Dvoretzky应用于分析期望的估计（随机梯度下降的另一种理解方式：低通滤波）/联系现代控制理论和动态系统辨识进行分析

1. 误差动态递推方程

设随机过程 $\{\omega_k\}_{k\geq 0}$ 定义如下：

$\omega_{k+1} = \omega_k + \alpha_k(x_k - \omega_k)$

这就是随机梯度下降的公式（书上P113最后一行）

其中：

- $\omega_k \in \mathbb{R}^n$：第k步的参数估计
- $\omega^* \in \mathbb{R}^n$：目标参数（通常是优化问题的最优解）
- $\alpha_k > 0$：步长序列（可取 $\alpha_k = 1/k$）
- x_k：带有噪声的观测

定义估计误差为：

$\Delta_k = \omega_k - \omega^*$

误差动态递推方程：

通过代数变换，我们得到误差的递推关系：

$$\Delta_{k+1} = (1-\alpha_k)\Delta_k + \alpha_k \cdot \underbrace{(x_k - \omega^*)}_{\eta_k}$$

其中 $\eta_k = x_k - \omega^*$ 是观测偏差，包含了算法的随机性。

或写成

$$\omega_{k+1} - \omega^* = (1-\alpha_k)(\omega_k - \omega^*) + \alpha_k \cdot \underbrace{(x_k - \omega^*)}_{\eta_k}$$

推导过程：

$\omega_{k+1} = \omega_k + \alpha_k(x_k - \omega_k)$

$\omega_{k+1} - \omega^* = (\omega_k - \omega^*) + \alpha_k(x_k - \omega_k)$

$$\Delta_{k+1} = \Delta_k + \alpha_k \underbrace{(x_k - \omega^*) - (\omega_k - \omega^*)}_{\eta_k - \Delta_k}$$

代入得：$\Delta_{k+1} = \Delta_k + \alpha_k(\eta_k - \Delta_k) = (1-\alpha_k)\Delta_k + \alpha_k\eta_k$

2. 按照控制系统的理论进行分析

$\omega_{k+1} - \omega^* = (1-\alpha_k)(\omega_k - \omega^*) + \alpha_k \cdot (x_k - \omega^*)$

将 x_k 视为系统的输入信号（可以是带噪声的阶跃）

ω_k 视为系统的输出

视 α_k 为时不变常数

分析系统的传递函数如下：

- 输出偏差：$Y_k = \omega_k - \omega^*$ (请注意常数的拉普拉斯变换不存在，这是个数学上的bug)
- 输入偏差：$U_k = x_k - \omega^*$

简化后的标准线性差分方程为：

$Y_{k+1} = (1-\alpha)Y_k + \alpha U_k$

2.1 传递函数推导 (Z域分析)

对简化方程两边进行 Z 变换（假设初始条件为零）：

$zY(z) = (1-\alpha)Y(z) + \alpha U(z)$

移项整理：

$[z - (1-\alpha)]Y(z) = \alpha U(z)$

得到从输入 $U(z)$ 到输出 $Y(z)$ 的传递函数 $H(z)$：

$$H(z) = \frac{Y(z)}{U(z)} = \frac{\alpha}{z - (1-\alpha)}$$

或者表示为负幂次形式（更适用于信号处理上下文）：

$$H(z) = \frac{\alpha z^{-1}}{1 - (1-\alpha)z^{-1}}$$

2.2 系统特性分析

极点与稳定性

- **极点位置**：$p = 1 - \alpha$
- **稳定性判据**：极点必须位于单位圆内，即 $|1-\alpha| < 1$。
- **参数范围**：
- 理论稳定范围：$0 < \alpha < 2$

- 工程常用范围：$0 < \alpha < 1$（无振荡收敛）

物理意义：一阶低通滤波器

该系统本质上是一个**指数加权移动平均（EMA）滤波器**。

- **当** $\alpha \to 0$：极点接近 1，带宽窄，滤波效果强，系统响应慢（大惯性）。
- **当** $\alpha \to 1$：极点接近 0，带宽宽，滤波效果弱，系统响应快（小惯性）。

如果 x_k 是慢变的，那么需要牺牲一部分的惯性来提高带宽，以对 x_k 进行跟踪。仍然需要注意的是，但凡过一个滤波器就会产生相位滞后！如果 x_k 不是一个类阶跃的变化，而是有类斜坡的变化，将产生相位滞后。

2.4 稳态响应分析

直流增益 (DC Gain)

利用终值定理或直接令 $z = 1$ 计算系统的静态增益：

$$G_{DC} = H(1) = \frac{\alpha}{1 - (1 - \alpha)} = 1$$

结论： 系统具有**单位增益**。这意味着如果输入 x_k 发生阶跃变化，输出 ω_k 最终将无静差地跟踪该变化。

4.2 频域响应 (抗噪性)

令 $z = e^{j\omega T}$ 分析频率特性：

- **低频信号 ($\omega \to 0$)**：增益 ≈ 1，信号无衰减通过。
- **高频噪声 ($\omega \to \pi/T$)**：增益 $\approx \frac{\alpha}{2 - \alpha}$。当 α 较小时，高频增益接近 0。

结论： 系统能有效滤除叠加在输入信号上的高频噪声，保留直流和低频分量。

2.5 时域解析解 (逆Z变换)

我们通过逆 Z 变换推导系统对**阶跃输入**的时域响应公式。

5.1 设定输入

假设输入 x_k 相对于 ω^* 发生幅值为 A 的阶跃：

$$U(z) = \frac{Az}{z - 1}$$

5.2 求解响应 $Y(z)$

$$Y(z) = H(z)U(z) = \frac{\alpha A z}{(z-1)(z-(1-\alpha))}$$

5.3 部分分式展开

对 $\frac{Y(z)}{z}$ 进行展开：

$$\frac{Y(z)}{z} = \frac{A}{z-1} - \frac{A}{z-(1-\alpha)}$$

两边同乘 z：

$$Y(z) = \frac{Az}{z-1} - \frac{Az}{z-(1-\alpha)}$$

5.4 逆变换与还原

查表进行逆 Z 变换：

$$y_k = A \cdot 1^k - A \cdot (1-\alpha)^k = A[1-(1-\alpha)^k]$$

代回物理变量（$y_k = \omega_k - \omega^*$，$A = x_{target} - \omega^*$）：

$$\omega_k = \omega^* + (x_{target} - \omega^*) \cdot [1-(1-\alpha)^k]$$

2.6 总结

该差分方程描述了一个经典的**一阶惯性环节**（或一阶低通滤波器）。

1. **传递函数**：$$H(z) = \frac{\alpha}{z - (1-\alpha)}$$。
2. **稳态特性**：系统是稳定的（当 $0<\alpha<2$），且具有单位直流增益，能够实现**无静差跟踪**。
3. **动态特性**：输出 ω_k 从初始值 ω^* 开始，以 $(1-\alpha)^k$ 的速率指数收敛至目标输入值 x_{target}。
4. **参数调节**：调节 α 是权衡“响应速度”与“抗噪能力”的关键。

3. 代码

代码块

```
%% Dvoretzky 离散控制系统仿真：阶跃响应分析
% 系统：Δ_{k+1} = (1-α_k)Δ_k + α_k·η_k
% 其中：Δ_k = ω_k - ω*，η_k = x_k - ω*
% 等价于：ω_{k+1} = (1-α_k)ω_k + α_k·x_k

clear all; close all; clc;
```

```
7
8
9   %% 参数设置
10  N = 1000;            % 仿真步数
11  omega_star = 0;      % 目标值（可以调整）
12  omega0 = 5;          % 初始估计值
13
14  % 定义阶跃输入（多台阶）
15  step_times = [0, 200, 400, 600, 800, N];  % 台阶切换时间点
16  step_values = [0, 1, -0.5, 2, -1, 0];      % 每个台阶的值
17
18  % 生成阶跃输入序列 x_k
19  x = zeros(1, N);
20  for i = 1:length(step_times)-1
21      start_idx = step_times(i)+1;
22      end_idx = step_times(i+1);
23      x(start_idx:end_idx) = step_values(i);
24  end
25
26  % 初始化变量
27  omega = zeros(1, N);
28  omega(1) = omega0;
29  Delta = zeros(1, N);
30  Delta(1) = omega(1) - omega_star;
31  eta = zeros(1, N);
32  %%%%%%%%%%
33  % 增益序列 α_k = 1/k （从k=2开始，避免除以0）α收敛太快，导致系统收敛太慢（极点近单位圆）
34  % alpha = zeros(1, N);
35  % alpha(1) = 1;  % α_1 = 1
36  % for k = 2:N
37  %     alpha(k) = 1/k;
38  % end
39  %%%%%%%%%%
40  alpha = zeros(1, N);
41  alpha(1) = 1;  % α_1 = 1  系数过大可能会把系统干不稳定
42  for k = 2:N
43      alpha(k) = 10/k;  % 这里的系数还好，可以稍微大一点，这又是个有趣的时变系统收敛性问
    题。
44  end
45  %%%%%%%%%%
46  %%固定alpha
47  % alpha = 0.9*ones(1, N);
48  %%%%%%%%%%
49
50  %% 系统仿真
51  for k = 1:N-1
52      % 计算 η_k = x_k - ω*
```

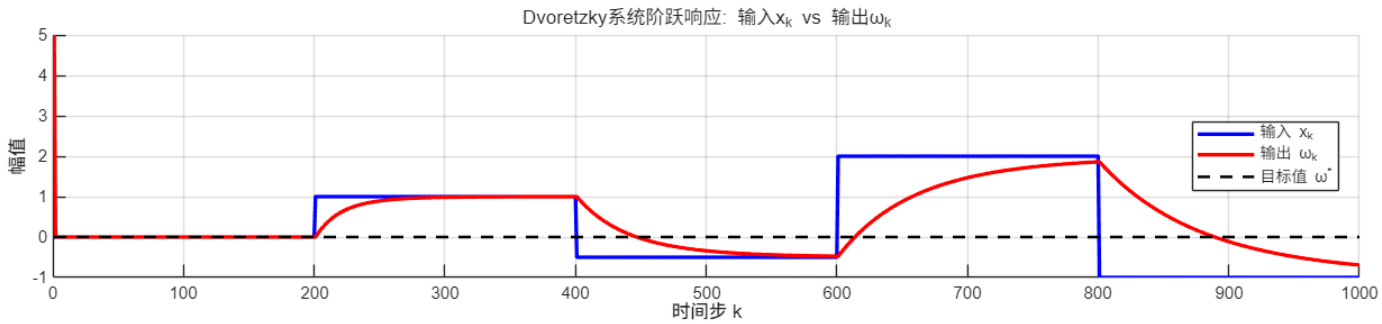
```
    eta(k) = x(k) - omega_star;

    % 更新 Δ_{k+1} = (1-α_k)Δ_k + α_k·η_k
    Delta(k+1) = (1 - alpha(k)) * Delta(k) + alpha(k) * eta(k);

    % 计算 ω_{k+1} = Δ_{k+1} + ω*
    omega(k+1) = Delta(k+1) + omega_star;
end

% 计算最后一个η值
eta(N) = x(N) - omega_star;

%% 绘制结果
figure('Position', [100, 100, 1200, 800]);

% 子图1：输入x_k和输出ω_k随时间变化
subplot(3, 2, [1, 2]);
hold on;
plot(1:N, x, 'b-', 'LineWidth', 2, 'DisplayName', '输入 x_k');
plot(1:N, omega, 'r-', 'LineWidth', 2, 'DisplayName', '输出 ω_k');
plot([1, N], [omega_star, omega_star], 'k--', 'LineWidth', 1.5, 'DisplayName',
'目标值 ω^*');
xlabel('时间步 k');
ylabel('幅值');
title('Dvoretzky系统阶跃响应: 输入x_k vs 输出ω_k');
legend('Location', 'best');
grid on;
hold off;

% 子图2：增益序列α_k
subplot(3, 2, 3);
semilogy(1:N, alpha, 'g-', 'LineWidth', 2);
xlabel('时间步 k');
ylabel('α_k (对数尺度)');
title('时变增益序列: α_k = 1/k');
grid on;

% 子图3：估计误差Δ_k
subplot(3, 2, 4);
plot(1:N, Delta, 'm-', 'LineWidth', 2);
hold on;
plot([1, N], [0, 0], 'k--', 'LineWidth', 1.5);
xlabel('时间步 k');
ylabel('估计误差 Δ_k');
title('估计误差动态: Δ_k = ω_k - ω^*');
grid on;
hold off;
```

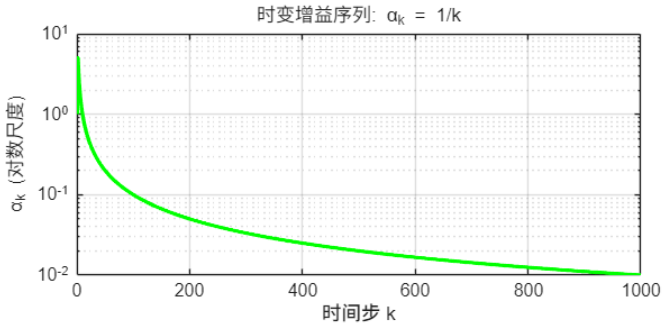
```
99
100 % 子图4：η_k序列
101 subplot(3, 2, 5);
102 plot(1:N, eta, 'c-', 'LineWidth', 2);
103 hold on;
104 plot([1, N], [0, 0], 'k--', 'LineWidth', 1.5);
105 xlabel('时间步 k');
106 ylabel('η_k');
107 title('观测偏差: η_k = x_k - ω^*');
108 grid on;
109 hold off;
110
111 % 子图5：系统动态的相平面分析 (Δ_k vs Δ_{k+1})
112 subplot(3, 2, 6);
113 plot(Delta(1:N-1), Delta(2:N), 'b.', 'MarkerSize', 8);
114 hold on;
115 plot([min(Delta), max(Delta)], [min(Delta), max(Delta)], 'k--', 'LineWidth',
    1.5);
116 xlabel('Δ_k');
117 ylabel('Δ_{k+1}');
118 title('系统相平面: Δ_{k+1} vs Δ_k');
119 axis equal;
120 grid on;
121 hold off;
122
123 %% 分析稳态性能
124 % 提取每个台阶稳定后的数据
125 steady_state_values = zeros(length(step_times)-1, 3);
126 steady_state_errors = zeros(length(step_times)-1, 1);
127
128 for i = 1:length(step_times)-1
129     start_idx = step_times(i)+1;
130     end_idx = step_times(i+1);
131
132     % 取后1/3作为稳态区间
133     steady_start = floor(start_idx + (end_idx - start_idx) * 2/3);
134     steady_range = steady_start:end_idx;
135
136     if ~isempty(steady_range)
137         steady_state_values(i, 1) = mean(x(steady_range));       % 输入平均值
138         steady_state_values(i, 2) = mean(omega(steady_range));  % 输出平均值
139         steady_state_values(i, 3) = mean(Delta(steady_range));  % 误差平均值
140         steady_state_errors(i) = abs(steady_state_values(i, 3)); % 绝对误差
141     end
142 end
143
144 %% 显示稳态分析结果
```

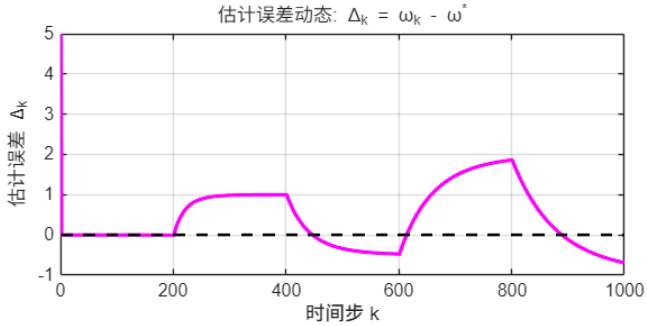
```
fprintf('========== Dvoretzky系统阶跃响应分析 ==========\n');
fprintf('仿真参数: N=%d, ω*=%d, ω0=%d\n\n', N, omega_star, omega0);
fprintf('阶跃输入配置:\n');
for i = 1:length(step_times)-1
    fprintf('  台阶%d: 时间[%d, %d], 幅值=%.2f\n', ...
        i, step_times(i), step_times(i+1), step_values(i));
end
fprintf('\n');

fprintf('稳态性能分析:\n');
fprintf('台阶 | 输入均值 | 输出均值 | 误差均值 | 绝对误差\n');
fprintf('-----|----------|----------|----------|----------\n');
for i = 1:length(step_times)-1
    fprintf('%4d | %8.4f | %8.4f | %8.4f | %8.4f\n', ...
        i, steady_state_values(i, 1), steady_state_values(i, 2), ...
        steady_state_values(i, 3), steady_state_errors(i));
end
fprintf('\n');

% 计算总体性能指标
final_error = abs(Delta(N));
fprintf('总体性能指标:\n');
fprintf('最终估计误差: %.6f\n', final_error);
fprintf('误差最大值: %.6f\n', max(abs(Delta)));
fprintf('误差均方根(RMS): %.6f\n', rms(Delta));
fprintf('\n');

%% 添加频域分析
figure('Position', [100, 100, 1000, 400]);

% 计算频率响应
Fs = 1;  % 采样频率为1 Hz（每个时间步一个样本）

% 输入和输出的频谱
N_fft = 2^nextpow2(N);
X_f = fft(x, N_fft);
Omega_f = fft(omega, N_fft);
frequencies = (0:N_fft-1)*(Fs/N_fft);

% 幅度谱
subplot(1, 2, 1);
semilogx(frequencies(2:N_fft/2), 20*log10(abs(X_f(2:N_fft/2))), 'b-',
'LineWidth', 1.5);
hold on;
semilogx(frequencies(2:N_fft/2), 20*log10(abs(Omega_f(2:N_fft/2))), 'r-',
'LineWidth', 1.5);
xlabel('频率 (Hz)');
```

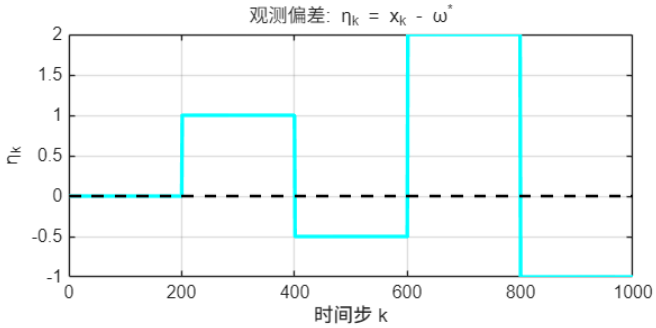
```
ylabel('幅度 (dB)');
title('输入/输出幅度谱');
legend('输入 x_k', '输出 ω_k');
grid on;
hold off;

% 估计系统的频率响应
H_est = Omega_f ./ X_f;
H_est(1) = 1;  % 避免直流分量除零

subplot(1, 2, 2);
semilogx(frequencies(2:N_fft/2), 20*log10(abs(H_est(2:N_fft/2))), 'g-',
'LineWidth', 2);
xlabel('频率 (Hz)');
ylabel('幅度 (dB)');
title('估计系统频率响应 |H(f)|');
grid on;

%% 对比不同增益策略的效果
figure('Position', [100, 100, 1200, 400]);

% 定义不同的增益策略
alpha_strategies = cell(3, 1);
alpha_strategies{1} = 1./(1:N);              % α_k = 1/k
alpha_strategies{2} = 10./(10 + (1:N));      % α_k = 10/(10+k)
alpha_strategies{3} = 0.1./(0.1 + (1:N).^0.5); % α_k = 0.1/(0.1+√k)

strategy_names = {'α_k = 1/k', 'α_k = 10/(10+k)', 'α_k = 0.1/(0.1+√k)'};
colors = {'r', 'g', 'b'};

% 仿真不同增益策略
omega_results = zeros(3, N);
for s = 1:3
    omega_s = zeros(1, N);
    omega_s(1) = omega0;

    for k = 1:N-1
        omega_s(k+1) = (1 - alpha_strategies{s}(k)) * omega_s(k) + ...
                       alpha_strategies{s}(k) * x(k);
    end

    omega_results(s, :) = omega_s;
end

% 绘制对比结果
subplot(1, 2, 1);
hold on;
```

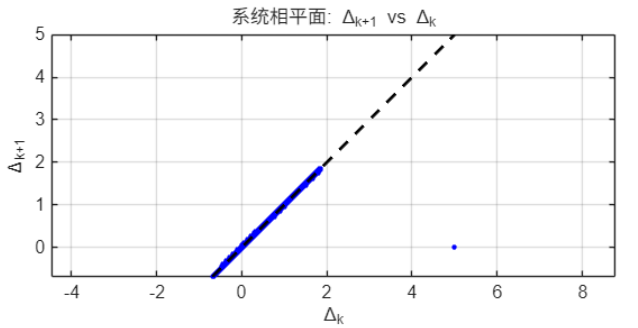
```
plot(1:N, x, 'k-', 'LineWidth', 3, 'DisplayName', '输入 x_k');
for s = 1:3
    plot(1:N, omega_results(s, :), colors{s}, 'LineWidth', 2, ...
         'DisplayName', strategy_names{s});
end
plot([1, N], [omega_star, omega_star], 'k--', 'LineWidth', 1.5, 'DisplayName',
'ω^*');
xlabel('时间步 k');
ylabel('幅值');
title('不同增益策略的阶跃响应对比');
legend('Location', 'best');
grid on;
hold off;

% 绘制增益序列对比
subplot(1, 2, 2);
hold on;
for s = 1:3
    plot(1:N, alpha_strategies{s}, colors{s}, 'LineWidth', 2, ...
         'DisplayName', strategy_names{s});
end
xlabel('时间步 k');
ylabel('α_k');
title('不同增益序列对比');
set(gca, 'YScale', 'log');
legend('Location', 'best');
grid on;
hold off;
```

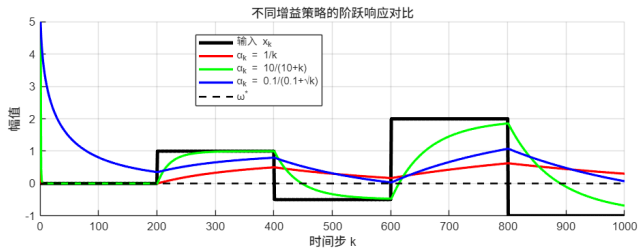


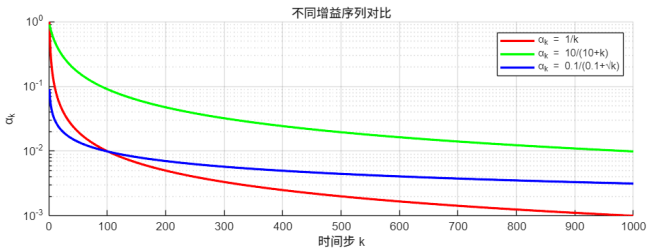




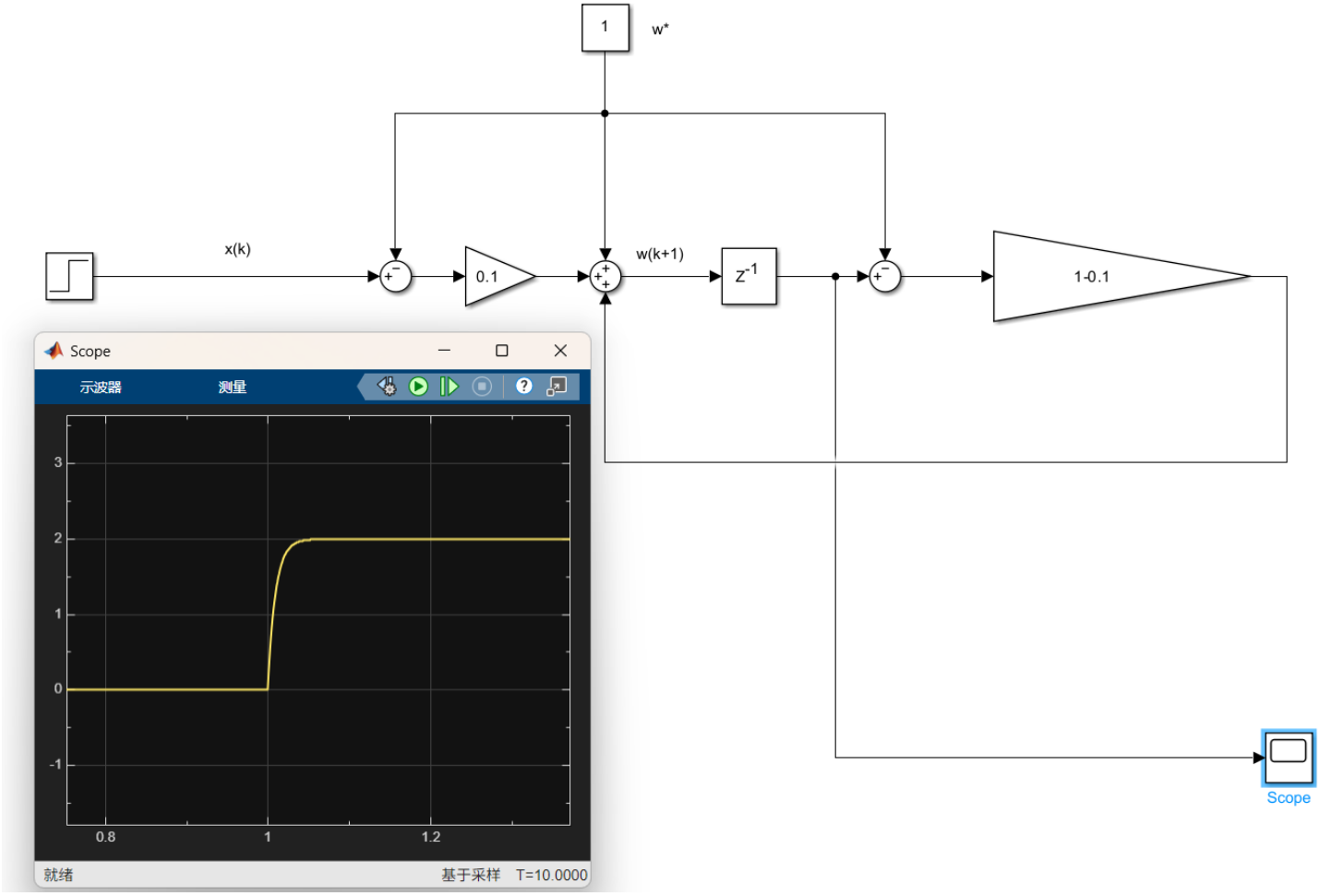




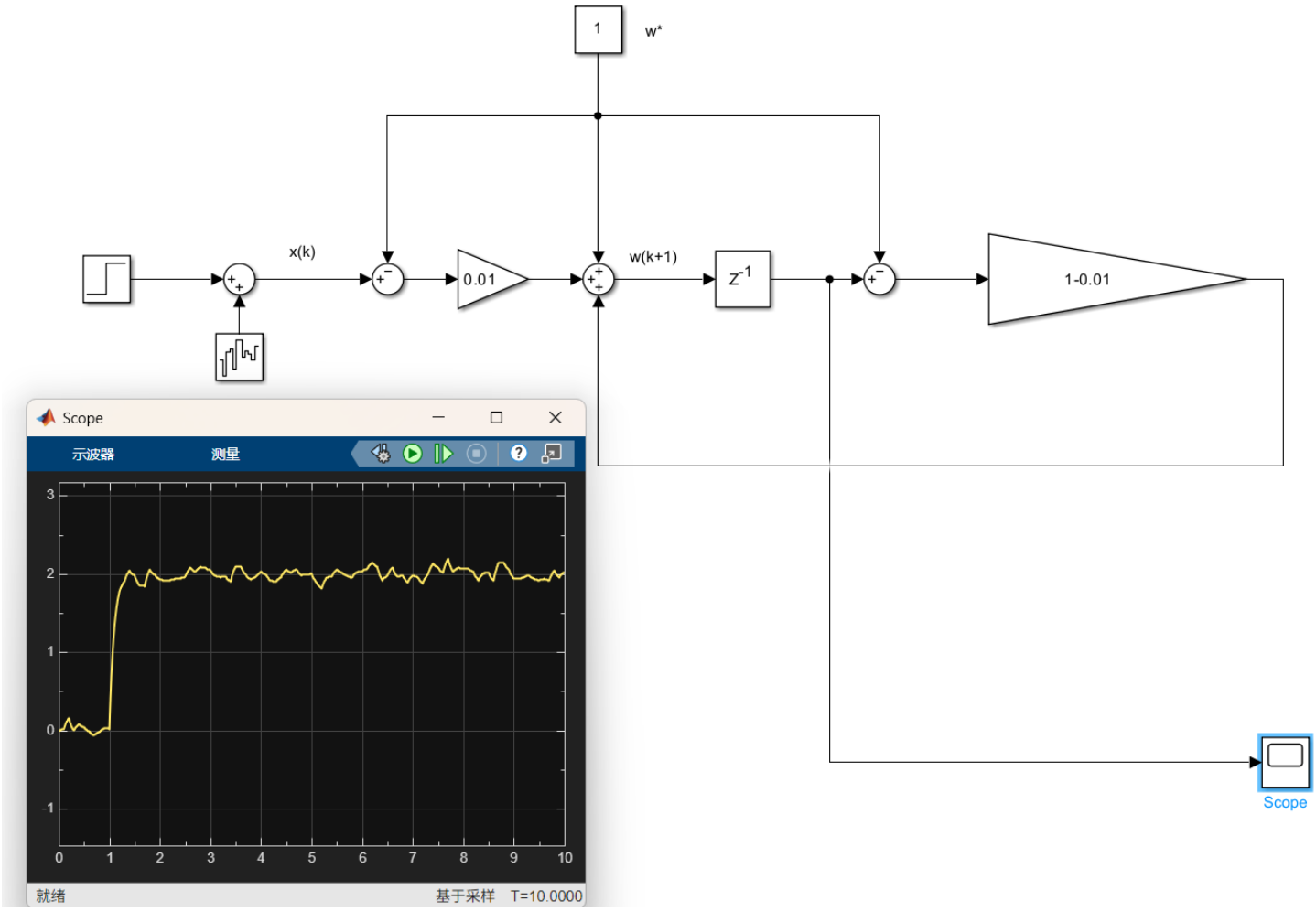




可以用simulink搭出这个系统的控制框图


1
w*
x(k)
0.1
w(k+1)
Z^{-1}
1-0.1
Scope
示波器
测量
3
2
1
0
-1
0.8
1
1.2
就绪
基于采样 T=10.0000
Scope

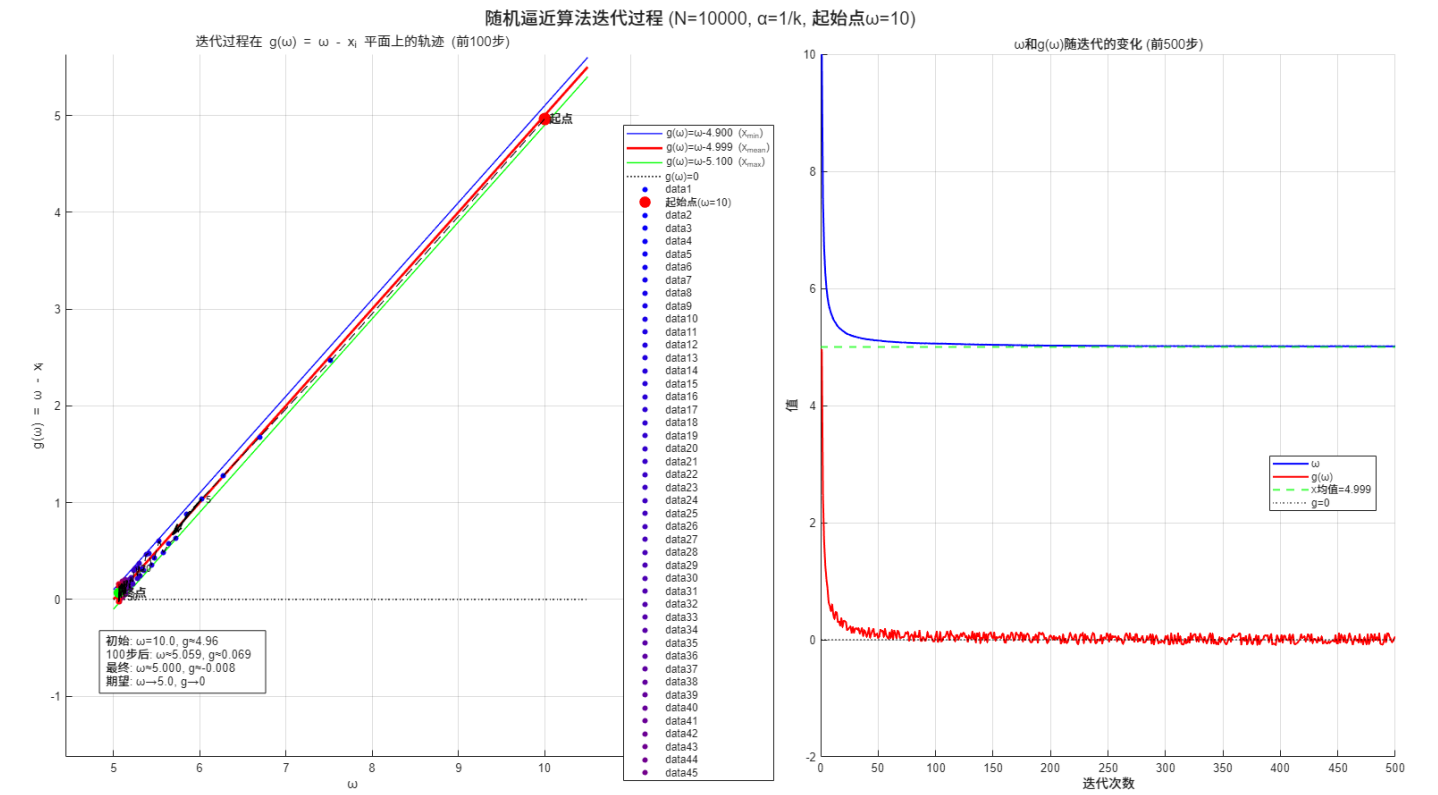
加上噪声：


1
w*
x(k)
0.01
w(k+1)
z-1
1-0.01
Scope
Scope
示波器
测量
3
2
1
0
-1
0
1
2
3
4
5
6
7
8
9
10
就绪
基于采样 T=10.0000

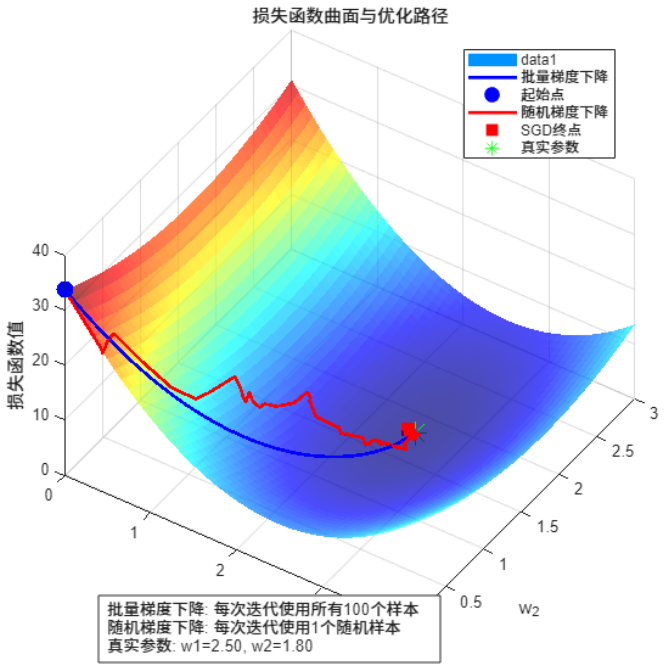
4. 书上的理解方式

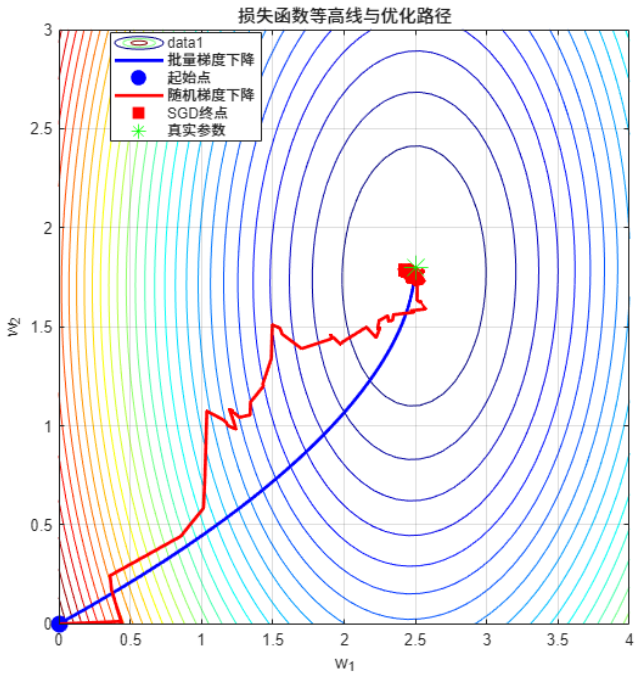
RM.m

这个代码里有公式推导，用matlab实时编辑器打开可以看到

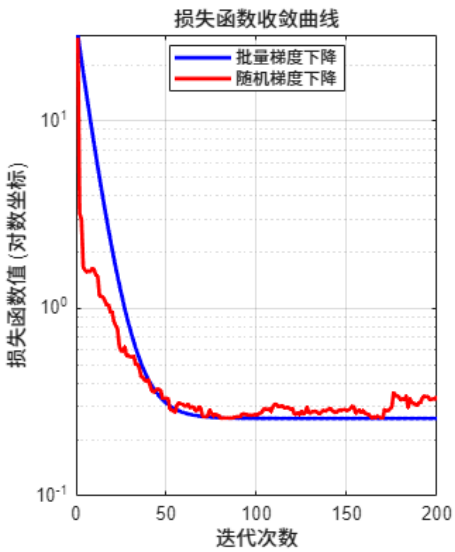


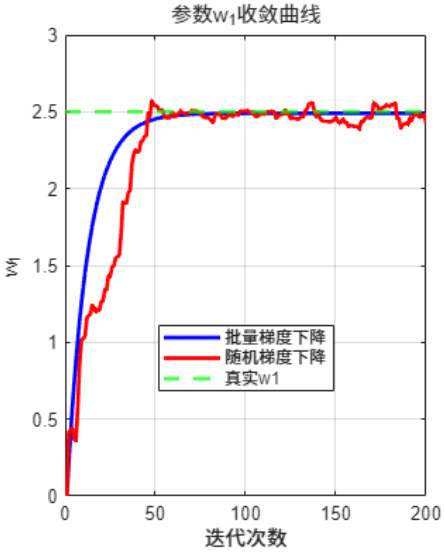
第一种理解方法，对应书上的图6.4

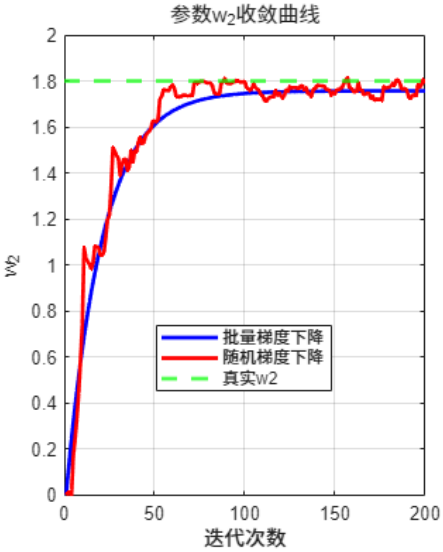


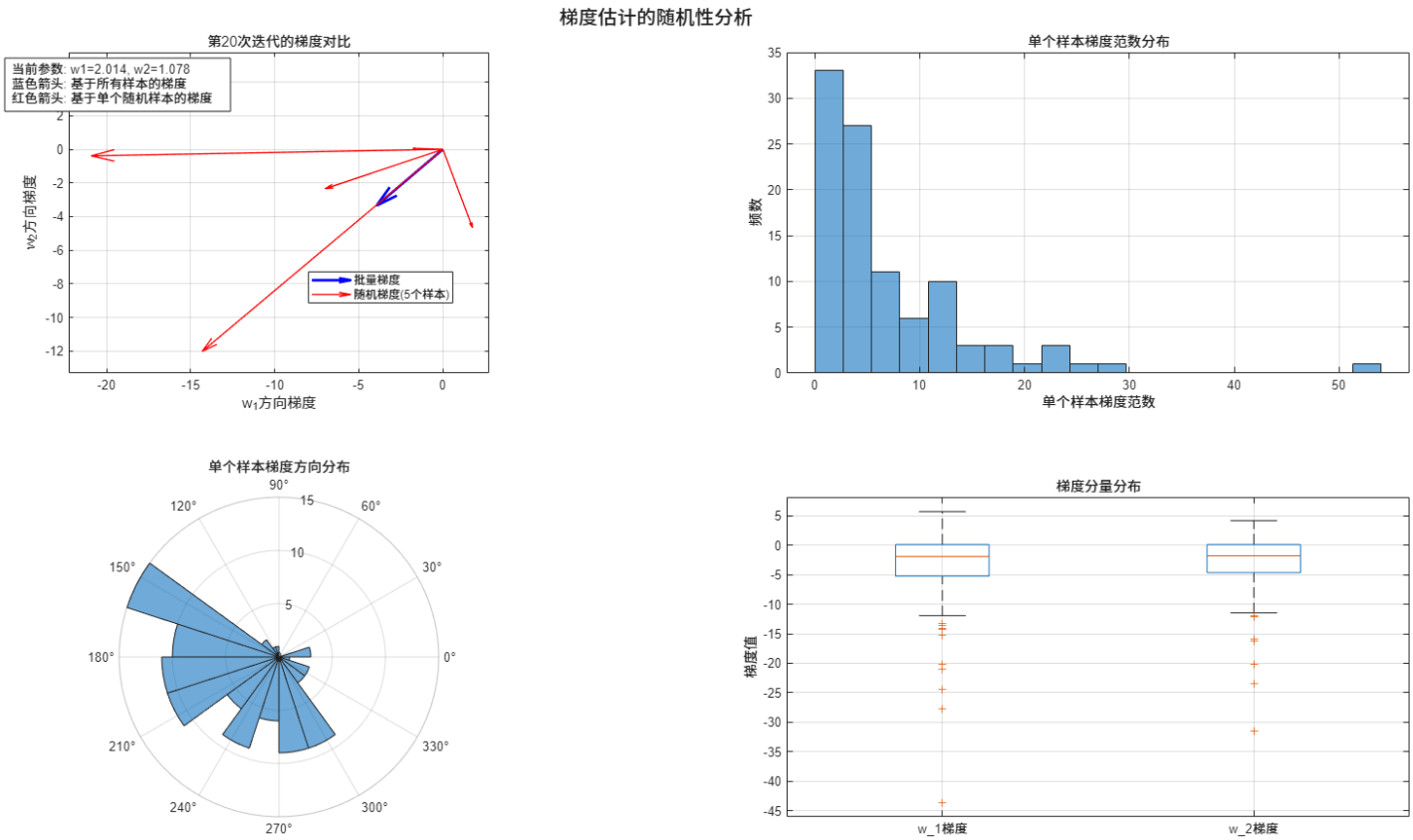


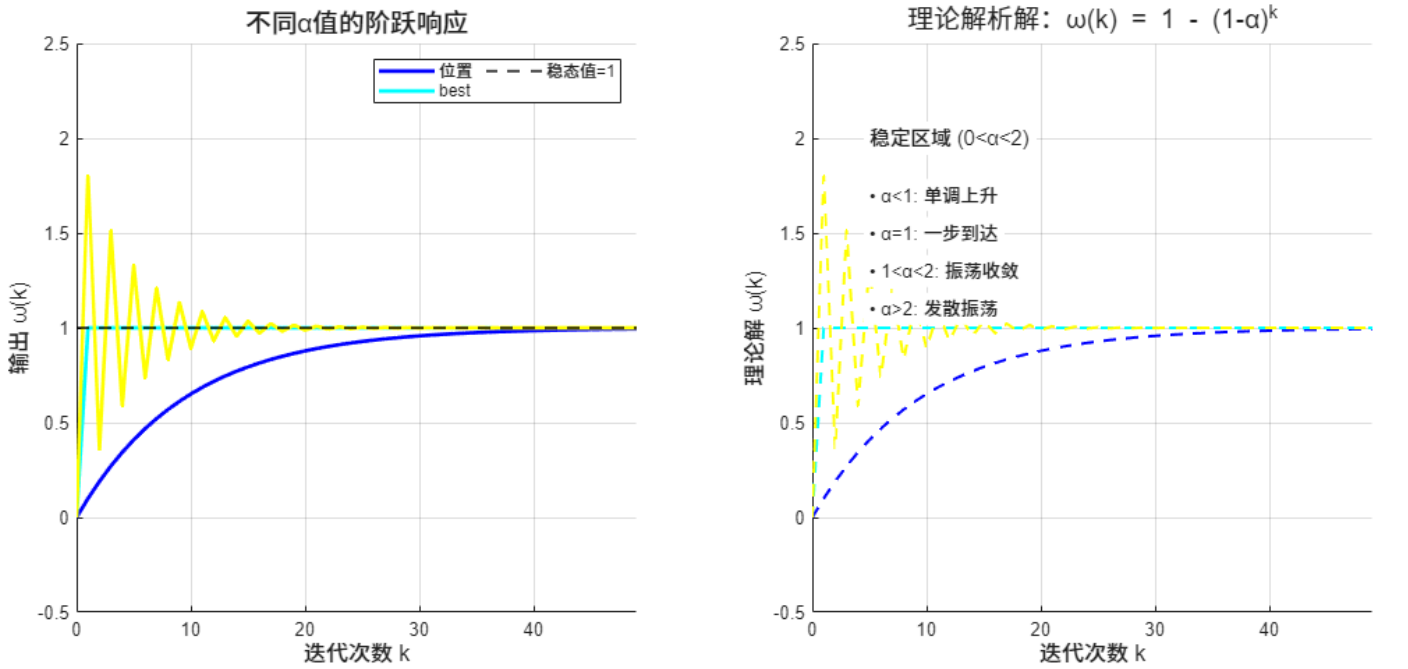
第二种理解方法，对应书上的图6.5











第1.2.3节阐述的理解方法，对应书上的6.3节Dvoretzky