功能实现

实验要求：实现 V-cycle 迭代方法。
代码结构：
main.m 主程序脚本[调用 V_cycle.m]
V_cycle.m 递归 V cycle 算法实现

2、算法简述

对模型问题

$$-v'' + \sigma v = f$$
$$v(0) = v(1) = 0$$

离散后，假设步长 h，格点处 v 与 f 为v_i, f_i，且$v_0 = v_n = 0$。

递归的 V-cycle 迭代方法伪代码如下：

1、 初始化估计 v = 0
2、 以初始 v 根据 f, sigma 迭代 nu1 次
3、 若层数为 0，跳至步骤 9，否则：
   a) 计算余项 r = f – Av
   b) 将余项 r 合并为间距两倍的情况
   c) 以 r, sigma 与减 1 的层数，在间距 2h 下递归
   d) 将返回值 r 插值到当前间距的情况
   e) v = v + r
4、 以 v 根据 f, sigma 迭代 nu2 次并返回

这里的迭代过程采用加权 Jacobi 迭代法，权重为 2/3。

另一个值得注意处为 Av 同样不需要显式构造矩阵，这是由于 A 的三对角性，有

$$r_i = f_i - \frac{(2 + \sigma h^2)v_i - v_{i-1} - v_{i+1}}{h^2}, i = 1, \dots, n-1$$

此外，在 n 为偶数的前提下，合并的方法为

$$r_{2i} = \frac{1}{2} r_{2i} + \frac{1}{4}(r_{2i-1} + r_{2i+1}), i = 1, \dots, \frac{n}{2} - 1$$

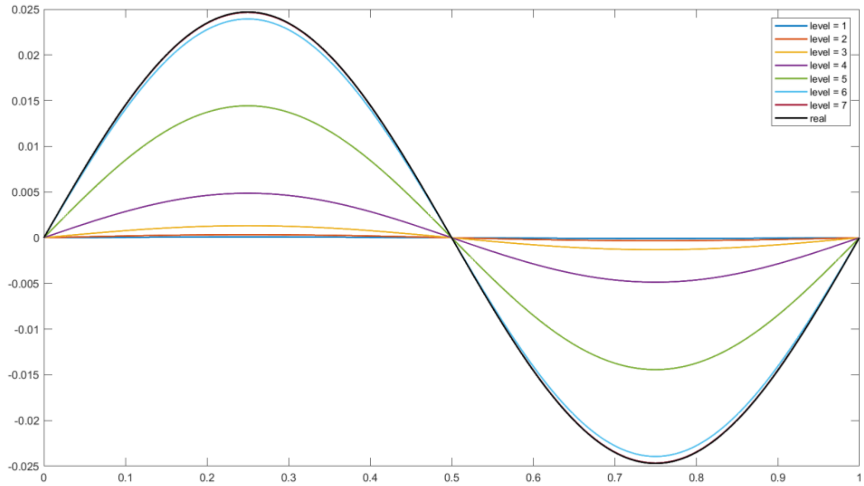
并取所有r_{2i}；将迭代后的结果重新写回r_{2i}后，插值的方法为（注意$r_0 = r_n = 0$）

$$r_{2i-1} = \frac{r_{2i-2} + r_{2i}}{2}, i = 1, \dots, \frac{n}{2}$$

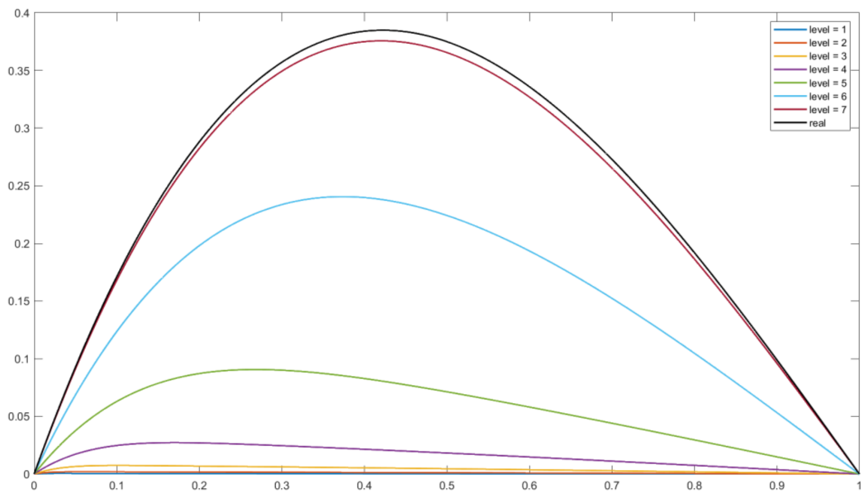
这样就得到了完整的数组r_i。

3、结果展示

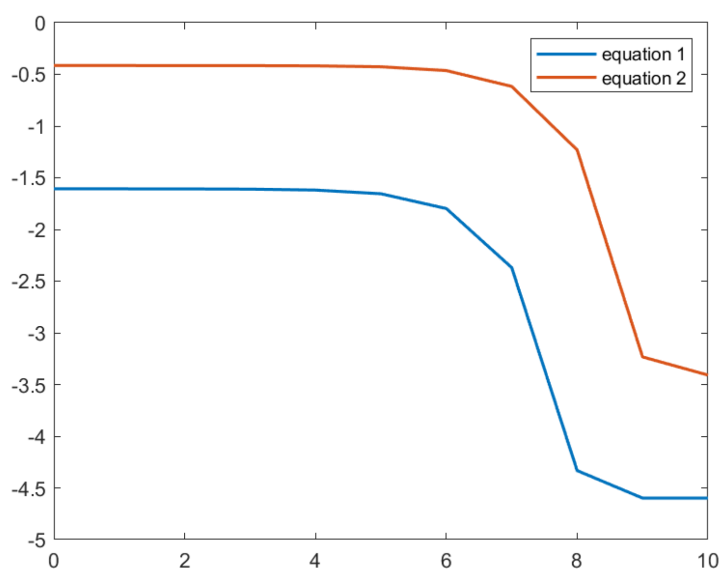
取定网格大小2^{-11}, $\nu_1 = \nu_2 = 100$, 对$f(x) = \sin 2\pi x$, $\sigma = 1$在不同层次迭代效果如下，其中黑线为精确解$\frac{\sin 2\pi x}{4\pi^2+1}$:



而对$f(x) = 6 - 4x - 3x^2 + x^3, \sigma = 1$在不同层次迭代效果如下，其中黑线为精确解$x(x-1)(x-2)$：



取网格大小2^{-13}，$\nu_1 = \nu_2 = 200$，重新对上方两问题在不同层次迭代，并以层数为横坐标无穷范数误差的 10 为底对数为纵坐标绘图：



4、讨论

从前两张图中，可以直观看出结果的正确性，也表明了层数增大时逼近效果的良好。注意到，由于每层网格数除以 2，实际的时间复杂度不会超过$2h^{-1}(\nu_1+\nu_2)T$，其中 T 为某常数，而若直接在原网格迭代，实验可发现至少需要$1000h^{-1}(\nu_1+\nu_2)$才能达到按层分解的效果，因此 V-cycle 大量节省了时间。

由于两结果函数自身的量级差异，第二张图中的形状一致已经表明了 V-cycle 方法的误差特点：在层数增加时，误差会先相对平稳地减少，而在某个特定层数开始大幅下降，直到收敛。值得注意的是，误差收敛的量级远没有达到机器精度，这意味着 V-cycle 方法节省时间的代价除了空间复杂度外还有精度的上限。

附录-代码

```
function v = V_cycle(f, sigma, omega, N, nu1, nu2, level)
    L = 2^N;
    h = 1 / L;
    v = zeros(1, L + 1);
    for i = 1:nu1
        v = wJacobiIter(v, f, sigma, omega);
    end
    if level > 0
        r = v;
        r(2:L) = f(2:L) - ((2 + sigma * h^2) * r(2:L) ...
            - r(1:L-1) - r(3:L+1)) / h^2;
        r(3:2:L-1) = r(3:2:L-1) / 2 + (r(2:2:L-2) + r(4:2:L)) / 4;
        r(1:2:L+1) = ...
            V_cycle(r(1:2:L+1), sigma, omega, N-1, nu1, nu2, level-1);
        r(2:2:L) = (r(1:2:L-1) + r(3:2:L+1)) / 2;
        v = v + r;
    end
    for i = 1:nu2
        v = wJacobiIter(v, f, sigma, omega);
    end
end

function v = wJacobiIter(v, f, sigma, omega)
    h = 1 / (length(v) - 1);
    v(2:end-1) = (1 - omega) * v(2:end-1) + omega * ...
        (v(1:end-2) + v(3:end) + h^2 * f(2:end-1)) / (2 + sigma * h^2);
end
```

```
omega = 2/3;
sigma = 1;
```

```
N = 11;
x = 0:(1/2^N):1;
f = sin(2 * pi * x);
y = sin(2 * pi * x) / (pi^2 * 4 + sigma);
figure;
for l = 1:7
    res = V_cycle(f, sigma, omega, N, 100, 100, l);
    plot(x, res, "linewidth", 1.5, "DisplayName", "level = " + num2str(l));
    hold on;
end
plot(x, y, "black", "linewidth", 1.5, "DisplayName", "real");
legend;

N = 13;
x = 0:(1/2^N):1;
f = sin(2 * pi * x);
y = sin(2 * pi * x) / (pi^2 * 4 + sigma);
err1 = zeros(1, 11);
for l = 0:10
    res = V_cycle(f, sigma, omega, N, 200, 200, l);
    err1(l+1) = log(max(abs(y - res))) / log(10);
end

N = 11;
x = 0:(1/2^N):1;
f = 6 - 4 * x - 3 * x.^2 + x.^3;
y = x .* (1 - x) .* (2 - x);
figure;
for l = 1:7
    res = V_cycle(f, sigma, omega, N, 100, 100, l);
    plot(x, res, "linewidth", 1.5, "DisplayName", "level = " + num2str(l));
    hold on;
end
plot(x, y, "black", "linewidth", 1.5, "DisplayName", "real");
legend;

N = 13;
x = 0:(1/2^N):1;
f = 6 - 4 * x - 3 * x.^2 + x.^3;
y = x .* (1 - x) .* (2 - x);
err2 = zeros(1, 11);
for l = 0:10
    res = V_cycle(f, sigma, omega, N, 200, 200, l);
    err2(l+1) = log(max(abs(y - res))) / log(10);
```

```
end

figure;
plot(0:10, err1, 0:10, err2, "linewidth", 1.5);
legend("equation 1", "equation 2");
```