重 庆 大 学

学 生 实 验 报 告

实验课程名称　　　　　数学模型

开课实验室　　　　　**D1128**

组员 **1** 姓 名　马梓恒　学 号 **20233124**

组员 **2** 姓 名　周宏仰　学 号 **20232647**

组员 **3** 姓 名　郑祺耀　学 号 **20230692**

开 课 时 间 **2024** 至 **2025** 学年第 一 学期

总 成 绩	

数 统 学 院 制

开课学院、实验室：数统学院，DS1128 实验时间 ： 2024 年 10 月 15 日

课程名称	数学实验	实验项目名　称	数学规划	实验项目类型				
				验证	演示	综合	设计	其他
指导教师	肖剑	成　绩				√		

题目 1

求解如下运输问题

$$\max \quad z=(2x_{11}+9x_{12}+10x_{13}+7x_{14})+(x_{21}+3x_{22}+4x_{23}+2x_{24}) \\ +(8x_{31}+4x_{32}+2x_{33}+5x_{34})$$

$$st.\begin{cases} x_{11}+x_{12}+x_{13}+x_{14}=9 \\ x_{21}+x_{22}+x_{23}+x_{24}=5 \\ x_{31}+x_{32}+x_{33}+x_{34}=7 \\ x_{11}+x_{21}+x_{31}=3 \\ x_{12}+x_{22}+x_{32}=8 \\ x_{13}+x_{23}+x_{33}=4 \\ x_{14}+x_{24}+x_{34}=6 \\ x_{ij}\geq 0,\ i=1,2,3;\ j=1,2,3,4 \end{cases}$$

程序

```
% 定义目标函数的系数 (将其负号取反以进行最小化)
f = -[2 9 10 7 1 3 4 2 8 4 2 5];

% 定义约束矩阵 A 和右边界 b
Aeq = [
    1 1 1 1 0 0 0 0 0 0 0 0;    % 源 1 的供应量
    0 0 0 0 1 1 1 1 0 0 0 0;    % 源 2 的供应量
    0 0 0 0 0 0 0 0 1 1 1 1;    % 源 3 的供应量
    1 0 0 0 1 0 0 0 1 0 0 0;    % 目的地 1 的需求量
    0 1 0 0 0 1 0 0 0 1 0 0;    % 目的地 2 的需求量
    0 0 1 0 0 0 1 0 0 0 1 0;    % 目的地 3 的需求量
    0 0 0 1 0 0 0 1 0 0 0 1;    % 目的地 4 的需求量
];

beq = [9; 5; 7; 3; 8; 4; 6];    % 供应量和需求量的右边界

% 非负性约束
lb = zeros(12,1);    % 决策变量不能为负

% 使用 linprog 进行求解
[x, fval] = linprog(f, [], [], Aeq, beq, lb);
```

```
% 输出结果
x = reshape(x, 4, 3)'  % 将结果按矩阵形式输出
fval = -fval  % 最大化目标函数值
```

结果

找到最优解。

```
x =

     0     5     4     0
     0     3     0     2
     3     0     0     4

fval =

   142
```

分析

这段 MATLAB 代码实现了一个线性规划问题，旨在最大化某种资源分配的效益。首先，通过将目标函数系数取负，转化为最小化问题的形式。然后，定义了一组等式约束，确保供应源的总供应量与各个目的地的需求量相匹配。具体而言，约束矩阵包含来自不同资源的供应量和到各个目的地的需求量，确保系统的平衡。此外，还设置了决策变量的非负性约束，以确保分配方案的实际可行性。通过调用 linprog 函数来求解该问题，最终结果以矩阵形式输出，便于分析每个源到每个目的地的具体分配量，同时将求解得到的目标函数值转回为最大化形式。这段代码有效地解决了资源分配中的优化问题，提供了一种高效的决策支持。

题目 2

求解约束非线性规划

$$\max z = \frac{0.201 x_1^4 x_2 x_3^2}{10^7}$$

$$s.t. \quad 675 - x_1^2 x_2 \ge 0$$

$$0.419 - \frac{x_1^2 x_3^2}{10^7} \ge 0$$

$$0 \le x_1 \le 36, 0 \le x_2 \le 5, 0 \le x_3 \le 125$$

程序

```
MODEL:
SETS:
   VARIABLES / x1, x2, x3 /;
ENDSETS

MAX = 0.201 * (x1^4) * (x2) * (x3^2) / (10^7);

675 - (x1^2) * (x2) >= 0;
0.419 - (x1^2) * (x3^2) / (10^7) >= 0;

x1 >= 0;
x1 <= 36;
x2 >= 0;
x2 <= 5;
x3 >= 0;
x3 <= 125;

END
```

结果

```
Local optimal solution found.
  Objective value:                              56.84794
  Infeasibilities:                            0.9933218E-06
  Extended solver steps:                               5
  Best multistart solution found at step:              1
  Total solver iterations:                           200
  Elapsed runtime seconds:                          0.32
```

分析

这段 LINGO 代码定义了一个优化模型，旨在最大化一个涉及决策变量 x_1、x_2 和 x_3 的复杂目标函数，形式为 $0.201\ x_1^4 x_2\ x_3^2/10^7$。模型设定了两个约束条件，确保 x_1 和 x_2 的组合不超过 675，同时 x_1 和 x_3 的组合也不超过 0.419。此外，还对每个变量设定了可行取值范围：x_1 的取值在 0 到 36 之间，x_2 在 0 到 5 之间，x_3 在 0 到 125 之间。通过运行这个模型，LINGO 将找出最优的 x_1、 x_2 和 x_3 值，以最大化目标函数，从而辅助决策在给定的约束条件下达到最佳效果。

题目 3

请自行查询某商业银行的整存整取年利率，填入下表：

一年期	二年期	三年期	五年期
1.35%	1.45%	1.75%	1.80%

现有 1 笔本金，准备 33 年后使用，若此期间利率不变，问应该采用怎样的存款方案？

程序

```
def calculate_best_option(principal, years, rates):
    # 初始化动态规划数组，存储每年结束时的最大收益和存款策略
    dp = [0] * (years + 1)
    strategy = [None] * (years + 1)

    # 遍历每一年的存款选择
    for year in range(1, years + 1):
        for term, rate in rates.items():
            if year >= term:
                # 计算当前的收益
                future_value = principal * (1 + rate / 100)
                if future_value > dp[year]:
                    dp[year] = future_value
                    # 记录存款策略
                    strategy[year] = (term, future_value)

    return dp, strategy

def get_saving_plan(principal, years, rates):
    # 计算最佳选项
    dp, strategy = calculate_best_option(principal, years, rates)

    # 输出最佳收益和存款步骤
    total_profit = dp[years]
    plan = []
    current_year = years
```

```python
    while current_year > 0:
        term, future_value = strategy[current_year]
        plan.append((current_year - term + 1, term))
        current_year -= term

    plan.reverse()  # 反转计划顺序，以便从开始到结束展示

    return total_profit, plan

# 利率设置
rates = {
    1: 1.35,
    2: 1.45,
    3: 1.75,
    5: 1.80
}

# 本金和投资年份
principal = 1  # 初始本金设为 1，用于计算总收益
years = 33

# 计算最佳存款方案
total_profit, saving_plan = get_saving_plan(principal, years, rates)

# 输出结果
print(f"总收益（本金 1 元，33 年后）：{total_profit:.2f}元")
print("存款策略（起始年, 存款年限）：")
for start_year, term in saving_plan:
    print(f"从第 {start_year} 年存款 {term} 年")
```

结果

总收益（本金 1 元，33 年后）：1.02 元

存款策略（起始年, 存款年限）：

从第 1 年存款 3 年

从第 4 年存款 5 年

从第 9 年存款 5 年

从第 14 年存款 5 年

从第 19 年存款 5 年

从第 24 年存款 5 年

从第 29 年存款 5 年

分析

这段代码实现了一个动态规划算法，用于计算基于不同存款期限和利率的最佳存款策略，以最大化投资收益。函数 `calculate_best_option` 初始化了两个数组：`dp` 用于存储每年结束时的最大收益，`strategy` 用于记录对应的存款策略。该函数遍历每个年份及其可选的存款期限和利率，计算并更新当前年份的最大收益。

另一个函数 `get_saving_plan` 利用 `calculate_best_option` 输出最终的总收益和具体的存款方案。代码中定义了一些固定的利率选项，并设定本金为 1 元，投资年限为 33 年。最后通过打印输出，展示了在指定条件下的总收益和详细的存款步骤。这种方法有效地将时间和收益结合，为用户提供了一种系统化的投资策略选择。

备注：

1、一门课程有多个实验项目的，应每一个实验项目一份，课程结束时将该课程所有实验项目内页与封面合并成一个电子文档上交。