本文主要介绍了非线性优化问题，通过凸性问题，非凸性问题的研究背景、内容及意义阐明了非线性优化问题的基于问题特性的算法选择方向。对于非线性优化问题的概述，本文从非线性优化问题的性质和算法原理两方面出发介绍，具体讲述了2个变量Rastrigin函数实验求解等内容。此外，通过举例具体的带有约束条件的非线性多变量函数的优化问题，编写Matlab程序求解，展示了关键函数的调用，实现了求该函数的最大值。

非线性优化 非凸函数 Matlab优化工具箱 全局优化算法

1. **绪论**

**1.1研究背景**

非线性优化问题作为运筹学与计算数学的核心领域，在工程优化、金融建模、机器学习等领域具有广泛应用价值。随着问题复杂度提升，非凸优化问题因其多局部最优解、高维搜索空间等特性成为研究难点。传统梯度类算法（如SQP、内点法）在凸优化问题中表现优异，但在非凸场景下易陷入局部最优。

MATLAB优化工具箱作为工业级数值计算平台，集成了从经典局部优化（fmincon, quadprog）到现代全局优化（particleswarm, simulannealbnd）的完整算法体系。

然而，现有研究多聚焦单一算法性能评估，缺乏：非凸函数特性与算法选择的系统映射关系，混合优化策略的协同机制分析，高维非凸问题的计算效率瓶颈突破。

本研究通过理论分析与实验验证相结合，构建非线性优化问题的"问题特性——算法选择"决策框架，为复杂工程优化提供方法论指导。

**1.2研究内容及其意义**

1. **非线性优化问题的重要概念**

**2.1定义**

目标函数或约束条件中包含非线性函数，称这种和规划问题为非线性规划问题。

数学形式如下

其中

f(x)：目标函数中决策变量x的系数值向量

x:决策变量构成的列向量

A：线性不等式约束的系数矩阵

b：线性不等式约束的右端常数向量

Aeq：线性等式约束的系数矩阵

beq：线性等式约束的右端常数向量

C(x)：非线性不等式约束的非线性向量函数

Ceq(x):非线性等式约束的非线性向量函数

lb：决策变量x的上界值向量

ub：决策变量x的下界值向量

**2.2非线性优化问题的分类**

按照约束条件和决策变量的类型，非线性规划问题可以分为几种类型： - 无约束非线性规划：没有任何约束条件，问题可以简化为寻找函数的极值点。 - 有约束非线性规划：有等式或不等式约束，这是最常见的非线性规划类型。 - 多目标非线性规划：有多个需要同时优化的目标函数，它们之间可能相互矛盾。 - 整数非线性规划：决策变量被限制为整数或离散值。

**2.2.1按照凸性分类**

非线性规划问题可以分为凸性问题，非凸性问题两种类型。

凸性问题是目标函数是凸函数，且可行域（约束集）是凸集的非线性规划问题，数学定义为

目标函数满足 。

约束集 满足：若 ，则 。

非凸性问题是目标函数或约束集中至少有一个不满足凸性。下面将重点讨论凸性和非凸性对非线性优化问题算法的选择和解的全局最优和局部最优的影响。

**2.3性质**

在非线性优化问题中，凸性是一个关键性质，它直接影响算法的选择和解的可靠性。

**2.3.1凸性问题任何局部最优解均为全局最优解**

反证法证明：假设存在局部最优解 和更优解 满足 。由凸性，对 ：

当 ，点 任意接近 ，与 的局部最优性矛盾。得出结论凸性问题任何局部最优解均为全局最优解。

**2.3.2非凸性问题局部最优不一定是全局最优：**

目标函数可能存在多个局部最优，一些传统的局部最优算法会停在最近的局部最优。后面以两个变量Rastrigin函数为具体实例可视化并分别用全局优化算法，局部最优算法求解Rastrigin函数最小值来说明

**2.4算法选择**

**2.4.1全局优化算法**

全局优化算法的基本原理:

全局优化问题是指在给定的可行域内寻找全局最优解的问题。与局部优化算法不同，全局优化算法的目标是找到全局最优解，而不是局部最优解。全局最优解是整个可行域中所有可能解中的最佳解，而局部最优解则是在某个局部区域内的最佳解。

全局优化算法可以根据其工作原理进行分类。一类是基于随机性原理的算法，例如模拟退火和遗传算法，它们通过概率性规则在全局搜索空间内进行有效的搜索。另一类是基于确定性原理的算法，如分枝定界法和穷举搜索，这些方法试图遍历整个搜索空间来找到全局最优解。

**2.4.2非线性优化问题的全局优化算法**

基于随机性原理的粒子群（PSO）、遗传算法（GA）、模拟退火（SA）避免局部最优，全局探索能力强。

**2.4.3非线性优化问题的局部优化算法**

梯度下降法（Gradient Descent）：通过负梯度方向迭代搜索最小值。

牛顿法（Newton's Method）：利用Hessian矩阵二阶信息，收敛速度快，但计算复杂度高。

序列二次规划（SQP, Sequential Quadratic Programming）：迭代求解二次规划子问题，MATLAB中 fmincon 的默认算法之一。

内点法（Interior Point Method）：通过障碍函数处理约束。

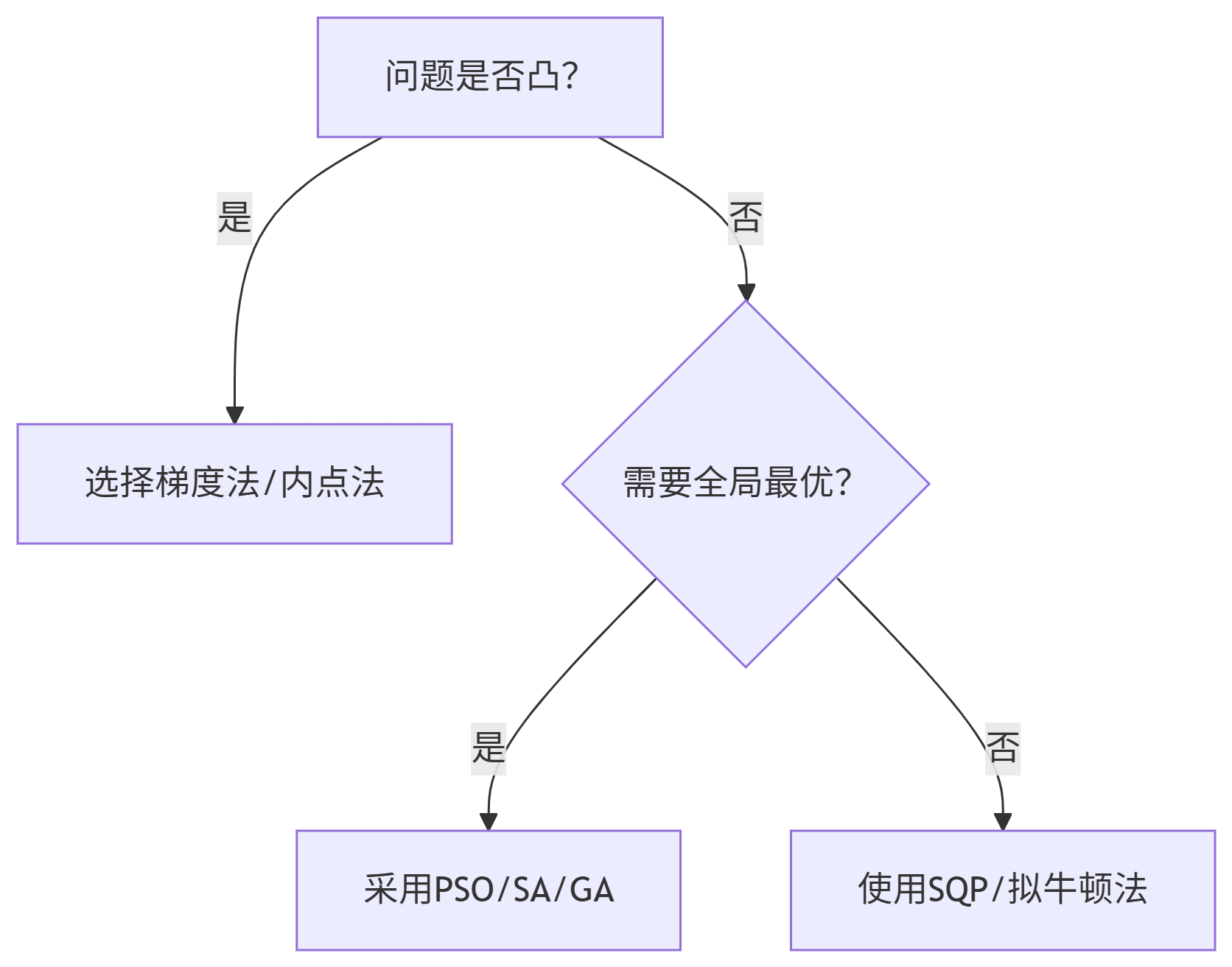
****

图2.4算法选择

**2.5Matlab优化工具箱概述**

**2.5.1MATLAB优化工具箱的基本功能**

MATLAB优化工具箱提供了一系列优化函数和算法，覆盖了从线性规划到复杂的非线性规划，从局部搜索算法到全局优化算法的广泛应用。这些算法不仅能处理有约束和无约束的问题，还能在必要时对问题的规模和复杂度进行适应。

**三、具体实例求解**

**3.1问题提出**

经典凸优化问题：二次规划（QP）

以两个变量Rastrigin函数为具体实例可视化并分别用全局优化算法，局部最优算法求解Rastrigin函数最小值。

**3.2经典凸优化问题示例**

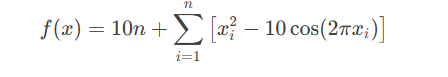
二次规划（QP）：

将二次规划转化为标准形式：

Matlab使用quadprog()函数采用二次规划方法得到结果x1=1.5714,x2=2.7857,最小值为-10.3214

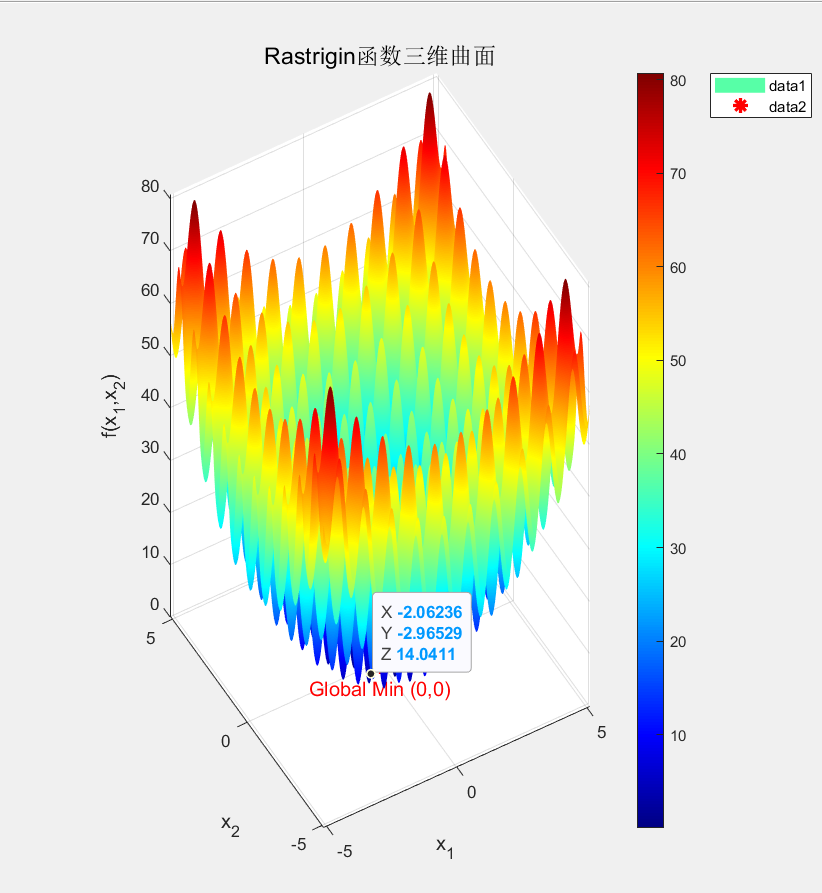
**3.3非凸函数实验求解**

Rastrigin函数是一个经典的测试函数，通常用于优化算法的性能评估。它既可以定义为单变量形式，也可以扩展到多变量形式。n个变量Rastrigin函数的函数表达式为

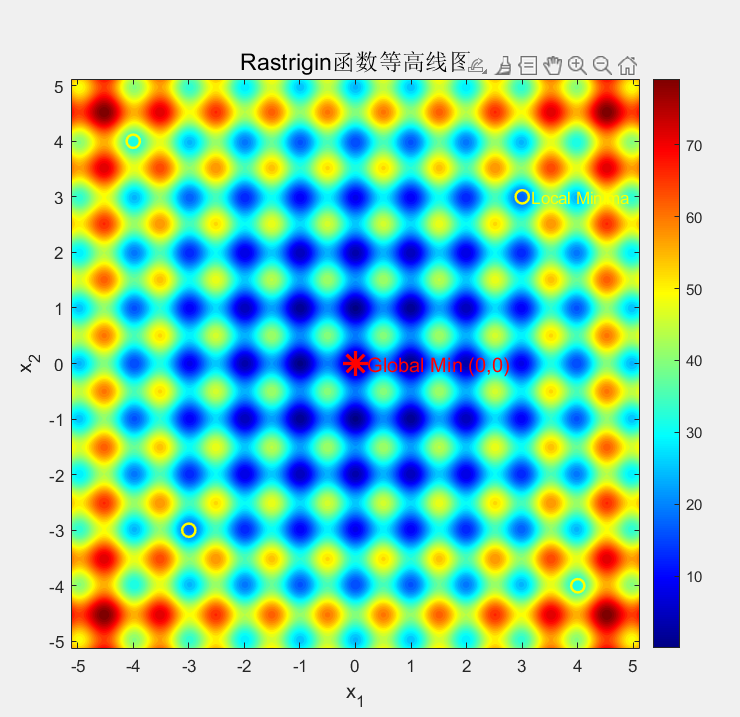


N=2时，2个变量Rastrigin函数可以让matlab把函数画出来

**3.3.1 matlab可视化**



**图3.3.1**Rastrigin函数 三维曲图



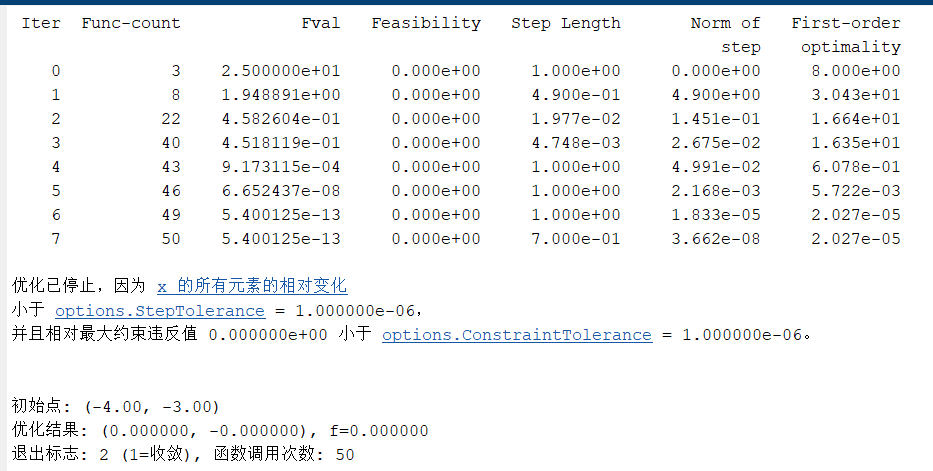
**图3.3.2**全局最优与测试点分布图

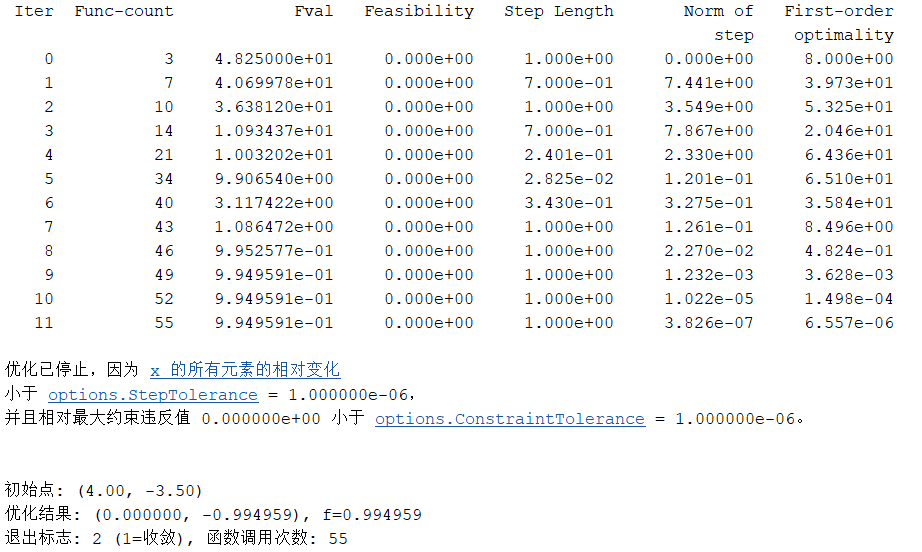
**3.3.2局部优化算法（fmincon）**

运行结果

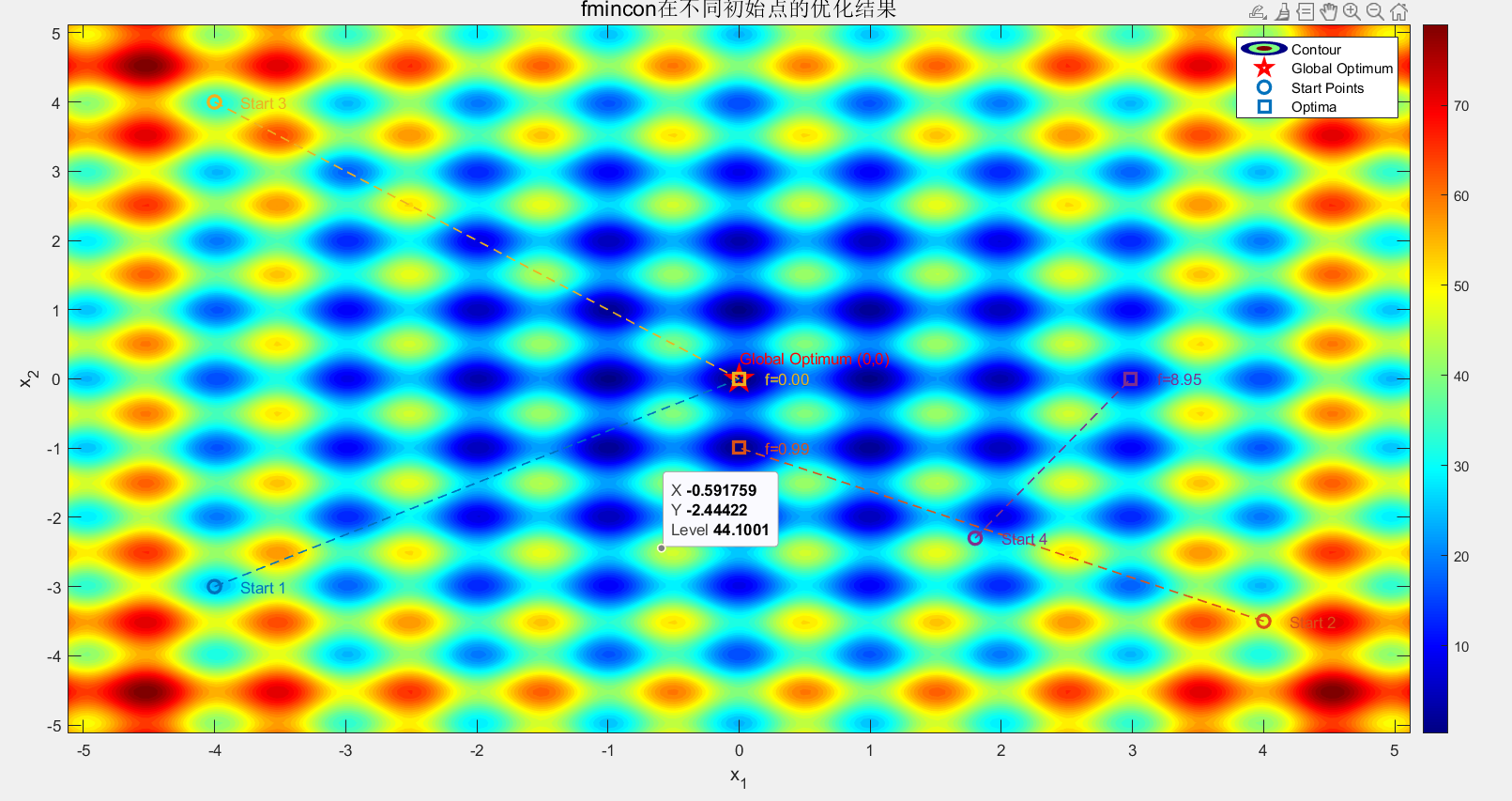
核心算法流程图：图加关键代码说明

结果说明：包括了对结果中所有输出值的解释，还有结束条件

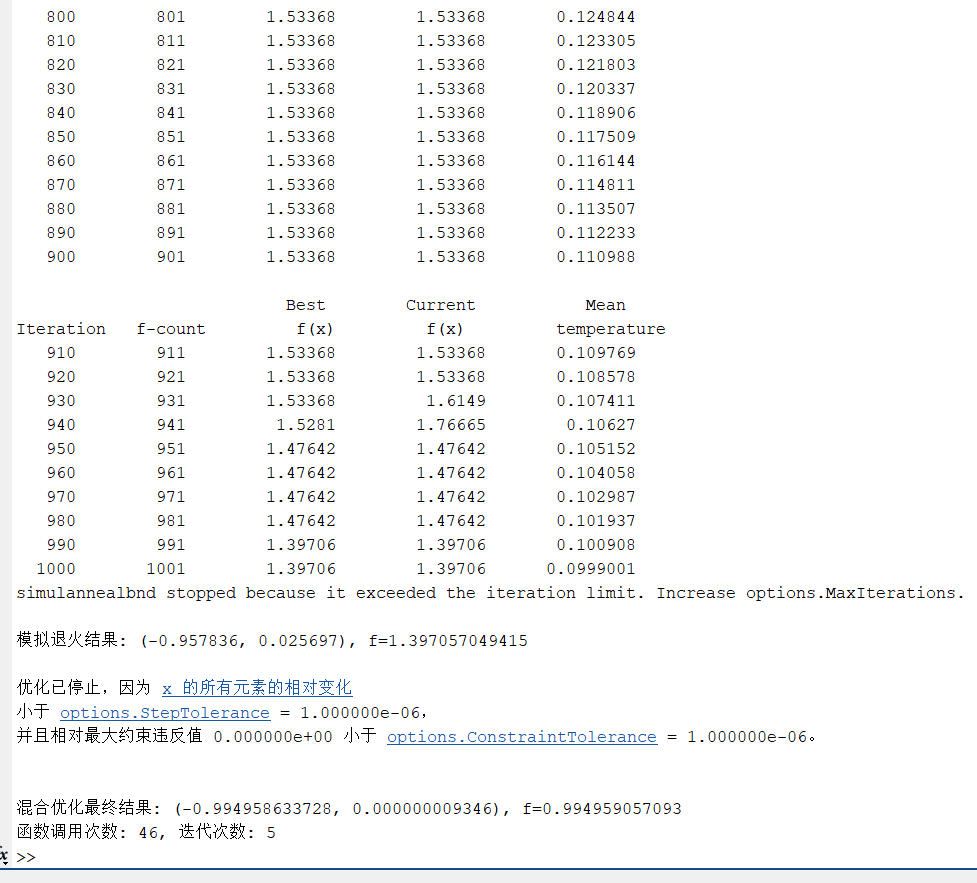




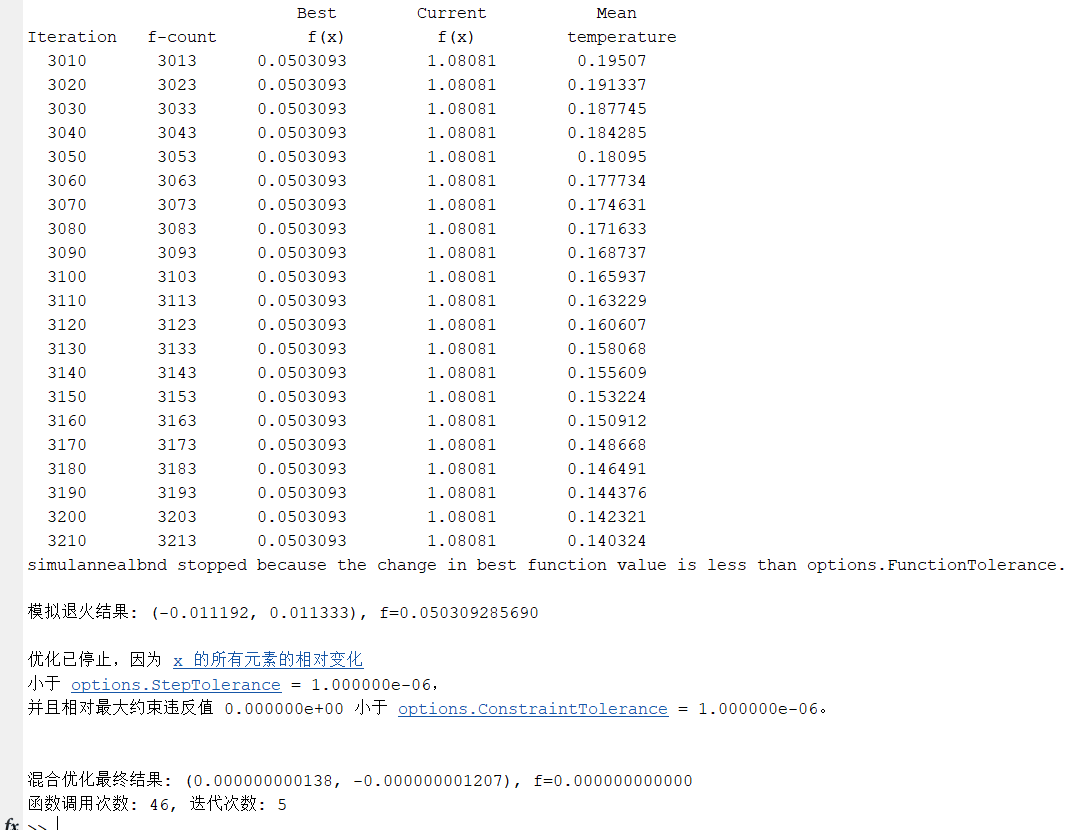




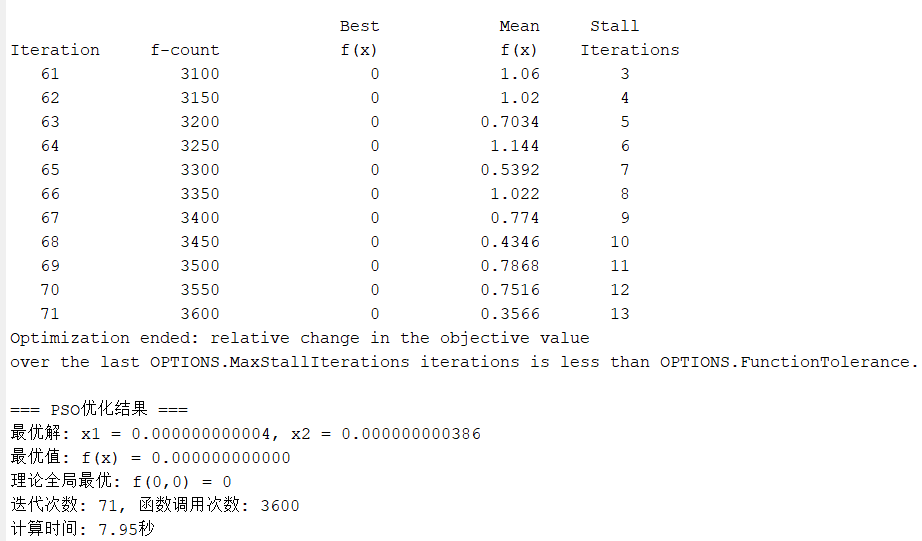
**3.2.3模拟退火算法**



**3.3.4混合算法策略**



**3.3.5粒子群算法**



**四、总结与展望**

**4.1总结**

本研究系统建立了非线性优化问题的"凸性判断-算法匹配-混合优化"方法论体系，通过MATLAB平台实现了理论到实践的完整转化。实验表明：针对复杂非凸问题，性能最好的是PSO算法，其次，相较于单一的模拟退火算法或单一的局部优化算法，混合合全局探索模拟退火算法与局部精炼（fmincon）的混合策略在求解精度和效率上有展现显著优势。

**4.2展望**

**五、附录**

% 两变量Rastrigin函数可视化

% 函数公式：f(x1,x2) = 20 + [x1^2 - 10\*cos(2\*pi\*x1)] + [x2^2 - 10\*cos(2\*pi\*x2)]

% 1. 基础设置

clear; clc; close all;

% 定义Rastrigin函数

rastrigin = @(x1, x2) 20 + (x1.^2 - 10\*cos(2\*pi\*x1)) + ...

(x2.^2 - 10\*cos(2\*pi\*x2));

% 设置绘图范围

x1 = linspace(-5.12, 5.12, 500);

x2 = linspace(-5.12, 5.12, 500);

[X1, X2] = meshgrid(x1, x2);

Z = rastrigin(X1, X2);

% 2. 三维曲面图

figure('Position', [100, 100, 1200, 500])

subplot(1,2,1)

% 绘制曲面

surf(X1, X2, Z, 'EdgeColor', 'none');

colormap(jet) % 使用jet色图增强对比度

colorbar

shading interp % 平滑着色

% 标记全局最优点

hold on

plot3(0, 0, 0, 'r\*', 'MarkerSize', 15, 'LineWidth', 2)

text(0, 0, -5, 'Global Min (0,0)', 'FontSize', 12, 'Color', 'r', 'HorizontalAlignment', 'center')

% 添加标签和标题

xlabel('x\_1', 'FontSize', 12)

ylabel('x\_2', 'FontSize', 12)

zlabel('f(x\_1,x\_2)', 'FontSize', 12)

title('Rastrigin函数三维曲面', 'FontSize', 14)

view(-30, 30) % 调整视角

axis tight

% 4. 优化路径可视化（对比局部优化和全局优化）

figure('Position', [100, 100, 1000, 800])

% 绘制等高线背景

contourf(X1, X2, Z, 50, 'LineColor', 'none');

colormap(parula)

colorbar

hold on

% 标记全局最优点

plot(0, 0, 'r\*', 'MarkerSize', 15, 'LineWidth', 2)

text(0, 0, ' Global Minimum', 'Color', 'r', 'FontSize', 12)

%% Rastrigin函数优化测试（演示局部最优陷阱）

% 使用fmincon求解器测试不同初始点对优化结果的影响

clear; clc; close all; % 清除工作区、命令窗口和图形窗口

%% 问题定义

% 定义二元Rastrigin函数（MATLAB匿名函数形式）

% 该函数在[0,0]处有全局最小值0，但包含许多局部极小点

rastrigin = @(x) 20 + (x(1)^2 - 10\*cos(2\*pi\*x(1))) + ...

(x(2)^2 - 10\*cos(2\*pi\*x(2)));

% 变量边界（标准Rastrigin函数测试范围）

lb = [-5.12, -5.12]; % 变量下限

ub = [5.12, 5.12]; % 变量上限

initial\_points = [-4, -3; % 靠近全局最优

4.0, -3.50; % 局部最优区域

-4.0, 4.0; % 边界局部最优

1.8, -2.3]; % 随机点

% 配置fmincon优化选项

options = optimoptions('fmincon', ...

'Algorithm', 'sqp', ... % 使用序列二次规划算法

'Display', 'iter-detailed', ... % 显示每次迭代详细信息

'MaxFunctionEvaluations', 1000, ... % 最大函数评估次数

'StepTolerance', 1e-6); % 迭代步长容差

%% 执行优化实验

% 预分配结果存储单元

results = cell(size(initial\_points,1), 5); % 列：[初始点, 最优解, 目标值, 退出标志, 耗时]

for i = 1:size(initial\_points,1)

x0 = initial\_points(i,:); % 当前测试的初始点

fprintf('\n=========== 测试 %d/%d ===========\n', i, size(initial\_points,1));

fprintf('初始点: [%.2f, %.2f]\n', x0(1), x0(2));

% 执行优化（使用内联函数定义确保可靠性）

tic; % 开始计时

[x\_opt, fval, exitflag, output] = fmincon(...

@(x) 20 + (x(1)^2 - 10\*cos(2\*pi\*x(1))) + (x(2)^2 - 10\*cos(2\*pi\*x(2))), ...

x0, [], [], [], [], lb, ub, [], options);

time\_cost = toc; % 结束计时

% 存储结果

results(i,:) = {x0, x\_opt, fval, exitflag, time\_cost};

% 打印详细结果

fprintf('--> 优化结果: [%.6f, %.6f]\n', x\_opt(1), x\_opt(2));

fprintf('--> 最优目标值: %.6f\n', fval);

fprintf('--> 退出标志: %d (1=收敛, 0=达到最大迭代, -1=其他)\n', exitflag);

fprintf('--> 耗时: %.4f秒, 函数调用次数: %d\n', time\_cost, output.funcCount);

end

混合算法

%% 混合优化：模拟退火(SA) + fmincon

clear; clc; close all;

% 定义Rastrigin函数

rastrigin = @(x) 20 + (x(1)^2 - 10\*cos(2\*pi\*x(1))) + ...

(x(2)^2 - 10\*cos(2\*pi\*x(2)));

% 变量边界

lb = [-5.12, -5.12];

ub = [5.12, 5.12];

%% 第一阶段：模拟退火全局探索

options\_sa = optimoptions('simulannealbnd', ...

'TemperatureFcn', @temperaturefast, ...

'AnnealingFcn', @annealingfast, ...

'ReannealInterval', 100, ...

'MaxIterations', 10000, ...

'Display', 'iter');

[x\_sa, fval\_sa, exitflag\_sa] = simulannealbnd(rastrigin, ...

rand(1,2).\*(ub-lb)+lb, lb, ub, options\_sa);

fprintf('\n模拟退火结果: (%.6f, %.6f), f=%.12f\n', x\_sa(1), x\_sa(2), fval\_sa);

%% 第二阶段：fmincon局部精炼

options\_fmincon = optimoptions('fmincon', ...

'Algorithm', 'sqp', ...

'Display', 'final-detailed', ...

'MaxFunctionEvaluations', 1000);

[x\_opt, fval, exitflag, output] = fmincon(rastrigin, x\_sa, ...

[], [], [], [], lb, ub, [], options\_fmincon);

fprintf('\n混合优化最终结果: (%.12f, %.12f), f=%.12f\n', ...

x\_opt(1), x\_opt(2), fval);

fprintf('函数调用次数: %d, 迭代次数: %d\n', ...

output.funcCount, output.iterations);

PSO算法

%% 粒子群优化(PSO)求解Rastrigin函数最小值

clear; clc; close all;

% 定义Rastrigin函数（全局最小值f(0,0)=0）

rastrigin = @(x) 20 + (x(1)^2 - 10\*cos(2\*pi\*x(1))) + ...

(x(2)^2 - 10\*cos(2\*pi\*x(2)));

% 变量边界（标准定义域）

lb = [-5.12, -5.12];

ub = [5.12, 5.12];

%% PSO参数设置

options = optimoptions('particleswarm', ...

'SwarmSize', 50, ... % 粒子数量

'InertiaRange', [0.1 1.1], ... % 惯性权重范围

'SelfAdjustmentWeight', 1.49, ... % 个体学习因子

'SocialAdjustmentWeight', 1.49, ... % 群体学习因子

'MaxIterations', 200, ... % 最大迭代次数

'FunctionTolerance', 1e-12, ... % 函数值容差

'Display', 'iter', ... % 显示迭代过程

'PlotFcn', @pswplotbestf, ... % 绘制收敛曲线

'UseParallel', true); % 启用并行计算

% 检查并行池

if isempty(gcp('nocreate'))

parpool('local'); % 启动默认并行池

end

%% 运行PSO优化

tic;

[x\_opt, fval, exitflag, output] = particleswarm(rastrigin, 2, lb, ub, options);

time\_cost = toc;

%% 结果输出

fprintf('\n=== PSO优化结果 ===\n');

fprintf('最优解: x1 = %.12f, x2 = %.12f\n', x\_opt(1), x\_opt(2));

fprintf('最优值: f(x) = %.12f\n', fval);

fprintf('理论全局最优: f(0,0) = 0\n');

fprintf('迭代次数: %d, 函数调用次数: %d\n', output.iterations, output.funccount);

fprintf('计算时间: %.2f秒\n', time\_cost);

%% 可视化分析

figure('Position', [100,100,1200,500])

% (1) 函数曲面与粒子轨迹

subplot(1,3,1)

[X,Y] = meshgrid(linspace(-5.12,5.12,200));

Z = arrayfun(@(x,y) rastrigin([x,y]), X, Y);

surf(X,Y,Z,'EdgeColor','none')

colormap(jet)

hold on

plot3(x\_opt(1), x\_opt(2), fval, 'r\*', 'MarkerSize',15,'LineWidth',2)

title('Rastrigin函数曲面','FontSize',12)

xlabel('x\_1'); ylabel('x\_2'); zlabel('f(x)')

view(-30,30)

% (2) 等高线图与粒子群分布

subplot(1,3,2)

contourf(X,Y,Z,50,'LineColor','none')

hold on

plot(x\_opt(1), x\_opt(2), 'r\*','MarkerSize',15,'LineWidth',2)

% 绘制最终粒子分布（需记录历史数据）

final\_positions = output.swarm;

plot(final\_positions(:,1), final\_positions(:,2), 'w.','MarkerSize',10)

title('粒子群最终分布','FontSize',12)

xlabel('x\_1'); ylabel('x\_2')

colorbar