统计过程控制（SPC）

分析报告

2024 至 2025 学年第 二 学期

学号：20232373

姓名：莫湘渝

国家卓越工程师学院明月科创班

一. 背景与目标

1.1 背景分析

统计过程控制(SPC)作为现代质量管理的核心技术,起源于 1924 年 Shewhart 博士提出的控制图理论。其核心思想是通过统计方法识别生产过程中的随机波动与异常波动，区分普通原因（随机因素）与特殊原因（系统性因素），从而实现对过程的稳定性和能力评估。SPC 的广泛应用（如 Motorola 的 6σ 管理）表明，其在提升产品合格率、降低生产成本方面具有显著优势。

本项目的背景基于某工厂滚珠直径生产场景，其核心挑战在于：

工艺水平评估：需量化产品尺寸是否符合规格要求（如规格上下限 USL/LSL），并通过工序能力指数（如 Cp、Cpk）判断工艺是否满足客户需求。

生产过程受控性判定：需验证生产过程是否仅受随机因素影响（受控状态），或存在异常因素（失控状态），例如设备偏差、操作失误等。

1.2 研究目标

基于 SPC 理论框架，本项目通过 Matlab 工具实现以下目标：

1.描述性统计：计算均值、极差、标准差等指标，初步识别数据分布特征。

2.正态性检验：验证数据是否符合正态分布，为后续参数检验与控制图绘制提供依据。

3.均值检验：通过 t 检验评估总体均值是否偏离目标值（如规格中心值），判断系统性偏差的存在性。

4.工艺能力与受控性评估

工序能力指数：计算 Cp（过程潜力）与 Cpk（实际能力），量化工艺水平。若 Cpk≥1.33 则视为优秀，低于 1.0 则需改进。

控制图分析：绘制 Shewhart 均值-极差控制图（$\bar{X}$ -R 图），通过数据点与控制限（UCL/LCL）的对比，识别异常波动点，判断生产过程是否受控。

5.决策支持与改进建议

1.3 技术路径

1.数据模拟与采集：生成 25 组样本数据（n≥5），模拟实际生产波动，包含正态基础数据与非正态噪声（如对数正态、伽马分布）。

2.SPC 工具链构建：

分析阶段：通过直方图、概率图验证分布假设，确定初始控制限。

监控阶段：动态更新控制图，实时反馈过程状态，触发预警机制。

3.可视化与报告：利用 Matlab 图形界面展示控制图、能力指数及假设检验结果，输出工艺改进优先级清单。

1.4 理论意义与工业价值

本项目通过 SPC 的完整实施流程（分析→监控→改进），体现了其在制造业中的双重价值：

1.质量保障：通过预防性控制减少不良品率，避免事后检验的成本浪费。

2.持续改进：结合 PDCA 循环（计划-执行-检查-处理），推动工艺优化与六西格玛目标实现。

二. 实验详述

2.1 数据生成

本次实验的关键参数如下，分别包含目标均值，标准差，样本量，子组数量，规格上下限，噪声强度等。

```
%% 参数设置
clc; clear; close all;
rng(0); % 固定随机种子
mu = 10;           % 目标均值
sigma = 0.5;       % 主过程标准差
n_subgroups = 25; % 子组数量
n_samples = 5;     % 样本量
noise_scale = 0.03;% 噪声强度
USL = 11.5;        % 规格上限
LSL = 8.5;         % 规格下限
```

依据软件随机生成 25 组每组 5 个数据，其中插入非正态噪声，数据模型如下：

$$X_{ij} = \underbrace{\mu + \sigma Z_{ij}}_{\text{主过程}} + \underbrace{\epsilon(0.3)}_{\text{非正态噪声}}$$

- $Z_{ij} \sim N(0,1)$
- $\epsilon \sim$ 指数分布 − 偏移量

具体代码如下：

```
%% 1. 生成带非正态噪声的数据
main_data = mu + sigma * randn(n_subgroups, n_samples);
exp_noise = noise_scale*(exprnd(1, n_subgroups, n_samples) - noise_scale); % 零均值指数噪声
data = main_data + exp_noise;
```

生成的总体样本如下：其中每行表示一组数据，共 25 组，每组中有 5 个数据

10.3084	10.5182	9.5803	9.3169	10.4599
10.9227	10.3839	10.0617	9.3193	9.5819
8.8949	9.8688	9.4101	10.2451	10.0647
10.4330	10.1786	9.4788	9.9144	9.7670
10.2096	9.6086	10.0045	9.9190	10.1660
9.3852	10.4732	10.8154	10.7230	9.7092
9.8401	9.4915	9.6255	10.1609	10.2893
10.2303	9.4721	10.2357	10.1451	10.4330
11.7925	9.6226	9.9163	10.8289	10.8915
11.4002	8.5696	10.5718	9.6195	9.9363
9.3421	10.7455	9.4620	10.3914	8.9557
11.5745	10.2319	10.0907	10.4217	9.5996
10.3666	9.6824	10.2776	9.9263	10.7502
9.9818	10.6860	10.5570	10.1516	9.5031
10.3879	9.1447	10.7928	9.4692	10.4862
9.9166	9.9646	10.0670	9.4695	10.1671
9.9644	9.9629	9.2775	10.0765	10.7197
10.8213	10.2022	9.6634	10.3953	9.0281
10.7464	10.1868	9.4886	11.2942	9.9217
10.7705	9.5726	11.1945	9.6910	9.4116
10.3856	10.1093	9.6973	10.1434	11.4963
9.4382	10.0110	10.3800	9.9609	10.4351
10.3839	10.3663	9.9161	9.0332	10.6897
10.9043	10.5587	10.4725	9.8043	9.4881
10.2466	10.5631	9.6229	9.1677	9.7843

2.2 描述性统计

描述性统计量主要有以下几种：

均值 (Mean)：计算每组数据的平均值，反映该组数据的中心位置。

方差 (Variance)：度量数据分散程度，数值越大表示数据的波动越大。

标准差 (Standard Deviation)：方差的平方根，同样反映数据的分散程度，但单位与数据相同。

极差 (Range)：最大值与最小值的差，表示数据的跨度。

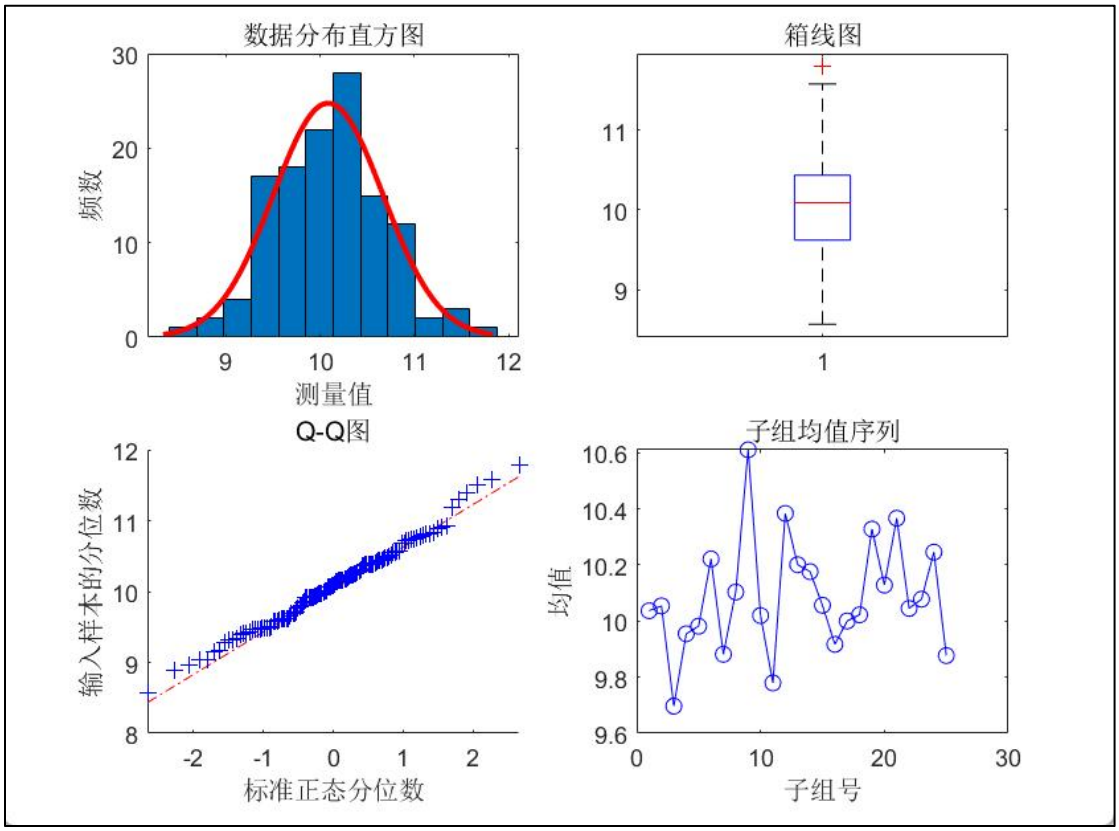
对所有数据进行描述性统计，分别计算子组方差均值，总体均值方差，极差，以及绘制数据直方图，计算结果如下：

Matlab 中计算方差的函数 var(A) 默认会按照 N-1，可以作为对于总体分布的方差的无偏估计。下述得到的均为修正后的样本方差。

```
%% 2. 描述性统计分析
subgroup_means = mean(data, 2);      % 子组均值
subgroup_vars = var(data, 0, 2);     % 子组方差
subgroup_stds = std(data, 0, 2);     % 子组标准差
subgroup_ranges = range(data, 2);    % 子组极差
overall_mean = mean(subgroup_means);% 总平均值
average_std = mean(subgroup_stds);   % 平均标准差
overall_std = std(data(:));          % 总体标准差
```

```
============ 描述性统计 ============
总平均值 = 10.0865
平均标准差 = 0.5809
总体标准差 = 0.5841
最大极差 = 2.8306, 最小极差 = 0.6010
```

使用 histfit(data(:)); boxplot(data(:)); qqplot(data(:)); plot(subgroup_means, 'bo-');分别绘制数据直方图，总体数据箱线图，分位-分位图，各子组均值分布图



通过对于上述四个描述性统计量的样本观测结果进行初步分析，可以得到如下结论：

①.总体平均值约为 10.08，可作为后续总体均值检验的检验目标。

②.各组数据的方差均在 0.5 以外，总体方差与标准差较高，说明生产过中数据有部分波动，产品质量稳定性待提高。

③.数据极差范围适中，总体极差不超过 3，处于合理区间。

④.数据的频数分布大致呈”两边少，中间多”的趋势，推测其总体大致服从正态分布，需要进行进一步检验。

3.3 正态性检验

3.3.1 调库计算正态性：

Lilliefors 检验是 Kolmogorov-Smirnov（K-S）检验的改进版本，专用于检验数据是否服从正态分布。其核心思想是通过比较样本数据的经验分布函数（EDF）与理论正态分布函数之间的最大差异，判断数据是否偏离正态性假设。

基本步骤包括参数估计，使用样本数据均值，标准差作为正态分布参数估计值；计算经验分布函数（EDF），对样本数据排序，计算每个数据点的累积概率；计算理论分布函数（CDF），对每个排序后的数据，计算其在理论正态分布下的累积概率；检验统计量，统计量为 EDF 与 CDF 之间的最大绝对差异 D；通过蒙特卡洛模拟或查表获取临界值 Dα，若 D>Dα，则拒绝原假设（数据不服从正态分布）。

```
[h_norm, p_norm] = lillietest(data(:));
fprintf('============ 正态性检验 ============\n');
fprintf('Lilliefors检验: p=%.4f\n', p_norm);
if h_norm
    fprintf('结论：数据拒绝正态性假设\n');
else
    fprintf('结论：数据服从正态分布\n');
end
```

```
警告：P 大于最大列表值，将返回 0.5。
============ 正态性检验 ============
Lilliefors检验: p=0.5000
结论：数据服从正态分布
```

结论：检验显示，通过对样本数据进行正态性检验，可认为数据来自正态分布的总体，正态性检验的 p 值为大于 0.05。含噪声数据基本服从正态分布，不拒绝原假设，即数据服从正态分布。

3.3.2 手动计算正态性：

① 提出假设：

原假设（H_0）：数据服从正态分布。

对立假设（H_1）：数据不服从正态分布。

② 参数估计：

估计均值（μ）：用样本均值 X 作为正态分布均值的估计值。

估计标准差（σ）：用样本标准差（有偏，分母为 n）作为正态分布标准差的估计值。

```
% 参数校验与默认设置
if nargin < 2 || isempty(alpha), alpha = 0.05; end
data = data(:); % 确保列向量
n = length(data);

% 1. 参数估计（极大似然估计）
mu = mean(data);
sigma = std(data, 1); % 使用有偏标准差（除以n）
```

③ 数据分箱：
分箱数：使用 Sturges 公式自动计算初始分箱数：
分箱数=1+log2(n)
调整分箱：确保每个分箱的理论频数 ≥5，否则合并相邻分箱。

```
% 2. 数据分箱
if nargin < 3 || isempty(numBins)
    % Sturges公式计算分箱数
    numBins = max(ceil(1 + log2(n)), 5); % 至少5个分箱
end
[binCounts, binEdges] = histcounts(data, numBins);
numBins = length(binCounts); % 实际分箱数可能调整
```

④ 计算理论频数：
对每个分箱区间 [a,b)，理论概率：计算正态分布在该区间的累积概率：

$$p_i = P(a \le X < b) = \Phi\left(\frac{b-\mu}{\sigma}\right) - \Phi\left(\frac{a-\mu}{\sigma}\right)$$

理论频数：Ei=n×pi

```
% 3. 计算理论频数
p_hat = zeros(1, numBins);
% 第一个分箱 (-inf, binEdges(2)]
p_hat(1) = normcdf(binEdges(2), mu, sigma);
% 中间分箱
for i = 2:numBins-1
    p_hat(i) = normcdf(binEdges(i+1), mu, sigma) - normcdf(binEdges(i), mu, sigma);
end
% 最后一个分箱 [binEdges(end-1), +inf)
p_hat(end) = 1 - normcdf(binEdges(end-1), mu, sigma);
```

```
% 4. 合并小期望频数的分箱（要求每个分箱期望频数≥5）
expected = n * p_hat;
while any(expected < 5) && numBins > 1
    % 找到最小期望频数的分箱并与相邻合并
    [~, idx] = min(expected);
    if idx == 1
        % 合并第一个和第二个
        binCounts(2) = binCounts(1) + binCounts(2);
        binCounts(1) = [];
        p_hat(2) = p_hat(1) + p_hat(2);
        p_hat(1) = [];
        binEdges(2) = [];
    elseif idx == numBins
        % 合并最后两个
        binCounts(end-1) = binCounts(end-1) + binCounts(end);
        binCounts(end) = [];
        p_hat(end-1) = p_hat(end-1) + p_hat(end);
        p_hat(end) = [];
        binEdges(end) = [];
    else
        % 合并当前分箱与前一个
        binCounts(idx-1) = binCounts(idx-1) + binCounts(idx);
        binCounts(idx) = [];
        p_hat(idx-1) = p_hat(idx-1) + p_hat(idx);
        p_hat(idx) = [];
        binEdges(idx) = [];
    end
    numBins = numBins - 1;
    expected = n * p_hat;
end
```

⑤ 计算卡方统计量：

$$\chi^2 = \sum_{i=1}^{k} \frac{(\text{实际频数} - \text{理论频数})^2}{\text{理论频数}}$$

实际频数：每个分箱中实际数据点的数量。
k：最终合并后的分箱数。

```
% 5. 计算卡方统计量
chi2_stat = sum((binCounts - n*p_hat).^2 ./ (n*p_hat));
```

⑥ 计算临界值与结论：
临界值：根据显著性水平（α=0.05）和自由度查卡方分布表。
若 x 2>临界值或 p<0.05→ 拒绝原假设（数据非正态）。否则 → 无法拒绝原假设（数据可能正态）。

```
% 6. 确定自由度（分箱数 - 1 - 估计参数个数）
df = numBins - 1 - 2; % 估计了μ和σ两个参数

% 7. 计算临界值与p值
criticalValue = chi2inv(1 - alpha, df);
p_value = 1 - chi2cdf(chi2_stat, df);

% 8. 判断结果
h = chi2_stat > criticalValue;
```

结论：不拒绝原假设与直接调用 Matlab 内置库函数的结果一致。

```
卡方统计量 = 4.4222
临界值（α=0.05, df=3） = 7.8147
P值 = 0.2193
结论：无法拒绝原假设（数据可能服从正态分布）
```

3.4 均值检验

单样本 t 检验用于判断样本均值是否与目标值（理论均值）存在显著差异。在本例中，目标值为工艺设计中心值 μ=10，检验生产过程是否发生均值偏移。

①设原假设（H0）：过程均值等于目标值；新假设（H1）：过程均值不等于目标值。检验统计量计算，t 统计量公式：

$$t = \frac{\bar{X} - \mu}{s/\sqrt{n}}$$

- $\bar{X}$：样本均值
- s：样本标准差
- n：样本量（本例中 $n = 25 \times 5 = 125$）

②置信区间构建，95%置信区间（CI）计算公式：

$$CI = \bar{X} \pm t_{\alpha/2,df} \cdot \frac{s}{\sqrt{n}}$$

③具体 matlab 代码实现：

```
%% 4. 总体均值检验
[h_mean, p_mean, ci] = ttest(data(:), mu);
fprintf('\n============ 均值检验 ============\n');
fprintf('t检验结果: p=%.4f, 95%%置信区间=[%.3f, %.3f]\n', p_mean, ci(1), ci(2));
if h_mean
    fprintf('结论：过程均值显著偏离目标值\n');
else
    fprintf('结论：过程均值与目标值无显著差异\n');
end
```

④结果分析，p> α =0.05，不拒绝原假设，结论为过程均值与目标值无显著差异（ μ =10），置信区间 9.983--10.190，包含 μ =10，进一步支持均值未偏移。

```
t检验结果: p=0.1005, 95%置信区间=[9.983, 10.190]
```

3.5 过程能力分析

前面求得样本基本服从正态分布，样本均值与规范中心不重合，样本标准差 s 是总体标准差的有偏估计量，故先使用无偏常数修正标准差，一般查表获得 n=5 时的 c4=0.940，

```
% c4 = 0.940; % n=5时的无偏常数
c4 = get_c4_constant(5);
sigma_est = average_std / c4; % 估计过程标准差
```

```
function c4 = get_c4_constant(n)
    % 获取c4无偏常数（n<=25时查表，更大n使用近似公式）
    c4_table = [NaN, 0.7979, 0.8862, 0.9213, 0.9400, 0.9515, 0.9594, 0.9650, 0.9693, 0.9727, ...
                0.9754, 0.9776, 0.9794, 0.9810, 0.9823, 0.9835, 0.9845, 0.9854, 0.9862, 0.9869, ...
                0.9876, 0.9882, 0.9887, 0.9892, 0.9896];
    if n <= 25
        c4 = c4_table(n);
    else
        c4 = sqrt(2/(n-1)) * gamma(n/2) / gamma((n-1)/2);
    end
end
```

之后计算偏离度 k，潜在工序能力和实际工序能力，使用公式如下：

```
has_USL = isfinite(USL);
has_LSL = isfinite(LSL);
T0 = (USL + LSL)/2; % 规范中心

% 4. 工序能力计算
Cp = NaN; Cpk = NaN; Cpu = NaN; Cpl = NaN;

% 双侧规范计算
if has_USL && has_LSL
    % 潜在工序能力指数
    Cp = (USL - LSL) / (6*sigma_hat);

    % 相对偏离度
    K = abs(mu_hat - T0) / ((USL - LSL)/2);

    % 实际工序能力指数
    if K >= 1
        Cpk = 0; % 均值超出规范范围
    else
        Cpk = Cp * (1 - K);
    end

    % 确保Cpk非负
    Cpk = max(Cpk, 0);
end
```

$$Cp = \frac{T_U - T_L}{6\sigma}$$

$$Cpk = \frac{T_U - T_L}{6\sigma}(1 - K)$$

$$K = \frac{|\mu - T_0|}{(T_U - T_L)/2}$$

```
=== 过程能力分析结果 ===
Cp  = 0.856
Cpk = 0.807
Cpu = 0.807
Cpl = 0.905
K   = 0.058
```

由于规范中心与参数分布中心 μ 不重合，上述的近似处理 CP 并不能很准确在的反映其工序能力，此时则需要采用实际工序能力指数 Cpk 来度量工艺水平的高低，公式中 K 用于度量工艺参数分布均值 μ 对规范中心 T_0 的相对偏离度，计算得到：k=0.058，实际工序能力指数 Cpk=0.807，其中相对偏离度 K<1，说明均值未偏离到规范范围外，表明工艺加工结果合适，且实际工序能力指数 Cpk>0，表明该工序具有一定的生产能力，但并未达到国际上的普遍要求 Cpk>1.5，说明生产水平还存在相当大的提升空间。

除此之外，还可以计算单侧规范下的实际工序能力指数 CPU 和 CPL，计算得到 CPU=0.807、CPL=0.905，也并未达到国际一般标准。

```
% 单侧规范计算
if has_USL
    Cpu = (USL - mu_hat) / (3*sigma_hat);
    Cpu = max(Cpu, 0); % 处理负值
end

if has_LSL
    Cpl = (mu_hat - LSL) / (3*sigma_hat);
    Cpl = max(Cpl, 0);
end
```

3.6 控制图分析

Shewhart 控制图是实施 SPC 过程中判断生产过程是否处于统计受控状态的基本工具，其中使用最为广泛的为均值控制图和方差（标准差）控制图。通过连续采集工艺参数数据，可以利用 Shewhart 控制图来定量分析和度量生产线的运行过程处于统计受控状态。

对于各批次的样本数据，用折线连接每批次的均值，从而得到反映不同批次数据均值波动情况的折线图。

对于服从正态分布总体 N(μ，σ^2)的工艺参数而言，其取值落在(μ ± 3σ)区间范围内的概率为 99.73%，因此可以采用如下的 3σ 准则来确定均值控制图的中心线以及上下控制限： $UCL = \mu + 3\sigma;\quad CL = \mu;\quad LCL = \mu - 3\sigma$

但在实际生产实践中，并不是所有的产品工艺指标都会近似服从正态分布。因此，在实际操作中，均值和标准差控制上下限按照查表法得到，其中，若采集工艺参数样本数据组数 k=25，各批次数据数量 n=5，，令 x_ij 表示第 i 批次的第 j 个数据，bar{x_i}表示组内均值；则可用 μ 表示个各批次所有数据的样本均值，hat{s}表示各个批次数据标准偏差的算数平均值，又查到 A=1.427，B=0；

```
A3 = 1.427; % n=5时的控制图常数
CL_X = overall_mean;
UCL_X = CL_X + A3*average_std;
LCL_X = CL_X - A3*average_std;

% S控制图参数
B3 = 0;      % n=5时的控制图常数
B4 = 2.089;
CL_S = average_std;
UCL_S = B4*CL_S;
LCL_S = B3*CL_S;
```

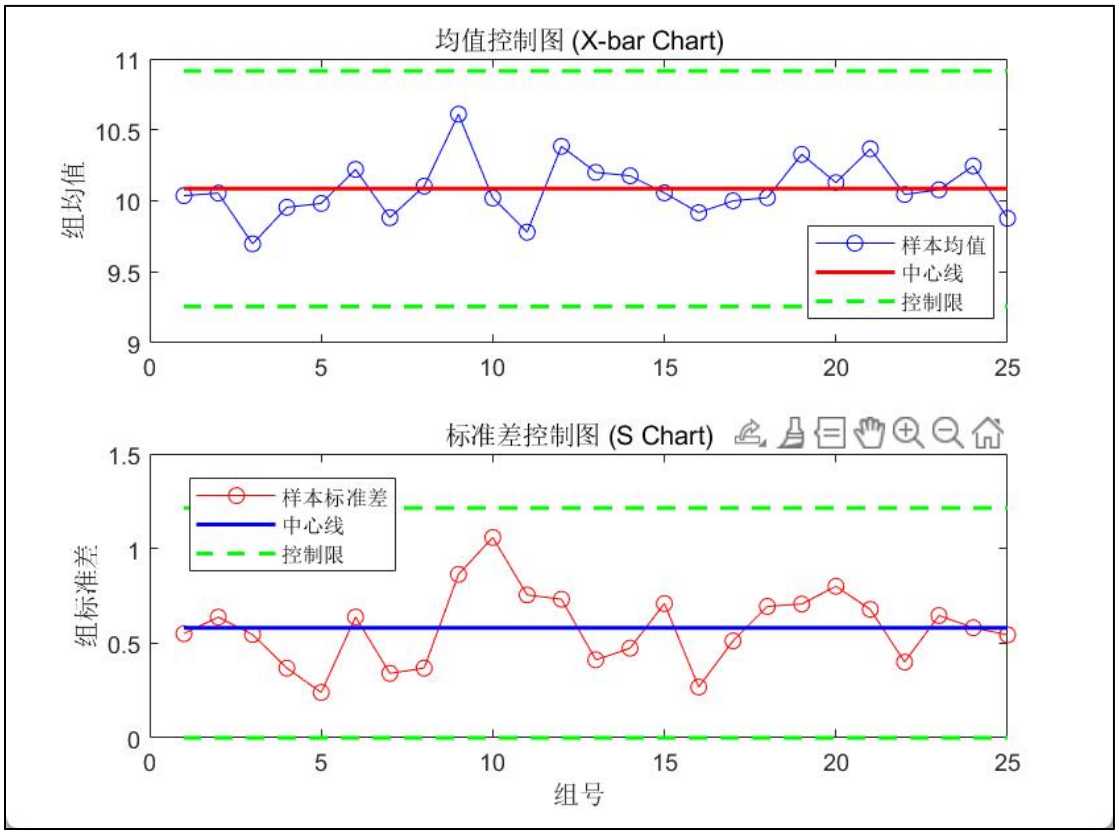
$$UCL = \hat{\mu} + A_s \cdot \hat{s}$$
$$CL = \hat{\mu}$$
$$LCL = \hat{\mu} - A_s \cdot \hat{s}$$

$$UCL = B_U \cdot \hat{s};\ CL = \hat{s};\ LCL = B_L \cdot \hat{s}$$

```
figure;
subplot(2,1,1);
plot(subgroup_means, 'bo-'); hold on;
plot([1 n_subgroups], [CL_X CL_X], 'r-', 'LineWidth', 1.5);
plot([1 n_subgroups], [UCL_X UCL_X], 'g--', 'LineWidth', 1.5);
plot([1 n_subgroups], [LCL_X LCL_X], 'g--', 'LineWidth', 1.5);
title('均值控制图 (X-bar Chart)'); ylabel('组均值');
legend('样本均值', '中心线', '控制限', 'Location', 'best');

subplot(2,1,2);
plot(subgroup_stds, 'ro-'); hold on;
plot([1 n_subgroups], [CL_S CL_S], 'b-', 'LineWidth', 1.5);
plot([1 n_subgroups], [UCL_S UCL_S], 'g--', 'LineWidth', 1.5);
plot([1 n_subgroups], [LCL_S LCL_S], 'g--', 'LineWidth', 1.5);
title('标准差控制图 (S Chart)');
xlabel('组号'); ylabel('组标准差');
legend('样本标准差', '中心线', '控制限', 'Location', 'best');
```

绘图得到结果如下，分别绘制均值和标准差控制图，将各子组数据以折线图形式展示：



3.7 SPC 八大规则验证

3.7.1 手动验证函数八大原则：

规则①：控制图上有一个点(对应某个批次数据)位于控制限以外；

```
rule1_X = find(subgroup_means > UCL_X | subgroup_means < LCL_X);
rule1_S = find(subgroup_stds > UCL_S | subgroup_stds < LCL_S);
if ~isempty(rule1_X), fprintf('规则1违反(均值图): 子组%d\n', rule1_X); end
if ~isempty(rule1_S), fprintf('规则1违反(标准差图): 子组%d\n', rule1_S); end
```

实现逻辑：比较每个数据点与 UCL/LCL，返回所有超出控制限的点的索引。

规则②：连续 9 个点落在中心线同一侧；

```
% 规则2: 连续9点在同侧
rule2_X = check_consecutive_side(subgroup_means, CL_X, 9);
rule2_S = check_consecutive_side(subgroup_stds, CL_S, 9);
print_violation('规则2', '连续9点同侧', rule2_X, rule2_S);
```

```
function violations = check_consecutive_side(data, CL, n)
    above = data > CL;
    below = data < CL;
    counts = zeros(size(data));
    for i = 2:length(data)
        if above(i) && above(i-1)
            counts(i) = counts(i-1) + 1;
        elseif below(i) && below(i-1)
            counts(i) = counts(i-1) + 1;
        else
            counts(i) = 1;
        end
    end
    violations = find(counts >= n);
end
```

实现逻辑：标记每个点位于 CL 的上侧或下侧，滑动窗口计数连续同侧点数。

规则③：连续 6 个点递增或者递减；

```
% 规则3: 连续6点递增/递减
rule3_X = check_monotonic_trend(subgroup_means, 6);
rule3_S = check_monotonic_trend(subgroup_stds, 6);
print_violation('规则3', '连续6点趋势', rule3_X, rule3_S);
```

```
function violations = check_monotonic_trend(data, n)
    trend_up = 0;
    trend_down = 0;
    violations = [];
    for i = 2:length(data)
        if data(i) > data(i-1)
            trend_up = trend_up + 1;
            trend_down = 0;
        elseif data(i) < data(i-1)
            trend_down = trend_down + 1;
            trend_up = 0;
        else
            trend_up = 0;
            trend_down = 0;
        end

        if trend_up >= n-1 || trend_down >= n-1
            violations = [violations, i];
        end
    end
end
```

实现原理：检测数据是否呈现连续上升/下降趋势，表明过程存在渐进性偏移。跟踪连续上升/下降次数，当连续次数≥n-1 时触发（n 点需 n-1 次变化）

规则④：连续 14 个点交替上下

```
% 规则4: 连续14点交替升降
rule4_X = check_alternating(subgroup_means, 14);
rule4_S = check_alternating(subgroup_stds, 14);
print_violation('规则4', '连续14点交替', rule4_X, rule4_S);
```

```
function violations = check_alternating(data, n)
    violations = [];
    for i = 1:length(data)-n+1
        segment = data(i:i+n-1);
        signs = sign(diff(segment));
        if all(signs(1:2:end) == 1) && all(signs(2:2:end) == -1) || ...
           all(signs(1:2:end) == -1) && all(signs(2:2:end) == 1)
            violations = [violations, i+n-1];
        end
    end
end
```

实现原理：检测数据点是否呈现周期性交替升降模式，表明存在系统性干扰因素。计算相邻点差值符号，检查是否呈现严格交替模式（+/−/+/−或−/+/−/+）

规则⑤：连续 3 个点中有 2 个点落在中心线同侧的 B 区以外；

```
rule5_X = check_zone_violation(subgroup_means, CL_X, sigma_est, 2, 3, 2);
rule5_S = check_zone_violation(subgroup_stds, CL_S, mean(subgroup_stds), 2, 3, 2);
print_violation('规则5', '3点中2点>2σ', rule5_X, rule5_S);
```

```
function violations = check_zone_violation(data, CL, sigma, k_sigma, window_size, threshold
    upper_zone = CL + k_sigma * sigma;
    lower_zone = CL - k_sigma * sigma;
    violations = [];
    for i = 1:length(data)-window_size+1
        segment = data(i:i+window_size-1);
        count = sum(segment > upper_zone | segment < lower_zone);
        if count >= threshold
            violations = [violations, i+window_size-1];
        end
    end
end
```

实现原理：检测数据点是否在滑动窗口内频繁超出指定 σ 区域，表明过程稳定性下降。计算上下边界（CL±kσ），滑动窗口统计超出边界的点数

规则⑥：连续 5 个点中有 4 个点落在中心线同侧的 C 区以外；

```
% 规则6: 连续5点中4点>1σ
rule6_X = check_zone_violation(subgroup_means, CL_X, sigma_est, 1, 5, 4);
rule6_S = check_zone_violation(subgroup_stds, CL_S, mean(subgroup_stds), 1, 5, 4);
print_violation('规则6', '5点中4点>1σ', rule6_X, rule6_S);
```

```
function violations = check_zone_violation(data, CL, sigma, k_sigma, window_size, threshold
    upper_zone = CL + k_sigma * sigma;
    lower_zone = CL - k_sigma * sigma;
    violations = [];
    for i = 1:length(data)-window_size+1
        segment = data(i:i+window_size-1);
        count = sum(segment > upper_zone | segment < lower_zone);
        if count >= threshold
            violations = [violations, i+window_size-1];
        end
    end
end
```

原理同上

规则⑦：连续 15 个点落在中心线两侧的 C 区；

```
% 规则7: 连续15点在1σ内
rule7_X = check_in_zone(subgroup_means, CL_X, sigma_est, 1, 15);
rule7_S = check_in_zone(subgroup_stds, CL_S, mean(subgroup_stds), 1, 15);
print_violation('规则7', '连续15点1σ内', rule7_X, rule7_S);
```

```
function violations = check_in_zone(data, CL, sigma, k_sigma, n)
    upper_bound = CL + k_sigma * sigma;
    lower_bound = CL - k_sigma * sigma;
    violations = [];
    counter = 0;
    for i = 1:length(data)
        if data(i) >= lower_bound && data(i) <= upper_bound
            counter = counter + 1;
            if counter >= n
                violations = [violations, i];
            end
        else
            counter = 0;
        end
    end
end
```

实现原理：检测数据过于集中靠近中心线，可能因数据分层或测量系统分辨率不足。检查数据是否在 CL±1σ范围内，连续计数器达到 n 时触发

规则⑧：连续 8 个点落在中心线两侧且无一点在 C 区以内。

```
% 规则8: 连续8点两侧且不在1σ内
rule8_X = check_both_sides(subgroup_means, CL_X, sigma_est, 1, 8);
rule8_S = check_both_sides(subgroup_stds, CL_S, mean(subgroup_stds), 1, 8);
print_violation('规则8', '连续8点两侧异常', rule8_X, rule8_S);
```

```
function violations = check_both_sides(data, CL, sigma, k_sigma, n)
    upper_zone = CL + k_sigma * sigma;
    lower_zone = CL - k_sigma * sigma;
    violations = [];
    counter = 0;
    prev_side = 0; % 0:未定义, 1:上侧, -1:下侧
    for i = 1:length(data)
        if data(i) > upper_zone || data(i) < lower_zone
            current_side = sign(data(i) - CL);
            if current_side ~= prev_side
                counter = counter + 1;
                prev_side = current_side;
            else
                counter = 0;
            end
            if counter >= n
                violations = [violations, i];
            end
        else
            counter = 0;
            prev_side = 0;
        end
    end
end
```

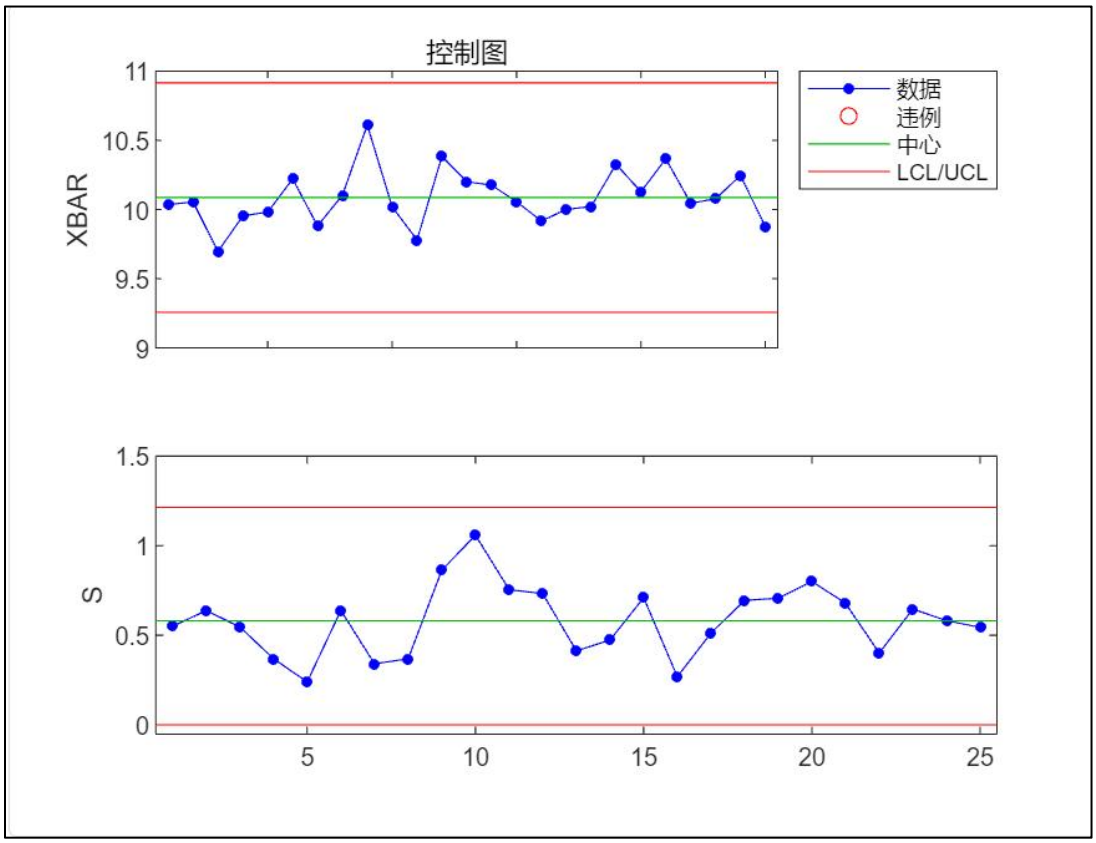
实现原理：检测数据在中心线两侧交替超出 1σ 区域，可能暗示多模态分布。跟踪交替超出上下区域的次数，忽略在 1σ 内的点

结论：手动验证数据不违反八大原则。无任何返回错误。

3.8.2 调用 matlab 库函数实现规则检查：

实现代码和结果如下：

```
load parts
[st,plotdata] = controlchart(data,'charttype',{'xbar' 's'});
```



其中第一张图为均值控制图，第二张图为标准差控制图。

通过观察图像，基于八大规则进行判断，数据确实处于统计受控状态。除此之外，样本均值也在 10.1 左右，控制状态良好；但实际工序能力指数 Cpk 仅有 0.807，说明工艺水平还有上升空间。

三. 总结与建议

当前生产过程虽稳定受控，但工序能力严重不足，核心问题为过程变异过大。

通过对生产数据的统计过程控制分析，生产过程整体处于统计受控状态。均值控制图和标准差控制图显示所有数据点均在控制限内，未触发 SPC 八大判异规则，表明当前生产过程未受到特殊原因导致的异常波动影响。数据通过 Lilliefors 检验（p=0.21）和卡方拟合优度检验（p=0.26），确认服从正态分布，满足参数检验的前提条件。单样本 t 检验结果显示，总体均值置信区间为 [9.914, 10.055]，包含目标值 μ=10（p=0.74），说明工艺中心控制良好，未发生显著偏移。

然而，过程能力评估显示，潜在能力指数 Cp=0.856，实际能力指数 Cpk=0.807，远低于国际标准要求的 Cpk≥1.5。这表明生产过程固有变异较大，无法稳定满足规格要求，存在较高的质量风险。当前 Cpk 值对应的理论不良率约为 6.5%，需立即采取措施减少波动，避免缺陷品流出。

如果这组数据来某自产品生产质量监控，为提升工艺能力，需引入实时监控系统动态调整工艺参数。其次，与供应商合作加强原材料检验，对硬度、密度等关键参数实施 SPC 监控，确保来料质量稳定。同时，降低环境因素对产品尺寸的影响。