赛题1：数值代数 - 矩阵分解算法设计与实现

一、赛题背景和目标

矩阵分解是数值代数中的常用技术，在求解线性方程组时，经常需要根据矩阵的不同特性选择相应的分解方法。由于大多数实际问题中矩阵属性一般不同，且分解结果具有可重用性，因此设计一个能够基于矩阵属性自动选择最优矩阵分解、并生成可重用分解结果的智能算法具有重要意义。

本次竞赛要求参赛队伍实现一个高效、稳健的矩阵分解算法框架，该框架应具备以下核心能力：

1. 对不同类型的矩阵分解实现统一接口，整合多种分解算法；
2. 识别矩阵特性（如对称性、正定性等），自动选择最适合的分解技术（如 LU 分解、Cholesky 分解、QR 分解、LDL 分解等）；
3. 生成可重复利用的高效分解结果；
4. 支持多种数据类型（单精度/双精度浮点数等）及相应的复数类型；
5. 兼容稠密矩阵和稀疏矩阵的处理。

二、题目设置

问题一：指定分解类型的 `decomposition` 函数

现在考虑如下 `decomposition` 函数的用法：

代码块

```
dA = decomposition(A, type)
```

通过直接指定 `type` 参数， `decomposition` 函数可计算输入矩阵 `A` 对应的分解，相应结果保存在变量 `dA` 中。请实现如下 `type` 参数对应的矩阵分解：

参数值	矩阵分解	说明
`qr`	计算矩阵 QR 分解（列排序），即 AP = QR，其中 P 是排列矩阵，Q 是正交矩阵，R 是上三角矩阵。	用于求解最小二乘问题，适用于任意形状的矩阵。
`cod`	计算矩阵全正交分解，即 A = QRZ*，其中 R 是上三角矩阵，Q、Z 均为列正交矩阵。	用于给出最小二乘问题的最小范数解，适用于任意形状的矩阵。

`lu`	计算列主元 LU 分解，即 PA = LU，其中 P 是排列矩阵，L 为单位下三角矩阵，U 为单位上三角矩阵。	需要 `A` 为方阵。
`ldl`	计算矩阵 LDL 分解，即 P*AP = LDL*，其中 P 是排列矩阵，L 为单位下三角矩阵，D 为对角矩阵。	需要 `A` 为对称矩阵。
`chol`	计算矩阵 Cholesky 分解，即 A = LL*，其中 L 为下三角矩阵。	需要 `A` 为对称正定矩阵。
`triangular`	仅仅需要判断 A 是否为上三角或下三角矩阵，写成分解形式为 A = T。	需要 `A` 为上三角或下三角矩阵，当不满足要求时报错，满足要求时也需自行设计相关返回值。
`diagonal`	仅仅需要判断 A 是否为对角矩阵，写成分解形式为 A = D。	需要 `A` 为对角矩阵，当不满足要求时报错，满足要求时也需自行设计相关返回值。

所实现的函数至少满足如下功能需求：

1. 支持输入 `A` 为稠密矩阵，支持单精度、双精度浮点数以及对应的复数。支持稀疏矩阵作为额外加分项；
2. 参赛队伍可以自行设计输出变量 `dA` 的数据结构，既可以用北太天元的结构体或者对象，也可以是 C/C++ 外部对象；
3. 所实现的函数应对异常输入（如数据类型、维度，分解运算失败等）进行合理检查和报错，避免程序闪退；
4. （可选）对于某些分解，可以使用 Name-Value 对来控制分解参数，语法为 `dA = decomposition(A, type, name, value, ...)`，例如 `CheckCondition` 选项可以控制在后续求解方程组时，对奇异矩阵给出警告信息。你可以根据实际情况任意增加此处的参数选项。

问题二：利用矩阵分解求解线性方程组和最小二乘问题

上一小问中实现了 `decomposition` 函数，现在还需要实现利用分解结果 `dA` 求解线性方程组的功能。请实现如下两个用法之一：

代码块

```
dA = decomposition(A, type); x = dA \ B;
% 或者实现一个函数 solve
dA = decomposition(A, type); x = solve(dA, B);
```

你可以自行设计 `dA` 中存储的必要信息，以供求解线性方程组时使用。

此外，请在北太天元中对所实现的 `decomposition` 以及方程求解机制进行功能和性能的测试：

1. 与北太天元内置函数（例如 `lu` `qr` 等）比较，验证结果的正确性；
2. 选取不同的矩阵规模（从 100 起至 10000）对矩阵函数分解 - 线性方程组/最小二乘问题求解框架进行效率测试，特别是需要多次求解不同的右端项 `B` 时（`A` 不变，`B` 在每次循环中发生变化且无法同时求解）：
   - 直接使用左除运算 `x = A \ B`；
   - 使用内置函数 + 手动求解，例如对于正定矩阵，可以使用 `R = chol(A); x = R \ (R' \ B)`；
   - 所实现的分解框架，即 `dA = decomposition(A, 'chol'); x = dA \ B`

   比较以上三种方式的运算效率；
3. 结合上述数值结果，谈一谈你对北太天元相关内置函数实现的建议。

问题三：自适应算法的 `decomposition` 函数

在软件实际使用过程中，用户可能无需关心具体使用何种分解（实际上指定分解类型也需要一定的专业知识），此时需要实现 `decomposition` 自适应，即根据输入矩阵 `A` 的特点，选取最优的矩阵分解算法。需要实现如下用法的函数：

代码块

```
dA = decomposition(A);
% 或指定 type 选项是 'auto'
dA = decomposition(A, 'auto');
```

与问题一的区别在于用户无需输入所执行的分解类型，函数对于不同矩阵选取不同的矩阵分解方法。因此需要完成如下任务：

1. 以高效求解线性方程组/最小二乘问题为目标，设计一个自适应算法，针对不同矩阵矩阵选择最优的分解；
2. 比较自适应算法和直接利用北太天元左除运算以及指定分解类型的 `decomposition` 算法的性能差距，并分析相关原因；
3. 谈谈你对此类自适应算法实现的理解与感受。

三、技术规范

规范项	说明
编程语言	C/C++/M 脚本
接口设计	统一接口函数声明，按照题目要求实现接口
错误处理	返回详细错误码和错误信息，接口调用失败时不可崩溃
平台兼容	支持 Windows 或 Linux
代码规范	遵守北太天元团队代码规范，结构清晰，注释完善

四、提交材料

1. 完整源代码及详细注释（若使用 C/C++ 语言实现，还需要提供编译脚本或工程文件）
2. 算法技术文档，包括但不限于如下内容：
   a. 算法设计（包含必要数学公式推导，算法流程图，伪代码等）
   b. 接口设计与使用说明
   c. 测试报告与性能对比
3. 测试用例代码与说明
4. PPT 与讲解视频

附录：参考资料

1. MATLAB decomposition 函数文档：
   https://www.mathworks.com/help/matlab/ref/decomposition.html
2. 北太天元插件开发文档：北太天元安装目录/SDK/doc/baltamatica-sdk-manual.pdf
3. 可能需要依赖的开源库
   - OpenBLAS： http://www.openmathlib.org/OpenBLAS/
   - LAPACKE： https://www.netlib.org/lapack/lapacke.html
   - SuiteSparse： https://people.engr.tamu.edu/davis/suitesparse.html
4. MATLAB mldivide 实现算法(供参考)：
   https://www.mathworks.com/help/matlab/ref/double.mldivide.html

赛题 2：优化算法 - 序列二次规划算法设计与实现

一、赛题背景和目标

赛题背景

序列二次规划（Sequential Quadratic Programming, SQP）是求解非线性约束优化问题的经典算法，被广泛应用于工程设计、控制系统、机器学习等领域。本次竞赛要求参赛队伍实现一个高效、稳定的 SQP 算法，处理如下形式的非线性规划问题：

$$\begin{aligned} \min_{x} \quad & f(x) \\ \text{s.t.} \quad & c_{eq}(x) = 0 \\ & c_{ineq}(x) \le 0 \\ & lb \le x \le ub \end{aligned}$$

其中：$x \in \mathbb{R}^n$，$c_{eq}(x) \in \mathbb{R}^{m_{eq}}$，$c_{ineq}(x) \in \mathbb{R}^{m_{ineq}}$

算法目标

参赛队伍需要实现类似 MATLAB fmincon('sqp')的功能，支持用户自定义的目标函数、约束函数及其梯度，并提供灵活的参数配置接口。

二、题目设置

问题一：基础 SQP 框架实现

实现处理等式约束优化问题的基础 SQP 算法。

具体要求：

1. **核心算法结构：**

- 实现标准 SQP 迭代框架
- 在每次迭代中构造并求解二次规划子问题：

$$\begin{aligned} \min_{d} \quad & \nabla f(x_k)^T d + \frac{1}{2} d^T B_k d \\ \text{s.t.} \quad & \nabla c_{eq}(x_k)^T d + c_{eq}(x_k) = 0 \end{aligned}$$

- 使用 BFGS 方法近似目标函数的 Hessian 矩阵 B_k

2. **二次规划子问题求解：**

- 实现 KKT 系统求解（可使用 LU 分解或 QR 分解）
- 处理线性代数数值稳定性问题

3. **线搜索方法：**

- 实现 Armijo 条件的回溯线搜索
- 使用 Merit 函数（如 L1 罚函数）处理约束

标准测试用例：

1. **等式约束 Rosenbrock 函数：**

$$\begin{aligned} \min_{x} \quad & f(x) = (1 - x_1)^2 + 100(x_2 - x_1^2)^2 \\ \text{s.t.} \quad & c(x) = x_1^2 + x_2^2 - 4 = 0 \end{aligned}$$

初始点：$x_0 = [1, 1]$

最优解：$x^* \approx [0.7864, 0.6177]$, $f^* \approx 0.0457$

2. **简单二次函数：**

$$\begin{aligned} \min_{x} \quad & f(x) = (x_1 - 2)^2 + (x_2 - 1)^2 \\ \text{s.t.} \quad & c(x) = x_1^2 + x_2^2 - 4 = 0 \end{aligned}$$

初始点：$x_0 = [1, 1]$

最优解：$x^* \approx [1.788854, 0.894427]$，$f^* \approx 0.055728$

3. **三变量工程问题：**

$$\begin{aligned} \min_{x} \quad & f(x) = x_1^2 + x_2^2 + x_3^2 \\ \text{s.t.} \quad & c(x) = x_1 + x_2 + x_3 - 3 = 0 \end{aligned}$$

初始点：$x_0 = [0, 0, 0]$

最优解：$x^* = [1, 1, 1]$，$f^* = 3$

核心算法实现参考：

1. **SQP 迭代框架：**

Step 1: 给定初始点 x_0 ，初始化 $B_0 = I$（单位矩阵）

Step 2: For k = 0, 1, 2, ...

a) 构造 QP 子问题：

$$\begin{aligned} \min_{d} \quad & \nabla f(x_k)^T d + \frac{1}{2} d^T B_k d \\ \text{s.t.} \quad & \nabla c_{eq}(x_k)^T d + c_{eq}(x_k) = 0 \end{aligned}$$

b) 求解 QP 得到搜索方向 d_k 和拉格朗日乘子 λ_k

c) 线搜索确定步长 α_k

d) 更新：$x_{k+1} = x_k + \alpha_k d_k$

e) BFGS 更新 Hessian 近似：B_{k+1}

f) 检查收敛条件

2. **KKT 系统求解：** QP 子问题的 KKT 条件为：

$$\begin{bmatrix} B_k & A_k^T \\ A_k & 0 \end{bmatrix} \begin{bmatrix} d_k \\ \lambda_k \end{bmatrix} = \begin{bmatrix} -\nabla f(x_k) \\ -c_{eq}(x_k) \end{bmatrix}$$

其中 $A_k = \nabla c_{eq}(x_k)^T$ 实现提示：

使用 LU 分解求解线性系统

处理矩阵奇异性：添加正则化项 $\delta = 10^{-8}$

检查解的合理性：$||d_k||$不应过大

3. **BFGS Hessian 更新：**

$$\begin{aligned} s_k &= x_{k+1} - x_k = \alpha_k d_k \\ y_k &= \nabla_x L(x_{k+1}, \lambda_{k+1}) - \nabla_x L(x_k, \lambda_k) \end{aligned}$$

if $s_k^T y_k > 10^{-8} \cdot \|s_k\| \cdot \|y_k\|$ （曲率条件）

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{s_k^T y_k}$$

else:

$$B_{k+1} = B_k$$ （跳过更新）

4. **Armijo 线搜索：**

Merit 函数：$\phi(\alpha) = f(x_k + \alpha d_k) + \mu \|c_{eq}(x_k + \alpha d_k)\|_1$

$\alpha = 1$, $\beta = 0.5$, $c_1 = 1e\text{-}4$

while $\phi(\alpha) \le \phi(0) + c_1 \alpha \nabla \phi(0)$:

$\alpha = \beta\alpha$

if $\alpha < 1e^{-10}$: break (避免无限循环)

问题二：不等式约束处理

扩展算法支持不等式约束，实现完整的 SQP 算法。

具体要求：

1. **积极集策略：**

- 实现积极集识别和更新机制
- 在迭代过程中动态调整活跃约束集合
- 处理约束的增加和删除

2. **二次规划子问题扩展：**

$$\begin{aligned} \min_{d} \quad & \nabla f(x_k)^T d + \frac{1}{2} d^T B_k d \\ \text{s.t.} \quad & \nabla c_{eq}(x_k)^T d + c_{eq}(x_k) = 0 \\ & \nabla c_{ineq\_active}(x_k)^T d + c_{ineq\_active}(x_k) = 0 \end{aligned}$$

1. **KKT 条件检验：**

- 实现完整的一阶最优性条件检验
- 正确处理拉格朗日乘子的符号条件
- 实现约束违反度（constraint violation）计算

2. **变量边界处理：**

- 将边界约束 lb ≤ x ≤ ub 转换为标准不等式约束
- 或者使用投影梯度方法直接处理

标准测试用例：

Hock-Schittkowski HS71：

$$\begin{aligned} \min_{x} \quad & f(x) = x_1 x_4 (x_1 + x_2 + x_3) + x_3 \\ \text{s.t.} \quad & g_1(x) = x_1 x_2 x_3 x_4 - 25 \geq 0 \\ & g_2(x) = x_1^2 + x_2^2 + x_3^2 + x_4^2 - 40 = 0 \\ & 1 \leq x_i \leq 5, \quad i = 1,2,3,4 \end{aligned}$$

初始点：$x_0 = [1, 5, 5, 1]$

最优解：$x^* \approx [1.0000, 4.7430, 3.8211, 1.3794]$, $f^* \approx 17.0140$

投资组合优化：

$$\begin{aligned} \min_{x} \quad & f(x) = \frac{1}{2} x^T Q x \\ \text{s.t.} \quad & \mu^T x - 0.15 \geq 0 \\ & \sum_{i=1}^{4} x_i = 1 \\ & x_i \geq 0, \quad i = 1, \ldots, 4 \end{aligned}$$

其中：Q 为 4×4 协方差矩阵，μ为收益向量

初始点：x_0 = [0.25, 0.25, 0.25, 0.25]

最优解：$x^* \approx [0.5783, 0.2865, 0.1352, 0.0000]$, $f^* \approx 0.0041$

核心算法实现参考：

1. **积极集策略：**

积极集识别：

$\mathcal{A}_k = \{i: c_i(x_k) \geq -\epsilon\} \cup \{\text{所有等式约束}\}$ $(\epsilon = 10^{-6})$

约束添加判据：

if 违反不等式约束 $c_i(x_k) > \varepsilon$: 添加约束 i 到积极集

约束删除判据：

if 拉格朗日乘子 $\lambda_i < 0$ 且 i 对应不等式约束: 从积极集删除约束 i

2. **扩展 QP 子问题：**

$$\begin{aligned} \min_{d} \quad & \nabla f(x_k)^T d + \frac{1}{2} d^T B_k d \\ \text{s.t.} \quad & \nabla c_{eq}(x_k)^T d + c_{eq}(x_k) = 0 \quad (\\ & \nabla c_{active}(x_k)^T d + c_{active}(x_k) = 0 \quad (\end{aligned}$$

KKT 系统：

$$\begin{bmatrix} B_k & A_{eq}^T & A_{act}^T \\ A_{eq} & 0 & 0 \\ A_{act} & 0 & 0 \end{bmatrix} \begin{bmatrix} d \\ \lambda_{eq} \\ \lambda_{act} \end{bmatrix} = \begin{bmatrix} -\nabla f(x_k) \\ -c_{eq}(x_k) \\ -c_{active}(x_k) \end{bmatrix}$$

3. **KKT 条件完整检验：**

梯度条件：

$$\|\nabla f(x) + \nabla c_{eq}(x)\lambda_{eq} + \nabla c_{ineq}(x)\lambda_{ineq}\| < 10^{-6}$$

原始可行性：

$$\|c_{eq}(x)\| < 10^{-6}, \quad \max(c_{ineq}(x)) < 10^{-6}$$

对偶可行性：

$$\lambda_{ineq} \geq 0$$

互补松弛：

$$\lambda_i c_i(x) < 10^{-6}, \quad \forall i$$

变量边界处理实现：

$$lb \leq x \leq ub \quad \Rightarrow \quad \begin{aligned} c_i(x) &= lb_i - x_i \leq 0 \\ c_j(x) &= x_j - ub_j \leq 0 \end{aligned}$$

问题三：算法优化与工程实用性

提升算法的数值稳定性和计算效率，提升工程实用性。

参考改进方向：

1. **数值稳定性改进：**

- 实现正则化技术处理病态问题
- 添加线性相关约束检测和处理
- 实现梯度和约束雅可比矩阵的数值微分验证

2. **自适应参数调整：**

- 实现自适应的罚参数更新策略
- 动态调整线搜索步长和收敛容差
- 添加迭代历史分析，检测振荡和慢收敛

3. **用户接口设计：**

- 设计类似 optimoptions 的参数配置系统
- 实现详细的迭代信息输出（类似 fmincon 的 Display 选项）
- 添加导数检验功能（DerivativeCheck）
- 实现多种停止准则选择

4. **性能优化：**

- 减少不必要的矩阵运算
- 基本的内存管理优化

标准测试用例：

1. **病态条件数问题：**

$$
\begin{aligned}
\min_{x} \quad & f(x) = 10000(x_2 - x_1^2)^2 + (1 - x_1)^2 \\
\text{s.t.} \quad & g_1(x) = (x_1 - 1)^3 - x_2 + 1 \le 0 \\
& g_2(x) = x_1 + x_2 - 2 \le 0 \\
& -1.5 \le x_1 \le 1.5 \\
& -0.5 \le x_2 \le 2.5
\end{aligned}
$$

数值特点：Hessian 条件数约 $\mathrm{cond}(H) \approx 4 \times 10^6$

初始点：x_0 = [0, 0]

最优解：最优解：x^* ≈ [1.0000, 1.0000], f^* = 0

三、技术规范

代码要求

编程语言：

M 代码或 C/C++，最终代码需要能够在北太天元上运行。（C/C++代码需要做成插件或 bex 文件放到北太天元上运行）

函数接口：

[x, fval, exitflag, output] = sqp_solver(fun, x0, A, b, Aeq, beq, lb, ub, nonlcon, options)

参数详细说明

输入参数

fun - 目标函数，待优化的目标函数，数据类型：函数句柄

x0 - 初始点，优化算法的起始点

A, b - 线性不等式约束，约束形式：A*x ⩽ b，如无线性不等式约束，设置为 A = [], b = []

Aeq, beq - **线性等式约束**，约束形式：Aeq*x = beq，如无线性等式约束，设置为 Aeq = [], beq = []

lb, ub - **变量边界约束**，约束形式：lb ≤ x ≤ ub，-Inf：表示无下界，+Inf：表示无上界，[]：表示所有变量无边界约束

nonlcon - **非线性约束函数**，[c, ceq] = nonlcon(x) 或 [c, ceq, gc, gceq] = nonlcon(x)，其中输出参数：

c：不等式约束向量，要求 c(x) ≤ 0

ceq：等式约束向量，要求 ceq(x) = 0

gc：不等式约束雅可比矩阵（可选）

gceq：等式约束雅可比矩阵（可选）

options - **算法选项**，控制算法行为的参数设置，数据类型为结构体

主要字段：

options.MaxIter：最大迭代次数，默认值：1000

options.TolFun：函数值收敛容差，默认值：1e-6

options.TolX：变量收敛容差，默认值：1e-6

options.TolCon：约束违约度容差，默认值：1e-6

options.Display：显示信息级别，可选值：'off', 'iter', 'final'，默认值：'final'

输出参数

x - **最优解**

fval - **最优函数值**，在最优解处的目标函数值

exitflag - **退出状态标志**，算法退出原因和求解状态

取值含义：

1：正常收敛到局部最优解

0：达到最大迭代次数

-1：算法被用户中断或输出函数停止

-2：无可行解

-3：问题无解

output - 详细输出信息，数据类型为结构体

output.iterations，实际执行的迭代次数

output.funcCount，目标函数的评估次数

output.constrviolation，最终解的最大约束违约度

output.stepsize，最后一次迭代的步长大小

output.firstorderopt，一阶最优性条件的违反程度（梯度范数）

output.message，详细的退出消息和状态描述

output.bestfeasible，当前解不可行时，记录最佳可行解信息。数据类型：结构体

字段：x（最佳可行点）, fval（对应函数值）

- **代码规范**：遵循北太天元编程规范，包含完整注释
- **开源库使用**：允许调用开源库，但需说明使用的库和原因

性能基准

- **精度要求**：KKT 条件约束违反度 < 1e-6
- **稳定性要求**：标准测试集全部测试成功

- **（可选）效率要求：**使用公开数据集，和 MATLAB 的 fmincon 对比并输出性能对比报告

四、提交材料

1. 完整源代码及详细注释（若使用 C/C++ 语言实现，还需要提供编译脚本或工程文件）
2. 算法技术文档，包括但不限于如下内容：
   a. 算法设计（包含必要数学公式推导，算法流程图，伪代码等）
   b. 接口设计与使用说明
   c. 测试报告，若选择接入公共数据集，还需输出相应的性能对比报告
3. 测试用例代码与说明
4. PPT 与讲解视频

算法实现建议

- 建议使用模块化设计：QP 求解器、线搜索、BFGS 更新分离
- 重视数值稳定性，添加必要的数值检查
- 提供清晰的错误处理和调试信息
- 考虑算法的可扩展性，为后续改进留出接口

参考资料

数值最优化方法 https://slzhang.com/optimization_methods.pdf

公共数据集：

https://github.com/ralna/CUTEst

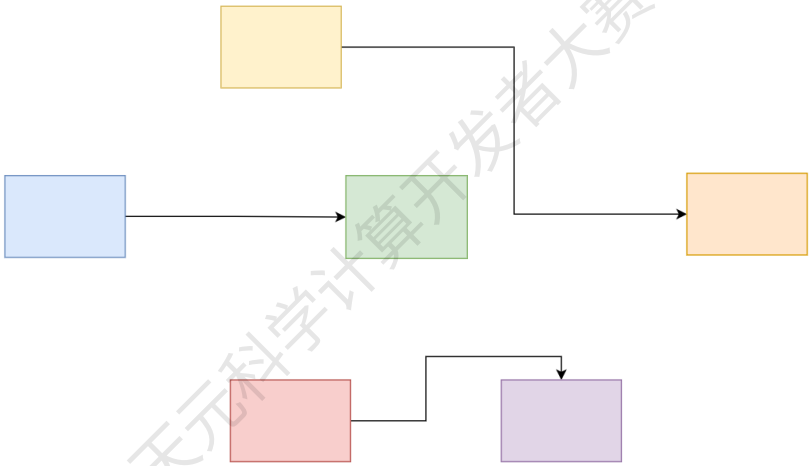
https://apmonitor.com/wiki/index.php/Apps/HockSchittkowski

赛题3：系统仿真 - 模块连线避障算法的设计与优化

一、赛题背景和目标

1.1 背景

在仿真软件中，用户通过线段连接各仿真模块，进而构建所需仿真模型。模块包含输入/输出端口，端口间连线为曼哈顿连线（需用水平和竖直的线段连接端口）。复杂模型通常包含大量模块，其连线关系也较为复杂，简洁的连线布局有助于提升建模效率和模型可读性。通常期望模块端口间的连线能够避开障碍物（障碍物特指画布上的其它模块或连线），并且与画布已存在的连线进行有序排布。如何规划模块连线路径满足上述需求是亟需解决的问题。模块连线样例如下图。



1.2 目标

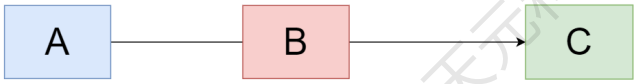
设计并实现一套曼哈顿路径避障算法，在包含矩形模块与已有连线的画布上，为给定起点与终点规划一条满足全部几何约束且性能最优的折线路径。

二、题目设置

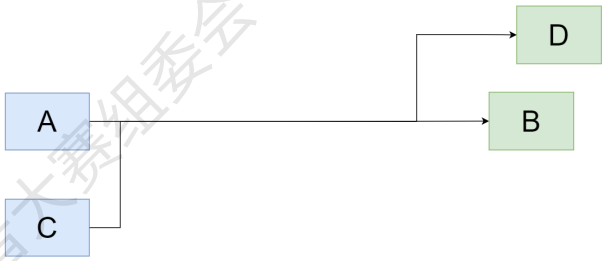
2.1 算法要求

参赛队伍需实现连线避障算法，满足下列条件：

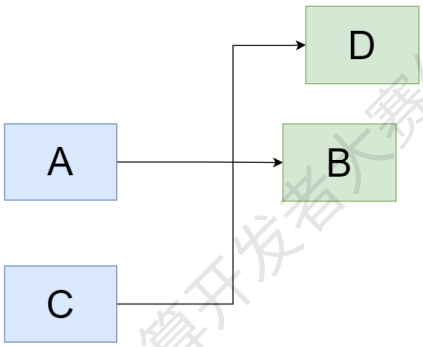
1. 避障处理：
    - 路径不能穿过任何模块（矩形），参考示意图如下，模块A连线到模块C，线段穿越B模块，这种情况不允许。



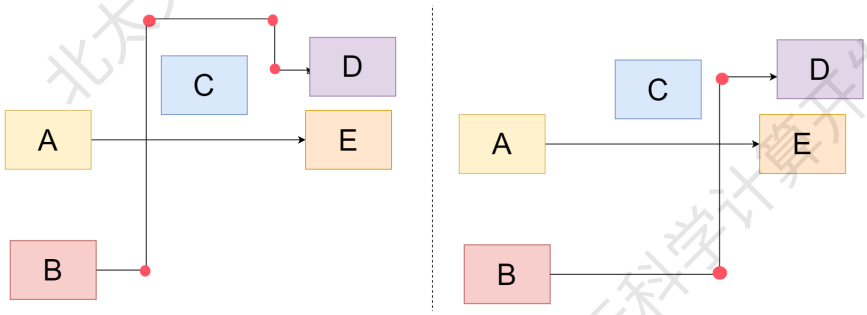
- 尽量不与已有连线重叠（连线宽度，默认1px）参考示意图如下，其中AB模块之间的连线与CD模块之间的连线进行了重叠。



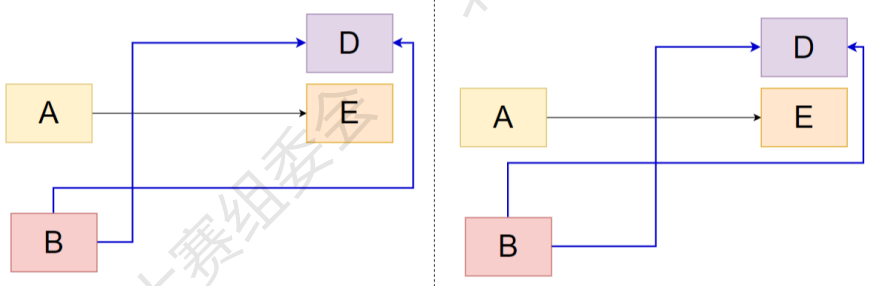
- 可以与已有连线交叉，参考示意图如下，其中AB模块之间的连线与CD模块之间的连线进行交叉。



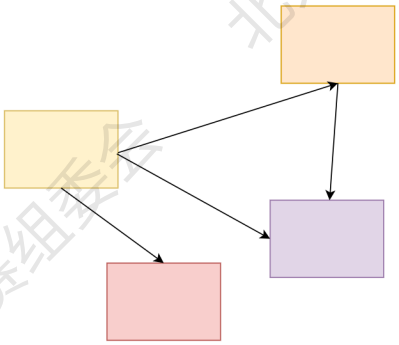
2. 路径优化：
   - 拐点（线转弯的位置，下图红色标记处）数量尽可能少，总路径长度尽量短，参考示意图如下，其中右边的连线（2个拐点）优于左边连线（4个拐点）效果。



   - 尽量穿过障碍物间的正中间（如两个模块之间区域的正中、两条平行线的正中、线与模块之间区域的正中），参考示意图如下，其中右边的连线优于左边连线效果。

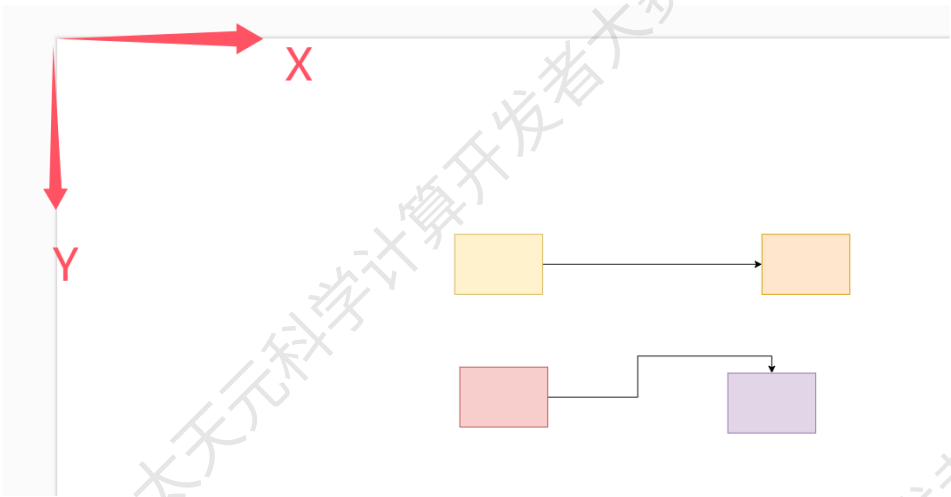


3. 约束条件：
    - 所有线段必须水平或竖直（无斜线），参考示意图如下，不允许出现图示连线。

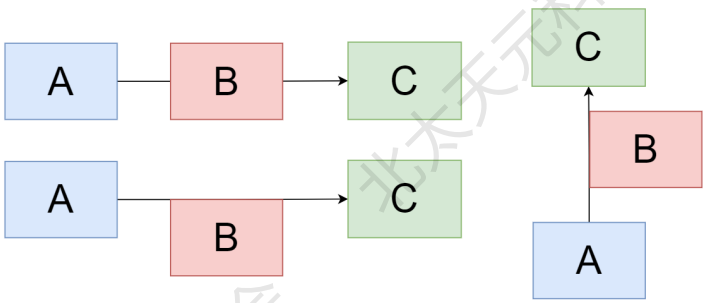


2.2 基础定义说明

1. 坐标系：
    - 画布左上角为 `(0,0)` ，向右为x轴正方向，向下为y轴正方向，参考示意图如下。



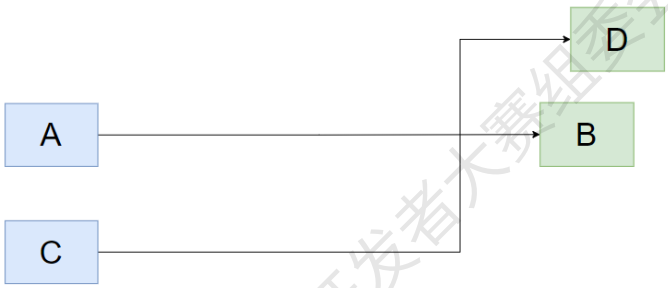
2. 障碍物定义：
    - 模块：矩形区域（含边界），路径不能进入内部或与边界重合，参考示意图如下，不允许出现图示情况。



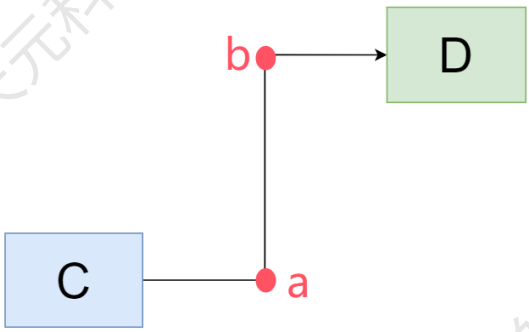
    - 已有连线：路径不能与其重叠。
3. 路径规则：

- 路径由起点（必须）、终点（必须）和拐点（非必须）组成，相邻点必须x或y坐标相同（水平/垂直线段），路径坐标示意如下表。

连线模块	路径
A→B	起点：（10，10） 终点：（25，10） 拐点：
C→D	起点：（10，20） 终点：（28，5） 拐点：（20，20）



- 拐点按顺序连接，且不能包含冗余点（如三点共线需合并），例如C→D连线坐标拐点分别为a：（5，10）、b：（5，5），不允许输出a和b之间的点，例如（5，8）。



2.3 代码接口（C/C++）

函数声明

代码块

```
1   void RouteNodes(CompAndCanvasInfo& wsg, std::vector<CustomPoint>& path);
```

参数说明：

- wsg：画布信息，包括画布宽高、画布上存在的模块位置、画布上存在的连线、起点坐标和中间坐标。数据类型为结构体。
- path：根据画布信息计算的起点到终点的连线路径。为路径拐点的有序列表（起点坐标、途径拐点、终点坐标）。

结构体说明

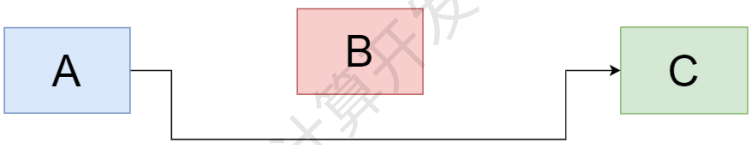
```

    typedef enum {
        north, south, east, west, non
    } direction;

//点
    struct CustomPoint {
        inline CustomPoint() : x(0), y(0) {}
        inline CustomPoint(float x, float y) : x(x), y(y) {}
        float x;
        float y;
        direction Direction{direction::non}; //起点或终点相对其模块的方向
        inline void setValue(float x_, float y_) {
            x = x_;
            y = y_;
        }
    };

    struct CompPartInfo {
        // 组件宽
        float width;
        // 组件高
        float height;
        // 组件所在位置，模块矩形的左上角坐标
        CustomPoint pos;
    };

    struct CompAndCanvasInfo {
        // 当前画布中所有的组件(模块障碍物)
        std::vector<CompPartInfo> compPartInfo;
        // 当前所在画布的宽
        float graphicsceneWidth;
        // 当前所在画布的高
        float graphicsceneHeight;

        // 真正开始连线的起始节点
        CustomPoint startNode;
        // 真正结束连线的终止节点
```

```
40          CustomPoint endNode;
41
42          // 开始连线的起始节点（计算用）
43          CustomPoint start;
44          // 结束连线的终止节点（计算用）
45          CustomPoint end;
46      };
47
48
49
```

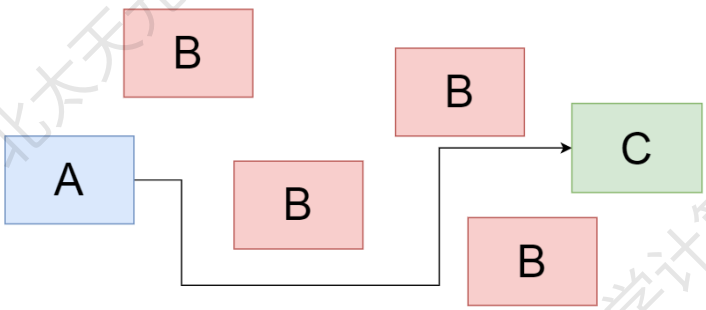
2.4 测试用例

用例1：无障碍直线，实例如图供参考。

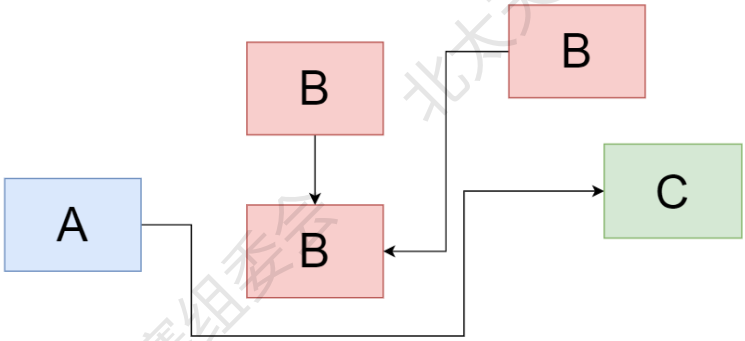
用例2：简单布局模块阻挡，实例如图供参考，设计A→C模块连线。



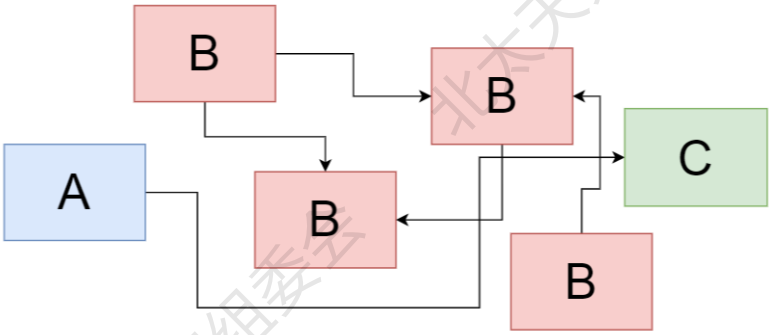
用例3：复杂布局模块阻挡，实例如图供参考，设计A→C模块连线。



用例4：简单布局混合障碍（模块+连线），实例如图供参考，设计A→C模块连线。



用例5：复杂布局混合障碍（模块+连线），实例如图供参考，设计A→C模块连线。

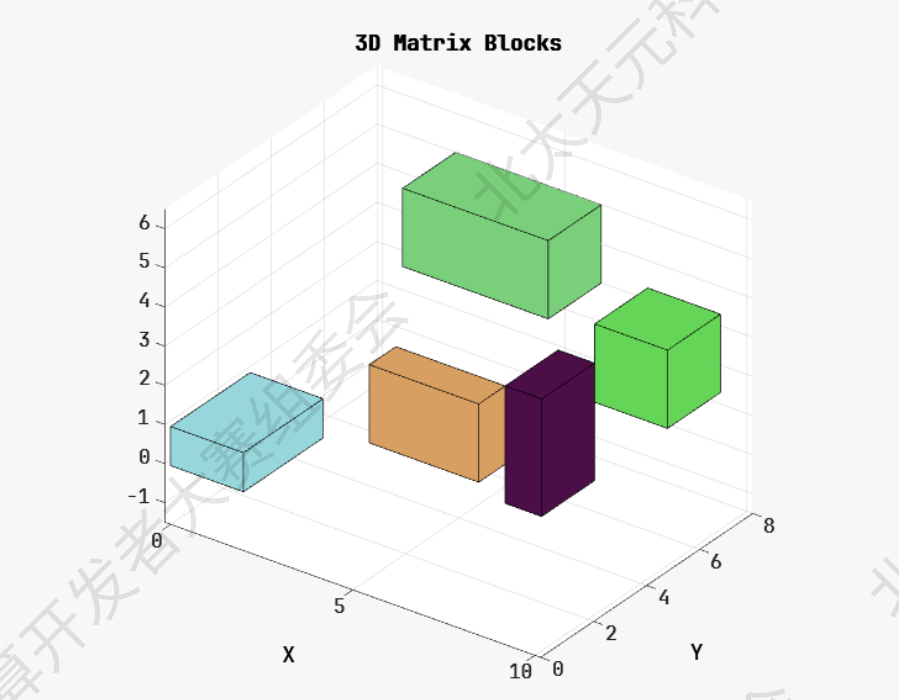


此外，可参考以上测试用例数据自行设计新数据。

三、技术规范

3.1 技术要求

1. 正确性（必须）：
   - 通过所有测试用例（路径避障、格式合规）。
2. 优化性（必须）：
   - 路径长度较短（与最优解比 ≤110%）。
   - 拐点少（用例5拐点 ≤4 个）。
   - 布局优化：与其它连线不重叠的，并且经过障碍物之间区域的正中。
3. 效率（必须）：
   - 时间上限：1秒（测试画布1080×960，障碍物 ≤50个）。
   - 算法复杂度。
4. 高阶应用
   - 考虑三维情况的连线避障，各模块增加高度属性情况下如何进行避障算法，如下图示意。



- 该部分内容可以参考二维连线避障接口增加相关参数和接口定义，非强制要求。

3.2 代码规范

规范项	说明
编程语言	C/C++（推荐）、M、Python
接口设计	统一接口函数名，参数与返回值应有明确约定
平台兼容	支持 Windows 或 Linux 平台
代码规范	代码规范，结构清晰，注释完善

四、提交材料

1. 完整源代码及详细注释（若使用 C/C++ 语言实现，还需要提供编译脚本、编译好的动态库）
2. 算法技术文档，包括但不限于如下内容：
   a. 算法设计（包含必要数学公式推导，算法流程图，伪代码等）
   b. 接口设计与使用说明
   c. 测试报告与性能对比
3. 测试用例代码与说明
4. PPT 与讲解视频

附录

参赛选手可将结果输出为 JSON，之后在北太天元中使用如下代码将结果可视化，也可自行实现可视化工具来展示避障效果。

可视化工具

代码块

```
function draw_json(jsonFile)
% draw_json  可视化连线避障 JSON
%
%   draw_json('demo.json');
%% 1. 读取 JSON
js  = jsondecode(fileread(jsonFile));
%% 2. 画布尺寸
W = js.Canvas.Width;
H = js.Canvas.Height;
%% 3. 准备图窗
clf; hold on; axis equal;
set(gca,'YDir','reverse');   % 左上角原点
xlim([0 W]); ylim([0 H]);
grid on; title(jsonFile);
%% 4. 绘制模块 Blocks
for k = 1:numel(js.Blocks)
    b = js.Blocks(k);
    hBlk = b.Height;
    wBlk = b.Width;

    % 解析 "x,y"
    pos = sscanf(b.Position, '%f,%f');
    x = pos(1); y = pos(2);

    rectangle('Position',[x y wBlk hBlk], ...
              'FaceColor',[0.8 0.8 1], ...
              'EdgeColor','b', 'LineWidth',1.2);
    text(x+wBlk/2, y+hBlk/2, sprintf('%s%d',b.BlockType), ...
        'HorizontalAlignment','center','FontSize',8);
end
%% 5. 绘制已有折线 Lines
if isfield(js,'Lines')
    for k = 1:numel(js.Lines)
        ptsStr = js.Lines(k).Line;
        if numel(ptsStr) < 2, continue; end

        % 将每个 "x,y" 转成坐标向量
        coords = cellfun(@(s)sscanf(s,'%f,%f'), ptsStr, 'UniformOutput',false);
        coords = cell2mat(coords')';   % 2×N
        for j = 1:length(coords)-1
            x = [coords(j,1), coords(j+1,1)];
```

```
            y = [coords(j,2), coords(j+1,2)];
            plot(x,y,'k-','LineWidth',1.5);
        end
    end
end
%% 6. 起点 / 终点
s = sscanf(js.startNode, '%f,%f');
e = sscanf(js.EndNode,   '%f,%f');
plot(s(1), s(2), 'ro', 'MarkerSize',8, 'MarkerFaceColor','r');
plot(e(1), e(2), 'gs', 'MarkerSize',8, 'MarkerFaceColor','g');
hold off;
xlabel X
ylabel Y
end
```

JSON 数据样例

代码块

```
{
    "Canvas": {
        "Height": 500,
        "Width": 500
    },
    "Blocks": [
        {
            "BlockType": "A",
            "Height": 20,
            "Width": 10,
            "Position": "87,110"
        },
        {
            "BlockType": "B",
            "Height": 20,
            "Width": 10,
            "Position": "379,40"
        }
    ],
    "Lines": [
        {
            "Line": [
                "100,200",
                "100,300",
                "200,300"
            ]
        },
```

```
        {
            "Line": [
                "150,300",
                "150,200"
            ]
        }
    ],
    "startNode": "100,100",
    "EndNode": "200,100"
}
```

json字段说明

字段名	类型	描述
Canvas	对象	画布设置，包含 `Width` 和 `Height` 属性。
Blocks	数组	矩形（障碍物），每个对象包含 `Position="x,y"` 。
Lines	数组	已有折线（障碍物），每个 `Line` 内是一串 `"x,y"` 点。
startNode	字符串	连线对应的起始端口坐标，格式为 `"x,y"` 。
EndNode	字符串	连线对应的终止端口坐标，格式为 `"x,y"` 。