运筹学 project1 测试报告

苏茂江 PB21000340

2023.12.18

1　问题介绍

用 C++ 编写单纯型法通用子程序，并求解线性规划标准问题。

2　算法原理

单纯形法的原理参考课程讲义《Lecture 4: 单纯形法和对偶理论》，主要包含大 M 法，最优判定和转轴运算。

3　编译环境及使用方法

3.1　编译环境

编译环境本人采用的是 Vscode，编译器为 g++（Mingw）。

而 project1.1 中文件.vscode 中 task.json 为 vscode 自动生成的编译文件，eigen-3.4.0 为 eigen 第三方库。

剩余的.cpp 和.h 文件是主要代码，.exe 文件是 vscode 生成的可执行文件，如果运行报错，请删除.exe 文件再试一次。

3.2　使用方法

project1.1.cpp 是主要测试文件，主要函数的定义和声明都在 function1.1.h 里面，randomtest.cpp 是用来随机生成测试案例，报告求解时间随着问题规模的变化图像，blandtest.cpp 用来测试 bland's rule。

本程序没有做模块 0，所以需要手动补充冗余变量，使得问题化为线性规划标准型

$$\left\{\min c^T x \mid Ax = b, x \geq 0\right\}$$

具体方法见各教科书。

由于数据结构是 eigen 库，于是对初始化进行说明。

```
79
80      // PPt example 4 (unbounded)
81      MatrixXd A(2, 2); // constraint matrix size=m*n
82      A << 1, -1,
83          -1, 1;
84      VectorXd b(2); // constraint vector size=m*1
85      b << 0, 0;
86      VectorXd c(2); // objective function  size=n*1
87      c << -1, 0;
88
89      linear_program_data p{A, b, c, 2};
90
91      simplex_method(p);
92      std::cout << "exitflag( 1 for solution found , -1 for unbounded, 0 for no feasible solution ) is:= " << p.exitflag << "\n";
93      if (p.exitflag == 1)
94      {
95          std::cout << "optimal value is:= " << p.v << "\n";
96          std::cout << "optimal solution is:= "
97                    << "\n";
98          for (int i = 0; i < p.n; i++)
99          {
100             std::cout << p.x_mini(i) << "\n";
101         }
102     }
103
104     return 0;
105 }
```

如图，上图是主要程序示例，在手动转换的到线性规划标准型问题的 A, b, c 之后，修改相应的 81-89 行即可，前面 80 行的注释是我测试的时候用的。

重点在于 linear_program_data p{A, b, c, 2}; 最后一个数字代表初始的 c 的行数，也就是初始变量个数（在化为标准型以后），因为后面大 M 法会修改 A，b 所以在这里对 n 进行说明。

3.3 各模块详解

3.3.1 模块 1

模块一由函数 void delete_duplicate_rows(linear_program_data &p) 实现，位置在 function1.1.h 的 25-77 行。

实现的方法是 PPT 中的方法 2

模块1：检查A是否满秩：方法2

使用QR分解

- A $\in \mathbf{R}^{\{m\times n\}}$，记B $= A^T$
- 有如下分解：B = QR，其中Q$\in \mathbf{R}^{\{n\times m\}}$ 是正交矩阵，R的对角元的绝对值从大到小排列。
- 如果 R的对角元都非零，则A满秩。
- 反之，R有对角元为零，说明可以删掉对应的Q的列,得到B′ 。
- 最后得到满秩矩阵 A′ $= (B')^T$
- 向量b删掉对应的行元素即可

在代码中，实现 QR 分解引用了 eigen 库来实现，其余步骤说明见 function1.1.h 的 25-77 行的注释。注意在代码中还检查了 A 和增广矩阵 Ab 的秩，初步判断问题解的情况。

3.3.2 模块 2

模块 2 中大 M 法实现初始化可行基解由函数 void ini_bigM(linear_program_data &p) 实现，位置在 function1.1.h 的 80-118 行。大 M 法的 M 设置为固定的 100000。

3.4 大 M 法

先求出初始可行基解后再进行第二阶段最优解的计算合并成一步作业的方法，称为**大 M 法**。

大 M 法考虑如下线性规划问题

$$\begin{aligned} \min \quad & \mathbf{c}^T\mathbf{x} + M\cdot\mathbf{1}^T\mathbf{y} \\ \text{s.t.} \quad & A\mathbf{x}+\mathbf{y}=\mathbf{b} \\ & \mathbf{x}\geq\mathbf{0},\mathbf{y}\geq\mathbf{0}. \end{aligned} \tag{4.7}$$

这里 $M > 0$ 为充分大的常数，我们也假设 $\mathbf{b} \geq \mathbf{0}$, $\mathbf{y}$ 和常数向量 $\mathbf{1}$ 与两阶段法中的一样。故，我们直接得到了一个可行基解

$$\begin{pmatrix}\mathbf{x}\\ \mathbf{y}\end{pmatrix} = \begin{pmatrix}0\\ \mathbf{b}\end{pmatrix}.$$

在代码中，新的问题会修改初始的 A 要注意，其余步骤说明见 function1.1.h 的 25-77 行的注释。

3.3.3 模块 3

模块 3 实现单纯形法的迭代，主要参考资料如下：

> 2. 最优判定
>
> 计算向量 $w = (B^T)^{-1}c_B$，对所有 $j \in N$ 计算 $z_j = w^T a_j$。若 $c_j \geq z_j$ 对一切 $j \in N$ 成立，则当前可行基解已是最优解，最优值为 $w^T b$，算法终止。否则，选择一个 $c_p - z_p < 0$ 的 $p \in N$ 进入下一步。
>
> 3. 转轴运算
>
> 计算向量 $y_p = B^{-1}a_p$。若 $y_p \leq 0$，则问题无有界解，算法终止。
>
> 否则，找出 r 使得 $\Delta = \frac{x_{B_r}}{y_{pr}}$ 是所有 $\frac{x_{B_i}}{y_{pi}}, y_{pi} > 0$ 中的最小值。
>
> 更新基变量：令 $x_p = \Delta$，$x_B = x_B - y_p\Delta$，这里 B 为原来的分量，更新后 x_{B_r} 由 Δ 定义成为 0。
>
> 更新 B, N：将 B 原本的第 r 列 a_{B_r} 用 a_p 替换，并将 p 从 N 中移入 B 中下标，B_r 从 B 中移入 N 中下标。

最优判定由函数 bool optimal_decision(linear_program_data &p) 实现，见 function1.1.h 的 120-162 行。
转轴运算由函数 void Axis_operation(linear_program_data &p) 实现，见 function1.1.h 的 164-207 行。
其余步骤说明见 function1.1.h 的 120-207 行的注释。

4 数据集说明

本次测试采用两个数据集，一个是助教发的《线性规划测试集》，另一个是老师 PPT《实验 1、2 要求》中 5 个测试案例，

5 测试结果

以下实验均在代码中用注释形式保留（如需测试记得修改初始 n）

5.1 《线性规划测试集》

测试结果一致，因不做要求故未在此展示。

5.2 《实验 1、2 要求》

5.2.1 有解

```
A = [1 2 2 1 0 0;
     2 1 2 0 1 0;
     2 2 1 0 0 1];
b = [20; 20; 20];
c = [-10; -12; -12; 0; 0; 0];
```

结果为

```
                                                                > & 'c:\Users\26557\.vscode\extensions\ms-vscode.cpptools-1.19
bugLauncher.exe' '--stdin=Microsoft-MIEngine-In-sqmmcajf.ejq' '--stdout=Microsoft-MIEngine-Out-3bwtwbwn.1mi' '--stderr=Microsoft-MIE
ft-MIEngine-Pid-blskar12.py3' '--dbgExe=C:\Program Files\mingw\mingw64\bin\gdb.exe' '--interpreter=mi'
exitflag( 1 for solution found , -1 for unbounded, 0 for no feasible solution ) is:= 1
optimal value is:= -136
optimal solution is:=
4
4
4
0
0
0
PS D:\STudy\the first semester of junior\Oprational research\homework>
```

5.2.2 有冗余秩

```
A = [1 2 3;
     2 4 6;
     1 1 1];
b = [6; 12; 3];
c = [1; 2; 3];
```

结果为

```
PS D:\STudy\the first semester of junior\Oprational research\homework>  & 'c:\Users\26557\.vscode\extensions\ms-vscode.cpptools-1.19.1-win32-x64\debug
bugLauncher.exe' '--stdin=Microsoft-MIEngine-In-h3gvvdws.odt' '--stdout=Microsoft-MIEngine-Out-xswlqmkr.cgc' '--stderr=Microsoft-MIEngine-Error-vbfmht
ft-MIEngine-Pid-l3q3mq4v.c05' '--dbgExe=C:\Program Files\mingw\mingw64\bin\gdb.exe' '--interpreter=mi'
1 2 3 1 0
1 1 1 0 1
exitflag( 1 for solution found , -1 for unbounded, 0 for no feasible solution ) is:= 1
optimal value is:= 6
optimal solution is:=
0
3
0
PS D:\STudy\the first semester of junior\Oprational research\homework>
```

注意到上面的矩阵是模块 1 和模块 2 工作过后的新矩阵 A，可见两模块均完成工作。

5.2.3 无可行域

```
A = [1 0 0;
     0 1 0
     0 0 1
     1 0 1];
b = [2; 2; 2; 2];
c = [1; 1; 1];
```

结果为

```
 & 'c:\Users\26557\.vscode\extensions\ms-vscode.cpptools-1.19.1-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEn
t=Microsoft-MIEngine-Out-5soet5bw.pl1' '--stderr=Microsoft-MIEngine-Error-t11mcj11.jgn' '--pid=Microsoft-MIEngine-Pid-35vh0irb.ftm' '--dbgExe=
\bin\gdb.exe' '--interpreter=mi'
1 0 0 1 0 0 0
0 1 0 0 1 0 0
0 0 1 0 0 1 0
1 0 1 0 0 0 1
exitflag( 1 for solution found , -1 for unbounded, 0 for no feasible solution ) is:= 0
PS D:\STudy\the first semester of junior\Oprational research\homework>
```

注意到上面的矩阵是模块 1 和模块 2 工作过后的新矩阵 A，可见两模块均完成工作。

5.2.4 无界

```
A = [1 -1 ;
    -1 1 ];
b = [0; 0;];
c = [-1; 0];
```

结果为

```
PS D:\STudy\the first semester of junior\Oprational research\homework>  & 'c:\Users\26557\.vscode\extensions\ms-vscode.cpptools-1.19.1-win32-x6
bugLauncher.exe' '--stdin=Microsoft-MIEngine-In-vojyovs0.m21' '--stdout=Microsoft-MIEngine-Out-c0eikrrk.fzv' '--stderr=Microsoft-MIEngine-Error
ft-MIEngine-Pid-31mnmsjn.n0a' '--dbgExe=C:\Program Files\mingw\mingw64\bin\gdb.exe' '--interpreter=mi'
 1 -1  1
exitflag( 1 for solution found , -1 for unbounded, 0 for no feasible solution ) is:= -1
PS D:\STudy\the first semester of junior\Oprational research\homework>
```

注意到上面的矩阵是模块 1 和模块 2 工作过后的新矩阵 A，可见两模块均完成工作。

5.2.5 Bland’s rule

Example 4.3 假设我们有如下初始单纯形表：

		x_1	x_2	x_3	x_4	x_5	x_6	x_7
	3	−3/4	20	−1/2	6	0	0	0
$x_5 =$	0	1/4*	−8	−1	9	1	0	0
$x_6 =$	0	1/2	−12	−1/2	3	0	1	0
$x_7 =$	1	0	0	1	0	0	0	1

采用如下规则选取出基入基下标：

1. 对于非基向量，选取下标 j 使得减少量 $\bar{c}_j$ 最小
2. 对于基向量，出基下标，选取所有满足条件中下标最小的。

跟踪单纯形法迭代的结果：输出为每次转轴运算的下标 B，注意实际下标应该加 1。

```
PS D:\STudy\the first semester of junior\Oprational research\homework>  & 'c:\Users\26557\.vscode\extensions\ms-vscode.cpptools-1.19.1-win32-x64\debu
bugLauncher.exe' '--stdin=Microsoft-MIEngine-In-mcjhq2lz.o4p' '--stdout=Microsoft-MIEngine-Out-resaflm2.vhz' '--stderr=Microsoft-MIEngine-Error-vnhr1
ft-MIEngine-Pid-1asv0vay.cu0' '--dbgExe=C:\Program Files\mingw\mingw64\bin\gdb.exe' '--interpreter=mi'
4 5 6
0 5 6
0 1 6
2 1 6
2 3 6
2 3 0
out of while loop
ce Lens.pdf
PS D:\STudy\the first semester of junior\Oprational research\homework>
```

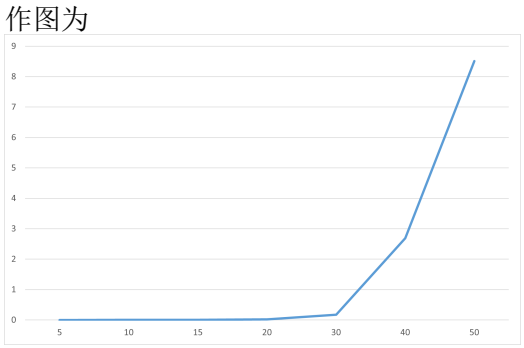
最后跳出了 while 循环，详细代码见

5.3 关于随机测试结果

在 randomtest.cpp 内测试随机矩阵结果，注意到 40 阶，50 阶因为随机生成很多为无解，所以选取 20 次成功成果进行输入。最后得到的平均时间和标准差。

```
ice Lens.pdf        time is 0.205
                    average time is (s)
                    0.00035
                     0.0012
                     0.0056
                     0.0164
                     0.1729
                     2.6875
                    8.50625
                    standard deviation is (s)
                    0.00047697
                        0.0004
                    0.00272764
                    0.00215407
                     0.0349312
                       3.77726
                       5.30964
                    PS D:\STudy\the first semester of junior\Oprational research\homework>
```

作图为



横坐标为 n，纵坐标为时间（s）。

6 分析与总结

单纯形法子程序所有测试集均测试通过，且其中各模块也验证正常工作，包括采用 Bland’s rule 避免退化解循环。在时间的复杂测试度上，发现随阶数 n 提高，耗时增长很快，标准差也比较大。

7 Code

代码见文件